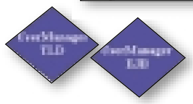
IMPROVE CONTACTS WITH YOUR CUSTOMER

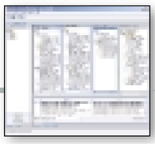
BEA WebLogic DEVELOPER'S JOURNAL

APRIL 2003 - Volume:2 Issue:4

weblogicdevelopersjournal.com

SELECTED NEWS HIGHLIGHTS FROM 2003 BEA eWorld
SEE PAGE 50





RETAILERS PLEASE DISPLAY UNTIL MAY 31, 2003
$8.99US $9.99CAN

SYS-CON MEDIA

ReportingEngines

www.reportingengines.com/download/wldj1.jsp

Attachmate

www.attachmate.com/SCb

Wily Technology

www.wilytech.com

BEA WebLogic DEVELOPER'S JOURNAL

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

FOUNDING EDITOR
PETER ZADROZNY

EDITOR-IN-CHIEF
JASON WESTRA

EDITORIAL DIRECTOR
JEREMY GEELAN

MANAGING EDITOR
SEAN RHODY

EXECUTIVE EDITOR
GAIL SCHULTZ

EDITOR
NANCY VALENTINE

ASSOCIATE EDITORS
JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR
JENNIFER STILLEY

WRITERS IN THIS ISSUE
KUMAR ALLAMRAJU, STEVE BUZZARD, MIKE CAREY, LEWIS CIRNE, PETER HOLDITCH, MANUEL HURTADO, MONTE KLUEMPER, RYAN LECOMPTE, NIGIN MANGTANI, BENJAMIN RENAUD, SEAN RHODY, MARK SECRIST, RAJESH SUMRA, RYAN UPTON, TRAVIS WISSINK

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

PRESIDENT AND CEO
FUAT A. KIRCAALI

VP, BUSINESS DEVELOPMENT
GRISHA DAVIDA

SENIOR VP, SALES & MARKETING
CARMEN GONZALEZ

PRODUCTION CONSULTANT
JIM MORGAN

ART DIRECTOR
ALEX BOTERO

ASSOCIATE ART DIRECTORS
LOUIS F. CUFFARI • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR
TAMI BEATTY

VP, SALES & MARKETING
MILES SILVERMAN

ADVERTISING SALES DIRECTOR
ROBYN FORMA

ADVERTISING ACCOUNT MANAGER
MEGAN RING-MUSSA

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

PRESIDENT, SYS-CON EVENTS
GRISHA DAVIDA

CONFERENCE MANAGER
MICHAEL LYNCH

EXHIBITOR RELATIONS
NIKI PANAGOPOULOS

FINANCIAL ANALYST
JOAN LAROSE

ACCOUNTS RECEIVABLE
KERRI VON ACHEN

ACCOUNTS PAYABLE
BETTY WHITE

VP, INFORMATION SYSTEMS
ROBERT DIAMOND

WEB DESIGNERS
STEPHEN KILMURRAY • CHRISTOPHER CROCE

ONLINE EDITOR
LIN GOETZ

CUSTOMER SERVICE REPRESENTATIVE
MARGIE DOWNS

JDJ STORE MANAGER
RACHEL MCGOURAN

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

SYS-CON MEDIA

Portal to the World

BY SEAN RHODY
MANAGING EDITOR

One of the more interesting conversations I have with IT organizations is over what constitutes a portal. While issues vary, it is clear that there is a business definition of a portal that is distinctly different from the technology definition of a portal. That isn't necessarily a bad thing, as long as you can separate the two and clearly make the transition between one definition and the other. When that doesn't happen, a bit of chaos can result.

Most businesses see a portal as a means of consolidating the way the Internet "views" their company. As such, it's used to standardize a look and feel, promote branding, and in many cases provide a personalized user experience that in theory increases the stickiness and friendliness of the site.

What sometimes gets in the way for the IT organization is its translation into applications. I've spent a great deal of time helping groups define the distinction between "content" and "application" because that's where the problems with portals really tend to crop up.

Portal technology is pretty good at the personalization of content, but mostly useless at the personalization of applications. There's always some overlap in this area that makes it difficult to discern, but in general portals are good at a couple of things. One is mixing and matching (personalizing) content (HTML for the most part) based on user profiles. Another is profiling itself – tracking the user implicitly via links that are clicked, or explicitly via user-supplied information or company-defined groupings. Finally, portals are good at defining and providing screen navigation.

Surprised that things like providing a 10% discount to frequent customers aren't part of that group? Well, they can be, but in most cases it's really additional vertical functionality such as an e-commerce package that is providing the ability, rather than the portal itself.

With this definition it's somewhat easier to understand how the business definition and expectations can get out of line with the technical definition. The business users in most cases want to unite their applications, and want to have a single interface to do so. They want to define business rules and have them carried out effectively through the delivery mechanism, which, of course, is a portal. Sometimes where the functionality actually resides gets lost in the shuffle, as the important thing is that the functionality exists, not which package implements it. As long as you're a business user, that is.

For the technical user, where the functionality is definitely has an impact. It determines whether you need someone who understands stored procedures, or who can write Java code, for example.

The problem with portals is that they are short on application functionality and long on content functionality, a result of their origin in the Internet world. Sometimes even if the portal provides functionality, the corporate architecture dictates a different approach. A clear example of this is that most portals provide a mechanism for direct database access, but most IT organizations prefer to restrict this and have the access go through a middle tier (for reasons such as caching and consolidation of business logic). This makes sense to the IT staff, but may be lost on the business side of the house. You end up with the classic argument: "We need a portal. No, what we really need is an application server."

continued on page 49

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

CONTACT: sean@sys-con.com

PORTALIZE

A GUIDE TO ADAPTING AN EXISTING WEB APPLICATION TO A PORTAL PARADIGM

BY **MARK SECRIST**

In recent years, Web portals have risen in popularity as a way of aggregating, organizing, and presenting content in a highly uniform, customizable, and personalized way.

As the technologies that enable the creation and management of these Web portals have evolved, it is not only information content that is being offered, but application functionality as well. With application functionality making its way to the Web portal, a whole new dilemma arises as developers attempt to adapt their application functionality to the characteristics and behavior of the Web portal.

Portal Concepts

In the simplest terms, a *portal* is a Web site that provides content and application functionality in a way that is both useful and meaningful to the end user. It also serves some purpose from the portal provider's perspective, whether it is a public portal trying to attract Web traffic to a site or an enterprise desiring to provide a central location for employees to obtain company information. Portals have evolved to enable the ability not only to provide a unified look and feel to the entire site, but to apply personalization and customization to the content. In this way, a person who accesses a portal and logs in with a predefined user name and password can customize not only *what* portal content is displayed for them, but *how* that content is displayed.

Most portals and portal frameworks contain the concept of a "portlet" as a window to a specific set of content within the overall context of the portal page. Portlets generally support the ability to customize the information displayed within this window. From the perspective of the portal framework, portlets tend to look and behave much the same as individual windows running in any windows-based operating system. They can be minimized, maximized, and rearranged to suit the individual portal user. From the developer perspective, a portlet is really a piece of code that plugs into a generalized framework. Different portal frameworks implement the concept of a portlet differently. In some cases, the

AUTHOR BIO...

Mark Secrist is a senior constultant for the HP Developer Resources Organization with more than 10 years of experience involving distributed object technologies and building n-tier, Web-based applications. He currently consults with enterprise customers on J2EE and Web services development.

CONTACT...

mark_secrist@hp.com

portlet is a collection of JSP pages. In other cases, a special type of class may implement certain interfaces. Regardless of how it is implemented, the portlet is generally responsible for presenting a specific set of content that may be tailored to a user's preferences. The portal framework handles the infrastructure services, such as providing the overall presentation of the selected portlets, user management, security, and personalization.

A Developer's Approach

When you think about adapting an existing application, it's important to understand that typical Web applications don't necessarily map easily to a portal paradigm. This is especially true for Web applications with multipage interactions, and is due to the fundamental difference in navigation behavior inside a portlet as compared to a traditional Web application or Web page. In a traditional Web page, a user would typically select a link on the page presented in the browser, which would lead to another HTTP request returning a completely new page of content. By contrast, the goal of a portlet within a portal is more to present to the user the impression that the contents of only that one portlet are changing. This presents a distinct challenge to developers who typically have designed their Web application navigation as a set of URL links that invoke JSPs, Java servlets, or even static HTML pages. Now they must think about how to specify navigation, not in the context of the Web application, but from the perspective of the overall portal.

In order to present to the user the appearance that only the portlet they are interacting with is changing, it is necessary to use functionality that allows the portal framework to intercept the result of URL invocations and redirect them to the portlet making the invocation. In BEA WebLogic Portal, a mechanism called Webflow can be used to define the flow of events that take place between components of the Web application. These Webflows effectively take the place of actual URLs in the execution of the various actions within the application. What follows is an example of how we adapted a Web application called "QuizGame" into an existing portal domain running on BEA WebLogic Portal.

The QuizGame application is a demo Web application built using a combination of a single controller servlet, a number of Enterprise JavaBeans, and a single JavaServer Page for handling the generation of the presentation. Figure 1 represents the architecture of this application.

In order to adapt the QuizGame Web application to WebLogic Portal, three key steps need to take place.

1. The Web application components must be added to the portal Web application.

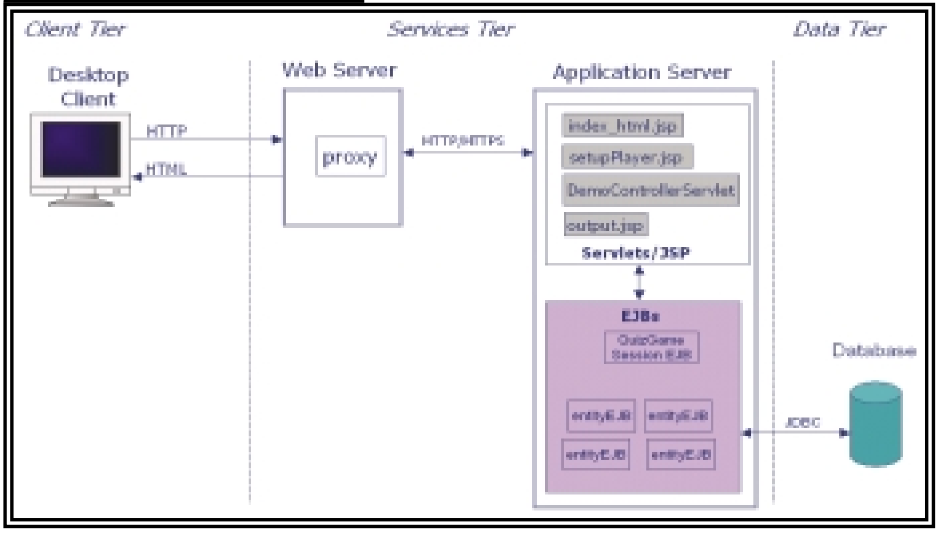
This includes the EJBs and WAR file containing the controller servlet and other helper classes used by the Web application. In WebLogic Portal, the portal itself is implemented as a sophisticated Web application. Within the portal Web application, a subdirectory is defined where the different portlets are stored. The portlets are typically JSPs connected together through the definition of Webflow events. Any servlets and helper classes should be stored in the typical Java Webapp structure with exploded individual classes stored under the "classes" directory and classes packaged in JAR files stored under the "lib" directory. In addition, servlets and the components they reference must be registered with the Web application by adding them to the Web.xml file. Once the portlet is created in the next step, the JSP files should be stored in the portlet-specific directory. These files will be adapted in the final step.

2. A new portlet must be created and added to the existing portal domain E-Business Control Center (EBCC).

This portlet will contain the pages representing the view for the Web application being added. Once the portlet itself is created, a Webflow needs to be defined to represent the interactions between the various components of the presentation. The events defined within this Webflow will replace the actual URLs previously defined in the QuizGame application. Figure 2 shows an example of the Webflow created in the EBCC.

Figure 2 shows two kinds of elements. Presentation nodes represent various kinds of presentation components, such as static HTML pages, Java servlets, or JSPs. These presentation nodes can have a user-friendly name, but must reference the actual

FIGURE 1



QuizGame architecture

component or page. The Webflow events are shown as arrows and represent invocations between the various presentation components. These events can be given user-defined names.

3. The URLs defined in the original Web application must be replaced with a special URL built from the Webflow events defined in the EBCC.

Due to how the QuizGame application was built, the URLs were largely contained in the output JSP. Changing these URLs in the JSP files is an easy task due to the availability of a helper TagLib. Listings 1 and 2 show how to replace a URL invocation to the DemoControllerServlet with the equivalent Webflow invocation to the same servlet. Listing 1 shows a traditional form with a URL invocation to the DemoControllerServlet as the action. Listing 2 shows the use of the event "QGControllerServlet.event" to represent the same request using a Webflow event. This was one of the events defined between the Index_html presentation node and the DemoControllerServlet presentation node in the EBCC Webflow designer tool.

These three steps will enable the application to run within the overall portal and behave correctly throughout the execution of the application. This is accomplished without significantly changing the look or actual behavior of the application. The main difference is that now the QuizGame application is simply one portlet window within the greater portal presentation.

Once the application has been successfully portalized, additional enhancements can be made, such as adding the ability to customize properties of the game through an edit page. This shouldn't be considered the end of the process, but simply the first step in integrating an application into the portal. BEA WebLogic Portal provides more sophisticated mechanisms for controlling the flow and behavior of the application through the use of pipeline components and processor components.

The use of these types of components can replace the need for the servlet and JSPs and enable better flow control and access to other portal features, such as security and personalization. At the same time, they still allow the ability to reuse the existing business logic components, such as EJBs.

Developing Portal-Ready Applications

Up to this point, this article has discussed portalization from the perspective of moving existing application functionality to a portal framework. If you are creating a new Web application, a number of techniques can be used to make the application more portal ready.

Tip 1: Use a Model-View-Control Architecture

The Model-View-Control (MVC) architecture is a way of grouping application functionality into three distinct categories, each serving a specific role in the overall application. The objective of using an architecture such as MVC is to allow the ability to isolate each of these functions so that it is possible to make changes to one component without significantly impacting the others. For example, a change may be made to the view aspect of an application without requiring any changes to either the model or the control modules.

This architecture provides a distinct advantage in portalization. MVC-based applications can be more easily adapted to a portal by simply making changes to the view and possibly the control code. Even if there is a certain business logic associated with the application, this can often be left untouched if it is not tightly woven into the view and control components. More specifically, an application can be adapted to the portal by first only changing the view. These changes are often superficial in order to adapt to a different form factor than may have been used for a browser. Then the control module can be modified or even replaced to suit the needs of navigation within the context of the portal.

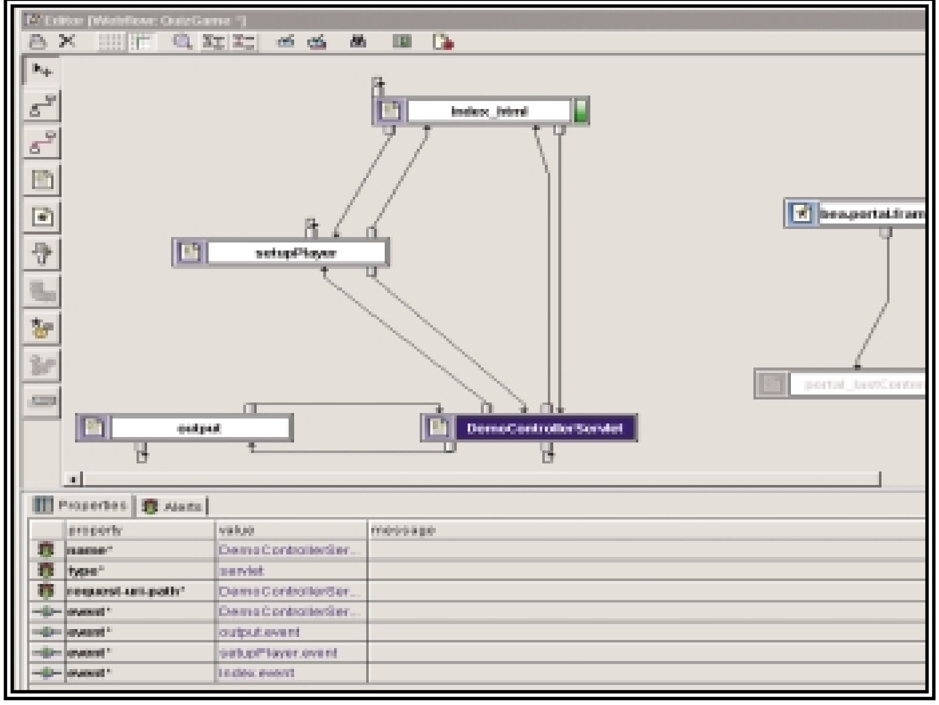
Tip 2: Use XML to Represent Content

In Web applications, XML is often used as a client-neutral way of describing what will be presented. XML coupled with transformation technologies such as XSLT allows the ability to delay the decision of how to present the content until the last stages of processing. Delaying this decision allows you to tailor the presentation of the content in very client-specific ways.

In the context of adapting functionality to portals, this allows the ability to contain the definition of how the content is presented to a single mechanism, such as XSLT and the corresponding stylesheets. Doing this makes it easier to change the presentation and navigation parts of the application without significantly impacting other parts of the application. In order to use XML in this way, the code responsible for controlling the behavior and interaction should be made to return XML rather than presentation-specific content such as HTML.

The focus in defining the presentation should be defining the basic presentation constructs, such as tables, forms, fields, etc. The explicit definition of things like fonts and background and foreground colors can defeat the ability to personalize the portlet to the overall portal color schemes and font styles.

FIGURE 2



property	value	message
name*	DemoControllerSer...	
type*	servlet	
request-uri-path*	DemoControllerSer...	
event*	DemoControllerSer...	
event*	output.event	
event*	setupPlayer.event	
event*	index.event	

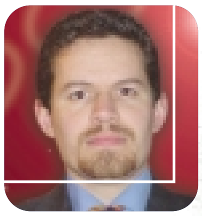
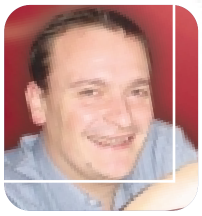
Using the EBCC to define a Webflow

continued on page 18

Quest Software

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

Extending the Product Catalog – Supporting Configurable Products

BY **MONTE KLUEMPER & MANUEL HURTADO**

AUTHOR BIOS...

Monte Kluemper got his first taste of Java while investigating technology options for a telco client in 1996. He has worked with such distinguished app servers as NetDynamics, JAM (Prolifics), PowerBuilder/J, NAS, and Oracle App Server, before dedicating himelf to WebLogic in 1999.

Manuel Hurtado joined BEA as a consultant in 2001. He has been working with Java since 1996, focusing mainly on J2EE architecture and security.

CONTACT...

monte.kluemper@bea.com
manuel.hurtado@bea.com

As more and more companies focus on providing higher levels of personalization in the products and services they offer, it's only natural that their online channels offer this same level of personalization. WebLogic Portal includes product catalog components capable of flexibly managing hierarchies of products and services. By default, products in the catalog are defined using standard Dublin Core attributes. This default configuration is sufficient to support most types of products sold over the Internet – books, hardware, camaras, etc.

However, certain types of products require that customers select from among desired options before the products can be purchased. We can group these "configurable" products into three main categories based on their configuration needs:

- ***Light product configurations:*** Products that have a small number of simple, mandatory configurable attributes. Product pricing is normally not affected by the options selected (e.g., T-shirts defined by color and size).
- ***Heavy product configurations:*** Products defined by multiple attributes or product options. Depending on the options selected, pricing and delivery may be affected (e.g., insurance policies or telecommunications services).
- ***Product bundles:*** Products composed of component products, each with its own attributes (e.g., desktop computers or automobiles).

Although the product catalog does not provide any OOTB components to support configurable products, we will attempt to illustrate how the catalog can easily be extended to support such products. We will use the T-shirt example to illustrate our solution. Later, we will discuss design strategies for heavy product configurations and product bundles.

We identified several goals in the solution design. First, the solution had to provide an acceptable level of performance for the users wanting to purchase configurable items. Second, any new components should integrate not only with the catalog, but also with the order process, discount, and campaign components. Third, new functionality should not require changes to existing components, thereby minimizing future migration problems. Finally, administrators should be able to easily manage configurable items within the catalog hierarchy as a single item.

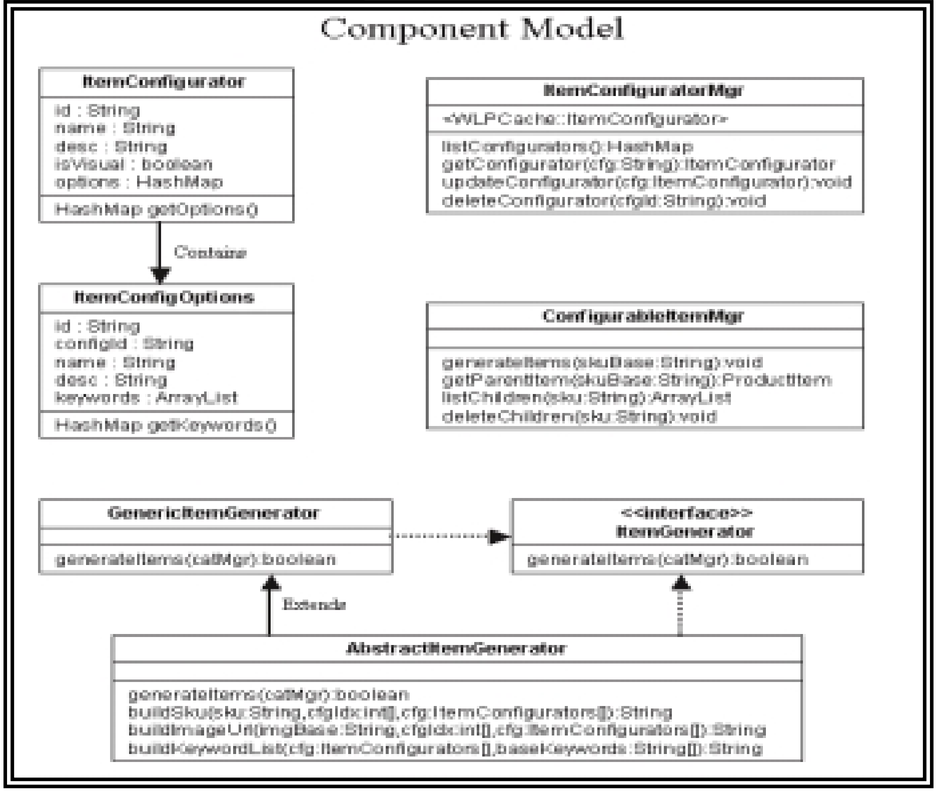
Item Configurators

In order to support light product configurations, we added a new component to the product catalog – item configurators. An *item configurator* represents a list of options that can be assigned to a product. In the T-shirts example, both color and size would be item configurators.

Figure 1 shows the component model representing the item configurators and the options associated with each configurator. Each component is represented in the database by a table of the same name. Instead of directly accessing the configurator components, we have created a Business Manager component called the ItemConfiguratorMgr. This component is responsible for providing configurator information to calling objects. The ItemConfiguratorMgr uses WebLogic Portal's Cache Framework to optimize access to the data. This framework also allows us to monitor the use of the product configurators.

We recommend that you name item configurators with reuse in mind. In the T-shirt example, if you don't expect the list of T-shirt colors to be reused for other items, you should name the color configurator explicitly for T-shirts. On the other hand, the list of T-shirt sizes will probably be identical to other types of shirts. Therefore, the size configurator should be given a more generic name in order to promote reuse.

FIGURE 1


Component Model
ItemConfigurator
id : String
name : String
desc : String
isVisual : boolean
options : HashMap
HashMap getOptions()
Contains
ItemConfigOptions
id : String
configId : String
name : String
desc : String
keywords : ArrayList
HashMap getKeywords()
ItemConfiguratorMgr
<WLPCache::ItemConfigurator>
listConfigurators():HashMap
getConfigurator(cfg:String):ItemConfigurator
updateConfigurator(cfg:ItemConfigurator):void
deleteConfigurator(cfgId:String):void
ConfigurableItemMgr
generateItems(skuBase:String):void
getParentItem(skuBase:String):ProductItem
listChildren(sku:String):ArrayList
deleteChildren(sku:String):void
GenericItemGenerator
generateItems(catMgr):boolean
<<interface>>
ItemGenerator
generateItems(catMgr):boolean
Extends
AbstractItemGenerator
generateItems(catMgr):boolean
buildSku(sku:String,cfgIdx:int[],cfg:ItemConfigurators[]):String
buildImageUrl(imgBase:String,cfgIdx:int[],cfg:ItemConfigurators[]):String
buildKeywordList(cfg:ItemConfigurators[],baseKeywords:String[]):String

Component Model

Creating Configurable Items

Now that we have created our two item configurators – color and size – we need to tie them to our T-shirts. This is accomplished by first creating a generic T-shirt item using the standard catalog administration screens. Since this generic item will not be shown directly to the customer, you should make sure that the visible flag is not checked.

Next, we need to associate the configurators with the item. In addition to the standard Dublin Core attributes, WebLogic Portal facilitates the creation of custom item attributes, grouped into Catalog Structures. We have created a new Catalog Structure called ConfigurableItems, containing the following properties:

- ***Configurator1…n:*** Ordered list of configurator IDs 1 through n. We created three configurators as the maximum number of configurators associated with any one product. If more configurators are desired, they can be added through the EBCC and synchronized with the instance of WebLogic.
- ***GeneratorClass:*** Full name of class used to generate the various item configurations. We have provided a ConfigurableItemGenerator class to be used by default.

The GeneratorClass is the key to our solution. This class will generate new item SKUs based on the order of the listed configurators for a given item. The format of the generated SKUs will be the base item SKU plus a hyphen-separated list of the corresponding options in the order defined by the ConfigurableItems structure.

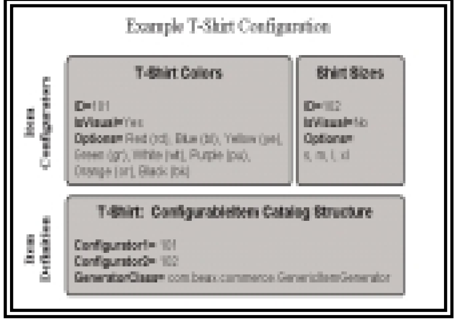
Going back to the T-shirt example, let's say we've created two configurators – T-Shirt Colors (ID=101) and Shirt Sizes (ID=102). We've also created a T-Shirt product item in the catalog with a SKU of 1001. Using our new ConfigurableItems Catalog Structure, we assign Configurator1 to 101 (colors) and Configurator2 to 102 (sizes). The class ConfigurableItemGenerator will create a new product for each combination of configurators. That is, if there are 8 colors and 4 sizes of T-shirts, the ConfigurableItemGenerator will create 32 new T-shirt items. The SKU associated with small, red T-shirts, for example, would be (1001-RD-S). Figure 2 shows an example configuration for our T-shirts.

Our ConfigurableItemGenerator can create distinct image names for each configured item. The "isVisual" attribute of the ItemConfigurators is used to determine which configurators will be used in generating image names. In our example, color is a visual configurator, but size is not. Therefore, only color will be used to generate the image name. If the base T-shirt product has an image URL of "images/tshirt.jpg", all red T-shirts will have "images/tshirt-RD.jpg" as their image URL. Likewise, if there are no visual configurators assigned to an item, all configured items will inherit the base item's image URL.

The ConfigurableItemGenerator also adds the keywords associated with each ItemConfigurator to the generated items. These keywords are added to the list of keywords associated with the base item.

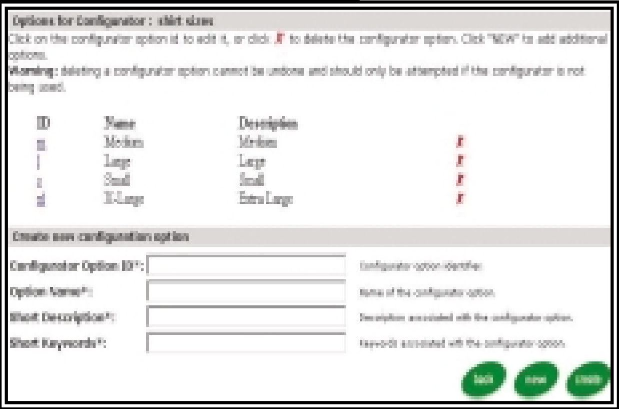
Each generic item can have a custom GeneratorClass that defines the process of generating configurable items. By allowing

FIGURE 2


Example T-Shirt Configuration
Item Configurators
T-Shirt Colors
ID=101
IsVisual=Yes
Options= Red (rd), Blue (bl), Yellow (ye), Green (gr), White (wt), Purple (pu), Orange (or), Black (bk)
Shirt Sizes
ID=102
IsVisual=No
Options=
s, m, l, xl
Item Definitions
T-Shirt: ConfigurableItem Catalog Structure
Configurator1= 101
Configurator2= 102
GeneratorClass= com.bea.commerce.GenericItemGenerator

Example Configuration

FIGURE 3



Options for Configurator : shirt sizes

Click on the configurator option id to edit it, or click to delete the configurator option. Click "NEW" to add additional options.

Warning: deleting a configurator option cannot be undone and should only be attempted if the configurator is not being used.

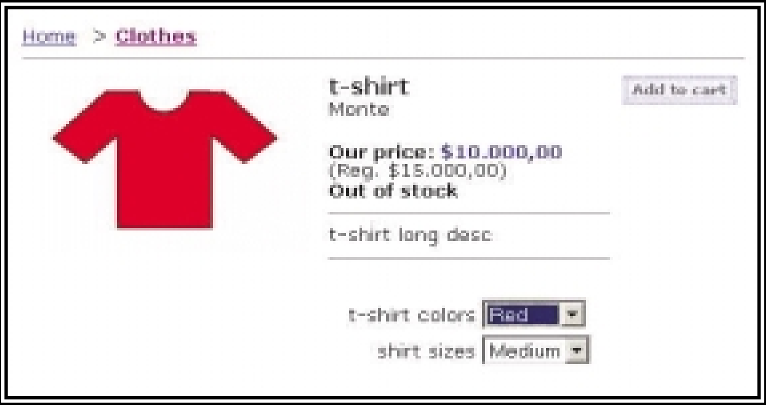
ID	Name	Description
m	Medium	Medium
l	Large	Large
s	Small	Small
xl	X-Large	Extra Large

Create new configuration option

Configurator Option ID*:		Configurator option identifier.
Option Name*:		Name of the configurator option.
Short Description*:		Description associated with the configurator option.
Short Keywords*:		Keywords associated with the configurator option.

Configurator Administration

FIGURE 4



Home > Clothes

t-shirt
Monte

Our price: $10.000,00
(Reg. $15.000,00)
Out of stock

t-shirt long desc

t-shirt colors: Red
shirt sizes: Medium

Product Detail Page

you to assign a different GeneratorClass to each item, we have attempted to make the solution as flexible and pluggable as possible.

Managing Configurable Items

In order to effectively manage configurable items, we need to add two new options to the Catalog Management administration page (see Figure 3):

- ***Item Configurators:*** Add, delete, search, and modify configurators
- ***Configurable Items:*** Generate, delete, and manage configurable items

These items will have a look-and-feel that is very similar to the current category and item administration pages.

Clicking on the Item Configurators page link brings you to a list of current item configurators. Administrators will be able to edit the information associated with a particular configurator or delete it from the database. The Create Item Configurator button brings the administrator to a form where he or she can create new configurators.

Clicking on the Configurable Items page brings you to a list of management tasks for managing configurable items in batch. First, you can view the list of generic items that have an associated GeneratorClass for creating configurable items. For each item, the administrator will have the ability to generate configured items, delete generated items, etc.

Much of the administration of Configurable Items will be accomplished using the standard administration pages. For example, to create a generic item you can use the standard item create page to create the non-visible item. Then simply edit the ConfigurableItems properties associated with the product and add the desired GeneratorClass and one or more Configurators.

Similarly, configurable products can be assigned to a category using the "Modify Items Assigned to Category" page. Here we made a slight change to the category_add_remove_items.jsp that checks added items to see if they have an associated GeneratorClass. If so, it also adds all generated items. Once all items have been assigned to the category, individual items can be unassigned using the same page.

User Navigation

Our solution does not require any changes to the user interface of the catalog. As you navigate through the hierarchy of categories and items, all the newly generated items will appear under their assigned categories. The search engine will also find any generated items that meet the search criteria.

However, in many cases it won't be desirable to show all combinations of configurable items as separate products. In these cases, we recommend that the configurable product be assigned to a dedicated category. Creating such a category will allow you to create and assign a custom layout that, instead of presenting a list of all generated items, presents the generic item information and the associated configurator options. Once the user selects the desired configurator options, the page automatically places the corresponding generated product in the user's shopping cart. Once you design a custom layout JSP, you can associate it with the dedicated category via the category attribute "Display JSP URL".

Creating a dedicated category offers an additional benefit, namely the ability to treat the configured products as a single entity when creating product discounts. Discounts can be offered on either individual products or products within a single category. Our dedicated category approach makes it easy to create a discount that requires the purchase of one or more products of any configuration from the category.

Benefits of this Approach

Our support for configurable products is based primarily on automating the process of generating new SKUs for each combination of configuration options available for a given product. We believe that this solution meets the objectives set out at the start of this article:

- ***End-user performance:*** Although this solution increases the total number of items in the catalog, the end user still benefits from the optimized cache of the Catalog Manager in navigating through the catalog.
- ***Integration with other Portal components:*** Since we have not touched the Catalog Manager nor the Shopping Cart in our implementation, all integration points among the components should be intact.
- ***Migration to future Portal versions:*** Since we have not altered any core portal components (only a few minor changes to JSPs), we have minimized the risk of losing functionality in future versions of WebLogic Portal.
- ***Easy administration:*** While there will always be ways to automate and facilitate administration tasks, we have provided a set of easy-to-use, integrated admin screens that should prevent configurated product generation from getting out-of-hand.

This example was built by extending the example wlcsDomain that comes with WebLogic Portal 7.0 sp2. It is published on http://dev2dev.bea.com as an unsupported example.

PANACYA

www.panacya.com

Customizing User Profiles with Portal

This article focuses on the User Management framework of WebLogic Portal. Specifically I will discuss a small portion of Portal's User Profile Management features by detailing how to customize Portal's User Manager functions and extend the User Management framework to build a robust Unified User Profile.

BY **RYAN UPTON**

AUTHOR BIO...

Ryan Upton is a senior technologist with Learning Voyage Inc., BEA Education Service's premier training partner. Ryan specializes in training and mentoring developers on all fronts of BEA technology and application development and server administration.

CONTACT...

rupton@learningvoyage.com

Personalization

Quite possibly one of the most widely used features of WebLogic Portal is the Personalization service. Personalization gives a portal application the ability to provide a personal touch by selecting specialized content and ads or by defining rules and user segments or even running complex business campaigns. These personalization choices are based on a set of user properties that can be defined in multiple places.

Properties

Properties represent pieces of useful user information: personal data, demographic data, or something of equal importance that can be logically grouped together into something called a property set and referenced directly from a portal application either through the Enterprise JavaBeans that map them or with JSP Custom Tag Library functions. Actually these properties and property sets can come from many sources: HTTP Requests, HTTP Session objects, predefined user preferences, or persistent attributes from a user profile. All of these are configured separately in the Electronic Business Control Center (EBCC) (see Figure 1).

Each property set created in the User Profile, Request, Session, or Events tabs can be accessed by your portal applications. Although the creation of property sets and their properties is done in a consistent fashion, this last one, User Profile, is the one we'll focus on here. By default, properties are stored in the Portal schema, the default-persistent framework that Portal uses to store all WebLogic Portal–persistent attributes for personalization and other services (see Figure 2).

Although a complete discussion of the Personalization framework is beyond the scope of this article, I'll discuss just enough of it to provide a context for managing user profiles.

The Moving Pieces

The Personalization service calls many different subservices in its zeal to provide the appropriate content; one of the main services called is the User

OFFERING THE PERSONAL TOUCH

TABLE 1

User Authentication	Working with WebLogic Security Services it handles user authentication
User to Group Association	Associating a user to one or more groups defined locally or in the Portal Schema or in an outside source
User Profile Management	Building a user profile consisting of attributes from user and group profile information defined in the Portal Schema or in an outside source

UserManager responsibilities

Management service. The User Management service consists of UserManager and GroupManager Session EJBs that act as the liaison between User Profile Management and User Security Management (see Table 1). Although user management is performed with both User and Group Management, the primary focus here will be on User Management. The Group Manager mainly allows for the inheritance of properties by defining Successor Keys; Successor Keys can be used by Profile Managers to build a more complex user profile.

The User Manager provides methods for creating users, creating groups, assigning users to groups, and deleting users. These methods directly interact with WebLogic Server's security realm. The User Manager also references the Profile Manager and Profile Wrapper classes that are used for creating and working with profiles and keeping profiles synchronized with users. A ProfileManager is a Session EJB that provides direct access to a user's profile. Both a user and group profile may be used to specify attributes for the individual user or to define a succession of attributes via the group. The ProfileManager accesses profiles via an EntityProperty-Manager EJB that is defined in the ProfileManager's deployment descriptor. The ProfileWrapper is a lightweight object used to provide direct access to various profiles through interaction with a single or multiple Profile Managers; you can think of this as a façade that clients can use to access the ProfileManager. The ProfileWrapper determines which ProfileManager to use based on the user or group identity it was initialized with. Initialization can also be done with both a group and a user identity that will allow for inheritance of properties through succession. ProfileWrappers are created with the Profile-Factory Singleton object; all three of these classes are located in the com.p13n.usermgt.profile factory and can be customized.

As I mentioned earlier, by default WebLogic Portal supports user profiles locally by storing properties in the portal schema; in addition, it provides the mechanisms to locate profile information from external sources such as legacy applications, other RDMBS', LDAP servers, or even flat files. All these properties are combined into a Unified User Profile (UUP; see Figure 3) that can be used directly from your portal applications. All types of profiles are, in fact, the user property sets that we will be using with Personalization.

The UUP allows you to build more complex profiles by leveraging existing user data already stored in an enterprise system. The benefits of this, of course, come from using existing user information without having to migrate that data into the portal schema; for example, you might have a UUP that contains order-specific properties from an order fulfillment database combined with personal customer data retrieved from an LDAP-provided customer profile the user created when registering with your site. A UUP provides many benefits; three of the most obvious are:

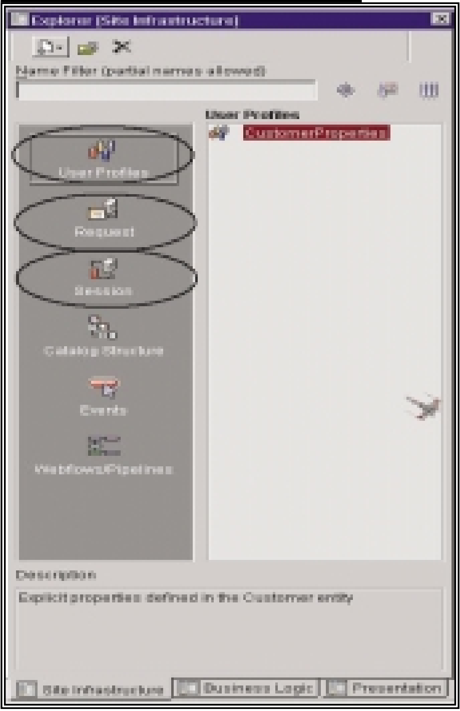
- Reuse of user information
- More complex user properties
- Transparent access to user profiles for portlet and content developers

UUPs are composed of multiple profile types managed by a ProfileWrapper working with the ProfileManager and standard or customized EntityPropertyManager objects. The EntityPropertyManager is located in the com.bea.p13n.property package and is simply a remote interface to a Session EJB. EntityPropertyManagers provide the persistence mapping between users and their properties and are directly accessed by the PropertyManager EJB to build the UUP. Portal ships with EnterprisePropertyManagers that map persistent attributes of the Portal schema but supports extensibility by allowing Portal developers to create their own property mappings. To build a complex UUP we take advantage of this extensibility and write our own Session EJBs to handle the mapping of properties to our other data sources.

Implementation

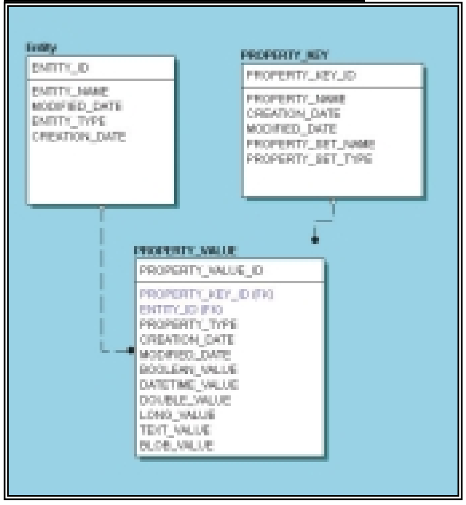
Building the UUP is pretty straightforward. Although I introduced a complex relationship between all the classes above, in most cases we don't need to worry about customizing the UserManager or ProfileManager EJBs, which will keep things fairly simple. In fact the only thing we really need to do is to create our own Session EJB that implements the com.bea.p13n.property.EntityPropertyManager interface and register it with the UserProfileManager EJB. When creating your custom EntityPropertyManager it's a good idea to extend the existing EntityPropertyManager and not replace it. You will want to leave the default EntityPropertyManager of the UserProfile-Manager in place and add your own

FIGURE 1



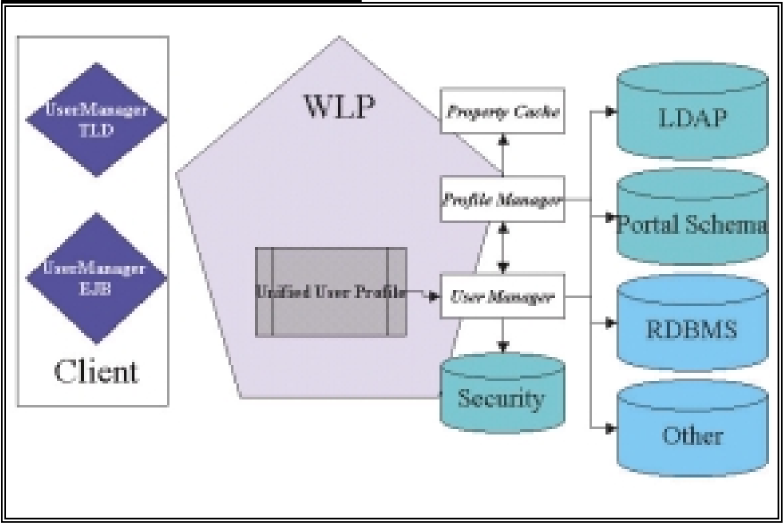
User Profile Explorer

FIGURE 2



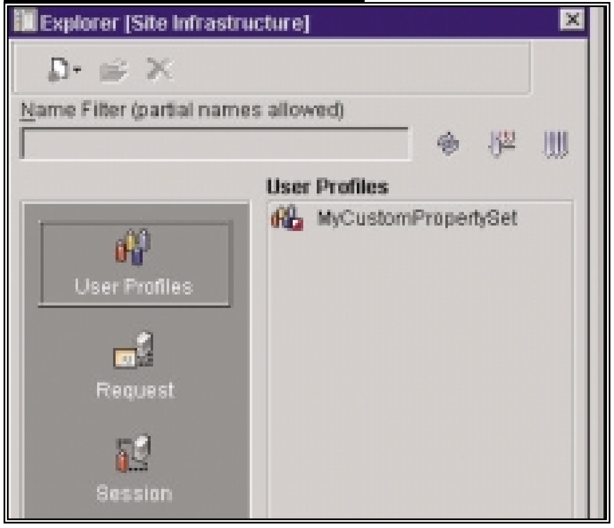
PropertySchema

FIGURE 3



Unified User Profile

FIGURE 4


Explorer [Site Infrastructure]
Name Filter (partial names allowed)
User Profiles
MyCustomPropertySet
User Profiles
Request
Session

Completed Electronic Business Control Center

EntityPropertyManagers to supplement it. One benefit of this is that you won't have to implement every method defined in the interface, you'll simply need to worry about implementing the basic methods to retrieve the properties from your data source. In our EntityPropertyManager EJB we must create the Home Interface, Remote Interface, and implementation class. Remember that in this case the EntityPropertyManager is the Remote Interface for our EJB. There's nothing remarkable about the Home Interface and our implementation class can either extend javax.ejb.SessionBean or a convenience class: com.bea.p13n.property.internal.EntityPropertyManagerImpl. If you aren't replacing the default EntityPropertyManager you won't have to implement the following methods:

- getDynamicProperties()
- getEntityNames()
- getHomeName()
- getPropertyLocator()
- getUniqueId()

In your implementation class these methods will simply throw java.lang.UnsupportedOperationException. Regardless of whether or not you are extending or customizing the default EntityPropertyManager you will have to support the following methods:

- getProperty()
- getProperties()
- setProperty()
- removeProperty()
- removeProperties()

This is where you define how the new EntityPropertyManager will connect to your data source and access user properties. Obviously this is pretty vague because it may be simple or complex depending on the type of data source you are accessing, but this is also where power and flexibility are introduced in the customization of User Profile management. One thing that should be mentioned here is that a call to any of these methods via JSP Tag Libraries or direct method invocation will result in a trip to the data source. Repetition of this is usually not a good thing so you'll want to take advantage of the property caching provided by the EntityPropertyCache. If your custom EntityPropertyManager supports the creation and removal of users then you will also need to provide implementation of the createUniqueId() and removeEntity() methods. Much as its name implies createUniqueId() is responsible for adding a user in the data source based on a unique value for that user, removeEntity() is responsible for not only removing the user but also all the user properties that have been added during his lifetime.

That's all there is to it. You're now ready to package your custom EntityPropertyManager EJB, deploy it, and register it with the UserProfileManager.

Deploy and Register

The User Manager framework is defined in the UserManager EJB located in usermgmt.jar. It is here that we can redefine not only the user and group managers but also the profile managers and their associated EntityPropertyManagers. After our EJB has been packaged we will need to make a slight modification to the deployment descriptor of the UserManager to map a property set to our new EntityPropertyManager. In the ejb-jar.xml deployment descriptor of the UserManager EJB we will add an environment entry setting to map our custom property set and an EJB reference to point to the EntityPropertyManager that handles our property set (see Listings 1 and 2).

Once the entries have been added to the ejb-jar.xml file the final step is to map them to physical EJB locations in the JNDI tree. The only thing we'll do here is find the existing entry for the UserProfileManager under <weblogic-enterprise-bean> and add a new reference description (see Listing 3).

Note that here ${APPNAME} is a substitution variable that points to the node portalApp under the InitialContext of our JNDI Tree. WebLogic Server will automatically perform the variable substitution. Once you have made the appropriate changes to the deployment descriptors you're ready to go. One last word on configuration: if your custom EntityPropertyManager supports adding and removing users then you will also need to add creator and remover environment entries as well (see Listing 4).

These are arbitrary since they will be mapped appropriately when a profile is added or removed.

Finishing Touches

The only thing remaining to do is to create a user profile in the EBCC that maps to the User Profile that you just created and registered with the UserManager EJB (see Figure 4).

Quest Software

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

Listing 1. Adding the environment entry

```
  <env-entry>
        <env-entry-name>PropertyMapping/MyCustomPropertySet</env-entry-
name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>MyCustomEntityPropertyManager</env-entry-value>
  </env-entry>
```

Listing 2. Adding the EJB reference

```
  <ejb-ref>
        <ejb-ref-name>ejb/ MyCustomEntityPropertyManager</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>com.myco.ProperyManagers.
MyCustomEntityPropertyManagerHome</home>
        <remote>
com.myco.ProperyManagers.CustomEntityPropertyManager</remote>
  </ejb-ref>
```

Listing 3. Adding the reference to our custom EntityPropertyManager

```
  <weblogic-enterprise-bean>
        <ejb-name>UserProfileManager</ejb-name>
        <reference-descriptor>
                 <ejb-reference-description>
                 <ejb-ref-name>ejb/EntityPropertyManager</ejb-ref-
name>
                       <jndi-name>
${APPNAME}.BEA_personalization.EntityPropertyManager</jndi-name>
                       </ejb-reference-description>

         <!—the entry above is there by default, now we add our custom
EJB mapping -->

<ejb-reference-description>
                       <ejb-ref-name>ejb/MyCustomEntityPropertyManager
</ejb-ref-name>
                       <jndi-name> ${APPNAME}.BEA_personalization.
MyCustomEntityPropertyManager </jndi-name>
                       </ejb-reference-description>
         </reference-descriptor>
```

Listing 4 Adding Creator and Remover entries

```
  <env-entry>
        <env-entry-name>Creator/ArbitraryCreatorName</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>MyCustomEntityPropertyManager</env-entry-value>
  </env-entry>
  <env-entry>
        <env-entry-name>Remover/ArbitraryRemoverName</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>MyCustomEntityPropertyManager</env-entry-value>
  </env-entry>
```

PORTAL

continued from page 8

Tip 3: Prepare to Leverage the Portal Security Mechanisms

Portal security is a fairly complex topic and not within the scope of this article except as it may affect the individual applications being adapted to the portal. One key benefit of a portal is the ability to sign on once to enable access to the applications offered within that portal – often referred to as “single sign-on.” For applications that will be hosted in the same middleware infrastructure as the portal framework, it is much easier to leverage the same security mechanisms. Many portals use a standard HTTP cookie to store session information. In order to enable the ability to have single sign-on, it is important that all applications share the same session information within the portal. The WebLogic Portal TagLib provides a number of helpful utility methods for assisting the portlet in obtaining and sharing session information from the portal Web application perspective.

Be careful in building a Web application not to define a unique security scheme. Rather, the Web application should leverage the existing infrastructure mechanisms as much as possible. For example, most J2EE-based application servers support a number of security mechanisms for propagating user authentication and authorization information to the various components. Typically, portals that are built on top of a J2EE application server will leverage these same mechanisms. For applications that aren’t colocated with the portal framework, it’s often necessary to build bridging interfaces that can map the portal-specific security mechanisms to those of the application being adapted to the portal.

Conclusion

This article discussed an approach for enabling an existing application to be adapted to a Web portal. In doing so, the goal has been to provide the necessary concepts and developer tips to enable a developer to plan for and ultimately perform the portalization. You should note that moving an existing Web application to a portal doesn’t have to take place all at once. It can be a multiphase process. The first phase is to present the existing content or functionality from within the portal. Once this is accomplished, the application will need to be better integrated with the portal so that it can leverage the portal navigation, personalization, and security mechanisms. The final phase is to enhance the personalization capabilities of the application by providing an edit page for allowing users to predefine values, such as their player ID and whether they prefer to have their top scores show.

Listing 1: Traditional form with URL invocation

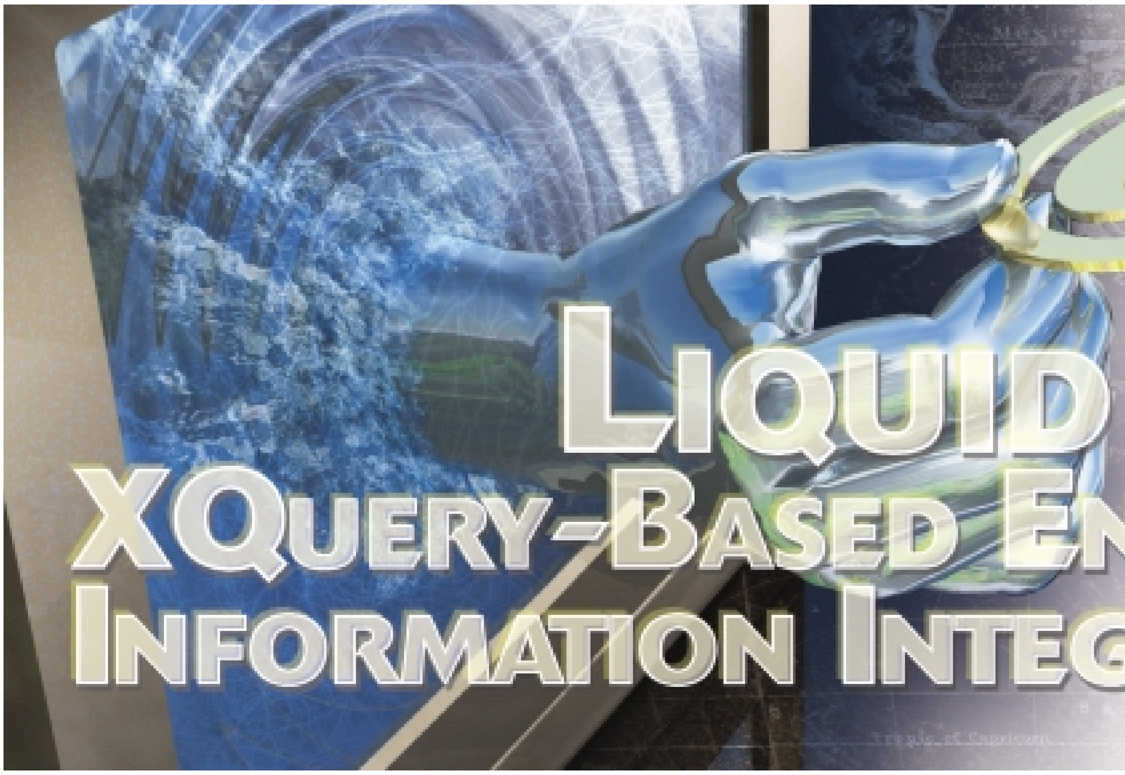
```
<form name="form1" method="post"
   action="QGControllerServlet?action=login">
 <b>PlayerId</b>
 <input type="text" name="playerId">
 <input type="submit" name="Submit" value="Submit">
</form>
```

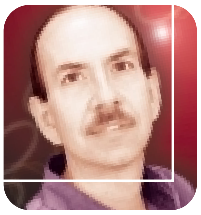
Listing 2: Using a Webflow event with the portlet TagLib

```
<%@ taglib uri="portlet.tld" prefix="portlet" %>
<form name="form1" method="post"
  action="<portlet:createWebflowURL
        event="QGControllerServlet.event"
        doRedirect="true"
        extraParams='action=login'/>">
 <b>PlayerId</b>
 <input type="text" name="playerId">
 <input type="submit" name="Submit" value="Submit">
</form>
```

Precise Software Solutions

www.precise.com/wldj





BY **NITIN MANGTANI & MIKE CAREY**

AUTHOR BIOS...

Nitin Mangtani is the technical program manager for BEA Liquid Data. He was instrumental in defining the overall product vision as well as shipping Liquid Data 1.0. Nitin has worked in both the business application world as well as the application infrastructure world.

Mike Carey worked on data integration, DB2, and XML middleware at IBM from 1995–2000. He joined BEA in 2001 after a stint at an Internet startup. Mike worked on XML data transformations in WebLogic Integration 8.1 and is now architect for Liquid Data. Mike is an ACM Fellow and a member of the National Academy of Engineering.

CONTACT...

nitnm@bea.com
mcarey@bea.com

Modern enterprises are drowning in a sea of information. Despite owning an ever-increasing volume of information, most enterprises cannot exploit it to even a fraction of its full potential.

This is because the information is strewn across many systems with diverse data formats and interfaces – systems that are largely unaware of one another and of the relationships their content has with information contained elsewhere. Therein lies the Enterprise Information Integration (EII) problem: enterprises need the ability to easily access information about a given business entity, such as a customer or an order, from a varied and distributed collection of information sources.

The EII problem isn't new – the database world has been struggling with it for over two decades – but today's business trends bring a new urgency to finding a solution. Luckily, we are also at a point today where an alignment of emerging technologies is finally enabling a solution. This article looks at why this is so and how BEA's Liquid Data for WebLogic exploits this alignment to offer a commercial EII solution today.

An Example Integration Scenario

Picture a large telecommunications vendor with a Wireless Division and a Broadband Division. Each division has its own order management system and its own customer support system. The vendor also has a centralized CRM system that maintains information about customer purchasing histories and product promotions. A closer look at the vendor's IT environment might reveal the following mix of back-end systems:

Wireless Division:
Order Management System: SAP ERP using Oracle as the RDBMS
Customer Support System: Clarify using Oracle as the RDBMS

Broadband Division:
Order Management System: PeopleSoft ERP using Sybase as the RDBMS
Customer Support System: Homegrown system with SQL Server as the RDBMS

Corporate CRM System:
Siebel using DB2 as the RDBMS

As you can see, customer-related information is spread across five different information systems. (In real situations, the overall number of systems in an enterprise's IT environment would likely be closer to the 10–100 range.) These systems use a mix of technologies from different independent software vendors, and the five systems are unaware of one another. Some allow

A FRESH APPROACH TO AN

direct database access, while others allow information access only through their business-level APIs.

The vendor wants to improve its customer responsiveness and operating efficiency by giving customers up-to-the-minute access to all of their account information via several channels – a phone tree system, live telephone customer support representatives, the customer service desk at its sales outlets, and a self-service Web portal. The vendor also hopes to boost sales by adding recommendation features to all four channels. To accomplish this, the vendor wants to create a comprehensive 360° "single view of customer" that can be shared across the channels. To meet the up-to-the-minute goal, the view's data must be obtained directly from the vendor's operational systems. This is a clear instance of the EII problem!

Current EII "Solutions"

To solve this problem, application developers face three basic engineering challenges:

- ***The Metadata Challenge:*** The metadata describing the information contained in different back-end systems is captured in various forms, including RDBMS system catalogs, WSDL files, application views (for J2EE-CA adaptors), and XML Schemas.
- ***The Data Access API Challenge:*** Information contained in different back-end systems must be accessed using the appropriate protocols, such as JDBC/SQL, ODBC/SQL, OCI/SQL (for Oracle data), SOAP, JCA, etc.
- ***The Data Model Challenge:*** The basic format of the data returned from a given back-end system depends on the nature of that system. Possible formats include relational rowsets, Java objects, and XML documents.

Table 1 lists four common approaches to tackling the EII problem today and briefly analyzes their main strengths and weaknesses. None of these approaches offers a full EII solution.

Liquid Data: An XML-Based EII Solution

In November of 2002, BEA introduced Liquid Data for WebLogic. Liquid Data provides real-time access and aggregation of data from disparate data sources, improving visibility for front-office applications. "Real-time" refers here to using the most up-to-date information available. Unlike a data warehouse, which houses a preintegrated copy of selected enterprise data that is days or weeks old, Liquid Data provides access to the current state of the enterprise by directly integrating and providing access to the data in the enterprise's various operational systems. Liquid Data's focus is on data visibility, or integrated read access. BEA's WebLogic Integration complements Liquid Data for applications that require update support. It targets front-office applications, which typically require fast access to an integrated view of one or a few business entities at a time. Our 360° view-of-customer scenario is a classic front-office application.

Liquid Data distinguishes itself from past EII approaches by taking an XML data integration approach. It combines many of the strengths of the EAI and relational data integration approaches without inheriting their weaknesses. Liquid Data provides an XML standards–based view of all enterprise data – enterprise data sources appear either as virtual XML documents or as a collection of functions that take XML parameters and produce XML results. The W3C XML Schema standard is the enterprise data model, and the W3C XML Query (XQuery) draft standard is the declarative language for specifying integrated reusable views and queries over data distributed across the enterprise. XQuery functions model the functional data sources. EAI-style adaptors are used to "XML-ify" packaged applications that don't natively expose XML APIs. Using XML rather than flat tables makes Liquid Data well suited to handling information from complex data sources, and using XML rather than a proprietary EAI object model positions Liquid Data to interact with applications that support standards-based XML and Web service APIs.

The Liquid Data Architecture

Figure 1 depicts the architecture of Liquid Data. At the bottom are the various flavors of data sources supported out of the box. Liquid Data natively supports JDBC/SQL sources (Oracle, DB2, Sybase, SQL Server, and PointBase), Web services (both simple RPC-style and complex document-style), XML data files, and application views of packaged or legacy applications (SAP, PeopleSoft, etc.) using BEA

TABLE 1

APPROACH	STRENGTHS	SHORTCOMINGS
Custom Coding	• Definitely serves the initial purpose	• Very high cost of creation and ongoing maintenance • Low reusability of code across departments
ETL and DataWarehousing	• Supports heavy analytics over very large historical data sets	• High maintenance cost • Data far from "real time" • Queries limited to data pre-selected for archiving • Project cycles usually long
Enterprise Application Integration (EAI)	• Adaptor architecture for back-end applications • Supports process-oriented integration (i.e., application orchestration)	• Developer must hand-code "query plans" to solve each instance of the EII problem • Doesn't support creation of queryable views
Relational Data Integration	• Supports SQL queries on multiple RDBMS tables • Supports queryable views	• Doesn't naturally integrate non-relational data, such as data from Web services and packaged applications • Possibly biased towards solution vendor's RDBMS

Potential EII "solutions" today

OLD PROBLEM

WebLogic Integration adaptors. For JDBC/SQL sources, Liquid Data automatically introspects their relational catalogs to produce default XML views of their content. In the case of Web services, it automatically introspects their WSDL descriptions. In addition to these sources, Liquid Data supports custom Java functions to enable developers to write XML plug-ins for data sources that are not natively supported by Liquid Data. Data views (discussed later), once defined, can also be used as Liquid Data data sources.

The top of Figure 1 shows applications making data query calls involving data views. The Liquid Data world has the following key players and corresponding responsibilities:

- ***System administrator:*** Configures the data sources for the Liquid Data Server.
- ***Data architect:*** Subject matter expert who understands the relevant business entities and data sources. Defines one or more layers of reusable integrated views (data views) of the business entities of interest. Defines stored queries on top of the data views.
- ***Application developer:*** Calls stored queries, which are usually parameterized, from within application code.

For the system administrator, Liquid Data extends BEA WebLogic Server's Web-based console. For the data architect, Liquid Data provides a tool, DataView Builder, for use in graphically defining data views and stored queries. Liquid Data focuses on parameterized stored queries because they provide a way to simplify application developers' lives while simultaneously protecting the enterprise's operational data stores from ad hoc query traffic. For the application developer, Liquid Data offers an EJB API, a JSP tag library API, and a Web services API for invoking stored queries.

At the center of Figure 1 is a distributed query execution engine. This engine processes XML queries by calling the underlying data sources, doing the required data transformations and computations, and returning integrated XML results to the application. Caching is also supported. The data architect can indicate that a given query's results may be cached for reuse when the same query or view is reaccessed (with identical parameters) within a specified time period. Not shown in Figure 1, but important, is that Liquid Data supports security policies on stored queries, data views, and base data sources. The Liquid Data server has been implemented as a J2EE application that can be deployed on a BEA WebLogic Server cluster.

XQuery: Liquid Data's Secret Sauce

Liquid Data uses XQuery, which is rapidly gaining broad industry support as the declarative language for defining integrated views of an enterprise's many data sources. We give a brief example here to illustrate how an XML query can stitch together data from multiple sources. The XQuery language is detailed at www.w3.org/TR/xquery; Liquid Data's use of XQuery is covered in http://edocs.bea.com/liquiddata/docs10/index.html. The most heavily used XQuery expression in Liquid Data is the FLWR expression, which is analogous to SELECT-FROM-WHERE queries in SQL. A FLWR expression has the following parts:

- ***A FOR clause*** that generates one or more value sequences, binding the values to query variables. The FOR clause in XQuery is similar to the FROM clause in SQL.
- ***A LET clause*** that binds a temporary variable to the result of an expression. The LET clause is similar to the provision of temporary views in some vendors' dialects of SQL.
- ***A WHERE clause*** that contains Boolean predicates that act to restrict the FOR clause's variable bindings. The WHERE clause in XQuery is directly analogous to the WHERE clause in SQL.
- ***A RETURN clause*** that specifies the query's desired XML output. The XQuery RETURN clause is roughly analogous to the SELECT clause in SQL, though the structures that it can specify are a good deal richer than those of SQL.

Consider our initial EII scenario. Suppose we want to get contact information for a given Wireless Division customer together with information about their orders for Broadband Division products. Listing 1 shows how XQuery can do this. The topmost FOR clause binds a variable to each wireless customer. The LET clause that follows concatenates the customer's first and last names for output formatting. The WHERE clause filters customers by only keeping those whose ID matches a query parameter. Finally, the large RETURN clause specifies the query's output, which consists of customer data from the Wireless Division's database plus the result of a nested query. The nested query pulls order data from the Broadband Division database; its WHERE clause ensures that only the orders for the customer of interest are queried, and its RETURN clause retrieves the desired order information. Listing 2 shows a sample of this query's output.

DataView Builder UI

To minimize the need for the data architect to write XQuery views and stored queries by hand, Liquid Data includes a tool called DataView Builder. DataView Builder supports a pictorial, drag-and-drop, mapping-based approach to query design and construction for XQuery. Figure 2 shows an example of the design view, where input data sources are mapped to target data views to graphically construct an XQuery query. DataView Builder also provides a test-view window where the query can be inspected, run, and optionally hand-tuned.

Programmatic APIs

Liquid Data provides three APIs that application developers can use to call Liquid Data queries from within application programs. They are:

- ***EJB API:*** This provides a Stateless Session Bean (SLSB) interface to the Liquid Data server. It is a JDBC-like interface that has methods to execute either stored queries (the norm) or ad hoc queries, with or without query parameters, against the integrated enterprise data known to the Liquid Data server.
- ***JSP Tag Library:*** For JSP and portal developers, Liquid Data provides a JSP tag library facade over its SLSB interface. Each SLSB method has a counterpart in the tag library.
- ***Web Service Interface:*** Developers can ask Liquid Data to convert stored queries into Web services. A corresponding WSDL file is produced for each such stored query. Any application that talks to Web services can utilize Liquid Data services through this API.

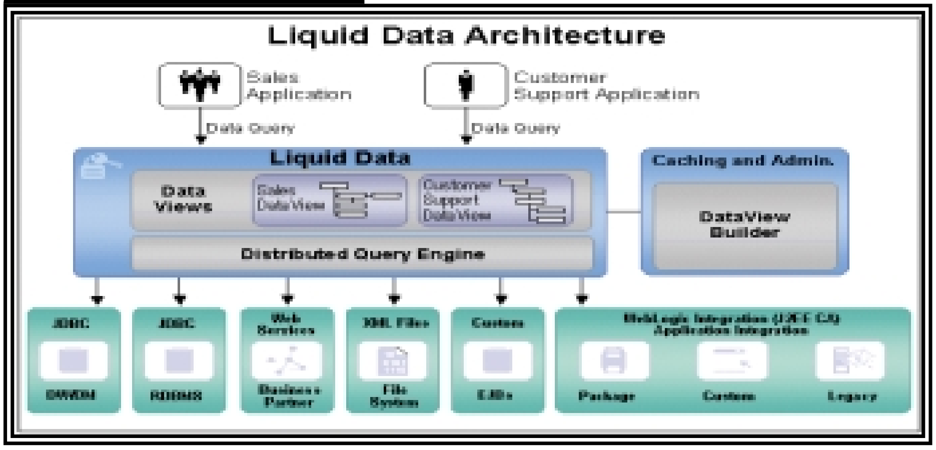
Other Features

The features of LiquidData are too numerous to be covered here. For more detailed information, we encourage you to browse the online documentation at http://edocs.bea.com/liquiddata/docs10/index.html and download and try the product itself, but we will highlight a few of the key additional features here.

BEA Systems

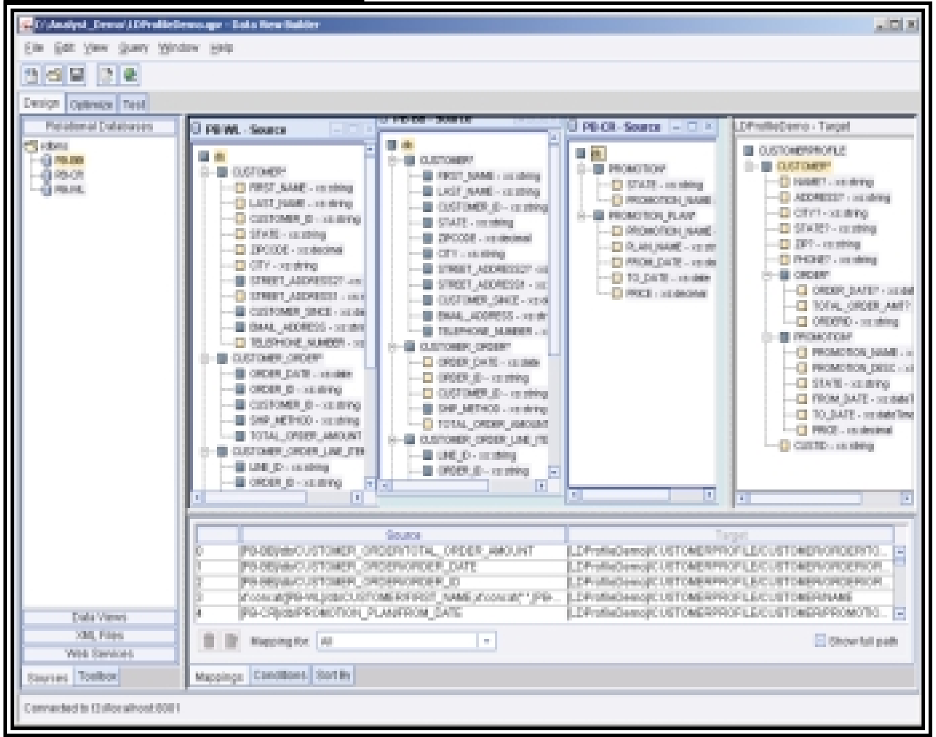
http://dev2dev.bea.com/useworkshop

FIGURE 1



Liquid Data architecture

FIGURE 2



DataView Builder – Design View Window

Administration Console

For system management, Liquid Data extends WebLogic Server's existing administration console. New console-supported activities include:

- Managing basic Liquid Data server settings
- Configuring new Liquid Data data sources
- Configuring the Liquid Data caching subsystem
- Defining Liquid Data security policies for data sources and data views
- Monitoring the activity of the Liquid Data server

Security

Liquid Data includes support for security policies and supports access control lists (ACLs) for each stored query. Liquid Data's ideas of users and groups are identical to those of BEA WebLogic Server. For ad hoc queries, Liquid Data supports ACLs at the data source level as well.

Repository

To manage EII assets, Liquid Data uses a lightweight, server-based repository to store source and target XML schemas, stored queries, WSDL files, code and metadata for custom functions, and so on. The repository is file-based and provides a simple notion of folders.

Product Combinations

Liquid Data is an integral part of the overall BEA Platform family. It can be used in combination with any of the existing BEA products. Interesting BEA product combinations include:

- ***Liquid Data plus WebLogic Portal:*** Portal handles Web presentation and personalization while Liquid Data handles enterprise data integration and aggregation.
- ***Liquid Data plus WebLogic Integration Adaptors:*** This enables Liquid Data to integrate data from packaged applications together with relational and Web service data.
- ***Liquid Data plus WebLogic Integration:*** Liquid Data gives workflows an integrated view of enterprise data while the workflows support acting on (e.g., updating) the data.
- ***Liquid Data plus WebLogic Workshop:*** Liquid Data provides an integrated view of enterprise data to Web service–based applications.

Summary

This article has presented a brief technical overview of BEA Liquid Data for WebLogic, which takes a fresh, XML-oriented approach to attacking the enterprise information integration problem. It provides a virtual XML document–based view of the enterprise's disparate data sources, and a declarative XQuery-based paradigm is used to create integrated views of the enterprise's key business entities and to define stored queries over those views.

References

- Landers, T., and Rosenberg, R., "An Overview of Multibase," Proceedings of the 2nd International Symposium on Distributed Data Bases, Berlin, Germany. North-Holland Publishing Co., September 1982.
- "BEA Liquid Data for Weblogic: Increasing Visibility in the Distributed Enterprise," BEA Systems White Paper, October 2002.
- *XQuery 1.0: An XML Query Language.* www.w3.org/TR/xquery/. W3C Working Draft, August 2002.
- Curbera, F., et al. "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," IEEE Internet Computing 6(2), March-April 2002.

- *Liquid Data for WebLogic Documentation:* http://edocs.bea.com/liquiddata/docs10/index.html
- *Liquid Data for WebLogic Download:* http://commerce.bea.com/downloads/liquid_data.jsp

Listing 1: XQuery Example

```
<CUSTOMERPROFILE>
{
    for $c1 in document("WIRELESS-DIVISION")/db/CUSTOMER
    let $fn := xf:concat($c1/FIRST_NAME," ",$c1/LAST_NAME)
    where $c1/CUSTOMER_ID eq #$CUST_ID_PARAM
    return
        <CUSTOMER>
                <CUSTID>{ xf:data($c1/CUSTOMER_ID) }</CUSTID>
                <FULL_NAME>{ $fn }</FULL_NAME>
  <ADDRESS>{ xf:data($c1/STREET_ADDRESS1) }</ADDRESS>
  <CITY>{ xf:data($c1/CITY) }</CITY>
  <STATE>{ xf:data($c1/STATE) }</STATE>
  <ZIP>{ xf:data($c1/ZIPCODE) }</ZIP>
  <PHONE>{ xf:data($c1/TELEPHONE_NUMBER) }</PHONE>
  {
      for $o1 in document("BROADBAND-DIVISION")/db/CUSTOMER_ORDER
      where ($c1/CUSTOMER_ID  eq  $o1/CUSTOMER_ID)
             return
      <ORDER>
         <ORDER_DATE>{ xf:data($o1/ORDER_DATE) }</ORDER_DATE>
                  <TOTAL_ORDER_AMT>{ xf:data($o1/TOTAL_ORDER_AMOUNT)
}</TOTAL_ORDER_AMT>
                                <ORDERID>{ xf:data($o1/ORDER_ID)
}</ORDERID>
                                <SHIP_METHOD>{
xf:data($o1/SHIP_METHOD) }</SHIP_METHOD>
                        </ORDER>
                }
        </CUSTOMER>
}
</CUSTOMERPROFILE>
```

Listing 2: Sample XQuery Output

```
<CUSTOMERPROFILE>
      <CUSTOMER>
            <CUSTID>CUSTOMER_1</CUSTID>
            <FULL_NAME>JOHN_1 KAY_1</FULL_NAME>
            <ADDRESS>NORTH FIRST STREET</ADDRESS>
            <CITY>SAN JOSE</CITY>
            <STATE>TX</STATE>
            <ZIP>93151</ZIP>
            <PHONE>4081231234</PHONE>
            <ORDER>
                  <ORDER_DATE>2002-04-09</ORDER_DATE>
                  <TOTAL_ORDER_AMT>1000.00</TOTAL_ORDER_AMT>
                  <ORDERID>ORDER_ID_1_0</ORDERID>
                  <SHIP_METHOD>AIR</SHIP_METHOD>
          </ORDER>
          <ORDER>
                  <ORDER_DATE>2002-04-09</ORDER_DATE>
                  <TOTAL_ORDER_AMT>1500.00</TOTAL_ORDER_AMT>
                  <ORDERID>ORDER_ID_1_1</ORDERID>
                  <SHIP_METHOD>AIR</SHIP_METHOD>
          </ORDER>
     </CUSTOMER>
</CUSTOMERPROFILE>
```

Consistent Content Management and Delivery

TIMELINESS AND ACCURACY VIA DOCUMENTUM AND BEA WEBLOGIC PORTAL

BY **TRAVIS WISSINK**

We know the mantra "Content is King" with the Internet is legitimate. Many now ask, "How do we make content a focus?" Whether you have a content-heavy Internet Web site or a vast base of enterprise content inside your corporate intranet, you want to provide your users with a quick and efficient way to access and produce relevant content.

Recently an architecture that assisted in easy content creation and consumption included a portal server and a content management system (CMS). These two products have many services that apply business rules to the creation and consumption process of enterprise content – hence, WebLogic Portal and Documentum 4i.

BEA WebLogic Portal 4.0 and Documentum 4i are two application platforms with many architecture integration issues and concerns. In this article, I'll discuss two architectures that describe how to secure, organize, manage, and deliver Web content. The difference is one installation is focused on intranet consumption, the other on Internet consumption.

AUTHOR BIO

Travis Wissink, an independent consultant, calls the Washington, DC Metro area his home. He specializes in WebLogic (J2EE) development and content management needs. Most recently, Travis is a lead WebLogic Portal consultant and is integrating BEA WebLogic Portal with Documentum.

CONTACT...

travis@wissinks.com

Documentum 4i

Documentum 4i is an Enterprise Content Management Solution. We are primarily concerned with how it handles Web content (XML and HTML). Documentum 4i's content repository is called the DocBase. Documentum has many add-on products that interact with the DocBase, including WebPublisher, WebCache, and eConnector,

WebPublisher is the Web front end and allows administrators, publishers, editors, and authors to create, edit, and manage the enterprise content. One of WebPublisher's best features is its Web-based XML authoring tool. This allows the content publishing team to fully manage XML content without knowing anything about XML and its specification. To create different outputs from the XML, WebPublisher allows "renditions" to be created from the XML. These rendition templates are standard XSLs.

WebCache is basically the deployment engine. It has an internal scheduler that runs deployment jobs that are configured to implement who, what, when, where, and why to deploy content in the Documentum DocBase. WebCache has a sender "WebCache Source" and a receiver "WebCache Target." The Source needs to interact with the DocBase to retrieve the content. The Target needs to be wherever your content delivery solution is.

Finally, we have the eConnector for BEA. The eConnector is Documentum's implementation of the WebLogic Content Manager. What is the Content Manager? This is the edocs.bea.com description:

The Content Manager runtime subsystem provides access to content through tags and EJBs. The Content Management tags allow a JSP developer to receive an enumeration of Content objects by querying the content database directly using a search expression syntax. The Content Manager component works alongside the other components to deliver personalized content, but does not have a GUI-based tool for edit-time customization.

WebLogic Portal

I won't go into all aspects of BEA WebLogic Portal, but will discuss some of what it can do for our Web content. I'll look at pieces of WebLogic Portal like the content manager, EJBs, content selectors, and the PZ tag library. At a high level, Portal's personalization server provides a framework so that developers can deliver content in a secure, organized, and dynamic way.

Similar Architectures; Different Requirements

The Internet Architecture

The Documentum architecture is typical for an Internet Web site installation (see Figure 1). I call it typical because Documentum, and the content, rests securely inside the corporate intranet and your content is behind many stringent firewall rules. In the corporate DMZ, WebLogic Portal waits to deliver the content to our Internet customers. WebCache Source is installed on the same server as the main Documentum product, inside the intranet. WebCache Target is installed on the same server as WebLogic Portal, in the DMZ. These products are configured to securely deliver "active" content from the Source through the firewalls to the Target. The eConnector product is installed using the WebCache implementation method.

The Intranet Architecture

In a corporate intranet portal (see Figure 2) we're not dreadfully concerned with the firewall and the security from the Internet to our CMS and our delivery solutions, as compared to our Internet architecture. We aren't concerned with it because all the installations are safely inside the corporate firewall. Documentum and WebLogic Portal will be installed so that they don't have any firewalls between them. Therefore, there's no need to have WebCache involved with this installation. You may ask, how will WebLogic Portal deliver the content from Documentum? The eConnector can be configured to interact directly with the DocBase. This is a lot easier because you don't have firewall or port restrictions, along with not having the WebCache set of products to install and configure.

Setting Up Documentum to Work with eConnector

Documentum is a large application with many different ways to manage structured and unstructured enterprise content. As a side note, I strongly suggest finding a senior-level Documentum consultant when you initially implement this product. As you dig into the guts of the product, and every installation will, there is a large learning curve. I'll point out several pieces of the product that directly impact the integration of these robust applications.

Creating the Content

Every piece of content that goes into the CMS needs to have an associated document type. A document type is a DocBase object that defines what a piece of content is to the CMS. We are mainly concerned with the document-type properties, that hold the content classification data. For us, the classification helps us cooperate with our Web site's taxonomy. The properties also assist with a more precise search for content inside the CMS and in our Web site.

After you set up the document type, you'll need to set up a content template in WebPublisher. You have to establish three different objects and set some configuration information. First, you need to assign a document type to this content type. The next property is a default workflow. Next, you'll need a content template – a blank XML document that contains all available XML tags. We all know that this should be a DTD but that's not what Documentum implemented. To complement the content template we need to define a rules file. The rules file is an XML document that maps content template tags to Web form controls. Finally, we have the renditions, also called output presentations. Presentations are XSLs that format the XML content to the various outputs you may need. For our purposes, we'll need an HTML presentation.

Now Documentum is ready to have an author create some content. I must point out that the XML authoring tool does have an embedded HTML authoring tool for the text body form format, but it doesn't perform much more than RTF capabilities (see Listing 1; the code for this article can be found online at www.sys-con.com/weblogic/sourcec.cfm).

Managing the Content

After the author creates and saves the content, the default workflow is initiated (see Figure 3). We will use a workflow that automatically promotes the content to an "active" status. This type of workflow is strongly discouraged. Usually an editor will need to view/edit and approve the content before it moves to active status. *Active content* is a Documentum term to describe a piece of content that has been fully approved through a workflow approval cycle. At this point our piece of content is ready for deployment to the Web site. Although I may not explain it in depth, there are many other benefits to using a CMS like Documentum. Features of CMS include, but are not limited to, user/content security, aggregation services, workflow, versioning, outputting/exporting services, and workgroup collaboration services.

Deploying the Content

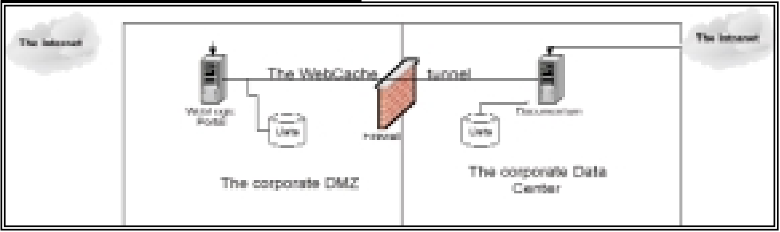
In comes WebCache source. We need to set up a Web Publishing Configuration to deploy our active content. We can set this up in the Documentum DocBase Administrator. Some of what we need to know is the IP address, database table, and target directory of the WebCache Target, so the Source can communicate with it. The most important part of setting up a Web Publishing Configuration is the additional attributes section in the setup. These additional attributes directly relate to the document-type properties. The Web Publisher sends the additional attributes to the database table, which will go into the eConnector setup. Finally, we'll need to set up a Documentum Job. The Documentum Job is the internal scheduler, like a cron tab. We want a Job to run the Web Publishing Configuration. It will need to run every *x* hours or *x* minutes. These WebCache Jobs run transactionally, meaning they only send content that has changed since the last time it ran, so it conserves network and processing resources.

Initial BEA WebLogic Setup Including the Documentum Client

One configuration in common with both architectures is that BEA WebLogic needs to be configured to communicate with the Documentum client. The Documentum client is a set of Java libraries, OS libraries, and some property files. The important Java API between WebLogic and the Documentum client is the dfc.jar file, the Documentum Foundation Client. This set of Java classes communicates to the Documentum client, which is only available on a few OSs. (Consult Documentum for their support requirements; I have used it on Solaris.)

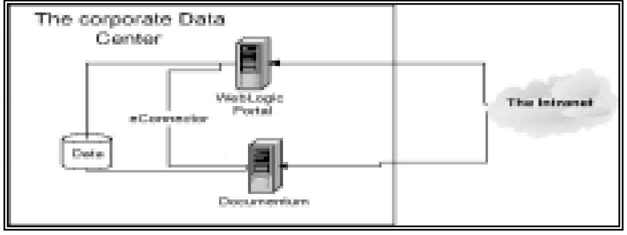
First, install eConnector. Next, add all necessary Documentum client files to the WebLogic paths. Listing 2 establishes the Documentum eConnector client home. Then

FIGURE 1



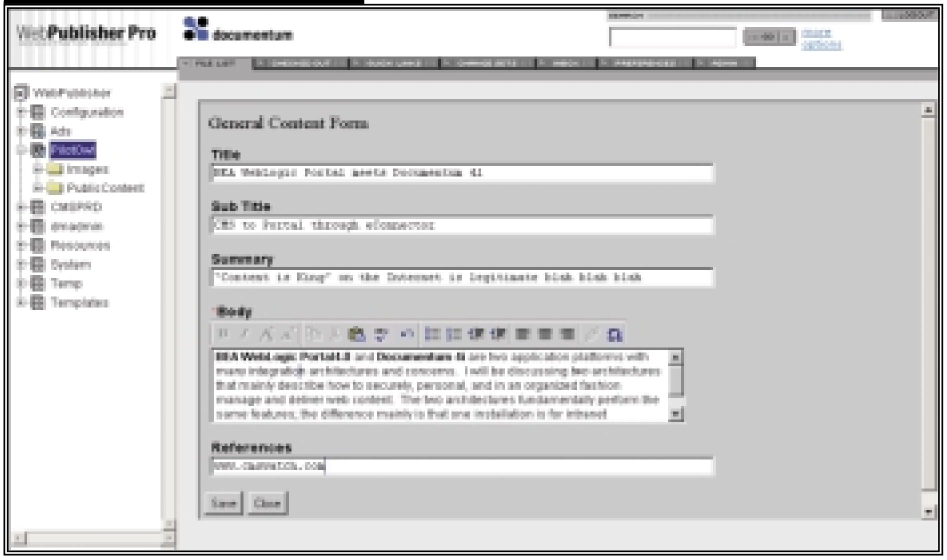
The Internet architecture

FIGURE 2



The intranet architecture

FIGURE 3



General Content Form

Title
BEA WebLogic Portal meets Documentum 4i

Sub Title
CMS to Portal through eConnector

Summary
"Content is King" on the Internet is legitimate blah blah blah

Body
BEA WebLogic Portal 4.0 and **Documentum 4i** are two application platforms with many integration architectures and concerns. I will be discussing two architectures that mainly describe how to securely, personal, and in an organized fashion manage and deliver web content. The two architectures fundamentally perform the same features, the difference mainly is that one installation is for intranet

References
www.cammatch.com

Creating content

put two JAR files, the dmjdbc.jar and the all-important dfc.jar, into the db_classpath along with the Oracle JDBC drivers. The db_classpath eventually goes into the classpath. Next we need a pointer to the Documentum client property file, dmcl.ini, in our shell environment. This file contains various properties to connect the Documentum client to the Documentum DocBase. Finally, we need the actual Documentum client in our LD_LIBRARY_PATH. The needed library file is libdmcl40.so, the OS-dependent file.

Configure Portal Application to Use eConnector

Now that we have our WebLogic environment set up with the Documentum client, we need to set up our portal application in order to use the eConnector. First, we install and deploy the appropriate eConnector EJB based on the chosen architecture; we're using the WebCache configuration and the Webcache-ejb.jar. This is the WebCache implementation of the WebLogic Portal Content Manager. Next we need to configure a JDBC connection pool (see Listing 3). Along with the connection pool in the WebLogic application the eConnector has a property file to contain more specific database information. Now, edit the dmjdbc.-property file. You'll need to put the actual database driver and connection information into the WebCache property file. We'll also need to edit the WebLogic Portal property EJB to look to our content manager EJB. Now, we need to change the following EJB descriptors: EJB Ref Name field to ejb/ContentManagers/-WebCacheDocument; EJB Reference Description to ejb/ContentManagers/WebCacheDocument; and the JNDI name field to ${APPNAME}.BEA_personalization.WebCache DocumentManager. Make sure that the new EJB and new connection pool is targeted and deployed to your server.

Finally, and most important, we need to map WebCache Source attributes to something that our JSP developers can write queries against. eConnector has a property file named {$DCTM_HOME}Webcache map.property. This file lets you map a comprehensible name to a WebCache attribute and is important because the WebCache attribute is our content metadata. Note that if there isn't a mapping setup, then the JSP developers will not be able to query against that attribute.

What Do We Do with the eConnector?

Now that our environment and portal application are set up to use the Documentum eConnector for BEA, we need to display the content. To pull the content from our new Content Manager, we use a function from the Personalization tag library. The pz:contentQuery function allows us to query the WebCache Content Manager for our specific content (see Listing 4).

Metadata

The metadata is the "glue" that binds the content between these two products and our Web users. The content's metadata is information (data) that classifies the content so that in a dynamic Web site the content can integrate with the Web site's taxonomy. A lot of requirements gathering and analysis is needed to carefully negotiate the metatagging scheme. Metadata could make or break your long-term content goals. You will need to carefully balance author dissatisfaction with lengthy entry forms with content rich classifications. Angry content authors could prove a good system stale. Alternatively, if you don't enrich your content with metadata, then your content may not be agile enough to repurpose to other Web services.

Clustering

Documentum has a form of clustering called *federating* – a way to allow multiple Documentum installations to interact with each other's content. WebLogic has many clustering features that other ***WLDJ*** articles have discussed. What I want to mention is different options in the Internet installation. The WebCache Target needs to be installed on each server that has WebLogic Portal running on it. This is needed because of the eConnector. You really can't use a shared drive to host the content between the different servers. In addition, you need the WebCache Source job to deploy content to all of the installed WebCache Targets.

Once again, the Documentum suite of products is very large. Bringing in one individual who has a wealth of Documentum experience will help to solidify a successful Documentum implementation.

Content Aggregation vs Content Presentation

Keep content templates separate from content presentation, that is, keep content style away from the authoring process. This is an essential but sometimes difficult piece of advice for any CMS project at a time when editors tend to want very specific WYSIWYG HTML authoring tools. If you allow the editors to mess around with the content entry system with presentation logic via HTML tags, then when you want to change the presentation, your content will also have to change and that will take many hours of content editing. Put time and effort into developing granular data capture templates instead, and take care of end-user presentation within your JSP pages on the delivery side.

Summary

The benefits of matching a CMS with a personalization server are endless. A CMS like Documentum can help manage, centralize, and organize your corporate content assets. On the delivery side, WebLogic Portal offers a strong platform that provides a reliable, scalable, and secure way of presenting your enterprise content assets to your Web-based audience.

Mercury Interactive

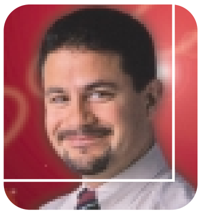
www.mercuryinteractive.com/bto

Adaptive computing, self-healing systems, Grid and on-demand computing, autonomic computing.... Vendors from all sides are throwing buzzwords around, a new one every day or so it seems.

Simplifying Infrastructure Software

FROM GRID TO ADAPTIVE COMPUTING

BY BENJAMIN RENAUD



AUTHOR BIO

Benjamin Renaud is a strategist in the Office of the CTO at BEA Systems. In that role he helps set BEA's technical vision and guides its execution. He came to BEA via the acquisition of WebLogic, where he was a pioneer in Java and Web application server technology. Prior to joining WebLogic, Benjamin worked on the original Java team for Sun Microsystems, where he helped create Java 1.0, 1.1, and 1.2.

CONTACT...

br@bea.com

This month we'll try to make sense of it all by looking at what is here today, what will be here tomorrow, and what is mere science fiction. More important, we'll examine how these new ideas impact your ability to develop and deploy applications.

In tough economic times, IT managers are asked to do more with less. That means driving down the cost of everything from laptops to enterprise-wide applications. The cost of deploying an application lies primarily in two areas: hardware and people. Standards-based software platforms such as BEA WebLogic have been instrumental in reducing cost and time-to-market of applications by providing a reliable base and a set of built-in services.

Increased Complexity

Standard platforms also allow a much higher degree of application interoperability by allowing applications to talk to each other using standard protocols like RMI, JCA, or SOAP (Web services). That move from silo architectures to interconnected, high-level architectures has increased the complexity of applications considerably. From client/server we've gone to Web client-Web server-business tier-database, with the business tier potentially connecting to many other applications, including legacy systems and Web services across the Internet. We have Web site objects, workflow objects, database connections, and JCA connections. The platform has kept up with this increased complexity, but only barely. To take applications to the next level a new set of technologies that radically simplify the development, deployment, and maintenance of applications will be required. We group these technologies under the label "Adaptive Computing" because in that model the infrastructure adapts itself to the application. It optimizes, provisions, and heals itself without the intervention of developer or administrator.

From Grid to Adaptive Computing

Grid computing is among the first ideas for the sharing and optimizing of computing resources, and comes straight from the halls of academia. The idea behind Grid is to take some large chunk of work, say the mapping of the human genome, and break it up into many small chunks spread over many computers. The Search for Extraterrestrial Intelligence at Home (SETI@home) project at Berkeley is an example of this. Its goal is to analyze radio telescope data to detect specific patterns indicative of extraterrestrial intelligence. The amount of data collected is so enormous that no single computer could possibly do it by itself, so researchers devised a scheme where anyone with an Internet connection and a PC could participate by running a screen-saver program capable of analyzing a small chunk of data. When your computer is idle, the program uses the idle CPU cycles for the project. To date, the SETI@Home project has had 4,257,524 users who contributed a total of 1,336,810.852 years of computer time. So far no one has found an extraterrestrial. (*Note:* This may or may not be entirely true. At least one signal matching the target profile was recorded by the Big Ear radio telescope at Ohio State University on the night of August 15, 1977. It was never detected again.) Since then many other projects have emulated SETI to solve hard scientific problems, from breaking cryptographic keys to finding a smallpox vaccine.

More recently, the Globus project has developed the Globus Toolkit, an open-source implementation of a Grid infrastructure, written in C. The toolkit is a "bag of services" that can be used to develop Grid applications and programming tools. While some companies are talking about using Globus in an enterprise setting, Globus is really designed for the scientific and engineering problems we just described rather than the problems found in corporate IT.

Programmer's Paradise

www.programmersparadise.com/BEA

Think Globally, Act Locally

Like many academic ideas, Grid needs to be refined before it can be used in the real world. The vast majority of businesses, enterprises, and government organizations don't want to spread their data or their applications all over the Internet, or even across computers they don't control completely. While interacting with computers and services on a different network or across the Internet is common practice, sending one's applications and data is not. IT professionals want to maintain administrative control over their IT infrastructure. IT departments want to make the most efficient use of their hardware and don't want idle CPUs. The solution is to evolve and broaden the Grid to the more powerful concept of Adaptive Computing. Adaptive Computing is an umbrella term for a far more intelligent application infrastructure. Such an infrastructure makes better use of resources through dynamic provisioning, self-healing, and self-tuning.

Better Provisioning

IT departments must often allocate enough machines to handle peak demand for a particular application, leaving most of their boxes idle most of the time. Traffic at e-commerce sites such as Amazon.com or FedEx may be highest in the weeks leading up to Christmas, but lowest after New Year's. A CRM application may peak during the day when customers call in while the inventory application could make use of the same hardware at night, when no one is calling in. Upcoming versions of application infrastructure will let applications share hardware and other resources effectively to minimize duplication and hardware costs.

While saving on hardware costs can generate large savings, development and maintenance costs dominate the cost of deploying an enterprise application. Companies like Microsoft and BEA have focused on reducing the cost of development with tools like BEA WebLogic Workshop and Visual Studio .NET. The cost of testing, optimization, management, and administration, however, is still too high. This is where so-called "self-tuning" and "self-healing" applications can save an enterprise a lot of money.

Imagine if you will a system that notices that its process performance is slowly degrading over time. After running a diagnostic procedure, it concludes that one of the applications running in the JVM is leaking memory. At that point it will notify an operator and take action by itself: it may quiesce the application in the question process (i.e., instruct the application not to take any new request, complete all outstanding requests, and shut down) and leave the other applications alone and the process running. It may quiesce all applications and either 1) restart the process minus the offending application or 2) restart the process with a bigger heap, until the application is fixed. The BEA WebLogic Platform provides robust self-healing features, including fail-over and automatic connection pool resizing, but this is just the beginning and you'll be seeing much more coming in that area.

"While self-healing is the ability to deal with exceptional conditions gracefully, self-tuning is about improving the application's performance under normal conditions"

While self-healing is the ability to deal with exceptional conditions gracefully, self-tuning is about improving the application's performance under normal conditions. In other words, self-tuning is the ability for the platform to optimize itself for a particular application. Platforms such as BEA WebLogic have hundreds if not thousands of configurations and tuning knobs. Today a typical application is tuned in a testing lab by a developer with a load simulator in one hand and a tuning guide in the other. A developer or an administrator can adjust many parameters, including memory heap size, the number of execution threads, the number of IO threads, the size of EJB caches, or the size of a JMS queue. The idea behind self-tuning is to let the infrastructure monitor the application, gather and analyze the data, and based on that data optimize the application automatically. This has the twin benefits of making the infrastructure easier to use and of improving application performance. As for self-healing, the BEA WebLogic Platform has been leading the pack with self-tuning features. Its J2EE JDBC drivers, the software that lets Java applications connect to databases, have long been self-tuning. There again there is much more we can do.

Easier Deployment

The idea behind dynamic provisioning, or the sharing of hardware resources, is to treat a large pool of computers (a distributed system) as if it were just one computer, much like a mainframe. We call this the "Virtualized Mainframe." BEA WebLogic has pioneered the most advanced and robust implementation of the key concepts needed to do this, such as clustering, load-balancing, and fail-over. There is much left to do however, and you should look for some exciting improvements in the coming years. These include distributed application deployment so that deployment, undeployment, and quiescing of applications across a domain becomes seamless. Application containment is included so that a specific application can be granted a specific amount of resources, but not more. This is key to ensuring that no one application can take down an entire IT domain, either by mistake through a programming error, or by design through a virus or a Trojan horse.

Reducing Complexity

All of these features have one aspect in common: automation. Automation, or letting the infrastructure do more and the administrator do less, is the only viable way to reduce complexity. Managing, optimizing, understanding, and debugging the applications of the future will only be possible through radical simplification. This is what Adaptive Computing is about.

Documentum

www.documentum.com/develop

Are you tired of going through the cumbersome process of creating local/remote component and home interfaces for your EJBs, as well as the necessary WebLogic XML deployment descriptors?

Simplifying EJB Development with XDoclet and BEA WebLogic

AN EASY WAY TO CONFIGURE YOUR BEANS

BY RYAN LECOMPTE

AUTHOR BIO

Ryan LeCompte is a research assistant with the ACIM Center of the University of Louisiana at Lafayette. He has extensive research and development experience in designing and implementing enterprise solutions based on EJBs and BEA WebLogic application server technologies. Ryan has developed various J2EE applications that have been deployed by the State of Louisiana, the university, and the private sector.

CONTACT...

ryanlecompte@louisiana.edu

Wouldn't it be wonderful to develop only the particular EJB bean file and have another tool generate all of the necessary interfaces and WebLogic deployment descriptors? Look no further: XDoclet to the rescue! XDoclet is an open-source tool that uses attribute-oriented programming concepts to automatically generate various source code files based on embedded XDoclet-specific Javadoc comments. XDoclet is used in conjunction with the Ant build tool to process the various XDoclet comments in the developer's main EJB file. In this article I'll demonstrate how to use XDoclet with BEA WebLogic server to handle the creation of CMP Entity beans and container-managed relationships among CMP entity beans, as well as how to use the various WebLogic-specific XDoclet tags. (*Note:* The source code was generated using XDoclet 1.2 Beta and Ant 1.5. The source code for this article can be found at www.sys-con.com/weblogic/sourcec.cfm).)

XDoclet Basics

So how does a developer actually use XDoclet? I've provided a simple business domain model that is implemented as three CMP entity beans with the following relationships:

- Doctor Bean [M <--> N] Patient Bean (Bidirectional)
- Patient Bean [1 <--> M] Illness Bean (Unidirectional)
- Doctor Bean [1 <--> 1] Illness Bean (Bidirectional)

To simplify matters we assume that:

1. A doctor can have multiple patients.
2. A patient can be seen by multiple doctors.
3. A patient can have more than one illness.
4. Only one patient can have a certain illness.
5. Only one doctor can cure any one illness.

XDoclet is comprised of two types of tags: class-level and method-level tags. Class-level tags are used to set properties of the EJB that characterize a main function, such as the actual bean type and the JNDI name that will be used to locate the EJB. Other options specify the name of the EJB, finder methods, transaction settings, as well as other WebLogic-specific settings. The following is an example from the Doctor bean:

```
/**
  * CMP Bean utilizing XDoclet tags to represent a
  * Doctor entity.
  *
  * @ejb.bean
  *      name="Doctor"
  *      type="CMP"
  *      local-jndi-name="Doctor"
  *      primkey-field="doctorId"
  *      view-type="local"
  *
  * @ejb.persistence
  *      table-name="doctor"
  *
  * @ejb.transaction
  *      type="Required"
  */
```

Method-level tags specify attributes such as a particular method's transactional configuration (if it should appear in the remote/local component interface), create methods, relationship attributes, etc. The following is an example from the Illness bean:

```
/**
  * @ejb.interface-method
  * @ejb.relation
  *      name="Doctor-Illness"
  *      role-name="Illness-Has-Doctor"
  *
  * @weblogic.column-map
  *      foreign-key-column="doctor_id"
  *      key-column="doctor_id"
  *
  */
  public abstract DoctorLocal getDoctor();
```

Container-Managed Relationships

One of the rather challenging tasks for XDoclet newcomers is establishing various container-managed relationships among CMP entity beans via the XDoclet tags. XDoclet handles relationships with the @ejb.relation tag and @weblogic.column-map for specifying the familiar <column-map> elements that must be listed in the weblogic-cmp-rdbms-jar.xml file. Let's look at an example from PatientBean.java:

```
/**
   * @ejb.interface-method
   * @ejb.relation
   *      name="Doctors-Patients"
   *      role-name="Patients-Have-Doctors"
   *
   * @weblogic.relation
   *      join-table-name="doctor_patient_link"
   *
   * @weblogic.column-map
   *      foreign-key-column="patient_id_fk"
   *      key-column="patient_id"
   *
   */
   public abstract Collection getDoctors();
```

Here we are specifying a many-to-many relationship between the Doctor and the Patient beans. The name and role-name attributes will be used to generate the necessary elements in the ejb-jar.xml file. The WebLogic-specific tags specify a common join table for the relationship as well as the <column-map> elements that will be used in the generated weblogic-cmp-rdbms-jar.xml file. A bidirectional relationship is implied here due to the corresponding XDoclet relationship tags in DoctorBean.java. To demonstrate how a windirectional relationship is described, let's look at PatientBean,java.

```
/**
   * @ejb.interface-method
   * @ejb.relation
   *      name="Patients-Illness"
   *      role-name="Patient-Has-Illness"
   *      target-ejb="Illness"
   *      target-role-name="Illness-Has-Patient"
   *      target-cascade-delete="yes"
   *
   * @weblogic.target-column-map
   *      foreign-key-column="patient_id"
   *      key-column="patient_id"
   */
   public abstract Collection getIllnesses();
```

Here we are specifying a unidirectional relationship between the Patient and Illness beans. The target-ejb, target-role-name, and target-cascade-delete are used to specify the other side of the unidirectional relationship. XDoclet tags describing this particular relationship are not included in the corresponding Illness bean, since we are explicitly specifying the related entity from the Patient bean. In a unidirectional relationship, the @weblogic.target-column-map tag is used to specify the related bean's <column-map> elements.

Other WebLogic-Specific XDoclet tags

XDoclet supports both WebLogic 6.1 and 7.0 deployment descriptor settings.

Instead of describing various WebLogic-specific characterstics for EJBs explicitly in the weblogic-ejb-jar.xml and weblogic-cmp-rdbms-jar.xml files, you can express them using XDoclet tags in the centralized EJB bean file. This eliminates the need to create these XML files manually.

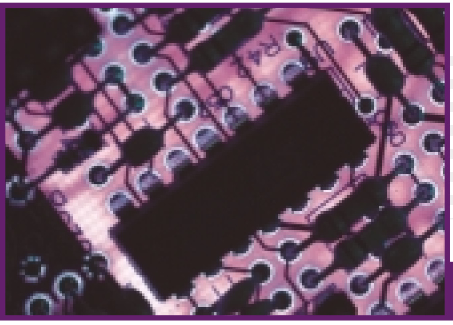
Following is an example of how to configure WebLogic to autoincrement an entity bean's primary key, set the concurrency strategy element to ReadOnly, and specify that the WebLogic server is not required to call ejbLoad() upon each new transaction:

```
/**
  * @weblogic.cache
  *      concurrency-strategy="ReadOnly"
  *      read-timeout-seconds="5"
  *
  * @weblogic.persistence
  *      db-is-shared="False"
  *
  * @weblogic.automatic-key-generation
  *      generator-type="ORACLE"
  *      generator-name="doctor_seq"
  *      key-cache-size="10"
  */
```

XDoclet provides many other WebLogic tags that allow the developer to conveniently configure features such as clustering, field groups, and relationship caching in a single bean file via Javadoc comments. Consult the XDoclet documentation for more information on these specific tags as well as other features provided by XDoclet. Utilizing the powerful capabilities of XDoclet with BEA WebLogic Server defintitely facilitates the development of EJBs.

References

- http://xdoclet.sourceforge.net
- http://jakarta.apache.org/ant
- http://dev2dev.bea.com



PLATFORM

If you've ever worked as a Weblogic consultant, chances are this scenario will look all too familiar:

Debugging WebLogic Platform Internals

SHARPEN YOUR TROUBLESHOOTING SKILLS

BY **STEVE BUZZARD**

You're at a high-profile client site as the "BEA WebLogic Expert." You were called in last minute because they are having "intermittent" problems in their newly deployed production system. The problems appear related to the WebLogic (Server/Portal/Integration) <insert here> subsystem, but you can't be sure. There is absolutely nothing unusual in the standard output; neither is there anything amiss in the WebLogic server or domain logs.

Despite poring over the docs, newsgroups, dev2dev, and the support site ("Ask BEA"), you still can't get a handle on the problem. You've double and triple checked the client's code and their WebLogic configuration, and verified that they're following the <insert here> specs. You finish off the usual checklist:

- *Is the OS Patch Level reasonably current (at least to the level required by the JVM used)?*
- *How's the OS environment (file descriptors, TCP parameters, network parameters, ulimit settings, etc.)?*
- *Anything really off with JVM version and/or parameters?*
- *Any type-2 JDBC drivers or other native code lurking about that could cause problems?*

No smoking gun here, unfortunately, and the client senior management is beginning to breathe down your neck. You'd fire up your favorite debugger in their QA environment (can you reproduce it there?) in an attempt to track it down, but it's not really that kind of "bug." It could very well be a product issue, you think. You check the latest release notes but nothing seems even vaguely related. Better open up a support case. You open up the case and the support engineer is researching the issue but can find no related CRs, so the odds of a quick fix or workaround are long. Maybe it's not a product issue …

Does this provide you with a touch of déja`vu? I've run into variations of this scenario countless times and have subsequently uncovered a set of "undocumented" debugging parameters and properties that have helped me track down the source of trouble and saved my hide on more than one occasion. Several of these parameters have been bandied about quite a bit on the various BEA newsgroups, but never specified as a whole.

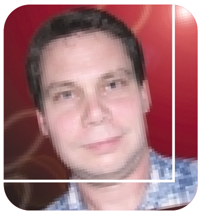
Warning: The debugging parameters I'm about to describe are undocumented. This means that they are also officially unsupported by BEA and subject to change (or removal) at any time.

WebLogic Server 5.1

For those who still indulge in 5.1, Table 1 lists its debugging parameters. They are specified as system properties (either on the command line or within the global, cluster, and/or server weblogic.properties files).

WebLogic Server 6.x/7.x

The WebLogic Server 6.x and 7.x Debugging Parameters are tied to the ServerDebug MBean within the app server's JMX infrastructure. A WebLogic Domain's admin server and each associated managed server have a corresponding <ServerDebug> element within the config.xml file (subordinate to that server's <Server> element). Note that because these parameters are undocumented, you cannot update them using the WebLogic Console but must hand-edit the config.xml file directly. For the uninitiated, the config.xml file holds the configuration for a WebLogic Domain and lives in the domain's top-level directory for the admin server. Almost all the parameters are of the "true/false" variety (a noted exception is "DebugJAXRPCDebugLevel", which is an integer value greater than 0: the higher the number, the more verbose the output). Be sure that the administration server is not running before you edit the config.xml (otherwise, your changes may get overwritten by the running server). Also be sure that you back up



AUTHOR BIO

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

CONTACT...

sbuzzard@anexinet.com

the config.xml file before you begin modifying it so that you have a valid configuration should you "mess up." Note that the debugging parameters I list below are based on 7.0 sp1 and some parameters are not valid in 6.1 (the server will quickly let you know which ones upon start up!). The debug information is printed to the standard output, so ensure that you are not redirecting stdout to /dev/null (as is often the case in a test or production environment). Also, setting these parameters to "true" can result in very verbose output, so flip the debug "switches" judiciously, based on the subsystem you suspect is associated with the problems you're trying to track down (e.g., if it's an EJB-related issue, set only the DebugEJB... parameters to "true"; see Listing 1).

In addition to these JMX-based debugging parameters, there is a set of system properties you can set for additional debugging information that is written to the WebLogic log. Most of the properties are set as such:

```
   -Dweblogic.Debug=<debug-parameter1,debug-
parameter2,debug-parameter3,…>
```

These debug parameters include:

IIOP
- weblogic.iiop.ots
- weblogic.iiop.transport
- weblogic.iiop.marshal
- weblogic.iiop.startup

JDBC/Data Sources
- weblogic.JDBCConn
- weblogic.JDBCSQL
- weblogic.JDBCConnStackTrace

JMX/Deployment
- weblogic.commoAdmin
- weblogic.commoProxy
- weblogic.deployerRuntime
- weblogic.MasterDeployer
- weblogic.deployTask
- weblogic.deployHelper
- weblogic.MasterDeployer
- weblogic.OamDelta
- weblogic.OamVersion
- weblogic.slaveDeployer.semaphore
- weblogic.slaveDeployer
- weblogic.ConfigMBean
- weblogic.ConfigMBeanEncrypt
- weblogic.ConfigMBeanSetAttribute
- weblogic.management.DynamicM BeanImpl
- weblogic.management.DynamicM BeanImpl.setget
- weblogic.mbeanProxyCache
- weblogic.mbeanDelete
- weblogic.mbeanQuery
- weblogic.MBeanInteropList
- weblogic.mbeanProxy
- weblogic.registerMBean
- weblogic.getMBeanInfo
- weblogic.getMBeanAttributes
- weblogic.addDependenciesRecursively
- weblogic.MBeanListener
- weblogic.application
- weblogic.deployer
- weblogic.appPoller
- weblogic.appManager
- weblogic.BootstrapServlet
- weblogic.fileDistributionServlet

Application Deployment
- weblogic.J2EEApplication
- weblogic.application
- weblogic.appPoller
- weblogic.appManager

JTA
- weblogic.JTAGateway
- weblogic.JTAGatewayStackTrace
- weblogic.JTA2PC
- weblogic.JTA2PCStackTrace
- weblogic.JTAHealth
- weblogic.JTAPropagate
- weblogic.JTARecovery
- weblogic.JTAXA
- weblogic.JTAXAStackTrace
- weblogic.JTAResourceHealth
- weblogic.JTAMigration
- weblogic.JTARecoveryStackTrace
- weblogic.JTANaming
- weblogic.JTATLOG
- weblogic.JTALifecycle

A concrete example of debugging a TxDataSource problem might be to set the following on your server's start-up command line:

```
-Dweblogic.Debug=weblogic.JDBCConn,weblogic.
    JDBCSQL,weblogic.JTA2PC
```

There are also a few other system properties floating around WebLogic Server that are set on the command line:
- Dweblogic.ejb.cache.debug
- Dweblogic.ejb.cache.verbose
- Dejb.enableCacheDump
- Dweblogic.ejb20.cmp.rdbms.debug
- Dweblogic.ejb20.cmp.rdbms.verbose
- Dweblogic.ejb20.persistence.debug
- Dweblogic.ejb20.persistence.verbose
- Dweblogic.ejb20.compliance.debug
- Dweblogic.ejb20.compliance.verbose
- Dweblogic.ejb.deployment.debug
- Dweblogic.ejb.deployment.verbose
- Dweblogic.ejb20.dd.xml
- Dweblogic.ejb.deployer.debug
- Dweblogic.ejb.deployer.verbose
- Dweblogic.ejb.verbose.deployment
- Dweblogic.ejb20.ejbc.debug
- Dweblogic.ejb20.ejbc.verbose
- Dweblogic.ejb.runtime.debug
- Dweblogic.ejb.runtime.verbose
- Dweblogic.ejb20.jms.poll.debug
- Dweblogic.ejb20.jms.poll.verbose
- Dweblogic.ejb20.security.debug
- Dweblogic.ejb20.security.verbose
- Dweblogic.ejb.locks.debug
- Dweblogic.ejb.locks.verbose
- Dweblogic.ejb.bean.manager.debug
- Dweblogic.ejb.bean.manager.verbose
- Dweblogic.ejb.pool.InstancePool.debug
- Dweblogic.ejb.pool.InstancePool. verbose
- Dweblogic.ejb.swap.debug
- Dweblogic.ejb.swap.verbose
- Dweblogic.j2ee.dd.xml
- Dweblogic.kernel.debug

WebLogic Portal

WebLogic Portal (both 4.0 and 7.0) uses System Properties to control their debug output. These Portal properties are tied to

"BEA Support often recommends trying out particular debugging parameters to help track down specific issues"

specific classes and can be set in two different ways:

1. ***Set one or more system properties on the command line.*** These property names are of the form:

```
debug.<fully qualified class name>.
```

For example, if I want debugging information on the processing of the Entitlements Access Controller class, I'd add the following to my Portal Server startup command line:

```
-Ddebug.com.bea.p13n.entitlements.
    accesscontroller.AccessController=true
```

2. ***Create a file entitled "debug.properties" in the Portal Server working directory (in 7.0, this is the domain directory; in 4.0, this is the parent of the "config" directory).*** For each Portal class you desire debug output from, add a line in this file of the form:

```
<fully qualified class name>=true.
```

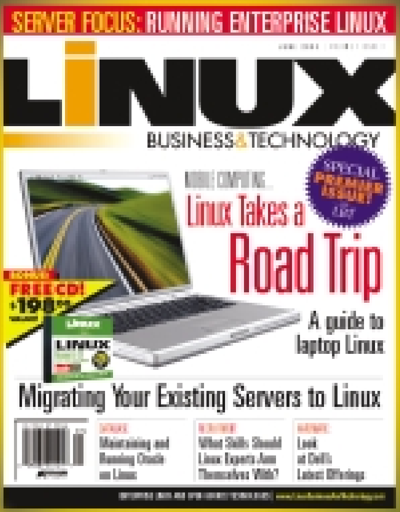
For example, if I want debugging informa-

Listing 1:

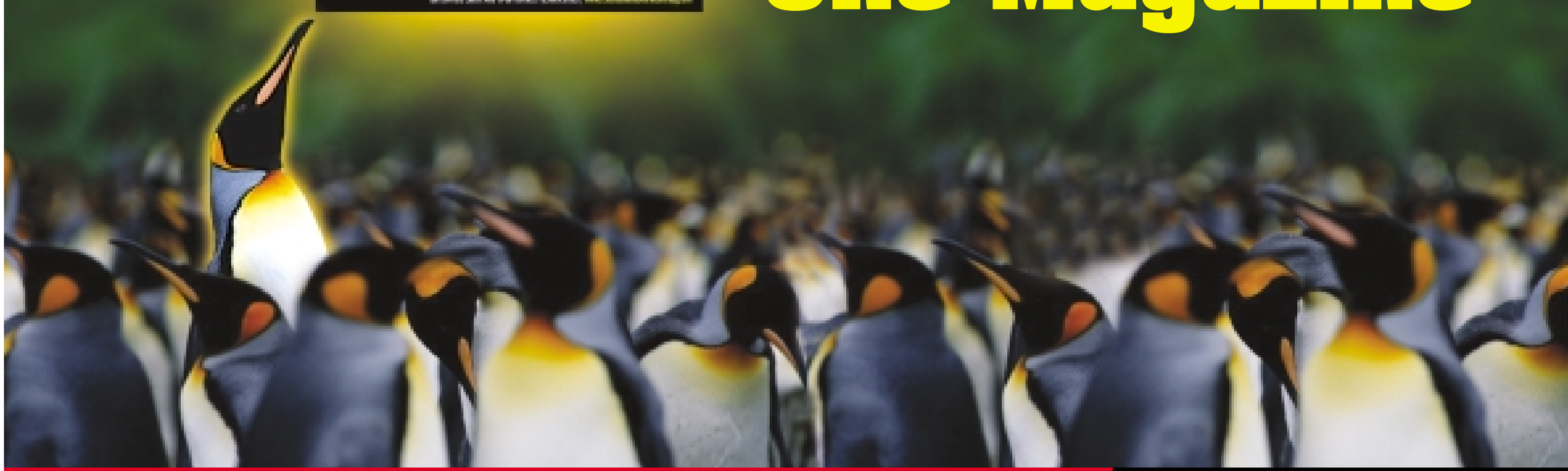
```
  <ServerDebug
         DebugAbbreviation="false" DebugCluster="false"
DebugClusterAnnouncements="false"
         DebugClusterFragments="false"   DebugClusterHeartbeats="false"
         DebugConnection="false"   DebugConnectorAllocConnection="false"
         DebugConnectorDetectedIdle="false"
DebugConnectorDetectedLeak="false"
         DebugConnectorDumpToConsole="false"
DebugConnectorDumpToLog="false"
         DebugConnectorFreeConnection="false"
DebugConnectorGetConnection="false"
         DebugConnectorPoolManagement="false"
DebugConnectorPoolModified="false"
         DebugConnectorPoolShutdown="false"
DebugConnectorPoolStartup="false"
         DebugConnectorServiceStartup="false"
DebugConnectorXAResource="false"
         DebugDGCEnrollment="false"   DebugDRSCalls="false"
         DebugDRSHeartbeats="false"   DebugDRSMessages="false"
         DebugDRSQueues="false"   DebugDRSStateTransitions="false"
         DebugDRSUpdateStatus="false"   DebugEJB="false"
         DebugEJBCache="false"   DebugEJBCalls="false"
         DebugEJBDeployment="false"   DebugEJBFreepool="false"
         DebugEJBLocking="false"   DebugEJBPersistence="false"
         DebugEJBSecurity="false"   DebugEmbeddedLDAP="false"
         DebugEmbeddedLDAPLogLevel="false"
DebugEmbeddedLDAPLogToConsole="false"
         DebugEmbeddedLDAPWriteOverrideProps="false"
DebugEventManager="false"
         DebugFailOver="false"   DebugHttp="false"   DebugIIOP="false"
         DebugJAXPDebugLevel="5"  DebugJAXPIncludeClass="false"
DebugJAXPIncludeLocation="false"
         DebugJAXPIncludeName="false"   DebugJAXPIncludeTime="false"
         DebugJAXPOutputStream="false"   DebugJAXPUseShortClass="false"
         DebugJMSBackEnd="false"   DebugJMSBoot="false"
         DebugJMSCommon="false"   DebugJMSConfig="false"
         DebugJMSDispatcher="false"   DebugJMSDurableSubscribers="false"
         DebugJMSFrontEnd="false"   DebugJMSJDBCScavengeOnFlush="false"
         DebugJMSLocking="false"   DebugJMSStore="false"
DebugJMSXA="false"
         DebugJNDI="false"    DebugJNDIFactories="false"
DebugJNDIResolution="false"
         DebugLeaderElection="false"    DebugLoadBalancing="false"
DebugMessaging="false"
         DebugMessagingBridgeDumpToConsole="false"
DebugMessagingBridgeDumpToLog="false"
         DebugMessagingBridgeRuntime="false"
DebugMessagingBridgeStartup="false"
         DebugRC4="false"   DebugRSA="false"   DebugReplication="false"
         DebugReplicationDetails="false"   DebugRouting="false"
DebugSSL="false"
         DebugSecurityAdjudicator="false"   DebugSecurityAtn="false"
         DebugSecurityAtz="false"   DebugSecurityAuditor="false"
         DebugSecurityCredMap="false"   DebugSecurityKeyStore="false"
         DebugSecurityPasswordPolicy="false"
DebugSecurityProfiler="false"
         DebugSecurityRealm="false"   DebugSecurityRoleMap="false"
         DebugSecurityUserLockout="false"
DebugTunnelingConnection="false"
         DebugTunnelingConnectionTimeout="false"
DebugURLResolution="false"
         DebugXMLEntityCacheDebugLevel="false"
DebugXMLEntityCacheDebugName="false"
         DebugXMLEntityCacheIncludeClass="false"
DebugXMLEntityCacheIncludeLocation"false"
         DebugXMLEntityCacheIncludeName="false"
DebugXMLEntityCacheIncludeTime="false"
         DebugXMLEntityCacheOutputStream="false"
DebugXMLEntityCacheUseShortClass="false"
         DebugXMLRegistryDebugLevel="false"
DebugXMLRegistryDebugName="false"
         DebugXMLRegistryIncludeClass="false"
DebugXMLRegistryIncludeLocation="false"
         DebugXMLRegistryIncludeName="false"
DebugXMLRegistryIncludeTime="false"
         DebugXMLRegistryOutputStream="false"
DebugXMLRegistryUseShortClass="false"
         ForceGCEachDGCPeriod="false"   JdbcConn="false"
         Jdbcsql="false"   Jta2pc="false"   Jta2pcStackTrace="false"
Jtaapi="false"
         JtaGateway="false"   JtaGatewayStackTrace="false"
JtaHealth="false"
         Jtajdbc="false"   JtaLifecycle="false"   JtaMigration="false"
         JtaNaming="false"   JtaNamingStackTrace="false"
JtaPropagate="false"
         Jtarmi="false"   JtaRecovery="false"
JtaRecoveryStackTrace="false"
         JtaResourceHealth="false"   Jtatlog="false"   Jtaxa="false"
         JtaxaStackTrace="false"   ListenThreadDebug="false"
LogDGCStatistics="false"
         MagicThreadDumpBackToSocket="false"
MagicThreadDumpEnabled="false"
         Name="Admin Or Managed Server Name"
```

Premiering
June 2003
at
JavaOne
Sun's 2003 Worldwide Java Developer Conference

www.sys-con.com



Millions of Linux Users One Magazine



Linux Business and Technology

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

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

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

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

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

Regular features will include:

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

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE $71.76

YOU PAY ONLY

$49.99

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

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

LINUX EDGE conference&expo

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

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

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

TABLE 1

PROPERTY NAME	DESCRIPTION	VALID VALUES
weblogic.debug.cluster	Debug info on cluster communications	0 – 3 (default: 0)
weblogic.debug.connection	Debug info on JVM-JVM connections	0 - ? (default: 0)
weblogic.debug.ejb.cache	Debug info on cache management	true or false (default: false)
weblogic.debug.ejb.calls	Debug info on life-cycle calls	true or false (default: false)
weblogic.debug.ejb.deployment	Debug info on deployment	true or false (default: false)
weblogic.debug.ejb.freepool	Debug info on freepool management	true or false (default: false)
weblogic.debug.ejb.locking	Debug info on container locking	true or false (default: false)
weblogic.debug.ejb.persistence	Debug info related to persistence	true or false (default: false)
weblogic.debug.ejb.security	Debug info on container security	true or false (default: false)
weblogic.debug.eventManager	Debug info related to WLS events	true or false (default: false)
weblogic.debug.gc	Debug info on garbage collection	true or false (default: false)
weblogic.debug.httpd	Debug info on HTTP traffic	true or false (default: false)
weblogic.debug.httpd.servlet	Debug info on Servlet operations	true or false (default: false)
weblogic.debug.jndi	Debug info on JNDI operations	0 - ? (default: 0)
weblogic.debug.logDGCStatistics	Distributed Garbage Collection (RMI)	true or false (default: false)
weblogic.debug.replication	Debug info on session replication	true or false (default: false)

WebLogic Debug Properties

tion on the Portal Management sub-system, I create a "debug.properties" file with content similar to:

```
com.bea.portal.manager.internal.PortalManagerDele
   gateImpl=true
com.bea.portal.manager.internal.PagePersonalizati
   onImpl=true
com.bea.portal.manager.internal.PortalParser=true
com.bea.portal.manager.internal.PortalPersistence
   Manager=true
com.bea.portal.manager.internal.PortletParser=tru
e
com.bea.portal.manager.internal.PortletPersistenc
   eManager=true
com.bea.portal.manager.internal.PortletStateImpl=
   true
com.bea.portal.manager.internal.UserPortalStatePo
   licy=true
```

WebLogic Integration

WebLogic Integration also uses System Properties to enable debugging information, but the scheme here is less complex (and, some would say, less flexible) than that of Portal. Simply set the following properties on your WLI Server start up command line according to your needs:

- Dwlc.debug.signature=true
- Deci.repository.helper.debug=true
- Dwli.bpm.client.security.debug=true
- Dwli.bpm.studio.timeprocessor.debug=true
- Dwli.bpm.studio.debug=true
- Dwli.bpm.server.common.timedevent.debug=true
- Dwli.bpm.server.common.xmltemplate.debug=true
- Dwli.bpm.server.eventprocessor.addrmsgdebug=true
- Dwli.bpm.server.eventprocessor.debug=true
- Dwli.bpm.server.jms.debug=true
- Dwli.bpm.server.plugin.debug=true
- Dwli.bpm.server.workflow.debug=true
- Dwli.bpm.server.businesscalendar.debug=true
- Dwli.bpm.server.busop.debug=true
- Dwli.bpm.server.workflow.action.taskduedate.debug=true
- Dwli.bpm.server.workflow.timedevent.debug=true
- Dwli.bpm.server.xml.debug=true
- Dwli.bpm.server.xslt.debug=true
- Dwli.bpm.server.workflow.start.debug=true
- Dwli.bpm.server.workflowprocessor.debug=true
- Dwli.common.server.errorlistener.debug=true

Summary

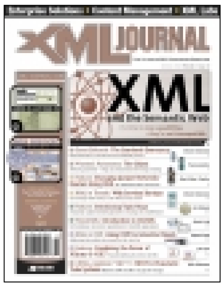
These debugging parameters can be of great help (BEA Support often recommends trying out particular debugging parameters to help track down specific issues) and can also aid in your understanding of how the WebLogic Server, Portal, and Integration internals operate (which in turn sharpens your troubleshooting skills). Although they are official undocumented and unsupported (who needs real "support" for a debug property?), they can truly be a lifesaver.

A LIMITED TIME SAVINGS OFFER FROM SYS-CON Media

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS



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

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

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

☐ **Linux Business & Technology**

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

☐ **Java Developer's Journal**

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

☐ **Web Services Journal**

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

☐ **.NET Developer's Journal**

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

☐ **XML-Journal**

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

☐ **WebLogic Developer's Journal**

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

☐ **ColdFusion Developer's Journal**

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

☐ **Wireless Business & Technology**

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

☐ **WebSphere Developer's Journal**

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

☐ **PowerBuilder Developer's Journal**

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

3-Pack

Pick any 3 of our magazines and save up to $275.00 Pay only $175 for a 1 year subscription plus a FREE CD

- 2 Year – $299.00
- Canada/Mexico – $245.00
- International – $315.00

6-Pack

Pick any 6 of our magazines and save up to $350.00 Pay only $395 for a 1 year subscription plus 2 FREE CDs

- 2 Year – $669.00
- Canada/Mexico – $555.00
- International – $710.00

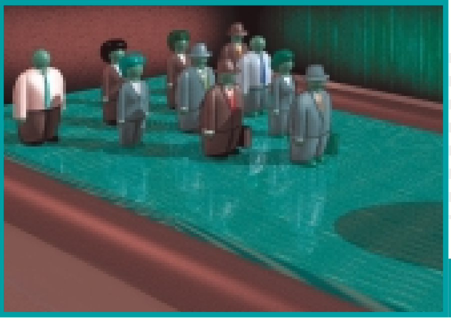
9-Pack

Pick 9 of our magazines and save up to $400.00 Pay only $495 for a 1 year subscription plus 3 FREE CDs

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

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

SYS-CON MEDIA

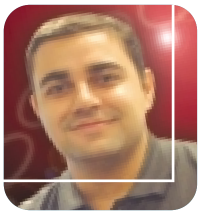


Web services are a type of service that can be shared by and used as components of distributed, Web-based applications. They are based on a collection of standards and protocols that allow us to make processing requests to remote systems by speaking a common, nonproprietary language and using common protocols (like HTTP, SOAP).

Developing JAX-RPC–based Web Services

WRITE A CLIENT AND A WEB SERVICE THAT HIDE COMPLEXITY

BY RAJESH SUMRA



AUTHOR BIO...

Rajesh Sumra is a senior software engineer in HP's Wireless Solutions Lab. He has worked with the e-speak project for more than a year and was involved in developing UDDI Server functionalities for HP's UDDI Server. Currently he is involved in designing and developing a Web services–based framework for mobile infrastructure. Rajesh holds a master's in information technology from IIIT, Bangalore.

CONTACT...

sumra@hp.com

The Java API for XML-based remote procedure calls (JAX-RPC) simplifies the process of building Web services that incorporate XML-based RPC. It defines mappings between Java types and XML types that attempt to hide the details of XML and provide a familiar method-call paradigm. This article looks at how developers can use JAX-RPC to implement and call SOAP-based Web services described by the Web Services Description Language (WSDL) on the BEA WebLogic platform.

JAX-RPC–Based Web Services

JAX-RPC fully embraces the heterogeneous nature of Web services. It allows a JAX-RPC client to talk to another Web service deployed on a different platform and coded in a different language. JAX-RPC provides the specification for invocation modes, client generation, parameter modes, mappings for Java to WSDL and WSDL to Java, and client-side APIs for invoking the Web service.

Invocation Modes and Clients

JAX-RPC supports three kinds of Web services invocation modes:

- ***Synchronous request–response:*** A client invokes a remote method on a Web service; the thread blocks while it is processed by the Web service and receives a return value or an exception.
- ***One-way RPC mode:*** A client invokes a remote method on a Web service in one-way mode; the thread does not block and continues execution. The client gets no return value.
- ***Nonblocking RPC mode:*** A client invokes a remote method on a Web service and continues processing in the same thread. Later, the client processes the remote method by performing blocking, receive, or polling for return values.

With these modes, JAX-RPC supports two types of Java client applications – namely, static client and dynamic client. In this article, we will see how both Java clients can be written or generated using synchronous request-response mode on the WebLogic platform.

Parameter Modes

The Web service invocation based on JAX-RPC uses pass-by-copy semantics for parameter passing. It does not support the pass-by-reference way of parameter passing. The following types of parameters are supported by jax-rpc:

- ***IN type:*** An IN parameter is passed as copy. The value of the IN parameter is copied before a Web service invocation. The return value is created as a copy and returned to the Web service client.
- ***OUT type:*** An OUT parameter is passed as copy without any input value to the Web service method. The Web service method fills the OUT parameter and returns it to the client.
- ***INOUT type:*** An INOUT parameter is passed as copy with an input value to the Web service method. The Web service method uses the input value, processes it, fills the INOUT parameter with a new value, and returns it to the client.

The parameter passing mode for OUT and INOUT parameters uses holder classes. Holder classes enable the mapping to preserve the intended WSDL signature and parameter-passing semantics. The JAX-RPC specification includes "holder classes" for the mapping of simple XML types to the Java data types. The holder classes for primitive ones (e.g., int, float, etc.) are available with JAX-RPC implementation under the avax.xml.rpc.holders package. For the complex XML data types, the name of the holder class is constructed by appending Holder to the name of the corresponding Java class. These generated

holder classes are packaged as part of the generated subpackage named holders in the WSDL-to-Java mapping.

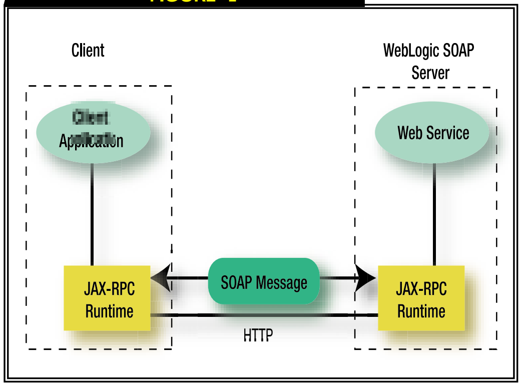
Each holder class provides the following methods and fields:

- ***A public field named "value":*** The mapped Java type
- ***A default constructor:*** Initializes the value field to a default value
- ***A constructor:*** Sets the value field to the passed parameter

Web Service Invocation

BEA WebLogic 7.0 comes with a JAX-RPC 1.0–compliant runtime engine, which has both client-side and service-side libraries and deployment tools. Figure 1 shows a normal Web service invocation flow for synchronous request-response mode. The client application uses WebLogic's JAX-RPC runtime to perform a remote procedure call to invoke a method on the Web service.

FIGURE 1



JAX-RPC–based Web service invocation

The runtime serializes Java objects to the SOAP message, which is sent to the Web service end point using HTTP transport. On the Web service side, WebLogic Platform receives this request and the service side JAX-RPC runtime deserializes the SOAP message into Java types and invokes the method on the Web service. The Web service, after processing the request, sends the response back to the client in a similar fashion.

I have explained the simple Web service invocation based on JAX-RPC. The remainder of this article develops a sample Web service, and a static and dynamic client for it, on the WebLogic platform.

An Order Processing Web Service

I chose an order processing example for its proximity to a real busines-use case.

This Web service will be capable of processing and updating a given "Order" using two methods – processOrder and updateOrder. ProcessOrder will take orderID string as an IN parameter and Order object as an OUT parameter. It will return status string as the return parameter. The updateOrder will take Order object as the INOUT parameter, update the orderDate, and return the Order object back to client. As both methods use a complex datatype, Order, that is also an OUT/INOUT parameter, a holder class has to be developed. The Order class and its holder class are shown in Listings 1 and 2 (the complete code can be found on the Web at www.sys-con.com/weblogic/sourcec.cfm):

Now let's develop our Web service for the described functionality. The code is shown in Listing 3.

The next step is to compile and deploy it on the WebLogic platform. WebLogic comes with some easy-to-use deployment tools using Ant (Apache's build tool) and includes predefined tasks for deploying the service on WebLogic Server 7.0. The sample "build.xml" is shown in Listing 4. This build file will generate an EAR file that will store information about the service endpoint, deployment descriptor (web-services.xml), and serializer and deserializer code (OrderCodec class); and will even generate a static client in the sample.client package and bundle it in a JAR file. The deployment descriptor (web-services.xml; it can be found in 'web-services.war' in 'ws_order.ear') has to be changed for the style of the parameter so the first method is OUT and the second method is INOUT. Once these changes are complete, repackage whole files into the EAR file (ws_order.ear) and deploy it using WebLogic's admin console. The OrderProcessing service will be available at the following URL:

```
http://<your_machine_name>:<port-
num>/<contextURI>/<serviceURI>
```

In our case it may look like :

```
http://localhost:7001/WebServices/OrderProcessing
```

OrderProcessing Web Service Clients

Static Client

The static client to invoke a Web service uses a more strongly typed Java interface. On WebLogic, writing a static client is really easy as the Ant tasks specified above generate the client JAR, which has the following:

- A Web service–specific implementation of the Service interface, which acts a stub factory
- Serialization class for non-built-in data types and their Java representations
- An interface and implementation of each SOAP port

A client application has to use the generated classes in the client JAR. The static client is invoking the processOrder () method of the OrderProcessing Web service (see Listing 5).

Dynamic Client

A dynamic client is analogous to looking up and invoking the Java class methods using the reflection APIs. All the information like target endpoint and method parameters must be set explicitly. Listing 6 shows you how to write a dynamic client for invoking the updateOrder method OrderProcessing Web service.

Running the Client

For running both types of client, set the environment using a WebLogic-specific "setWLSEnv.bat" file. This .bat/.sh will set the classpath variable to include JAX-RPC–related JAR files. Additionally, we have to add a client-specific JAR file generated by running the Ant task earlier. After the environment is set, simply give the following commands:

For the static client:

```
<prompt>java sample.client.StaticClient
```

For the dynamic client:

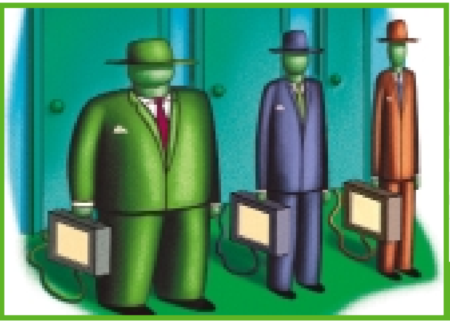
```
<prompt>java sample.client.DynamicClient
```

Conclusion

In this article I tried to demonstrate how to develop JAX-RPC–based Web services. The method described here gives the developer the freedom to write client and Web services in a way that hides all the complexities of serializing objects on-the-wire XML format. For the developer, it will simply appear to be a Java method invocation. This article also highlights how easy it is to expose a given "Java class" as a Web service.

References

- http://e-docs.bea.com/wls/docs70/webserv/index.html
- http://java.sun.com/xml/jaxrpc

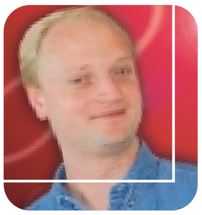


Lightning Never Strikes Twice?

THE EASY ANSWER IS "YES"

BY PETER HOLDITCH

I thought I would devote this month's column to a subject that appeared a while ago in the weblogic.developer.interest.transaction newsgroup on newsgroups.bea.com. As an opening comment, if you have never seen these newsgroups and you are a WebLogic developer, then go find them immediately! They're a mine of useful information, and a great place to get questions answered or collect opinions on design ideas or the like.



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

Plug over, what was the thread about? Well, the bit I'd like to write up went as follows…

Assume I have a successful transaction that has not yet been committed to the database, but the client has already been notified about its success. The client can start another transaction that tries to modify the same data (or a portion of it). It seems logical that a second transaction should not be able to succeed before the first is 'really' complete. So there should not be a case when there are two transactions that affect the same piece of data and overlap in time after being reported complete (i.e. in the AbandonTimeout interval).

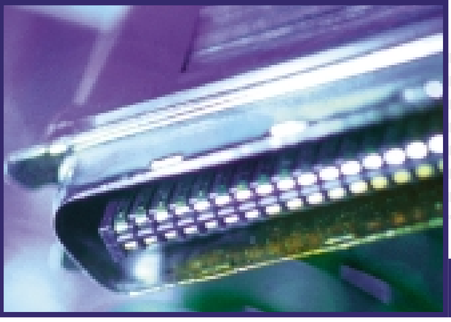
The short answer to this is "yes," but since one-word articles aren't in vogue this season, I'd like to walk through this true statement and highlight the important parts.

An Everyday Story of Transactional Folk

So the first part… a transaction is started, some data is accessed, and the transaction is committed. This is normal enough, an everyday tale of transactional folk; bread and butter for the average J2EE application developer. Less obvious, but in fact no less bread and butter, is that the client could see a successful outcome to a commit call prior to all the database changes that made up the transaction being committed. From the transaction manager's point of view, once all the participants in a transaction acknowledge their ability to commit if asked (that is, reply "YES" to a prepare request) and the decision to commit implied by a unanimous set of yes votes is written to the transaction log, then the transaction is as good as complete. Okay, so maybe the soon-to-be-permanent changes to the data aren't yet visible to the outside world, but come what may, eventually they will be. At this point, therefore, the TM can respond affirmatively to the client's commit request so that the client can get on with the rest of its life. At some future time the commit will complete. Note the phrase "some future time" – I cannot be more specific than that…. The time to complete a transaction will be dependent on many factors – how fast are the networks, databases, and so on involved? How many participants did the transaction have? Will all the databases and the transaction manager remain up while the completion happens? In practice, completion could be nearly instantaneous, or it could take a very long time indeed. However, I already said that the client doesn't care – it's off minding its own business while the transaction manager takes care of this stuff.

Of course, to say that the client doesn't care is possibly an oversimplification, as pointed out in the newsgroup quote. Imagine that the client wants to alter the same piece of data twice. The transaction manager has told it that transaction1 is complete, so it goes ahead, starts transaction2, and does the second set of updates. If the transaction2 updates affect the same pieces of database as transaction1 then we may have a problem. If the commit is complete, then everything is fine and the second transaction can go ahead. If, however, transaction1 has been given the green light to commit, but completion hasn't happened, the data will probably still be locked on its behalf, so transaction2 will fail since it is trying to modify locked data. Here we have a timing window. Sometimes the app works, sometimes it doesn't. And guess what? The sometimes will vary. In the beginning of a deployment there might be only a few resources involved in the transaction, and the transactions most often complete serially. No problem shows up. Now imagine that the deployment got more complex, and the data got spread over more databases. The commit takes longer to complete, and suddenly code that apparently worked is broken more often than not. This could clearly be a debugging headache of an unpleasant kind, especially if you're not the guy that wrote the original code (or if you are, it's so long ago you forgot how it works).

continued on page 47



The release of BEA WebLogic Server 7.0 offered several new features such as Web services; a pluggable, flexible security infrastructure; a migratable framework; new developer tools; JMS; and a new deployment model to provide a highly reliable framework for scalable and secure applications.

New Deployment Model in BEA WebLogic Server 7.0

PROVIDE A HOMOGENEOUS DEPLOYMENT IN A CLUSTERED ENVIRONMENT

BY **KUMAR ALLAMRAJU**

AUTHOR BIO...

Kumar Allamraju is a senior developer relations engineert at BEA Systems in the WebLogic Server support division. Kumar has five years of experience in object-oriented programming and in J2EE-related technologies.

CONTACT...

kumara@bea.com

In this article I'll talk about the deployment model introduced in WebLogic Server 7.0 and the different modes of application staging.

New Deployment Model

The new deployment model helps us prevent inconsistent deployment states across servers, especially clustered servers.

In previous versions of WebLogic Server, when an application was deployed, the admin server sent a copy of the application file(s) to all the targeted servers. The main problem with this approach is, if deployment of an application fails to any of these targeted servers, the entire deployment's state across its target servers became inconsistent. The second problem was, since the admin server copyied the files to managed servers upon their restart, server start-up was delayed as all the JSP's had to be compiled.

The new deployment model (aka "2 phase deployment protocol") was introduced to solve this problem in addition to maintaining domain consistency. In other words, this model ensures homogeneous application deployment in Clusters.

It also ensures better status reporting for success/failure of deployments. The WebLogic Server 6.x deployment style will continue to work as is; however, you are strongly encouraged to use the new deployment model as it is highly flexible and rich in terms of its new features:

- ***Consistent deployment states for clusters:*** If an application targeted to a cluster fails on any of the cluster members in the prepare phase or in the activate phase, the application is not activated on any of the cluster members. This helps to ensure that the cluster is kept homogeneous.
- ***Application ordering:*** At server startup, you set the order of application activations.
- ***Improved API:*** A simple API separates configuration from the actual deployment operations. When a deployment is requested, this API will create the necessary configuration (MBeans) for you.
- ***Deployment status:*** It is now easier to track the progress of a deployment, especially when it has multiple targets.

Now let's talk about the phases in the new deployment model.

- ***Prepare Phase (also known as first phase):*** The container prepares the application across all target servers. The container distributes or copies files and prepares the application and its components for activation, validating them and performing error checks on them. The purpose of the prepare phase is to ensure that the application and its components are in a state in which they can be reliably deployed.
- ***Activation Phase (also known as second phase):*** The actual deployment of the application or activation is done in this phase. After it is completed, the application is made available to clients.

If deployment of an application fails in either the prepare or activation phase, then the cluster deployment will fail. In the case of a non-cluster environment, the application won't be prepared or activated.

Application Staging

To efficiently support Content Management Providers, WebLogic Server 7.0 introduced several staging modes, described below.

Nostage

You do not need to copy application files to another location. A server in nostage mode will run applications deployed to it directly from their source directories. For example:

```
java weblogic. Deployer url {admin-url} -username
{admin-user}  -activate -nostage
```

```
 -source c:/myapps/myapp.ear -targets managed-
server1 -password {admin-password}
```

This command activates the EAR specified with the name myapp. The application is deployed to the targeted server, which in this case is called managed-server1.

The -nostage option indicates that the application path "c:/myapps" is available on all of the servers in the domain and that it is not necessary to "stage" or transfer the files to the managed servers (e.g., shared file system).

This staging mode may not be helpful if admin & managed servers are in different machines or in different Operating Systems.

Stage

The stage mode copies application files to servers targeted in deployment phase. This means that the application will automatically be copied to the staging directory on the server. The servers will initialize and run the application from this directory.

```
 java weblogic.Deployer url {admin-url} -user-
name {admin-user}  -password {admin-password} -
activate -stage  -source c:/myapps/myapp.ear -
targets managed-server1
```

This command activates the EAR specified with the name myapp. The application is deployed to the targeted server, which in this case is called managed-server1.

The -stage option indicates that the application files are copied from -source directory to a directory with the application name under each target server's staging directory.

External_stage

The administrator manually copies files to the server's staging directory.

In the external stage mode, the application will be run from a staging directory to which an external entity is expected to distribute the files. (e.g., CMP's)

```
 java weblogic.Deployer url {admin-url} -user-
name {admin-user}  -password {admin-password} -
activate -external_stage
 -source C:/myapp/myapp.ear -targets managed-
server1, managed-server2
```

This command activates the EAR specified with the name myapp. The application is deployed to the targeted server, which in this case is called managed-server1.

The -external_stage option indicates that the application files expect to be copied under each target server's staging directory.

Default Staging modes

For Admin servers, the default is "nostage". For Managed Servers, the default is "stage".

The staging directory for each server is located under $WL_HOME/user_projects/mydomain/{server_name}/stage

Available Deployment Tools for Two-Phase Deployment

WebLogic Server 7.0 provides the following deployment tools to deploy applications using the two-phase deployment model:

1. weblogic.Deployer command line utility
2. WLS admin console
3. WebLogic Builder
4. Deployment via applications folder

Conclusion

In this article we looked at the new deployment model introduced in WebLogic Server 7.0, which ensures homogeneous deployments in a clustered environment. I hope this will help you choose the appropriate staging mode depending on your deployment needs.

References

- http://e-docs.bea.com/wls/docs70/programming/deploying.html
- *java weblogic. Deployer – examples:* http://e-docs.bea.com/wls/docs70/faq/deploy.html#733195

"The new deployment model helps us prevent inconsistent deployment states across servers, especially clustered servers"

continued from page 44

So the moral of the story is simple – the whole two-phase transaction management idea assumes that the data being modified in any given transaction is independent of the data that was modified in the last one, or the data that will be modified in the next one. If you get unlucky and hit the same data twice accidentally, then exceptions get thrown, but that's fine because it should be an exceptional case. In practice, that's not a problem since online transaction processing (OLTP)–style applications tend to be like that – a call center where the operator deals with a different customer per call; an expenses application where an employee adds line items to an expense report one at a time; and so on. If, however, you find yourself designing an application that expects to repeatedly update the same piece of data, then put all the work related to it in a single transaction. This is one of the cases where a bigger transaction is better than a smaller one (the rule of thumb is usually that smaller is better).

Aliens Abducted My Data!

So, what does the AbandonTimeout interval have to do with this, I hear the observant among you cry. Well, conceptually very little. However, if for some reason WebLogic's transaction manager can't complete a transaction it has decided to commit (say aliens abducted one of the databases during the commit processing), then it won't keep banging its head against a wall forever. Periodically, the transaction manager retries the commit but if it still hasn't succeeded after the AbandonTimeout has been reached, it will give up and log its disappointment in the log (leaving a harassed human administrator to sort the situation out) This avoids wasting server resources to no good effect. For our completion timing discussion, however, it governs the maximum time a transaction will await completion. The default for this value is 24 hours (remember, I did say completion time could be long!).

So that's this month's article complete, more transactions next month. Until then, I'm off to Alpha Centauri to retrieve my database.

Everywhere I go developers, operations people, and IT executives ask me how best to keep their mission-critical applications up and running at peak performance. To address these concerns, WebLogic Developer's Journal is introducing a new column to answer questions from real WebLogic users like you on a range of Java application management topics.

Stay Up and Running

ANSWERS TO YOUR APPLICATION PERFORMANCE PROBLEMS

BY LEWIS CIRNE

Whether it involves the JVM, the application itself, the application server, the operating system, or the connections between your application and back-end supporting systems, submit your questions to "Ask the Expert" today.

Here are some examples of the topics that I'll be covering in this column.

What Are the Most Common Application Performance Issues?

Though I've come across many, I can safely narrow the bulk of performance and availability issues into three categories:

- ***Bottlenecks in back-end systems such as the database or CICS transaction systems:*** Caused by using potentially expensive resources ineffectively or more often than anticipated.
- ***Application code issues:*** These are not necessarily coding bugs you identify and tune up with a profiler, but rather higher-level issues related to the overall design of the application itself – how well components are behaving when they're interacting with each other under a real load in a real environment.
- ***Java runtime issues such as the garbage collector, the amount of heap used, and the amount of object thrash:*** Understanding how to tune your garbage collector is an important aspect of ensuring better application performance. Another issue that affects the Java runtime is a memory leak, which can of course, cause system failure.

What Areas of WebLogic Server Would You Recommend for Tuning?

One of the great advantages of J2EE application environments is that you have a lot of flexibility and freedom to tune a lot of parameters and that's great because those tunable parameters deliver the performance you need. Regarding application servers, focus on the contended resources that if improperly configured could introduce severe performance problems for your application. You must have a handle on:

- ***The number of threads available in your servlet thread pools:*** Too few and servlet requests can be blocked. Too many and you may be using system resources unnecessarily.
- ***The number of anticipated HTTP sessions in a single JVM:*** You need to make sure that you have enough execute threads available to support the average number of concurrent users anticipated for the application. If this number is too low, application performance will begin to degrade.
- ***The number of JDBC connections available in the pool***: To ensure optimal performance, make sure that the maximum number of JDBC connections in the pool is greater than the maximum anticipated active connections.
- ***How your application is utilizing the heap within the JVM:*** Your application needs to be monitored over time for rising trends in memory utilization. If allocating more memory doesn't fix the problem, you probably have a memory leak.
- ***The size of entity EJB pools:*** If your entity bean pools are too large, they may take up too much RAM; too small, and caching is ineffective.

What Is the Typical Production Environment for WebLogic Applications?

One of the real strengths of J2EE and WebLogic is their cross-platform nature. I've seen every type of conceivable production environment in which mission-critical WebLogic applications run. As such there are no clear, standard production hardware or OS environments. I've seen WebLogic running on almost every conceivable platform, including Linux, Solaris, AIX, HP-UX, even the mainframe. The fact is that every J2EE application is built to be unique, and the very nature of J2EE means that there is no standard production environment.

• • •

If you have a question for me, I invite you to send it to asklew@sys-con.com.

AUTHOR BIO

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

CONTACT...

asklew@sys-con.com

EDITORIAL

continued from page 5

Fortunately for BEA customers, it's not an either/or proposition. BEA WebLogic Portal sits on top of BEA WebLogic Server and integrates tightly into the J2EE container, making most of these architecture discussions an exercise in semantics. With the release of the revised BEA WebLogic Workshop, it will become even easier to create portal applications alongside standard J2EE components like EJBs.

As you might have guessed, the focus of this issue is portals.

Also, on a housekeeping note, Jason Westra has stepped down as editor-in-chief of *WLDJ* after accepting a new position in the industry that requires a much greater time commitment from him. While our new appointment to the top spot, to be announced shortly, gets ready for the year ahead, I'll be serving as managing editor and keeping everything running smoothly. I've been on the Editorial Advisory Board since its inception, so I'll be able to provide all the continuity we need to continue this fine publication. Good luck Jason, and thanks from us all for your hard work.

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altaworks	www.altaworks.com	603-598-2582	52
Attachmate	www.attachmate.com/SCb	800-426-6283	3
BEA Systems	http://dev2dev.bea.com/useworkshop	800-817-4BEA	23
Dirig Software	www.dirig.com/illusion/weblogic	603-889-2777	51
Documentum	www.documentum.com/develop	800-607-9546	33
Linux Business & Technology	www.sys-con.com	888-303-5282	39
Mercury Interactive	www.mercuryinteractive.com/bto	800-837-8911	29
PANACYA	www.panacya.com	877-726-2292	13
Precise Software Solutions	www.precise.com/wldj	800-310-4777	19
Programmer's Paradise	www.programmersparadise.com/BEA	800-445-7899	31
Quest Software	http://java.quest.com/jcsv/wldj	800-663-4723	17
Quest Software	http://java.quest.com/performance/wldj	800-663-4723	9
ReportingEngines	www.reportingengines.com/download/wldj1.jsp	888-884-8665	2
SYS-CON E-Newsletters	www.sys-con.com	201-802-3020	25
SYS-CON Publications	www.sys-con.com	888-303-5282	41, 45
SYS-CON Reprints	www.sys-con.com	201-802-3026	45
Web Services Journal	www.sys-con.com	888-303-5282	45
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	35
Wily Technology	www.wilytech.com	888-GET-WILY	4

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

LOOK WHAT'S COMING NEXT MONTH

Leveraging the Unified User Profile
by Karl Banke
A look at the programming and configuration steps necessary to use WebLogic Portal's built-in rules engine together with external user profile data to provide a personalized portal presentation.

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

There's Nothing to Fear About Mainframe Integration...Are You Listening BEA Developers?!
by William D. Mills
Why do organizations fear the mainframe? Problems start at "the line." This article looks at what we're afraid of, what the industry needs, and the "ideal" solution, including integration with WebLogic Workshop and finding the cause of the problems.

Health Monitoring and Notification of Servers in a Cluster
by Apurb Kumar
Let's look at some of the options available to help you stay on top of this problem. The answers can be both external and internal.

JMSML - XML-Based Mark-Up Language for BEA WebLogic JMS/JMX Programming, Monitoring, and Testing
by Kathiravan Sengodovan
JMSML is a mark-up language designed and developed to make Java Message Service (JMS) and Java Management Extensions (JMX) programming easy by hiding all the JMS and JMX Java API complexity behind a few easy-to-use XML tags.

BEA WebLogic DEVELOPER'S JOURNAL

News & Developments

BEA Extends Product Portfolio

(Orlando, FL) – During BEA eWorld 2003, its eighth annual technology conference, BEA Systems, Inc., the world's leading application infrastructure software company, unveiled new products and services designed to meet the main challenge facing enterprises today – business integration. BEA WebLogic Platform 8.1 provides custom application development and integration into a single infrastructure.

The new application platform suite enables customers to leverage their existing investments while bringing in new technologies that are designed to insure against rapid obsolescence. This approach offers enterprises a unified and simplified way to build, integrate, extend, and deploy enterprise applications and Web services. The benefit is an industrial-strength software platform that is easier to use and that delivers much faster time to value. www.bea.com

HP Makes Strategic Investments in Web Services Management

(Orlando, FL) – At the BEA eWorld user conference HP Chairman and Chief Executive Officer Carly Fiorina outlined several HP investments and technologies to help customers address key management issues that may inhibit the adoption of Web services. Fiorina focused on the critical role Web services management will play in overcoming IT complexity, unifying systems, and accelerating a return on IT investments.

HP will take steps to overcome management barriers to full-scale Web services adoption, including creating a dedicated Web Services Management team within the HP OpenView business unit; a dedicated Web services management and deployment practice for J2EE environments within HP Services; an expanded HP OpenView software suite that will help customers manage and optimize all industry-leading application servers and Web services environments; and plans to make a major contribution to standards efforts, allowing companies to incorporate management capabilities in Web services, rather than take a "build later" approach. www.hp.com.

NEON Systems' Shadow Supports WebLogic Workshop, Liquid Data

(Orlando, FL) – NEON Systems, Inc., has announced support for BEA WebLogic Workshop and BEA Liquid Data for WebLogic. The combination of Shadow technology with BEA WebLogic Workshop and Liquid Data for WebLogic will enable developers to assemble composite applications that integrate mainframe resources using drag-and-drop techniques, while masking mainframe complexity and reducing the need for advanced Java programming skills. www.neonsys.com

Wily Integrates Introscope 4.0 with WebLogic Server 8.1

(Orlando, FL) – Wily Technology has announced the direct integration of Introscope 4.0, the newest version of their comprehensive Enterprise Java Application Management solution, with BEA WebLogic Server 8.1.

Introscope looks into the entire application environment, allowing IT teams to identify performance problems whether inside the application itself, the application server, or in back-end transactional and supporting systems. www.wilytech.com

Cyanea Systems Provides App Management for WebLogic Server

(Orlando, FL) – Cyanea Systems Corp. has integrated its flagship solution, Cyanea/One, with BEA WebLogic Server to provide comprehensive monitoring and management of large-scale applications.

Cyanea develops advanced monitoring and management solutions for Web-based applications, providing application developers, test personnel, and data center managers with the ability to rapidly identify and resolve problems, and minimize downtime of critical business applications. www.cyanea.com

BEA Announces Factory Integration Using Sun Microsystems' Program

(Orlando, FL) – BEA Systems, Inc., has announced factory integration of BEA products through Sun Microsystems' Customer Ready Systems (Sun CRS) program, demonstrating yet another advantage of best-in-class solutions by way of a global alliance. Through the Sun CRS program, customers can get a complete BEA/Sun solution, integrated in Sun factories and built to customers' specifications. Use of Sun CRS is intended to speed time to ROI, maximize productivity, and help lower total cost of ownership for BEA application infrastructure solutions. www.sun.com

SandCherry Offers Speech, Multimodal Capabilities

(Orlando, FL) – SandCherry, Inc., will partner with BEA Systems to offer speech and multimodal capabilities to businesses that already rely on BEA WebLogic Server and BEA WebLogic Portal for customer service, workforce management, content, and messaging applications. The complementary solution is designed to provide a reliable, high-performance platform to run Web applications, service elements to the media components on the SandCherry SoftServer platform, and integrate with back-office systems and databases.

SandCherry's SoftServer platform provides applications with dynamic, reliable, managed access to a variety of speech recognition, text-to-speech, and announcement components from leading vendors to deliver the audio components for voice and multimodal applications. www.sandcherry.com

SEAGULL Joins BEA Star Partner Program

(Orlando, FL) – SEAGULL has joined BEA Systems' Star Partner Program. SEAGULL's LegaSuite software solutions complement the BEA WebLogic Enterprise Platform by automating and accelerating the seamless integration of legacy business processes into new applications. www.seagullsw.com

Dirig Software

www.dirig.com/illusion/weblogic