SYS-CON

ARE YOU A WEBLOGIC EXPERT? PAGE 4

NOV/DEC 2004

wldj™

THE LEADING INDEPENDENT MAGAZINE FOR WEBLOGIC™ PROFESSIONALS

WWW.WLDJ.COM

Database Controls Best Practices

AVOIDING POTENTIAL PITFALLS DURING THE DEVELOPMENT LIFE CYCLE ...8

PLUS...

$9.99US $9.99CAN

12>

0 74470 03424 7

Are You a WebLogic Expert?

BY JOE MITCHKO

In my travels I am often asked whether I am a WebLogic expert. I don't blame people for asking, since it is assumed that anyone who is editor-in-chief of a magazine dedicated to keeping its readers up to date on the myriad of BEA products must know everything there is to know about this end of the software spectrum. You might assume that by now I would be used to the question, "Are you a WebLogic expert, you know, a guru?" Every time though, it stops me in my tracks and I have to stop for a second and compose myself. Am I an expert? Umm, err…say something, quick. I usually respond with, "Well, I think I know enough to get by, how can I help you?" In most cases, I am able to provide some wisdom or at least guide the individual towards an answer to the particular problem or architectural design conundrum in which he or she embroiled.

The question is so difficult to respond to because the term "WebLogic" covers a lot of ground. I am often left puzzled for a day or two, asking myself, should I be a WebLogic expert? If I were an expert, what would I know beyond what I already know?

This time around, the subject at hand centered on deployment planning and the best approach for selecting the types of servers to use, arrangement of BEA WebLogic domains, memory and CPU requirements, and so forth. So, on went my WebLogic administrator's hat along with years of experience dealing with Unix, Linux, symmetric multiprocessing, and Java Virtual Machine technology, to name a few. In addition, the architectural discussion also involved application scaling, concurrent and peak usage patterns, and so forth. Between the two of us we were able to work through an optimal solution.

But, there is an endless supply of questions that can be asked, spanning years of technology and experience levels, making it difficult for any one person to know it all. For instance, to be a WebLogic expert you need to have expertise in Java technology, a full understanding of the functional capabilities and specifications of J2EE, and of course, knowledge of how to design systems using WebLogic Workshop, WebLogic Integrator, Liquid Data, and so on. You also need to be fluent in XML, XSL, XQuery, and similar data-related technologies. You also need to be fully aware of how to deploy Web applications and components, including servlets and EJBs, and have a good understanding of deployment descriptors. You also need to know your way around database drivers, connection pools, and other performance-improving mechanisms.

In the portal technology area, you need to understand how to develop and deploy portlets to WebLogic Portal, and have a solid background on all of the relevant standards that are out there. You also cannot escape reaching the expert status without having full knowledge of SOAP, WSDL, BPEL4WS, and all of the other emerging standards required for an SOA based enterprise. Shall I continue?

Assimilating all of this requires years of practical experience, after which maybe, just maybe, you may reach the guru status. In reality, the experts in the field are few and far between. More often, we rely on networking and our fellow associates to help us work through architectural and design issues, and we also rely on the knowledge and expertise of the "experts" who publish articles for this magazine. And that, of course, is why we are here at ***WLDJ*** – to help you sort through what you need to know as a WebLogic developer.

Have a great and prosperous New Year!

AUTHOR BIO...

Joe Mitchko is the editor-in-chief of *WLDJ* and a senior technical specialist for a leading consulting services company.

CONTACT: joe@sys-con.com

wldj THE LEADING INDEPENDENT MAGAZINE FOR WEBLOGIC PROFESSIONALS

EDITORIAL ADVISORY BOARD
LEWIS CIRNE, WAYNE LESLEY LUND,
CHRIS PELTZ, SEAN RHODY

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JOE MITCHKO

PRODUCT REVIEW EDITOR
JASON SNYDER

EXECUTIVE EDITOR
GAIL SCHULTZ

EDITOR
NANCY VALENTINE

ASSOCIATE EDITOR
JAMIE MATUSOW,

ASSISTANT EDITOR
NATALIE CHARTERS

RESEARCH EDITOR
BAHADIR KARUV, PhD

WRITERS IN THIS ISSUE
PATRICK CHANG, PETER HOLDITCH,
BALAMURALI KOTHANDARAMAN, SAI KUMAR,
MARIUS LARSEN, PRAKASH MALANI, JOE MITCHKO,
VIDAR MOE, GERALD NUNN, RAJUL RANA,
JASON SNYDER, ANJALI ANAGOL-SUBBARAO

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

PRESIDENT AND CEO
FUAT KIRCAALI

VP, BUSINESS DEVELOPMENT
GRISHA DAVIDA

GROUP PUBLISHER
JEREMY GEELAN

PRODUCTION CONSULTANT
JIM MORGAN

ART DIRECTOR
ALEX BOTERO

LEAD DESIGNER
ABRAHAM ADDO

ASSOCIATE ART DIRECTORS
LOUIS F. CUFFARI • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR
TAMI BEATTY

SENIOR VP, SALES & MARKETING
CARMEN GONZALEZ

VP, SALES & MARKETING
MILES SILVERMAN

ADVERTISING SALES DIRECTOR
ROBYN FORMA

NATIONAL SALES & MARKETING MANAGER
DEENNIS LEAVEY

ADVERTISING SALES MANAGER
MEGAN MUSSA

ASSOCIATE SALES MANAGERS
KRISTIN KUHNLE • DOROTHY GIL • KIM HUGHES

PRESIDENT, SYS-CON EVENTS
GRISHA DAVIDA

NATIONAL SALES MANAGER
JIM HANCHROW

FINANCIAL ANALYST
JOAN LAROSE

ACCOUNTS PAYABLE
BETTY WHITE

INFORMATION SYSTEMS CONSULTANT
ROBERT DIAMOND

WEB DESIGNERS
STEPHEN KILMURRAY • MATTHEW POLLOTA

ONLINE EDITOR
MARTIN WEZDECKI

CIRCULATION SERVICE COORDINATORS
EDNA EARLE RUSSELL • LINDA LIPTON
MONIQUE FLOYD

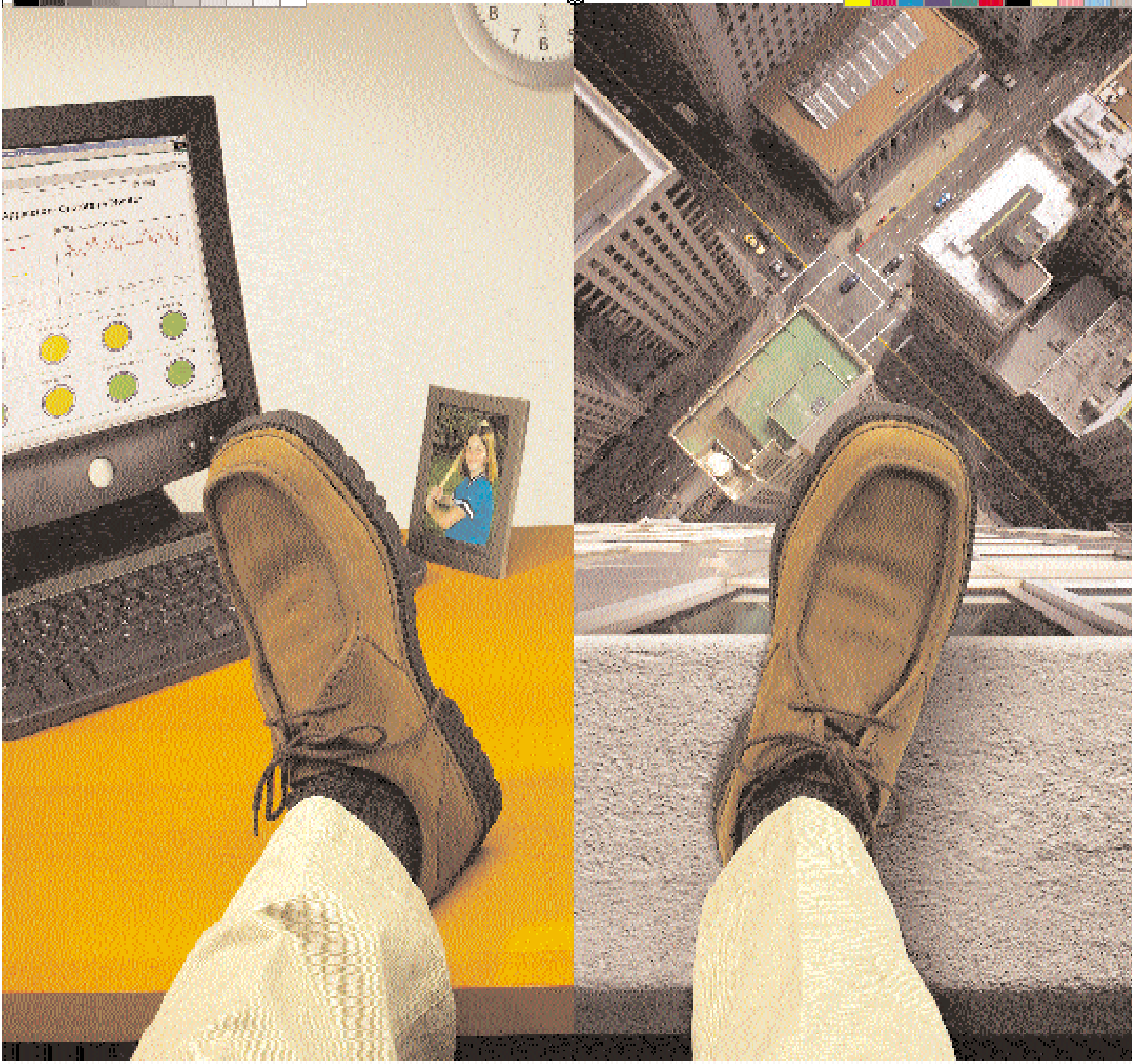
SUBSCRIPTIONS:
888-123-4567
201-123-4566
subscribe@sys-con.com

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

WORLDWIDE NEWSSTAND DISTRIBUTION
Curtis Circulation Company. New Milford, NJ

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

SYS-CON EVENTS



Open Source Technologies

APACHE BEEHIVE AND WORKSHOP

BY KUNAL MITTAL

Earlier this year, BEA donated several proprietary technologies to the open source community primarily to increase the adoption of BEA WebLogic Workshop, which is the basic entry point into the WebLogic Platform suite. Although for typical J2EE applications deployed on the WebLogic Server, Workshop serves only as a basic IDE; for development in WebLogic Portal, BEA WebLogic Integration or BEA Liquid Data for WebLogic, Workshop forms the only IDE that you can really use.

By open sourcing several technologies such as XML Beans, Java PageFlows, Controls, and others, BEA hopes that several plugins will be created so that more popular development tools can be used to develop J2EE and Web services applications on WebLogic Workshop.

XMLBeans

In September 2003, BEA released the XMLBeans technology to open source. Like OR mapping tools allow you to make relational data to objects, XMLBeans provide a way to map XML data to objects. XMLBeans fully support XML Schemas and provide an easy API to access XML data as if you were accessing data in a JavaBean. XMLBeans 1.0 is currently a stable release, and a lot of work is being done to make it more suitable for large Web services applications.

Assume you have an XML Schema. You can easily generate a set of classes that allow you to work with this schema. In BEA WebLogic Workshop, this can be done using an XML Schema project. You get a JAR file with the classes that are used to work with any XML file that conforms to this schema. For example, you can access data from the XML file using these simple commands.

```
// Load an XML document
MyDocument myDocument =
MyDocument.Factory.parse(inputFile);
Document doc = myDocument.getDocument();

// get an element from the head
Header header = doc.getHeader();
Element element = header.getElement();
Attribute attribute = element.getAttribute();
```

One good way to use XMLBeans is as Data Transfer objects from your View layer to your Model layer.

Apache Beehive Project

The Beehive project was created with the Java PageFlow technologies that were introduced in BEA WebLogic Workshop. The main focus of the project is to simplify J2EE and Web services programming by using annotations. Java PageFlows are directly on top of Struts, and provides support for JSR-175, a metadata specification that allows you to reduce the J2EE coding required.

In addition to PageFlows, the Beehive project also has support for the technology called “Controls” – a component framework with support for metadata. The final piece of the Beehive project is JSR-181 – an annotation-driven model for Web services.

Initially, development for Java PageFlows and Controls could only be done using WebLogic Workshop as your IDE. Recently the Eclipse Foundation launched a project called “Pollinate” to build support for Beehive into the Eclipse IDEs. Other popular IDEs, such as IDEA IntelliJ, NetBeans, and JBuilder will soon fully support Beehive projects. (*Note*: This is not to say that you cannot use these IDEs today. See the Beehive Wiki in the References for more information.)

Beehive has been released under the Apache License v2.0 and is being promoted widely by BEA. Only 50% of the committers to Beehive are BEA employees, which should encourage other developers to contribute to this project.

My Take

Personally, I believe BEA made a great decision by open sourcing several of their proprietary technologies. Not only will this foster a greater adoption of BEA WebLogic Workshop as a tool, but as more J2EE applications are built faster, it will help the WebLogic Server find greater adoption. J2EE and Web services developers will be able to build applications faster and more easily using the metadata annotation for a lot of core/redundant tasks (like writing getters and setters for your Java Beans).

References

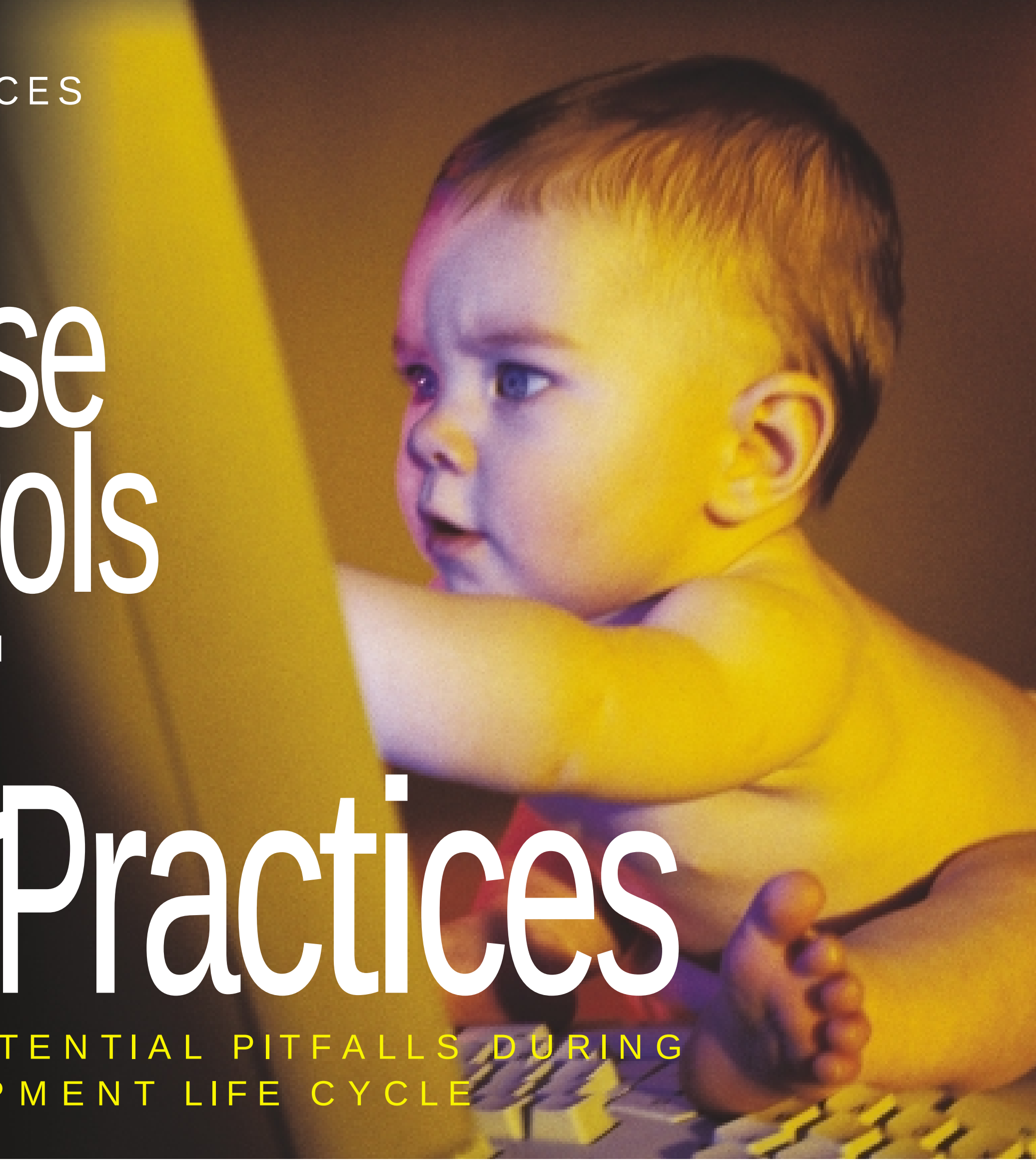
- *Apache XML Beans Web site:* http://xmlbeans.apache.org
- *BEA's dev2dev from BEA:* http://dev2dev.bea.com
- *Apache Beehive:* http://incubator.apache.org/beehive
- *Beehive Wiki:* http://wiki.apache.org/beehive
- *Eclipse Pollinate:* www.eclipse.org/pollinate

AUTHOR BIO...

Kunal Mittal is a portal architect at Sony Pictures Entertainment, and a consultant for the implementation and strategy of Web services. He has coauthored and contributed to several books on Java, WebLogic, and Web services. Kunal has worked on numerous projects using WebLogic Server, WebLogic Portal, and WebLogic Workshop in finance, real estate, supply chain, broadband, entertainment and ISVs in the Web services space.

CONTACT: kunal@kunalmittal.com

Create software so brilliant
it can manage itself.



Database Controls Best Practices

AVOIDING POTENTIAL PITFALLS DURING THE DEVELOPMENT LIFE CYCLE

BY **GERALD NUNN**

AUTHOR BIO...

Gerald Nunn works as a principal consultant at BEA Systems assisting clients to implement solutions on the WebLogic Platform. Prior to joining BEA, Gerald was the director of Java development at Workbrain.

CONTACT...

gnunn@bea.com

Database controls extend the BEA WebLogic controls architecture to enable developers to easily interact with relational databases with maximum productivity. Utilizing a database control, a developer can quickly create methods to issue commands and retrieve data as required. Database controls can often be a more productive choice over other technology alternatives such as Enterprise JavaBeans (EJB).

While database controls were introduced in BEA WebLogic Workshop 7.0, they were not overly popular until the introduction of the 8.1 version of the WebLogic Platform and the dramatic increase in WebLogic Workshop functionality in the 8.1 release. Developers now approaching these controls for the first time can be forgiven for thinking that the controls are so simple that no best practices are required. In fact, the very simplicity of the control is what can often cause maintenance problems down the road as the database control is utilized in less than optimum ways.

This article will discuss a set of best practices pertaining to the use of the database controls. I assume that readers are already somewhat familiar with database controls and their usage; developers new to these controls are encouraged to review the help section in WebLogic Workshop, which contains an excellent overview.

Rather than complete a laundry list of do's and don'ts, we have opted to examine best practices in the context of a specific use case. In this hypothetical scenario we are a developer for a company called "Music Search" that runs a WebLogic portal that connects music industry employees with up-and-coming musical acts. As a developer for this site, we have been tasked to create a portlet that enables the site administrators to poll the site's users as a mechanism to gather information. The functionality consists of displaying a single question with multiple answers in the portlet. The user's response to the question is recorded; once answered, the percentage of responses for each answer is displayed to the user.

The database schema for the functionality required by the portlet will consist of three tables (see Figure 1).

The three tables are Question, Answer, and Response. The Question table is used to store an individual question while the Answer table is used to store the available choices for a given question. Finally, the Response table is used to store a user's answer to the displayed question. While linking the response table to the question table through the QUESTION_ID foreign key is not strictly necessary, it has been done here as a convenience for reporting.

Our first order of business is to create the logic required to add, update, and delete questions, answers, and responses. To accomplish this, we will create three classes to represent the domain model. The business logic will manage the relationship between the Question and Answer classes; however, the Response class will only be used for inserting new responses, as the users will utilize a reporting tool for analyzing polls.

A simplified UML diagram of our three domain classes is shown in Figure 2. Note that the Answer class is aggregated by the Question class and is only accessible through an instance of the Question class.

When To Use

One of the most important considerations of database controls is knowing when to use them within a given application. Database controls provide many of the same services that developers might alternatively use session and entity EJBs to encapsulate. Thus a common question would be, "when is it appropriate to use a database control versus other solutions?"

Database controls in their current incarnation are intended to enable developers to write code that interacts with the database in a highly productive manner. While the increased productivity of database controls is a boon, developers should understand that they provide considerably less flexibility than an equivalent EJB. They are particularly well suited for supporting smaller applications and discrete pieces of functionality where the large feature set and complexity of EJBs is not required. In these cases the increased productivity of the database control is the compelling argument for their usage.

However, there are certainly cases where the power and flexibility of EJB is needed and database controls should not be used. Specifically, database controls may be less than optimal in applications whose components require complex transaction management or declarative transactions. These applications should strongly consider using an EJB architecture instead of database controls as database controls provide limited flexibility with respect to transaction management.

Second, with respect to object persistence, larger applications that feature a highly complex domain model with many relationships between data elements and are evaluating database controls for CRUD operations should consider choosing a more robust and flexible persistence mechanism such as entity beans, Hibernate, or Toplink. This is because the database control provides no mechanism for transparently managing relationships between objects and developers can quickly find themselves overwhelmed in writing this glue code in complex situations.

Best Practice: Carefully consider the requirements for your particular needs prior to choosing a technological architecture; database controls are not a panacea for every situation.

Database Control Encapsulation

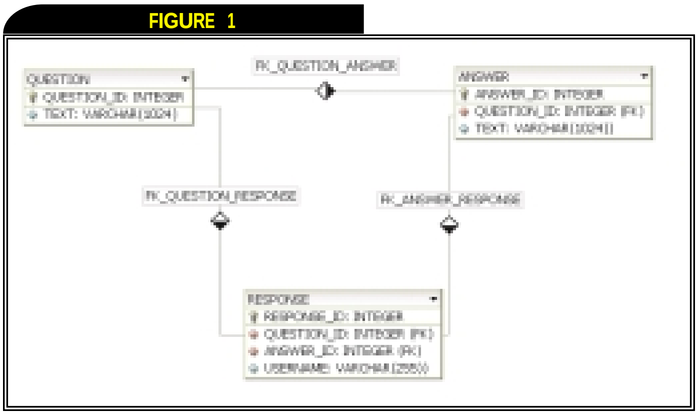
Our first task is to decide on the organization of code for our polling portlet. It is here that many developers using the database control make their first mistake – a mistake they sometimes do not realize until much later in the development cycle.

The problem arises from the fact that unlike many of the controls in WebLogic Workshop, a database control is a Java interface, not a class. When the control is compiled, the WebLogic Workshop framework creates an implementing class based on the attributes and methods defined by the developer in the database control interface. The sample method below retrieves an instance of the Question class based on an identifier.

```
/**
 * @jc:sql statement::
 *      SELECT QUESTION_ID ID,TEXT
 *      FROM QUESTION
 *      WHERE QUESTION_ID={questionId}
 * ::
 */
Question getQuestion(long questionId) throws SQLException;
```

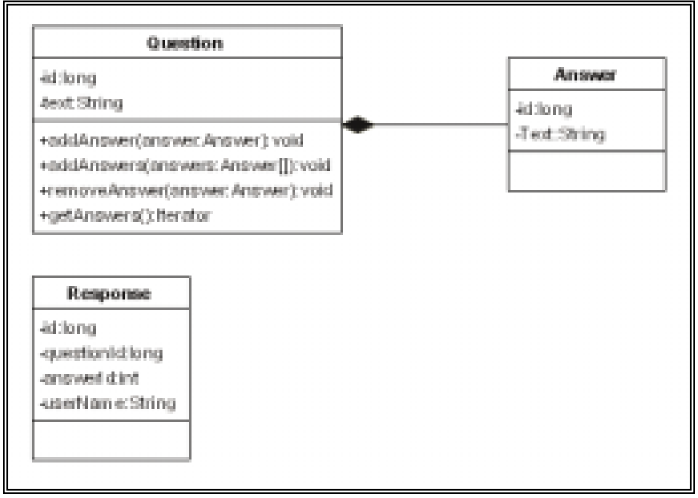
Often a developer who is new to database controls will initially start with a simple control that performs basic CRUD operations. The developer will then start using this database control in page

FIGURE 1



Question, Answer, and Response tables

FIGURE 2



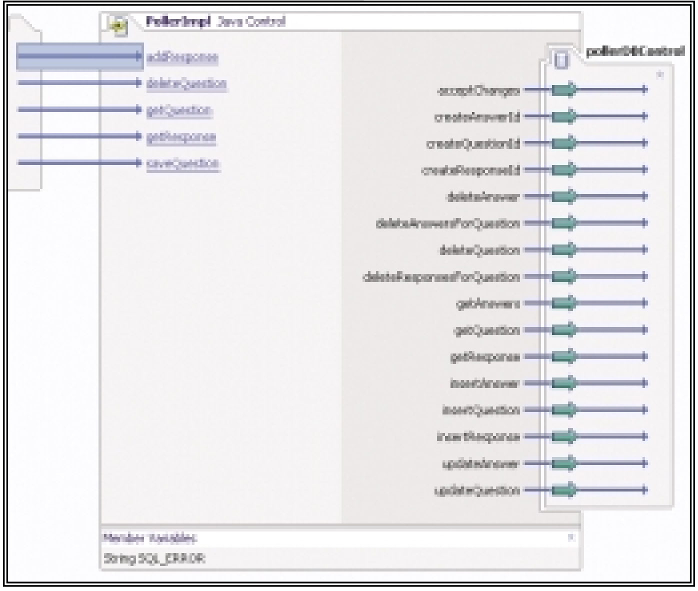
Question, Answer and Response classes

flows and processes as needed. At some point though, the developer will realize that he needs to perform a complex operation that cannot be encapsulated in a single method call. This is a problem we will also face in our polling portlet in many instances, for example deleting a question also requires us to delete all of the answers and responses.

Since the database control is defined as an interface, there is no opportunity for the developer to add custom code to the database control itself. Now the developer is forced to add two or more methods and rely on the user of the database control to call those methods in the correct order. For example, to delete a question we would need to have three database control methods, one to delete the question, one to delete the question's answers, and one to delete the question's responses.

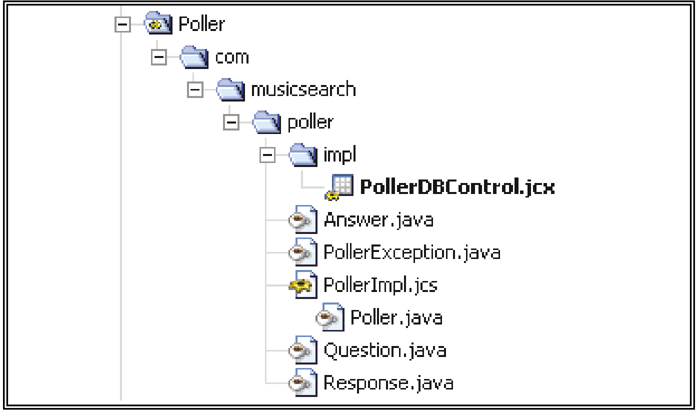
When the presentation layer has to delete a question, it is clearly undesirable to force the presentation to call these three methods individually in the correct order. Further complicating the issue is that the presentation layer would have to call these methods within the context of a transaction in order to ensure the deletion is rolled back should the second or third method calls on the database control fail.

FIGURE 3



PollerImpl control with embedded database control

FIGURE 4



How to place the database control

A far better approach is to encapsulate the database control within a Java custom control. In this way we never expose the database control directly to consumers such as page flows and processes; they will use the Java custom control instead. Thus these consumers are completely oblivious to the fact that database controls are being used as the implementation mechanism; all they perceive is the encapsulating Java custom control.

This approach has two primary benefits:

- ***Increased maintainability:*** The control developer can expose methods on the database control directly through encapsulation. If these methods require additional logic or that operations be added later in the development life cycle, the control developer can expand the functionality as needed in the Java custom control without negatively impacting code that is using the control. If the database control were exposed directly, this would not be possible.

  An additional maintenance benefit is that if the underlying database control needs to be replaced with a more robust mechanism such as an EJB, this can be done without necessarily needing to modify code external to the encapsulating custom control.
- ***Reduced transaction complexity***: Each call to a control method starts a transaction if one has not already been started. Thus a call to two database control methods will happen as two transactions unless another transaction, such as a UserTransaction, has already been started. If the two database control calls were placed in a single Java custom control method, the transaction would automatically be started by the call to the custom control and the two database control methods would happen in the context of a single transaction as desired, with no additional code being written by the developer.

Best Practice: Always encapsulate a database control within a Java custom control and avoid using database controls directly in page flows and processes.

Now that we have established the importance of encapsulating the database control within a Java custom control, the next area of discussion is organizing your controls with an appropriate package hierarchy. Given that controls are compiled into Java classes, I highly recommend that you utilize standard Java naming conventions for packages. In WebLogic Workshop, this means creating folders for each level of the package hierarchy. In our particular use case, we will place our code and controls within the package com.musicsearch.poller. The implementing Java custom control we will use will be called PollerImpl.

Figure 3 shows our PollerImpl control with the embedded database control PollerDBControl.

Best Practice: Use standard Java naming conventions for packages when creating folders for containing controls.

Next we have to determine where to place the database control that we will use for working with the underlying tables. It is important to convey to developers that the database control is an implementation-specific class not meant for public consumption. Many organizations have a naming convention in place for identifying packages that contain implementatio- specific classes and these packages are often named internal or impl.

In our example we will place the database control within a subpackage called *impl*. The package structure used in build-

ing our control project to support the portlet is shown in Figure 4.

Best Practice: Place database controls within an implementation package to clearly indicate it is an implementation specific artifact.

Object Persistence and Database Controls

In order for the portlet to display a question and answer to the user we need to add appropriate methods to the database control to load the Question and Answer objects from a select statement. This is done quite easily, as illustrated in the code sample below.

```
/**
 * @jc:sql statement::
 *     SELECT ANSWER_ID ID, QUESTION_ID QUESTIONID, TEXT
 *     FROM ANSWER
 *     WHERE ANSWER_ID={questionId}
 * ::
 */
Answer[] getAnswers(long questionId) throws SQLException;
```

BEA documentation in WebLogic 8.1 GA and SP2 implies that your member fields must be public in order for the objects to be able to interact with a database control. This is in fact incorrect; you should create your value objects in accordance with standard Java practices, namely private fields with public method accessors, i.e., getters and setters.

Therefore, instead of declaring the Answer class with public fields as follows:

```
public class Answer
{
    public long id = -1;
    public String text;
}
```

we declare it as per standard Java conventions using private fields and public method accessors:

```
public class Answer
{
    private long id = -1;
    private String text;

    public long getId()
    {
        return id;
    }

    public void setId(long value)
    {
        this.id = value;
    }

    public String getText()
    {
        return text;
    }

    public void setText(String value)
    {
        text = value;
    }
}
```

"Carefully consider the requirements for your particular needs prior to choosing a technological architecture"

Best Practice: Create classes for use with database controls in accordance with standard Java practices and conventions.

Another issue that often arises is naming conventions for class members. In order to load an object with values from a select statement, the database control requires that the member names be identical to column names in the database. Given that naming conventions for database columns and Java members vary considerably, this requirement could potentially result in poor names in either the database or Java class scope.

Fortunately, an easy way to resolve the issue is to simply alias the columns as needed within the select statement in the database control so that column aliases conform to the correct member names. In the example below, we alias the column QUESTION_ID to ID in order for the database control to apply the column value to the Id member without mangling names of either the database column or the Java class member.

```
/**
 * @jc:sql statement::
 *     SELECT QUESTION_ID ID,TEXT
 *     FROM QUESTION
 *     WHERE QUESTION_ID={questionId}
 * ::
 */
Question getQuestion(long questionId) throws SQLException;
```

Best Practice: Avoid mangling member names by aliasing Database columns in queries to match the member name as required.

Conclusion

This article demonstrated a variety of best practices with respect to the database control in the context of a polling portlet. Using these techniques will enable you to avoid potential pitfalls with database controls during the development life cycle.

Need to solve performance problems in BEA WebLogic™ 8.1 applications?

Manage Complex Enterprise Applications

Acsera has the advanced technology needed to manage complex BEA WebLogic™ 8.1 applications in production.

100% Automatic Discovery and Blueprinting

Acsera discovers your WebLogic clusters, nodes, servers and deployed applications automatically in minutes.

Performance Management and Monitoring

Acsera is Performance Centric – it helps you deliver the high throughput and reliability your enterprise needs from WLI.

Complete Visibility into Business Processes

No more 'black box' applications in production. Acsera shows business processes and their complete hierarchy.

Root Cause Analysis on Running Applications

No more 'mystery performance problems' on WLI. Acsera helps you diagnose and isolate problems while in production.

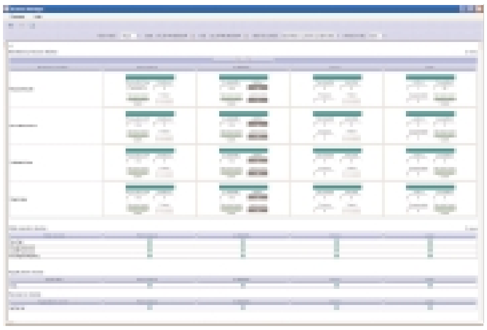
Manages Across Clusters

Your WLI applications deploy across nodes and clusters – don't be trapped into managing with server-by-server tools.

Results in Minutes

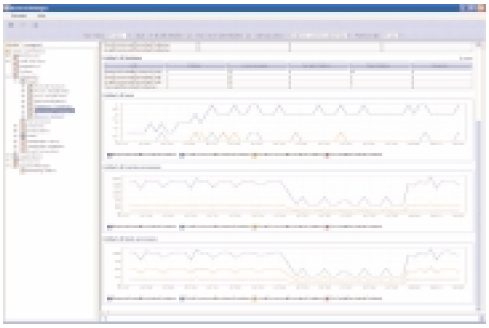
You can't afford to wait hours or days to find WLI problems and correct them – Acsera delivers results in minutes.

Attend a WebLogic Integration performance webinar:
www.acsera.com/register



Performance Management Console
Shows application performance and health.

Application Blueprint Console
Gives 100% visibility into application hierarchy.



Root Cause Analysis Console
Isolates performance problems in minutes.

ACSERA

Application Monitoring and Performance
Management for BEA WebLogic™ 8.1

BEA WebLogic is a trademark of BEA Systems, Inc.
Acsera is a trademark of Acsera Corporation.

Enabling Next-Generation Portals

BRING INFORMATION AND SERVICES FROM DISPARATE SYSTEMS TOGETHER

BY **PATRICK CHANG**

AUTHOR BIO...

Patrick Chang is a senior product manager at Wily Technology, the leader in enterprise application management. He is responsible for defining product strategy, driving product specification, and working with industry partners to develop innovative management solutions for the enterprise application market.

CONTACT...

pchang@wilytech.com

Enterprise portals are fast becoming the foundation of the Web-based economy thanks to their ability to give enterprises, trading partners, and customers global access to enterprise applications, back-office systems, and IT infrastructures. This ability has made enterprise portals appealing as the infrastructure of choice for enterprise IT organizations and has helped organizations justify the considerable expense of migrating from legacy systems.

However, what Web-based systems, such as portals, gain in global access and economy, they can sacrifice in stability and manageability compared to traditional closed, point-to-point systems. Portals are subject to the availability and performance problems that come with reliance on a Web-based infrastructure that is publicly shared, constantly changing, and ever growing.

This article discusses how performance management systems can be used to enable the development of critical, next-generation portals that are reliable and bulletproof.

Portals: Providing a Multi-Functional, Unified Structure

Despite the popularity of Web-based systems, Web technology is still in its infancy. The Web provides global desktop access to the corporate network, but without some organizing principle, this access can be merely an amorphous view of legacy systems and applications that have been hastily ported for Web delivery to a desktop already cluttered with any of a million other offerings available on the Web. By providing core business functionalities in a single integrated solution, enterprise portals such as BEA WebLogic Portal supply the organizing principle and technology needed to make sense of the desktop view of enterprise Web-based networks.

Companies typically deploy two types of portal solutions: external portals and internal portals. External portals provide organized access to a company's products and services. Internal portals integrate the internal systems, applications, and information services of an enterprise by providing a common desktop interface and the ability to customize the interface to suit a particular department or role.

All portals require certain core functionalities such as a Model/View/Controller (MVC) paradigm and flexible controller implementation, user and group management, entitlements, personalization, security, and content management. Some portals also provide extendable rules engines, shopping carts, pricing systems, and targeted content engines.

Sophisticated, large-scale portal applications built with BEA WebLogic Portal provide customized access to content by user/group using complex, dynamic business logic and also maintain persistent session/transaction information, permissions/entitlements, and context. For example, to support a shopping cart or airline reservation, the portal must provide access to the appropriate internal databases and applications for the duration of an extended transaction. When the user clicks "submit" in one pane/tab, various options must be updated in other panes/tabs and customized based on the information submitted. The portal maintains this complicated context for user sessions and transactions. High-performing portals provide all the elements above to rapidly and reliably deliver useful information and services; enabling financial transactions, tracking internal processes, or allowing access to information systems and internal applications on a customizable basis.

Managing Portal Complexity

Increasingly, vendors such as BEA are providing "out-of-the-box" portal functionality that enables development teams to put together much more complicated and higher-functioning applications on the same schedule and budget previously allocated for simpler applications. By providing a common integrated application framework, portals make modular development possible (e.g., separate development groups might work on separate "tabs" of an application). But as the underlying IT system becomes more complex, managing, identifying, and controlling system failures can become more difficult.

Portal-based applications usually run in the same environment as other Web applications and are given to the same teams to manage. However, when performance issues arise, the sheer number of components and transactions can overwhelm the IT staff members who must deploy and manage portal applications, fine tune them for performance, monitor the entire system for emerging trouble, diagnose problems, and rapidly solve them with a minimum of downtime.

Typically, these problems do not surface for the portal administrator until user complaints have come in; that is, after the problem has become severe enough to impair portal functionality. When problems do occur, their source is not always obvious. It may be in a portlet, a database, or any of the supporting back-end systems. Isolating and resolving problems can become a mind-numbing procedure of rebuilding and analyzing every portlet in its own "test" window after the failure has occurred. Portal administrators need portal management tools that let them "look under the hood" at internal portal processes and their interactions. Without such tools, isolating performance problems will become increasingly difficult.

To help them maintain high performance and 24x7 availability, portal administrators need effective management tools built specifically for monitoring not only individual portal components and workflow processes, but also the environment surrounding the portal. This includes connections to databases, transaction servers, mainframe systems, and other back-end systems. The tool portal administrators choose should:

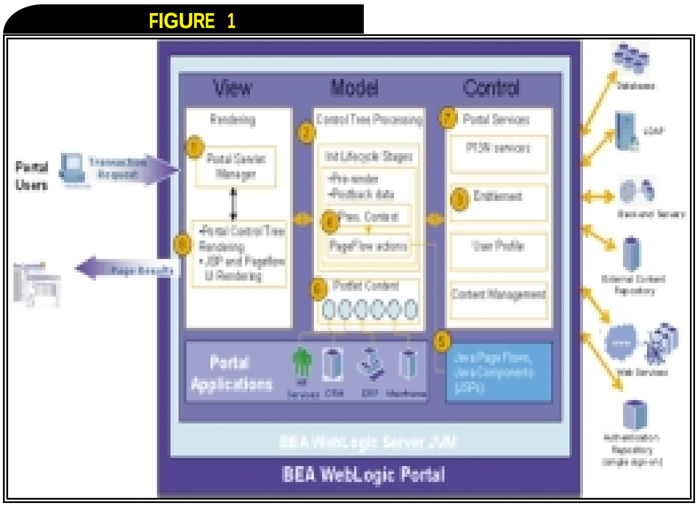
- Monitor the complex, dynamic interactions taking place across the entire workflow and within individual processes
- Present the resulting data in a clear, simple display that highlights problems (and where they occur in the portal workflow) and allows the administrator to drill quickly down – to individual portlets and transactions, if need be – to the source of the problem
- Summarize overall performance as well as performance in the key portal workflow areas: portal servlet, control tree processing, JSP backing files, Java Pageflows, portlets, connections to back-end systems, and portal services

BEA WebLogic Portal: How It Works

BEA WebLogic Portal is one of the industry's most comprehensive portal offerings and many enterprises are using it to develop and deploy critical portal applications. It contains a wide range of technologies for developing and maintaining sophisticated business-to-customer (B2C), business-to-business (B2B), and business-to-employee (B2E) portals. To better understand the requirements for monitoring WebLogic Portal, it will be helpful to discuss in general terms how it works.

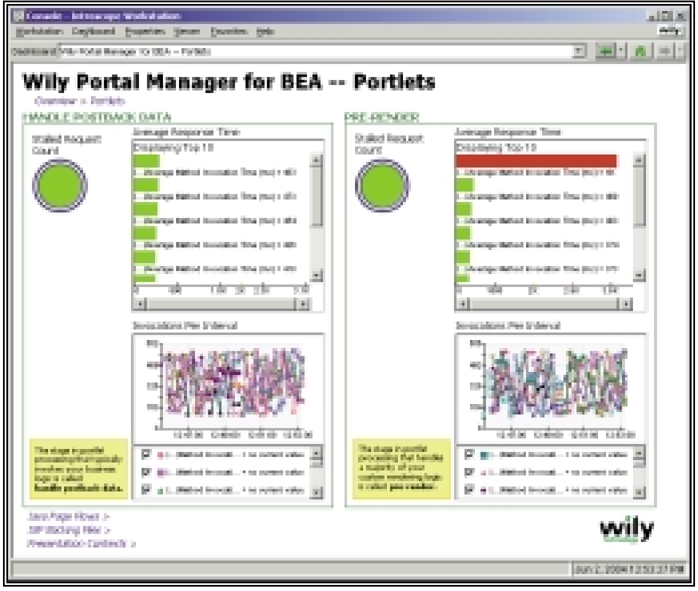
As shown in Figure 1, the Portal Servlet Manager (1) fields the incoming user request, which in turn initiates Control Tree processing (2). The Control Tree represents all the structural elements in the portal. It also serves as the foundation for building a new portal page. During Control Tree processing, a new Control Tree is

FIGURE 1



Portal Servlet Manager

FIGURE 3



A customizable graphical interface allows WebLogic Portal administrators to monitor key portal resources and to set predictive alarms so that problems can be identified before calls come in to the help desk.

TABLE 1

KEY AREAS TO MONITOR

WHAT	WHY
COMPONENTS & WORKFLOW	
Java Components	Individual J2EE components can affect portal performance and availability.
Portlet Servlet	The Portlet Servlet handles all incoming requests. Its performance and availability indicate system health.
Control Tree Processing	The Control Tree represents all of the structural elements in the portal and acts as the infrastructure on which a new portal page will be built. Almost all of the elements in the Workshop design GUI correspond to controls in this tree. It is necessary to have visibility into the complex processes taking place within the control tree as well as its interactions with the "View" and "Control" elements of the portal.
JSP Backing Files	Backing files work in conjunction with JSPs. The JSPs allow the developer to code the presentation logic, whereas the backing files allow the developer to code simple business logic. Backing files are always run before the JSPs. Typically, a lot of custom rendering code is here (additionally, some developers make callouts to back-end systems to fetch additional data to render). Poor performance is usually an indication of misbehaving custom rendering code.
Java PageFlows	The page flow itself is entirely defined by the developer. Slowness in the Java PageFlow can usually be diagnosed by the author, and isn't usually caused by trouble with any back-end system. It may also be helpful to correlate the J2EE standard page flows to the portal control tree processing architecture to make a determination which page flow is tied to which desktop.
Portlets	Portlets are one of the basic elements of any portal solution. Applications, JSP-based portlets, Web services, or other available J2EE resources can all be exposed as portlets. If a performance slowdown occurs, application support personnel need the ability to determine quickly which individual portlets may be responsible. Within the portlet lifecycle, handling postback data and pre-rendering are processes that are especially important to monitor for performance.
Connections to back-end systems	The power of portal solutions is that that they allow enterprises to bring information and services from disparate systems together in one interface. This requires communication with a number of systems outside of the portal itself. Effective performance management of the portal must therefore include monitoring connections to back-end systems such as databases, transactions systems, mainframes, and more.
PORTAL SERVICES	
Entitlements	The entitlement system provides role-based authorization to individual portal resources. Naturally, there are a lot of API calls involving roles. Entitlements are used heavily by all aspects of the portal, so any slowness impacts the whole system. Often, delayed responses and stalled threads are caused by trouble in the back-end systems supporting entitlements, such as LDAP. Additionally, entitling too many objects at a fine level of granularity can overload the Entitlements system.
Personalization	This service, implemented through advislets, modifies the information displayed in the portal preferences. Advislets can use many mechanisms: an internal rules engine, explicit personalization, or even events. Overuse of the Personalization system overall is a common cause of performance problems.
User Profiles	This is the repository for additional user information, anything from contact information to favorite ice cream flavor. Often, delayed responses and stalled threads are caused by trouble in the back-end systems, such as a database used for supporting User Profiles.
Content Management	This is a common API that can interface with a number of commercially available content management systems. The Content Management API provides a uniform interface for back-end communication with these content management systems. When stalled threads occur here, one of the first things to check is whether the back-end content system is performing normally.

Key Areas for Portal Monitoring

created (or drawn from cache if it already exists). The Control Tree is built throughout its life-cycle phases, coordinated by the Lifecycle Manager and the Control Tree Walker.

As a Control Tree is built, user Entitlements (3) are verified to determine which portal resources are available to the user. During the pre-render and postback data operations (the two primary operations during the life-cycle), the Presentation Context (4) is called. In the meantime, JSP backing files are also checked for custom code written by the development team. The Presentation Context, in turn, calls Java Page Flows and JavaServer Pages (JSPs; 5). The Control Tree then pulls data from individual portlets as necessary, depending on the type of user request. Portal services (6) such as personalization, content management, and user profiles are then initiated as needed. The final page is built and rendered as the processes described here are completed.

Resolving Complexity with Simplicity

In the workflow described previously, several areas are potential sites for problems that can affect portal performance and availability. An effective portal management system can help developers and portal managers identify the exact location and root cause of errors regardless of where in the environment the error occurs. When portal management systems are used throughout the portal lifecycle, developers can use the data to proactively monitor and manage session and transaction information, permissions, entitlements, and context in production, QA, and development environments. The result is a more stable and reliable portal. (See Table 1 for a list of the important BEA WebLogic Portal components and processes that administrators should monitor.)

Using customizable management UIs, administrators can rapidly assess the performance and availability problems described previously and immediately take corrective action. For example, by displaying key performance data in easy-to-use dashboards, portal managers can immediately view overall portal performance, as well as performance in key workflow areas. Traffic lights, time plots, and worst performing graphs plainly show administrators where to focus their troubleshooting efforts. By simply clicking on a tree view, administrators can drill down to the root cause, precisely isolating performance issues in individual components.

Administrators can use a portal management tool to react to problems more quickly. However, businesses have shown an increasing desire to use management tools in a more proactive manner. For example, a management tool can be used to anticipate problems by establishing baseline settings and border conditions for portal performance. Using this information, alarm thresholds can be assigned to target specific workflow processes so they can be proactively monitored (see Figure 2). Not only can administrators improve portal performance by monitoring and managing portals throughout the application lifecycle, but performance information can be used over the long term for trend analysis and capacity planning.

Conclusions

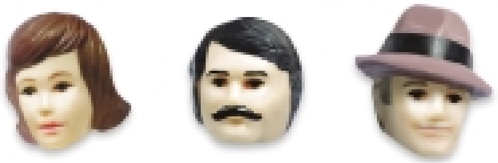
Enterprise portal solutions such as WebLogic Portal enable enterprises to successfully deliver Web-based IT services to customers, partners, and employees through a single, unified interface. However, as portals become more complex, so does the challenge of managing portal performance and availability. By adopting a portal management solution built specifically for monitoring and diagnosing portal performance, businesses can accelerate the resolution of portal-related errors and proactively deploy next-generation portals that deliver superior service and functionality.

YOUR TESTING SUCKS.

1. Writing test code isn't testing – it's coding.
2. Your developers aren't making testable software.
3. It's impossible to get the business side involved in quality.
4. Your existing testing tools have become shelfware.
5. Your customers have become the primary testers.

LISA from iTKO.

Because everyone should own quality.



iTKO Try a free eval of LISA 2.5 at **www.itko.com**

We understand.

You've probably invested considerable time and money trying to automate your testing. But it just didn't work.

You were sold a tool that let you point and click to test on Day 1, and on Day 2 you were maintaining a test script - inserting bugs into your tests at the same rate as you inserted bugs into the code during development.

The tools were too hard for the QA teams to use, they required intense training and developer knowledge, and the seats were more expensive than a presidential fundraiser.

It's OK. We've seen this scenario at every customer we talk to.

LISA is different.

LISA is a comprehensive, no-code automated testing solution ideal for distributed applications and teams.

Using "inline testing" technology, LISA allows even non-developers to instrument and test web services, J2EE apps, databases and websites.

LISA automates test design, staging and monitoring for unit, regression, load and performance tests. And since LISA is pure Java, she's equally stylish on Linux, OS X, Unix or Win clients.

But don't take this ad's word for it. Try LISA for free and talk to the experts at iTKO. We'll show you how everyone can own quality.

Security Best Practices

SERVER SAFETY IS FIRST

BY **BALAMURALI KOTHANDARAMAN**

AUTHOR BIO...

Balamurali Kothandaraman is a delivery technologist for education services at BEA Systems Inc. He has over 7 years of experience in Java and J2EE technologies and is a BEA Certified Server Specialist, Administrator, and Instructor. Bala is a frequent speaker at various conferences, including eWorld and JavaOne.

CONTACT...

bala@bea.com

The demands of security have gone far beyond simply managing user accounts and restricting access between internal and external networks. Emerging business practices challenge the enterprise to create flexible and robust security architectures that meet rapidly changing business demands. This article provides best practices tips that an administrator can implement to secure an enterprise. The land mines are highlighted so you know what to avoid.

The open, flexible, easy-to-configure security architecture of BEA WebLogic Server delivers advantages at multiple levels and introduces an advanced security design for application servers. Systems have to be protected from both insiders and outsiders. Misuse of a system or improper administration can leave it open for intrusion or attack.

Using development mode for domains running in production mode is not recommended. Never use development mode for production servers; it relaxes the security constraints for *all* servers in a domain. If you are using compatible security (using 6.x security configuration in 8.x), disable guest logins in production so that they cannot be used to access WebLogic resources in a WebLogic Server domain.

SerializedSystemIni.dat contains hashes for the passwords in a domain; ensure that you store a copy of this file in a safe place. Give read privileges for SerializedSystemIni.dat *only* to the WebLogic system administrator account. If you lose the administrative password, and the boot identity is not stored in the form of a boot.properties file, you cannot restart your servers. Later, I will explain how to recover the administrative password if you lose it.

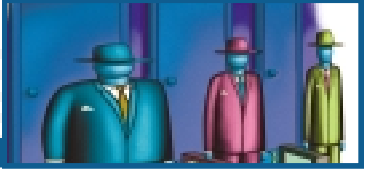
While creating scripts to automate WebLogic administration tasks, the option –Dweblogic.system.BootIdentityFile lets you avoid hard-coding a username and password in your text script. Also, more than anything else file system security of the WebLogic installation directories is most important. Allow access only to the user account that will be used to run or administer the WebLogic domain.

Tips

- Install JRE instead of the complete SDK.
- Remove or delete the development tools provided by BEA, such as the Configuration Wizard, WebLogic Builder, and jCOM tools.
- The Pointbase database provided with WebLogic installation is for evaluation purposes and not supported in a production environment. Delete it.
- Do not install the sample domains while installing in production or delete the sample domains if already installed.
- Instead of hard-coding user identity in the start scripts, store the encrypted boot identity of the user who has privileges to start WebLogic Server in the boot.properties file.
- BEA recommends using security roles (rather than users or groups) to secure WebLogic resources; assign users to groups, then create role statements.
- Do not install or run WebLogic Server software as root. If you must bind to a privileged port, use postbind UID or postbind GID in the WebLogic machine configuration.
- Set the ownership of the WebLogic installation and applications directory for access only by the user account that runs the server.

J2EE Application Security

Changing a security policy defined in a J2EE deployment descriptor requires redeployment;



changing an embedded LDAP policy in the admin console is dynamic. Create application policies using the WebLogic Administration Console. When you create a security policy using the administration console, if inherited policy statements are present in the Inherited Policy Statement box of the Policy Editor page, the new policy overrides them.

Always keep source code off the production machine. Do not install uncompiled JSPs and other source code on the production machine. Configure your applications to use SSL. Set the transport- guarantee to CONFIDENTIAL in the user-data-constraint element of the web.xml file to secure specific resources of the web application using SSL.

Recovering Administrator Password

WebLogic provides four default administrative roles. The administrative user created during domain creation will be a part of the "Admin" role. Configure additional administrative users to roles such as Admin, Deployer, Monitor, or Operator. When using the default authenticator, if you have not modified the global Admin role (which by default is granted to the Administrator's group), you can recover the administrator password in a WebLogic domain.

To recover the administrator password in a WebLogic domain:

- At the command line, change directory to the domain and run the setEnv script to set the PATH and CLASSPATH.
- Create a new DefaultAuthenticatorInit.ldift: run java weblogic.security.utils.AdminAccount <tempadmin> <temppassword>.
- Remove the initialized status file, DefaultAuthenticatormyrealmInit.initialized from the <Domain>/<Server>/ldap subdirectory.
- Restart the server, using the new user identity.
- To change the old admin user identity, log into the admin console (Optional).

SSL

When using SSL with WebLogic Server, use *keystores*; storing identity (private keys and certs) and trust (CA) in *files* is deprecated. Migrating from an earlier version might require you to create keystores from private keys, certs, or trust files.

If the network that connects WebLogic Servers in a domain is not trusted, enable SSL on each server in the domain so that LDAP replication between the admin server and managed servers uses SSL connections. Enabling the administration port of the domain enforces all the servers in the WebLogic domain to use SSL.

The default WebLogic installation represents exportable-strength SSL implementation (the maximum SSL strength is 512-bit keys with 40-bit bulk encryption). Key lengths longer than 512 bits require a domestic-strength SSL license key from BEA. If you use SSL in your production environment, use high-strength SSL. Key lengths of less than 1024 bits are generally considered weak.

SSL hardware accelerators: Running SSL on the WebLogic Servers is a tremendous drain on server resources. By offloading SSL processing, the resources can be applied to WebLogic functions. SSL processing can be handled by Web servers, load balancers, firewalls, or switches.

Incoming connections in WebLogic Server can be controlled by filtering them. WebLogic Server provides a default implementation of connection filter that you can configure in the admin console.

TIPS

- To avoid compromising application security, install and configure server-specific SSL certificates and enable host-name verification on production servers.
- In production, do not use the sample SSL certificates that are provided with WebLogic.
- Use a load balancer with built-in secure sockets layer (SSL) support, or run WebLogic Server on a machine that has SSL hardware, with Java Cryptography Extension (JCE).
- Use SSL with WebLogic Server only if it is necessary. SSL degrades performance.
- To control the types of connections accepted by WebLogic Server instances, use Connection Filters.

Securing the Admin Console

Applications should be installed on the managed server rather than installing them on the administration server. But if you use the admin server to serve applications (or in a single-server domain), do the following for better security:

- Change the default admin user and password to custom.
- Change the admin console context root path.
- Enable domain-wide administration port.
- Consider disabling the admin console.

Authentication Providers

If you use an external LDAP provider, store the server boot identity in the embedded LDAP server, and set timeouts on the external LDAP authentication provider. This way, if the external LDAP server is unavailable you can continue to restart and to serve unprotected data with WebLogic Server. Also before you apply any changes, set the control flag for all authentication providers to OPTIONAL; this prevents a configuration error from causing a production server not to restart.

WebLogic Server provides a custom realm, the NTRealm, based on older security realm API that supports native Windows domain authentication. NTRealm is useful with Windows domains that are not set up to use Active Directory.

Tips

- Store the server boot identity in the embedded LDAP server.
- For finer control of a production environment, use Active Directory authentication, rather than native Windows domain (NTRealm) authentication.
- To prevent denial-of-service attacks, modify the timeout and maximum-size values for the incoming protocol ports (T3, COM, IIOP, HTTP Post time out) on the server.
- Have a security audit performed by an internal or external auditing group.

Backup and Recovery

To migrate or recover WebLogic domains in case of failures, periodically back up the entire domain directory tree from the administration server machine. This way you can recover from a hardware or system failure by merely restoring the domain directory and restarting the admin server. Some important files to be watched/backed up periodically from the administration server machine in a WebLogic production domain are:

- ***config.xml:*** Domain configuration repository.
- ***config.xml.booted:*** Good backup of domain configuration repository from a successful boot.

continued on page 48

Strategies for WebLogic Domain Configuration

MANUAL AND TEMPLATING OPTIONS, PART 1

BY **PRAKASH MALANI**

AUTHOR BIO...

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

CONTACT...

pmalani@malani.org

A domain contains configuration information for a BEA WebLogic Server instance. It has configuration information about servers, clusters, and machines. A domain also contains configuration information about resources such as Java DataBase Connectivity (JDBC) connection pools, JDBC data sources, connection factories, and Java Message Service (JMS) queues. In addition, it contains configuration information about the applications deployed to the instance. The domain configuration information is persisted in a config.xml file. In this article, I evaluate various strategies for WebLogic domain configuration.

The config.xml can be directly manipulated (e.g., hand-editing the file). However, most tools evaluated in the article will manipulate the config.xml indirectly. Most tools (e.g., WebLogic Console) manipulate Java Management eXtension (JMX) Management Beans (MBeans). The WebLogic Server instance persists the state of the MBeans in the config.xml file.

Domain Configuration and JMX

The config.xml file contains persistent information about a WebLogic Server instance. When a WebLogic Server instance is started, WebLogic Server reads the config.xml file and deploys the resources and applications. The resources and applications have corresponding JMX MBeans. The server exposes configuration information via these MBeans. There are many different types of MBeans. In this article, I focus on MBeans that correspond to entries in the config.xml file.

When an attribute value of an MBean is changed at runtime, the new value is made persistent by storing it in the config.xml file. For example, when an MBean attribute value is changed in the WebLogic Server Console, the attribute value is also saved in the config.xml file. Thus, the configuration attribute value is available between server restarts since the configuration attribute value is persisted in the config.xml file.

The following section describes different alternatives for domain creation and configuration, and classifies the different options available.

Options, Options, and More Options

Fortunately, many different ways to create and configure a domain are available. The most common way to create a domain is through the Domain Configuration Wizard and the most common way to configure the domain is by using the WebLogic Console. Many other alternatives are possible, including:

- Using domain configuration wizard
- Using domain configuration template
- Using silent scripts

- Using WLShell scripts
- Using WLST scripts
- Using Java code and the JMX Application Programming Interface (API) directly
- Using a token replacement (e.g., Bedrock)
- Using weblogic.Admin command
- Using wlconfig and wldeploy Ant tasks
- Hand-editing the config.xml file

The following is a way to classify the above options based on the technique utilized for domain configuration:

- ***Manual:*** WebLogic Console, Complete Domain Configuration Wizard
- ***Template:*** Custom token replacement scripts (e.g., using Ant), Bedrock, Complete Template, Extension Template
- ***Scripting:*** Silent, WLShell, WLST, Ant
- Programmatic using Java

Another way to classify the above options is by whether they ship with the WebLogic platform:

- ***Included (and supported):*** WebLogic Console, Complete Domain Configuration Wizard, Complete Template, Extension Template, Silent Script, Ant
- ***Additional:*** WL Shell, WLST, Custom token replacement scripts, Bedrock

Yet another way to classify these options is by whether they make changes while the server is online or offline. Online changes are made while an instance of the server is running. The online changes are made to JMX MBeans and persisted to config.xml file by the server. Modifying the config.xml directly while the server is running is not recommended!

- ***Online:*** WebLogic Console, *ANT*, WLST
- ***Offline:*** Complete Domain Configuration Wizard, Complete Template, Extension Template, silent scripts, WLShell, custom token replacement scripts, Bedrock

In order to demonstrate various strategies for domain configuration, you will create a domain and make changes to it.

High-Level Objectives

You will achieve the following objectives using different strategies for domain creation and configuration. Please do not follow these steps right now. I will ask you to create and configure the domain later employing a different strategy each time.

1. Create a WebLogic Workshop domain
2. Change the server debug level to info.
3. Change the server to enable debug to stdout
4. Configure a JDBC Connection Pool named semJDBCConnectionPool
5. Configure a JDBC data source named jdbc/semJDBCDataSource with JNDI name jdbc/semJDBCDataSource
6. Configure a JMS Connection Factory named semJMSConnectionFactory
7. Configure a JMSJDBCStore named semJMSJDBCStore
8. Configure a JMSServer named semJMSServer
9. Configure semJMSServer's store to semJMSJDBCStore
10. Configure a JMS Queue named PersonQueue with JNDI name jms/PersonQueue
11. Deploy the sample application: SEMApp Enterprise ARchive (EAR)
12. Verify the domain configuration by testing the application

For each strategy, certain steps such as domain creation, database configuration, and verifying the configuration are common. The following sections describe these steps.

Domain Creation

Follow these steps to create a WebLogic Workshop domain using the Domain Configuration Wizard. Do not perform these steps right now. I will ask you to perform these steps later when I explain the different strategies in detail.

1. Launch Domain Configuration Wizard (e.g., on Windows select Start > All Programs > BEA WebLogic Platform > Configuration Wizard)
2. Select Create a new WebLogic Configuration and click Next
3. Select Basic WebLogic Workshop Domain and click Next
4. Select Express and click Next
5. Specify User name as system
6. Specify User password and Confirm user password as password and click Next
7. Select Development Mode and click Next
8. Specify Configuration Name as something meaningful and click Create
9. Click Done

Before you can configure the domain, you need to configure the database. The database configuration occurs after the domain is created, but before the domain is configured.

Database Configuration

All the examples use PointBase as the database. Starting an instance of WebLogic Server (of Workshop Domain) also starts PointBase. Set up user, schema, and tables in that instance of PointBase. Do not perform these steps right now. I will ask you to perform these steps later when I explain the different strategies in detail.

1. Verify that an instance of WebLogic Server is running
2. Launch PointBase Console (e.g., on Windows select Start > All Programs > BEA WebLogic Platform > Examples > WebLogic Workshop > PointBase Console)
3. In the Connect to the database dialog specify the following
4. Verify that the database URL is jdbc:pointbase:server://localhost:9093/workshop
5. Verify User as WebLogic
6. Specify Password as WebLogic
7. Create a new user (e.g., Ctrl + u) named sem with password as sem
8. Disconnect from the database (DBA Menu and Disconnect from Database)
9. Connect to database (DBA Menu and Connect to Database or F3) using sem as the User and sem as the Password
10. Create a schema (e.g., Ctrl + h) named sem
11. Copy and paste contents of db/sem.sql into Enter SQL Commands window
12. Select Execute All toolbar item
13. Verify that sem schema is populated with tables

Once the domain is created, the database is configured, and the domain is configured with resources and the sample application, the set-up needs to be verified.

Verifying the Domain Configuration

To verify that the domain is properly created and configured, run simple JUnit tests. The included source code example contains a sample application as well as an Ant file that runs JUnit tests. Copy sample-build.properties to build.properties and edit the entries in build.properties to suit

> "Online changes are made while an instance of the server is running"

your environment. The creation of build.properties needs to happen only once. I will ask you to invoke the following command later, once you have created and configured a domain.

```
ant – v code fact.
```

If the tests run successfully, then the application is deployed properly and the resources are configured correctly. If there are any exceptions and the tests fail, then the application or the resources are not configured or deployed correctly. Carefully review and analyze the stack traces and proceed to debug the problem.

Once the tests run successfully, shut down the WebLogic Server instance e.g., the domain directory/stopWebLogic.bat or sh). Always shut down the WebLogic Server instance using the script instead of drastic means such as ctrl+c.

Manual Express Configuration

Using the steps described in the Domain Creation section, create a new domain with SEMDomain-Manual-Express as the Configuration Name. Start the WebLogic Server instance and use the steps described in the Database Configuration section to configure the database.

On many different projects, I have successfully done manual configuration using the WebLogic Server Console.

Changing the Logging Level

1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Servers / cgServer
5. Select Logging tab
6. Check Debug to Stdout checkbox
7. Select Stdout Severity Threshold as Info from the drop down list
8. Select Apply

Configuring the Connection Pool

1. Make sure WebLogic Server is running.
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Services/JDBC/Connection Pools
5. Select Configure a new JDBC Connection Pool
6. Select PointBase from the Database Type dropdown list
7. Select PointBase's Driver (Type 4) Versions: 4.X from Database Driver
8. Select Continue
9. Specify Name as semJDBCConnectionPool
10. Specify Database Name as workshop
11. Specify Host Name as localhost

12 Specify Port as 9093

13. Specify Database User Name as sem
14. Specify Password as sem
15. Specify Confirm Password as sem
16. Select Continue
17. Test the driver configuration
18. Create and deploy

Configuring the DataSource

1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)

3 Log into the Console

4. Select Services / JDBC / Data Sources
5. Select Configure a new JDBC data Source
6. Specify Name as jdbc/semJDBCDataSource
7. Specify JNDI Name as jdbc/semJDBCDataSource
8. Select Emulate Two-Phase Commit for non-XA Driver.
9. Select Continue
10. Specify Pool Name as semJDBCConnectionPool
11. Select Continue
12. Target the data source by clicking on Create

Configuring the JMS Connection Factory

1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Services / JMS / Connection Factories
5. Select Configure a new JMS Connection Factory
6. Specify Name as semJMSConnectionFactory
7. Specify JNDI Name as jms/semJMSConnectionFactory.
8. Select Create
9. Target and deploy on cgServer
10. Select Configuration tab
11. Select Transactions sub tab
12. Select XA Connection Factory Enabled
13. Select Apply

Configuring the JMS JDBC Store

1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Services / JMS / Stores in the left navigation pane
5. Select Configure a new JMS JDBC Store
6. Specify Name as semJMSJDBCStore
7. Select Connection Pool as semJDBCConnectionPool
8. Select Create

Configuring the JMS Server

1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Services/JMS/Servers in the left navigation pane
5. Select Configure a new JMS Server
6. Specify Name as semJMSServer
7. Specify Persistence Store as semJMSJDBCStore by selecting it from the dropdown list
8. Select Create
9. Target the semJMSServer to the cgServer

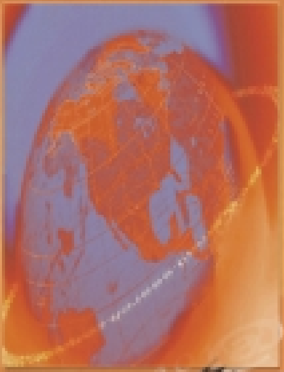
Configuring the JMS Queue

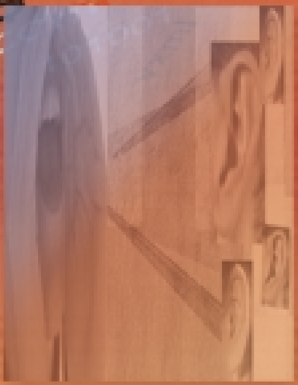
1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Services/JMS/Servers/semJMSServer /Destinations in the left navigation pane
5. Select Configure a new JMS Queue
6. Specify Name as PersonQueue
7. Specify JNDI Name as jms/PersonQueue
8. Select create
9. Select the Redelivery sub tab
10. Specify Redelivery Limit as 3
11. Select Apply

Now you are done configuring the application resources. However, before you can deploy the application, you still need to configure workshop-related resources.

Configuring Workshop-Related Resources

1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Services/JMS/Servers/semJMSServer/ Destinations in the left navigation pane
5. Select Configure a new JMS Queue
6. Specify Name as SEMAppWeb.queue.-AsyncDispatcher_error.
7. Specify JNDI Name as SEMAppWeb.-queue.AsyncDispatcher_error
8. Select Create
9. Select the Redelivery sub tab
10. Specify Redelivery Limit as 0
11. Select Apply





12. Select Services/JMS/Servers/semJMSServer/ Destinations in the left navigation pane
13. Select Configure a new JMS Queue
14. Specify Name as SEMAppWeb.queue.-AsyncDispatcher
15. Specify JNDI Name as SEMAppWeb.-queue.AsyncDispatcher
16. Select Create
17. Select the Redelivery sub tab
18. Select Error Destination as SEMAppWeb.queue.AsyncDispatcher_error
19. Select Apply

You are now ready to deploy the application.

Deploying the Application

1. Make sure WebLogic Server is running
2. Launch the WebLogic Server Console (e.g., http://localhost:7001/console)
3. Log into the Console
4. Select Deployments/Applications in the left navigation pane
5. Select Deploy a new Application
6. Browse to the location of the SEMApp.ear file
7. Select SEMApp.ear
8. Select Continue
9. Select Deploy

Once the application is successfully deployed, restart the server. Run the tests as described in the Verifying the Domain Configuration section.

Other manual strategies are possible using the Domain Configuration Wizard. One option is to create and configure a brand new domain by selecting the Custom (instead of the Express) option. When you are prompted to create resources such as JDBC Connection Pools in the wizard, follow the steps to create and deploy the resources. Another option is to configure an already created domain using the Domain Configuration Wizard. In the very first step of the Domain Configuration Wizard, select Extend an existing WebLogic configuration.

> "A special tool named Configuration Template Builder helps with template creation"

Creating the domains as described above is left as an exercise for you. However, not all of the configuration is possible using these strategies, such as configuring a destination's Redelivery attribute.

In order to use the manual strategies successfully, a project needs to maintain a configuration guide. The configuration guide contains the step-by-step instructions as described above. However, the manual process is tedious and time consuming. Setting up another domain takes extra time and effort. Moreover, the process is error-prone. Small configuration errors, such as misspelling a JNDI name, can lead to serious troubleshooting and debugging.

Automated options to create and configure a domain are preferred. Java is a powerful language and the MBeans can be directly manipulated with it. However, this approach is too low level. There are alternative approaches such as templating and scripting. In manual templating, the config.xml is templated with tags. The tags are later replaced with actual values. The manual templating approach requires a very deep and thorough understanding of config.xml file. I have used manual templating on some projects. Bedrock is an Ant-based tool that supports such templating. However, Bedrock does not seem to be actively maintained. Another important templating technique is using a domain template. This technique does not require an in-depth knowledge of the config.xml. The creation of templates is nicely supported via the Configuration Template Builder tool. The next section discusses automating domain configuration via templating using the Configuration Template Builder tool.

Domain Template

When you created the domain manually, you used the workshop template. But, what is a template? A template is a Java ARchive (JAR) file that contains configuration information; a relevant snippet of config.xml file. There are two types of templates: complete and extension. With a complete template, a new domain is created using the template. The workshop template used above is an example of a complete template. With an extension template a domain is extended. The DefaultWebApp template used later is an example of an extension template.

How do you create a template? A special tool named Configuration Template Builder helps. A template, either complete or extension, is created in two ways. First, a template is an extension to another extension template. Second, a template is based on an existing domain. In the next section, you will create a complete template based on an existing domain.

Complete Template

In this section, I will illustrate creation of a complete template based on an existing, preconfigured, and working domain. Follow these steps to create a complete template.

1. Launch Configuration Template Builder (On Windows select Start > All Programs > BEA WebLogic Platform > Other Development Tools > Configuration Template Builder)
2. Select Create a Configuration Template and click Next
3. Select Basic WebLogic Workshop Domain and click Next
4. Select an existing, preconfigured, and working SEM domain (e.g., SEMDomain-Manual-Express) and click Next
5. Specify Name as SEMTemplate-Complete
6. Specify Author (e.g., Bart)
7. Specify Category (e.g., SEM)
8. Specify Description and click Next
9. Take the defaults by clicking Next on the subsequent dialogs
10. In Create Template, specify a name for the template (e.g., SEMTemplate-Complete), observe and remember the location of the template JAR file %BEA_HOME%/user_templates/, Click Create
11. Click Done

Now, create a new domain based on the complete template. Follow these steps to create a complete domain.

Domain Creation Based on Complete Template

1. Launch Domain Configuration Wizard (On Windows select Start > All Programs > BEA WebLogic Platform > Configuration Wizard)
2. Select Create a new WebLogic Configuration and click Next.
3. Instead of selecting one of the default templates, browse to location of complete template (%BEA_HOME%/user_templates)
4. Select SEMTemplate-Complete and click Next.

–continued on page 38

diagnoSys

Are your Web applications open 24 hours? Get the tips and best practices you need to keep your apps running at peak efficiency 24 hours a day with WebApp News, the free newsletter from the performance experts at H&W.

Sign up today at:
www.hwcs.com/wldj08.asp

You might as well be closed for business if your Web applications aren't running at top speed 24 hours a day. With poor Web performance costing businesses $25 billion a year, you need to find problems, and you need to find them fast.

You need DiagnoSys.

DiagnoSys is the first intelligent performance management solution that analyzes your J2EE and .NET applications from end to end. DiagnoSys starts analyzing quickly, gathers data directly from all application resources, and proactively finds and helps fix the real cause of problems – before they cost your business big time.

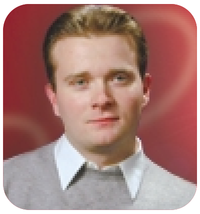
So when it's absolutely essential that your Web applications say "Open 24 hours," trust DiagnoSys.

H&W
www.hwcs.com | 1.800.338.6692

Bridging the Enterprise Computing Gap For 25 Years

Creating WebLogic Domains in Silent Mode

THE BENEFITS OF SCRIPTED DOMAIN CONFIGURATIONS



BY MARIUS LARSEN & VIDAR MOE

AUTHOR BIOs...

Marius Larsen is a senior consultant at Capgemini Norway, developing and configuring applications on the BEA WebLogic Platform and consulting with customers on architectural decisions within J2EE applications. He has been working with the BEA WebLogic Platform for several years.

Vidar Moe is a senior consultant at Capgemini Norway, providing technical consulting to customers on J2EE architectures and application development. He has been working with enterprise Java solutions on the BEA WebLogic Platform for several years.

CONTACT...

marius.larsen@capgemini.com
vidar.moe@capgemini.com

An enterprise development project deploying to the WebLogic platform needs to maintain several different domain configurations during the project's development, test, and deployment phases.

Version control and continuous rollout of ever-changing domain configurations to the different environments can be a challenging task. The Domain Configuration Wizard can help alleviate these challenges, because there is more to this Wizard than immediately meets the eye.

The WebLogic domain has grown from a nonexistent concept in WebLogic 5.1 to become the foundation of all enterprise applications running on the latest version of the WebLogic Platform. A domain defines vital parts of an application's runtime environment, such as deciding on whether or not the application will run in a cluster, and which physical databases the application will use through the connection pool definitions.

The domain concept appeared in WebLogic Server 6.0. There wasn't much help from the platform itself when it came to creating new domains. The suggested way of doing it was to make a copy of the default domain that came with the installation, start the server in the copied domain, and use the Administration Console to add changes to it. The first version of the Domain Configuration Wizard was introduced in WebLogic Server 7.0. This version of the wizard did not have support for either WebLogic Integration or WebLogic Portal domains. In the latest version of WebLogic there are numerous ways of creating a domain using various tools in the WebLogic installation: you can use the Domain Configuration Wizard, the WebLogic.Server command, or Ant scripts with the wlserver and wlconfig Ant tasks. This shows that BEA has taken the message from the developers seriously: help us make it easier to create WebLogic domains! In this article we will take a closer look at the Domain Configuration Wizard, focusing on how to use the least documented (until now) of its three running modes, the Silent mode. (In addition, as you probably know, it can be run in GUI mode and Console mode.)

The Domain Configuration Wizard has matured, and now covers the complete range of domains by default. The GUI mode is relatively easy to use, even if you have to be prepared to go through a lot of screens to configure, say, a clustered portal domain with a couple of connection pools and some JMS destinations. We feel that the main challenge with this approach in an enterprise project is maintaining several changing domain configurations for parallel environments like development, test, and production. On top of this, developers normally have their own local servers with local database settings and local JMS destinations. For all but the very smallest projects, we cannot base these domain configurations on manual creation through the Domain Configuration Wizard. We need to be able to repeatedly create our domains from scratch in a consistent way on various deployment targets. We also want to be able to put our various domain configurations under source control. Domain Configuration Wizard's Silent Mode gives us the opportunity to do just that through domain configuration scripting.

The Domain Configuration Wizard is located below WL_HOME/common/bin, where WL_HOME is BEA_HOME/weblogic81. Start it by running the config.cmd. To run it in Silent Mode, you set the mode parameter to silent, and tell it where to find your silent configuration script.

On Windows:

```
config.cmd -mode=silent -silent_script=<silent
script file>
```

On Unix/Linux:

```
./config.sh -mode=silent -silent_script=<silent
script file>
```

It all happens inside the silent script file. This script file contains the complete domain configuration. The language used in this file is fairly simple to understand, even if it is proprietary. It is well documented in BEA edocs.

The default installation comes with an example silent script located in WL_HOME/common/templates/domains / silent_scripts”. We'll now go through this example step by step.

A WebLogic domain can be one of five different domain types, accessible through domain templates. The first thing to do is to decide on the type of domain by choosing what domain template file we want to use. The domain template files are located in the “WL_HOME/common/templates/domains/” directory. The different domain templates are:

- Workshop domain (wlw.jar)
- Integration domain (wli.jar)
- Portal domain (wlp.jar)
- Server domain (wls.jar)
- WebLogic Platform domain (platform.jar).

The domain template files are included in the installation of BEA WebLogic Platform 8.1. For this example we chose a WebLogic Server domain; hence wls.jar is the template file. We now read in all the data that is included in the default server domain.

```
//Read in a domain template.
read template from
"WL_HOME/weblogic81/common/templates/domains/wls
.jar";
```

The default server domain has configured an admin server called “myserver”. We now set the address, the port, and SSL.

```
//Find and configure the Admin Server.
find Server "myserver" as s1;
set s1.ListenAddress "localhost";
set s1.ListenPort "7001";
set s1.SSL.Enabled "true";
set s1.SSL.ListenPort "7002";
```

We want to create a JMSQueue in the domain. Fist we create a JMSServer, then we create the JMSQueue and set the properties for this queue. Finally, we target the JMSServer to the “myserver” that is the default server in the “wls.jar” template file.

```
//Create a JMSQueue.
//A JMSServer has to be created first.
create JMSServer "myJMSServer" as jmsserver;
create JMSQueue "myJMSQueue" as myq;
//required attribute
set myq.JNDIName "jms/myjmsqueue";
//required attribute
set myq.JMSServer "myJMSServer";
//optional attribute
//set myq.StoreEnabled "false";
//target "myJMSServer" to server "myserver"
assign JMSServer "myJMSServer" to target
"myserver";
```

We want to create a connection pool in the WebLogic Server domain. In the example we use the PointBase database thet follows the BEA Weblogic Platform 8.1 installation. The connection pool is created and targeted to “myserver”.

```
//Create a JDBCConnectionPool.
create JDBCConnectionPool "demoPool" as mypool;
//required attribute
set mypool.DriverName
"com.pointbase.jdbc.jdbcUniversalDriver";
//required attribute
set mypool.URL "jdbc:pointbase:server://local-
host:9092/demo";
//required attribute
set mypool.Password "PBPUBLIC";
//optional attribute (but it's recommended you
set the db user...)
set mypool.Properties "user=PBPUBLIC";
//target all JDBC connection pools to server
"myserver"
assign JDBCConnectionPool "*" to target
"myserver";
```

It is possible to target applications to the server during configuration of the domain. The “wls.jar” template does not contain any application and hence this code is commented out:

```
//target existing applications.
//target applications only when they exist in
current domain template
//assign application "*" to target "myserver";

We now want to change the password for the
webloic user. We find the user and then sets
the password.
```

```
//Create the admin user and password.
find User "weblogic" as u1;
set u1.password "weblogic";
```

Now we'll write the domain to disk. The domain is written to the specified path. The last folder in the path is the name of the domain. The domain name is wls.

```
//Write out the domain.
set OverwriteDomain "true";
```

Write the domain to "C:\bea/user_projects/domains/wls";

At last we close the template file.

```
//Close domain template to indicate completion
of work.
close template;
```

In Listing 3 we have added a more complex example. Here we create a cluster with an admin server and two managed servers in a WebLogic Platform domain. The database used is an Oracle 9*i* database.

To be able to easily handle different domain configurations, we have replaced the script file's values with tokens. We use an Ant script to build the domain and replace the tokens with actual properties. Extracting the domain configuration file's values into property files makes it simple to handle different domain configurations, and they can easily be put under source control. The property file for the Ant script is referenced in the top of the Ant build file; in this example the name of the property file is "domain.properties". The property file must be modified to fit the chosen environment with properties for domain name, domains folder, portal database attributes such as URL, username, pass, etc., and all server attributes such as server URL, port, SSL-port etc.

The domain is now easily created by running the "build.domain" target in the Ant build file. Because of dependencies in the Ant build file, the following targets are called in sequence:

1. ***"init" target:*** Creates the temporary build folder
2. ***"copy" target:*** Copies the original silent script file into the temporary build folder
3. ***"replace.silentfile" target:*** Replaces all the tokens in the silent script file with the value given in the "domain.properties" file
4. ***"build.domain" target:*** Calls the Configuration Wizard Program (config.cmd) with the newly created silent script file as parameter
5. ***"clean" target:*** Removes all the temporary files and folders

The domain is now created in the path and the name specified in the "domain.properties" file and ready to use. The Ant script and an example property file are shown in Listings 2 and 4.

Conclusion

The domain concept has become a vital part of the BEA WebLogic Platform, defining the applications' runtime environment. In projects with repeated deployments to various targets, it is important to be able to create and recreate the WebLogic domains in an effective and consistent way. It is also important to have these domain configurations under source control. Domain Configuration Wizard's Silent Mode achieves this by providing support for scripting of the domain configuration. This article described the benefits of using scripted domain configurations, and how to use the Silent Mode through an explanation of a scripted domain configuration. It also showed how to extract the Silent Mode script's values into a property file and how to use an Ant script to control the domain creation. We used Silent Mode on several projects and believe this is a good way to effectively and consistently create WebLogic domains.

References

- ***BEA eDocs:*** http://edocs.bea.com

Listing 1 Create a connection pool

```
//Create a JDBCConnectionPool.

create JDBCConnectionPool "demoPool" as mypool;
//required attribute
set mypool.DriverName "com.pointbase.jdbc.jdbcUniversalDriver";
//required attribute
set mypool.URL "jdbc:pointbase:server://localhost:9092/demo";
//required attribute
set mypool.Password "PBPUBLIC";
//optional attribute (but it's recommended you set the db user...)
set mypool.Properties "user=PBPUBLIC";
//target all JDBC connection pools to server "myserver"
assign JDBCConnectionPool "*" to target "myserver";
```

Listing 2 Ant build file

```
<project name="silent config" default="build.domain" basedir=".">
    <description>
        Builds a new domain from the template-config specified in the
domain.properties file
    </description>

  <!-- set properties for this build -->
  <property file="domain.properties"/>
  <property name="propertyfile" value="domain.properties"/>
  <property name="build" location="build"/>
  <property name="silent_config" value="silent_config_build.txt"/>

  <target name="init" depends="clean,clean.domain">
    <!-- Create the temp catalog for scripts -->
    <mkdir dir="${build}"/>
  </target>

  <target name="copy" depends="init"
        description="copy buildfiles to build dir" >
   <copy file="theproject_silent.txt" tofile="${build}/${silent_con-
fig}"/>
  </target>
   <target name="replace.silentfile" depends="copy"
         description="replace buildfiles with properties" >
  <replace
      file="${build}/${silent_config}"
      propertyfile="${propertyfile}">
    <replacefilter
      token="@BEAHOME@"
      property="beahome"/>
    <replacefilter
      token="@DOMAINSPATH@"
      property="domainspath"/>
    <replacefilter
      token="@DOMAINNAME@"
      property="domainname"/>
    <replacefilter
      token="@ORACLESID@"
      property="oraclesid"/>
    <replacefilter
      token="@ORACLEHOST@"
      property="oraclehost"/>
    <replacefilter
      token="@ORACLEPORT@"
      property="oracleport"/>
    <replacefilter
      token="@ORACLEUSER@"
      property="oracleuser"/>
    <replacefilter
      token="@ORACLEPASSWORD@"
      property="oraclepassword"/>
    <replacefilter
      token="@WEBLOGICPASSWORD@"
      property="weblogicpassword"/>
    <replacefilter
      token="@SSLENABLED@"
      property="sslenabled"/>
    <replacefilter
      token="@ADMIN_NAME@"
      property="adminname"/>
    <replacefilter
      token="@ADMIN_ADDR@"
      property="adminadress"/>
    <replacefilter
```

```
      token="@ADMIN_PORT@"
      property="adminport"/>
    <replacefilter
      token="@ADMIN_SSL_PORT@"
      property="adminsslport"/>
    <replacefilter
      token="@MS1_NAME@"
      property="server1name"/>
    <replacefilter
      token="@MS1_ADDR@"
      property="server1adress"/>
    <replacefilter
      token="@MS1_PORT@"
      property="server1port"/>
    <replacefilter
      token="@MS1_SSL_PORT@"
      property="server1sslport"/>
    <replacefilter
      token="@MS2_NAME@"
      property="server2name"/>
    <replacefilter
      token="@MS2_ADDR@"
      property="server2adress"/>
    <replacefilter
      token="@MS2_PORT@"
      property="server2port"/>
    <replacefilter
      token="@MS2_SSL_PORT@"
      property="server2sslport"/>
    <replacefilter
      token="@CLUSTER_NAME@"
      property="clustername"/>
    <replacefilter
      token="@MULTICAST_PORT@"
      property="clustermuliticastport"/>
    <replacefilter
      token="@MULTICAST_ADDR@"
      property="clustermuliticastadress"/>
 </replace>
  </target>

  <target name="build.domain"
depends="replace.silentfile"
        description="builds the domain using
the config.cmd file from BEA in silent mode" >
<exec vmlauncher="false" executable="${beahome}/weblogic81/common/bin/config" failonerror="true">
    <arg value="-mode=silent"/>
    <arg value="-silent_script=${build}/${silent_config}"/>
    <arg value="-logfile=log.log"/>
 </exec>
 <antcall target="clean"/>
</target>

  <target name="clean"
        description="clean up" >
    <!-- Delete the ${build} -->
    <delete dir="${build}"/>
  </target>
  <target name="clean.domain"
        description="clean up domain" >
    <!-- Delete the ${domain} -->
    <delete dir="${domainspath}/${domainname}"/>
  </target>
</project>
```

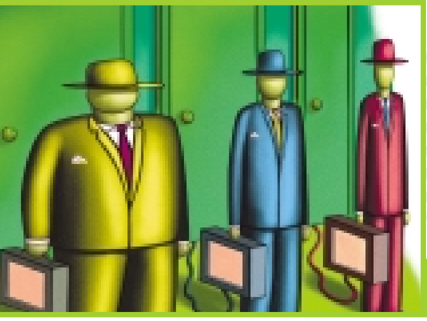
Listing 3 domain.properties

```
#The folder where the domains are placed
#for example C:/bea/user_projects
domainspath=C:/theproject/domains

#The name of the domain you want to create
#for example development
domainname=theproject

#The common WLS STUFF
#Where your bea home is
#for example C:/bea
beahome=C:/bea
#The Weblogic user's password
weblogicpassword=password

#The ORCLE STUFF
#The SID of the Oracle database server used in
the platform
oraclesid=THEPROJECT
#The ipadress of the Oracle database
oraclehost=development.theproject.com
#The port used by the Oracle database
oracleport=1521
#The Oracle user used by the platform
oracleuser=CLUSTER
#The Oracle user's password
oraclepassword=password

#The ADMIN server STUFF
#The name of the adminserver
adminname=cgServer
#The ipadress of the WLP
adminadress=localhost
#The port of the WLP
adminport=7001
##if SSL is enabled (true / false
sslenabled=true
##The SSL listning port
adminsslport=7002

#The 1.managed server STUFF
#The name of the 1.managed server
server1name=managedServer1
#The ipadress of the 1.managed server
server1adress=127.0.0.1
#The port of the WLP
server1port=7003
##The SSL listning port
server1sslport=7004

#The 2.managed server STUFF
#The name of the 2.managed server
server2name=managedServer2
#The ipadress of the 2.managed server
server2adress=127.0.0.1
#The port of the WLP
server2port=7005
##The SSL listning port
server2sslport=7006

#The cluster STUFF
#the name of the cluster
clustername=theCluster
#the multicast prt of the cluster
clustermuliticastport=7777
#the mulitcast adress og the cluster
clustermuliticastadress=localhost
```

Listing 4 Silent Mode script file for BEA WebLogic Platform Cluster

```
//Read in a domain template.

read template from "@BEAHOME@/weblogic81/common/templates/domains/platform.jar";

//Find and configure the Admin Server.
//The default platform domain has configured
an admin server called cgServer.
//We now set the address, the port, and SSL.

find Server @ADMIN_NAME@ as admin;
set admin.Name "@ADMIN_NAME@";
set admin.ListenAddress "@ADMIN_ADDR@";
set admin.ListenPort "@ADMIN_PORT@";
set admin.SSL.Enabled "@SSLENABLED@";
set admin.SSL.ListenPort "@ADMIN_SSL_PORT@";

//Create and configure the 1. managed server
//We create a new server which is the first of
the managed servers.
//The address, ports and SSLenabled are set as
in the configuration of the admin server.

create Server "@MS1_NAME@" as ms1;
set ms1.ListenAddress "@MS1_ADDR@";
set ms1.ListenPort "@MS1_PORT@";
set ms1.SSL.Enabled "@SSLENABLED@";
set ms1.SSL.ListenPort "@MS1_SSL_PORT@";


//Create and configure the 2. managed server
//We create a new server which is the second
of the managed servers.
//The address, ports and SSL are set as in the
configuration of the admin server.

create Server "@MS2_NAME@" as ms2;
set ms2.ListenAddress "@MS2_ADDR@";
set ms2.ListenPort "@MS2_PORT@";
set ms2.SSL.Enabled "@SSLENABLED@";
set ms2.SSL.ListenPort "@MS2_SSL_PORT@";

//We create the cluster which is a cluster of
the two managed servers created above.
//Create a Cluster.

create Cluster "@CLUSTER_NAME@" as cluster;
set cluster.MulticastPort "@MULTICAST_PORT@";
set cluster.MulticastAddress
"@MULTICAST_ADDR@";
set cluster.ClusterAddress
"@MS1_ADDR@:@MS1_PORT@,@MS2_ADDR@:@MS2_PORT@";
assign Server "@MS1_NAME@, @MS2_NAME@" to
Cluster "@CLUSTER_NAME@";


//Change the settings for the
JDBCConnectionPools "cgPool" and "bpmArchPool"
//We now want to change the connection pools
that are configured in the WebLogic Platform
domain.
//There are two connection pools that we have
to change, these are the pools used by
WebLogic Portal
//and WebLogic Integration. In this example we
use an Oracle 9i database.
//It is also possible to create new database
connection pools and new datasources.

//Updating settings for "cgPool"
find JDBCConnectionPool "cgPool" as cgPool;
//required attribute
set cgPool.DriverName
"weblogic.jdbc.oracle.OracleDriver";
//required attribute
set cgPool.DbmsName "@ORACLESID@";
//required attribute
set cgPool.DbmsHost "@ORACLEHOST@";
//required attribute
set cgPool.DbmsPort "@ORACLEPORT@";
//required attribute
set cgPool.UserName "@ORACLEUSER@";
//required attribute
set cgPool.Password "@ORACLEPASSWORD@";

//Updating settings for "bpmArchPool"
find JDBCConnectionPool "bpmArchPool" as
bpmArchPool;
//required attribute
set bpmArchPool.DriverName "weblogic.jdbc.oracle.OracleDriver";
//required attribute
set bpmArchPool.DbmsName "@ORACLESID@";
//required attribute
set bpmArchPool.DbmsHost "@ORACLEHOST@";
//required attribute
set bpmArchPool.DbmsPort "@ORACLEPORT@";
//required attribute
set bpmArchPool.UserName "@ORACLEUSER@";
//required attribute
set bpmArchPool.Password "@ORACLEPASSWORD@";


//target the JDBC connection pools to the
cluster

assign JDBCConnectionPool "*" to target
"@CLUSTER_NAME@";


//Create the admin user and password.

find User "weblogic" as weblogic;
set weblogic.password "@WEBLOGICPASSWORD@";


//Load Database.
//The database is loaded with all tables and
content needed in Weblogic Integration
//and Weblogic Portal

//load database version "9i" using cgPool;


//Close domain template to indicate completion
of work.
//Finally we close the configuration template
file.

close template;

//The domain has now been created.
```

oops.
N26MA
Forget something?



And Now for Something Completely Different

ARE YOU IN THE FRAME?

BY **PETER HOLDITCH**

AUTHOR BIO...

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

CONTACT...

peter.holditch@bea.com

This issue, in an uncharacteristic attempt to fit in with the Zeitgeist, I propose to depart slightly from my well-trodden path to the transaction manager and take a look at frameworks. I expect you can guess which particular framework I am going to take a pass at, too.

For nearly as long as there have been microprocessors, there have been frameworks. J2EE application servers can be said to be frameworks, as could messaging systems or even operating systems. It is also pretty easily arguable that frameworks are the only elements of software that have succeeded in delivering the holy grail of reuse. While instances of truly reusable business logic remain hard to identify, frameworks abound (especially if you accept my definition of frameworks, which includes application servers and such – what could be seen as more reusable than the code comprising a widely adopted commercial product like BEA WebLogic Server?).

Frameworks are entities that exist in layers, each taking care of some generic issues that exist at their architectural layer, and thereby provide a higher level of abstraction to their consumers sitting in the level above. The application server, for instance, takes care of low-level technical issues such as thread and socket management, thereby leaving the programmer of the components that use it able to put more of their focus on the business functionality he or she is trying to implement.

Despite the raised level of abstraction that the application server so effectively provides, pretty much every application server customer that I am aware of does not simply run projects that result in the production of a set of EJBs and servlets created from the ground up to implement some business function. In almost all cases, the app server customers have written a framework themselves, and their application developers write code that exists within their own bespoke framework. Typically, these frameworks perform such functions as reading data from centralized corporate repositories to configure static reference data and business rules. In many cases the frameworks also constrain the degrees of freedom afforded to the application developers – J2EE provides the ability for its users to make many architectural decisions. Is my logic in a servlet, or an EJB? Is the EJB a session or an entity bean? Is the session bean stateful or stateless … the choices roll on and on. These choices provide a powerful toolkit for solving a wide range of server-side, logic-style problems, but in the 80% case the problems application server customers are trying to solve are not wide ranging. Many applications follow very common standard usage patterns. These patterns drive a common set of J2EE usage patterns, or at least they should. By giving the powerful tools to the application developers, you have forced them to be capable of making wise decisions as to how best employ their toolkit.

What is the business value of an entire team of experienced J2EE architects? Why do you need to staff up with people capable of making these hard decisions? Moreover, how will you retain them? If they spend their lives solving problems that are all pretty much identical from a technical perspective they will probably look for another job – ("Brain the size of a planet, and here I am parking cars") or maybe spend the vast majority of their time dreaming up and writing a new framework for you that you may or may not want, or benefit from. Even if you are in the situation where you do have a team of experts and they are disciplined enough not to go off on architectural off-road odysseys of this nature, the very fact that you have many talented individuals doing their own thing in the pursuit will inevitably lead to multiple code styles and variations in practice, which will be a future maintenance cost as the next generation of

programmers looks to maintain the portfolio of business J2EE applications in production and quietly running the business. In production, regularity and conformity are your friends!

So, frameworks are needed (not simply to keep the gurus happy, either!) to provide consistency to developments and to enable "ordinary mortals" to be productive, not just superheroes. However, most application server customers find hidden peril in their bespoke frameworks. When they recruit new team members, these individuals cannot be immediately productive – the first few weeks of their employment will be spent grappling with the bespoke framework they have to work within. Inevitably, documentation will be scanty or nonexistent, so the new hires, however experienced in J2EE they may be, will spend lots of time badgering the guru in the corner about how they should get going. Not only does this waste their time, but the guru in the corner spends half his or her life answering basic questions, not delivering the high-value code they are paid for. This problem gets worse if you look to outsource some of the development. Developers in another organization, or maybe even on another continent, cannot bother the guru in the corner – he or she is not in their corner – with the obvious impact on productivity and so on. The framework has brought us consistency and an ability to have a multi-skill level team, at the expense of the benefit of skill transferability, which was the reason we adopted J2EE in the first place.

So, the net of all this is that while we need a framework to enable productivity, we have identified a set of requirements of any truly useful framework, namely:

- It must be widely adopted and fully documented to maximize developer transferability.
- It must be easy to use to minimize the need of developers to engage in the low-level details of the plumbing of server-side computing. This means it must be toolable.

Enter the Beehive Framework

It is the next generation of the runtime framework powering today's BEA WebLogic Platform applications and comprises three basic elements: PageFlow, controls, and JSR 181 Web services.

All three elements use code annotations to drive what is deployed from a runtime perspective – resulting in declarative setting of runtime behavior confining most of the code-level developer interaction to the direct pursuit of implementing the business requirements. Code annotations on the controller allow the pageflow subsystem to handle the minutiae of struts; annotations at the class and method levels allow things close to business objects to be exposed easily as Web services; and annotations allow developers to make declarations about the reusable chunks of functionality that are delivered in controls. The best news of all: although the annotations are collected together with the code they relate to and could be manipulated with the trusty vi editor, they lend themselves to intelligent manipulations via a Beehive-aware IDE (WebLogic Workshop being the first, but other tool vendors – not to mention the Apache Pollinate project – lining up to ensure it's not the only) that allows developers to easily interact, not only with the Beehive development model, but with custom controls in custom ways – for example, take a look at today's database control – creating one in the IDE prompts you to find a datasource from your running server, and can even go and look at what tables are in the schema. This intelligent, context-sensitive prompting is clearly going to get developers up the learning curve fast, and unlike a wizard-based, code generator approach, the novice user will never be left with hundreds of lines of code and other artifacts that he or she didn't write, doesn't understand, and now has to maintain (anybody remember the 4GL COBOL generator tools?!).

Of course, BEA's existing customers benefit from this approach to development today, as documented in studies by Gartner and Crossvale among others, but Captain Paranoia on their shoulder is whispering the dreaded "proprietary" word, and muttering about vendor lock-in So for these people, Beehive represents the possibility of benefiting from BEA's innovation without needing to commit to a lifetime of BEA infrastructure (although why that would scare anyone is beyond me!)

> "So, frameworks are needed to provide consistency to developments and to enable "ordinary mortals" to be productive, not just superheroes"

For the rest of the world, Beehive is not only providing an open, proven, toolable development framework supported by a major vendor to the world at large, but it is bringing ease of development to server-side Java, all in a way that is sympathetic to the current vogue for services-oriented architecture.

The final bastion of resistance to the tooled framework approach often comes from the J2EE developers themselves. They see their ability to chant the arcane incantations of JNDI lookup on demand as their value – "if tools and abstraction make things so simple, then we will be out of a job!" I believe this is misguided for two reason. First, all the code that does the lookups and so on is invariably cut-and-pasted from samples, or "one I prepared earlier." In my days as a Tuxedo consultant, I lost count of the number of production systems whose comments claimed they put a string into upper case (even the comments were cut and pasted from the "simpapp" sample application). It is the same in the J2EE world – any J2EE developer who thinks his superior ability to cut and paste lines of code will guarantee him job security is, well, misguided must be the polite term. Moreover, frameworks don't mean projects can get by with no J2EE developers – an architect or two will always be required on every development. With a framework, there is now a clear distinction between J2EE architects and business developers. Far from devaluing expertise, the framework actually provides some sort of career path for the developers – highlighting the true value of experience. Finally, if Java (and hence the market value of associated skills) is to survive in the enterprise for the long term, it is imperative that the experts target their expertise wisely, and that the masses can mass produce productively.

So come one, come all… wake up and smell the honey! Beehive has arrived!

References

- *"Application Platform Suites An Architectural Cost Analysis"*: www.bea.com/content/news_events/white_papers/Gartner_APS_Architectural_Cost_Analysis.pdf
- *"A Study in Enterprise Development Productivity"*: http://download.crossvale.com:88/wsad_vs_wlw/J2EEDevelopment-Productivity-Analysis.pdf

web services EDGE conference&expo

Web Services Edge 2005 East

International Web Services Conference & Expo

New for 2005:

- Colocating with LinuxWorld Conference & Expo badges give access to both shows
- Guaranteed Minimum Attendance 3000 Delegates
- New, Hot Sessions & Seminars

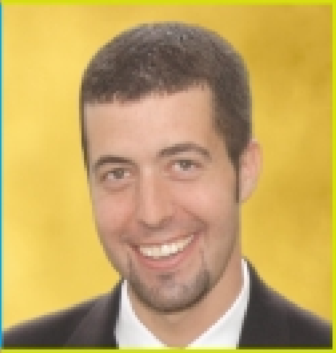
The Largest *i*-Technology Event of the Year!

Tuesday, February 15
11 a.m.

Matt Ackley
Senior Director, eBay Developers Program

Topic:
Web Services for e-Commerce

Keynote Speakers & Featured Guests



Wednesday, February 16
11 a.m.

Ari Bixhorn
Director, Web Services Strategies, Microsoft Corporation

Topic:
Indigo and the Future of Web Services

Anne Thomas Manes
Burton Group

Application Server Shoot-Out Facilitator

Hynes Convention Center
Boston, MA
February 15-17, 2005

For Exhibit and Sponsorship Information:
Jim Hanchrow, 201-802-3066, jimh@sys-con.com

Register Now! Hot Sessions & Seminars

- Ensuring Web Services Interoperability
- Web Services Standards: Going Behind the Mask
- The Role of Policy in Web Services Integration – It's More Than Just Security
- Four Abilities SOA will lack without a Registry
- Driving SOA Governance
- BPEL Best Practices from Real-World Projects
- SOA: From Pattern to Production
- Lessons From the Front Line – Building Interoperable Web Services
- Developing E-Commerce Applications with Web Services
- Developing Enterprise Class Web Services
- Orchestrating FORCEnet Engagement Packs with BPEL for Web Services
- CPI: A Global Integrated Problem Tracking and Resolution System using Java Web Services
- Using Service-Oriented Architecture and Web Services to Issue Business Licenses in the District of Columbia
- Developing Web Services with Eclipse
- The Interoperability Challenge of Web Services Security Standards
- Securing Web Services with WS-Security
- Anatomy of a Web Services Attack
- Using a Mobile Phone as an SSO Authentication Device in SOA Solutions
- XML Content Attacks
- XACML and Agnostic Authorities

Register Today: www.sys-con/edge2005east/registernew.cfm

Come and see what everyone is talking about…

The Application Server Shootout!

Find out which server best fits your performance and ROI metrics

Join us in delivering the latest, freshest, and most proven Web services solutions…

At the *Fifth Annual Web Services Edge 2005 East – International Conference & Expo* as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics.

Exhibit in *The Web Services Pavilion* and see thousands of buyers!

Becoming a Web Services Edge Exhibitor, Sponsor or Partner offers you a unique opportunity to present your organization's message to a targeted audience of Web services professionals. Make your plans now to reach the most qualified software developers, engineers, system architects, analysts, consultants, group leaders, and C-level managers responsible for Web services, initiatives, deployment, development and management at the region's best-known IT business address– The Hynes Convention Center in Boston.

3-Day Conference & Education Program features:

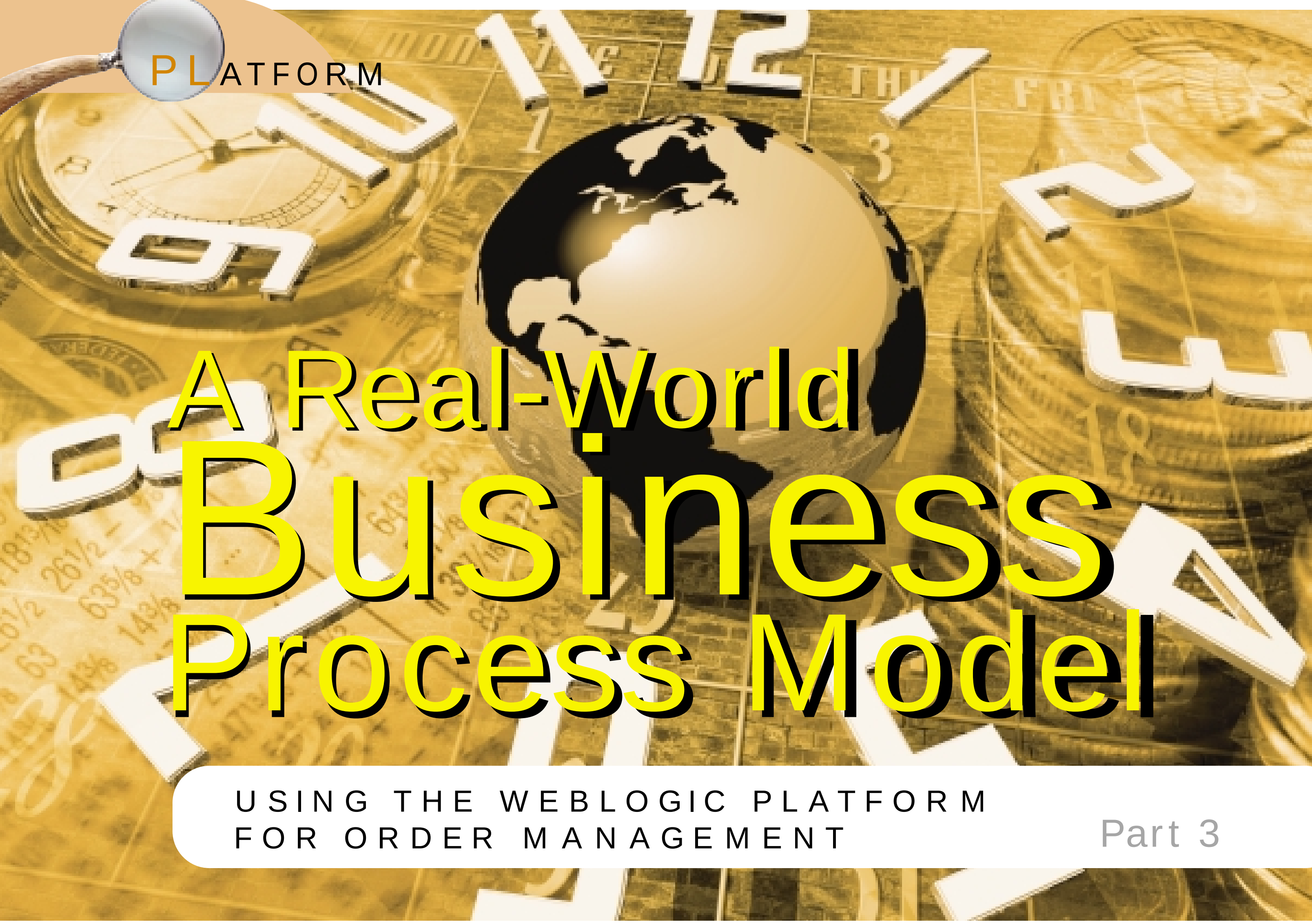
- Daily keynotes from companies building successful and secure Web services
- Daily keynote panels from each technology track
- Over 60 sessions and seminars to choose from
- Daily training programs that will cover Web Service Security, J2EE, and ASP.NET
- FREE full-day tutorials on .NET, J2EE, MX, and WebSphere
- Opening night reception

PRODUCED BY SYS-CON EVENTS

www.sys-con.com/edge

Sponsored by: Information Storage+Security Journal, WLDJ, WebSphere Journal, JDJ, XML Journal, WebServices Journal, MX Developer's Journal, SD Times, eaIndustry Consortium, .NET Developer's Journal, asp.netPRO

All brand and product names mentioned above are trade names, service marks or trademarks of their respective companies.



A Real-World Business Process Model

USING THE WEBLOGIC PLATFORM FOR ORDER MANAGEMENT

Part 3

BY **ANJALI ANAGOL-SUBBARAO**

AUTHOR BIO...

Anjali Anagol-Subbarao works in HP's IT organization as an IT architect. She has 12 years of IT experience, the last 5 in Web services. Her book on J2EE Web services on BEA WebLogic will be published this year.

CONTACT...

anjali.anagol-subbarao@hp.com

In Part 1 (Vol. 3, issue 6) of this series I gave you an overview of business process management (BPM) and covered the specifications in this area. I described the order change example and the steps needed to create the business process in WebLogic Integration. In Part 2 (Vol. 3, issue 7), we created a process application – orderChange. In this application we created a new process called orderChange.jpd. To start the process we added a ClientRequest received. Then we added the Web service validate config.

This month we will add a decision point to handle the result from validate config Web service. The decision point helps in handling both the positive and negative outcomes of the result from the process. Then we will add a database control to check the status of the order to be changed and lastly, we will add another decision node to handle the result from the database Control.

Add a Decision Node

When the validate config process node is executed we have two results – one that the product is valid and the other that it is not valid. To decide what to do with the two results we need to add a decision node in the business process. If the configuration is valid, the process will proceed; otherwise, it will end. To add this node, click Decision in the palette, and drag-and-drop the decision node onto the business process in design view, as shown in Figure 1.

You need to define a condition in this decision node. To do this, double-click the condition node to invoke the decision builder. Variable is selected by default. You should use variable as you design the decision based on the value of an element in an XML document namely status, which is valid against an XML Schema. Select the XML element on which the decision is made. To do so, you need to select the attribute status from outValidateConfig. Select the = operator from the Operator list and enter true in the Right Hand Expression field. Click Add to add the condition you just created (BEA) –

```
data($outValidateConfig/ns:Status)= "true"
```

This completes the design of the first condition on this node. Figure 2 shows that if the validateConfig status is true, you can go to the next step; if the decision is false, the process ends. During runtime the decision point is evaluated to determine the path of the process.

Add a Database Control

If the configuration is valid the process goes to the next step, which is executed through a database control. Database controls are part of the control framework. (We talked about the control framework in the last article.)

You need to add a database control to the process. The database control provides access to a database containing orderStatus for a particular AccountID. This control sends the AccountID to the database table ORDERSTATUS, which sends a response to reflect whether the order is changeable. You can access a relational database through a database control from your application. Using the database control, you can issue SQL commands to the database which accesses the database through the JDBC driver. You must specify a data source that is configured in WebLogic – in this example, it is cgdatasource. The database control automatically performs the translation from database queries to Java objects so that you can easily access query results.

First you will create a new Java class called record.java. The Record class is a Java object that represents an individual record within a database. In particular, it represents an individual record of the ORDERSTATUS table in the database. This is the code to add to record.java:

```
public class Record
{
     public String orderStatus;
}
```

Create a database control file that queries the ORDER STATUS table and then returns a record object containing the results of the query. This database control file is called OrderStatus DB.jcx. JCX stands for Java Control extension. A JCX file extends one of WebLogic Workshop's prebuilt control classes. In this case, it is the com.bea.control.DatabaseControl class, which offers easy access to a database. Most of the built-in controls provided with BEA WebLogic Workshop are customizable – that is, when you add a new built-in control to a project, WebLogic Workshop generates a JCX file that extends the control. In some cases – such as with the database control or the JMS control – you can customize the control by adding or editing methods defined in the JCX file. WebLogic Workshop customizes the EJB control for you based on the EJB that the control will be accessing.

Now add a method named getOrder Status to the database Control file OrderStatus DB.jcx. Then add a SQL query to the method in the property editor, as shown:

```
SELECT ORDER_STATUS FROM ORDERSTATUS WHERE ACCOUNT_ID={accountId}
```

In the Java pane, this change is reflected in the method:

```
public Record getOrderStatus(String accountId)
```

In source view, the code shows that by passing the AccountID, you get the order_status from the database table ORDERSTATUS (see Listing 1).

To pass the AccountID from the XML received from the client to the database control a transformation is used. This transformation maps the AccountId from the received XML to the AccountID sent by the database control.

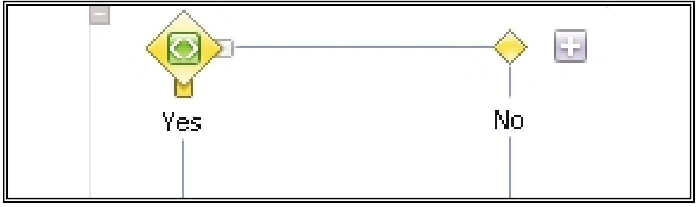
Once the order status is obtained, as shown in Listing 1, it can be sent to the next node in the process.

Add Another Decision Node

The next step is to check whether the orderStatus result allows for the order to be changed. For this, insert another decision point. To make the decision here, use a Java method instead of a variable. To use a Java method, select a method as shown in Figure 3. Select condition in the Java method name. If the condition's return value is boolean, as shown in the code below, then the order is changeable. If the order is changeable, go to the next step; otherwise, stop the process.

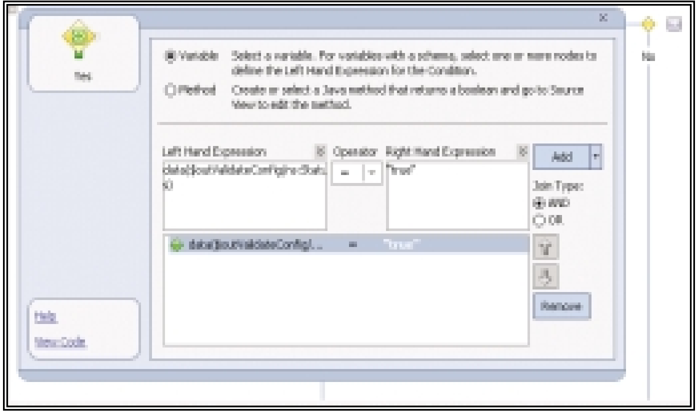
```
public boolean condition()
    {
       record = "OK";
```

FIGURE 1



Click Yes to add Decision mode

FIGURE 2



● Variable Select a variable. For variables with a schema, select one or more nodes to define the Left Hand Expression for the Condition.
○ Method Create or select a Java method that returns a boolean and go to Source View to edit the method.

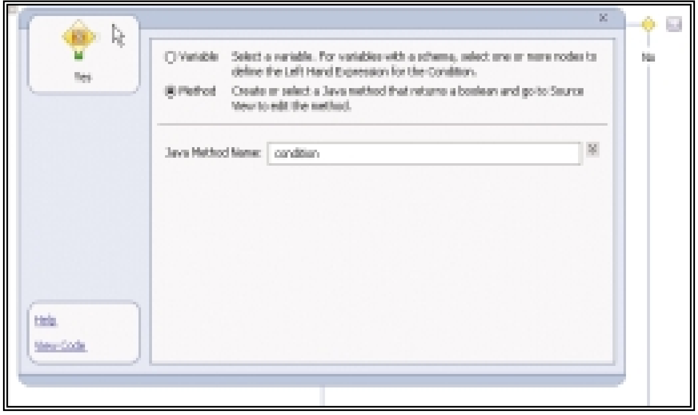
Left Hand Expression: data/$out/ValidateConfig/ns:Stat...()
Operator: =
Right Hand Expression: "true"

Join Type:
● AND
○ OR

data/$out/ValidateConfig/... = "true"

Go to the next step

FIGURE 3



○ Variable Select a variable. For variables with a schema, select one or more nodes to define the Left Hand Expression for the Condition.
● Method Create or select a Java method that returns a boolean and go to Source View to edit the method.

Java Method Name: condition

Select a Java method

```
    return true;
  }
```

Summary

In this article we have seen how the first decision node is added to the process. This node helps to handle the two results obtained from the validate config Web service, namely whether the configuration is valid or not valid. If the configuration is valid, the process calls the next node, which is a database control.

We also looked at the details of creating and adding a database control that checks the database for the order status. This order status will determine if the order is changeable. Then we added a decision to handle the result of the output from the database control.

In my next article we will see if the order status is changeable and how we write this change to a file. This change can then be uploaded to an ERP system like SAP and the order can be changed. In the final article in this series, we will see how this process is executed and monitored.

References

- ***BEA WebLogic Workshop Help:*** http://e-docs.bea.com/workshop/docs81/doc/en/core/index.html

Listing 1 Passing the AccountID to get the order_status

```
import com.bea.control.*;
import java.sql.SQLException;
*
 * @jc:connection data-source-jndi-name="cgDataSource"
 */

public interface OrderStatusDatabase
extends DatabaseControl,
com.bea.control.ControlExtension
{
     static final long serialVersionUID = 1L;

    /**
     * @jc:sql statement="SELECT ORDER_STATUS FROM
 ORDERSTATUS WHERE ACCOUNT_ID={accountId}"
     */
    public Record getOrderStatus(String accountId);
}
```

–continued from page 24

Strategies for WebLogic Domain Configuration

5. Take the defaults by clicking Next on the subsequent dialogs
6. Specify Configuration Name as something meaningful (e.g., SEMDomain-Complete) and click Create
7. Click Done

However, starting a WebLogic Server instance of this domain fails! The encrypted security credentials conflict. Make the following changes manually to the config.xml file.

Changes to config.xml

1 Change SecurityConfiguration element's Credential attribute's value to ""
2. Change EmbeddedLDAP element's Credential attribute's value to ""
3. Change cgPool Password attribute's value to "weblogic"
4. Change semJDBCConnectionPool Password attribute's value to "sem"

> "Another important templating strategy is to create an extension template"

Follow the steps described in Verifying the Domain Configuration to verify that the domain is working properly.

Another important templating strategy is to create an extension template; for example, an extension template that extends DefaultWebApp template and configures all the necessary resources. Creating the extension template, creating a domain, and extending the domain with the extension template are left as an exercise for you.

Conclusion

Domain configuration by hand-editing the config.xml file is error-prone and dangerous. On the other extreme, domain configuration by writing Java code is tedious. Common alternatives include domain configuration by using the WebLogic Console or Domain Configuration Wizard. However, both alternatives are manual and likely to be repetitive and time-consuming. In search of automated solutions, I evaluated templating. Please stay tuned for my next article, which will evaluate scripting strategies. I hope this article helps you make a knowledgeable and informed decision about choosing tools and techniques for automating domain configuration.

References

- To discuss the article and ask questions start here: www.bartssandbox.com. Free membership is required.

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...

>Linux >Java >Web Services >.NET >XML >Wireless >Storage >Security

The Premier Resource for Today's Corporate & IT Decision Makers

VOL: 1 ISSUE: 1 SPRING 2004

IT solutions >GUIDE

WWW.SYS-CON.COM/IT

TECHNOLOGIES YOU NEED NOW!

How to Manage Your Ideas Using Today's *i*-Technologies

Top IT Solution Providers!

- > Candle
- > ClearNova
- > Computer Associates
- > Creative Science
- > DralaSoft
- > GraphOn
- > InterSystems
- > iTKO
- > Merant
- > O'Reilly
- > Parasoft
- > QuadBase
- > SAP
- > Wily Technology
- > Zero G

Offering leading software, services, and hardware vendors an opportunity to ***speak to over 100,000 purchasing decision makers*** *about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management*

Don't Miss Your Opportunity to Be a Part of the Next Issue!

Get Listed as a Top 20* Solutions Provider

For Advertising Details Call 201 802-3021 Today!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE.

SYS-CON MEDIA

The World's Leading i-Technology Publisher

SOAPtest

A VALUABLE WEB SERVICE TESTING RESOURCE

Reviewed by JASON SNYDER

Testing Web services creates an entirely new set of problems for development and testing teams. JUnits can be created to test parts of the Web service, but do not provide the overall functionality needed to ensure adequate validation, and make updating data values difficult. There are also many Web service components to be tested that do not find their way into other, more generic testing tools. Web service testing can become a nightmare very quickly, which is particularly dangerous because this is one area that can involve external partners fairly early in the development process.

SOAPtest provides an entire suite of Web service–focused testing tools that enable the development team and/or testing teams to use one consistent tool for proving out Web services from Web Service Definition Language (WSDL) validation to performance testing without having to build and update testing clients themselves.

One key feature of SOAPtest is the wide array of data sources the tool supports. Creating test data can be time consuming and error prone, but the SOAPtest tool definitely helped lessen the overall effort. SOAPtest supports test data from any ODBC/JDBC connectable database, a .csv file, an internal table, an Excel spreadsheet, or an aggregate of any of those listed. Adding the data source is straightforward – you pick the type and location. I used an Excel spreadsheet provided in the tutorial.

I began by running a series of WSDL validations. WSDL controls access to the Web service by describing the service and indicating location. The screen interface is fairly intuitive; I was quickly able to validate a series of WSDL files.

You can also check that any URL links in the WSDL are still valid. Most interesting is the ability to perform a Web Services-Interoperability Organization validation. An impressive report is generated that can be analyzed by the development and test teams and distributed to your partners as appropriate.

WSDL comparison and regression testing is also supported, allowing the development teams to easily catch and confirm WSDL changes prior to additional testing efforts.

Once the WSDL has been confirmed and validated, the next key step is to ensure that the Web service operations are functional at the unit test level. This is performed by creating a series of unit tests, logically grouped in a series of test suites. SOAPtest supports positive testing (testing expected scenarios), negative testing (error conditions), and regression testing (performing change validation). The test data can be fully or partially from the variety of data sources, or can be generated on-the-fly by the tool. The data-generation aspect is fairly feature-rich and offers the ability to generate the data according to a series of rules defined by the user. One negative was that I attempted to save just the individual test itself and was unable to figure out how to do so.

Once the developer has completed the unit testing, SOAPtest offers the ability to create actual scenario tests. Scenario tests represent business scenarios recreated during the testing process. The scenario tests can be created from scratch or combine a series of already created unit tests merged with additional tests defined by the testing team.

SOAPtest provides a spectrum of data validation abilities. The data can be validated by element, or by only those elements that you wish to compare.

Another key aspect of many Web services is their asynchronous nature. Testing tools that expect the traditional synchronous responses of other applications have difficulty with this typical aspect of Web services. SOAPtest offers the ability to test asynchronous calls that use the communication protocols Parlay, SCP, and WS-Addressing. This is key because the nature of many of these services involves not getting an acknowledgement or response directly. SOAPtest includes a Tomcat server that allows it to test the "Call Back" responses expected when the Result Call back pattern of asynchrony is employed. One suggested update is to have the Tomcat server started automatically when the Call Back test is employed. The results of a series of asynchro-

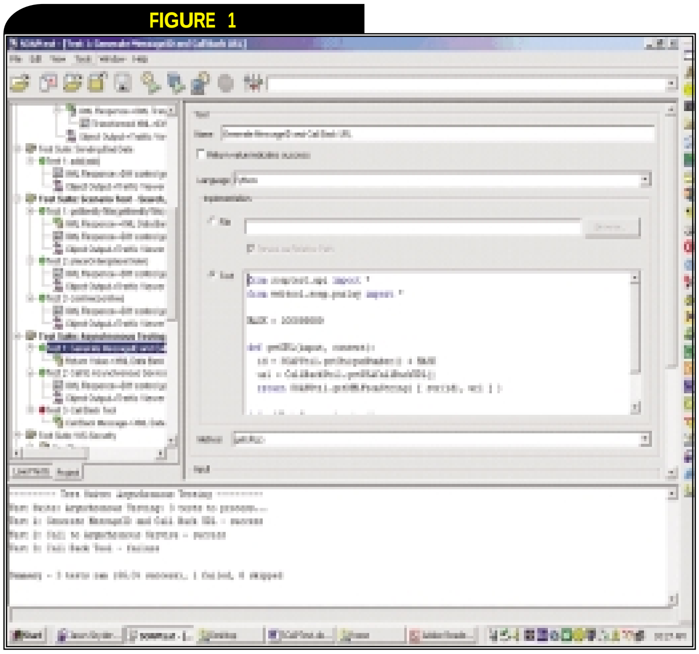
Corporate info:
Parasoft
101 E. Huntington Drive
Monrovia, CA 91016
626 305-9090
www.parasoft.com
sales phone:
888 305-0041
sales e-mail:
info@parasoft.com

nous tests are shown in Figure 1

Another key detail when developing Web services involves the importance of security. The very nature of exposing Web services requires security forethought; hence, testing the security features is integral to any Web service validation effort. SOAPtest offers an encryption tool that supports full or partial encryption of messages. Encryption standards offers include Triple DES, AES 128, AES 192, or AES 256. The numbers stand for the bit key sizes. When using encryption testing, be sure to have the JCE (Java Cryptography Extension) Unlimited Strength Jurisdiction Policy File installed as the test will not work without it.

The other key security features offered by SOAPtest include the XML Signer Tool, the XML Verifier Tool, and support for user name and SAML tokens. The XML Signer Tool provides digital signing ability. If your Web service requires certain digital signatures, the tool can be used to validate that functionality. Likewise, the XML Verifier tool allows the user to encrypt/decrypt/verify the digitally signed message by using a public/private key store file. Key store file formats supported include JKS, PKCS12, BKS, and UBER.

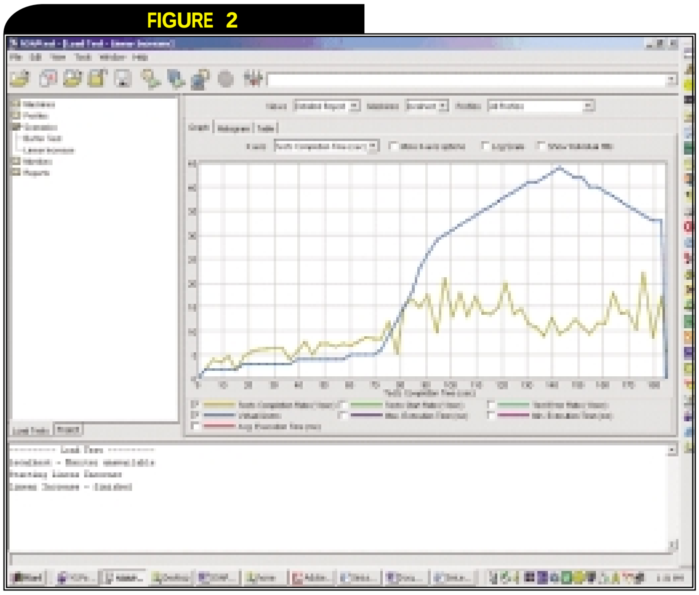
FIGURE 1



```
---------- Test Suite: Asynchronous Testing ----------
Test Suite: Asynchronous Testing: 3 tests to process...
Test 1: Generate MessageID and Call Back URL - success
Test 2: Call to Asynchronous Service - success
Test 3: Call Back Tool - failure

Summary - 3 tests ran (66.7% success), 1 failed, 0 skipped
```

Asynchronous test results

FIGURE 2



```
---------- Load Test ----------
localhost - Monitor unavailable
Starting Linear Increase
Linear Increase - finished
```

Linear increase Load Test

Security Assertion Markup Language (SAML) provides a standard for exchanging security information. SAML provides a means for Web services to exchange authentication and authorization assertions. This gives the team an opportunity to provide identity management services across an organization. SOAPtest supports SAML verification before a failure occurs.

If your Web service testing does not fall within the features identified previously, SOAPtest allows ultimate flexibility by offering custom test scripting abilities. The scripts can be written in Java, JavaScript, or Jython, and the tool supports the importing of files. The code can perform whatever additional testing logic is needed for the testing to complete.

Another key aspect when testing Web services is confirming performance requirements and identifying performance boundaries. SOAPtest is fairly unique among Web service testing tools in that it offers vast capabilities in the Load Testing arena.

Accessing the Load Test toolkit for SOAPtest involves selecting the Load Test tab from the left-hand navigation panel. Once there, existing scenarios can be used from the earlier functional tests, or new scenarios can be created. The tool provides the ability to run these tests across the network by enslaving associated machines. This allows the load levels to be increased dramatically as you can avoid local machine limitations.

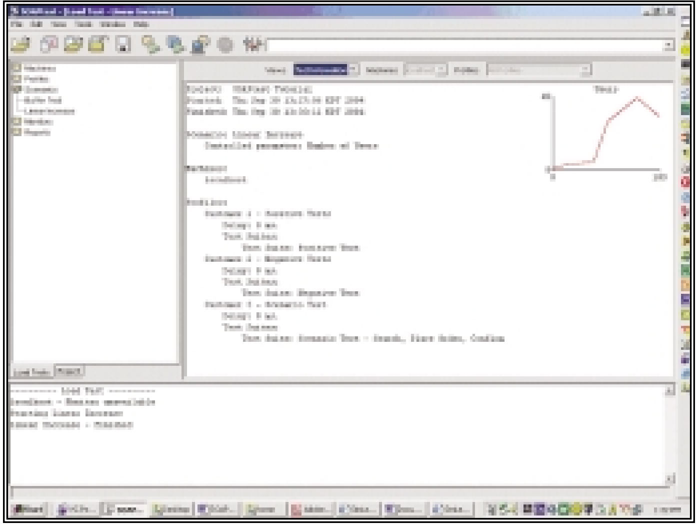
Functional tests are incorporated into the Load Testing effort by creating user profiles. Each profile is made up of one or multiple functional tests and can be included or removed from every Load Test effort. I used profiles based on the positive and negative test scenarios created earlier.

Each profile can also be delayed so as to better monitor the impact of each functional test on load. The profile ratio is adjusted using the Profiles tab associated with the scenario. It displays the information graphically and allows adjustments of users and hits per second throughout the scheduled test cycle. Simply right-click anywhere on the graph to create a point, and then move that point to the appropriate location. Hits per second and number of users are also dynamically adjustable.

I created an initial slow increase in users, followed by a dramatic escalation a minute and a half into testing, and then a slight decrease. I also altered my ratio of user profiles to decrease the amount of positive tests over time as well. The duration of the test is also entirely up to the user, and it provides values for days. I opted for minutes – three to be exact. I ran the test and the results are shown in Figure 2.

I could also see a text-based graph summary (see Figure 3).

FIGURE 3



Text-based graph summary

Many operating system statistics can be added and monitored by the testing community. SOAPtest supports adding SMNP, Windows Perfmon, and JMS monitoring packages. I used the default windows.

SOAPtest allows reports to be generated from all of the test results reviewed. The reports can be generated in HTML, XML, or .csv format.

Conclusion

In conclusion, SOAPtest provides a valuable Web service testing resource across the entire development life cycle. Its interface is easy enough for both developers and testers to master. It encourages the sharing of tests across teams, and provides test suites capable of supporting the smallest unit testing efforts to full Web service load testing. It supports a wide variety of data sources, sophisticated test scenario creation, and automated testing.

Asynchronous Web service validation and complex security verification are additional features that really differentiate SOAPtest from other testing products.

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...

jsnyder1@csc.com

FREE RESOURCE CD INCLUDED!
information
STORAGE+
SECURITY
journal
www.ISSJournal.com
PREMIER 2004 VOL.1 ISSUE 1
In this issue:
■ IP Storage
■ SAN NAS
■ Disaster Recovery
■ Enterprise Security
■ Infosecurity
Microsoft
RSA
Security
SecureID for Windows
SAN
NAS
Emerging technology trends
and market maneuvers

A LIMITED TIME SAVINGS

SUBSCRIBE TODAY TO MULTI

Go To www.sys-con.com

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

SAVE UP TO $340 FOR MULTIPLE SUBSCRIPTIONS
3-PACK • 6-PACK • 9-PACK– Limited Time Offer

PLEASE CHOOSE ANY OF THE FOLLOWING VALUE PACKS

3-Pack Pick any 3 of our magazines for only $99.00 save up to $279 for a 1 year subscription and receive a FREE CD

OR

6-Pack Pick any 6 of our magazines for only $199.00 save up to $398 for a 1 year subscription and receive a 2 FREE CDs

OR

9-Pack Pick any 9 of our magazines for only $399.00 save up to $279 for a 1 year subscription and receive a 3 FREE CDs

SYS-CON Media, Inc.

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

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

Pick the CDs to go with your Multi-Pack order

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

□ Web Services Resource CD
□ Java Resource CD
□ WLDJ Resource CD
□ ColdFusion Resource CD
□ XML Resource CD
□ Linux Resource CD
□ .NET Resource CD

Your order will be processed the same day!

WebServices Journal
Web Services
Resource CD
Secrets of the Web Services Masters
$198 VALUE
Inside:
Every Issue of WSJ Ever Published on One CD!

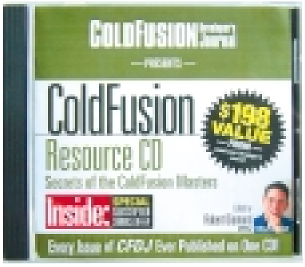
More than 1,400 Web services and Java articles on one CD! Edited by well-known editors-in-chief Sean Rhody and Alan Williamson, these articles are organized into more than 50 chapters on UDDI, distributed computing, e-business, applets, SOAP, and many other topics. Plus, editorials, interviews, tips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

JDJ
JDJ
Resource CD
Secrets of the Java Masters
$198 VALUE
Inside:
Every Issue of JDJ Ever Published on One CD!

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

wldj
WLDJ
Resource CD
Secrets of the WLDJ Masters
$198 VALUE
Inside:
Every Issue of WLDJ Ever Published on One CD!

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


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

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

XML JOURNAL
XML
Resource CD
Secrets of the XML Masters
$198 VALUE
Inside:
Every Issue of XML-J Ever Published on One CD!

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

LinuxWorld
Linux
Resource CD
Secrets of the Linux Masters
Inside:
The Complete LinuxWorld Archives on One CD!

The Secrets of the Linux Masters: Linux Resource CD has every *LWM* article on one CD. The CD is edited by *LWM* editor-in-chief Kevin Bedell and is organized into 38 chapters, containing more than 2600 exclusive ***LWM*** articles. The CD is in an easy-to navigate HTML format, and includes full source code.
LIST PRICE $198

.NET JOURNAL
.NET
Resource CD
Secrets of the .NET Masters
$198 VALUE
Inside:
Every Issue of .NETDJ Ever Published on One CD!

The most complete library of exclusive *.NETDJ* articles is here, on one CD edited by *.NETDJ* editor-in-chief Derek Ferguson. More than 250 articles cover interfaces, open source, migration, object-oriented programming, ASP.NET, VS.NET, certification exams, mobile .NET, interviews, product and book reviews, and more.
LIST PRICE $198

Subscribe Online Today

OFFER FROM SYS-CON MEDIA

PLE MAGAZINES ONLINE

AND SAVE UP TO $340 AND RECEIVE UP TO 3 FREE CDs!*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to $210⁰⁰

Pay only $99 for a 1 year subscription plus a FREE CD

- 2 Year – $179.00
- Canada/Mexico – $189.00
- International – $199.00

6-Pack

Pick any 6 of our magazines and save up to $340⁰⁰

Pay only $199 for a 1 year subscription plus 2 FREE CDs

- 2 Year – $379.00
- Canada/Mexico – $399.00
- International – $449.00

9-Pack

Pick 9 of our magazines and save up to $270⁰⁰

Pay only $399 for a 1 year subscription plus 3 FREE CDs

- 2 Year – $699.00
- Canada/Mexico – $749.00
- International – $849.00

CALL TODAY! 888-303-5282

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.

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.

□ **LinuxWorld Magazine**

U.S. - Two Years (24) Cover: $143	You Pay: $79.99 /	Save: $63 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int’l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int’l - One Year (12) $108	You Pay: $99.99 /	Save: $8

□ **JDJ**

U.S. - Two Years (24) Cover: $144	You Pay: $99 /	Save: $45 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $69.99 /	Save: $12
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $89.99 /	Save: $40
Int’l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int’l - One Year (12) $108	You Pay: $99.99 /	Save: $8

□ **Web Services Journal**

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

□ **.NET Developer’s Journal**

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

□ **Information Storage + Security Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $93 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $39
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $88 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $34
Int’l - Two Years (24) $216	You Pay: $89.99 /	Save: $126 + FREE $198 CD
Int’l - One Year (12) $108	You Pay: $59.99 /	Save: $48

□ **Wireless Business & Technology**

U.S. - Two Years (12) Cover: $120	You Pay: $49.00 /	Save: $71 + FREE $198 CD
U.S. - One Year (6) Cover: $60	You Pay: $29.99 /	Save: $30
Can/Mex - Two Years (12) $120	You Pay: $69.99 /	Save: $51 + FREE $198 CD
Can/Mex - One Year (6) $60	You Pay: $49.99 /	Save: $10
Int’l - Two Years (12) $120	You Pay: $99.99 /	Save: $20 + FREE $198 CD
Int’l - One Year (6) $72	You Pay: $69.99 /	Save: $2

□ **MX Developer’s Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $93 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $88 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $34
Int’l - Two Years (24) $216	You Pay: $89.99 /	Save: $126 + FREE $198 CD
Int’l - One Year (12) $108	You Pay: $59.99 /	Save: $48

□ **ColdFusion Developer’s Journal**

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

□ **WebSphere Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129.00 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int’l - Two Years (24) $264	You Pay: $189.00 /	Save: $75
Int’l - One Year (12) $132	You Pay: $129.99 /	Save: $2

□ **PowerBuilder Developer’s Journal**

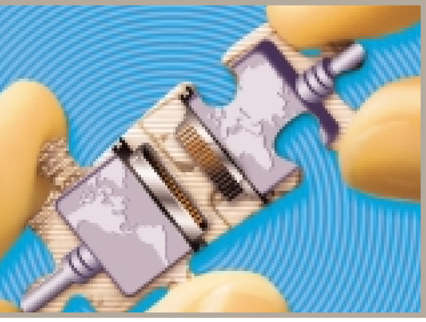
U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int’l - Two Years (24) $360	You Pay: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

□ **WLDJ**

U.S. - Four Years (24) Cover: $240	You Pay: $99.99 /	Save: $140 + FREE $198 CD
U.S. - Two Year (12) Cover: $120	You Pay: $49.99 /	Save: $70
Can/Mex - Four Years (24) $240	You Pay: $99.99 /	Save: $140 + FREE $198 CD
Can/Mex - Two Year (12) $120	You Pay: $69.99 /	Save: $50
Int’l - Four Years (24) $240	You Pay: $120 /	Save: $120 + FREE $198 CD
Int’l - Two Year (12) $120	You Pay: $79.99 /	Save: $40

www.sys-con.com/suboffer.cfm

SYS-CON MEDIA



Enterprise Information Bus

SERVICE ON-DEMAND PORTALS PART 2

BY **RAJUL RANA & SAI KUMAR**

AUTHOR BIOs...

Rajul Rana is a senior architect with Mphasis Corporation (www.mphasis.com), a global IT services organization. Rajul leads the Portal Practice at Mphasis and has architected several large, successful e-business and portal projects for Fortune 500 companies, built on a variety of products. His areas of interest span J2EE technologies, portal, content management, and SOA.

Dr. Sai Kumar is a senior architect with Mphasis Corporation, where he heads the Web services and SOA practice. He has provided strategic consulting in SOA to various companies and has architected several enterprise solutions for financial, health care, and retail industries

CONTACT...

rajul.rana@mphasis.com
saikumar@mphasis.com.

In part one (Vol.3, issue 7) of this two-part article, we discussed the "on demand" information delivery architecture based on portal technologies and WSRP. In this part, the concept of on demand will be extended beyond the information delivery layer to the information aggregation and integration layer. We'll introduce the Enterprise Information Bus (EIB) and discuss its role in building service-on-demand portals.

Emergence of Information Aggregation

Portals are never implemented as standalone infrastructures; they have to integrate with the larger enterprise ecosystem. Portals are the face of this ecosystem. They need to integrate with various information applications across the enterprise. Information applications include, but are not restricted to, various back-end information systems, and modern applications that are specialized applications for a specific product or process. Portals have to integrate with them to provide summarized and detailed views into these applications. Though modern applications are built on standards-based technology, back-end applications have evolved over time and are built on heterogeneous platforms. Gathering information from these disparate applications poses a major integration challenge. Without a loosely coupled, low-cost, integration infrastructure, a portal project turns into an enterprise integration project rather than an information delivery project. Portal products, as a technology, are not designed to take on this enterprise integration challenge completely.

The key to a service-on-demand enterprise is to build an infrastructure and architecture that demonstrates loose coupling and flexibility. The service-on-demand enterprise architecture should also provide a layer of abstraction between the various heterogeneous system interfaces provided by the back-end legacy applications, the business applications that consume the information, and the different business entities. A myriad of business entities and information objects from assorted information sources are required by most of the business and middleware applications.

Key to a customer-centric or business process–centric application development capability is the ability to access business entities and information from various back end-systems. The challenges an enterprise faces in accessing business entities is knowing the:

- Authoritative sources of information
- Location of these authoritative sources of information
- Access mechanisms to these sources
- Ability to collate information from them in a consistent manner

For example, in a large-scale manufacturing industry, various applications, such as a contact center application, sales force application, and customer relationship management system, depend heavily on the right business entities and information for making decisions and supporting business transactions. This information is conventionally gathered by accessing various APIs offered by legacy applications, databases, and heterogeneous systems, and applying a variety of

transformations to finally extract a meaningful information entity.

Currently, in most enterprise architectures the burden of identifying the right source of information, the ability to access it and associated mechanisms, and understanding the semantics of information entities lies heavily on the consuming applications. So in a sense these applications are logically tightly coupled with the back-end information sources and require a great deal of knowledge to access and use them.

Emergence of various information aggregation and collation technologies have attempted to solve this problem, and have gone to the extent of providing Enterprise Service Bus (ESB) where all the back-end system integration is abstracted and information is available as a set of services for applications to consume. This still leaves the problem of identifying the right source(s) for business components, its schema and the ability to access the business component as entity versus service interface is still unresolved.

Below we briefly discuss the integration technology evolution followed by an introduction to the Enterprise Information Bus.

Portal Integration Evolution

Back-end application integration into the portal has evolved over time. Listed below are some popular integration styles used in the past (see Figure 1):

- Point-to-Point integration
- Message bus
- Enterprise Application Integration (EAI)
- Enterprise Service Bus (ESB)

Note: This article assumes that you are familiar with the basic concepts of the above integration styles and does not attempt to elaborate on them.

Point-to-Point Integration

In this integration style, the portal directly integrates with one or more back-end applications. The portal leverages the application server and a variety of integration standards supported by the platform. For example, a portal based on a J2EE platform could leverage integration technologies such as JCA, JMS, Web services, and HTTP over XML among others. With this integration style, the integration tier is mixed with the presentation tier (information delivery tier) and cannot be leveraged for enterprise use.

Message Bus Integration

Message Oriented Middleware (MOM) involves passing messages (in any format – XML is now becoming popular) between two applications, usually in an asynchronous fashion. MOM can be implemented as point-to-point, or as a hub-and-spoke solution. This integration style provides distributed integration technology with separation of the information delivery and integration tier (e.g., MQSeries, MSMQ). MOM allows reuse of message formats and centralized message routing and processing, but the presentation tier is largely duplicated within consuming applications. Each client application is tied to a queue and is aware of the services that are offered.

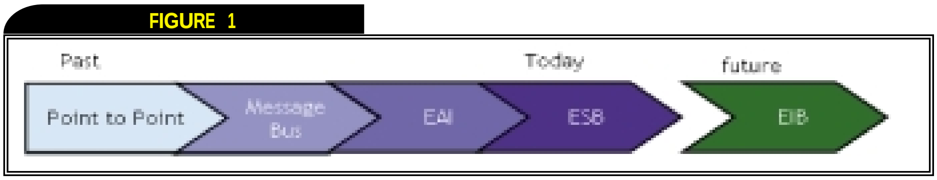
EAI Integration

Integration brokers provide another class of integration technology known as EAI. In addition to MOM, they provide adapter (though proprietary) based integration to various back-end systems. They also allow business process definition and execution separate from the integration logic. EAI implementations follow a hub-and-spoke model, which requires multiple portal applications to go through a single EAI hub. This model does not work well with cross-organizational boundaries.

ESB Integration

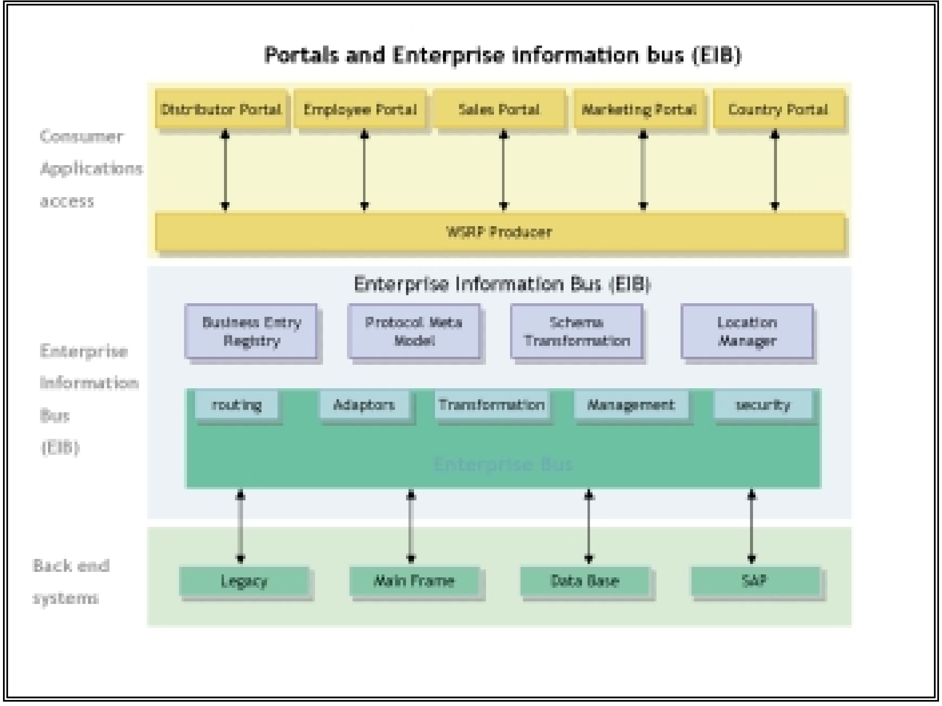
Enterprise service buses (ESBs) are the latest generation of middleware, and combine features from several types of middleware discussed above. Many ESBs currently support Web services protocols such as Simple Object Access Protocol (SOAP) and use Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI). In addition, some multiprotocol ESBs allow applications to communicate using a variety of protocol standards such as Web services, JCA, JMS, HTTP/XML, TCP, and messaging (MQSeries, MSMQ, etc.). It can also support custom APIs (Java/J2EE, COM, etc.). It has the advantage of being highly distributed and can cross organizational boundaries. With this integration style, the portal needs to know only one interface to communicate with the ESB.

FIGURE 1



Portal integration evolution

FIGURE 2



WSRP and EDI = service portals on demand portals

Enterprise Information Bus (EIB)

The Enterprise Information Bus (EIB) is an architectural topology that integrates all the back-end information sources and systems seamlessly and provides business and information entities to applications, consuming them through various loosely coupled protocols.

EIB provides the following capabilities above the conventional managed data bus:

- ***Business registry:*** This component makes the information registry available for the consuming application to look into. Typically in large enterprises, identifying which business components exist within the organization and its semantics is essential. This becomes far more important in a geographically distributed enterprise. For example, in a financial organization it is important to know where and how entities such as customer, account, credit, and check information can be available. Similarly, in a manufacturing enterprise, order, account, part index, and customer information are key.
- ***Location management:*** The location of information constantly changes as various back-end systems change. For example, in a manufacturing organization the shipment management system can be the source of authority for order information and can potentially move toward the order booking system to be the source of authority due to changes in the supply chain model, or systems upgrade model. Many of the problems in an enterprise stem from the fact that consuming business applications tend to construct business entities from wrong sources of information. For consistent and reliable access, entity definitions and locations should be centralized and handled at the EIB level rather than each application having the knowledge of the information source.
- ***Meta model:*** Meta models orchestrate and define meta entities, which need to be constructed from other business entities. For example, construction of relationship entity from account, customer, and order entities is very common in customer relationship management applications.

These features and capabilities, apart from the management, routing, security, and adapter features of a regular service/-message bus, are important to make information access more business entity–centric rather than interface-centric. Entitlement can be associated with each business entity based on the roles of consuming applications.

WSRP and EIB – A Great Combination

In the first article, we saw the capabilities of WSRP and its relevance to federated portals. WSRP provides the information and rendering as an extension to a conventional SOAP stack. It facilitates federation and loose coupling capability at the information delivery layer through portals.

To extend the power of federation, on demand services, and loose coupling beyond the information delivery layer to the integration layer, EIB needs to be harnessed along with the WSRP infrastructure.

EIB and WSRP are a great combination in terms of providing information aggregation, federated information access, loose coupling, and business entity–level access. A WSRP-enabled EIB can provide an on-demand portal architecture.

The EIB can plug in applications on the bus using WSRP in addition to a variety of protocols available with the enterprise bus. On the other side, a WSRP wrapper layer that leverages the EIB can allow organizations to construct presentation information once to be used everywhere.

Figure 2 represents an enterprise-scale design pattern for implementing a completely cohesive, loosely coupled, and business entity–driven service on-demand information gathering and distribution architecture.

Conclusion

These are the key concepts discussed in this series:

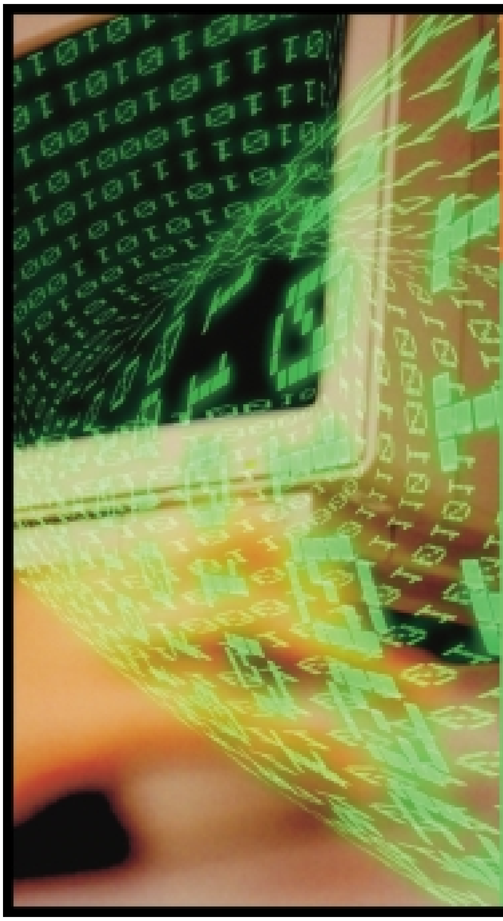
- WSRP is a promising, emerging technology that will benefit large organizations to draw their portal strategy.
- Federated portal architecture allows organizations to reuse existing assets, common enterprise view, and autonomy at the same time.
- While portal servers are the face of the enterprise ecosystem, the EIB serves as the glue that joins various other components (information applications).
- EIBs are the integration infrastructure to aggregate, define, and deliver information across the organization in a location-independent way

Security Best Practices

continued from page 19

- ***boot.properties:*** Encrypted username and password for booting administration server.
- ***running-managed-servers.xml:*** List of dependent managed servers currently running. This file is used for discovering managed servers if administration server gets restarted while the managed servers are running.
- ***domain/configArchive/:*** Contains the copies of the Domain Configuration Repository files. When updated using administration tools, the administration server copies the old config.xml to this directory.
- ***domain\adminserver\ldap\ldap-files:*** Embedded LDAP data files currently being used by the domain's administration server.
- ****.ldift files:*** Can be used to initialize the WebLogic Domain Embedded LDAP server back to when the Domain was created.
- ***domain/adminserver/ldap/backup/EmbeddedLDAPBackup.zip:*** Backup of the WebLogic Domain embedded LDAP server. Embedded LDAP is used to store users, groups, roles, policies by the default security realm, myrealm's security providers.
- ***SerializedSystemIni.dat:*** Contains encrypted security data that must be present to boot the server.
- ***Security Certificates:*** Security certificates and keys being used by the servers of a domain should also be backed up. The location of these files is user configurable.
- ***Batch/Shell Scripts:*** setEnv.cmd/sh, startWebLogic.cmd/sh, startManagedWebLogic.cmd/sh.

The WebLogic administrator can check for BEA security advisories to download security-related patches and register to receive notifications of newly available security advisories. Refer to the BEA Advisories & Notifications page on the dev2dev Web site at http://dev2dev.bea.com/advisories.



WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Acsera	www.acsera.com/demo	408-245-1000	13
Compuware	www.compuware.com	800-521-9353	29
EV1 Servers	www.ev1servers.net	800-504-SURF	11
H&W Computer	www.hwcs.com/wldj08.asp	800-338-6692	25
HP	http://devresource.hp.com/d2d.htm	800-752-0900	7
Information Storage & Security Journal	www.issjournal.com	888-303-5282	43
Intersperse	www.intersperse.com	626-535-2000	52
iTKO	www.itko.com	877-BUY-ITKO	17
IT Solutions Guide	www.sys-con.com	201-802-3021	39
Motive	www.motive.com/within1	512-531-2527	51
NetIQ	www.netiq.com/solutions/web	408-856-3000	31
Pantero	www.pantero.com	781-890-2890	3
SandCherry	www.sandcherry.com	866-383-4500	23
SYS-CON Publications	www.sys-con.com/suboffer.cfm	888-303-5282	44-45
SYS-CON Reprints	www.sys-con.com	201-802-3026	27
Tangosol	www.tangosol.com	617-623-5782	2
Web Services Edge 2005 East	www.sys-con.com/edge	201-802-3004	34-35
Wily Technology	www.wilytech.com	888-GET-WILY	5
WLDJ	www.sys-con.com/wldj	888-303-5282	

This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions

News & Developments

BEA Ships SOA–Based Mobile Solution

(San Jose, CA) – BEA Systems, a world leader in enterprise infrastructure software, has released the MobileAware Interaction Server, BEA WebLogic Edition, an integrated, mobile platform designed to help customers mobilize online data and business processes. Targeting enterprises in both the telecommunications industry and other business markets, this innovation can extend the service-oriented architecture (SOA) technology within BEA WebLogic Portal 8.1 to provide a simpler, faster, and cost-effective approach to accessing and delivering a broad set of data via diverse mobile devices.

MobileAware Interaction Server, BEA WebLogic Edition, is a new technology platform designed to extend WebLogic Portal to the mobile domain, helping to provide multi-channel access to Web and online assets. Because it is designed to allow companies to leverage existing processes, applications, and software components across multiple channels, the MobileAware Interaction Server, BEA WebLogic Edition can be highly cost-effective. www.bea.com

Intersperse Completes BEA Validation Program

(Pasadena, CA) – Intersperse Manager 3.0, from Intersperse, Inc., a provider of management solutions for service-oriented enterprise applications, has completed the BEA Validation Program and is now certified on WebLogic Platform 8.1.

Intersperse provides Global 2000 companies and large public sector organizations with products designed for whole-system visibility and real-time control over an enterprise's distributed or service-oriented application environments.

BEA Validation means that Intersperse can offer unique and powerful solutions for the delivery of mission critical applications in a J2EE environment. www.intersperse.com

New York Board of Trade Implements SOA

(San Jose, CA) – The New York Board of Trade (NYBOT) has implemented a BEA-powered SOA, replacing outdated proprietary mainframe technology with the Extensible Clearing System from onExchange, a derivatives and securities trade-processing technology vendor to clearinghouses, banks, and brokerage firms. The system, powered by BEA WebLogic Server, can enable automatic integration with IBM WebSphereMQ messaging products, a business integration product series. The newly implemented NYBOT SOA system will provide reduced risk, cost, and complexity, and ultimately help change the IT economics of the exchange. www.nybot.com

Arrow Electronics, BEA Systems Launch Channel Partnership

(San Jose, CA, and Englewood, CO) – The North American Computer Products group of Arrow Electronics, Inc., and BEA Systems, Inc., have signed a distribution agreement for North America. The agreement provides a select number of Arrow's 1,500 Sun, HP, and IBM partners with access to the BEA WebLogic Enterprise Platform product line.

This partnership marks BEA's first distribution agreement with a Sun-authorized distributor, giving BEA greater ability to reach the Sun-authorized VAR base.

The North American Computer Products group of Arrow Electronics, Inc., is a leading distributor of enterprise and embedded computing systems to resellers and OEM customers in North America. www.arrow.com

BEA ExpandsPortlet Library; Offers New Tools for WebLogic Portal

(San Jose, CA) – BEA Systems, Inc., has expanded its portlet library, upgraded existing portlet technologies, and enhanced collaboration and integration tools for BEA WebLogic Portal 8.1. These expansions and upgrades include new portlets-services and applications that are designed to be easily integrated into portals – for SAP and Siebel; Compoze Portlets, WebLogic Edition for groupware and collaboration; and improved WebLogic Edition tools for collaboration and integration from Kapow Technologies. www.bea.com

BEA Systems and BMC Software Expand Alliance

(New York) – BMC Software, Inc., a provider of enterprise management solutions, and BEA Systems, Inc., have expanded their alliance to create products and services designed to extend BSM solutions and help integrate BMC Service Impact Manager and BEA Business Process Management (a component of WebLogic Integration). Together, these solutions will be designed to deliver comprehensive business process management as part of an infrastructure, application, service impact, and event management strategy that can help enable customers to manage IT to meet the changing needs of business.

The expanded BMC Software and BEA relationship will create solutions that can help customers take preemptive action to correct IT problems before they create a business process outage. BSM complements BEA Systems' services-oriented architecture (SOA) approach, which can help customers manage services based on business priority. www.bmc.com

Calpine Deploys Portals on BEA WebLogic Portal 8.1

(San Jose, CA) – Calpine Corporation, a North American power company, has implemented two business-critical portals on BEA WebLogic Portal 8.1.

Calpine employs 3,500 people at plants it owns or leases in 21 states in the U.S., Canada, The Netherlands, and Great Britain. Processes implemented when it was a smaller company needed to evolve to support the requirements of a bigger one.

Calpine was able to implement the new portals on existing hardware, potentially saving the cost of new infrastructure investment. In addition, they were able to incorporate existing applications, which were built with the company's incumbent development technology standard (.NET), into the BEA-powered portals. www.calpine.com

True application management starts from within.

A problem isn't a problem if it never happens.
Increase your visibility.
Decrease your risk.
Get control.
Intersperse Manager.

Enterprise information systems are more complex – and vital – than ever. But visibility into these dynamic architectures is minimal, increasing the chances of system failure and serious risk to your company.

Intersperse Manager™ removes the risks by proactively monitoring and managing distributed and SOA applications deployed on BEA. It gives users the ability to dive deeply into system infrastructure. To quickly pinpoint and correlate the root causes of problems. To automatically or manually correct issues in real time.

Intersperse Manager gives you complete visibility – and that puts you in complete control.

Call Intersperse today for
a free demonstration

866.499.2202

intersperse™

Management solutions
for the evolving enterprise.

www.intersperse.com