

How Do You Handle Your Integration Strategy?

BEA WebLogic

DEVELOPER'S JOURNAL

JULY 2003 - Volume:2 Issue:7

weblogicdevelopersjournal.com

Web Services Edge ^{WEST} 2003

web services **EDGE**
conference & expo

SEPT. 30 - OCT. 2, 2003
Santa Clara, CA

PAGE 35

FROM THE EDITOR
Colonial Workshop
by Sean Rhody
page 5

DEVELOPER
RESOURCES
The dev2dev
Subscription Program
by Ryan O'Hara
page 46

ASK THE EXPERT
Application Performance
Best Practices
by Lewis Cirne
page 48

NEWS &
DEVELOPMENTS
page 50

RETAILERS PLEASE DISPLAY
UNTIL AUGUST 31, 2003

\$8.99US \$9.99CAN



SYS-CON
MEDIA

BEA WEBLOGIC APPLICATION CONSOLIDATION STRATEGIES

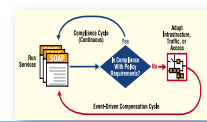
ALEX HEUBLEIN 6

INTEGRATION: Building Infrastructure-Aware Applications The power of shared infrastructure



12
Robert Shear

INTEGRATION: Integration: Not Just Aggregation Meeting the goals of true application integration



16
Laurence Moroney

MANAGEMENT: Enhancing Application Manageability with BEA's WebLogic Server/HP's OpenView PART 2
Management tools and their use of JMX



20
Justin Murray

WORKSHOP: Java Page Flow Web Application development made easy



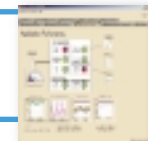
26
Doug Dew

FROM THE OFFICE OF THE CTO: Making Sense of Web Services Standards Current and upcoming standards improve interoperability



32
David Orchard

PRODUCT REVIEW: Wily Introscope 4.0 from Wily Technology Value across the development life cycle



36
Jason Snyder

REUSE: Enhanced Component-Based Development Component integration is a breeze

40
Chris Brooke

Informatica

www.informatica.com

BEA Systems

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

Wily Technology

www.wilytech.com



EDITORIAL ADVISORY BOARD

TOM ANGELUCCI, LEWIS CIRNE, STUART HALLOWAY,
KEVIN JONES, TYLER JEWELL, WAYNE LESLEY LUND,
SEAN RHODY, BLAKE STONE

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JASON WESTRA

EDITORIAL DIRECTOR
JEREMY GEELAN

MANAGING EDITOR
SEAN RHODY

PRODUCT REVIEW EDITOR
JASON SNYDER

EXECUTIVE EDITOR
GAIL SCHULTZ

EDITOR
NANCY VALENTINE

ASSOCIATE EDITORS
JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR
JENNIFER VAN WINCKEL

WRITERS IN THIS ISSUE

CHRIS BROOKE, LEWIS CIRNE, DOUG DEW,
ALEX HEUBLEIN, LAURENCE MORONEY,
JUSTIN MURRAY, RYAN O'HARA, DAVE ORCHARD,
SEAN RHODY, ROBERT SHEAR, JASON SNYDER,

SUBSCRIPTIONS

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department,
SUBSCRIPTION HOTLINE:

888-303-5282

Cover Price: \$8.99/Issue
Domestic: \$149/YR (12 Issues)

Canada/Mexico: \$169/YR

Overseas: \$179/YR
(U.S. Banks or Money Orders)

PRESIDENT AND CEO
FUAT A. KIRCAALI

VP, BUSINESS DEVELOPMENT
GRISHA DAVIDA

SENIOR VP, SALES & MARKETING
CARMEN GONZALEZ

PRODUCT CONSULTANT
JIM MORGAN

ART DIRECTOR
ALEX BOTERO

ASSOCIATE ART DIRECTORS
LOUIS F. CUFFARI • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR
TAMI BEATTY

VP, SALES & MARKETING
MILES SILVERMAN

ADVERTISING SALES DIRECTOR
ROBYN FORMA

DIRECTOR OF SALES AND MARKETING
MEGAN RING-MUSSA

ADVERTISING SALES MANAGER
ALISA CATALANO

ASSOCIATE SALES MANAGERS
CARRIE GEBERT • KRISTIN KUHNLE

PRESIDENT, SYS-CON EVENTS
GRISHA DAVIDA

CONFERENCE MANAGER
MICHAEL LYNCH

FINANCIAL ANALYST
JOAN LAROSE

ACCOUNTS RECEIVABLE
KERRI VON ACHEN

ACCOUNTS PAYABLE
BETTY WHITE

VP, INFORMATION SYSTEMS
ROBERT DIAMOND

WEB DESIGNERS
STEPHEN KILMURRAY • CHRISTOPHER CROCE

ONLINE EDITOR
LIN GOETZ

CIRCULATION SERVICE COORDINATORS
NIKI PANAGOPOULOS • SHELIA DICKERSON

JDJ STORE MANAGER
RACHEL MCGOURAN

EDITORIAL OFFICES

SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637

SUBSCRIBE@SYS-CON.COM
BEA WebLogic Developer's Journal (ISSN# 1535-9581)

is published monthly (12 times a year)

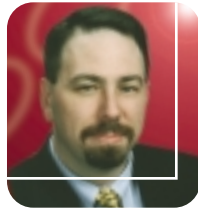
Postmaster: Send Address Changes to
BEA WEBLOGIC DEVELOPER'S JOURNAL,
SYS-CON Publications, Inc.

135 Chestnut Ridge Road, Montvale, NJ 07645

WORLDWIDE NEWSSTAND DISTRIBUTION
Curtis Circulation Company, New Milford, NJ



© COPYRIGHT 2003 BY SYS-CON PUBLICATIONS, INC. ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM, WITHOUT THE WRITTEN PERMISSION OF SYS-CON PUBLICATIONS, INC. CONTACT REPRINTS, PERMISSIONS, OR ADVERTISING COORDINATOR: SYS-CON PUBLICATIONS, INC., RESERVES THE RIGHT TO REVISE, REPHRASE, AND AUTHORIZE THE READER TO USE THE ARTICLES SUBMITTED FOR PUBLICATION. THE BEA AND PRODUCT NAMES USED ON THESE PAGES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES. SYS-CON PUBLICATIONS, INC., IS NOT AFFILIATED WITH THE COMPANIES OR PRODUCTS COVERED IN WEBLOGIC DEVELOPER'S JOURNAL.



BY SEAN RHODY
MANAGING EDITOR

Colonial Workshop

Have you ever visited one of those theme parks that depict life as it was in colonial times? I'm always fascinated by the blacksmith and other craftsmen who show you just how hard it actually was to produce items that we take for granted, and how many modern inventions, such as electricity, they did without. They look happy, but I know it's just a show for the tourists. No one wants to use archaic methods to create things.

In the BEA world, I'm reminded of the differences between modern convenience and these theme parks whenever I work with BEA WebLogic Workshop. I've had some time in front of the 2.0 beta, and I'm impressed with their direction.

I worked with the first version, which was primarily geared to making it easy to create Web services, and was not particularly useful for most other aspects of working with the WebLogic Platform. Still, I liked the easy way that you could assemble different functionality into controls (like a database query, or an EJB), then use them in a drag-and-drop IDE to create Web services (you did read my bio, didn't you? I'm big on Web services.). From an ease-of-use standpoint, it was light years ahead of most of the other Java tools. Creating a Web service took a few minutes, and you even got a free client out of it. Of course, it wasn't a very useful client, but at least you could test if your service worked.

But there's more to life than Web services, and Workshop 1.0 didn't do much for other things in WebLogic, like Portal or Integration.

A marketing fellow I once knew described programmers as falling into three different segments. First there were the hard-core, VI-using, command-line gurus who do the really intense programming – the guys who write Linux kernel drivers for fun. Then there were the corporate

programmers, who are more concerned with getting the next release of the corporate accounting package out on time. Then there were business analysts. Don't ask me how they got into the programming category; I never got a straight answer on that one either.

The point this guy had was that each segment needed different tools because they approached fundamentally different problems. The hardcore programmer attacks highly detailed, complex technical problems. For that, and because they build the fundamental building blocks of the organization, they need down-to-the-metal access, much like C++.

The corporate programmers have a different problem – getting a large application of relatively simple tasks (yes, I know, nothing is simple, but some things are more complex than others) put together under tight deadlines. They need less flexibility, and more speed and effort saving devices.

Don't ask what the business analysts need.

Workshop 2.0 is for the corporate programmer. You can do Web services, as well as develop portlets and work with BEA WebLogic Integration in one drag-and-drop interface that treats complex objects (such as Web services or JCA) as simple controls that can be wired together.

Not that it's perfect mind you – you can't do thick-client development at all, and it's still got some rough edges on what you can do – but it's certainly a much better environment than using text pad. This is a corporate developers' tool and will compete directly with .NET Studio.

Enjoy BEA WebLogic Workshop. It's a refreshing change from more hard-core programming environments, even if you ultimately have to go back to them for certain tasks. Now, I think my WebLogic Server just threw a horseshoe. Got to go find a code blacksmith...

AUTHOR BIO...

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

CONTACT: sean@sys-con.com

BEA WebLogic Application Consolidation

HOW TO TURN THE PROMISE



BY ALEX HEUBLEIN

AUTHOR BIO...

Alex Heublein is the managing principal of the Next-Generation Enterprise Technologies team within HP Consulting and Integration's Technology Leadership Group. Alex's team works to evaluate new and emerging enterprise technologies in order to help assess their impact on the strategic direction of HPC&I's customers. Prior to joining HP, Alex was the CTO of an IT consulting firm focused on B2B integration technologies and was the managing principal of the Application Server & Middleware group for IBM Global Services in North America. He holds a Bachelor of Science degree in computer science from the Georgia Institute of Technology.

CONTACT...

alex.heublein@hp.com

Given the current global economic downturn, it is certainly no surprise that large organizations are putting cost-cutting measures at the top of their priority lists.

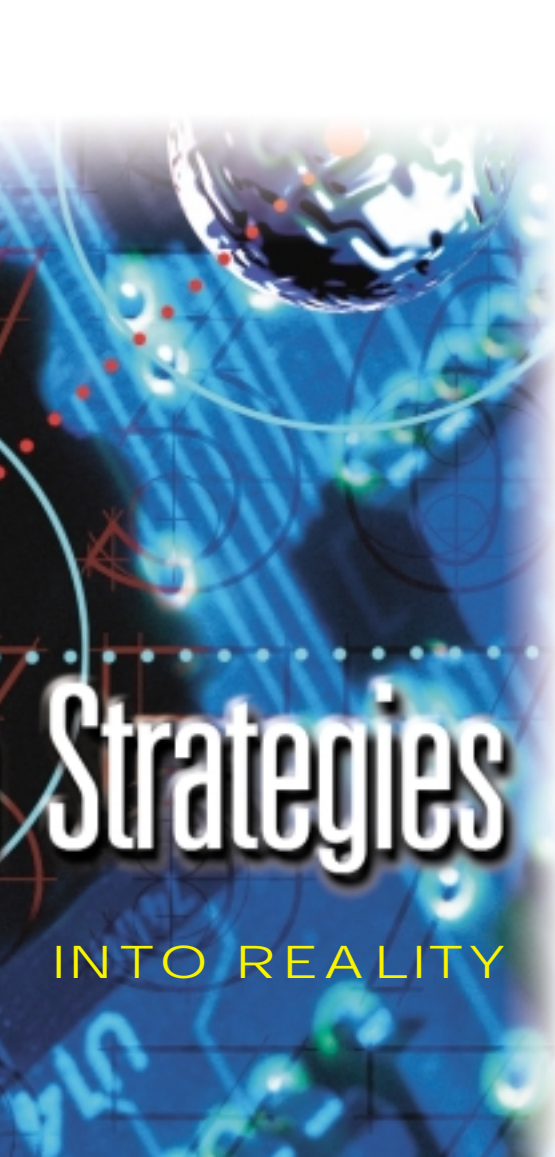
This trend is particularly true in the information technology (IT) arena, as the overspending of the last few years and the associated lack of ROI has resulted in intense scrutiny on IT spending. One of the initial responses to this new focus on cost-cutting from many IT organizations has been to consolidate much of their server infrastructure. By controlling and reversing the "server sprawl" trends of the last few years, organizations have been able to achieve substantial savings in server administration and management. Now organizations are seeking to achieve even greater savings by consolidating not only their server infrastructures, but the associated applications running on those servers – a decidedly more complex, yet potentially more rewarding, undertaking. In addition to the potential cost savings, organizations that are able to consolidate their applications will be positioning themselves for the even greater benefits of a truly adaptive

enterprise. This article explores the various strategies for consolidating J2EE applications running on BEA WebLogic, and the associated challenges and benefits of doing so.

The Promise

Consolidating J2EE applications, such as portals, running on WebLogic into a single hardware infrastructure holds the promise of extensive cost savings and improved application reliability. As Java and J2EE have evolved over the past several years, IT organizations have tended to deploy both ISV and internally developed applications into what amounts to a "one application per server" model. This made a lot of sense at the time, given the rapid pace of change in the Java platform over the last few years. Now that the platform has matured, IT organizations are left to deal with the resulting server sprawl and the low resource utilization associated with this model.

The good news is that tremendous cost savings can result from consolidating these applications onto more efficient platforms. Consolidating underutilized applications onto fewer but more modern processors can free existing servers for



Strategies

INTO REALITY

multiple, independent applications on the same hardware. Application consolidation immediately raises several issues:

- How can I guarantee service levels between applications?
- How can I prevent one errant application from affecting other applications?
- How can I reconcile differing needs for operating system patch levels between applications?...and so on.

It quickly becomes clear that any application-consolidation initiative has a high potential for failure unless these and many other issues are proactively addressed and accounted for.

The primary issue underlying application consolidation is striking the right balance between application isolation and resource utilization. In an ideal world, every application would be completely isolated from every other application running on the shared infrastructure, while simultaneously achieving optimal resource utilization. Of course, there is no ideal world and, to make matters worse, the goals of application isolation and maximum resource utilization tend to be inversely proportional to one another. Generally speaking, the more application isolation you achieve, the less resource utilization you achieve and vice versa. Figure 1 illustrates the various possible application isolation strategies and their relative level of resource utilization (note the linearity of almost all the strategies).

Flexible Strategies

Depending upon the underlying operat-

ing system, there are several different approaches to application consolidation. For brevity, I'll focus on two operating systems – HP-UX and Linux – that are representative of the types of platforms that IT organizations are likely to run across in real-world situations.

Single WebLogic Instance

The first and most obvious strategy of application consolidation using WebLogic is to simply run many applications in the same instance of the application server (i.e., in a single Java Virtual Machine [JVM]; see Figure 2). Modern versions of WebLogic have the ability to run multiple, independent applications on a single server instance in a secure fashion. This appears to be a relatively straightforward solution that would ensure optimal resource utilization within a shared infrastructure (in both single-server and clustered environments).

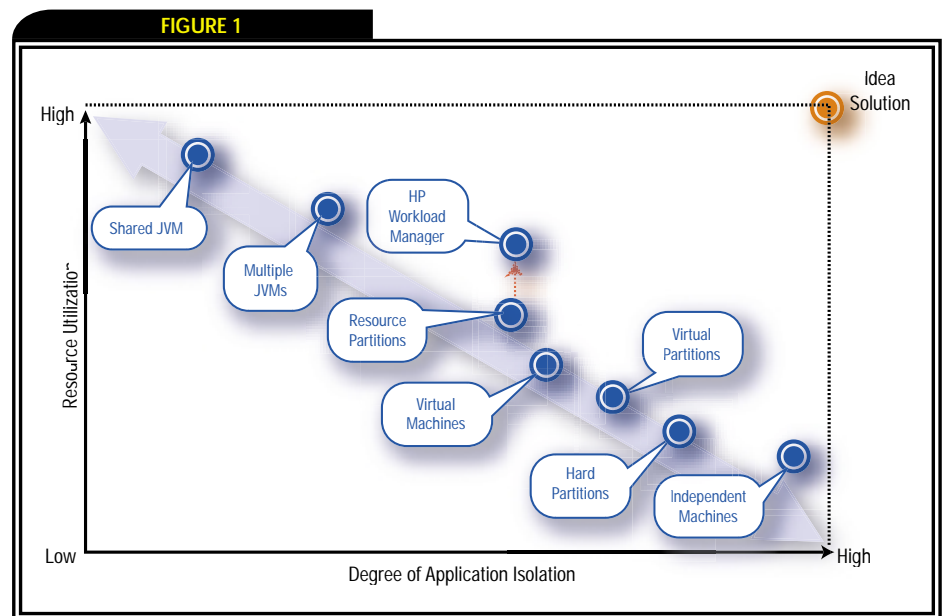
However, the drawbacks of this design quickly become obvious when you consider all but the most trivial scenarios. A single JVM environment allows for only a superficial level of isolation between applications, and therefore presents the following issues:

- Errant applications can negatively impact other running applications.
- There is no way to guarantee performance service levels for different applications.
- All applications must work with the same WebLogic version, JVM version, and associated patch levels.
- All applications must work with the same operating-system version and patch level.

new development or round out application testing and staging environments. As an added benefit, WebLogic server licenses can potentially be consolidated and recovered for use in future initiatives. These results can be further extended by utilizing a more recent version of WebLogic – with performance improvements of approximately 30% when moving from version 7.0 to 8.1, and over 100% when moving from prior versions to 8.1. Finally, the reliability of these applications can also be dramatically improved by moving single-server applications into fault-tolerant WebLogic clusters – thus reducing support and downtime costs.

The Challenge

At first glance, application consolidation appears to be only incrementally more difficult than server consolidation. However, while server consolidation generally consists of moving servers into a centralized administration and management infrastructure and eliminating redundancies, application consolidation involves running



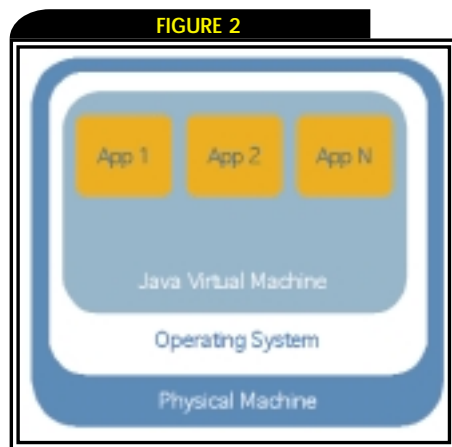
Application isolation vs resource utilization

- The potential for upgrade deadlocks between applications (i.e., an OS or JVM patch that one application requires ends up breaking another application).
- The potential for significant support issues when running third-party ISV applications in a shared infrastructure.
- JVMs tend not to scale well past four processor configurations, which limits the viability of the single-JVM model on larger machines.

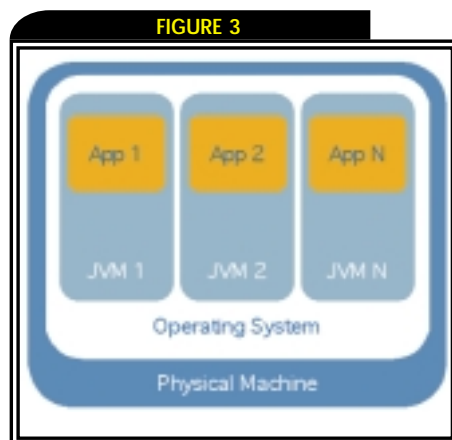
As a result of these issues, a single JVM shared environment is a viable model only in situations where extremely rigorous coding and testing standards are the norm and where only a very limited number of ISV applications (if any) are needed.

Multiple WebLogic Instances

The next logical strategy for running multiple WebLogic applications on a shared infrastructure is to run multiple instances of the application server (multiple JVMs) on the same physical machine (see Figure 3). This scenario affords much more application isolation than the single JVM model,



Single JVM Environment



Multiple JVM Environment

while giving up only a small degree of resource utilization. Running multiple WebLogic instances makes it more difficult for an errant application to impact other applications and allows different applications to use different WebLogic versions and different JVM versions and patch levels. This model also solves many potential support issues when running third-party ISV applications since each instance has its own virtual machine, heap space, and database connection pools.

While running multiple WebLogic instances clearly has some advantages over a single-instance model, it also has some significant drawbacks, including:

- Errant applications still have a chance (albeit a small chance) of negatively impacting other running applications.
- There is no way to completely ensure performance service levels for different applications.
- All applications must work with the same operating system version and patch level.
- Memory overhead from running multiple JVMs.

Despite these challenges, running multiple instances of WebLogic is a viable solution for application consolidation in many circumstances and represents a good balance between application isolation and resource utilization.

Virtual Machines and Virtual Partitions

While a multiple JVM configuration provides some degree of application isolation, there are many circumstances where greater isolation is necessary between applications. The next logical level of isolation is at the operating system level and is achieved through the use of virtual machines (in the case of Linux) or virtual partitions (or “vPars” in the HP-UX world). In the virtual machine scenario, a single physical machine runs a primary operating system (in our case, Linux) that serves as a “host” OS for various “guest” operating systems that run virtual memory spaces. The host OS essentially virtualizes physical system resources (CPU, disk, I/O, etc.) and coordinates access to these resources by the guest operating systems (see Figure 4).

Virtual partitions employ similar concepts, but they do not require a dedicated “host” operating system (there is a vPar monitor that performs this function) nor do they truly virtualize system resources (physical resources are assigned to each virtual partition).

Implementing virtual machines on Linux requires a third-party application, such as the VMWare ESX platform, while virtual partitions are a built-in feature of HP-UX 11i. From a performance standpoint, VMWare virtual machines and HP-UX virtual partitions differ somewhat in their implementation details. Being part of the base OS and not having to virtualize all resources, HP-UX virtual partitions tend to have a lower overhead penalty than VMWare virtual machines, but are limited to running multiple versions of HP-UX only. Conversely, VMWare virtual machines pay a higher overhead penalty but are able to host a wider range of “guest” operating systems, including Windows NT/2000/2003, Red Hat Linux 7.x / 8.x, Red Hat Advanced Server 2.1, SuSE Linux 7.3, and FreeBSD 4.5 (all on the IA-32 architecture).

Since each virtual machine/partition runs a completely independent operating system instance, they achieve a high degree of application isolation, including the ability to dedicate a certain percentage of system resources to particular virtual machines to achieve service-level commitments. (Note: In the case of HP-UX virtual partitions, processor resources are allocatable only on a per-CPU level of granularity.) In effect, each application is completely unaware that it is running in a virtual, rather than a physical, machine. Each virtual machine/partition can also be quickly reconfigured based upon demand, giving system administrators a very high degree of flexibility. However, the price for such independence is the inherent overhead involved in managing multiple OS instances and their associated memory requirements.

“Hard” Partitions

Aside from actually using multiple physical servers, the far extreme of the application isolation scale is the use of “hard” partitions on high-end Unix systems such as the HP Superdome platform. A hard partition conceptually resembles a virtual partition, except that it is actually implemented in the hardware architecture of the machine. Each hard partition is physically allocated a certain number of CPUs, memory, storage, and so on from a pool of resources available within the machine. Once allocated, these partitions act, for all intents and purposes (including fault tolerance), as separate physical machines, thus providing the highest possible degree of application isolation but the lowest overall resource utilization.

Quest Software

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

From a consolidation standpoint, hard partitions can achieve a marginally higher level of resource utilization than independent machines due to their reconfiguration flexibility and manageability. The resources of hard partitions can quickly be reallocated to other partitions to meet the ever-changing needs of the organization. Management and operational costs can also be reduced through the use of hard partitions by eliminating the redundant system administration resources needed to manage a decentralized environment.

Resource Partitions and HP Workload Manager

A resource partition is similar to a virtual partition or a virtual machine, but it utilizes only a single instance of the HP-UX operating system. In this scenario, each resource partition gets its own allocation of CPU resources, which is managed by its own process scheduler, and its own allocation of memory resources, which is managed by its own memory management subsystem. Thus, a resource partition can simplistically be viewed as a “mini” virtual machine – without as much overhead, but restricting

the system to a single instance (and hence, version) of the operating system.

While resource partitions represent an interesting compromise between multiple JVMs and virtual machines, they are able to achieve even higher levels of resource utilization by using an HP product called Workload Manager (WLM). WLM essentially sits on top of a set of resource partitions and periodically monitors their performance characteristics versus a set of pre-established service-level objectives (SLOs). If the SLO of any resource partition is not being met, then WLM can dynamically reallocate processor and memory resources from other resource partitions to achieve the correct service level. That is, if one partition needs additional CPU or memory resources, WLM can dynamically reallocate resources from another partition and add them to the needed partition – all automatically and without human intervention. Ultimately, this increases overall resource utilization while simultaneously ensuring that system resource allocations never fall below a minimum threshold specified by an application owner (see Figure 5).

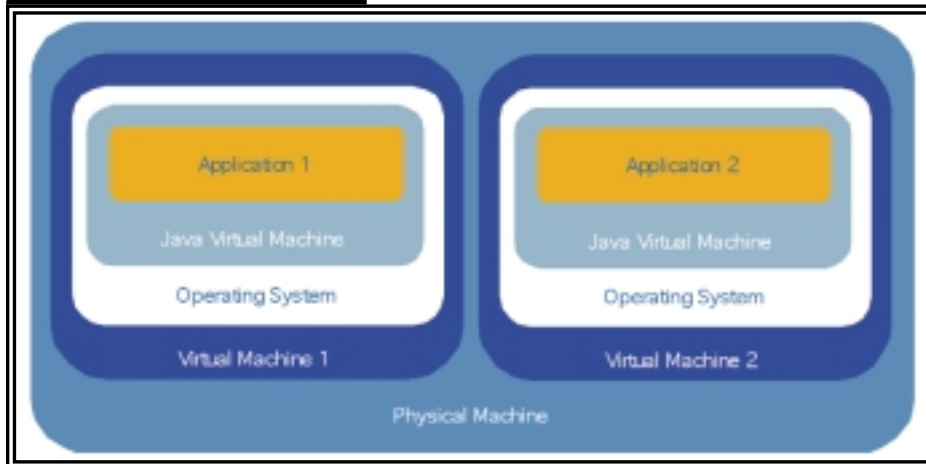
Workload Manager also has WebLogic-specific monitoring functionality that allows it to dynamically reallocate resources to partitions based upon BEA WebLogic Server’s free thread count and/or work queue length. This further ensures that system resources are utilized in an optimal fashion while maintaining a high degree of application isolation.

On large, multiprocessor HP-UX systems, HP Workload Manager probably represents the best combination of application isolation and optimal resource utilization for BEA WebLogic for most circumstances.

Combining Application Isolation Strategies

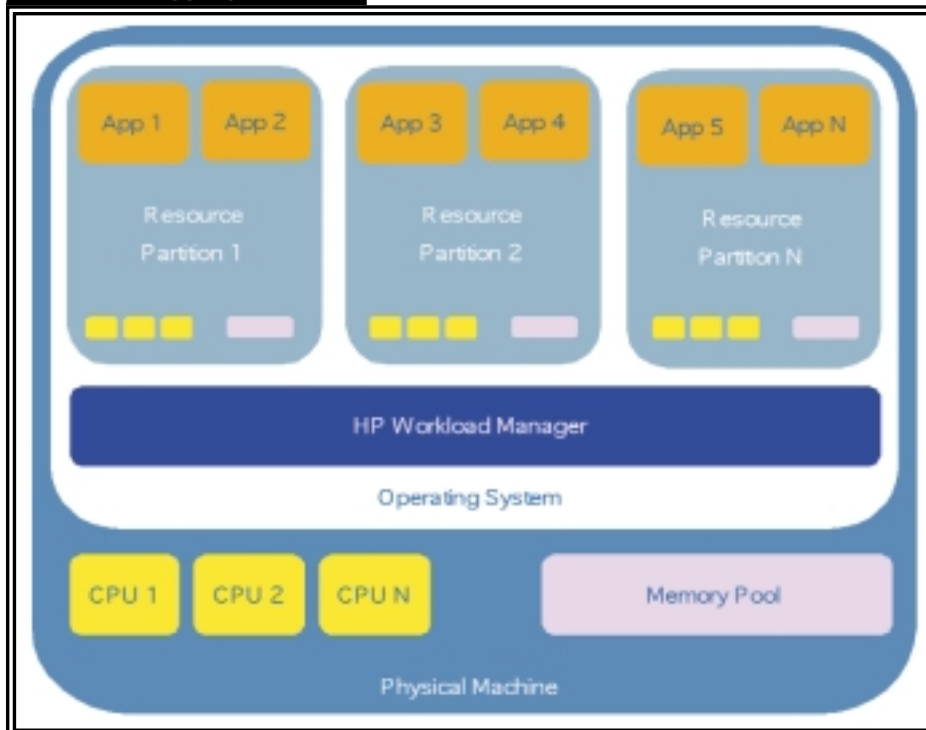
The good news is that many of these application isolation strategies can be combined to achieve the right level of granularity to effectively balance isolation needs with resource utilization targets. In order to achieve the desired results from an application isolation strategy, it is likely that many enterprises may actually need to combine several solutions to precisely fit their needs. To illustrate the possible combinations and the decision-making processes behind choosing the right combination strategy, I’ll look at two different scenarios involving multiple application isolation requirements.

FIGURE 4



Virtual machines/partitions

FIGURE 5



Resource partitions and Workload Manager

continued on page 24

Veritas

www.veritas.com

Building Infrastructure-Aware Applications



BY ROBERT SHEAR

AUTHOR BIO...

Robert Shear is the director of product management for Blue Titan Software. He has more than 10 years of experience in the high-tech industry. Before joining Blue Titan, Robert was director of marketing for CommerceRoute, a pioneer in XML appliances, and a development engineer for the Scripps Institution of Oceanography. He holds a BS in physics from UC San Diego and an MBA from the Haas School of Business.

CONTACT...

rshear@bluetitan.com

THE POWER OF A SHARED INFRASTRUCTURE

Just as service-based applications leverage a shared set of application resources, infrastructure-aware applications leverage a shared infrastructure that can adapt to meet the needs of the application. This article will introduce the basics of a shared infrastructure, look at the Blue Titan Network Director and how it interoperates with WebLogic Enterprise Platform to enable infrastructure-aware application development, and provide a glimpse into the future of adaptive applications.

Infrastructure-Aware Applications

Applications have traditionally been stand-alone islands of functionality. Web services are changing this by providing a natural way to decompose applications into granular pieces, allowing them to be recomposed and recombined across multiple systems for different uses. This granularity, along with the ability to compose higher-level applications out of individual pieces, provides developers with new facilities to build extremely sophisticated applications with a high degree of reuse and efficiency.

A number of emerging standards and technologies will extend this granularity and reusability down into the application infrastructure itself – allowing *infrastructure-aware applications* to

interact with the application backbone in real time to serve broader service-level or reliability requirements. For example, if the latency of a Web service increases to an unacceptable level due to a traffic spike, an infrastructure-aware application would be able to increase capacity by adding another service instance, or alternatively, limiting access to only priority customers. Either of these actions would ensure that performance stays within the bounds of service-level agreements (SLAs) established with your most important customers.

Blue Titan gives developers a practical way to create infrastructure-aware applications by providing a *shared infrastructure* that virtualizes core functions such as SLA enforcement, routing, failover, access control, logging, and prioritization into a common set of shared services that can be reused across any number of Web service applications. This clean separation of application from infrastructure functionality allows developers to focus on creating core business logic without concern for deployment details, while also allowing administrators to define and enforce consistent, enterprise-wide control policies for all Web services.

Shared Infrastructure

The benefits of shared infrastructure are numerous and clear. From a development perspective, developers are no longer burdened with

An Application Traffic Control System

As an analogy, consider the infrastructure that supports the airline industry. Aeronautical engineers have specialized knowledge of structures, flight systems, and aerodynamics; and concentrate on building the best aircraft possible. These engineers explicitly do not think about building good airports or how operational 737s are routed between cities. However, they do need to ensure that 737s meet the interface requirements of the infrastructure. For example, all aircraft must be able to take off and land on existing runways; they must be able to communicate with the air traffic control system via appropriate radio links; and they must be able to use standard jet fuel. Should the aircraft meet these interface requirements, it can be operated and managed easily using the existing infrastructure. Developing applications that run within a shared infrastructure is similar. Developers are responsible for making the best application possible. The only constraint is that it has the appropriate interfaces to take advantage of the shared infrastructure.

If you are currently building routing, failover, security, and prioritization into your Web service applications, your life will be much easier with a shared infrastructure in place.

building infrastructure into every application. Instead, they simply utilize a set of shared services that are executed in the network. From an administration perspective, administrators can define and apply uniform infrastructure practices to all of the applications they manage, and maintain a well-defined, common set of services and infrastructure facilities. From a resource utilization perspective, infrastructure can be managed and scaled as a cohesive set of resources to meet broad per-

formance or reliability requirements, rather than as a disparate set of isolated islands.

To create an adaptive application backbone a shared infrastructure must have four key attributes:

- **A set of shared network nodes that monitor and route Web service traffic in accordance with established infrastructure policies:** These nodes proxy all Web service traffic and are generally managed by the enterprise IT organization.
- **A set of shared infrastructure services that can be provisioned across an organization to be utilized by multiple developer groups:** These services either run on the shared network nodes or on specialized enterprise systems (for example, a shared authentication service would probably run on an existing identity system).
- **The ability to define and enforce unified policies across the entire shared infrastructure:** These policies govern such shared functions as authentication and prioritization of requestors, logging of inbound messages, and compensation for breached SLAs.
- **An open, service-based interface that provides access to infrastructure control functions and service metadata.**

With these four pieces in place it's possible to create an adaptive feedback loop between real-time service performance and the underlying application infrastructure.

In Figure 1, a continuous compliance cycle runs in the background of every application to verify that all associated infrastructure policies are met. If all policy requirements are met, this cycle just hums away unnoticed. When a policy is breached (unacceptable latency, multiple unauthorized requests, etc.), an event is generated that triggers a set of compensation services. These services adapt operational properties of the shared infrastructure to return the service to a compliant state.

A Practical Implementation of Shared Infrastructure

Blue Titan Network Director is enterprise software that enables the control and coordination of Web services. It provides a shared infrastructure of Control Points (network nodes) and Fabric Services (shared infrastructure services). When used in conjunction with the BEA WebLogic Platform, Network Director enables a practical path toward infrastructure-aware applications.

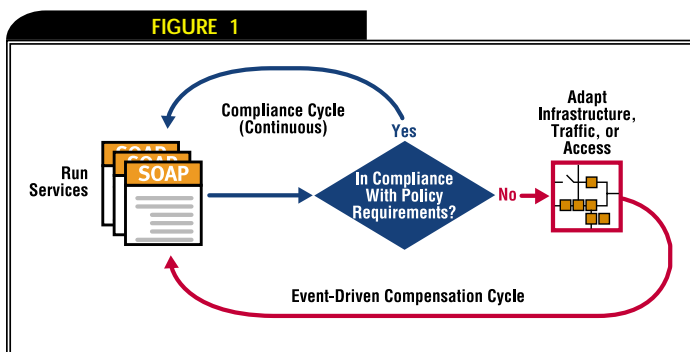
Services created in BEA WebLogic Workshop are registered with the Network Director, where they become manageable network resources. Once registered, services are assigned specific security, management, and monitoring policies that are enforced across the enterprise in a consistent, unified way. Developers are responsible for creating services with well-defined interfaces and WSDLs, and IT handles the rest. This way, if policies change (say a new logging requirement is imposed), the service itself remains valid and usable.

Fabric Services

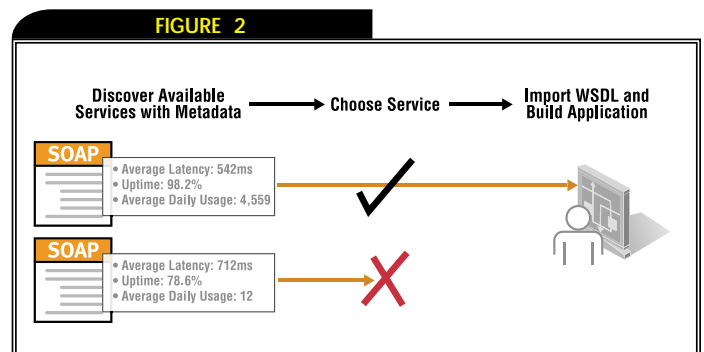
Currently Blue Titan provides a set of over 80 shared, reusable services out of the box. These Fabric Services allow developers and administrators to programmatically interact with the Network Director without coding or scripting. They give programmatic access to service metadata, service management functions, and control functions. This lets IT build infrastructure policies that can affect traffic flow, capacity, and access properties of the network to meet operational requirements.

Indirect Addressing Through Control Points

Blue Titan Control Points execute and enforce Web service control policies across an enterprise. Control Points are servlet-based components that run on BEA WebLogic Server and are deployed across a

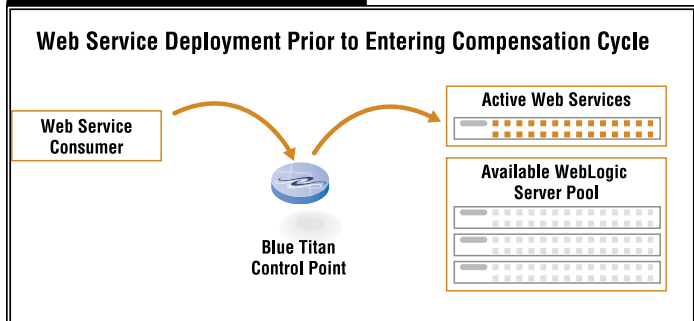


Adaptive infrastructure flowchart



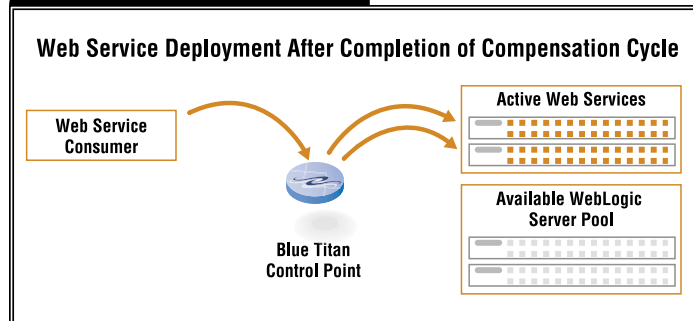
Analyzing service metadata at build time

FIGURE 3



Deploying additional service instances based on SLA breach

FIGURE 4



Deploying additional service instances based on SLA breach

network as dictated by traffic, security, or reliability requirements. Upon registration with the Network Director, all Web service interfaces are assigned a virtual address that references a Control Point. All requests for the service are routed first to a Blue Titan Control Point and then to the corresponding service provider. This indirect addressing enables all of the control, coordination, and metadata functions provided by Network Director. As requests and responses pass through the Control Point, policy is enforced (access, failover, etc.) and operational metadata (response time, uptime percentage, etc.) is gathered and stored in logs. This information can then be referenced by any application with any Web service-compatible application, including WebLogic Workshop or Portal.

Infrastructure-Aware Application Examples

Using the BEA WebLogic Enterprise Platform and the Blue Titan Network Director, developers and enterprise IT can collaborate to create an adaptive infrastructure. Given that all visibility and control functions are Web services themselves, the flexibility of the resulting infrastructure is almost unlimited. The examples below demonstrate a few simple ways that a developer can utilize the shared infrastructure to make more reliable and responsive applications.

Analyzing Service Metadata at Build Time

Using the Blue Titan Network Director and BEA WebLogic Workshop, a developer can query up-to-the-minute performance information for a specific service. With this information, decisions can be made on whether to use a specific service or an alternative with a better performance and reliability history. (see Figure 2).

Using WebLogic Workshop a developer

would call the Blue Titan Fabric Service `listConsumableServices` to display which services are available for use. In this example, assume that there are two versions of a service: `ServiceA.1` and `ServiceA.2`. Then the Fabric Services `getLatencyByService`, `getUptimeByService`, and `getUsageByService` would be called to view the metadata for `ServiceA.1` and `ServiceA.2`. The resulting information would then be used to decide which service is better suited to the application (see Figure 2).

Note that this sequence of Fabric Service calls could easily be aggregated into a single coarse-grained Web service named `listServiceMetadata`. This service could then be provisioned to developers for future use, made into a Workshop Control, or hooked into BEA WebLogic Portal to provide a graphical interface. The result is a streamlined process for discovering and acting upon service metadata at build time.

Deploying Additional Service Instances Based on SLA Breach

An infrastructure-aware application can interact with the underlying WebLogic Platform to fulfill service-level agreement (SLA) requirements. Assume that a certain high-value service, `tradeStatus`, has a maximum latency constraint of 500ms. In other words, the service provider has guaranteed that `tradeStatus` will execute within 500ms or the usage fee will be returned to the consumer. To enforce this constraint the service provider utilizes the Blue Titan Network Director.

Look again at the flowchart in Figure 1. The compliance cycle runs continuously to monitor the operational metrics of `tradeStatus`. A usage surge causes latency to exceed 500ms and an event is generated that kicks off the compensation cycle. Given that the compensation cycle simply calls other (infrastructure) Web services, it

can perform any number of actions in response to the event. In this case, the compensation cycle will call two Workshop Controls, `deployWebService` and `registerWithBlueTitan`, which will bring an additional instance of `tradeStatus` online, increasing capacity and reducing latency. (see Figures 3 and 4).

The Workshop Control `deployWebService` invokes a previously defined set of Ant tasks that automatically build an EAR in Workshop and deploy it to an available WebLogic Server. Then the Workshop Control `registerWithBlueTitan` invokes a series of Fabric Services that registers the new WSDL with the Network Director and assigns appropriate policy. Once that is done the Web service will be accessed automatically by consumers and included in the compliance cycle. Should the latency remain out of bounds, the process can be repeated.

Conclusion

BEA and Blue Titan are actively collaborating to provide a best-in-class Web service infrastructure. The WebLogic Platform is purpose built to compose and run business-critical Web services. Blue Titan Network Director augments these capabilities by providing a common control infrastructure to ensure that all Web services are utilized in a consistent way across an enterprise. Today this offers customers a practical way to operate reliable and scalable Web service applications. In the near future, as customers utilize conversations to support asynchronous processes, BEA and Blue Titan will extend the flexibility of Web services into event-driven stateful processes that span multiple systems or domains. The end result is a 100% standards-based approach to enterprise computing and a dramatic reduction in the cost of building, integrating, and operating business applications. 🔴

E.piphany
www.epiphany.com/psgreport



MEETING THE GOALS OF TRUE APPLICATION INTEGRATION



BY LAURENCE MORONEY

AUTHOR BIO...

Laurence Moroney is a senior architect with the Reuters Innovation Labs and Rapid Development Groups based in New York City. He has worked in the computer field for over 10 years, and has developed software in many fields, from financial services to professional soccer to casino security and surveillance. He is a member of the Microsoft .NET advisory council and sits on various councils for XML development.

CONTACT...

Laurence.Moroney@reuters.com

When talking about enterprise application integration, we tend to think of using Web services technologies such as SOAP and UDDI to virtualize a data model across a large enterprise. The thinking is that with a consistent interface, the data stores of the company can be abstracted behind a Web services layer and reported in XML, which can then be kneaded to the particular needs of your application.

In an ideal world, this would be simple. However, we don't live in an ideal world, so there are a number of problems. The first problem is that we often want to take data from one service and use it as a parameter to call another, take returned data from that service and use it to call another, and so forth. We would then want to take pieces of data from each and aggregate them together to form our response to the original call.

In addition, data-type conversion may be necessary to map parameters to each other. For example, service 1 may return a double that is required by service 2 as an index, but service 2 expects it as a float. Interoperability is more than just being able to parse WSDL to create a proxy – operations such as data-type conversion are paramount.

Finally, you cannot expect all the services to run on the same platform. There may be .NET Web services, COM, DCOM, or other J2EE Web services that need to be integrated, to name a few. This is far more complicated than simple data aggrega-

tion, but is a real-world requirement; and with Web services development platforms alone it would require a lot of hand coding, defeating the purpose.

Here's where a tool such as BEA WebLogic Workshop 8.1 and a platform such as BEA WebLogic Integration 8.1 come into play.

A Real-World Use Case

A fictional company, The Pension Source (TPS), provides pension administration for their customers. Their pension plans have always been based on funds based in the U.S. but they want to branch into international funds. Their offices in Europe have a Web service that provides foreign exchange information, and these offices are standardized on Oracle as an application server. In addition, they have a domestic database that contains the recent price history and currency code for each fund. They would like to integrate these Web services to provide a new application. What will this entail?

1. Exposing an interface that allows the user to query the dollar price of a particular fund
2. Pulling the price of the fund from the database
3. Pulling the currency code of the fund from the database
4. Converting the price to U.S. dollars
5. Returning the converted price to the user

In step 2, the fund price is returned to the user as a float. In step 4, the distant Web service requires the price to be passed to it as a double.

We'll see how straightforward the design and implementation of such a workflow and orchestration is with BEA WebLogic Workshop.

Building the Integrated Application

The downloads for this article (located at www.sys-con.com/weblogic/sourcec.cfm) include a zip file (CurrencyConv.zip) containing an emulation of the currency conversion Web service as an Oracle JDeveloper project that may be installed on Oracle JDeveloper and run on OC4J. Please check the readme file included with the download for instruction. In addition, PensionAPI.zip contains a WebLogic Workshop 8.1 application that emulates a service that retrieves data from the pension fund's database, and exposes it as an API. For simplicity, neither application is tied to a back-end database; they simply emulate a real-world scenario where they would.

Once both services are up and running, start WebLogic Workshop 8.1.

Getting Started with the Workflow and Adding a Client Request

- Create a new empty application called TPSReport. For the server, make sure that you select the "Integration" server on the dropdown list.
- Create a new empty workflow project called GetFundInformation.
- Within the workflow folder you will see a file called workflow.jwf. Double-click it to open the designer.
- Double-click the "Starting Event" and select "Invoked By a Client Request"
- Double-click the Client Request node, select "String", and click Apply.
- Select the "Receive Data" tab. This is where you will assign the parameter that has been passed by the user to a variable that is internal to your workflow. That variable does not yet exist, so in the next

steps you can add it.

- On the right side of your screen, you should see the Data Palette. Click "Add" besides the Variables tab, and then call the variable strSymbol and make it a string.
- On the "Receive Data" tab of the Client Request node, click "Select Variable", variable strSymbol is now on the list.
- Click "Apply" and then "Close" to update the Client Request node.

You've finished the first step of the workflow, taking the data in from a client request. Your screen should look something like Figure 1.

Consuming the Fund Price Web Service

Now that you have the Client Input, as a string denoting the symbol of the fund you are interested in, the next step is to consume the Web service that exposes the Fund Price History and Currency Code.

Consuming the Price History:

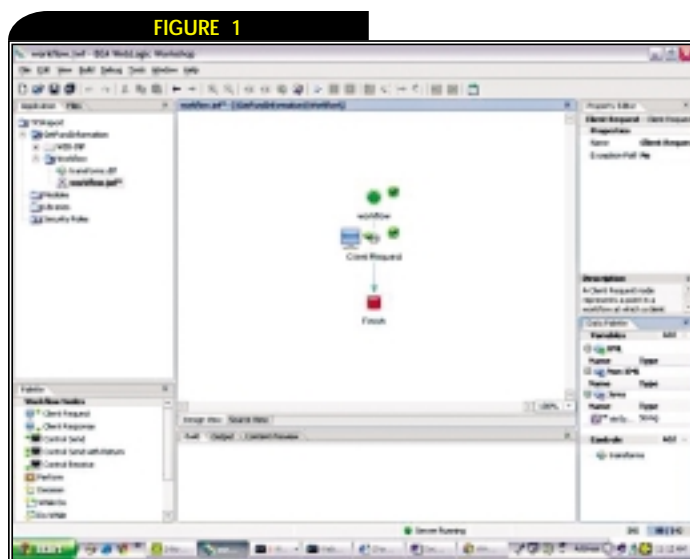
- From the palette on the bottom left side of the screen, select "Control Send With Return" and drag it onto the green line between the Client Request node and the Finish.
- Double-click the node that appears.
- You can see that this node requires a control to continue. With the designer you can add a new one on the fly as follows:
 - On the right side of the screen (use Figure 1 as a guide) the Data Palette contains a Controls tab. On it, click "Add" and select "Web Service".
 - In the ensuing dialog, on step 1, call the control ctlPensionDetails
- In step 2, select "Create a New Web

Services Control to Use" and call it ctlPensionsAPI

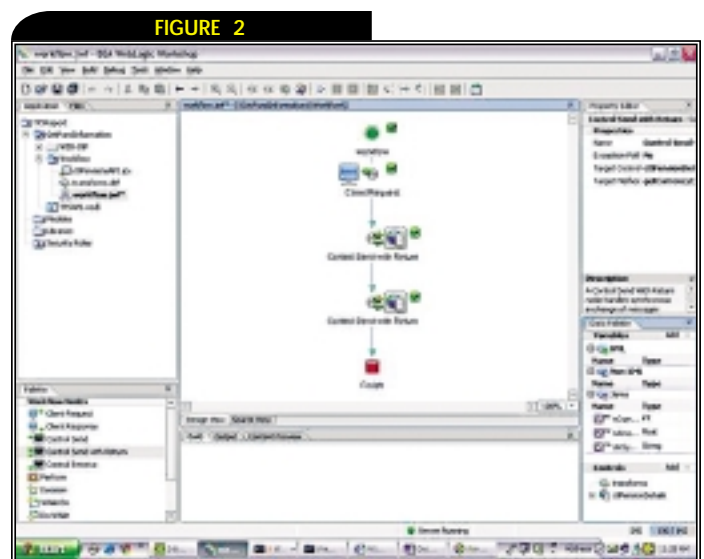
- In step 3, enter the URL to the WSDL of the Pensions API Web service that you downloaded and installed earlier.
- Click "Create" and you will return to the node designer.
- You can now select "ctlPensionDetails" as the control implementing the Web service, and when selected, the methods list box is populated.
- Select the method "getPrice" and click "Apply".
- Select the "Send Data" tab. This is where you set up the parameters that you are passing to the Web service and where they come from.
- Click "Select Variable" and select "strSymbol". This will take the variable created by the User Call in the previous node and dispatch it as a parameter to the PensionsAPI Web service. Click "Apply".
- Select the "Receive Data" tab. This is where you specify what to do with the data that comes back from the Web service.
- We don't have a variable to hold this data, so create a new one as before. Make it a Java float, and call it nAmount.
- It will now be available on the "Select Variable" list, so select it and click "Apply".
- Click "Close". You now have a node in your workflow that dispatches the client request to a Web service and gets a value back.

Repeat these steps to add another node, calling the same control, but this time calling the "getCurrency" method. Map the returned value to a new integer called nCurrency.

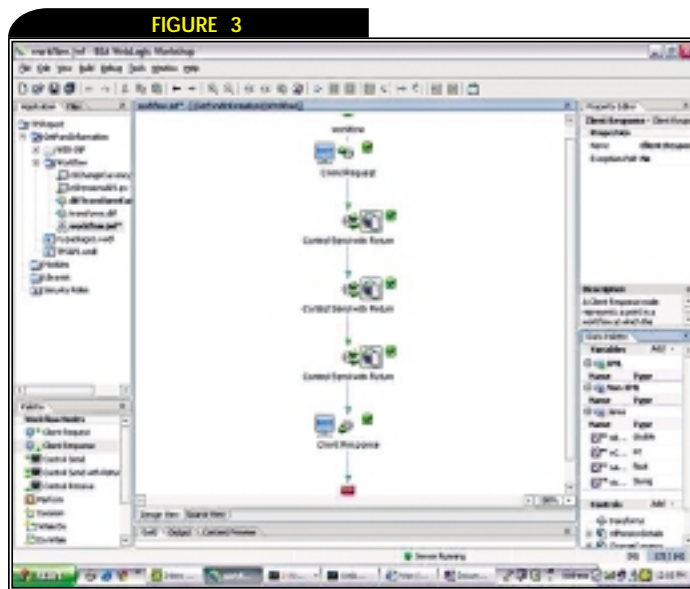
Your screen should now look like Figure 2.



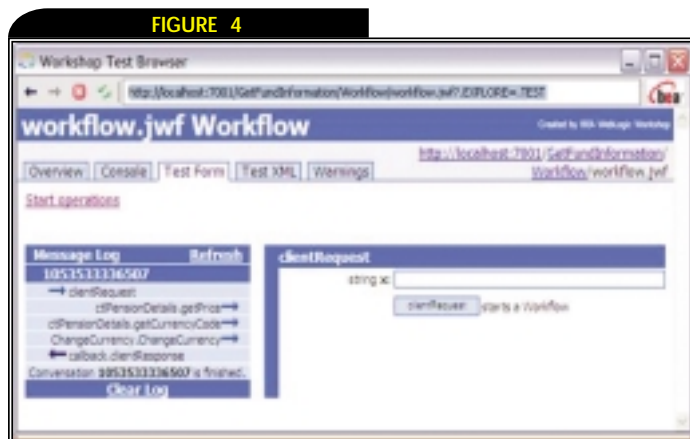
The workflow containing the Client Request



The workflow with the calls to Pension Funds API installed



The completed workflow



The test harness for the workflow

Orchestrating the Change Currency Web Service

Now it gets interesting. At this point you have a workflow using all WebLogic Web services. You are going to integrate the service running on another platform – in this case the Currency Conversion service running on Oracle OC4J.

As before, drag a “Control Send with Return” node onto the workflow, dropping it just above the “Finish” node.

- Double-click it to configure it.
- You do not yet have a control for the Web service, so click “Add” near “Controls” on the Data Palette.
- Select Web Service”.
 - Step 1: Call the instance ChangeCurrency
 - Step 2: Select “Create a new Control”. Call it ctlChangeCurrency.
 - Step 3: Enter the WSDL of the running Web service. If you followed the instructions in the download it should look something like: <http://machinename:8888/WebLogicArticle-CurrencyConv-contextroot/mypackage1.ChangeCurrency?WSDL>
 - Click “Close”.
- Return to configuring the node.
 - On “General Settings” select the Control “ChangeCurrency” and the node “ChangeCurrency” and click “Apply”

- On “Send Data” select nAmount and nCurrency in step 1.
- You’ll notice on the dialog that the input variables are a “double” and an “int”, whereas the values you are about to pass are a “float” and an “int”. To transform them, on step 2a in the dialog do the following:
 - Click “New”. In step 1 set the Variable Name to “TransformForCurrency” and in step 2, set the control name to dtfTransformCurrency
 - Enter “TransformAmount” in the Method box and click “Create Transformation” in step 2b.
 - The Map Variables dialog appears. Drag the float to arg1 and the int to arg 2.
 - Click OK when you’re done.
 - Click Apply to update the “Send Data” settings
 - On the “Return Data” node, the double needs to be mapped to a variable. Create a new double variable as before, and call it nReturn. Select it, click “Apply” and then click close.

You have now integrated the distant Web service, and it was done as seamlessly as the Web services running on the same WebLogic platform.

Returning the Data to the User

The user has now passed a string containing the fund that they are interested in to your workflow, and the workflow has pulled the details out, and translated the currency to US dollars. The last step is to return the currency-translated value back to the user.

This is very simple. On the Palette, select a “Client Response” node and drop it just above the “Finish” node.

- Double-click the “Client Response” node and select a “Double” on the “General Settings” tab.
- On the “Send Data” tab, select the variable called “nReturn”, click “Apply”, and then click “Close”.

Your finished workflow should now look like Figure 3.

Testing the Workflow

Build the application and run it. The test harness will run and display the screen shown in Figure 4.

Enter “ABC” in the textbox and click “clientRequest”. Wait a few seconds and click “Refresh” on the Message Log. Keep doing this until you see a callback.clientResponse entry on the Message Log.

By clicking on the nodes within the harness, you can see the calls and the data being passed around the different Web services.

In the case of “ABC” the amount was being reported in currency code 1 (dollars) so no conversion was required. As an exercise, try to amend the workflow so that it doesn’t bother to call the Currency Conversion Web service in this case. (Hint: Use a decision node.)

By trying the symbols “BBC” and “CBC” you can investigate the currency conversion as it is happening.

One really great thing about this is that this compiled workflow is actually running as a Web service, so it in turn could become a node in another, more complex orchestration!

Conclusion

In this example you built a simple orchestration. You have barely touched on the features that are available in BEA WebLogic Workshop 8.1 for orchestrations. Other nodes can have Java code associated with them, be conditional, loop, or be asynchronous. The goals of true application integration – data aggregation, service chaining, and orchestration – are now in your hands. 🍷

Yantra Corporation

www.yantra.com

Enhancing Application Manageability with BEA's WebLogic Server and HP's OpenView

PART 2



BY JUSTIN MURRAY

AUTHOR BIO...

Justin Murray is a technical consultant in the application development resources organization in HP. He has worked for HP in various consulting and teaching roles. Justin has consulted at a technical level on customer projects involving Java, J2EE, and performance management, as well as specializing on application performance tuning for HP-UX. Justin has published several technical papers on these subjects.

CONTACT...

justin.murray@hp.com

In Part 1 of this article (WLDJ, Vol. 2, issue 6), I claimed that manageability is a vital aspect for any application that will be deployed into production, where it will spend most of its life being managed by people who may not be the original designers. These systems and applications managers will need good tools and application visibility for the deployment of the application to be successful. We looked at the basics of manageability, from logging files to log file encapsulation to writing our own JMX beans in order to achieve ever higher levels of manageability in the application.

In this article, I'll look at the means by which the various management tools, from the BEA console and WShell utilities to the HP OpenView features, help the developer and set templates in place for building better applications.

There are three JMX client applications available with BEA WebLogic Server today: the Administration Console, the WShell tool, and the `weblogic.Admin` command-line utility. The next section gives an outline of the first two as useful tools for the developer. The command-line utility functionality is also seen in the WShell tool, with a friendlier front end, so it will not be described here.

The WebLogic Server Administration Console

This tool, one of several different BEA consoles, is specifically geared to managing WebLogic Server. There are other BEA consoles for the Portal Server, the B2B server, and the Integration Server Application management that we will not discuss here.

Since WebLogic Server 7.0, the WebLogic Server administration console has been a browser-based management environment for many aspects of the WebLogic Server and parts of the applications this server supports. The WebLogic Server Admin console can report and change the status of any servers that it can connect to and any services within those servers, such as a JDBC connection pool service.

One of the console's screens is shown in Figure 1. It allows the application developer to visualize the effects of his/her application on the WebLogic Server and to set various changeable aspects of that environment to help the application run well.

At runtime, the WebLogic Server automatically supplies a JMX Managed Bean (MBean) object for the server itself (called "`weblogic.management.configuration.ServerMBean`"), for internal services and for each application servlet and application Enterprise JavaBean (EJB, or business object) under its control.

These WebLogic Server-generated MBeans, called `RunTimeMBeans`, then become manage-

MANAGEMENT TOOLS AND THEIR USE OF JMX

ment-oriented proxies for their companion application objects. The attributes of the RuntimeMBeans for WebLogic Server-owned objects can be queried, and in some cases set to new values and certain operations executed on them. This is done through the forms that the WebLogic console shows on its user interface. There is a full help system that explains the many forms of MBeans that WebLogic supports in the WebLogic Console, seen as question marks beside their names. Figure 2 shows the name and attributes of one particular MBean entry. This example takes the MBean representing the WebLogic Server process itself and shows us the value of the "ListenPortEnabled" attribute.

To support these facilities, WebLogic Server has an MBean server capability which carries out the registration of its generated RunTimeMBeans. The user can therefore see (and adjust) the allowed attributes of these MBeans through the WebLogic Server Admin console. The benefit of having a server-supplied management MBean object representing each servlet and each EJB is that the application is already visible for management down to these levels of granularity. The developer can now work with the MBean logic to add new functionality as required. For example, the properties or attributes of each object can be exposed through forms and data entry fields in the console, under the developer's control.

Taking the ShoppingCart example, this object may have been implemented as a Session EJB by the developer. We can then interrogate the attributes of the ShoppingCartMBean object in the Admin console. By constructing the logic in the MBean such that it has a link to the ShoppingCart object and reads and writes data to the ShoppingCart attributes, we provide the management tools with a connection into the business object.

This can be summarized as an orderly way to build control, as described in the manageability definition, into the system.

The WLS Shell Tool

This tool is a command-line interpreter that connects to a WebLogic Server and translates user commands to JMX calls. These calls are applied to those MBeans registered in the MBean server supplied by BEA, but they might also be applied to user-defined MBeans, if necessary, provided those user-defined MBeans have been registered with the MBean server.

The tool allows developers to name a specific MBean that they are interested in

monitoring or controlling and to issue commands to it, such as "get the value of the MaxCapacity attribute of MBean named X" or "set the MaxCapacity attribute to 300 on MBean Y". The tool uses a familiar tree-like file system analogy for navigating the domain name, MBean type, name of the MBean, and finally the attribute of the MBean that you wish to get or set. Because the name of a JMC MBean can be long and tedious to type, the tool allows for shorter navigation to an attribute name from a longer JMX name.

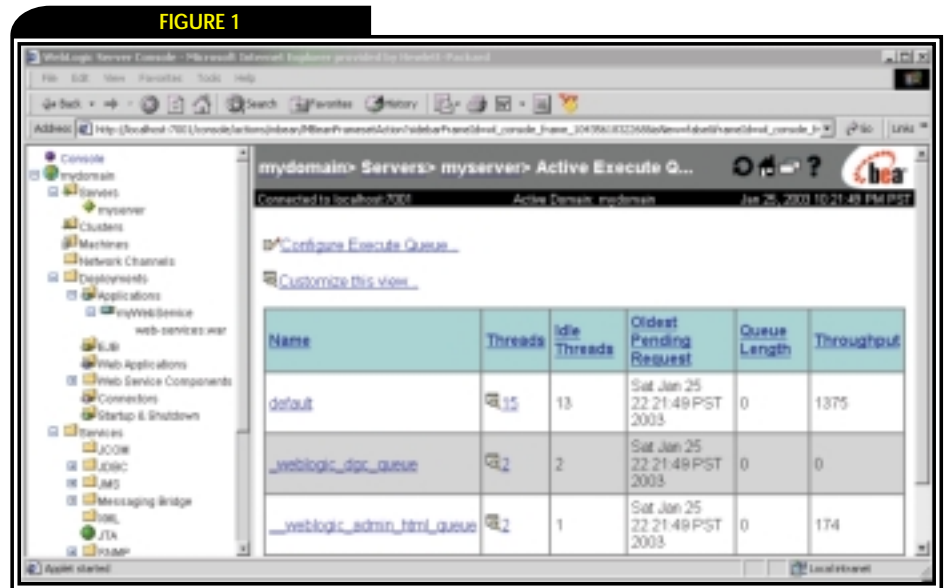
A JMX-Compatible Smart Plug-in for HP OpenView Operations

HP enhances the BEA WebLogic Server management world by providing a WebLogic Server-specific Smart Plug-in (SPI) module that integrates the manage-

ment of the WebLogic Server and any associated applications that depend on it. This SPI integrates several sources of information into one screen. It

depend on them *from the same console and using the same familiar tools*. The OVO WebLogic Server SPI uses the JMX MBean server within the WebLogic Server runtime and derives some of its information from the MBeans present in that server. The HP SPI may be used by developers in three ways:

- Existing information (contained in existing MBeans) may be used to provide information through existing OVO-supplied metrics. There are 55 such metrics available for use by the IT operator. These metrics include the number and percentage of available JDBC connections in the overall WebLogic Server connection pool.
- Existing information may be used to define new user-defined metrics (UDMs).
- New, custom-built MBeans may be

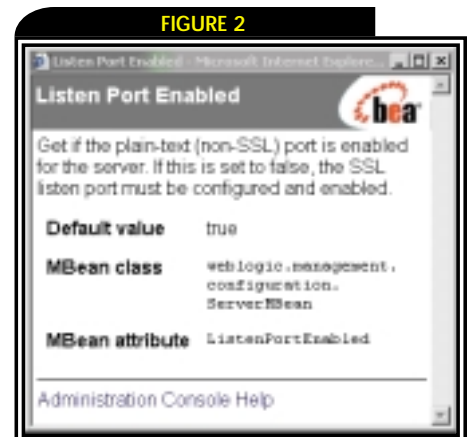


Console screen

ment of the WebLogic Server and any associated applications that depend on it. This SPI integrates several sources of information into one screen. It

- Gathers data from any supplied MBeans, such as those supplied by the WebLogic Server
- Performs calculations on that data to be more meaningful to the end operator
- Brings filtered application logging messages to bear on the management task

This provides a highly desirable management integration, as many IT operations departments want to be able to monitor the health of their computers, networks, application servers, and any applications that



Name and attributes of a particular MBean entry

designed by the application developer to provide new information through new metrics. This last approach applies in those situations where business-specific metrics require action by operations personnel. An example of this would be the number of customers who did or did not purchase items at my site in the past day (those who completed their Shopping-Cart checkout transaction, or failed to do so, perhaps because the system was not functioning adequately).

Look at the OVO Metrics Guide for the full set of OpenView-supplied metrics. These provide a comprehensive view of the internal operation of the WebLogic Server, adding the cross-correlation of items supplied by the base MBeans.

User-Defined Metrics in HP OpenView Operations

Application developers can, if necessary, add their own metrics using the features of the OVO console. By defining their own metrics, developers help the operator to monitor the applications by allowing new combinations of data to be derived from existing sources. These user-defined metrics are fully described in the HP OVO Administrator's Guide.

Writing Custom, Application-Specific MBeans

The JMX features previously described leave the software designer with the prob-

lem of deciding which application objects are in need of management visibility and which are not. Should all business objects be capable of being manipulated through their associated MBean, or just a chosen few?

The software designer has the option of mapping one MBean to one or to many business objects, depending on the need for management of them.

If one business object depends completely on another to get the work done (i.e., they are in a "uses" relationship), then there is an argument for managing them using one MBean. Correspondingly, if objects are entirely independent of each other in the business context, the designer can choose to have a management view of each one, using separate MBeans.

It is unlikely that all objects in any system will need a dedicated MBean of their own. The software designer chooses the Java object or EJB that he or she wishes to visualize and control using a management tool. The rules described above regarding conforming to certain interfaces or building a separate MBean object are then applied. Finally, the developer will register the new MBean with the appropriate MBean server process. In the case of the developer using WebLogic Server, that server is the BEA-supplied MBean server.

From that point onward, the MBean is manageable using JMX-aware tools. The

associated business object is manageable to the degree that its behavior is determined by its guardian MBean. Developers may find that the WebLogic-supplied MBeans are suitable for their management needs. However, there are situations where each object as an independent unit of management is not enough. For example, if I need to know how many users have abandoned their ShoppingCart objects and how many have not, then I may need to create an aggregation MBean that counts the number of relevant associations and allows this number to be seen at the management level.

Second, I may wish to have a listener MBean, for example, whose job it is to watch for users abandoning their ShoppingCart objects without taking them to the checkout phase. This is a custom MBean that I will need to register in my own application code with the MBean server and thus expose it to the tools.

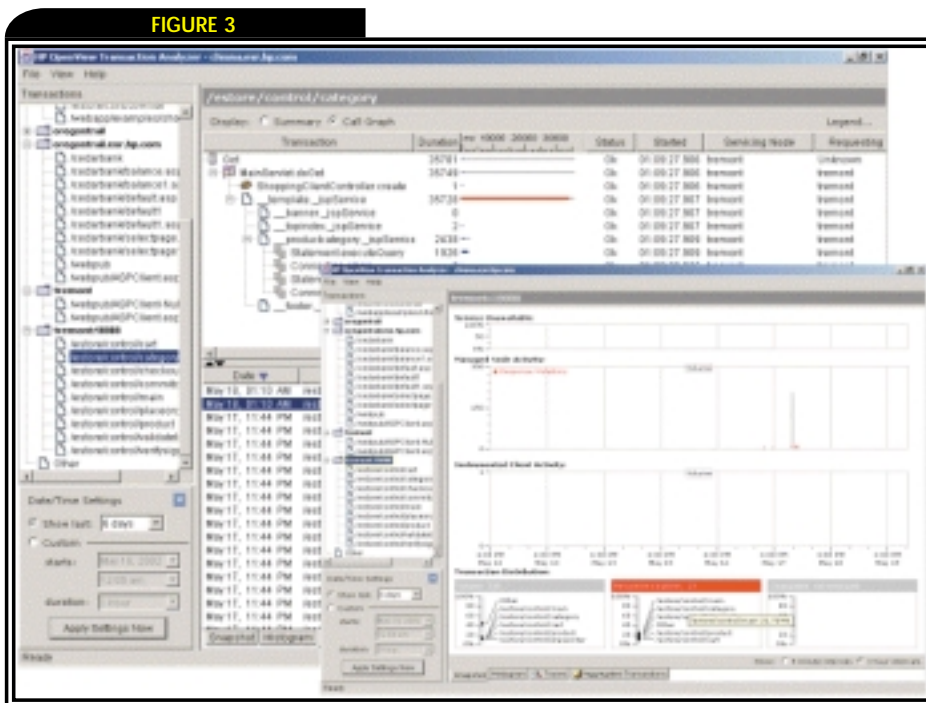
Summary

The HP OpenView Smart Plug-in technology provides management of the J2EE application server's internal objects without the application developer having to do extra coding work. Using the OpenView SPI to extend the metrics it supplies to the operator also gives us a powerful tool for manageability of our application.

Tracking in a J2EE Application with OpenView Transaction Analyzer

Many developers are concerned with finding the bottlenecks that inhibit high performance in their J2EE applications. This is not an easy task due to the distributed nature of these applications. Developers have found, in many cases, that instrumenting their application with calls that conform to the Application Response Measurement (ARM) API is more intrusive on their applications than they wish to be. This API is designed to allow for detailed accounting of CPU time spent in sections of the application. It is in use in some designs for close monitoring of the processor time spent in certain key areas of a program.

There are now tools on the market, such as the OpenView Transaction Analyzer (OVTA), that use ARM measurement and logging techniques without needing the application developer to make program changes. By exploiting purpose-built class loaders and byte code manipulation, these tools can intercept the entry and exit



OpenView Transaction Analyzer screens

Quest Software

<http://java.quest.com/qcj/wldj>

points from each method in an application, whether that method is in a JSP, servlet, or EJB, and report the time it consumes through ARM calls. The latter calls log data to a separate measurement server that collects the logs for subsequent processing and display without further interference on the application. The ARM API is therefore completely hidden from the application developer and is built into the tool. The degree of accuracy provided by ARM is maintained.

This is a very powerful method for analyzing the full J2EE application at development time, or even later at deployment time, for performance bottlenecks. The application developer has to make no changes to code to carry this process out. Figure 3 provides a snapshot of the type of screen that HP's OVTA product produces to aid the developer in performance analysis. It can trace quickly down to the point of the problem in a complex application.

In Figure 3, highlighted in red is the key culprit for performance concerns with this particular application. This tool fulfills the final requirement for the application developer in building manageability into an application, that of tracking the paths through the code.

Conclusion

This article discussed a number of technologies and standards for enabling the manageability aspects of a J2EE or Web services application. Different technologies have benefits based on the nature of the application they are managing and the environment in which they are set.

The lowest level is that of application logging. Applications that already have some logging built in can benefit from this approach without change. This provides a base level of monitoring in the manageability hierarchy.

For those applications that are built on Java and J2EE and adhere to the standards in those fields, JMX poses an attractive choice. JMX and SNMP-style management are not mutually exclusive – JMX agents can generate events that are SNMP compatible.

Many applications will require a combination of two or more of the technologies surveyed here to fully satisfy their operational management needs. The technologies are not competing with each other for the application programmer's attention. Instead, they should be regarded as a spectrum-of-choice of levels of sophistication in manageability.

Developers should always consider the functionality implemented by the management platform that will be used by the operations staff, before proceeding to instrument their application code with these technologies. Some management platforms provide many of the features required without resorting to extra coding work.

References

- *BEA JMX*: <http://edocs.bea.com/wls/docs70/javadocs/index.html>
- *JSR3*: www.jcp.org/en/jsr/detail?id=3
- *JSR77*: www.jcp.org/en/jsr/detail?id=77
- *OVO-Metrics*: HP OpenView Operations – Metrics Guide
- *OVO-Admin*: HP OpenView Operations – Administrators Guide
- *OVO-Concepts*: HP OpenView Operations – Concepts Guide
- *OVTA*: HP OpenView Transaction Analyzer: www.openview.hp.com/products/transaction_analyzer/index.asp
- *Poole*: HP OpenView Architectures for Managing Network Based Services

–continued from page 10

Scenario 1

In this scenario, an IT organization is tasked with consolidating three internally developed applications. Two of the applications were recently developed by the same team and are deployed on BEA WebLogic 7.0 running on Red Hat Linux 7.1. The third application was developed by an outside consulting firm (and is currently under a support contract with that firm) and is deployed on WebLogic 6.1 running on Red Hat 7.1. In this case, the best singular application isolation strategy would be to deploy each application in its own instance of WebLogic to ensure that there were no support conflicts with the outside vendor's application. However, the IT organization could cut out a significant amount of overhead by running the externally developed application in its own JVM and having the other two applications share a JVM.

Scenario 2

In the second scenario, an IT organization is tasked with consolidating four different WebLogic applications – a third-party ISV application that runs on top of WebLogic and three internally developed integration applications. The third-party ISV application is only supported under WebLogic 6.1 running on Windows 2000 while the two internally developed applications are currently running on WebLogic 7.0 on Red Hat Linux 7.2. At first glance, the only singular application isolation strategy that would work for all three applications would be a virtual machine implementation on Linux – resulting in each application running in its own virtual machine and the high overhead requirements associated with doing so. However, by combining a multiple JVM strategy with the virtual machine implementation, this overhead could be reduced significantly by running the third-party ISV application in its own virtual machine and then running three instances of WebLogic in a second virtual machine to accommodate the three internally developed applications.

Obviously, the number of possible combinations is almost endless and real-world consolidation scenarios are likely to be significantly more complex than the two that we have outlined here. However, in order to avoid long-term support issues, it probably makes sense to standardize on a particular subset of combinations that fit the majority of the applications deployed in a particular enterprise.

Summary

It is clear that many IT organizations can potentially achieve extensive cost savings and improved application reliability by consolidating their J2EE applications running on WebLogic onto a common hardware infrastructure. However, there are many complex issues to consider, and achieving the right balance between application isolation and resource utilization is a critical success factor.

For most large enterprises, there is probably no single application isolation strategy that will work in every situation. Simply put, different applications have varying isolation needs and any overall strategy must be flexible enough to accommodate the majority of those needs. The good news is that a platform as flexible as BEA WebLogic, combined with adaptive infrastructure solutions and services from HP, can make the promise of J2EE application consolidation a reality.

Interwoven

www.interwoven.com



WORKSHOP

Web application development is hard. Or rather, Web application development used to be hard.

Java Page Flow

WEB APPLICATION DEVELOPMENT MADE EASY

BY DOUG DEW

Web application development used to be an activity that required developers to learn and use complex programming models. Web application development used to be an activity that required developers to manage the myriad details of configuring their Web application so that the various pieces of the application worked together. Web application development used to be an activity that was performed outside of the helpful environment of an IDE.

No longer.

The new Java Page Flow feature set of WebLogic Workshop version 2 makes Web application development easy. It provides a simple, easy-to-understand Web application programming model. Java Page Flow automatically manages Web application configuration details. And it provides a set of tools that help developers quickly and correctly build Web applications, and integrate those applications with business logic.



AUTHOR BIO

Doug Dew is the Java Page Flow Program Manager at BEA Systems.

CONTACT...

doug.dew@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

What Is It?

Java Page Flow is a feature set built on a Struts-based Web application programming model. It leverages the power and extensibility of Struts while eliminating the difficulties and challenges of building Struts-based applications. Java Page Flow features include runtime support for the Web application programming model and tools that enable developers to quickly and easily build applications based upon the model.

The central concept and construct of Java Page Flow is called *page flow*. Basically, a page flow is a directory of Web app files that work together to implement a UI feature. For example, a page flow could implement a Web app's user registration wizard feature. The files of this page flow could be arranged in a userRegistration directory as shown in Figure 1.

The userRegistration directory contains several *.jsp files, a *.jcx file and a *.jpf file. The *.jsp files are standard JavaServer Pages files that contain markup that describes the visual aspect of the user registration wizard. For example, name.jsp contains markup that describes a First Name and Last Name data entry form. The *.jcx file is a BEA innovation. It contains annotated Java code that implements logic used by the user registration wizard. UserManager.jcx contains code that implements a createUser() function.

The *.jpf file is also a BEA innovation and is the main focus of this article. It contains annotated Java code that implements the navigation and state management logic of the user registration wizard and makes calls to business logic. For example, UserRegistrationController.jpf contains code that decides that name.jsp should be presented before address.jsp, gathers the firstName and lastName information from name.jsp before presenting address.jsp, and calls the createUser() function of UserManager.jcx.

For those of you familiar with Struts and Model-View-Controller (MVC) programming, the files of the user registration wizard may be mapped to the 'M', the 'V', and the 'C' of MVC as seen in Table 1.

Page Flow Controllers

Page flow controllers are the main focus of this article. Let's take a step-by-step look at the contents of the UserRegistrationController.jpf file and how the file performs its navigation, state management, and logic-calling functions.

First, UserRegistrationController.jpf contains a class definition whose skeleton looks like this:

```
package userRegistration;

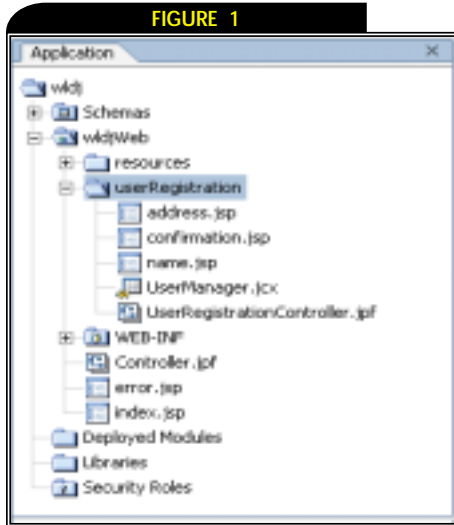
import com.bea.wlw.netui.pageflow.PageFlowController;

public class UserRegistrationController
extends PageFlowController
{
...
}
```

TABLE 1

Model (M)	UserManager.jcx
View (V)	name.jsp address.jsp confirmation.jsp
Controller (C)	UserRegistrationController.jpf

MVC file mapping



Page flow files in a userRegistration directory

UserRegistrationController extends PageFlowController, which is a class that provides useful base class functionality and derives indirectly from org.apache.struts.action.Action. The base class functionality of PageFlowController provides support for things such as login and logout. The derivation from Action is an important aspect of the Struts interoperability of PageFlowController. Many classes in Java Page Flow inherit from Struts classes and interoperate with the Struts plumbing that serves as the foundation for the Java Page Flow runtime.

Actions and Navigation

Since UserRegistrationController acts as the controller for the user registration wizard, it contains code, in the form of action methods, that performs the page navigation decisions for the wizard. Action methods are methods invoked in response to things like form submissions. For example, UserRegistrationController contains an action method named name_next (see Listing 1).

The name_next action method is invoked whenever a user submits the form of the name.jsp page (see Listing 2).

The name_next action method decides that the address.jsp page should be the next page presented to a user. Because the name_next action method is invoked whenever a user submits the form of the name.jsp page, the name_next method effectively causes navigation from the name.jsp page to the address.jsp page.

All action methods have similar signatures. For example, they all have Forward as their return type. All action methods are specially annotated with @jpf:action to

indicate to the *.jpf compiler that they should be configured as action methods in the auto-generated Struts configuration files. Also, action methods may be annotated with @jpf:forward to indicate to the *.jpf compiler and to the IDE tools the possible navigation decisions that the action methods may make such as deciding to forward to a page like address.jsp. Action methods are called by the page flow runtime and express their navigation decisions to the runtime by returning objects of type Forward to the runtime. The Forward objects encapsulate information described by the @jpf:forward annotations on the action methods.

The page flow runtime is responsible for selecting and calling action methods in controllers as part of the runtime's request processing life cycle. It is also responsible for executing the navigation decisions made by the action methods. In other words, if a user submits the form of address.jsp, the page flow runtime performs a request processing life cycle that includes these steps:

1. Intercept the submission when it arrives at the application server.
2. Determine that the name_next action method should be called.
3. Call the name_next method.
4. Receive the Forward("getAddress") object from the name_next method.
5. Forward it to the address.jsp page described by the Forward("getAddress") object.

Page flow controllers may contain any number of action methods. For example, UserRegistrationController contains five action methods in addition to the name_next action method (see Listing 3).

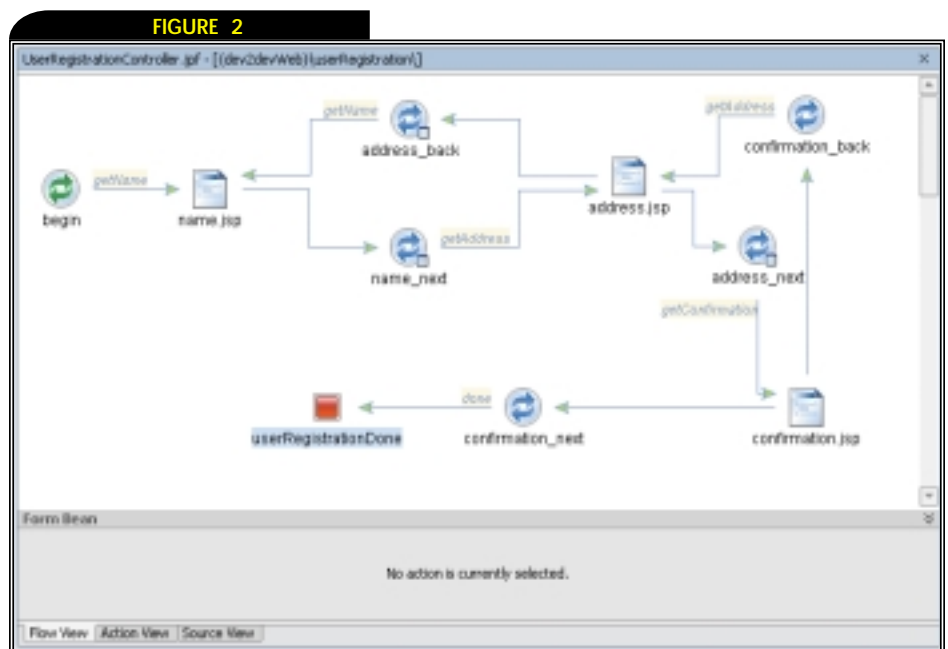
Forms

In addition to selecting and calling action methods, the page flow runtime also passes to the action methods any form data that may have been submitted. For example, when a user submits the form of the name.jsp page, the page flow runtime captures the firstName and lastName data of the form in properties of a NameForm object, and passes the populated NameForm object to the name_next action method: package userRegistration (see Listing 4).

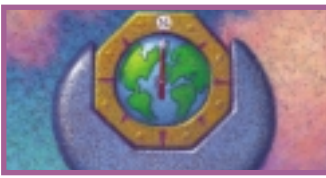
State Management

Java Page Flow provides a powerful, yet easy-to-use, foundation for state management that makes it simple for UserRegistrationController to perform its state management responsibilities. For example, to manage the form data that is received from pages such as name.jsp, code may be added to UserRegistrationController that captures the data in convenient instance members (see Listing 5).

Since the page flow runtime makes it possible to cache state in controller instance members, it's easy to implement support for a back button in a wizard. For example, if the confirmation.jsp page con-



New flow view



tained a back button, UserRegistration-Controller could respond to a press of the back button by navigating backward to the address.jsp page with all previously filled-out address.jsp form data still intact (see Listing 6; Listings 6–8 can be found online at www.sys-con.com/weblogic/source.cfm).

As you can see, the confirmation_back action method forwards backwards to the address.jsp page. As part of the forward, confirmation_back passes to address.jsp the address data that had already been received from address.jsp as an argument to the address_next action method.

Calling Business Logic

BEA WebLogic Workshop makes it easy to expose business logic as controls, and the Java Page Flow feature makes it easy to call controls from action methods. Because the page flow runtime performs automatic instantiation and initialization of controls for all instance members that are annotated with @common:control, action methods

in page flow controllers may call Workshop controls without having to include code that performs control instantiation and initialization. For example, the confirmation_next action method can simply and directly call the createUser() method of UserManager.jcx: package userRegistration (see Listing 7).

BEA WebLogic Workshop makes it easy to expose all kinds of business logic as controls, including business logic that is implemented as EJBs. In other words, WebLogic Workshop even makes it easy to build Web applications over EJBs.

Additional Features

Java Page Flow contains many features in addition to those described above. For example, you may have noticed that the @jpf:forward annotation for the confirmation_next action method was a bit different from the other @jpf:forward annotations (see Listing 8).

The returnAction="userRegistration-


Done" attribute enables UserRegistration-Controller to participate in a Java Page Flow feature called "nesting."

Tools

As I mentioned earlier, the Java Page Flow feature set includes some IDE tools that help developers build Web applications. One example of such a tool is the new flow view that enables developers to architect entire Web applications using little more than simple drag -and- drop gestures (see Figure 2).

Like nesting, data binding, and custom tags, we'll take a look at the IDE tools in a future *WLDJ* article.

Take It for a Test Spin

For now, the best thing you can do is tdownload BEA WebLogic Workshop v2 and experiment with the new Java Page Flow features. See for yourself just how much easier Web application development can be. 

Listing 1

```
package userRegistration;

import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;

public class UserRegistrationController
extends PageFlowController
{
    /**
     * @jpf:action
     * @jpf:forward name="getAddress"
     *               path="address.jsp"
     */
    public Forward name_next( ... )
    {
        ...

        return new Forward("getAddress");
    }

    ...
}
```

Listing 2

```
<%@ taglib uri="netui-tags-html.tld" prefix="netui"%>

<html>
<head>
<title>Name Page</title>
</head>
<body>
<netui:form action="name_next" focus="">
<table>
<tr>
<th align="right" valign="top">First Name:</th>
<td align="left">
<netui:textBox
dataSource="{actionForm.firstName}"
size="16"
maxLength="18"/>
</td>
</tr>
<tr>
<th align="right" valign="top">Last Name:</th>
<td align="left">
<netui:textBox
```

```
dataSource="{actionForm.lastName}"
size="16"
maxLength="18"/>
</td>
</tr>
<tr>
<td align="right"></td>
<td align="left">
<netui:button value="name_next"/>
</td>
</tr>
</table>
</netui:form>
</body>
</html>
```

Listing 3

```
package userRegistration;

import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;

public class UserRegistrationController
extends PageFlowController
{
    ...

    /**
     * @jpf:action
     * @jpf:forward name="getName"
     *               path="name.jsp"
     */
    public Forward begin( ... )
    {
        ...

        return new Forward("getName");
    }

    /**
     * @jpf:action
     * @jpf:forward name="getAddress"
     *               path="address.jsp"
     */
    public Forward name_next( ... )
    {
        ...
```

LinuxWorld Conference & Expo

www.linuxworldexpo.com

```

return new Forward("getAddress");
}

/**
 * @jpf:action
 * @jpf:forward name="getConfirmation"
 *             path="confirmation.jsp"
 */
public Forward address_next( ... )
{
...

return new Forward("getConfirmation");
}

/**
 * @jpf:action
 * @jpf:forward name="done"
 *             returnAction="userRegistrationDone"
 */
public Forward confirmation_next( ... )
{
...

return new Forward("done");
}

/**
 * @jpf:action
 * @jpf:forward name="getName"
 *             path="name.jsp"
 */
public Forward address_back( ... )
{
...

return new Forward("getName");
}

/**
 * @jpf:action
 * @jpf:forward name="getAddress"
 *             path="address.jsp"
 */
public Forward confirmation_back( ... )
{
...

return new Forward("getAddress");
}
}

```

Listing 4

```

import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;

public class UserRegistrationController
extends PageFlowController
{
...

/**
 * @jpf:action
 * @jpf:forward name="getAddress"
 *             path="address.jsp"
 */
public Forward name_next( NameForm form )
{
...

return new Forward("getAddress");
}

public static class NameForm
extends FormData
{
private String firstName;
private String lastName;

public String getFirstName()
{
return firstName;
}

public void setFirstName(String firstName)
{
this.firstName = firstName;
}

public String getLastName()
{

```

```

return lastName;
}

public void setLastName(String lastName)
{
this.lastName = lastName;
}

...
}

```

Listing 5

```

package userRegistration;

import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;

public class UserRegistrationController
extends PageFlowController
{
...

public String firstName;
public String lastName;

/**
 * @jpf:action
 * @jpf:forward name="getAddress"
 *             path="address.jsp"
 */
public Forward name_next( NameForm form )
{
firstName = form.getFirstName();
lastName = form.getLastName();

return new Forward("getAddress");
}

public static class NameForm
extends FormData
{
private String firstName;
private String lastName;

public String getFirstName()
{
return firstName;
}

public void setFirstName(String firstName)
{
this.firstName = firstName;
}

public String getLastName()
{
return lastName;
}

public void setLastName(String lastName)
{
this.lastName = lastName;
}

...
}

```


Wily Technology

www.wilytech.com



FROM THE OFFICE OF THE CTO

The Web is all about people using computers (Web browsers) to talk to other computers (Web servers). Web services are about computers talking to computers without a human at the helm.

Making Sense of Web Services Standards

CURRENT AND UPCOMING STANDARDS IMPROVE INTEROPERABILITY

BY DAVID ORCHARD

It turns out that humans are pretty darned smart. We can figure out a variety of interactions such as:

- How to enter usernames and passwords into different pages
- When to stop waiting for a page to return and resubmit the request
- When timed out of a site, how to reenter data and get to the right pages
- How to enter an e-mail address in a Web form, and then switch over to the e-mail client to view the resulting e-mail message
- What ordering (or flow) of Web pages to go through to achieve a task

Computers, when talking amongst themselves, aren't nearly as smart as this. Things that people deal with naturally – such as security, reliability, sessions, or asynchrony – have to be explicitly programmed into computers. This is why there are so many “WS-*” specifications; computers require a formal specification to fully explain each of the things that we can do naturally.

For Web services to be successful and cost-effective, we need a single standard in each of these areas. It's okay for each Web site to require a username/password combination (as well as different methods of entering them) because people can figure this out very quickly. But imagine if each Web service used a different mechanism for authentication; the job of writing software to deal with each service's different mechanism would be enormous. Standards are all about achieving

economies of scale so that vendors can provide the same functionality in the same way. Then, developers can more easily write software to offer or consume Web services.

This article describes the set of Web services standards that BEA believes are important to customers and developers. We classify them by three criteria: the type of problem they address, the readiness of the specification for real-world use, and how fundamental the functionality of the specification is to the Web services stack.

The Web Services Standards Stack

In any discussion of technology, a diagram must inevitably be provided. Figure 1 shows the Web services specifications addressed in this article, their types, and their relationship to what we call the “core” set of Web services specifications.

Classification Methodology

We've chosen two different axes for classifying Web services specifications. The first describes the functionality of the specification according to the “3-Ds” of Web services; delivery, description, and discovery. Delivery is about how a message is sent between services; description specifications describe the message(s) that are exchanged; discovery is how the services or descriptions are found.

The second axis is the preeminence and timing of the specification. There is a set of specifications that are core to Web services; we anticipate that most Web services will use them. For example, SOAP and WSDL are obvious core specifications. Some core specifications are available now; others will be available in the future. Specifications that are useful for some Web services, but will probably not be integral to most, are considered extensions.

Core Standards

I won't go into the details of the current stable specifications, as most people have heard of SOAP, WSDL, and UDDI by now. In short, the current core consists of XML 1.0, XML Schema 1.0, SOAP 1.1, WSDL 1.1, HTTP 1.1, SSL, and TLS. It should be no surprise that many of these are also specifications in the WS-I Basic Profile.

Emerging Core

We can already see the form the next generation of core Web services will take. Delivery will be expanded to include more robust asynchronous messaging, clearer standards for addressing messages, a full-fledged Web service layer mes-

AUTHOR BIO

David Orchard is technical director in BEA Systems' CTO Office, focusing on Web services standards. He is a member of the W3C Technical Architecture Group, Web Services Architecture, XML Protocol, and Advisory committees. He has written numerous technical articles and is a frequent speaker on various Internet-related technologies.

CONTACT...

david.orchard@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.



sage routing system, and more advanced security facilities. Description will be enhanced by the addition of sophisticated policy description capabilities. And discovery will be enhanced with facilities to allow Web services to be dynamically queried for a description of their capabilities. The addition of asynchrony, security, reliability and addressing information into delivery, WS-Policy into description, and URI-based metadata exchange into discovery rounds out what BEA considers to be the next level of core Web service specifications.

Delivery: Asynchrony, Addressing, and Routing

One of the promising aspects of Web services is asynchrony; that is, the ability to deliver responses and messages without an existing connection, so that a service consumer's and provider's execution is decoupled. WS-Addressing defines basic mechanisms for asynchronous messaging; in particular, the Reply-To header allows a "callback" to be performed.

WS-Addressing also specifies how addressing information (typically used for routing) should be conveyed in a SOAP message, by defining common headers such as message identifiers and to and from fields.

Finally, WS-Addressing provides an *EndpointReference* structure that can contain information about Web service endpoints. An endpoint, which is defined in WSDL 1.2 and is renamed from WSDL 1.1 Port, is an access point of a Web service. It can contain information that is required to access or reference the service, such as a

conversation ID. It is often necessary to exchange endpoint references, particularly for callbacks and long-running conversations. In the future, there will be a WS-EndpointResolution specification that will enable two parties to negotiate or refine EndpointReferences.

Delivery: Reliability

Reliable messaging protocols define mechanisms for ensuring that the sending and receiving parties know whether or not a message was delivered. Typically there is an acknowledgement message with a retry algorithm. WS-ReliableMessaging specifies a reliable messaging protocol for Web services using a number of different SOAP headers as well as configuration information. These SOAP headers allow communication of a sequence number, acknowledgment of a sequence of messages, and the ability to request acknowledgement of a sequence of messages. There is an OASIS Committee, WS-Reliability, which is looking at a similar work – WS-Reliability – from Fujitsu, Sun, Oracle, Sonic, and others.

It is worthwhile to note that BEA has published and implemented a number of individual specifications in this area: WS-Acknowledgement, WS-MessageData, WS-Callback, and SOAP-Conversation. This has been done to allow others to interoperate with BEA, to deliver early functionality to our customers, and to provide useful implementation experience. We anticipate our experience and implementation will be helpful in making the specifications above into stable, full-featured standards.

Delivery: Security

WS-Security is composed of a specification at the OASIS Web Service Security technical committee and a roadmap of related specifications. The WS-Security (WSS) specification has three main functions:

- **Message integrity/authentication:** Guaranteeing that the message was not modified since it left a known person
- **Message confidentiality:** Guaranteeing that only the right people can look at the message
- **Message authorization:** Providing the information needed to decide if a request should be granted

The WSS specification uses the XML Digital Signatures specification and the XML Encryption specification, and introduces new authorization elements. All of these WSS features are expressed as SOAP header blocks. A number of vendors have implemented WS-Security, and many more will be involved in WSS interoperability testing.

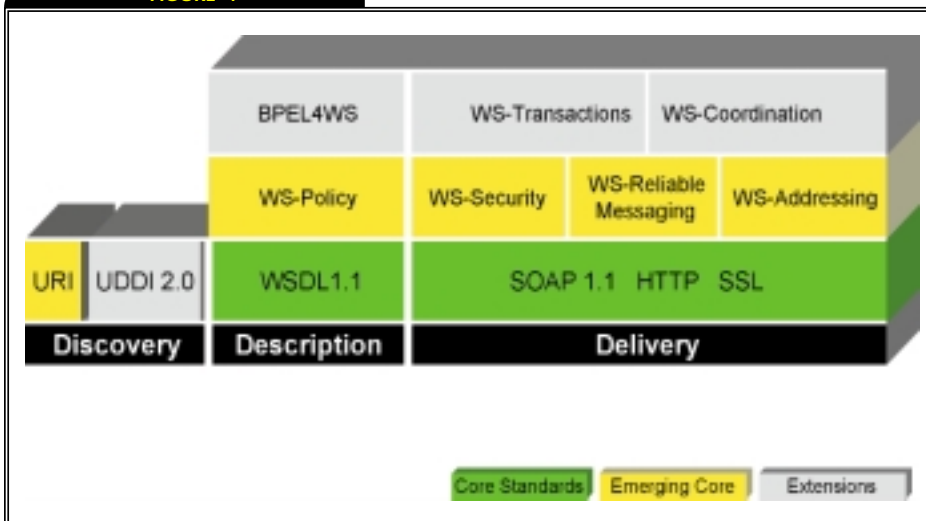
The WS-Security roadmap contains three other recently published specifications. WS-SecurityPolicy defines how policies for WS-Security are expressed in the WS-Policy language (see below). WS-Trust provides a protocol for requesting and receiving WS-Security tokens, including a challenge response protocol. WS-SecureConversation defines a security context that can be shared during a long-running conversation between two parties, and a refinement of WS-Trust for setting up the context.

Description: Policy

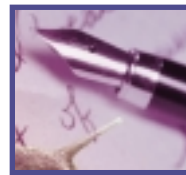
An emerging problem in Web services is describing the capabilities and requirements of endpoints. For example, WS-Security describes a number of security mechanisms but doesn't force services to use any particular one. Likewise, there are several configuration options in WS-ReliableMessaging that allow services and their consumers to fine-tune the nature of their message exchange. Currently, these and many other aspects of a Web service's operation must be done by private negotiation because WSDL isn't up to the task.

While all of these things could be specified and negotiated on a point-by-point basis, it's sensible to define a common framework for configuration, requirements, and capabilities – together referred to as Policy – in Web services. WS-Policy aims to fill this gap with three related specifications.

FIGURE 1



Web services specifications stack



THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS! CHOOSE ONE – OR TRY THEM ALL!

- JAVA Industry Newsletter
- WebServices Industry Newsletter
- JOURNAL Industry Newsletter
- wireless Industry Newsletter
- WebLogic Industry Newsletter
- LINUX WEEK Industry Newsletter
- GOLD FUSION Industry Newsletter
- .NET Journal Industry Newsletter

FREE

E-Newsletters SIGN UP TODAY!

Go to www.SYS-CON.com

The most innovative products, new releases, interviews, industry developments, and plenty of solid *i*-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at www.sys-con.com.

Exclusively from the World's Leading *i*-Technology Publisher

Don't Delay! Subscribe for FREE!

at www.sys-con.com



WS-PolicyFramework describes an overall approach to expressing policy assertions (e.g., "I support the DES encryption algorithm," "That message is required to be SOAP 1.2") and combining them (e.g., requiring one out of a list of values, or requiring all members of a set).

WS-PolicyAssertions defines some specific assertions – like a document's encoding, or support for a particular specification – to demonstrate the use of the framework, as well as allow for reuse of common components. Finally, WS-PolicyAttachment specifies how to associate policy assertions with XML elements, WSDL-type definitions, and UDDI entries.

Discovery: URI-Based Discovery

There is one more item that Web services needs for the core, which is how to discover WSDL and WS-Policy statements about a particular Web service or service provider, given a URI. We expect that a WS-MetadataExchange specification will fill the role of UR-based discovery.

Extensions

Extensions are specifications that are very useful, but in more of a refined area than what we have called core.

WS-Transaction defines coordination of a variety of atomic transactions – two-phase commit and four other types – and compensating transactions. It uses WS-Coordination as a pattern for clients and coordinators to register and exchange coordination messages. WS-Transaction and WS-Coordination involve one or more coordinators that are involved in completing the interaction.

WSBPPEL (Web Services Business Process Execution Language) is an OASIS technical committee that is chartered with producing a platform-independent, XML-based programming language for writing Web service control logic. The W3C Choreography Working Group is working on similar technologies but at the time of this writing it was too early to tell if the groups will be complementary.

"We can already see the form the next generation of core Web services will take"

Conclusion

The set of specifications described here is expected to provide a solid foundation for meeting our customer's application and business-process needs. They are gradually making their way toward becoming standards, starting with SOAP, WSDL, and UDDI, and more recently followed by WS-Security and BPEL4WS. We expect that all of these specifications will be standardized.

Given that you are BEA WebLogic developers, I'll tie these specifications back to BEA and BEA's leadership role. BEA is a coauthor of most of the WS-* specifications. BEA is also an aggressive implementer of the specifications in WebLogic products. For example, BEA WebLogic Platform 8.1 implements a very useful portion of the WS-Security specification, even though the WS-Security TC has not yet held formal interop tests.

In summary, developers can expect to build interoperable Web services using these specifications in current and upcoming WebLogic releases. ●

International Web Services Conference & Expo

WEST

Web Services Edge 2003

SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



web services **EDGE**
conference & expo

**EXTENDING THE ENTERPRISE
WITH WEB SERVICES THROUGH JAVA,
Mac OS X AND XML TECHNOLOGIES**



XML

WebLogic




LINUX EDGE
conference & expo

web services EDGE
conference & expo

BOSTON
February 24-27, 2004

Featured technologies and topics will include:

- Focus on Java
- Focus on Web Services
- Focus on WebLogic
- Focus on Mac OS X
- Focus on XML

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



Over 100 participating companies will display and demonstrate over 300 developer products and solutions. Over 2,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo. Over 60 of the latest sessions on training, certifications, seminars, case studies, and panel discussions will deliver REAL World benefits, the industry pulse and proven strategies.

Contact information: U.S. Events: 201 802-3069 or e-mail grisha@sys-con.com

WebServices JOURNAL

JAVA DEVELOPERS JOURNAL

LINUX BUSINESS TECHNOLOGY

XML JOURNAL

SAMS

WebLogic

wireless BUSINESS TECHNOLOGY

CF Advisor

ColdFusion DEVELOPERS JOURNAL

PowerBuilder DEVELOPERS JOURNAL

Java is a registered trademark of Sun Microsystems, Mac OS X is a registered trademark of Apple Computer, Inc., All other product names herein are the properties of their respective companies.

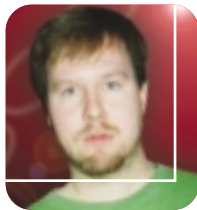
**WEB SERVICES EDGE WEST 2003
CALL FOR PAPERS NOW OPEN**
Submit your papers online at:
www.sys-con.com/webservices2003west

PRODUCED BY
sys-con
EVENTS

Wily Introscope 4.0 from Wily Technology

VALUE ACROSS THE DEVELOPMENT LIFE CYCLE

Reviewed by Jason Snyder



Application performance horror stories exist everywhere. Tales of molasses-like response times, high-risk transactions that periodically and mysteriously slow to a crawl, and search engine combinations that paralyze the browser are common enough that interest in application performance management solutions is at an all-time high.

I recently had the good fortune to participate in a review presentation of Wily Technology's Introscope 4.0, which can help developers determine the causes of performance bottlenecks.

Response times can be analyzed in detail, under a variety of conditions. Within the JVM, class and method-level detail statistics can be gathered and reviewed. There is no place for performance problems to hide.

Introscope 4.0 can also be used for production monitoring, thanks to its low overhead. No longer will a company be blindsided with client complaints about performance after the fact. Outages can be identified and responded to immediately. User activity and usage patterns can be monitored as they happen.

The product also allows for preproduction performance QA monitoring. Misconfigured servers or third-party products can be identified prior to production panic.

Introscope has three distinct functional areas: Agent, Enterprise Manager, and Workstation. Agent provides the ability to collect information from any component within the application or operating system. Enterprise Manager allows various agents to be tied together. The Workstation provides a means to view this information.

Agent

Agents provide the base data to be monitored. Metrics they collect can be organized into Metric Groupings and stored in Management Modules, which allow any number of Agents, each with any number of attributes, to be grouped together to create more comprehensive views of performance, such as cluster performance. Introscope comes with several Metric Groupings predefined. Creating a Metric Grouping is relatively easy, and allows the greatest degree of flexibility for monitoring.

Wily provides add-ons to the base product, such as an SNMP Agent that enables publishing of any performance data through an SNMP MIB. Another is the SQL Agent, which provides views into the database. Individual SQL statements can be viewed, allowing greater flexibility when tuning the database or identifying problem queries.

To set up an Agent, the developer must identify the data gathering frequency, the type of metric required (response times, counts, rates), and stalled/threshold values before erroring. An element (calculator) exists to programmatically perform mathematical computations on these values to create new metrics. One minor point of confusion is that these values use Perl's scripting syntax, which may not be known by the developer. Although it is easy to learn, making this area more graphical may provide some benefit to less experienced developers.

Once the metrics are identified, resulting actions like Workstation notification or launching a shell script can be assigned to any data values. Multiple actions are acceptable, and the ability to launch a shell script provides a means for kicking off almost any resulting program.

Enterprise Manager

The Enterprise Manager collects the information provided by the various agents. It allows multiple agents to be tied together into a cohesive data detail. Multiple applications or servers can be grouped into one central location, allowing for greater ease of monitoring. A central monitoring repository can be created, providing the ability to achieve the holy grail of production monitoring – a one-page status of all applications.

Wily Technology, Inc.
8000 Marina Boulevard,
Suite 700
Brisbane, California 94005

1 888 GET WILY
< U.S. toll free >
+1 415 562 2000 < direct >
+1 415 562 2100 < fax >

Sales:
sales@wilytech.com

Information:
info@wilytech.com

Workstation (Console)

Application dashboards containing metrics collected by the agents can be created with the editor portion of the Workstation.

The developer can create any series of views of the information (see Figures 1–3). The editor provides the ability for role-specific workstations so that the help desk can have one view while a performance engineer has another. Developers can create dashboards with bar charts, graphs, traffic lights, string views, and graphic imports to create a fully functional and professional application. Agent information can be stored in a database, and a historical view of this information can also be provided. Historical information can be drilled down for greater detail.

A Transaction Tracer can be set up to provide tracking by user or URL. This feature provides a sequence diagram-like view of the transaction, with detailed performance statistics for each call throughout the transaction. This allows detailed investigation in a valuable user interface for long-running transactions. It is also an easy means of showing specifically what is going wrong for any transaction.

Cross-server transaction traces are currently not supported, but a future enhancement providing this level of detail is expected. Additional traces beyond the current server can be created to allow capturing this information but it would not be within the same trace.

Power Packs provide out-of-the-box, vendor-specific extensions for monitoring. The WebLogic Power Pack provides access to key WebLogic Admin monitoring values. The EJB pool, thread queue, and JDBC connection pool can be reviewed for accurate problem diagnosis. Web services, values

within the HTTP session, WebLogic JMS queues, or topics can also be explored. In the near future, Power Packs will be available for WebLogic Integration and WebLogic Portal.

Leak Hunter, which is an extension to the core product, identifies common memory leaks under production load. The current implementation of this concept focuses on misuse of collection classes like dangling references.

The product currently requires a user to restart the server first installing Agents, which may be inconvenient in some production environments.

Another great feature is the EPA – Environment Performance Agent – a script scheduler that provides interfaces into agent metrics. This provides the ability to extend monitoring beyond Java through scripts for Apache Web Server, IIS, Solaris Kernel/Disk, and HTTP URLs for example.

Workstation (Explorer)

The Workstation Explorer provides a tree-structure view of the performance data being reported by each and every Introscope Agent and allows user manipulation of that information."

The Workstation provides the ability to access EJBs, JDBC, JMS, JMX, JSP, JNDI, and JTA information among others. All Agents can be accessed for viewing as well. The Workstation also provides the ability to quickly view cross-application top 10 usage, consumers of CPU, and slow performers; and lets you know the greatest problems instantly with a single click. A unique feature called Blame offers the ability to identify performance problems to the method level, which allows an ease of accurate problem detection.

The Workstation also pro-

vides the ability for conditional activity notification to occur via e-mail, pop-up, or preexisting exception handlers. In addition, performance metrics can be grouped in a variety of ways. This allows for high-level views of an application or key parts of the application, which can then be presented in the form of an overall application status.

The current version is not browser-based, but Wily hopes to provide Workstation access through these standard means in the future. Another feature that would be helpful would be

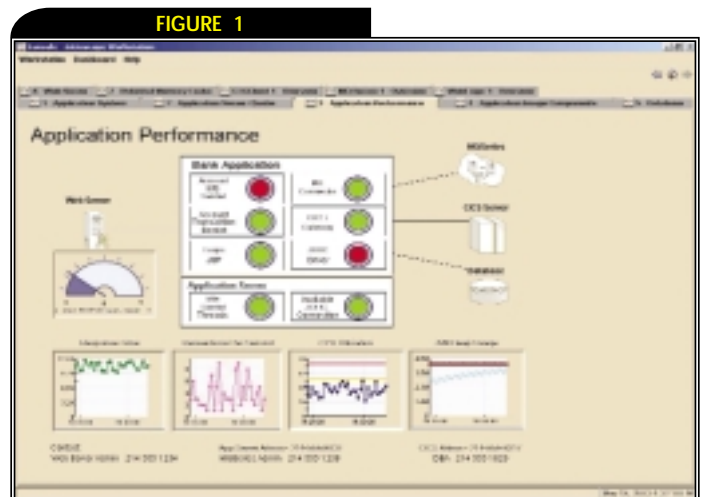


Figure 1 Console view showing a customizable dashboard for monitoring a application/WebLogic environment

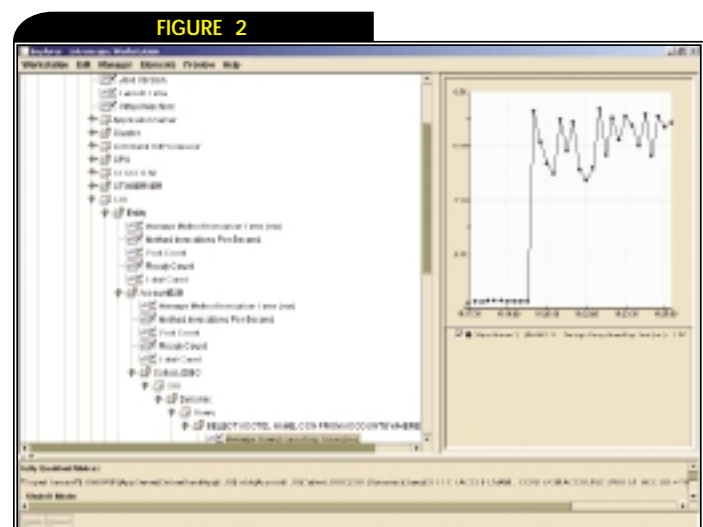
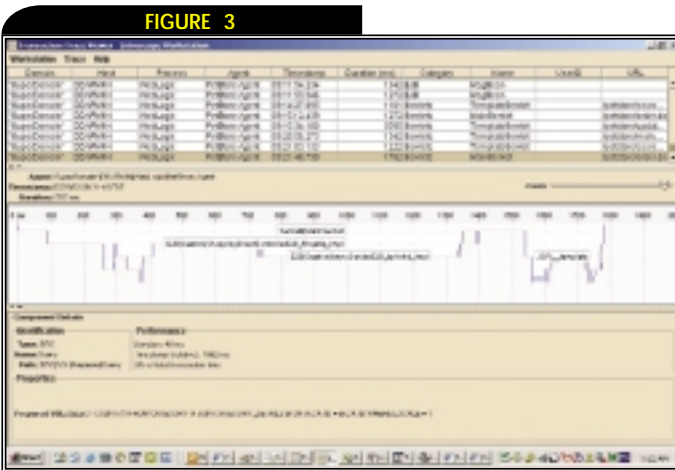


Figure 2 Explorer view showing the types of performance data collected and drill-down capability using Introscope's Blame feature

the ability to drill down through application modules to their component details. For example, if an application showed critical status on a high-level summary, the user must select the appropriate



Transaction Tracer view showing how Introscope can monitor slow user transactions and provide complete details on an individual transaction


application tab to see its causes, versus double clicking on the application module itself.

Setup

Setting up Introscope for existing applications is fairly straightforward. The product is *un-tar'ed* into a specific directory, the Start WLS script is updated with ClassPath and Property information, and the product is ready to go.

Conclusion

Wily's Introscope 4.0 provides additional functionality to the application performance management market. Key features include deep visibility into the JVM/Application/Operating System, a detailed end-to-end transaction view, and low performance overhead, allowing continuous production monitoring. The Workstation is intuitively built for displaying agent metrics. Being a pure Java solution,

the product can run in any environment. The ease of integration with existing applications is also a benefit in that it does not require an application overhaul for use. Power Packs exist for specializing the performance measurement to your environment. The greatest benefit the product offers, in my opinion, is its ability to provide value across the software development life cycle, making it unique for vendor solutions in the application performance arena. 

AUTHOR BIO

Jason Snyder is an architectural expert for CSC Consulting in Boston, and has served as the lead architect for several J2EE development projects. He has over 10 years of experience in software development, OO design, and application architecture.

CONTACT...

jasonsnyder@townisp.com

The World's Leading Java Resource!

JAVA DEVELOPERS JOURNAL

Here's what you'll find in every issue of *JDJ*:

- Industry insights
- The latest software trends
- Technical expertise
- Career opportunities
- In-depth articles on Java technologies

Subscribe Today & **SAVE 30% Off** the annual cover price

ANNUAL COVER PRICE ~~\$71.88~~
 YOU PAY **\$49.99**
 YOU SAVE **30%** Off the annual cover price

Sign up **ONLINE** at www.javadevelopersjournal.com

Premiering
August
2003



www.LinuxWorld.com

The Leading Magazine for Corporate and IT Managers

LinuxWorld Magazine

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

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

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

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

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

*Regular features
will include:*

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

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY

\$49⁹⁹

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR

VISIT WWW.SYS-CON.COM



The World's Leading i-Technology Publisher

LINUXWORLD® IS THE REGISTERED TRADEMARK OF INTERNATIONAL DATA GROUP, INC.
SYS-CON IS USING THE MARK PURSUANT TO A LICENSE AGREEMENT FROM IDG
ALL BRAND AND PRODUCT NAMES USED ON THIS PAGE ARE TRADE NAMES, SERVICE MARKS, OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.



REUSE

Component-based development (CBD) has been around for years. Anyone who has been a developer for any length of time has no doubt leveraged some form of reuse in an application.

Enhanced Component-Based Development

COMPONENT INTEGRATION IS A BREEZE

BY CHRIS BROOKE



AUTHOR BIO

Chris Brooke is director of Enterprise Technology for ComponentSource, a software asset value provider and global award-winning marketplace for reusable software assets for all platforms. He has contributed to both books and periodicals, and speaks regularly on reuse methodologies and technologies at conferences around the country.

CONTACT...

chrisb@componentsource.com

It may have been as simple as using a Swing component on the form of a client application. As with many areas of technology, software development and the evolution of developer tools is marked not so much by huge, dramatic leaps forward, but rather by constant innovation and improvement. When Sun Microsystems first introduced the Enterprise JavaBean (EJB) specification in 1997, no one thought that managing persistence would be such a big deal. But a big deal it was. Sun listened, and introduced Container-Managed Persistence (CMP) in the EJB 2.0 specification.

More recently, BEA has listened to its customers and provided an integrated development environment for the WebLogic platform that goes a long way toward facilitating effective CBD: BEA WebLogic Workshop. Over the course of this article, I'm going to talk about some of the challenges you face when utilizing components in Enterprise Java development and how the process can be simplified by utilizing the BEA control framework.

Planning

The time to think about how to leverage reuse in your application is at the project-planning phase. As with most aspects of software development, proper planning is essential to success. Once the program requirements are established, the first thing you should do is look in your repository and identify component candidates

that can possibly be leveraged. In addition to providing the required functionality, the component must also meet the technical requirements of your application. As you go through the planning stages, be prepared with the answers to some very specific questions: Is the application intended to be Web based? Will a JavaServer Pages application in conjunction with servlets be adequate, or should you also include Web services for scalability and cross-platform flexibility? Are you building a distributed, thin-client app? Will this need the scalability and/or ease of deployment/updating provided by EJBs?

Assuming that your component repository is well documented, you should have no trouble identifying which components will be useful once these questions are answered and the program requirements are established. In many cases, the amount of "glue code" needed to integrate the component into the new project will be minimal. In other cases, there may be a fair amount of work involved. If that work estimate exceeds the amount of time it took to create the component originally, you won't see any reuse cost avoidance and you'll need to either look for another component or create one from scratch.

Another part of the planning process is determining what development environment you will use to create the individual pieces of your application. You have some flexibility here. For instance, when building an *n*-tier distributed app, you will probably use an IDE such as JBuilder for the client portion. You might use JBuilder for the server-side development as well, or you might decide to go with WebLogic Workshop.

WebLogic Workshop and the BEA Control Framework

BEA WebLogic Server is a powerful platform for Enterprise Java development. BEA WebLogic Workshop is tightly integrated to provide an integrated development and testing environment for Web services (and, soon, Web applications) for the WebLogic platform. As such, it is important for you to consider the added benefits that the BEA control framework provides.

For example, let's assume that as a result of your project planning, you have identified several EJB components that you wish to leverage in your new project. By taking advantage of WebLogic Workshop and the BEA control framework, as well as the inherent flexibility of the J2EE platform, you can create a distributed, scalable, back-end system that supports a variety of client deploy-



ments, including a Web application, a thin-client Java application, cross-platform deployment, and more!

To demonstrate this, I'm going to take you through some sample applications that utilize EJBs to provide back-end business logic. We'll also look at how the BEA control framework simplifies the process of integrating these EJBs into our Web services and Web applications.

Note: The sample applications referenced here come packaged with WebLogic Workshop. This allows me to focus on the component-specific aspects within the broader context of a JWS project and enables you to reference the sample projects in your own Workshop environment and experience a working demo.

Entity Beans

Entity beans represent data. They are usually mapped either to records of a relational database or to objects of an object-oriented database. They are transactional in that changes to their state typically occur within the context of a transaction. An instance of an entity bean may be referenced by multiple clients. The container – in this case WebLogic Server – is responsible for ensuring that the data in the bean and the corresponding database are synchronized. Figure 1 shows the sample project, “AccountEJBClient.jws”, loaded into WebLogic Workshop. As you can see, Workshop has wrapped the bean as an EJB control and has implemented control interfaces that map to corresponding EJB methods.

If we were using this bean in another environment, we would have to go through

a bit of a process in order to use it. First, we would have to look up the EJB in the JNDI registry and obtain the home interface. Next, we would need to obtain an instance of the EJB. Finally, we could invoke its methods. While this is by no means a difficult or complicated process, utilizing the control framework makes instantiating and accessing the bean even simpler. All we have to do is add a new EJB control (see Figure 2), point to the EJB (see Figure 3), and we're done. As you can see, once you've browsed to the EJB, Workshop automatically finds the home and remote interfaces. It takes every method of the bean and creates a corresponding control interface. It creates a corresponding control file (.CTRL) for each control created. Listing 1 shows the control file for this EJB control.

Note: Some controls can be modified by editing the CTRL file. Do **not** edit an EJB's CTRL file. It is created by Workshop based upon information it obtains from the EJB's code.

Now that the control has been created, it automatically handles the J2EE plumbing such as instantiating and destroying the bean, accessing the methods, and the like. Please remember that you must have a local copy of the EJB's JAR file in the WEB-INF\lib directory of your Workshop project in order for Workshop to discover the EJB's home and remote interfaces.

Listing 2 shows how WebLogic Workshop accesses the EJB and is the code for the createNewAccount operation. It calls the Create method of the EJB control (which wraps the J2EE interfaces of the EJB's Create method). As you can see, Workshop has done

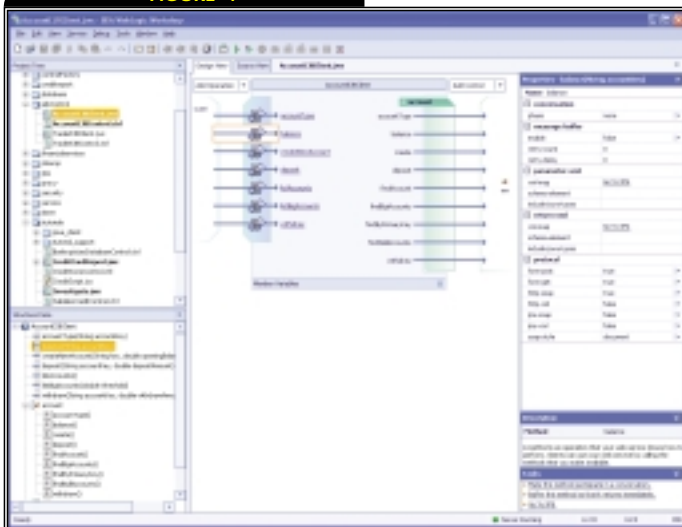
the work of specifying the method parameters as well as putting in exception handling. Since this is a simple “Web Service method to EJB method” mapping, there is little else to do (assuming you trust the functionality of the EJB). However, you can add custom validation code here if you wish.

Session Beans

Session beans represent particular business process steps. They capture the interaction or conversation between the user and the system. Session beans come in two flavors: stateful and stateless. Stateful session beans maintain state across multiple remote method invocations by a client on the same client-proxy of the session bean. Every method invocation by the client is routed to the same session bean instance on the server. These are useful when you need to carry out a long conversation with the server and you need the server object to save the conversational state. An example of a stateful session bean is an online shopping cart, where a client might use remote methods to add, remove, or modify the products in the shopping cart.

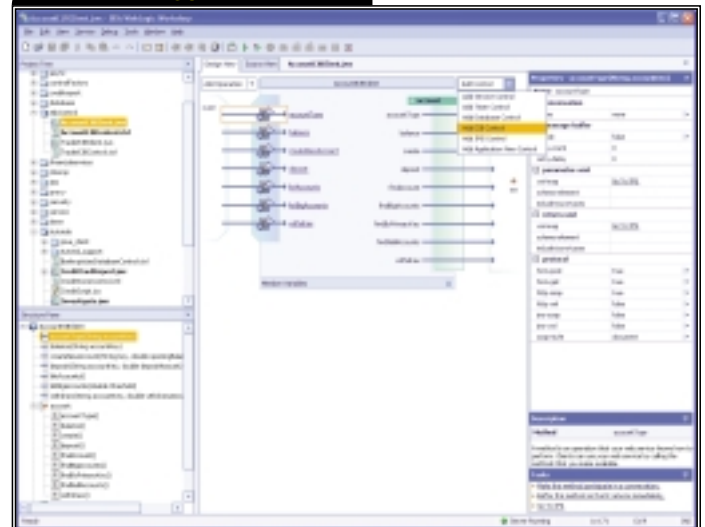
Stateless session beans do not maintain any state that the client can depend on between multiple remote method invocations. These are useful for situations where the duration of the method execution constitutes a complete unit of work from the client's perspective. For an example of a Stateless Session Bean, open the “TraderEJBClient.jws” sample project in WebLogic Workshop. As you can see from the JWS operations (see Figure 4), this component has only two methods: Buy and Sell.

FIGURE 1

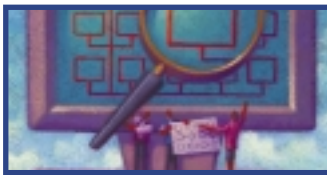


AccountEJBClient.jws, loaded into WebLogic Workshop.

FIGURE 2

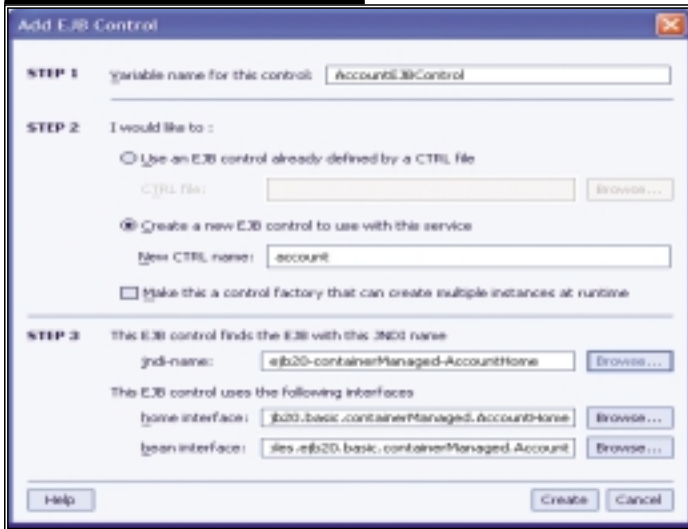


Add a new EJB control



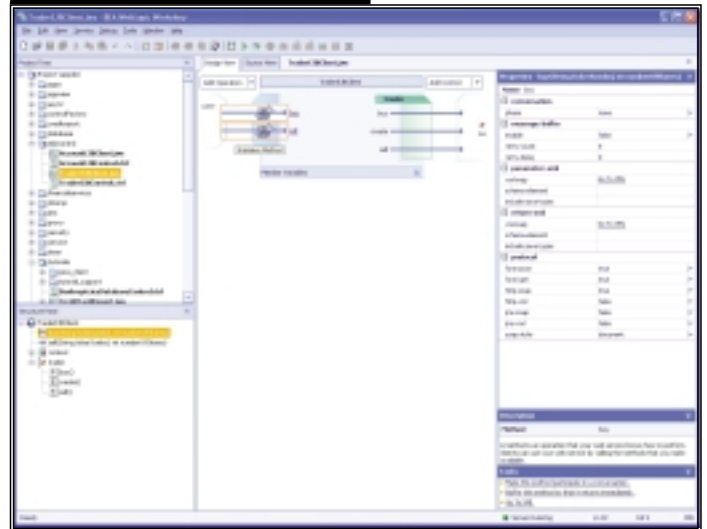
REUSE

FIGURE 3



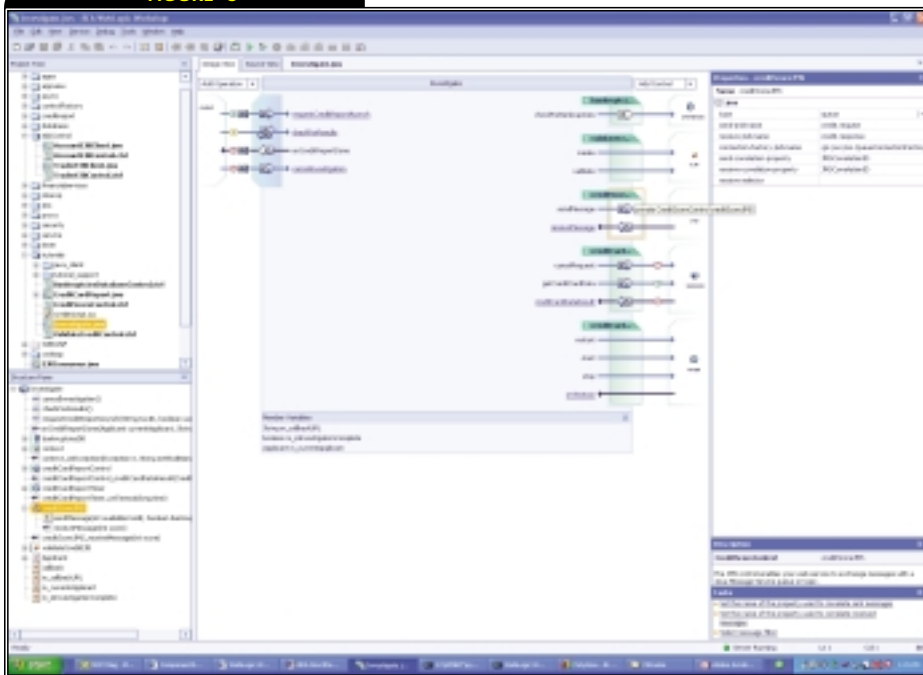
Point to the EJB

FIGURE 4



JWS operations in WebLogic Workshop

FIGURE 5



JWS project using a JMS control to receive credit information

The EJB control (which wraps the interfaces of the EJB itself) actually has a third interface: Create. In fact, every session and entity EJB contains this method. This is how the instance of the bean is actually created. When dealing with entity beans, this method must be explicitly called – since creating an instance of an entity bean actually creates a new row in the corresponding datasource. However, with WebLogic Workshop it is not necessary to fire off the Create method of session beans. The BEA control framework automatically creates an

instance of the bean when any of the methods are called. Even if you do explicitly call the Create method, you might not get a new instance. WebLogic Server may simply return an existing instance of the bean.


Since this is a fine-grained component, I have included both the Buy and Sell method calls in Listing 3. You'll notice that there is very little (if any) validation going on here. One glaring weakness is the lack of any validation that the user actually owns a sufficient quantity of shares to cover the sale. This validation is best added into the

client portion of your application, or as part of a larger Web service – one that integrates an entity bean for account management and a stateful session bean to handle an extended conversation with the server (perhaps for transacting multiple stock sales). You want this validation complete before you even instantiate this bean.

Message-Driven Beans

Message-driven Beans (MDBs) enable the development of asynchronous applications. They essentially sit on the server and wait for Java Message Service (JMS) messages. They are anonymous in that no client may hold a reference to an MDB – only the EJB container may access the bean directly. They have neither a home nor a remote interface. As such, they cannot be used inside WebLogic Workshop. There is, however, an alternative. The BEA control framework includes a JMS control that you can add to your Workshop project. It can then listen for callbacks. Figure 5 shows a JWS project for credit reporting that uses a JMS control to receive credit information from the queue. Listing 4 shows the receiveMessage callback handler that processes the callback. This provides the same basic functionality of an MDB from within the Workshop environment.

Summary

Mission-critical applications that depend on tried-and-tested code benefit greatly from component reuse. WebLogic Server – in conjunction with WebLogic Workshop – can make the integration of these components a breeze. 

A LIMITED TIME SAVINGS OFFER FROM SYS-CON Media

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!

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.

<input type="checkbox"/> 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
	Intl - Two Years (24) \$216 You Pay: \$176 / Save: \$40 + FREE \$198 CD
	Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

<input type="checkbox"/> 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
	Intl - Two Years (24) \$360 You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
	Intl - One Year (12) \$180 You Pay: \$179 / Save: \$1

<input type="checkbox"/> 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
	Intl - Two Years (24) \$216 You Pay: \$176 / Save: \$40 + FREE \$198 CD
	Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

<input type="checkbox"/> 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
	Intl - Two Years (24) \$264 You Pay: \$189 / Save: \$75 + FREE \$198 CD
	Intl - One Year (12) \$132 You Pay: \$129.99 / Save: \$2

<input type="checkbox"/> 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
	Intl - Two Years (24) \$216 You Pay: \$170 / Save: \$46 + FREE \$198 CD
	Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

<input type="checkbox"/> 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
	Intl - Two Years (24) \$216 You Pay: \$170 / Save: \$46 + FREE \$198 CD
	Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

<input type="checkbox"/> .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
	Intl - Two Years (24) \$216 You Pay: \$170 / Save: \$46 + FREE \$198 CD
	Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

<input type="checkbox"/> 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
	Intl - Two Years (24) \$360 You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
	Intl - One Year (12) \$180 You Pay: \$179 / Save: \$1

<input type="checkbox"/> 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
	Intl - Two Years (24) \$216 You Pay: \$170 / Save: \$46 + FREE \$198 CD
	Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

<input type="checkbox"/> 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
	Intl - Two Years (24) \$360 You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
	Intl - One Year (12) \$180 You Pay: \$179 / Save: \$1

3-Pack
 Pick any 3 of our magazines and save up to \$275⁰⁰
 Pay only \$175 for a 1 year subscription plus a FREE CD
 • 2 Year - \$299.00
 • Canada/Mexico - \$245.00
 • International - \$315.00

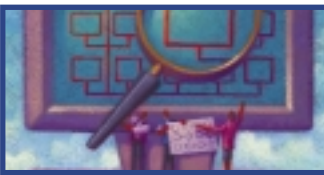
6-Pack
 Pick any 6 of our magazines and save up to \$350⁰⁰
 Pay only \$395 for a 1 year subscription plus 2 FREE CDs
 • 2 Year - \$669.00
 • Canada/Mexico - \$555.00
 • International - \$710.00

9-Pack
 Pick 9 of our magazines and save up to \$400⁰⁰
 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

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



**Listing 1**

```

package ejbControl;

import weblogic.jws.*;
import weblogic.jws.control.*;

/**
 * @jws:ejb home-jndi-name="ejb20-containerManaged-AccountHome"
 * @editor-info:ejb home="AccountEJB.jar" bean="AccountEJB.jar"
 */
public interface AccountEJBControl
    extends examples.ejb20.basic.containerManaged.AccountHome, //
    home interface
        examples.ejb20.basic.containerManaged.Account, //
    bean interface
        weblogic.jws.control.EntityEJBControl // control
    interface
    {

```

Listing 2

```

/**
 * <p>Invokes the target EJB's <b>create</b> method and returns the
 * result.</p>
 *
 * <ul>
 * <li><b>key</b> is the account identifier (must be unique for each
 * account).</li>
 * <li><b>openingBalance</b> is the starting balance of the
 * account.</li>
 * <li><b>type</b> is the account type, e.g. "checking" or
 * "savings".</li>
 * </ul>
 *
 * @return A String describing the result of the account creation.
 *
 * @jws:operation
 */
public String createNewAccount(String key, double openingBalance,
    String type)
{
    try
    {
        /** We can add custom validation code here
         * if we so desire, prior to calling the
         * create method.
         */
        account.create(key, openingBalance, type);
    }
    catch (CreateException ce)
    {
        /** We can put code to process the exception
         * here (or below), prior to returning the
         * error message.
         */
        return "Error " + ce.getLocalizedMessage();
    }
    catch (RemoteException re)
    {
        return "Error " + re.getLocalizedMessage();
    }
    return "Successful";
}

```

Listing 3

```

/**
 * <p>Invokes the target EJB's <b>buy</b> method and returns the
 * result.
 * Trades are restricted to 500 shares.</p>
 *
 * @return A String describing the result of the transaction.
 *
 * @jws:operation
 */
public String buy(String tickerSymbol, int numberOfShares)
{
    TradeResult tr;
    try
    {
        tr = trader.buy(tickerSymbol, numberOfShares);
    }
    catch (RemoteException re)
    {
        return "Error " + re.getLocalizedMessage();
    }
    return String.valueOf(tr.getNumberTraded()) + " shares of " +

```

```

        tr.getStockSymbol() + " bought.";
    }
}

/**
 * <p>Invokes the target EJB's <b>sell</b> method and returns the
 * result.
 * Trades are restricted to 500 shares.</p>
 *
 * @return A String describing the result of the transaction.
 *
 * @jws:operation
 */
public String sell(String tickerSymbol, int numberOfShares)
{
    TradeResult tr;
    try
    {
        tr = trader.sell(tickerSymbol, numberOfShares);
    }
    catch (RemoteException re)
    {
        return "Error " + re.getLocalizedMessage();
    }
    return String.valueOf(tr.getNumberTraded()) + " shares of " +
        tr.getStockSymbol() + " sold.";
}
}

```

Listing 4

```

/**
 * Callback handler for the CreditScoreControl JMS control's
 * receiveMessage callback. This handler is invoked when the
 * CreditScoreControl invokes its receiveMessage callback.
 */
private void creditScoreJMS_receiveMessage(int score)
    throws java.rmi.RemoteException
{
    /**
     * Add the score received by the credit scoring application to
     * the data known about the applicant.
     */
    m_currentApplicant.creditScore = score;

    /**
     * Using the ValidateCreditControl EJB control, request a credit
     * rating from an Enterprise Java Bean (EJB). Store the return
     * value in the approvalLevel field of the m_currentApplicant
     * object.
     */
    m_currentApplicant.approvalLevel =
        validateCreditEJB.validate(m_currentApplicant.creditScore);

    /**
     * If the member variable m_callbackURL has value other than
     * null, then send a message to the client via the callback
     */
    if ( m_callbackURL != null )
    {
        callback.onCreditReportDone(m_currentApplicant, "Credit
            score received.");
    }

    /**
     * If the member variable m_callbackURL has the value null,
     * then the client cannot accept callbacks.
     * Make the message available to the client through the
     * synchronous method checkForResults.
     */
    else
    {
        /**
         * Mark the investigation as complete by setting the
         * isInvestigationComplete field to "true". Clients that
         * cannot accept callbacks will call checkForResults to
         * poll for investigation results;
         * m_isInvestigationComplete is used to determine what
         * checkForResults should return.
         */
        m_isInvestigationComplete = true;
    }
}
}

```


WebServices JOURNAL

.NET J2EE XML

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

LEARN WEB SERVICES. GET A NEW JOB !

SUBSCRIBE TODAY TO THE WORLD'S LEADING WEB SERVICES RESOURCE

Get Up to Speed with the Fourth Wave in Software Development

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



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

Once you're in it...

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

reprint it...

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

Re Prints

SYS-CON MEDIA The world's leading *i*-technology publisher

HOME SUBSCRIBE ABOUT NEWS EMPLOYMENT CONTACT DIRECTIONS

publications

- WebServices
- JAVA
- wireless
- WebLogic
- XML JOURNAL
- COLDFUSION
- WebLogic
- JAVA
- PowerBuilder JOURNAL

SUBSCRIBE NOW TO THE FINEST TECHNICAL JOURNALS IN THE INDUSTRY!

800 513-7111
www.sys-con.com

The world's leading *i*-technology publisher



DEVELOPER RESOURCES

The upcoming release of BEA's WebLogic Platform 8.1 marks a breakthrough in application infrastructure technology.

the subscription program, and will review the benefits delivered at each program level.

BEA dev2dev Subscriptions – Trial Edition

The introduction of the Trial Edition is one of the most exciting changes BEA has ever made in its licensing practices. The Trial Subscription delivers free development seats for all developers on the WebLogic Platform. BEA's decision to grant such liberal access to the platform is directly tied to the technology investments that we have made in the product. For example, take BEA WebLogic Workshop 8.1. Workshop's simplified programming model and visual development environment make it incredibly easy for developers to build applications. This technology is about making more developers more productive. Because this is BEA's focus, it makes sense for us to make this technology widely available to all developers. This means that developers can come to the dev2dev subscriptions site, <http://dev2dev.bea.com/subscriptions>, and request that we ship them a CD that contains the latest WebLogic Platform bits and a development license that is good for one year. For developers who would prefer to get started right away, you can also download the bits directly from our site.

BEA dev2dev Subscriptions – Platform Edition

At \$850 per developer the Platform Edition is the first of our "premium offerings" within the dev2dev subscription program. The Platform Edition delivers more than just the product bits; it joins the product together with BEA's technical support, partner software, and development resources like **WebLogic Developer's Journal**. Typically, customers who are interested in the Platform Edition are developers who have moved from the platform evaluation phase to building applications. This shift toward implementation often requires technical support. The Platform Edition delivers BEA Development Support, pro-

The dev2dev Subscription Program

OFFERING GREATER ACCESS ON SEVERAL LEVELS

BY RYAN O'HARA

In this release, BEA has delivered on its vision of a unified development environment, enabling any IT developer – from mainstream IT developers to J2EE experts – to participate on any IT project across the platform. Using WebLogic Platform 8.1 developers can now move seamlessly from project to project, easily using and reusing any IT asset in a service-oriented manner. In conjunction with this release, BEA has announced the dev2dev subscription program. This program aims to provide WebLogic developers with greater access to the platform technologies so they can enjoy the benefits of this development environment.

BEA dev2dev subscriptions were launched in March 2003 at BEA's annual eWorld conference, promising three levels of access to BEA development resources: Trial Edition, Platform Edition, and Tools Edition. This article provides an overview of



AUTHOR BIO

Ryan O'Hara is a director of developer marketing for BEA Systems, and manages BEA's dev2dev subscription program. He focuses on building new developer offerings that provide developers with greater access to BEA's platform technologies.

CONTACT...

subscriptions@bea.com

"In this release, BEA has delivered on its vision of a unified development environment"

viding 9x5 Support Center coverage with the ability to log incidents via phone, Web, and fax.

In addition to technical support, the Platform Edition also delivers product maintenance. Platform Edition subscribers receive a welcome kit when they sign up for the program. This kit contains the most current releases of the BEA WebLogic Platform and additional development resources. On a quarterly basis, each subscriber will receive a CD update kit in the mail. These kits will have the latest releases of the WebLogic Platform as well as any public betas that may be available. Of course, products are also available via our download center for immediate download. This enhanced delivery of software maintenance means that developers will always have ready access to the products they need. Over the course of one year, subscribers will build a library of WebLogic Platform resources that they can use for any development project. Whether you're working in your office in a connected state, or traveling in an off-line capacity, you will have what you need to get your job done.

Finally, the Platform Edition provides subscribers with a variety of development resources, including trial editions of BEA and partner software, technical book titles from leading publishers on CD, developer education materials, and a 12-month subscription to **WLDJ**. All of these elements make the Platform Edition a comprehensive approach to outfitting development on the WebLogic Platform.

BEA dev2dev Subscriptions – Tools Edition

The \$4,780 Tools Edition builds on the Platform Edition, and adds to it the benefits of Borland JBuilder, WebLogic Edition. JBuilder, WebLogic Edition is preconfigured and optimized to make building and deployment on WebLogic Server especially easy. Subscribers to the Tools Edition not only receive JBuilder but also receive 12 months of software assurance on their JBuilder license. Once again, support is included in this bundle, as BEA provides technical support for both the WebLogic Platform and JBuilder.

Conclusion

The 8.1 release of the WebLogic Platform will make the WebLogic developer community uniquely productive in building enterprise applications. BEA dev2dev Subscriptions seek to deliver that productivity when you need it, where you need it. BEA dev2dev Subscriptions are available for purchase online and through any BEA sales representative. Specific licensing terms and pricing can be found at <http://dev2dev.bea.com/subscriptions>.

SAVE 16% OFF

COLD FUSION Developer's Journal

12 Issues for

\$89⁹⁹

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

- Exclusive feature articles • Interviews with the hottest names in ColdFusion
- Latest *CFDJ* product reviews • Code examples you can use in your applications • *CFDJ* tips and techniques

That's a savings of **\$17⁹⁹** off the annual newsstand rate. Visit our site at www.sys-con.com/coldfusion/ or call **1-800-303-5282** and subscribe today!

THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany
collaboration on a global scale

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

SPECIAL
OFFER!

SAVE \$31*

OFFER SUBJECT TO
CHANGE WITHOUT
NOTICE

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

Go Online
& Subscribe
Today!

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



WebLogicDevelopersJournal.com

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

Application Performance Best Practices

LOGGING AS A TOOL AND THE BENEFITS OF USING EAR FILES

BY LEWIS CIRNE



AUTHOR BIO

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

CONTACT...

asklew@sys-con.com

Q. With respect to logging, how much of a good thing is too much of a good thing?

A. Logging is a powerful application tool that, in my opinion, has been under-utilized. Many architects may have implemented some sort of a strategy for logging common application events to a text file for debugging. In a few cases, full-featured audit strategies have been implemented for complete documentation of every customer transaction, perhaps for review at a later date. Most of the time, however, logging is being used just to report major warning or error conditions within the application code.

If you haven't looked at the cost of logging to your production application, you will probably be surprised by just how many CPU cycles are burned by it. Look for a long, steady burn rate in your CPU utilization – whatever the CPU utilization of your WLS process “settles” to between any visible spikes. If your WLS instance isn't handling any highly concurrent user loads, this “sizzle” or “baseline” can often be accounted for by the writes to your log file, particularly if you see much of the utilization at the system level of the OS.

One of the most common culprits is unhandled exceptions on core application paths. As any performance primer will explain, exception-handling routines are among the most expensive in any application's code. It gets even more expensive when your handling strategy is to print every single exception – whether meaningful or not – to standard out or standard error. In some cases, Wily has seen customers generating megabytes of log-file every hour – all with the same exception throws from the same use case in the application. I don't recommend doing that.

Once you get unnecessary exceptions out of the logging picture, you should still be thinking carefully about what information, if any, you

want your application to be logging to a file. JMX is another great option for handling many of the problems that some architects have only considered logging to solve, but make sure you have thought through what you are going to do with the data you publish via JMX.

Q. Why should I deploy my application as an EAR file?

A. In most organizations that are currently using Java in production, there are at least a few legacy projects from the early years of Java's development as an enterprise platform. As the Java platform matured, feature and functionality were added to address many of the most common business problems encountered by the early pioneers. With today's full J2EE solution there are a myriad of facilities available that I recommend you take advantage of. To that end, many people have asked me why they should spend their time repackaging old applications.


When confronted with that question, some IT managers protest by saying that they really don't need the ability to run multiple versions of a single application in a single application server. They say that they're happy continuing to get away with the arrangement they have used to date, loading their application code on the system classpath.

There are, however, far more compelling reasons to go ahead and do the work. Perhaps the most important should be fear of the unknown. With even a minor revision to your application server new classes and new versions of classes can appear unexpectedly on your system classpath ahead of your application code. A new revision of an application that has been running perfectly fine, and has gone through all its QA smoke tests without issue, can die inexplicably in production as a result of an uncontrolled change to one of the key classes that it happens to share with the server. Unless you truly know everything that's going in and out of weblogic.jar with each point release of the BEA WebLogic Server, this should scare you. Bugs like these can be extremely difficult to discover.

If you have an environment where you're running more than one application per server instance, then the problem gets exponentially more complicated from there. Repackaging your application as a proper enterprise archive not only shelters you from many of these types of application problems, but allows you to take full advantage of a modern application server's ability to host dozens of different applications without

conflict. Build-and-release management becomes less costly and more straightforward when each revision of an application is encapsulated by a single file – you can perform rollbacks more quickly and easily, should they be necessary, and you have the additional peace of mind that any tool that purports to support Java enterprise ought to read your application in this format. Additionally, any future work on your application should become easier as well.

...

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

SAVE 17%

DEVELOPER'S PowerBuilder Journal

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

12 Issues for \$149

- New PowerBuilder features
- Tips, tricks, and techniques in server-side programming
- DB programming techniques
- Tips on creating live PowerBuilder sites
- Product reviews
- Display ads for the best add-on products

That's a savings of \$31 off the annual newsstand rate. Visit our site at www.sys-con.com/pbdj/ or call 1-800-303-5282 and subscribe today!

WLDJ ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
Alltaworks	www.alltaworks.com/wldj	888-877-7111	COVER IV
BEA Systems	http://dev2dev.bea.com/useworkshop	800-817-4BEA	3
ColdFusion Developer's Journal	www.sys-con.com/coldfusion	888-303-5282	47
E.piphany	www.epiphany.com/psgreport	877-764-4163	15
Informatica	www.informatica.com	800-653-3871	2
Interwoven	www.interwoven.com	408-774-2000	25
LinuxWorld Magazine	www.linuxworld.com	888-303-5282	39
LinuxWorld Conference & Expo	www.linuxworldexpo.com	800-657-1474	29
PowerBuilder Developer's Journal	www.sys-con.com/pbdj	888-303-5282	49
Veritas	www.veritas.com	650-527-8000	11
Quest Software	http://java.quest.com/jcsv/wldj	800-663-4723	9
Quest Software	http://java.quest.com/qcj/wldj	800-663-4723	23
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	43, 45
SYS-CON Reprints	www.sys-con.com	201-802-3026	45
Web Services Edge West	www.sys-con.com	201-802-3069	35
Web Services Journal	www.wsj2.com	888-303-5282	45
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	47
Wily Technology	www.wilytech.com	888-GET-WILY	4, 31
Yahoo	www.enterprise.yahoo.com/bea-testdrive	866-267-7946	51
Yantra Corporation	www.yantra.com	888-2YANTRA	19

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



The Promise of Patterns

Although many software developers have heard of patterns and seen the advertised benefits, most still do not fully understand what they are and how to effectively apply them in their daily work. Despite the exciting promise of patterns, they are still unreachable for many. This article looks at what patterns really are, clears up some common misconceptions, and explores how patterns are great catalysts for great software.



Improved Performance with the EJSelect Data Aggregation Pattern

It's not a silver bullet or a golden hammer but it is a pretty nifty pair of pliers. And, it may be a way to get better performance without losing CMP transaction handling.



Plus, Peter Holditch returns with more on transaction management:

Freedom, Disasters, and Getting Something for Nothing

What is certain is that somewhere, at some level in your solution, whatever shape it takes, is a transaction manager (or something that smells like one) holding things together.



Server-Side VM Architecture

Java uses a virtual machine to imitate a standardized computer system from the programmers' point of view. This month, members of BEA's Office of the CTO look at how the server side is optimized for J2EE work.



News & Developments

HP/BEA to Deliver Integration Solutions

(Palo Alto, CA) – BEA WebLogic Server is now available on HP AlphaServer systems running HP OpenVMS, HP ProLiant servers running Linux, and HP NonStop servers. HP has also announced a cooperative support agreement with BEA that provides customers running WebLogic Server on HP platforms with support from either HP or BEA.

This agreement establishes a formal support relationship between HP and BEA that outlines processes to be used to resolve interoperability concerns for joint customers.

www.hp.com
www.bea.com



BEA Expands Alliance with CSC's Consulting Group

(San Jose, CA) – Computer Sciences Corporation will leverage the BEA WebLogic Enterprise Platform on its CSC e4 enterprise architecture. This will enhance the ability of its Fortune 2000 clients to react quickly to change, institute process improvements, and improve collaboration – both internally and externally.

Based on open standards and best-of-breed technologies, CSC e4 delivers a complete, practical operational business process management (BPM) environment.

www.csc.com



Wily Technology in Leader Quadrant

(Brisbane, CA) – Wily Technology, a leader in Enterprise Java Application Management, announced that Gartner Inc. has positioned it in the leader quadrant in the J2EE Application Server

Management Magic Quadrant.

Characteristics of companies in the leader quadrant include extensive management visibility, advanced functionality, increased ease-of-use features, and broad support for a wide range of J2EE offerings as well as operating systems.

www.wilytech.com



Covalent Delivers Web App Management Solution

(San Francisco) – Covalent Technologies has announced the immediate availability of Covalent Application Manager (CAM), its comprehensive Web application management solution designed specifically for IT Operations. CAM presents an integrated application view by automatically discovering and then correlating multiple components of a Web application.

www.covalent.net



BEA and Salesforce.com Announce Alliance

(San Jose, CA and New York) – BEA Systems, and salesforce.com have announced a strategic alliance to provide services-oriented application development solutions based on BEA WebLogic Workshop 8.1 and the BEA WebLogic Enterprise Platform. This alliance will help advance sforce, the first client/service application development framework to enable enterprises to rapidly build and deliver business applications based on the software-as-service model.

Salesforce.com will offer BEA WebLogic Workshop 8.1 Java controls for sforce. The alliance will make it easier for customers to build, extend, and integrate service-oriented applications on the BEA

WebLogic Enterprise Platform 8.1. The Java controls are service-oriented components, introduced with BEA WebLogic Workshop that make it easier to connect to and use any IT system or asset as a reusable service.

www.sforce.com
www.bea.com



Ubiquity, BEA Announce Joint Solution

(Atlanta) – Ubiquity, a developer of SIP-based communications software for service providers, has announced a joint solution with BEA Systems. Ubiquity has been selected as part of the new carrier-grade BEA WebLogic Service Delivery, which presents a standardized software infrastructure platform for developing, integrating, and deploying new digital services. The result leverages BEA WebLogic Workshop's integrated development framework to empower all application developers – not just J2EE or SIP experts – to rapidly create and deliver real-time, SIP-based services such as voice calling, presence, and instant messaging to end customers.

www.ubiquity.net



Persistence Software Enhances EdgeXtend for C++

(San Mateo, CA) – Persistence Software, a provider of real-time data services software for financial services and other data-intensive markets, has announced the availability of EdgeXtend for C++ Release 8.0. EdgeXtend for C++ joins EdgeXtend for BEA WebLogic in the EdgeXtend family of Persistence products.

www.persistence.com



Cap Gemini Ernst & Young, BEA Launch IT Solution

(New York and San Jose, CA) – Cap Gemini Ernst & Young U.S. LLC and BEA Systems have unveiled OFL(SM), Open Framework for Leasing, a comprehensive solution powered by



BEA WebLogic Platform 8.1. OFL is a business integration suite designed to help companies consolidate and integrate leasing and lending applications more quickly and cost-efficiently, and can be customized to support any financing business.

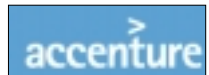
www.cgey.com

Accenture Accelerator Eases Integration

(New York) – Accenture has introduced the Accenture Platform Accelerator, a new software platform designed to help companies build and integrate e-business applications on BEA WebLogic Platform more quickly and cost-effectively than previously possible.

Developed with BEA Systems, the Accenture Platform Accelerator will also help users integrate applications seamlessly, and minimizes the resources needed to run business processes and build applications on the WebLogic Platform.

www.accenture.com



Yahoo

www.enterprise.yahoo.com/bea-testdrive

Altaworks

www.altaworks.com/wldj