SYS-CON

AROUND THE WEBLOGIC WORLD IN 80 BLOGS PG 46

MAY/JUNE 2005 VOL:4 ISSUE:3

wldj™

THE LEADING INDEPENDENT MAGAZINE FOR WEBLOGIC™ PROFESSIONALS

WWW.WLDJ.COM

Java Messaging Services Clustering Part 2 ...28

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

Parasoft SOAPtest®

Keep Your Skills Ahead of the Crowd

Keeping your IT skills ahead of the crowd is not as difficult as most people fear. Staying on top of the trends may seem like a daunting task if, like most people, you assume that each new technology is a completely new invention that you must learn from the ground up. Fortunately, nothing is really all that new. Inventors typically create new technologies by studying existing technologies, then building upon them in ways that extend and improve them. 100% new technological advancements are very rare.

Inventors almost always leverage legacy technologies as they invent new ones. Why not leverage your own knowledge of those legacy technologies as you try to learn about the new inventions? To learn about new technologies as painlessly as possible, consider how each new advancement is similar to what you already know.

For example, consider Web services. Web services are a new trend, but — at a technological level — the parts of a Web service are not all that unique. Web services are based on remote procedural calls — messages sent to a server, which calls the requested function. RPCs were developed years ago, and are hardly a new concept. Really, the only "new" thing in Web services is the standard that is being used to write the application. If you break down Web services in this way, it's easy to learn about them. To continue with this process, you might next explore the payload requirements, the process for determining what function to call, and how the call works. As you can imagine, it's a lot more efficient — and interesting — to learn about a new technology based on its relation to familiar technologies than to learn about it by reading the specification cover to cover.

As always, the devil is in the details. But most details are critical only if you want to specialize in a given technology. For instance, if you want to specialize in Web services, you need to familiarize yourself with the details of Web service development. In that case, your next step would be to learn how to format the messages, how to expose Web services, and so on.

– Adam Kolawa, *Ph.D.*
Chairman/CEO of Parasoft

It's automated. It's fast. And it's the most versatile Web Services testing tool.

SOAPtest® enables you to deliver better Web Services in less time.

The simple fact is, no other Web service testing product can do what SOAPtest can do. From verifying functionality and performance to ensuring security and interoperability, SOAPtest automates all critical testing processes across the entire lifecycle of your Web service.

But don't take our word for it.... Go to www.parasoft.com/soaptest_WSD and try it for free — no commitment, no obligation. If you like it (and we suspect you will) just let us know. We'll be more than happy to help you and your development team get up and running.

For Downloads go to www.parasoft.com/soaptest_WSD PARASOFT®

Email: soaptest@parasoft.com Call: 888-305-0041 x1209

Features

- WSDL schema verification and compliance to standards
- Automatic test creation using WSDL and HTTP Traffic
- Data-driven testing through data sources (Excel, CSV, Database Queries, etc)
- Scenario-based testing through XML Data Bank and Test Suite Logic
- Flexible scripting with Java, JavaScript, Python
- WS-I Conformance: Basic Profile 1.0
- WS-Security, SAML, Username Token, X.509, XML Encryption, and XML Signature support
- WS-Security Interop testing emulator
- MIME Attachment support
- Asynchronous Testing: JMS, Parlay (X), SCP, and WS-Addressing support
- Windows Perfmon, SNMP, and JMX monitors
- Detailed Report generation in HTML, XML and Text formats
- Real-Time graphs and charts

Benefits

- Uniform test suites can be rolled over from unit testing to functional testing to load testing
- Prevent errors, pinpoint weaknesses, and stress test long before deployment
- Ensure the reliability, quality, security and interoperability of your Web service
- Verify data integrity and server/client functionality
- Identify server capabilities under stress and load
- Accelerate time to market

Protocol Support

- HTTP 1.0
- HTTP 1.1 w/Keep-Alive Connection
- HTTPS
- TCP/IP
- JMS

Platforms

- Windows 2000/XP
- Linux
- Solaris

Contact Info:

Parasoft Corporation
101 E. Huntington Dr., 2nd Flr.
Monrovia, CA 91016

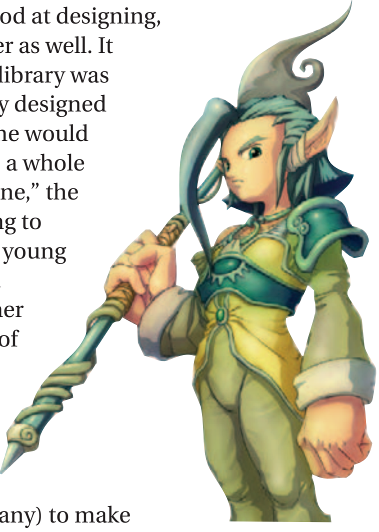
www.parasoft.com

Framework Tales

By Joe Mitchko

Once upon a time, on a project not too far away, a bright young software architect had a thought. “Why, things are getting a bit complex here,” he said. “Perhaps I can make things easier by developing a common framework that can take care of a lot of the drudgery involved in developing software.” So this architect got to work right away to develop the framework, perhaps gathering requirements here and there and dipping into the standard pile of available software patterns for appropriate design ideas. Over time the framework came into being, and the architect considered it to be good. And in fact it was good. From a design perspective, this young architect had thought of things no one else had thought of before, and would proudly describe its inner workings to anyone who had the time to listen.

Not only was our architect good at designing, he was a pretty good programmer as well. It was not long before a prototype library was available based on his beautifully designed framework. “Now, if only everyone would use my framework, life would be a whole lot easier around here for everyone,” the architect thought. So, not wanting to keep his creation to himself, our young architect placed his creation in a public place, and cobbled together documentation, an explanation of how great his creation was, and how you would use it on your own project.



Over time, our young architect became busier and busier, and had less time (if any) to make modifications and improvements to his creation, but was still optimistic about its potential. However there was a problem. Even though there was some interest in his framework, people weren’t ready to incorporate it into their application. “It’s too complex,” some would say, after reading through its technical architecture and not really trusting that it could do all that it claims. “I’m looking for something simpler.” Others looked at it, and unwilling to change, went and developed their application using traditional approaches. Still others looked at it, and not wanting to trust their project to outside developers, walked away – only to develop their own version of the framework.

As word spread about the existence of the framework, so too did various opinions develop that were shared among developers. “Why, I heard it was difficult to use,” was one such discussion overheard by the jammed LaserJet printer. “Why, I understand that it has poor performance characteristics,” was the conversation overheard at the coffee machine. In fact, these people had never really looked at the framework at all, but were just repeating what they heard from others.

Disheartened, our young architect was ready to move on to other things, and left his creation for others to worry about. Those who did use the framework quickly discovered that there was no support available for the library, and were forced into maintaining the code themselves. Furthermore, it wasn’t long before there were several versions of the framework, all with their own extensions.

I tell this story because I’ve seen the plot develop in real life too many times, both in large and small scale initiatives. Often, the greatest challenge in our industry is not technical at all, but revolves around human psychology and getting everyone on board with a particular new idea. Perhaps this is why certain new endeavors, including new state of the art software development products, do not succeed as well as they technically should have – and eventually end up in the hands of the open source community.

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

Contact: joe@sys-con.com

wldj THE LEADING INDEPENDENT MAGAZINE FOR WEBLOGIC™ PROFESSIONALS

EDITORIAL ADVISORY BOARD
Lewis Cirne, Wayne Lesley Lund,
Chris Peltz, Sean Rhody

EDITORIAL

Founding Editor
Peter Zadrozny

Editor-in-Chief
Joe Mitchko joe@sys-con.com

Managing Editor
Seta Papazian seta@sys-on.com

Editor
Nancy Valentine nancy@sys-con.com

Research Editor
Bahadir Karuv, PhD bahadir@sys-con.com

PRODUCTION

Production Consultant
Jim Morgan jim@sys-con.com

Lead Designer
Abraham Addo abraham@sys-con.com

Art Director
Alex Botero alex@sys-con.com

Associate Art Directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com
Andrea Boden andrea@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Andy Armstrong, Raffi Basmajian, Peter Holditch, Balamurali Kothandaraman, Bahar Limaye, Alex Maclinovsky, Joe Mitchko, Eric Newcomer, Blair Taylor, WLDJ News Desk

EDITORIAL OFFICES

SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9638
WLDJ (ISSN #1535-9581)
is published bimonthly (6 times a year)
by SYS-CON Publications, Inc.

Postmaster: send address changes to:
WLDJ: SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
FOR LIST RENTAL INFORMATION:
Kevin Collopy: 845 731-2684,
kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832,
frank.cipolla@epostdirect.com

REPRINTS

For promotional reprints, contact reprint coordinator Dorothy Gil dorothy@sys-con.com. SYS-CON Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.

Application: Operations Monitor

Best Practices for Exchanging Valid Data Through Web Services

By Andy Armstrong

A MODEL-BASED APPROACH TO SEMANTIC INTEROPERABILITY

Web services fit nicely into a service-oriented architecture (SOA) and appear to be one component of the ideal solution to business application integration. When exposing applications through Web services, the contract of each service is evident through its Web Services Description Language (WSDL) description.

The WSDL document describes the names of the operations, the names and types of each of the parameters, and the name and type of the return value. After you add reliable messaging support (to ensure delivery) and a business process management engine (to route based on high-level business logic), your integration problems are over.

Of course it's not so simple. Even when using the loose coupling afforded by Web services, you must marshal data into the correct format. Organizations that do not carefully plan their integration strategy usually end up with a multiplicity of point-to-point integrations, because the simplest way to expose a Web service is through a thin veneer on application data structures. A single point-to-point integration is manageable; dozens become a maintenance nightmare. Semantic inconsistency arises between disparate systems. Exposing application data as a Web service without considering the consumers of the data defeats the loose-coupling tenet of an SOA. Fortunately, a set of best practices is emerging to address these concerns. The essence of these practices is to separate data logic from business logic, capturing the data logic in a model with accompanying rules and transformations that are preserved as metadata. This model-based approach to integration promotes reuse, error management, and controlled change.

Use Data Models

The first step in improving integration is to use an application-agnostic data model. (See Jim Gabriel's *Best Practices in Integrating Data Models for SOA*.) The idea here is to transform data into a common, application-neutral format. (This format can be virtual – it may exist in the development environment only to be compiled away at run time.) Then each data type in the integration project should be mapped onto this format. Adding a new data type requires a single mapping to the neutral format rather than an unmanageably large number of mappings to all of the other types in the system. This can provide maximum flexibility in replacing individual systems without forcing you to rewrite the integration. When you take this approach, you commit to modeling the inputs and outputs of all of the components in the integration project along with the transformations between them. The tools you select will capture all of those relationships as metadata, preserving the intent for future generations of programmers who must maintain the software. Because schemas change over time, many tools provide capabilities for managing schema and interface evolution in a natural way.

Resolve Semantic Inconsistency

Adopting an application-agnostic data model is the starting point, but it's not sufficient. The next step is to ensure semantic interoperability. Enterprises often have multiple definitions for customer, product, or order, as well as business requirements

Author Bio:

Andy Armstrong is a software architect at Pantero Corporation. His responsibilities include the architecture and the overall user experience of Pantero software.

Contact:

Andy.Armstrong@pantero.com

PRESIDENT & CEO

Fuat Kircaali fuat@sys-con.com

VP, BUSINESS DEVELOPMENT

Grisha Davida grisha@sys-con.com

GROUP PUBLISHER

Jeremy Geelan jeremy@sys-con.com

ADVERTISING

Senior VP, Sales & Marketing
Carmen Gonzalez carmen@sys-con.com

VP, Sales & Marketing
Miles Silverman miles@sys-con.com

Advertising Director
Robyn Forma robyn@sys-con.com

Advertising Manager
Megan Mussa megan@sys-con.com

Advertising Sales Manager
Dennis Leavey dennis@sys-con.com

Associate Sales Managers
Dorothy Gil dorothy@sys-con.com
Kim Hughes kim@sys-con.com

SYS-CON EVENTS

President, Events
Grisha Davida grisha@sys-con.com

National Sales Manager
Jim Hanchrow jimh@sys-con.com

CUSTOMER RELATIONS

Circulation Service Coordinators
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com

Manager, JDJ Store
Brunilda Staropoli bruni@sys-con.com

SYS-CON.COM

VP, Information Systems
Robert Diamond robert@sys-con.com

Web Designers
Stephen Kilmurray stephen@sys-con.com
Percy Yip percy@sys-con.com
Vincent Santaiti vincent@sys-con.com

Online Editor
Roger Strukhoff roger@sys-con.com

ACCOUNTING

Financial Analyst
Joan LaRose joan@sys-con.com

Accounts Payable
Betty White betty@sys-con.com

Credits & Accounts Receivable
Gail Naples gailn@sys-con.com

SUBSCRIPTIONS

subscribe@sys-con.com
Call 1-888-303-5282
For subscriptions and requests for bulk orders,
please send your letters to Subscription Department
Cover Price: $8.99/issue
Domestic: $149/yr (12 issues)
Canada/Mexico: $169/yr   All other countries $179/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. (plus S&H)

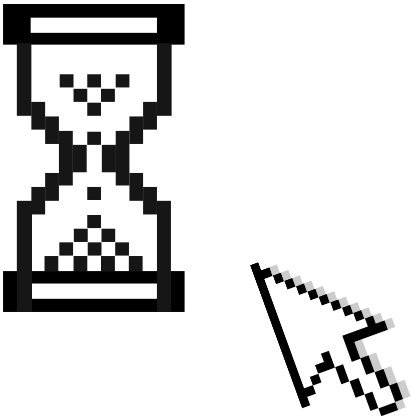
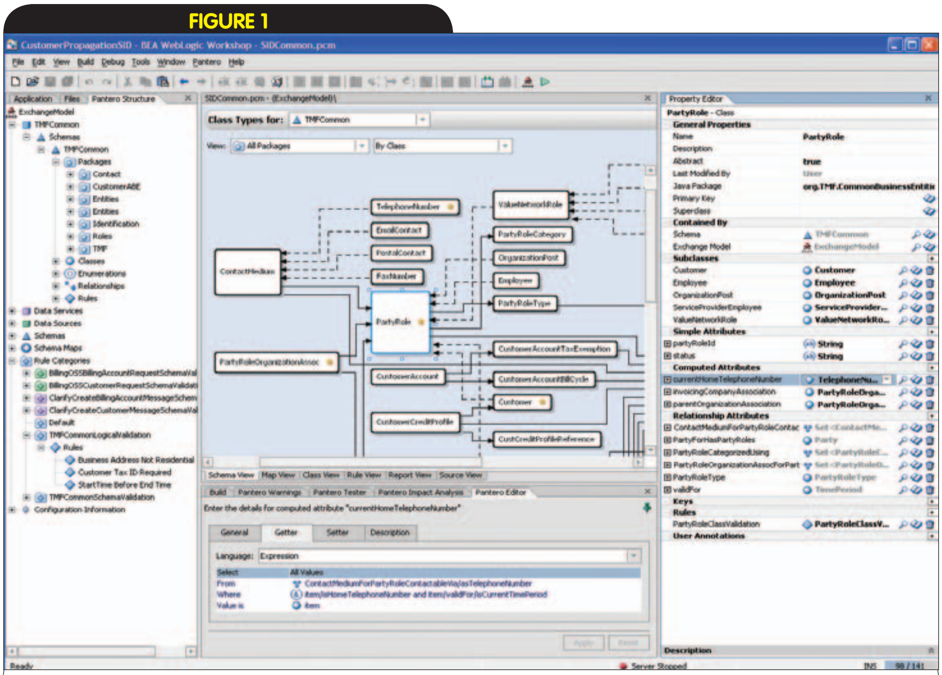


FIGURE 1


CustomerPropagationSID - BEA WebLogic Workshop - SIDCommon.pcm
File Edit View Build Debug Tools Window Pantero Help
Application Files Pantero Structure
ExchangeModel
TMFCommon
Schemas
TMFCommon
Packages
Contact
CustomerABE
Entities
Entities
Identification
Roles
TMF
Classes
Enumerations
Relationships
Rules
Data Services
Data Sources
Schemas
Schema Maps
Rule Categories
BillingOSSBillingAccountRequestSchemaVal
BillingOSSCustomerRequestSchemaValidati
ClarifyCreateBillingAccountMessageSchem
ClarifyCreateCustomerMessageSchemaVal
Default
TMFCommonLogicalValidation
Rules
Business Address Not Residential
Customer Tax ID Required
StartTime Before End Time
TMFCommonSchemaValidation
Configuration Information
SIDCommon.pcm - (ExchangeModel)\
Class Types for: TMFCommon
View: All Packages
By Class
TelephoneNumber
EmailContact
PostalContact
FaxNumber
ContactMedium
PartyRole
PartyRoleOrganizationAssoc
CustomerAccount
CustomerCreditProfile
ValueNetworkRole
PartyRoleCategory
OrganizationPost
Employee
PartyRoleType
CustomerAccountTaxExemption
CustomerAccountBillCycle
Customer
CustCreditProfileReference
Schema View Map View Class View Rule View Report View Source View
Build Pantero Warnings Pantero Tester Pantero Impact Analysis Pantero Editor
General Getter Setter Description
Apply Reset
Property Editor
Description
Ready
Server Stopped
INS 98 / 141

Enter the details for computed attribute "currentHomeTelephoneNumber"

Language: Expression

Select	All Values
From	ContactMediumForPartyRoleContactableViaIsTelephoneNumber
Where	item/isHomeTelephoneNumber and item/validFor/isCurrentTimePeriod
Value is	item

PartyRole - Class

General Properties	
Name	PartyRole
Description	
Abstract	true
Last Modified By	User
Java Package	org.TMF.CommonBusinessEntitie
Primary Key	
Superclass	
Contained By	
Schema	TMFCommon
Exchange Model	ExchangeModel
Subclasses	
Customer	Customer
Employee	Employee
OrganizationPost	OrganizationPost
ServiceProviderEmployee	ServiceProvider...
ValueNetworkRole	ValueNetworkRo...
Simple Attributes	
partyRoleId	String
status	String
Computed Attributes	
currentHomeTelephoneNumber	TelephoneNu...
invoicingCompanyAssociation	PartyRoleOrga...
parentOrganizationAssociation	PartyRoleOrga...
Relationship Attributes	
ContactMediumForPartyRoleContac	Set <ContactMe...
PartyForHasPartyRoles	Party
PartyRoleCategorizedUsing	Set <PartyRoleC...
PartyRoleOrganizationAssocForPart	Set <PartyRoleO...
PartyRoleType	PartyRoleType
validFor	TimePeriod
Keys	
Rules	
PartyRoleClassValidation	PartyRoleClassV...
User Annotations	

Pantero software defines rules and computations graphically within exchange models

TABLE 1

Rule Type	Description
Comparative constraints	Two values share a comparative rule. "BirthDate must be earlier than DeathDate."
Valid enumeration values	The message format is mapped to values from internal systems. "BankName must be one of Bank of America, Citibank, or Wells Fargo."
Multifield validation checks	One field affects the rules on another. "DependentCount must match the number of Dependent elements in the message."
Derived properties	One value is calculated from data within the message and other data in enterprise systems. "LoanToValueRatio."
Merging content from multiple data sources	Multiple systems have different values for the same field. "Main Street" vs. "Main St."
Dynamic constraints	A constraint is determined from an external source. "Do not ship if the account is delinquent."
Definition resolution	Definitions are mapped to a higher level of abstraction. "OEM and Distributor equals Customer."

Examples of rules that cannot be expressed in XML Schema

to present data about these entities differently according to rules and roles within the business. To take a real-world example, consider the task of integrating billing and order-entry systems at a major telecommunications provider. The billing system has specific rules about what constitutes a valid request. For example, you aren't allowed to use the "residential" bill format for a commercial account. These rules aren't known to the order-entry system, and they're not expressible in the schema for an order. Simply exposing "order" as a parameter to a Web service does not obviate the underlying constraint. In fact, exposing this concept via a Web service increases the likelihood that inconsistent or incorrect data will be supplied, with little feedback provided to the suppliers of the bad data. A properly written integration project must take explicit steps to resolve these inconsistencies.

Separate Data Logic from Business Logic

These sorts of inconsistencies are all too often swept under the rug or addressed outside the scope of the integration project. A far-too-common approach is to change human processes to avoid entering "bad" data into the system in the first place. One customer concluded that it would be easier to train the data-entry staff to never enter a purchase order containing incompatible feature combinations than it would be to fix the integration to detect and correct this situation. These errors are typically addressed in one of two ways: by writing custom Java code in the individual service adapters or by adding decision nodes to the business process definition (in WebLogic, a Java Process Definition, or JPD). New business process decision nodes mean interspersing low-level data integration logic with the high-level application logic, resulting in a business process flow that is no longer understandable to the business user for whom it is intended. Custom Java code generally works well but results in a fragile integration. The Java code is tied to a specific version of the data schema – as soon as the schema changes, you must locate and revisit the Java code. Data logic should be tightly coupled with the data definition, not distributed throughout the integration project.

Use Model-Based Integration

An emerging best practice to address these concerns is to attach the rules relating to the data (validity constraints, logical calculations) directly to the model as metadata. In this case, the application-agnostic data model mentioned earlier includes not only the definitions of the data types of the various business applications, but also the custom reconciliation rules you must define to ensure semantic interoperability. This model-based approach results in comprehensive metadata for the entire integration project and helps you manage extensions to the models and rules over time. Model-based integration projects are self-documenting and significantly reduce maintenance costs. For example, you can develop validation rules and mapping transformations graphically in conjunction with application owners instead of using the commonly accepted but slow and error-prone practice of e-mailing spreadsheets across organization boundaries.

Reuse Common Computations

A model-based integration starts with definitions of the systems involved. Some organizations have these definitions defined in XML Schema, some in Unified Modeling Language (UML), some as object-relational mappings. In any case, these definitions are incorporated into a larger model that defines each of the payloads and the relationships between them. This model may be enriched with additional data that is not present in the original schemas but spans multiple systems. For example, an enterprise may have a well-defined delineation of "loan-to-value ratio," which is critical to its notion of mortgage data exchange. This calculation should be defined once and reused throughout the set of integration projects. Ideally this calculation would be attached directly to the model, resulting in what is often called an exchange model. (See Roy Schulte's *Information Architecture for the Virtual Enterprise*.)

With sufficient tool support, you can define these computations graphically, often without writing any Java code. Figure 1 shows how Pantero software uses graphically built expressions to define a method for deriving a home telephone number that combines data from several classes in a complex data model. Because this method is defined in the common data model, it is available to all services that are mapped to the model. (See the sidebar for further explanation of this example.)

Define Validation Rules Beyond XML Schema

Although payload specifications are often expressed as XML Schema, a model-based integration project contains rules that cannot be expressed in XML Schema. (See Table 1 for examples.) Virtually every industry schema is accompanied by a document describing constraints that must be enforced outside the schema itself. For example, the mortgage industry's MISMO schema is accompanied by a document that explains that a cardinality constraint must be enforced across DependentCount and Dependent; the MISMO DTD defines a sequence of zero or more Dependent elements, and their number must match DependentCount. These constraints are best defined on the exchange model itself, where they are enforced across all systems that integrate with it. Virtually every industry schema is abstract, to allow for extensibility, whereas the systems to be integrated have concrete models. The exchange model can serve as the place where the abstract-to-concrete logic takes place, again ensuring consistency and reuse across all consumers. (See the sidebar for a real-world example.)

Develop Error-Management Models

Most integration projects handle errors in the orchestration tool (typically the business process management engine). As with custom code, to perform semantic conversions this error management is heavily dependent upon the data types involved without being specifically coupled to them. Again this leads to fragile implementations. A better approach is to model the errors as well, tying them to specific validation or integrity constraints. This method preserves separation between business logic and data logic. You can introduce algorithms for correcting improper data without changing the flow of the high-level business processes. Most important, an error-management model can improve the user experience, letting you plan recovery paths and avoid dead ends with no alternative to restarting transactions from the beginning.

Benefits from Exchange Models

An exchange model comprises the schemas, mappings, rules, and computations you develop as you follow the practices outlined in this article. Using an exchange model for integration projects has significant benefits.

- Using a common data model provides *the flexibility to add, remove, or change individual systems* without requiring changes to other systems or how those systems interact in the integration.
- Disentangling business and data logic allows *reuse of business processes by keeping the business process* focused on business, even as data definitions change.
- Defining common computations and rules in one model *promotes reuse of data-validation* logic and its deployment as a single custom control that multiple systems can invoke.
- An error model provides means of *communicating and repairing data errors* without disrupting the general flow of the business process.
- Capturing an exchange model as metadata enables *analysis of the impact and prediction of the cost of proposed modifications*, resulting in better management and control of change.

Resources

- Gabriel, J. (2005). "Best Practices in Integrating Data Models for SOA." *Web Services Journal*. SYS-CON Publications, Inc. Vol. 5, issue 2.
- Roy W. Schulte, *Information Architecture for the Virtual Enterprise*, Gartner Research ID Number COM-17-1884, July 31, 2002 www.gartner.com/DisplayDocument?id=365491

Integration and the TM Forum SID

The TeleManagement Forum (www.tmforum.org) has developed the Shared Information/Data (SID) model as a common vocabulary to bridge the business and systems perspectives in its Next Generation OSS (NGOSS) framework. The SID uses UML to formalize the expression of the needs of a particular view.

Using the SID for integration projects is complicated by the fact that it is a highly abstract model. A "Customer" doesn't have a "Home Telephone Number." Instead, a Customer has a collection of ContactMediums. A ContactMedium contains "effective date" information. Some ContactMediums are telephone numbers. Telephone numbers have type classification. So, a Java program that would like to compare a number supplied in an XML message with a customer's home telephone number implements the following pseudocode:

```
1. getCustomer.getContactMediums()
2. iterate through all returned ContactMediums
3. for each ContactMedium that is instanceOf
     telephoneNumber
4. if the following conditions are met:
   4a. today() is between validFor.startDateTime
       and validFor.endDateTime
   4b. cast ContactMedium to telephoneNumber
   4c. telephoneNumber.type is HomeNumber
5. then map the telephoneNumber.number to
     home_telephone_number
```

Writing this code results in a fragile integration project. What happens when the model changes subtly? The code may no longer work. However because the Java code is completely disjointed from the underlying data model, you have to seek it out (rediscover the dependency) in order to reassess the code.

Users of a rich integration modeling environment attach the "home telephone number" concept directly to the exchange model as metadata, coupling the logic with the data specification and preserving the definition through subsequent refinements to the base model. If the names or types of attributes change, the modeling tool automatically updates this calculation as well. In addition, because the code to traverse the model is generated rather than coded by hand, it is more likely to be correct. Figure 1 illustrates such an implementation of the "home telephone number" algorithm within a data model based on the SID, using Pantero software's graphical expression capabilities.

Extending the J2EE Deployment API for Disruption-Free Service

WEBLOGIC SERVER GOES ABOVE AND BEYOND THE SPECIFICATION

By Balamurali Kothandaraman

J2EE has been very successful in providing the specifications for detailing the arrangement of resources in a server-side application such as a:

- Web application (*.war file)
- EJB application (*.jar file)
- Enterprise application (*.ear file)

This in itself has helped make J2EE popular in the enterprise. The next step is to ease the process of deploying applications across various environments by providing a uniform deployment API supported by different J2EE product vendors. Along with the deployment API, there needs to be a uniform management API supported by different J2EE product vendors to manage the deployed application in production.

It is common for a J2EE application server vendor to offer some deployment tools along with the application server distribution. In fact, administrators depend on the application deployment and application configuration tools they provide. Creating scripts or automating the deployment process involves implementing vendor-specific APIs. Standardizing this aspect will simplify the development of deployment and application configuration tools. There would only be one package to implement.

J2EE Deployment API

This specification defines standard APIs that enable any deployment tool that uses the Deployment API to deploy any assembled application onto a J2EE-compatible platform.

The API addresses the three-stage deployment process:

- *Installation:* Moves the properly packaged components to the server
- *Configuration:* The resolution of all external dependencies declared by the application
- *Undeployment:* Removes the application from the server

The Need for Extensions

The J2EE Deployment API in its current 1.1 version as part of the J2EE 1.4 specification pretty much achieved this goal, but it can be enhanced in certain areas to provide more practical features. For example, the redeployment process isn't detailed and the maintenance of the HTTP session objects during application updates is omitted. Versioning of applications isn't part of the Deployment API either.

Live application upgrade is one of the main factors in achieving disruption-free service. Even though the failover feature provided by WebLogic Cluster will increase the availability of an application, a live application update is vital to providing continual uptime of the service to clients.

In this article we'll discuss some of the specification enhancements needed. To illustrate, we'll detail them on Diablo, the upcoming WebLogic Server 9.0.

WebLogic Server Extensions to the J2EE Deployment API

Exporting Application to Deploy Across Multiple Environments

One of the biggest challenges for an administrator of a J2EE deployment platform is to successfully transition the application from the development environment to the testing and production environments. Along with the application to be deployed developers need to transfer the knowledge about the application's dependent resources. It's always a challenge for application deployers and system administrators to resolve such dependencies for each application when transitioning the application from a development to a live environment.

The J2EE runtime environment or container provided by an application server vendor provides many infrastructure services to the application that can be configured with the deployment descriptors. J2EE also provides a way to abstract everything that is not constant in an application by configuring them in the deployment descriptors rather than hard coding them in the application. Changing the application deployment descriptors doesn't require

Author Bio:
Balamurali Kothandaraman is a delivery technologist for education services at BEA Systems, Inc. He has more than seven years of experience in Java and J2EE technologies and is a BEA Certified Server Specialist, Administrator, Developer, and Instructor. Bala frequently speaks at various conferences including JavaOne, eWorld, and BEA User Conferences.

Contact:
bala@bea.com

any change in the application source.

Having accepted the premise that change is constant in this world, change in an application involves changing deployment descriptors, which involves unpacking the application archive and repackaging it. To take advantage of the vendor-specific application configuration options used for optimizing the performance of the application components in production, the vendor-specific deployment descriptors inside the archive should be modified such as, for example, EJB pool size.

Developers can use weblogic.Configure, a command-line tool provided by BEA, to export an application's configuration. Then they can select the WebLogic-specific deployment descriptor properties that have to change when the application is deployed in another environment such as EJB pool size, EJB cache size, etc. The tool will create a deployment plan with variable definitions for the descriptor properties selected. Later when the application is being deployed using any one of the WebLogic deployment tools, the plan variables are presented to a deployer for customization.

WebLogic-specific deployment descriptor properties are classified as:

- ***Non-configurable:*** Properties that can't be changed by an administrator
- ***Dependency:*** Properties that resolve to a resource configured in the target WebLogic domain (for example, the JDBC Data Source being used by an EJB)
- ***Declaration:*** Properties to declare a resource that other applications will use (for example, the JNDI name of an EJB)
- ***Configurable:*** Properties that aren't configured as Dependency or Declaration will be configurable (WebLogic-specific properties such as an EJB cache size fall into this category)

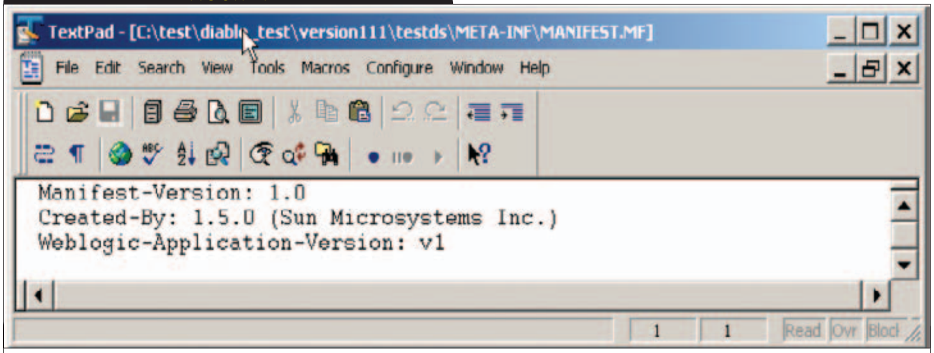
For an entire category for deployment descriptors a template deployment plan can be generated through the command-line tool weblogic. Configure.

```
java weblogic.Configure -root install_root
     [-type (ear|war|jar|car|rar|jms|jdbc)]
     [-export (all|dependencies|declarations|d
ynamics)]
```

Deployment Validation

This feature covers validating the deployment unit and runtime resources needed to deploy. The WebLogic Server Administration Console and WebLogic Workshop tools provide interactive phases of configuration, custom configuration, and pre-deployment validation. While deploying applications with these tools the user will be prompted for an option to configure/deploy and external resources that the application depends on.

FIGURE 1

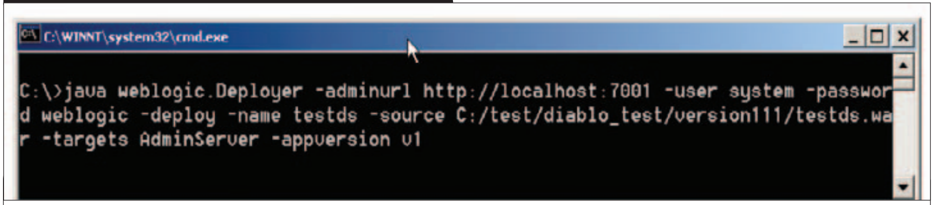


TextPad - [C:\test\diablo_test\version111\testds\META-INF\MANIFEST.MF]
File Edit Search View Tools Macros Configure Window Help
1 1 Read Ovr Block

```
Manifest-Version: 1.0
Created-By: 1.5.0 (Sun Microsystems Inc.)
Weblogic-Application-Version: v1
```

Specifying an application version in the manifest file

FIGURE 2



```
C:\>java weblogic.Deployer -adminurl http://localhost:7001 -user system -passwor
d weblogic -deploy -name testds -source C:/test/diablo_test/version111/testds.wa
r -targets AdminServer -appversion v1
```

Specifying an application version during command-line deployment

For example, if an application depends on a JDBC Data Source the containers will check to see if that Data Source (and its corresponding connection pool) exists. If it doesn't exist, the user will be given the option to configure/deploy that resource. This includes the enhancements that are required in the Deployment API to incorporate the error details needed by the tools to create the missing resources or select suitable alternatives.

Production Redeployment

The J2EE Deployment API outlines a basic redeployment strategy where older versions of the application will be replaced by the newer version. There's no mention about the state of HTTP sessions left in the container. In WebLogic Server 9.0 (Diablo) two redeployment strategies are provided.

PRODUCTION/SIDE-BY-SIDE REDEPLOYMENT

WebLogic Server 9.0 introduces a controlled process for deploying new versions of Web applications without disrupting service. A new version of an application can be deployed alongside an existing version and WebLogic will gradually migrate all of the traffic to the new one. So when redeploying an existing application, the old version can be retired gracefully after all of the existing sessions end. If required, a time-out can also be set to retire the old version.

IN-PLACE REDEPLOYMENT

The new version immediately replaces the old version. This strategy is similar to what the J2EE Deployment API outlines.

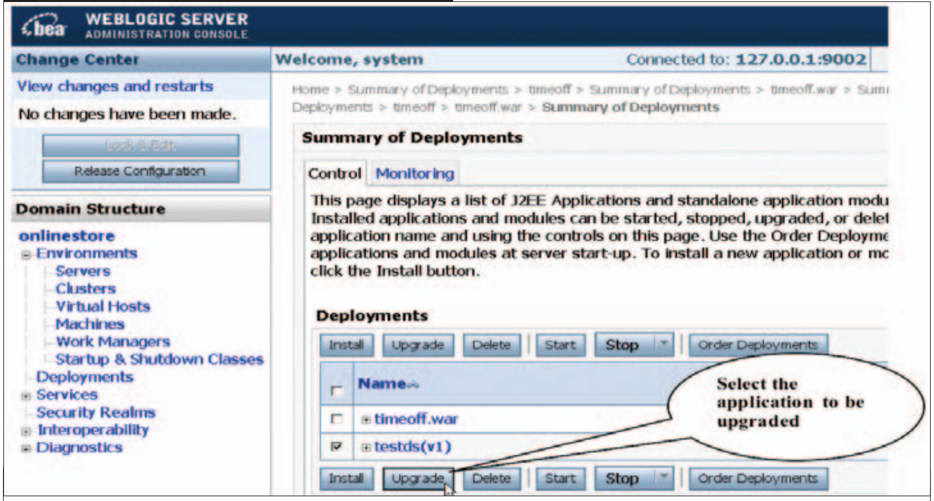
WebLogic Server 9.0 can also roll back deployments.

WebLogic Server will automatically use the production redeployment strategy when the redeployed application version specifies a different version identifier from the currently deployed application version. A developer can assign a version identifier to an application by specifying it in the application's manifest file. The version identifier can also be assigned to an application during deployment or redeployment using the appversion option in the weblogic.Deployer tool.

Production redeployment can be done either through the administration console or through command-line tools such as weblogic.Deployer and WLST.

When using the administration console in

FIGURE 3



No changes have been made.

Connected to: 127.0.0.1:9002

Summary of Deployments

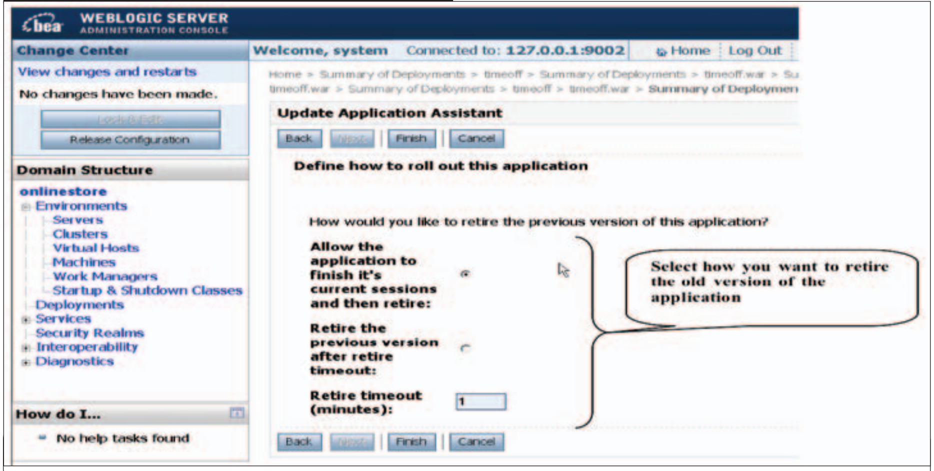
This page displays a list of J2EE Applications and standalone application modu
Installed applications and modules can be started, stopped, upgraded, or delet
application name and using the controls on this page. Use the Order Deployme
applications and modules at server start-up. To install a new application or mc
click the Install button.

Deployments

□	Name
□	timeoff.war
☑	testds(v1)

Upgrade the application

FIGURE 4



No changes have been made.

Connected to: 127.0.0.1:9002

No help tasks found

Update Application Assistant

Define how to roll out this application

How would you like to retire the previous version of this application?

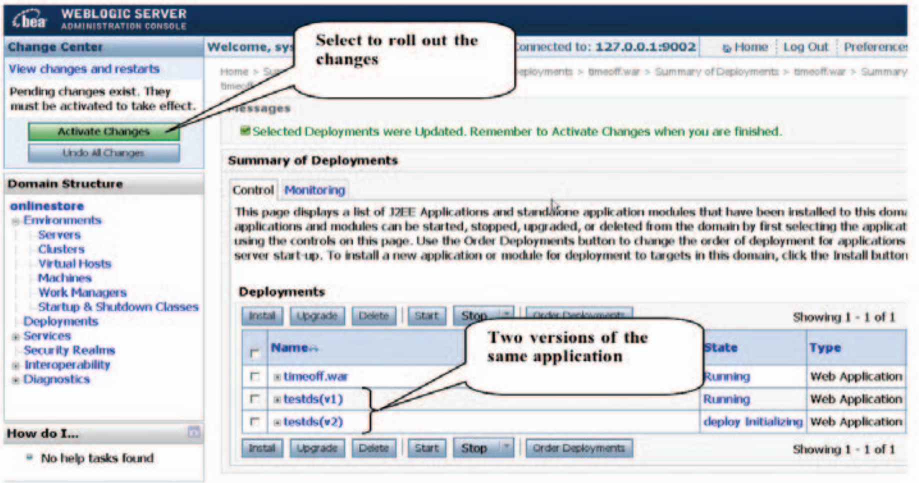
Allow the application to finish it's current sessions and then retire: ●

Retire the previous version after retire timeout: ○

Retire timeout (minutes): 1

Choosing the retirement strategy

FIGURE 5



Pending changes exist. They must be activated to take effect.

onnected to: 127.0.0.1:9002

No help tasks found

Selected Deployments were Updated. Remember to Activate Changes when you are finished.

Summary of Deployments

This page displays a list of J2EE Applications and standalone application modules that have been installed to this dom
applications and modules can be started, stopped, upgraded, or deleted from the domain by first selecting the applicat
using the controls on this page. Use the Order Deployments button to change the order of deployment for applications
server start-up. To install a new application or module for deployment to targets in this domain, click the Install button

Deployments

Showing 1 - 1 of 1

□	Name	State	Type
□	timeoff.war	Running	Web Application
□	testds(v1)	Running	Web Application
□	testds(v2)	deploy Initializing	Web Application

Showing 1 - 1 of 1

Activating the configuration changes

Diablo you start making changes by acquiring a lock over the config.xml configuration repository. The changes are left in a pending state and don't take effect until you activate them. This provides a predictable and secure way for distributing configuration changes in a domain. One has to click on the "Lock and Edit" button on the change center on the console to start deploying applications and finish by clicking the "Activate Changes" button on the change center.

When using the administration console the version identifier has to be configured on the application's manifest file. But the weblogic.Deployer tool can configure on the manifest file or specify using the –appversion option.

The weblogic.Deployer tool can also be used to redeploy a new version of an existing application. The old version can be retired by specifying the –retiretimeout option.

Understanding the Implications of Production Redeployment

By default, WebLogic Server destroys current user sessions when you redeploy a Web app. If you want to preserve the user sessions during redeployment, set "save-sessions-enabled" to "true" in the container-descriptor stanza of the weblogic.xml deployment descriptor file. Note, however, that the application still remains unavailable during the time the in-place redeployment takes place.

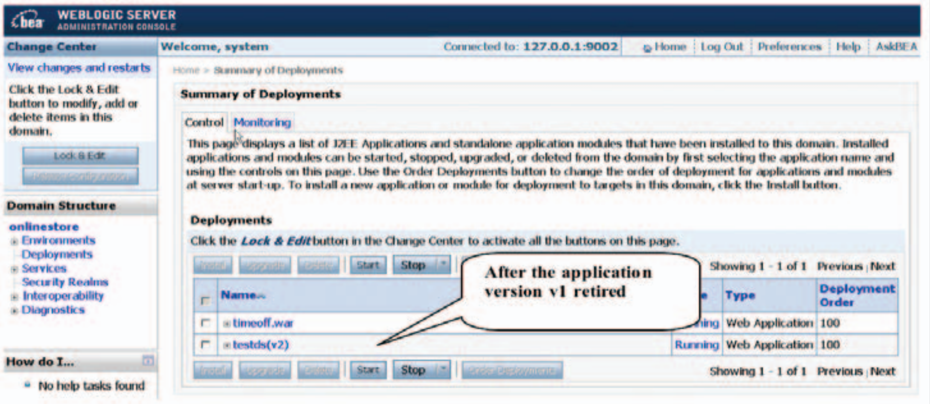
Testing Applications in Admin Mode

New application versions should to be tested before opening them to external clients. WebLogic Server puts an application in a special Administration Mode when you distribute a new version. External clients can't access applications that are in Admin Mode; an application is only available through an Administration Channel. An Administration Channel can be created in a domain by enabling the domain's Administration Port property. The default Administration Port is 9002. The Administration Channel is a secure channel that will accept only SSL connections. Once it is enabled in a domain all the administration traffic must go through that port.

For example, if a Web application (timeoff.war) with the context root "/timeoff" is deployed to a server in a domain with Administration Port 9002, then the

Create software so brilliant
it can manage itself.

FIGURE 6



Connected to: 127.0.0.1:9002

Click the Lock & Edit button to modify, add or delete items in this domain.

No help tasks found

Summary of Deployments

This page displays a list of J2EE Applications and standalone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, upgraded, or deleted from the domain by first selecting the application name and using the controls on this page. Use the Order Deployments button to change the order of deployment for applications and modules at server start-up. To install a new application or module for deployment to targets in this domain, click the Install button.

Deployments

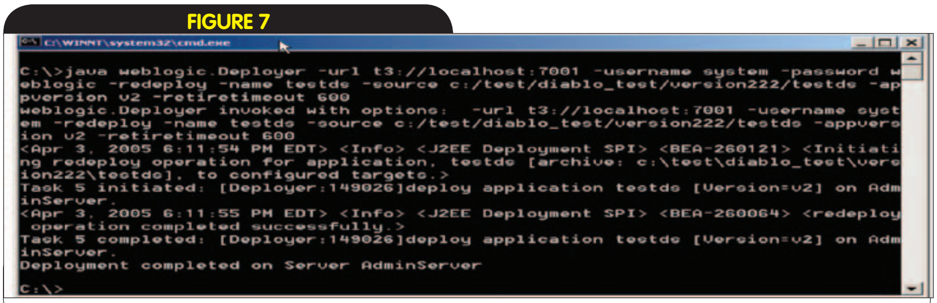
Click the ***Lock & Edit*** button in the Change Center to activate all the buttons on this page.

□	Name	e	Type	Deployment Order
□	timeoff.war	ning	Web Application	100
□	testds(v2)	Running	Web Application	100

After the application version v1 retired

Summary of Deployments

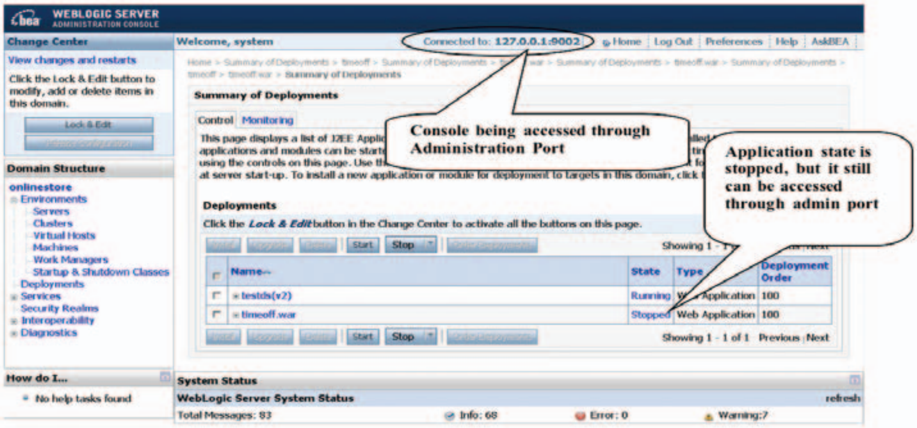
FIGURE 7



```
C:\>java weblogic.Deployer -url t3://localhost:7001 -username system -password w
eblogic -redeploy -name testds -source c:/test/diablo_test/version222/testds -ap
pversion v2 -retiretimeout 600
weblogic.Deployer invoked with options:  -url t3://localhost:7001 -username syst
em -redeploy -name testds -source c:/test/diablo_test/version222/testds -appvers
ion v2 -retiretimeout 600
<Apr 3, 2005 6:11:54 PM EDT> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiati
ng redeploy operation for application, testds [archive: c:\test\diablo_test\vers
ion222\testds], to configured targets.>
Task 5 initiated: [Deployer:149026]deploy application testds [Version=v2] on Adm
inServer.
<Apr 3, 2005 6:11:55 PM EDT> <Info> <J2EE Deployment SPI> <BEA-260064> <redeploy
 operation completed successfully.>
Task 5 completed: [Deployer:149026]deploy application testds [Version=v2] on Adm
inServer.
Deployment completed on Server AdminServer

C:\>
```

Command Line Redeployment

FIGURE 8



Connected to: 127.0.0.1:9002

Click the Lock & Edit button to modify, add or delete items in this domain.

No help tasks found

Summary of Deployments

This page displays a list of J2EE Applic applications and modules can be start using the controls on this page. Use th at server start-up. To install a new application or module for deployment to targets in this domain, click

Deployments

Click the ***Lock & Edit*** button in the Change Center to activate all the buttons on this page.

□	Name	State	Type	Deployment Order
□	testds(v2)	Running	Web Application	100
□	timeoff.war	Stopped	Web Application	100

System Status

WebLogic Server System Status

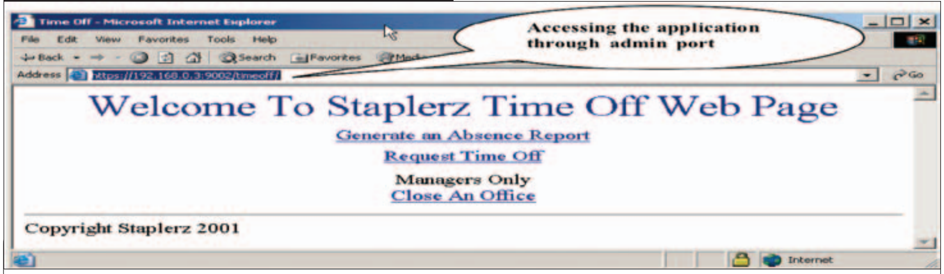
Total Messages: 83 Info: 68 Error: 0 Warning:7

Console being accessed through Administration Port

Application state is stopped, but it still can be accessed through admin port

Accessing the sonsole using the administration port

FIGURE 9


Time Off - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Internet

https://192.168.0.3:9002/timeoff/

Welcome To Staplerz Time Off Web Page

Generate an Absence Report

Request Time Off

Managers Only

Close An Office

Copyright Staplerz 2001

Accessing the application through admin port

Testing the application using administration port

application can be accessed by the URL https://serverhostname:9002/timeoff/.

If the testing yields successful results, the Web application can be started. Then external clients can access it through the regular listen ports configured for the targeted server(s). An application can also be started through tools like weblogic.Deployer or WLST.

Used along with side-by-side deployment or production redeployment, this feature can isolate the new application version. While the old version is still available to external clients without disruption, the new version can be sanity checked.

Module-Level Deployment and Redeployment for Enterprise Applications

WebLogic Server lets you target individual modules of an enterprise application at different server targets or deploy only a subset of available modules of the enterprise application. This provides flexible packaging options so you can bundle a group of related modules together in an enterprise application, but deploy only selected ones on individual servers or clusters in a domain.

This module-level targeting provides fine-grained control in deploying the modules. Module-level targeting can be done using any standard WebLogic deployment tool such as administration console, WLST, or weblogic.Deployer. This can simplify application packaging and distribution by packaging multiple modules in a single distributable EAR, but target only the modules you need to the right servers in a domain.

Module-level deployment using weblogic.Deployer:

```
java weblogic.Deployer -adminurl http://local-
host:7001
-user weblogic -password weblogic
-name mywebapp -targets welcomewebapp@server1
-stage -deploy c:\files\myApp.ear
```

Being able to do a live application upgrade is a major factor in achieving a disruption-free service along with other candidates like disruption-free server upgrade, improved visibility into runtime operations and the application runtime information, and flexible, customizable administration tools. With features in WebLogic Diablo like live server upgrades, diagnostic framework, automatic server migration, clustering, customizable administration console, and powerful scripting tools you can achieve a disruption-free service.

PAID EDITORIAL

Intersperse Manages the Risk in SOA Deployments

Over the past 25 years, enterprise application systems have advanced tremendously, increasing functional scope, geographic distribution, degree of interconnectedness, and speed of information delivery. While this evolution yielded enormous productivity benefits and competitive advantages, it also caused IT complexity to explode, as we've moved through computing phases: from the mainframe, through client/server, into web apps, and finally on to SOA applications. In addition to this increased complexity, each of these phases was shorter than the last, allowing IT teams less room for error (see Figure 1).

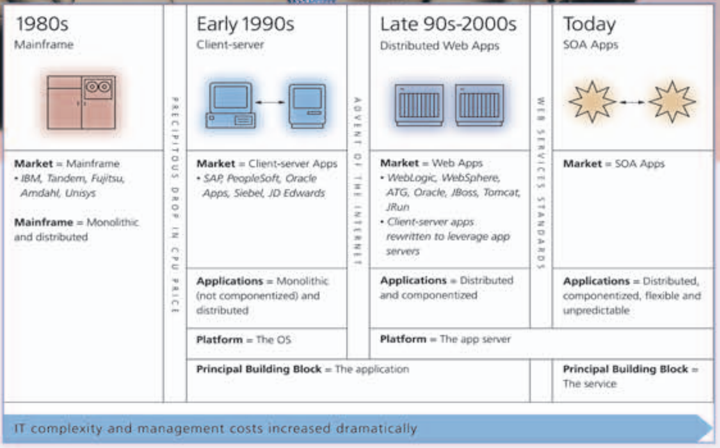
This all results in a relentless increase in risk – risk that has always been mitigated by IT management tools. New management solutions emerged each time we moved into a new phase, providing features and functions targeted to solve the specific needs of that phase.

This remains true today. existing management tools were designed to address the needs of previous generations, and not to address the problems generated by today's advanced computing systems, which are increasingly built on J2EE and based on SOA principles.

Despite their many benefits, today's J2EE-based SOA application systems bring new challenges. Among other things, these systems:

- Are highly distributed, leveraging the Internet to coordinate business processes within and between enterprises.
- Have many more points of failure, as systems break down into granular software components, services, and interfaces, with dynamic, complex interdependencies, and relationships, any of which can cause failure either individually or through its interactions.
- Blur the line between application development and application integration, as systems are increasingly "built to integrate" with an emphasis on service reuse.

The BEA WebLogic Platform, from Workshop to WebLogic Integration, provides the market's most comprehensive toolset to design, build, and deploy SOA applications. The last piece of the production SOA puzzle is comprehensive, proactive management of SOA applications. This is a critical piece, as traditional management tools are insufficient to manage distributed, integrated, cross-application and cross-enterprise business processes – they were designed for another time and another world.



1980s Mainframe	PRECIPITOUS DROP IN CPU PRICE	Early 1990s Client-server	ADVENT OF THE INTERNET	Late 90s-2000s Distributed Web Apps	WEB SERVICES STANDARDS	Today SOA Apps
Market = Mainframe • *IBM, Tandem, Fujitsu, Amdahl, Unisys* **Mainframe** = Monolithic and distributed		**Market** = Client-server Apps • *SAP, PeopleSoft, Oracle Apps, Siebel, JD Edwards*		**Market** = Web Apps • *WebLogic, WebSphere, ATG, Oracle, JBoss, Tomcat, JRun* • *Client-server apps rewritten to leverage app servers*		**Market** = SOA Apps
		Applications = Monolithic (not componentized) and distributed		**Applications** = Distributed and componentized		**Applications** = Distributed, componentized, flexible and unpredictable
		Platform = The OS		**Platform** = The app server		
		Principal Building Block = The application				**Principal Building Block** = The service

IT complexity and management costs increased dramatically

Intersperse Manager is the only product designed specifically to provide production management of SOA applications. It discovers, monitors, and manages J2EE-based SOA applications deployed on Application, Integration, Process, or Portal Servers, and it ensures the continuity of the vital, integrated business processes that run today's enterprises. By simultaneously managing both vertical applications and horizontal integrations, Manager delivers proactive production management of service-oriented BEA applications. Intersperse extends management to the complete BEA WebLogic Platform, including WebLogic Integration and WebLogic Portal.

Intersperse Manager is truly a production management solution, leveraging the increasingly ubiquitous Java Management Extensions (JMX) standard. JMX facilitates standardized discovery, monitoring, and management of components, applications, and services, as well as enabling hot deployment and removal from already running systems. This is in contrast to some tools today that have defaulted to the use of invasive, application-altering byte-code instrumentation as their means to gather management data.

To fully reap the benefits and mitigate the risks of J2EE-based SOA deployments, businesses need a new kind of management tool, one designed specifically to serve the needs of these environments. Intersperse Manager is the only management tool designed to help master the rapidly growing complexity of SOA business systems. It gives developers, operators, and analysts comprehensive visibility into all relevant tiers, tools to proactively monitor and analyze system performance in business contexts, and the control to automatically correct problems in production. As organizations evolve their business systems to BEA-based SOAs, they cannot afford to be without Intersperse Manager.

intersperse.

Implementing a Cluster-Aware Cache Using WebLogic

A LAZILY RECONSTRUCTABLE CACHE SYSTEM THAT NOTIFIES CLUSTER MEMBERS OF DELETES

By Bahar Limaye

Caching information on the WebLogic tier can significantly increase performance and reduce the number of external system calls needed for data retrieval. This is especially true when an application wants to store bits of information that rarely change, such as a list of countries or catalog entries. Nowadays, memory is so cheap that application architectures can benefit from caching data on the Web/EJB tiers.

This is not a new concept; most developers already do this. Whether it's accessing data from a simple java.util.Hashtable or java.util.Hashmap, or using a sophisticated LRU cache mechanism (a derivative of LinkedList), there is minimal overhead in accessing information. The real question isn't how the data is cached or what mechanism is used for storage, but how to preserve the data's integrity and notify cluster members of changes in a highly available, clustered environment.

For example, suppose you're building an e-commerce storefront application that sells clothing. The product information such as the item name, description, SKU, price, and image can be stored in memory on the application tier. The benefit of storing the data in memory is faster page loading and a reduction in database calls.

Now, what happens when your business user wants to change the price or description of the catalog item? How would you notify all of the members of the cluster of this "delta" change? Remember, you probably still want the data to be stored in memory if possible for performance reasons. Also, you probably want the information to be updated in close real-time and don't want to reload all of the data if it hasn't changed.

This article illustrates a simple technique for implementing a "lazily reconstructable" cache system that notifies cluster members of invalidations (deletes).

For example, look at the following code:

```
static Hashtable cache = new Hashtable();

public Product getProduct(String productID) {
Product product = (Product) cache.get(productID);
if (product == null) {
  // get the product from the database
  product = ProductDAO.getProduct(productID);
  cache.put(productID, product);
}
return product;
}
```

A simple *java.util.Hashtable* stores the list of products indexed by the product identifier in a static cache. Products are retrieved from the database and added to the cache lazily if they don't already exist. This code snippet demonstrates how to retrieve a product by looking into cache and grabbing it if it doesn't exist.

What happens if a product description changes for a specific product? Look at this code:

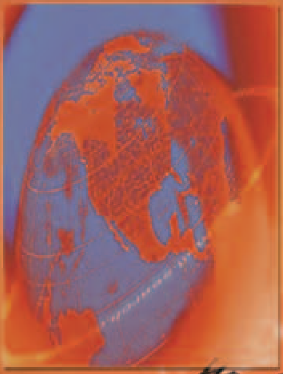
```
public void changeDescription(String productID, String
newDescription) throws
                              ProductNotFoundException {

  ProductDAO.changeDescription(productID, newDescrip-
tion);

  Command command = new InvalidateCacheCommand(productID
);
        ClusterNotifier.notify(command);
}
```

The code behind the *ProductDAO.changeDescription* method updates the product ID in the database with the new description. The *ClusterNotifier.notify* method (see Listing 2) will

Author Bio:
Bahar Limaye is a system architect at The College Board. He has extensive experience building distributed object-oriented systems.

Contact:
blimaye@collegeboard.org

BEA's Workshop can be even more amazing for the phone.



One Application.
Web. Voice. It's Easy.

AppDev™ VXML for BEA's WebLogic™ Workshop

Delivering Web applications to phone users is as easy as clicking a mouse.

For additional information:

SandCherry, Inc.
1715 38th Street
Boulder, CO 80301

+1 (720) 562-4500 Phone
+1 (866) 383-4500 Toll Free
+1 (720) 562-4501 Fax

Info@sandcherry.com
www.sandcherry.com

Download
30 Day Free Trial
www.sandcherry.com

SandCherry™

delete the product from the local cache and send an invalidation event to the cluster members to delete the product from its local cache. Once this is done, all of the members in the cluster will have the product removed from memory. If a user requests this product on any node, it becomes lazily initialized with the new description that was set. See the snippet below from *getProduct()* example:

```
if (product == null) {
  // get the product from the database
  product = ProductDAO.getProduct(productID);
  cache.put(productID, product);
}
```

This will reinitialize the product from the database and insert the information back into cache.

How does this work? Well, let's take a look.

Deep Dive into the Classes

Note: This article uses unpublished private APIs from WebLogic. They work with WebLogic 7.x and 8.x, but there's no guarantee they'll be supported in future versions. Many of the underlying details were "introspected" by putting the WebLogic.jar in the classpath of the IDE.

Command Interface (Generic)

First, let's take a look at the *Command* interface. It's simple. It defines a single method called "execute."

```
public interface Command extends Serializable {
  public void execute();
}
```

The *Command* interface is used to do the work that needs to be done in the cluster. Notice that it is *Serializable*. It contains the logic to do a task when it reaches each cluster member. This interface is generic enough to do any operation. This article discusses how this interface can be used to invalidate keys in a cache across the cluster.

InvalidateCacheCommand Concrete Implementation (Specific)

The class in Listing 1 is a concrete implementation of the *Command* interface to invalidate the keys in a cache. Notice that the execute method clears the entry of a statically bound cache. How does this work?

ClusterNotifier Implementation (Specific)

Now, let's take a look at the ClusterNotifier class (Listing 2). There's one method of interest:

```
notify(Command command)
```

This method is a synchronous call and takes in a command object as input that notifies all members of the cluster.

Internally, this class leverages the multicast facilities of WebLogic to send information across the cluster. WebLogic automatically handles partial fragment loss, resends, etc.

All of the classes in this article must be in the system classpath because the underlying WebLogic implementation is in the system classpath and needs to be able to find the ClusterNotifier class and delegate message class.

Summary

When most people think of a clustered-aware cache, they think of a truly synchronized and replicated coherent cache across the cluster. This model can be implemented for clustered caching, but has several issues such as the preservation of data integrity, transactionality, and network thrashing (performance implications).

An invalidation mechanism has several benefits. It only notifies the cluster members of the delta changes to *remove* an entry from its local store. Since it's reconstructable, it will only load the updated data on demand.

Cluster member notification is necessary for distributed programming, especially in the context of this article when updating information in cache. The concepts of this article can be extended to create custom *Command* objects such as a clustered "event notification" system (distributed Observer/Observable), asyn-

LISTING 1

```
public class InvalidateCacheCommand implements Command {

   private final static Map GLOBAL_CACHE = new HashMap();

   private final String key;

   public InvalidateCacheCommand(String key) {
         if (key == null) {
                  throw new IllegalArgumentException(“Key is
null.”);
         }
         this.key = key;
   }

   public void execute() {
         System.out.println(“Clearing cache for key: “ + key);
         GLOBAL_CACHE.remove(key);
   }
}
```

LISTING 2

```
// NOTE: This class uses undocumented private WebLogic APIs not guar-
anteed to be supported by future versions of WebLogic. Use at your own
risk.

public final class ClusterNotifier {

   private static MulticastSession session;

   private ClusterNotifier() {}

   public static void notify(Command command) {
         if (command == null) {
                  throw new IllegalArgumentException(“Command is
null.”);
         }

         if (session == null) {
                  System.out.println(“Not running in a cluster.
Invoking command locally..”);
                  command.execute();
         }
         else
         {
                  System.out.println(“Running in a cluster.  Sending
message to cluster members..”);
                  try
```

Sun microsystems

</Java_spoken_here>

Java™

INNOVATE
with the Power of Java™

June 27–30, 2005

JavaOne℠ Pavilion: June 27–29, 2005
Moscone Center, San Francisco, CA

Connect with the full power of Java™ technology at the JavaOne℠ conference.

In-depth EDUCATION
Evolve your skills in 300+ of expert-led technical and Bof sessions.

Real-world INNOVATION
Evaluate proven tools and technologies in the JavaOne℠ Pavilion.

Global COMMUNITY
Celebrate the tenth anniversary of Java technology.

Visionary INSIGHT
Hear what the future holds from industry leaders.

Experience a week unlike any other

EXPERIENCE THE 10TH ANNUAL JAVAONE℠ CONFERENCE.

Core Platform | Core Enterprise | Desktop | Web Tier | Tools | Mobility and Devices | Cool Stuff

Save $100!
REGISTER
by June 27, 2005 at java.sun.com/javaone/sf.

JavaOne℠
Sun's 2005 Worldwide Java Developer Conference℠

```
                    {
                              session.
send(new DelegateMessage(command));
                    } catch (IOException
io) {

io.printStackTrace();
                              System.err.
println(“An IO Exception occurred while
sending the “ +

          “message to the cluster.” +

            io.getMessage());
                    }
          }

   }

   public static void main(String argv[]) {
          Command command = new InvalidateC
acheCommand(“TestKey”);
          ClusterNotifier.notify(command);
   }

    private static synchronized Multicast-
Session getSession()
     {
        if(session == null)
        {
            ClusterServices cluster = Clus-
terService.getServices();
            if(cluster == null)
                return null;
            session = cluster.createMultica
stSession(null, -1, false);
        }
       return session;
     }

   public static class DelegateMessage
implements GroupMessage {
          private final Command command;

          public DelegateMessage() {
                    this(null);
          }

          public DelegateMessage(Command
command) {
                    this.command = command;
          }

          public void execute(HostID ig-
noredHostID) {
                    if (command != null) {
                              command.
execute();
                    }
          }
   }
}
```

blog-n-play.com™
i-Technology Blogs Read by Millions
beta
— This site will go beta February 15, 2005!

[Engage and Explore] the technologies, solutions and applications that are driving today's Web services initiatives and strategies…

www.sys-con.com/edge2005

web services EDGE conference&expo
FALL SERIES

CALL FOR PAPERS NOW OPEN!*

Coming to a City Near You ►

Web Services Edge
Fall Conference Series

3 Dynamic Conference Programs
Targeting Major Industry Markets

20+ seminars within 5 tracks will address the hottest topics & issues:

- ► Web Services: The Benefits and Challenges
- ► Web Services Security
- ► SOA (Service-Oriented Architecture) and ESB (Enterprise Service Bus) Strategies
- ► Interoperability, Incremental Integration, & Open Source
- ► The Management Process in Developing a Web Services Strategy

Program Features:

- ► Keynotes
- ► Tutorials
- ► Panel Discussions

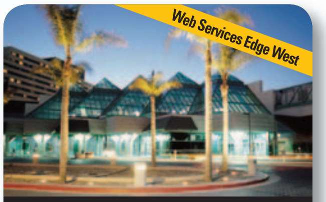
Why Attend:

- ► Improve the return on your technology investment
- ► Develop & sharpen your strategy and identify key action steps
- ► Find new ways to reach and impress customers with Web services
- ► Maximize the power of your enterprise
- ► Protect your business from security threats
- ► Assess Web services as a viable option

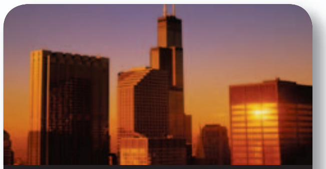
Attention Exhibitors:

- ► An Exhibit-Forum will display leading Web services products, services, and solutions

The Westin Washington, D.C.
Washington, D.C.
September 7-8, 2005
1



The Westin Santa Clara Convention Center
Santa Clara, CA
October 24-25, 2005
2



Hyatt O'Hare Airport
Chicago, IL
Oct 31-Nov 1, 2005
3

Register Today! www.SYS-CON.com/Edge2005

Sponsored by

WebServices Journal · XML Journal · .NET Developer's Journal · eclipse developer's journal · WebSphere Journal · Information Storage+Security Journal · WLDJ · JDJ · LinuxWorld Magazine · MX developer's journal · asp.netPRO · SD Times · CoDe · Software Test & Performance

*Call for Papers email: grisha@sys-con.com

For Exhibit and Sponsorship Information ► Call 201 802-3066

Produced by SYS-CON EVENTS

THE INTEGRATION CHALLENGE

DEVELOPING AN SOA-BASED INTEGRATION SOLUTION USING WEB SERVICES

By Eric Newcomer

Author Bio:
In his role as chief technology officer at IONA, Eric Newcomer is responsible for directing and communicating IONA's technology roadmap, as well as IONA's product strategy as it relates to standards adoption, architecture, and design. Eric has been involved in Web services standardization activities from the beginning. He was a founding member of the XML Protocols Working Group at W3C, which produced SOAP 1.2. He has recently co-written Understanding SOA with Web Services (Addison Wesley)..

Contact:
eric.newcomer@iona.com

In the early days of business computing, little attention was paid to the concept of sharing application logic and data across multiple machines. The big question faced by an organization was how to develop computer systems to successfully automate previously manual operations such as billing, accounting, payroll, and order management. Solving any one of these individual problems was challenging enough, without considering the possibility of basing all of a company's systems on a common, reusable architecture.

With the majority of operational business functions now automated, the next phase of evolution revolves around improving the ability of these systems to meet new requirements. Information technology (IT) departments are adding new user interfaces, combining multiple data sources into a single view, exploring methods for extending applications to mobile devices, and initiating efforts to replace old applications with more modern ones. These are common reasons given by CIOs for investing in new projects.

The industry also must face a paradigm shift toward service-oriented development and architectures based upon services. While service-oriented architecture (SOA) has been around for more than a decade, the concept is now becoming more mainstream because of Web services and is gathering momentum as the IT world makes the truly important shift from developing new systems to getting more out of earlier investments. Developing services and deploying them using an SOA is the best way to utilize existing IT systems to meet new challenges.

Service-Oriented Development

Complexity is a fact of life in IT, and it presents a major challenge in dealing with business requirements for new applications, evolving existing applications, and simply keeping up with all of the maintenance and enhancement requests of installed systems. Add the requirement of having the myriad systems found in the typical IT infrastructure actually work together to produce meaningful results, and the complexity can increase exponentially.

Now imagine a world in which all applications, regardless of whether new or previously deployed, use a common service-oriented programming interface (i.e., WSDL) and interoperability protocol (i.e., SOAP).

In such a world, the job of IT is much simpler; complexity is reduced and existing functionality is more easily reused. A common interface and interoperability protocol, through which any application can be accessed, also allows existing IT infrastructure to be more easily modernized or replaced.

This is the promise that service orientation brings to the IT world and when deployed in an SOA, becomes the foundation for more easily fulfilling a variety of strategic requirements, including multichannel access, business process automation, and rapid application integration.

The Integration Challenge

How does promise relate to reality though? Integration is required to resolve multiple types of business and technical application requirements among systems that, more often than not, were developed by different programming teams using different technologies and that were intended to solve different business problems. Integration projects have the unenviable job of reconciling the differences between multiple IT systems whenever those systems need to interact, whether or not they were designed for that purpose (and typically they were not). The more quickly integration solutions can be deployed, and at reasonable cost, the better.

Some of the common reasons for investing in integration solutions include:

- Mergers and Acquisitions (M&A): M&A activity typically results in multiple IT systems that handle similar transactions that need to be consolidated before the business value of the M&A can be fully realized. For example, bank mergers often result in multiple customer systems that may not easily allow cross-account management.
- Internal Reorganization: Although not as dramatic as M&As, internal reorganizations pose many of the same problems, and they often occur more frequently. Internal reorganizations may combine functions of multiple departments, including their supporting IT infrastructures, such as combining manufacturing operations across multiple product lines.
- Application/System Consolidation: Multiple IT systems often can be consolidated or replaced to save money, maximize efficiency of IT staff, and streamline business operations. For example, a telecommunications company that has multiple billing systems for wireless, wire line, and broadband could save considerable time and money by consolidating them.
- Inconsistent/Duplicated/Fragmented Data: Important business data may be spread across many systems and must be consolidated or "cleansed" to facilitate better decision making. For example, most businesses want to give service representatives a single view of the customer, which can only be accomplished if all of the pertinent customer information, contained in a variety of different systems, can be shared.
- New Business Strategies: Innovative companies frequently implement new business strategies that redefine the business environment and require IT systems to work together in novel ways. Eventually, other companies in the same industry have to adopt the same changes to stay competitive. Classic examples include just in time manufacturing and straight through processing of financial transactions.
- Compliance with Government Regulations: New government regulations may require redefining business processes to protect consumers and meet new information reporting requirements. For example, Sarbanes Oxley requires significant investment in reporting, auditing, and process improvement.
- Streamlining Business Processes: Old business processes that required data to be manually entered into multiple systems often need to be replaced with newer systems where transactions flow without human intervention. For example, a Web commerce company might previously have received orders via the Internet, but manually entered them into the order management and manufacturing control systems. Improved integration solutions allow the company to receive orders via the Web site and automatically enter them into the order management and manufacturing control systems.

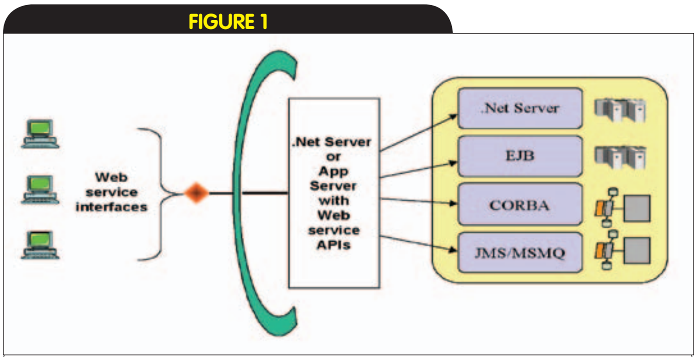
These problems illustrate that an integration solution is often not as simple as making two disparate systems work together. Because integration can represent a multifaceted problem, many different technologies, products, and processes have been used over the years to address it.

A few years ago, enterprise application integration (EAI) products became popular, based on the hub-and-spoke architecture, for addressing integration requirements. However, EAI products have proven expensive to purchase, consume considerable time and effort to deploy, and are subject to high project failure rates. Also, since most of these special purpose integration products relied on proprietary transports and protocol, integration projects faced additional difficulties and complexity whenever a company invested in more than one EAI solution. Instead of being able to easily meet new challenges and business requirements, they found themselves face with multiple, isolated islands of software, representing additional integration problems.

Similarly, integration approaches based on application servers suffer from development and deployment difficulties inherent in the three-tier architecture (as does their precursor, the TP monitor). These difficulties include a high degree of hand coding for integration logic, a less flexible (i.e., more tightly coupled) request/response style messaging design center, and large runtime deployment footprint (often with associated additional hardware cost).

Recent experience with Web services and SOA-based solutions shows that a better answer is available. Instead of attempting to deal directly with the complexity of multiple incompatible applications on multiple computers through a single type of product such as an EAI broker or application server, it is now possible to add a layer of abstraction that is open and standards based, and through its neutral architecture is easy to integrate with virtually any new or existing environment.

FIGURE 1



Web service enabling application servers

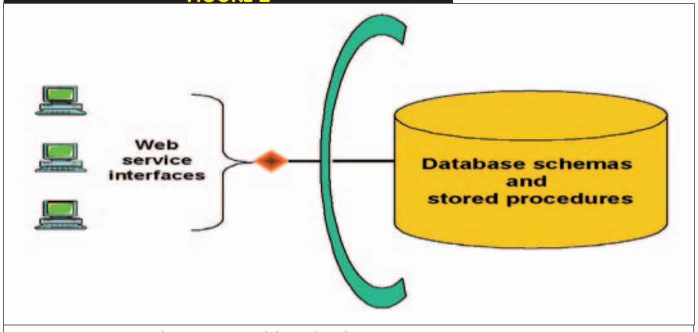
SOA-Based Integration

The leading approach for using Web services in integration solutions is SOA. Using Web services in the context of an SOA provides a strategic and systematic method to solving integration and interoperability problems.

SOA tends to work best for organizations that are trying to maximize the long-term results of an integration architecture by heavily investing in one for strategic advantage. It addresses the challenge of making different pieces of an enterprise work together and making corporate infrastructure less complex – with fewer platforms and more standardization.

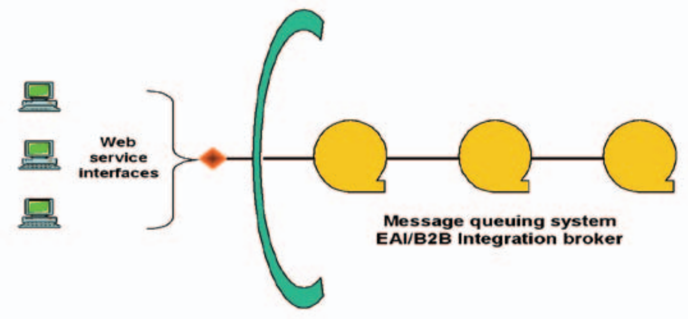
SOA is by definition technology independent, since it is an architecture or style of design, not a product. An SOA can be (and historically has been) implemented using a variety of technologies, including CORBA and message queuing systems. Web services provide the best modern technology platform for implementing an SOA.

FIGURE 2



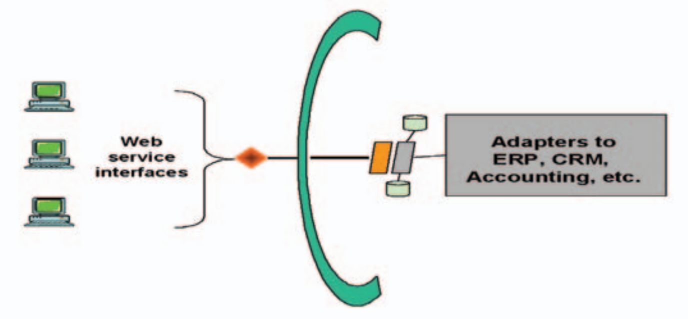
Web service enabling database management systems

FIGURE 3



Web service enabling message queuing system

FIGURE 4



Web service enabling ERP/CRM systems

Unlike a traditional "rip and replace approach" where legacy environments are replaced wholesale with lower cost alternatives, SOA fulfills the need to utilize functionality in existing applications to support innovation. SOA meets this requirement by encapsulating functionality and exposing it through standard Web services interfaces. The resulting services can then be easily consumed by new applications, and replaced incrementally.

Figure 1 illustrates the way in which application server technology (including .NET and J2EE servers) can be enabled using Web services to integrate with a variety of back-end systems. CORBA systems can also be viewed as an instance of application server technology. Although they are not three-tier structured the way J2EE and .NET servers are, sometimes CORBA-based systems are used to provide the equivalent features and functions for C++ (and Java) applications.

Many modern application server environments, including CORBA-based products, also include a form of asynchronous messaging that can be service enabled as well. Technologies in this category, broadly including transaction processing monitors, also run on mainframes.

Figure 2 illustrates that Web service interfaces are available for direct access to database management systems, either by transforming SQL Schema defined tables directly to XML equivalents, or by providing a wrapper to invoke stored procedures.

Figure 3 illustrates how Web service interfaces are applied to message queuing systems, whether used as plain message transports, or in the context of enterprise application integration (EAI), or business to business (B2B) broker architectures.

Figure 4 illustrates the way in which enterprise resource planning (ERP), customer relationship management (CRM), and other enterprise application packages are enabled using Web services interfaces. More and more of these systems are becoming Web service–enabled over time, but today the industry continues to see a mixture of Web service–enabled adapters and Web service interfaces that access packaged application functionality directly.

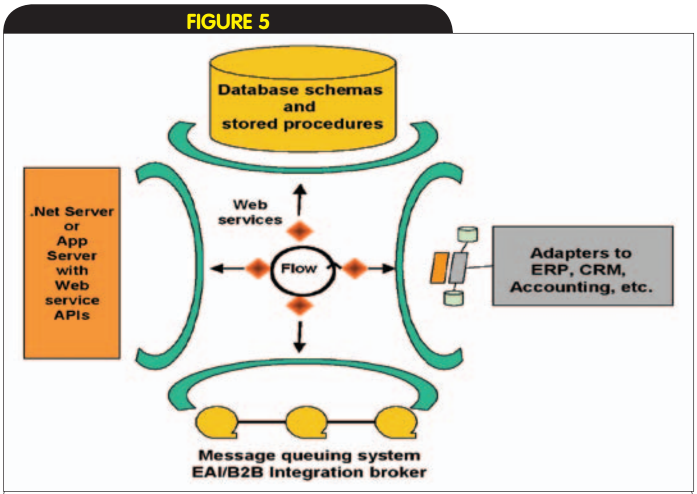
Figure 5 illustrates how basic integration flows can be defined once existing (and new) software assets are Web service–enabled. The flows can include three-tier application architectures (i.e., application servers and TP monitors), message queue architectures (including hub-and-spoke EAI systems), direct database access, and packaged application adapters.

Integrating applications such as this within the context of an SOA involves defining the basic Web services interoperability layer, as well as the enterprise quality of service layer, to bridge features and functions such as security, reliability, and transactions used in current applications. Over time this approach also involves the ability to define automated business process execution flows across Web services once the SOA is in place.

Applying SOA with Web Services for Integration: The ESB

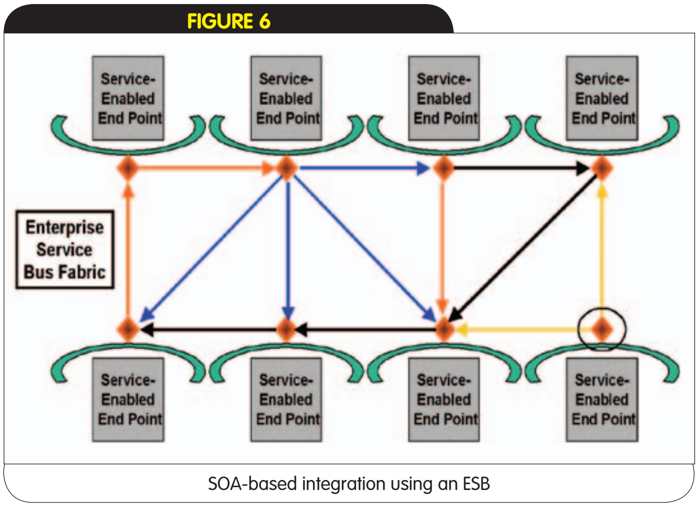
While SOA and Web services technologies can be used to solve a broad range of IT problems (especially when used to quickly and easily join the disparate systems that comprise the typical enterprise IT infrastructure), enterprises typically require additional infrastructure beyond the basic interoperability provided by WSDL

FIGURE 5



Defining execution flow at the Web services level

FIGURE 6



SOA-based integration using an ESB

and SOAP. Web services–based interoperability solutions can easily be extended for use in enterprise integration solutions that lower cost and increase IT value by using an Enterprise Service Bus (ESB) to connect service-enabled endpoints.

ESBs support the development and deployment of a fabric of services that enables system interoperability and rapid application integration in an SOA while preserving the mission-critical features and functions on which IT solutions depend. ESBs provide implementations of the higher-level Web services specifications, such as security, transactions, metadata management, and reliability.

ESB technology addresses some of the most common integration challenges that IT organizations face today: How should the business anad technical rules for satisfying information requests and data transformation be defined and applied while simultaneously reconciling the competing goals of speed, interoperability, portability, and flexibility? How can we implement an architecturally neutral solution to connect newly developed services to services defined for a myriad of existing applications and software systems? How can we really handle the mixture of two-tier, three-tier, hub-and-spoke, and other IT architectures in a common way?

The ESB is the Web services–based infrastructure designed for IT organizations with multiple generations of business applications, technologies, and architectures. ESBs combine features from several previous types of middleware into one package to make IT assets work together, thus forming the basis of an agile SOA integration solution.

Figure 6 illustrates how an ESB connects Web service–enabled endpoints in a variety of architectures, including the potential for peer to peer, two-tier, three-tier, hub-and-spoke, and other interaction patterns that can all be achieved using an architecturally neutral ESB. Instead of viewing the enterprise integration problem through a single product approach, the ESB leverages the power of the Web services abstraction to easily connect service-enabled endpoints using a variety of common integration patterns. The ESB facilitates the creation of composite Web services and of course any kind of IT architecture can be hidden behind the WSDL interface. Once services are available for new applications and enabled for existing applications, the ESB fabric provides the foundation for automating business process flows.

For example, an ESB accepts information on requests from service requesters (e.g., "retrieve all information based on John's Accounts with ABC Bank"), and returns the requested information using metadata describing what information is available from a collection of services. Reusable technical and business services can be built on top of the Web services platform and accessed using the ESB.

Summary

Today's answer to application integration problems is an SOA with Web services. An SOA maps easily and directly to a business's operational processes and supports a better division of labor between technical and business staff. Additionally, SOA uses an interface model capable of unifying existing and new systems. When Web services interfaces are available throughout a department or an enterprise, service-oriented integration projects can focus on composing Web services instead of dealing with the complexity of incompatible applications on multiple computers, programming languages, and application packages.

Whether using Java, C++, Visual Studio, CORBA, WebSphere MQ, or any other development platform or software system (such as J2EE or the .NET Framework), SOA provides the architecture and Web services provides the unifying glue.

As businesses and governments continue to struggle to align IT expenditures with the bottom line to achieve strategic market objectives, software productivity gains can be elusive, especially compared to gains in hardware price and performance. With the widespread adoption and implementation of Web services, ESB products are finally in the position for enterprises to improve software productivity.

In summation, Web services provide the technical tools needed for application integration and SOA offers a strategic and systematic architectural approach to apply them. Using Web services for integration with an ESB is the best way of applying SOA to solve integration problems.

This article was adapted from the author's books, Understanding Web Services, *Addison Wesley, 2002, and* Understanding SOA with Web Services, *written with Greg Lomow; Addison Wesley, December 2004. The two books cover all major Web services standards, service-oriented architecture, and business process management.*

Java Messaging Services Clustering Part 2

JMS CLUSTERED ARCHITECTURES, COLLOCATION, AND DISTRIBUTED LOAD BALANCING

By Raffi Basmajian

Author Bio:
Raffi Basmajian is an application architect for a major financial services firm in New York City. He specializes in architecting Java/J2EE distributed systems and integration frameworks.

Contact:
raffibasmajian@yahoo.com

In Part 1 of this article series we discussed the fundamental aspects of clustering JMS resources in a WebLogic cluster. In Part 2 we will discuss JMS clustering in the context of design and configuration strategies that demonstrate how to create efficient JMS architectures.

Clustering Concepts

Enterprise applications come is a variety of sizes. For small-scale deployments, a J2EE application might be deployed on a single WebLogic server. Such a configuration, for example, would call for little consideration to the location of JMS and JDBC resources since clients that need to access those resources would, in most cases, reside on the same physical server. In contrast, mid- or large-scale J2EE deployments are most likely clustered across many physical servers to provide the application with high availability and load balancing capabilities. A clustered application, however, is more complex and network-demanding. Server instances in a cluster provide services to other server instances, sometimes across remote connections. It is important to consider the location of clients in an application cluster with respect to the resources they need to access. In some cases you may prefer to minimize remote connections by placing clients and resources on the same physical server; this is known as *collocation*. In other cases, however, you may want to distribute the work load among the physical servers in a cluster to utilize all available processing power; this is known as *distributed load balancing*.

The concept of collocation is important to consider in any clustered application. The idea behind collocation is simple: target your application server resources, such as JMS and JDBC, to physical server instances where clients are located and need access to those resources. For example, a JMS message producer first must find a connection factory, and then establish a connection to a JMS server for accessing a specific destination. If the JMS client uses a connection factory located on a remote server instance, a remote connection will be created. Additionally, if the connection factory and JMS server hosting the destination are located on separate physical servers, a remote JMS connection is created. While there is nothing preventing the JMS client from using remote connection factories and JMS resources, a more efficient approach is to collocate the JMS client with the connection factory and JMS resources it needs, thereby avoiding the use of remote connections and reducing network traffic.

With distributed load balancing, WebLogic allocates the work load to all server instances in the cluster to provide uniform use of available resources.

In a JMS cluster configuration, for example, messages sent to a distributed queue are load balanced among the destination members hosted by JMS servers on separate server instances. This approach helps to ensure accurate contribution of processing capacity from all physical servers in the cluster based on the preconfigured load balancing scheme for the distributed destination.

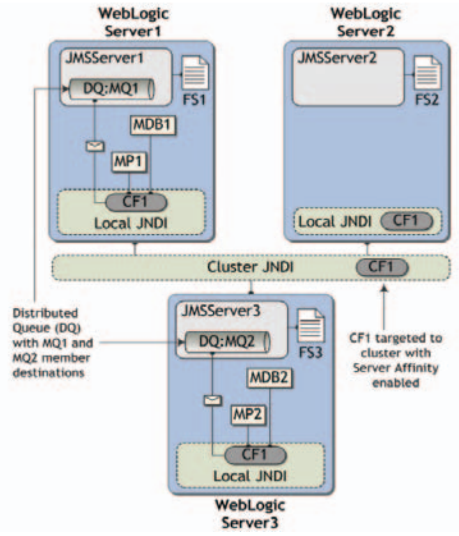
Collocation and distributed load balancing in the context of JMS are key concepts in this article. The next series of architectures illustrate different configurations and how they affect the behavior of clustered JMS applications.

Internal JMS Clients

Figure 1 illustrates a typical WebLogic cluster with distributed JMS resources. The cluster consists of three physical server instances. The WebLogic server instances each hosts its own JMS server with persistent file store (FS1, FS2, and FS3). A distributed queue (DQ) consists of two physical queue members, MQ1 and MQ2, which are deployed to WebLogic server instances 1 and 3, respectively. A connection factory (CF1) is targeted to the cluster making it available to each WebLogic server instance as a local JNDI object. The JMS clients, including message producers MP1, MP2, and message-driven beans MDB1, MDB2, reside on the WebLogic instances that host their desired destination, DQ.

Let's first look at the location of the connection factory, CF1.

FIGURE 1

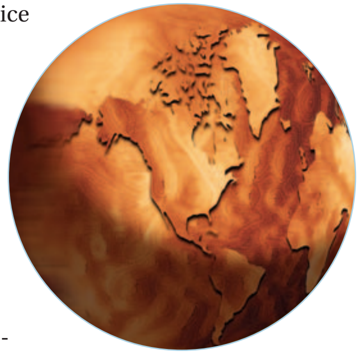


Internal JMS clients collocated with members of a distributed queue

By targeting CF1 to the cluster, WebLogic first attempts to load balance consumers and producers on the same physical server instance. As depicted on WebLogic Servers 1 and 3, the JMS clients first connect locally to CF1. The connection factory then creates a JMS connection to the local JMS server on that WebLogic instance, connecting the clients with a local member destination of DQ. In this case, local JMS connections are created for the JMS clients.

It should be clear from Figure 1 that all JMS clients, including the connection factories, JMS servers, and distributed destination members, are collocated. This is ideal for WebLogic as it avoids the cost of load balancing across the remote member destinations of DQ while routing messages to collocated destinations instead.

To ensure WebLogic will first load balance locally, enable *Server Affinity* from the WebLogic Administration console for any connection factory collocated with JMS resources. In Figure 1, we enabled this option on CF1, and then targeted the connection factory to the entire cluster. Note that connection factories are clients to JMS servers, so it is good practice to collocate these types of administered objects as well, not just JMS producers and consumers.



WebLogic Server 2 does not host any queue members for DQ, so we don't target JMS clients to that server instance to avoid routing messages to remote destinations members. This architecture is suitable for systems that prefer increased performance at the cost of some distributed load balancing. This expense is attributed to the preference of local resources over remote resources. Even the persistent stores configured for the JMS servers use a local file resource. The file stores may be configured for superior performance by setting the Synchronous Write Policy to *Disabled* (configurable from the Administration console for each file store), which completes transactions once they are stored in memory. In terms of reliability, however, this is the least desirable policy, and is not recommended for applications that cannot afford to lose messages. The default policy *Cache-Flush*, which forces all writes to disk before transactions are complete, is a good trade-off between performance and transactional reliability.

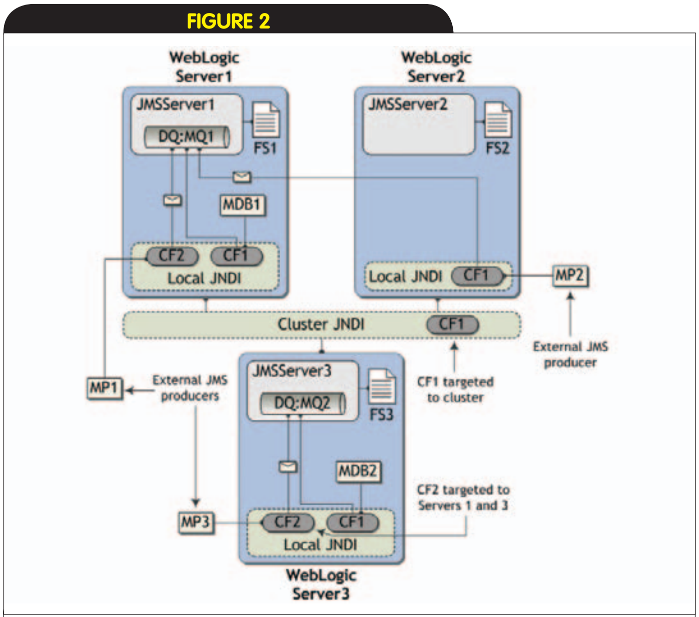
External JMS Clients

The WebLogic cluster depicted in Figure 2 is similar to Figure 1, except here the message producers MP1, MP2, and MP3 are external JMS clients. These types of clients may represent batch processes executed throughout the day to perform specific message-based tasks. The connection factory CF1 is targeted to the cluster with *Server Affinity* enabled. The message-driven beans MDB1 and MDB2 connect with the collocated CF1 on their respective WebLogic server instances to access DQ without the need for remote JMS connections, just as in Figure 1.

The external JMS clients may also use CF1 to access DQ, but this may cause unnecessary remote connections and remote routing of message traffic. As depicted in the illustration, MP2 connects to CF1 on WebLogic Server 2. The connection factory must now connect MP2 with a physical queue member of DQ. Connection factory CF1 is enabled with *Server Affinity*, so it first attempts to connect MP2 with a local member destination of DQ. Since JMSServer2 does not host a collocated queue member, CF1 is forced to create a remote JMS connection to MQ1 on WebLogic Server 1, which effectively routes all message traffic from MP2 across that remote connection. This can create an undesirable network-intensive process if MP2 is generating hundreds, if not thousands, of messages per day, such as in a financial trading application that publishes stock prices nonstop to trader desks throughout the daily trading session.

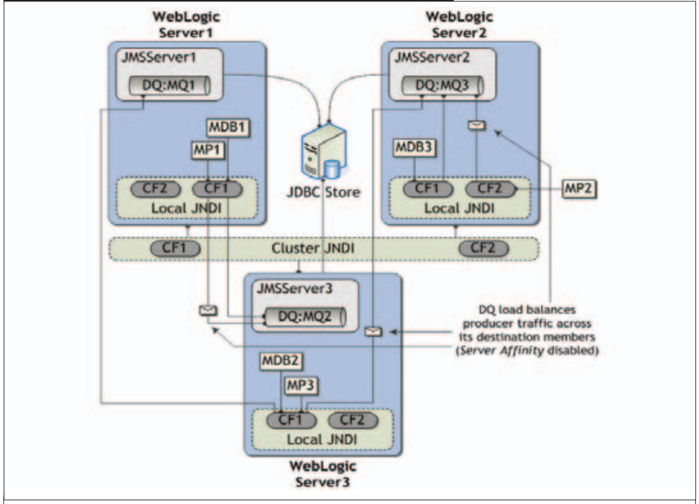
To avoid this scenario and create a more efficient architecture, we deploy a new connection factory (CF2) with *Server Affinity* enabled and target the resource to those server instances hosting member destinations for DQ, namely, WebLogic Server 1 and 3. Additionally, to prevent external JMS clients from connecting to CF1, we disclose the availability of CF2 as the *only* connection factory accessible to external JMS clients. Thus, when an external message producer connects with CF2 to access DQ, a queue member destination is available as a collocated resource. Although this approach routes all message traffic from a message producer to a local queue destination, keep in mind that the potential exists for the physical server instance hosting the destination to become overutilized if a single JMS client is producing messages while other producer clients are not.

FIGURE 2



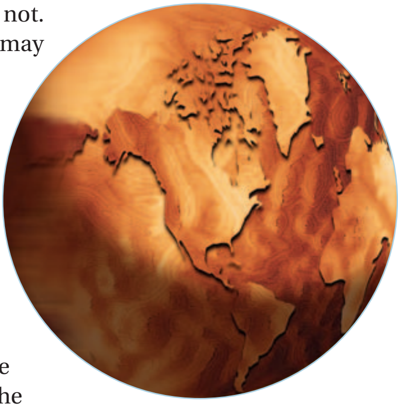
Targeting connection factories and distributed queue members requires special consideration for internal and external JMS clients

FIGURE 3



Distributed load balancing is achieved by disabling Server Affinity on connection factories

Also note that external JMS clients may conveniently use the cluster address and JNDI name of CF2 to locate the connection factory without consideration as to whether the connection factory is targeted to the cluster or to individual server instances.



Distributed Load Balancing

The JMS cluster in Figure 3 illustrates how to achieve the full capabilities of distributed load balancing while providing a level of high availability. The connection factories CF1 and CF2 are both targeted to the cluster, but in contrast to the previous architectures, *Server Affinity* is disabled. In this case, the load balancer for the distributed destination DQ will first favor a queue member with an active consumer and persistent store for any messages sent from the message producer clients, even if the destination is located on a remote server. In Figure 3, for example, MP1 connects to CF1 on WebLogic Server 1. Then, CF1 creates a JMS connection to the remote member destination MQ2 to receive the message. Since *Server Affinity* is disabled, DQ will always make a decision based on its load balancing algorithm to distribute each message sent from a message producer among its destination members. The next message sent from MP1 will most likely be directed to a different member destination of DQ.

For a newly created, message-driven bean, the connection factory CF1 will first favor a queue member destination of DQ *without* any current consumers. This behavior helps to ensure each member destination has at least one message consumer ready to process a new message on the queue, and this is clear in the diagram as each physical queue member is pinned with a message-driven bean listener. Again, since *Server Affinity* is disabled on CF1, a message consumer might be pinned to a member queue over a remote JMS connection, as is the case with MDB1 and MDB2.

Finally, in this architecture we used a JDBC store as the persistence mechanism for all JMS servers. A JDBC store has an advantage over a file store in that it facilitates recovery in a failure scenario. If, for example, a WebLogic server instance hosting a JMS server fails, the JMS server and all its destinations can be easily migrated to another WebLogic instance, provided the JDBC store connection pool is targeted on the new server instance receiving the migrated JMS server. Migrating a JMS server using a file store, however, will require the new server instance to have access to the file store, which may be achieved by using a shared disk; otherwise, you will need to copy the file store data files and transaction logs to allow WebLogic to recover any persisted messages. Note that a JDBC store may also represent a single point of failure if the database fails. For this reason it is ideal to use a fault-tolerant database, or, at a minimum, use the *Test Connections on Reserve* attribute on the JDBC store connection pool to reconnect with the failed database after it is available again without having to restart the WebLogic server instance.

JMS Clustering Best Practices

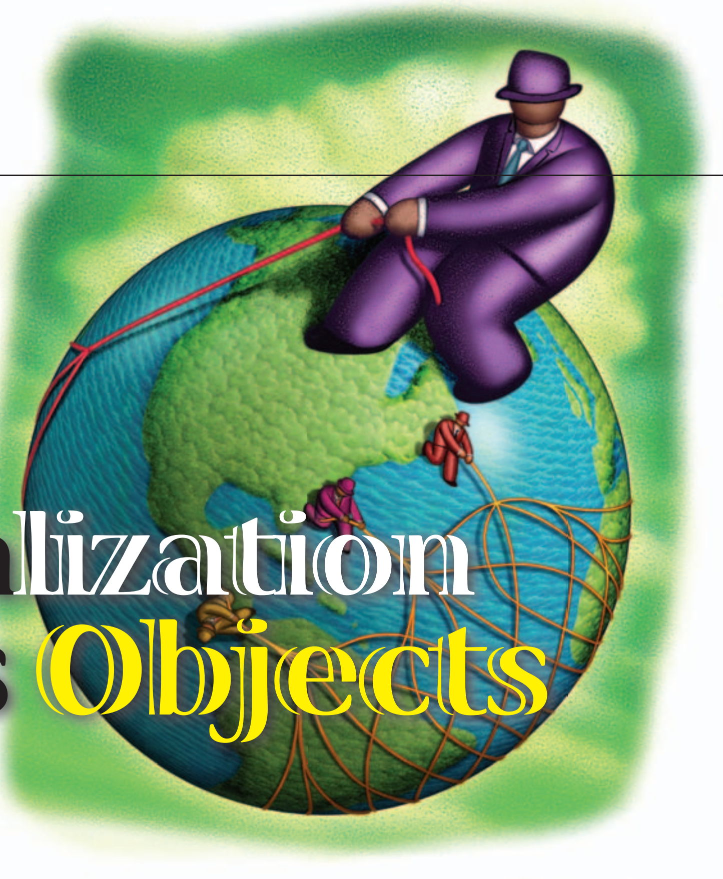
Here is a list of best practices to follow when designing an application that utilizes JMS clustering.

- ***Collocate to reduce network overhead***. Collocate JMS clients with JMS resources for better performance while reducing remote JMS connections and network traffic.
- ***Use distributed load balancing by disabling Server Affinity***. Disable *Server Affinity* on JMS connection factories to distribute message traffic to all distributed destination members.
- ***Use file stores for increased performance***. Use a file store to increase the performance of persistent messages.
- ***Use JDBC stores for reliability.*** Use a JDBC store to easily migrate JMS resources from a failed WebLogic server.
- ***Use message-driven beans for reliable message consumption***. Message-driven beans are ideal for reliable, asynchronous message consumption. If you deploy a MDB listening on a distributed destination, WebLogic ensures there are consumers listening on each destination member. This is an ideal approach for reliability and performance, coupled with the features of load balancing and fail over provided by WebLogic.
- ***Distributed topics for nondurable subscribers***. For nondurable topic subscribers, ensure that all JMS servers hosting topic members for a distributed topic are configured without a JMS store; otherwise, WebLogic will treat all consumers as durable subscriptions, which can seriously degrade performance. If you can not remove the store from a JMS server because it is used by another destination, then an alternative is to set the StoreEnabled parameter to False for each individual topic member destination.

Conclusion

This article covered JMS cluster architectures and illustrated the pros and cons of collocation and distributed load balancing. Collocation reduces the need for remote connections and improves the performance of processing JMS message traffic, but may lead to disparate utilization of processing capacity among physical servers. Distributed load balancing promotes equal distribution of message processing among physical JMS destinations in a cluster, but also increases network overhead. However, just as with everything else in life, including your friends, relationships, your job, pets, and, distributed JMS systems too, it's all give and take, so the decision you make should be based on the requirements and context of the application you are designing.

That's it for this article! Please feel free to e-mail me with your comments and questions.



Adding Internationalization to Business Objects

TEACHING YOUR OBJECTS TO SPEAK KLINGON

By Alex Maclinovsky

Author Bio:
Alex Maclinovsky is a principal application architect with SeeBeyond Technology's Center of Excellence, specializing in Composite Applications based on J2EE technologies. For over 15 years he has focused on architecting and developing large distributed object systems on the enterprise, national, and global scale. His professional interests include solution-oriented architectures, adaptive frameworks, and OO methodologies. Alex can be contacted at AMaclinovsky@SeeBeyond.com and his Professional Profile can be found at www.geocities.com/maclinovsky/pro.

AMaclinovsky@SeeBeyond.com

The Web transcends national boundaries and many sites reach global audiences, which brings into the spotlight the problem of internationalization of Web applications. The Java community has an established approach for supporting multilingual applications through resources stored in `ResourceBundle`, which works well in examples but has many shortcomings when applied to real world problems.

In this area, BEA has closely followed mainstream Java. The WebLogic internationalization toolkit consists of a thin wrapper around ResourceBundle and tools for maintaining parallel UI components, thus inheriting the same limitations.

On the other end of the spectrum, Content Management Systems are often used for internationalizing large information portals. They find and deliver the appropriate content by tagging it with language metadata. They offer high scalability and near real-time content availability, but can only internationalize *content*, or files, offering no support for application data. Unfortunately neither of these approaches offers a viable solution for complex data-driven sites like Amazon or Travelocity. As a result, developers often resort to maintaining parallel sites for each language.

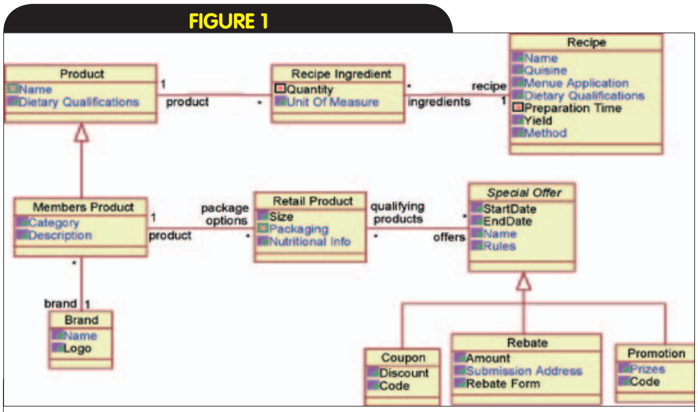
This article expands the topic of *Business Objects* in Web applications introduced in the feature "Publishing Business Objects in Portals" (*WLDJ* July 2004). It follows the Food Portal and Knowledge Repository examples, whose internationalization requirements could not be met with traditional Java or BEA tools and called for a different approach. This resulted in the development of *Polyglot architecture*, which closes the gap by adding the internationalization support on the level of Business Objects.

The Object of Internationalization

As a reminder, in the first article we introduced the concept of Business Objects in the context of Web portals as *highly structured application data* that has the following characteristics:

- Business Objects are strongly typed.
- Business Objects must have all of their attributes present and often have stringent constraints on their values (e.g., Coupon.Discount must be a positive number between 1 and 99, SpecialOffer.EndDate must be a valid date in future, etc.).
- Business Objects can relate to other Business Objects (e.g., Coupon *extends* SpecialOffer), which in turn can *feature* Products, which *belong* to a Brand, etc. Some attributes can be complex types or Dependent Objects; for example, Recipe includes a list of RecipeIngredients, which consists of Product, Quantity, and UnitOfMeasure. When presenting such objects through a portal, it is essential to be able to traverse these relationships and maintain their integrity, as dependents and parents are published or unpublished.
- Business Objects differ from other types of application data, such as bank account information

FIGURE 1



Business Objects from the Food Portal

or historical stock prices, because they have the same publishability requirements and associated life cycle as other types of content; e.g., Products are introduced and discontinued, Special-Offers expire, Vacancies are filled.

We then illustrated the concept of Business Objects and their role in the portal applications with two examples based on previous projects.

Food & Nutrition Portal

The portal is targeted to the food enthusiasts. It features a wide range of content about food, various cuisines, and related matters. It receives content via distributed submission and publishes from a broad base of contributors as well as member companies, who use it to promote their products. The portal uses traditional content, such as articles, ads, features, campaigns, as well as business objects including products, brands, and packaging options, which can be added to the product catalog; recipes featuring these products, which go into the recipe box; and special offers promoting these products, etc. The simplified domain object model is presented in 1. The system supports various modes of browsing and searching and has rich navigation capabilities along class relationships.

Global Knowledge Repository

This repository is an Information Portal that aggregates systematized information from diverse Information Sources, and presents it organized along the taxonomy, similar to the one found in Yahoo's Directory. This taxonomy, implemented as a traversable multi-hierarchy of Navigation Nodes, is capable of building indexes at any level. All information on the site is tied back to its Sources, which form another hierarchy with administrative and semantic delegation.

All informational pages on the site, including navigation nodes, info sources, and standalone pages, are publishable by the contributors without any involvement of developers or site administrators. Both content and navigational assets are assembled at rendering time from atomic elements or contentlets. The simplified domain object model is presented in 2.

Both systems were developed on WebLogic, and in this article we will keep using these examples to highlight the internationalization requirements of data-rich portals. We used blue font to identify language-dependent attributes in the class models.

The Problem

As highlighted in the preceding examples, we need a *uniform* way to add internationalization to both View and Model tiers of Web applications. The ideal approach should be *scalable* to real-life limits in terms of both the number of localized elements and supported languages. It should *not rely on replication* of data and should be *architecturally neutral*, supporting different presentation technologies and types of data access, including Entity Beans, where an instance of Recipe can be read once from the database, cached in the Container and then rendered for different users in their preferred language. Finally, that approach should be *object oriented*, so that adding internationalization to application requirements does not distort its Domain Object Model.

Existing Solutions

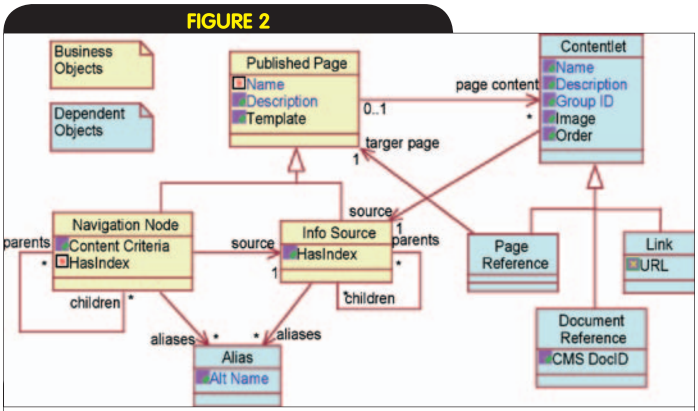
Maintaining Parallel Sites – The Traditional Way

This is a tried-and-true *brute force* solution. It is the easiest to implement, but it does not meet most of the above criteria and has many disadvantages: it does not scale well when you have to add support for additional languages, requiring almost linear efforts. And since parallel sites require parallel databases, the probability of introducing data inconsistency grows even faster. Nevertheless, this approach is widely used and is recommended by BEA for localizing WLP 8.1.

Using ResourceBundle – The Java Way

J2SE applications store all locale-specific resources in classes called resource bundles. Resource bundles support internationalization by separating localized data from the code that uses it. Each ResourceBundle instance contains localized resources for a particular locale, usually loaded from a file. A resource bundle is a map of key/value pairs, where keys are the names that are shared across locales, and values are the resources localized to the resource bundle's locale. Application code references locale-specific resources by key rather than storing them directly. It works well for transactional applications with modest UIs, e.g., online banking, where language-dependent features are limited to a few thousand messages. However, this approach would never scale to millions of messages. Moreover, since bundle keys are typically used in the application code, the code has to be changed if you want to add even a single new message. Ironically, adding support for a whole new language can be done without touching the code; however, you can guess

FIGURE 2



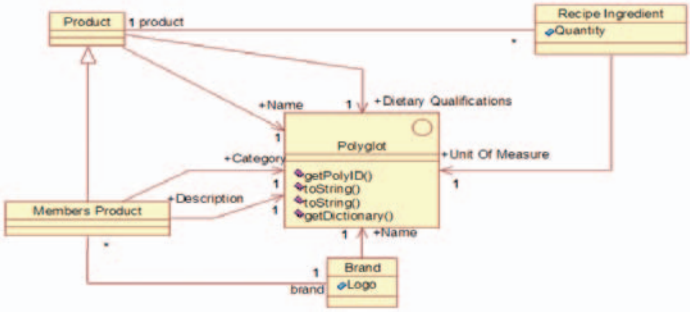
Business Objects from the Global Knowledge Repository which has a bigger effect on application maintainability. Finally, it offers no help in internationalization of Business Objects – referencing bundle keys form database schema would lead to an integrity maintenance nightmare, and bundles are loaded atomically, which leads to scalability issues when working with large datasets.

Directories, Tags & Tools – The WebLogic Way

The WLP I18N framework builds on the ResourceBundle approach and adds a number of useful features and capabilities:

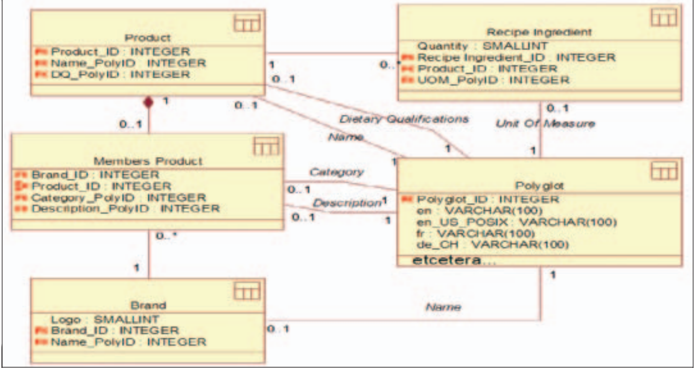
- Tag i18n:getMessage provides a convenient wrapper around MessageBundle. It now supports reloadable and non-escaped message bundles.
- Tag i18n:localize allows defining the language, country, variant, and base bundle name to be used throughout a page.
- Tags l10n:include, l10n:forward, and l10n:resolve can be used for searching for a localized version of the page.
- The framework looks up and loads localized copies of resources, such as portals, portlet files, skeleton files, portlet content JSPs, static HTML, and image files contained in locale directories.
- Localization of component property pages through locale markup in the portal administration.
- Ability to add custom locale providers.

FIGURE 3



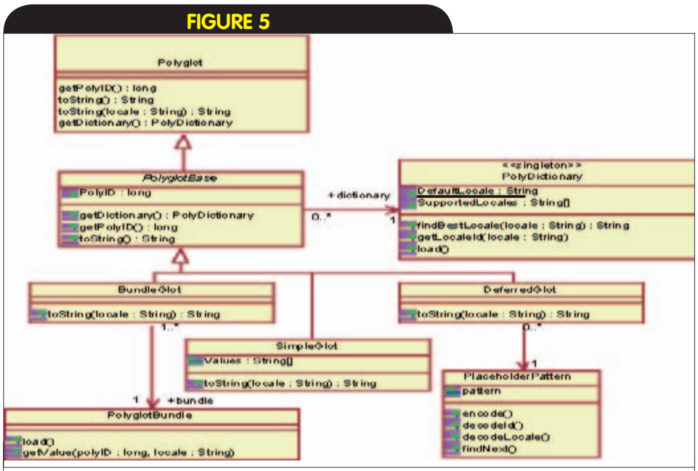
Food Portal object model with Polyglot

FIGURE 4



Portal data model with Polyglot

FIGURE 5


Polyglot
getPolyID() : long
toString() : String
toString(locale : String) : String
getDictionary() : PolyDictionary
PolyglotBase
PolyID : long
getDictionary() : PolyDictionary
getPolyID() : long
toString() : String
+dictionary
0..*
1
<<singleton>>
PolyDictionary
DefaultLocale : String
SupportedLocales : String[]
findBestLocale(locale : String) : String
getLocaleId(locale : String)
load()
BundleGlot
toString(locale : String) : String
DeferredGlot
toString(locale : String) : String
1..*
SimpleGlot
Values : String[]
toString(locale : String) : String
0..*
1
PlaceholderPattern
pattern
encode()
decodeld()
decodeLocale()
findNext()
1
+bundle
PolyglotBundle
load()
getValue(polyID : long, locale : String)

Implementation of Polyglot framework

Although WLS does not offer any specific tools for internationalization, the above tag libraries are available in the Workshop and should be usable in non-portal applications built on the WebLogic platform.

Since BEA's I18N framework is based on the standard J2SE internationalization technology, taking aside the convenience, they exhibit identical weaknesses. And although the Portal stores some *locale markup* data in the database, the lookup and retrieval mechanism is encapsulated in the framework and is not open to the applications for use with their own data.

Polyglot Architecture

When it became apparent that none of the existing internationalization technologies can meet the above requirements either as-is or with minimal enhancements, we set out to design our own, code-named *Polyglot*. Our approach is centered on the use of Polyglot objects in place of all language-dependent attributes of Business Objects and presentation elements. This allows all language information to be carried with the object throughout its life cycle within Model and Controller layers. The language-specific rendering is deferred to the View, permitting a single entity to be accessed by multiple users who want to see it in different languages.

Adding Polyglot to an existing application is as simple as changing the type of all language-dependent attributes from String to Polyglot. For example, the top left corner of the Food Portal model from Figure 1 will now look as shown in Figure 3.

The database schema will change as shown in Figure 4. As you can see all language-dependent columns are replaced with foreign keys to the Polyglot table that consists of a Polyglot_ID, which serves as a primary key, and a VARCHAR column for each supported locale. The column names match Java locale names exactly. The first column designates the *default locale* and is not *nullable*. All other values are optional.

No two Web applications are exactly alike, and neither are their internationalization and performance requirements. To give portal developers a wider range of options, the Polyglot framework shown in Figure 5 comes with three different concrete implementations, which offer different language resolution strategies. They all implement Polyglot interface and share a common base class PolyglotBase. The main method toString() takes a single argument, identifying desired locale, and returns localized value. A second toString() without parameters returns the value for the default locale.

Class PolyDictionary is a singleton that reads the meta-information from the polyglot table. It keeps the list of supported locales, and keeps track of the default locale. Method getLocaleId() finds closest available matching locale for a requested locale, e.g., for the polyglot table presented in Figure 4 it would return en_US_POSIX for en_US, en for en_UK and de_CH for de. It also assigns integer indexes to locales that are used for fast access to values.

Polyglot Implementations

Class BundleGlot is the most lightweight Polyglot implementation and is perfect for applications with a smaller number of language-dependent messages. It uses class PolyglotBundle to bulk-load the entire Polyglot table into memory, so BundleGlot instances contain just the ID. It has similar or better performance than the ResourceBundle. However, since it uses the same database schema and Java infrastructure as the other two, it can always be scaled up transparently to the rest of the application as the message count increases.

Class SimpleGlot is the direct implementation of the Polyglot pattern; it contains the message values for all of the supported languages, which are read from the database at the same time as the parent object. This option works best for applications that have a large or volatile set of messages but need to support just a few languages.

Finally, DeferredGlot offers the most scalable option in terms of number of messages and supported languages, beyond the limits of when caching stops being feasible. Similarly to BundleGlot, its instances only contain the ID. However, instead of preloading the messages, it does quite the opposite. Its method toString() uses the PlaceholderPattern object to create specially formatted placeholders containing polyglot ID and resolved locale values. The application then uses a response Filter to parse ServletResponse and find all of the placeholders. Having determined all of the message/locale combinations required for rendering the current page, polyglot filter reads all of the values from the database in a single hit, and replaces the placeholders with the appropriate messages.

Conclusion

Polyglot architecture offers a useful alternative to the internationalization mechanisms available in Java and specifically in the WebLogic platform. It can be used to meet performance and volume requirements of a wide variety of Portals and other Web Applications, and is free from many of the limitations of solutions based on ResourceBundle. Polyglots offer a uniform internationalization mechanism, spanning the *Model* to the *View* layers, and can be used to handle tasks on both ends of the spectrum, thus offering a universal and scalable approach to internationalization.

References

- *Publishing Business Objects In Portals (The July 2004 issue of* WLDJ *contains detailed descriptions of the examples used in this article):* www.sys-con.com/story/?storyid=45559&DE=1
- *The "J2EE Internationalization and Localization" chapter of* Designing Enterprise Applications with the J2EE Platform, Second Edition *offers a good overview of Java internationalization capabilities:* http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/i18n/i18n.html
- *WebLogic Server Internationalization Guide:* http://e-docs.bea.com/wls/docs81/i18n/
- *Documentation on BEA's Internationalization and Localization JSP Tags:* http://e-docs.bea.com/workshop/docs81/doc/en/core/in-

Acid Reign

THE TRANSACTION PROCESSING MONITOR IS DEAD; LONG LIVE THE TRANSACTION PROCESSING MONITOR

By Peter Holditch

In browsing around the Web, as one occasionally does in a free nanosecond, I read an interesting article about twp-phase commit transactions by Gregor Hohpe of ThoughtWorks ("Your Coffee Shop Does Not Use Two Phase Commit"). Gregor comes at the subject from the direction opposite the one I usually take in this column, since I am of a TP persuasion, but he covers the same arguments that I have explored in the past and comes to similar conclusions.

"Your Coffee Shop Doesn't Use Two-Phase Commit": Should You?

Briefly, two-phase commit can be costly from a performance standpoint - in terms of both end-to-end transaction time and throughput - and you need to make sure you cost-justify your decision to use it when you do. On a finer point of detail, I would take issue with the implication in the article that there is a straight choice between synchronous processing and two-phase commit – most of the transaction systems I have ever seen involve access to one database and one asynchronous reliable queue in the context of a single transaction – thus guaranteeing that some updates happen *and* a message is guaranteed to be delivered to the next step in the process at some time in the future as an atomic unit, but that's another story.

So, back from my digression, the reason I wanted to write a follow-on to this article was because it seemed to me that the article illustrates a good point about the benefits of transactions and transaction management – particularly in the broadest sense of the term. To briefly summarize it so that the rest of my article makes sense, the thesis is that Starbucks uses an asynchronous model for accepting coffee requests into its main business system (the barista) in order to maximize the potential throughput of coffee from the shop, and hence maximize revenue. The potential cost of this optimization to the "happy day" scenario – the assumption that all is going forward with no errors – is the odd need to pause and hold up the line while wrong drinks are disposed of and remade, or money is refunded to unhappy (and still thirsty) punters. Indeed, if this is the only cost of this assumption, then the asynchronous case is clearly the correct design – throughput is maximized and the cost of unwinding the odd failure is outweighed by the less costly, less complex system we have built.

However, back in the real world away from analogy land, this is often a better "project phase one" argument than a "deployment lifetime" argument. To return to the analogy by way of illustration, imagine that our coffee production line is now in place and we are happily raking in money for strangely named coffees, and suddenly we get a good idea: How about improved quality of service for regular customers? For those really important coffee drinkers – rendered hyper-impatient by the caffeine they are wired with – we want to tell them how long it will be until they can expect their coffee. This poses a problem for us. From the moment a name got written on the empty coffee cup and it got queued on top of the coffee machine, we lost track of it. We are relying on the customer to hang about and wait

Author Bio:
Peter Holditch joined BEA as a consultant in the Northern European Professional Services organization in September 1996. He now works as a pre-sales architect in the UK. Peter has a degree in electronic and computer engineering from the University of Birmingham. When he's not pre-selling architecture, he likes to build furniture, brew beer, and enjoy the long hot British summers. Peter can be reached at peter.holditch@bea.com.

Contact:
peter.holditch@bea.com

until his name is called before customer and cup (now replete with coffee) are reunited. To track the cup in the queue, we will need to adopt a new strategy – maybe we have a yellow cup for the priority customer, so we can see the yellow cups processing up the queue. Well, that's fine until we have too many privileged customers, when we will no longer be able to distinguish *which* yellow cup we are looking for. What now? Suddenly we start to wish that we had a nice synchronous coffee production process, so we knew where we stood; of course, the analogy is starting to creak here a bit – the time it takes to produce a coffee is relatively short, and there is only one business system in the process, so this is not really that great an issue (which is a pity, because I was just about to propose attaching RFID tags to each cup, to allow us to associate them with their intended recipients…), but there is a core of truth here.

Attach an RFID Tag to Each Cup

Because of the apparent simplicity of asynchronous queuing-based systems, they are widely deployed. Additionally, a widely felt pain they create is the lack of visibility into "in-flight" business transactions. There is a clear and present demand for "executive dashboard" type facilities, so that managers have some insight into how the systems for which they are responsible are running – and some ability to foresee and forestall problems. This usually causes the nice simple phase one MOM-based systems to be glued to a lot of information gathering infrastructure (usually, more queues) with some kind of event correlation machinery on the back end to give an indication of what is passing through the system, and what looks unusual or possibly erroneous. Where did that original elegant simplicity go?

One oft-overlooked advantage of building systems with a transaction manager coordinating them is that we get an out-of-the-box central place that we can go to see which business event has touched which resource, and what the outcome is expected to be.

Of course, that said we still do have the bottleneck that the enforcement of ACID properties places on our throughput (chiefly, because of the database locks that ACID implies). It is here that I can mount another of my favorite hobbyhorses. One way to get the benefits of central coordination of transactions (in the loose sense, simply meaning correlated business activities) without incurring the penalties of data locking and contention is to relax the (technical) strictures of the ACID rules and allow a more easygoing, business reality–focused view of how the transactions correlate the business events. This is the idea behind a "next-generation" transaction concept such as cohesions, which first surfaced in the OASIS BTP standard and are now informing the debate within the WSBusinessActivity proposed standard.

Have Your Coffee and Drink It Too

Maybe, when these transaction standards mature, we will at last be able to have our coffee and drink it too?

References

- Hohpe, G. "Your Coffee Shop Does Not Use Two Phase Commit." www.eaipatterns.com/articles.html

›Linux ›Java ›Web Services ›.NET ›XML ›Wireless ›Storage ›Security
The Premier Resource for Today's Corporate & IT Decision Makers
IT solutions
›GUIDE
WWW.SYS-CON.COM
Top IT Solution Providers!
› Candle
› ClearNova
› Computer Associates
› Creative Science
› OralaSoft
› GraphOn
› InterSystems
› ITKO
› Merant
› O'Reilly
› Parasoft
› QuadBase
› SAP
› Wily Technology
› Zero G
TECHNOLOGIES YOU NEED NOW!
How to Manage Your Ideas Using Today's i-Technologies

Integrating an ASP.NET Application with J2EE PART 2

PLATFORM INTEROPERABILITY

By Blair Taylor

In Part 1 of this two-part article I discussed interoperability between .NET and Weblogic 8.1 and issues that arise when transferring data between Web serivces. In Part 2 I will cover some advanced issues: SOAP exception handling, uploading binary information via Web services, and application navigation and workflow.

SOAP Exception Handling

In non-Web Services environments, we're used to building a hierarchy of exceptions and throwing the appropriate exception to indicate the correct feedback to the client. For example, we can define an application exception called "UserNotLoggedInException" (that extends System.ApplicationException) to indicate to the caller that the application has failed because the user isn't logged into the application properly. When the calling method catches this exception, it's aware of the type of error that's occurred.

Exception handling in Web Services differs from the structured exception handling offered in Java or C#, where a developer puts code that can cause an exception in a try/catch clause. Exceptions are caught in the catch clause or bubble up to the calling method if not handled in one of the catch clauses. When an exception is thrown to a calling method, the method understands the data type of the exception and can deal with it accordingly.

With Web Services, exception handling gets a bit more complicated. All exceptions thrown in a Web Service are wrapped by .NET (or WebLogic) as a SoapException. A SoapException generates a SoapResponse that replaces the body of the SOAP message with a SoapFault element. A SoapFault is an XML element that contains elements with information indicating the SoapFault type, an error number, an error message, and an error source. The SoapFault also indicates a URI, which is usually the method that generated the exception. Table 1 displays the layout of a SoapFault. The SoapFault element provides information to the calling method about the exception that occurred.

.NET wraps native exceptions in a SoapException because the Web Service client doesn't understand the native type of the exceptions that occurred in the Web Service. In the last example, a client application calling a Web Service that threw a UserNotLoggedInException exception wouldn't understand this exception class and wouldn't know how to handle it properly. So a Web Service client will always need to handle a SoapException and may need to process the exception further to determine the root of the problem.

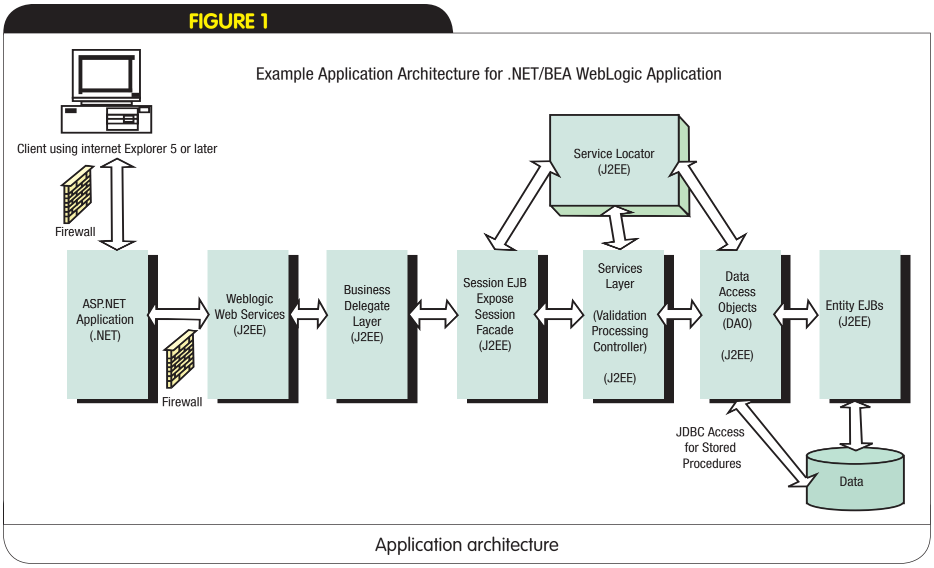
The faultcode element is usually set to either "client" or "server." The faultcode is the mechanism that a Web Service can use to indicate to the caller if the source of the error was a server error or a client error. For example, if the application wasn't able to connect to the database, the SoapFault should indicate a (namespace qualified) faultcode element of "server." It indicates that the server encountered an error and the error wasn't dependent on client input. In this case, you would log the error on the server, and return an incident number to the client. The client would display a generic user-friendly error page that indicates to the user that an error occurred and provide an incident number to contact the help desk. The help desk could then review the incident details and attempt to diagnose the problem.

When an error occurs that's dependant on client input, additional processing has to occur. An example of this occurs when the client attempts to access a resource when they are not logged in. In this case, it doesn't make sense to display a generic user-friendly error page; we'd like to let the user know the cause of the error so he can correct the situation. In this case, we would like to redirect the user to a login page so he can be authenticated. A similar (and more likely) scenario occurs when a user attempts to violate a foreign key relationship. For example, a user may try to delete a client who has orders. In this case, we'd like to notify the client via a message that the client can't be deleted because they have dependant data (orders). In both cases, we need to provide additional processing.

Author Bio:
Blair Taylor is president of JustWebSolutions.com, a Canadian company specializing in the architecture and development of distributed systems. Blair has authored several publications covering client-server and distributed technologies and is certified in both Java and .NET technologies. Blair can be reached via e-mail at feedback@justwebsolutions.com or at www.justwebsolutions.com.

Contact:
feedback@justwebsolutions.com

FIGURE 1


Example Application Architecture for .NET/BEA WebLogic Application
Client using internet Explorer 5 or later
Firewall
ASP.NET Application (.NET)
Firewall
Weblogic Web Services (J2EE)
Business Delegate Layer (J2EE)
Session EJB Expose Session Facade (J2EE)
Service Locator (J2EE)
Services Layer (Validation Processing Controller) (J2EE)
Data Access Objects (DAO) (J2EE)
Entity EJBs (J2EE)
JDBC Access for Stored Procedures
Data

Application architecture

TABLE 1

Soap Element	Description
faultcode	This indicates the source of the error. It is a qualified name within the http://schemas.xmlsoap.org/soap/envelope namespace. Possible options are VersionMismatch, MustUnderstand, Client, Server.
faultstring	Indicates a description of the error
faultactor	Provides information about what caused the error. This is represented by a URI.
detail	This is a user-defined element which can be used to contain detailed error information.

SoapFault Elements

To provide additional information to the Web Services client, we generate and throw a SoapException and set the SOAP fault type to “client.” This overrides the generation of a default SoapException by .NET (or WebLogic) and returns our custom SoapException. By setting the faultcode to “client,” we notify the caller that the error was caused by client input or action.

The SOAP specification indicates that a SoapFault contains a “details” element. The contents of the details element are left open to the developer to define the error information they’d like to communicate to the client. The first step is to define an XML schema to declare the structure of the SoapFault detail element. This is the contract between the Web Service and client that indicates the data that will be passed when a custom SoapException is thrown.

When an exception occurs, we want to pass a message to display to the client. We also want to pass the server exception message and a stack trace to log for assistance in diagnosing the production problems. Listing 1 shows the XML schema we defined to pass this information via a custom SoapException. Listing 2 shows the SoapException XML generated.

In our sample application, we throw a custom SoapException from a WebLogic Web Service, catch, and process it in an ASP.NET application. The throwClientSoapException method throws a custom exception. The code for the throwClientSoapException method is shown in Listing 3.

The code calls the buildSoapFaultException method in the SOAPUtil class. This is a static helper method that builds the custom SoapException (see Listing 4).

The SOAP_NAMESPACE is defined as a Java constant:

```
 public static final String SOAP_NAMESPACE =
“http://www.justwebsolutions.com/articles”;
```

The ASP.NET client code catches the SoapException and then creates a DataSet, and uses the SoapExceptionDetail.xsd schema we defined to read the XML schema into the DataSet (via the ReadXmlSchema method).

```
//Get SoapExceptionDetail.xsd full path
string schemaRelativePath = “../schemas/SoapEx-
ceptionDetail.xsd”;
string schemaFullPath = HttpContext.Current.
Server.MapPath(schemaRelativePath);

//Read the Schema into the DataSet
dsError.ReadXmlSchema(schemaFullPath);
```

The application then loads the XML string passed from the Web Service into the DataSet via the ReadXml method.

```
//Get exception detail xml string
string xmlString = ex.Detail.OuterXml;

//Read the xml string into a StringReader object
StringReader stringReader = new
StringReader(xmlString);

//Read the StringReader into the XmlReader
object
XmlReader xmlReader = new XmlTextReader(string
Reader);

//Read the Xml into the DataSet
dsError.ReadXml(xmlReader, XmlReadMode.Ignore-
Schema);
```

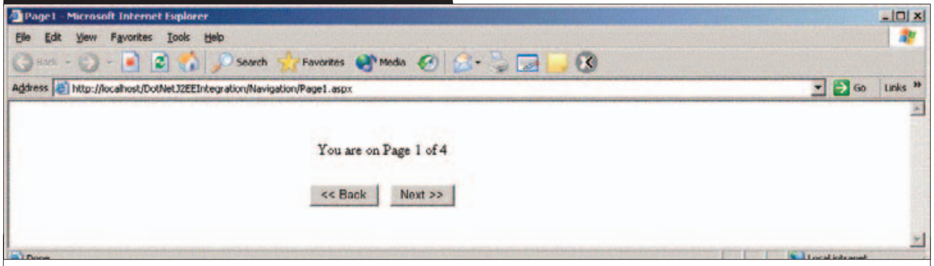
The client error message is then accessible from the DataSet and is displayed in the txtSOAPClientMessage text.

```
// Create an instance of  DataView object
dataView = new DataView();

// Specify the table
dataView.Table = dsError.Tables[“Error”];

// Get the first row
DataRow dsErrorRow = dataView.Table.Rows[0];

// Get the relevant Error message
message = dsErrorRow[“ClientErrorMessage”].
ToString();
txtSOAPClientMessage.Text = message;
```

The complete code for this processing is shown in Listing 5.

There’s one additional issue to consider when processing a SoapException thrown via a WebLogic Web Service. If you choose to expose a Session EJB as a Web Service, there’s an additional caveat. Runtime (non-checked) exceptions thrown in Session EJBs are packaged by the EJBContainer as an EJBException before being thrown to the caller. This is mandated by the J2EE specification. EJBExceptions in a Web Service are automatically processed by WebLogic and wrapped using default

FIGURE 2


Page1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Go
Links
<< Back
Next >>

Address http://localhost/DotNetJ2EEIntegration/Navigation/Page1.aspx

You are on Page 1 of 4

Page 1

SoapException processing. It doesn't let you override this processing to generate a custom SoapException. It's one of the reasons we used a business delegate layer (see Figure 1). It lets us throw our exception as a custom application exception in WebLogic, and in the business delegate class (which is a simple Java class), it allows us to catch the application exception, generate, and throw a custom SoapException.

Uploading binary information

In our application we needed to let users upload binary images to our site. The images then had to be available to other users of the application. The images contained confidential information for our clients, so we needed to ensure that the images couldn't be viewed easily in the Web application by manipulating the URL. As part of out business analysis for the site, we viewed the sites of some of our client's competitors and were surprised to see that we could often guess the names (and sequence numbers) of the images and could see information that was private and confidential.

So we stored the images on our WebLogic server (on the file system) and streamed the binary contents of the file via a Web Service. The client, an ASP.NET Web page, would gather the file's binary contents and stream them to the client browser.

The first issue we needed to resolve was the technology we were going to use to send the binary information via Web Services. There were two different alternatives: send the data via a binary data type or via the SOAP With Attachments protocol. We did some research and found an interesting paper written by several Microsoft and BEA architects at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/infoset_whitepaper.asp. The paper advocates encoding the data via a binary data type such as xs:base64binary or xs:hexBinary because of security issues with out-of-band attachment data as well as the fact that SOAP Messages With Attachments don't conform to the Basic Profile 1.0 (BP) published by the Web Services Interoperability Organization (WS-I). That raised some potential performance issues with using text representations of binary code, but, since we weren't transferring a lot of binary data, optimal performance wasn't the most important issue. Based on that white paper, especially given the authors and the equal representation of Microsoft and BEA, we decided to use binary data types to send image data in our Web Service rather than using the SOAP With Attachments API. Once we established our approach, we needed to map out the steps. We would use the .NET upload HTML control to send the binary data to the Web server. Once the user uploaded the image to the Web server, the application needed to do the following in the ASP.NET application:

1. Check that the image extension was valid.
2. Define a temporary file name for the image and call the fileInput.PostedFile.SaveAs method to save the file to a temporary area on the Web server.
3. Check the size of the image to make sure it wasn't beyond the limit allowed.
4. Convert the image to a byte array.
5. Call the Web Service passing the byte array.
6. Delete the temporary file.

Listing 6 shows the ASP.NET code that did these steps. It's pretty straightforward so I won't go through it in detail.

The Web Service code in BEA's WebLogic needed to get the byte array, create a file on the file system from it, and store the name of the image in the database so the image could be viewed in the future.

Listing 7 shows the Java code that did these steps. In the example we simply stored the file on the application server's file system. In our production application, we added a couple of additional steps to obfuscate the file name and to store the files in a hierarchical structure on the file system (rather than in the same directory) to make file system navigation and lookup more efficient.

To view the images, the process is also pretty straightforward. The WebLogic Web Service opens the file based on the file name passed as a parameter. The Web Service then gets the binary data from the file and returns it via a binary data type. The ASP.NET client code calls the Web Service, sets the content type for the Web page (via Response.ContentType) based on the file extension and then sends the binary stream via the Response.BinaryWrite method. The WebLogic code is shown in Listing 8 while the ASP.NET code is shown in Listing 9.

Application Navigation and Workflow

One of the areas in which many Web applications are difficult to maintain is application workflow. I define application workflow as the way in which applications validate data and handle navigation. One of the common issues we've seen in ASP and ASP.NET applications is that there's a lot of page navigation logic buried inside code – behind pages or ASP scripts – and changing the location of the name of a Web page causes broken links in the application. J2EE applications commonly take advantage of an open source framework called "Struts." Struts provides a number of useful features:

- Struts abstracts page navigation into XML configuration files. Pages are referenced by name rather than URL so that URLs can be easily changed in the XML configuration file.
- Struts encourages development using a MVC (Model-View-Controller) architecture.
- Struts abstracts HTTP response data into separate classes ("ActionForm" classes) that are simple classes with attributes that match the response parameters and accessor/mutator (get/set) methods.
- Struts provides a main controller class (servlet) to which every page will submit. The controller class abstracts the HTTP response data into ActionForm classes, delegates control to a business class (Action class) that applies complex validations, and determines the next page to access.

Having built a few successful J2EE applications, we wanted to capture the best elements of the Struts framework for our ASP.NET application. Some Struts features are already built into ASP.NET. For example, ASP.NET provides a number of validation controls (RequiredFieldValidator, CompareValidator, RangeValidator, Regular ExpressionValidator, and CustomValidator) that assist in validating form on an ASP.NET page. Struts also provides page validation framework support and a validation summary control type.

While Struts offers a central controller to which all pages must submit, ASP.NET offers a different model in which pages submit back to themselves. This enables the event-driven programming model and page state (ViewState) features in ASP.NET. The two architectures are quite different. After taking a look at the options, we felt that developing a central controller similar to Struts would take a great deal of effort and the result would be a less optimal architecture. So we decided not to develop a central controller like Struts.

Several benefits of Struts are predicated on the submission of each Web page to a main controller class and wouldn't work well in an ASP.NET environment. Specifically, the abstraction of HTTP request data into separate classes and the delegation to a business processing and validation class were dependent on the submission to a main controller class. Besides, delegation to a business processing class was something we were already doing in an ASP.NET environment. One of the features of Struts that we felt we could adapt relatively easily that didn't rely on the submission of each Web page to a main controller class was the abstract page navigation available in Struts.

Abstracting Page Navigation

When developing an application under Struts, a developer defines an XML configuration file that indicates absolute URL addresses for Web pages and assigns a logical name for the page. In an ASP.NET application, the logical name "LoginPage" can be assigned to "http://myserver/myapplication/Login.aspx." Developers can code Response.Redirect statements and URL links using the logical name rather than the absolute URL address. Should the page name or location have to change in the future, a simple change to the XML configuration file is all that's needed.
For example, the code to redirect to the login page would be:

```
Response.Redirect(getPageURL(Page.Login));
```

To implement this concept in ASP.NET, we implemented a class called "NavigationController." It would read and cache the page name and URL information stored in an XML configuration file (NavigationConfig.xml), offer helper methods to return the absolute page URL when passed the logical name, and provide generic methods to redirect forward and backward in the application.

The example application provides a simple example of using the NavigationController. The page navigation information is stored in the NavigationConfig.xml file:

```
<?xml version=”1.0” encoding=”utf-8” ?>
<!--The order of items determine the sequence in
which they appear in the module -->
<MyApplication>
  <MyModule>
         <Page Name=”Page1” Url=”~/Navigation/
Page1.aspx”/>
         <Page Name=”Page2” Url=”~/Navigation/
Page2.aspx”/>
         <Page Name=”Page3” Url=”~/Navigation/
Page3.aspx”/>
         <Page Name=”Page4” Url=”~/Navigation/
Page4.aspx”/>
  </MyModule>
</MyApplication>
```

The NavigationController class reads the NavigationConfig.xml file and builds a collection of PageConfiguration objects. Each PageConfiguration object contains the navigation information for a page. A Hashtable collection of PageConfiguration objects is built and the Hashtable collection is stored in the ASP.NET Application object.

There are four pages, Pages 1-4, all with back and next buttons. The back button executes the RedirectBack method in the NavigationController class. The next button executes the RedirectForward method in the NavigationController class.

The code for the RedirectForward method is shown in Listing 10.

The current page for the application is stored in the ASP.NET Session object for each user. This is set to the first page of the application ("Page1") via code in the

Article Information Box

List of Prerequisites

Some knowledge of Java and J2EE or BEA Weblogic
Some experience with ASP.NET and XML Web services
Some knowledge of XML and XML schemas

Level of Difficulty

2 (of 3)

Summary

This article discusses some of the issues that arise when using .NET with a J2EE application server such as BEA Weblogic. The tools used are Visual Studio.NET 2003 and BEA Weblogic Server 8.1. The article discusses several issues: integration between the two technologies; how to transfer data from Weblogic to ASP.NET so that data can be easily bound to ASP.NET components; proper exception handling within Web services; and how to store and retrieve binary data. The article also looks at the Apache Struts framework and how navigation within an ASP.NET application can be configured via an XML file in a manner similar to the Struts framework.

Background Information

- BEA Weblogic Web services: http://edocs.bea.com/wls/docs81/webserv/index.html
- XMLBeans: http://dev2dev.bea.com/technologies/xmlbeans/index.jsp
- XML technologies: www.w3.org/XML/, www.w3.org/XML/Schema
- SOAP tutorial: www.w3schools.com/soap/default.asp
- The Apache Struts Web Application Framework: http://jakarta.apache.org/struts/

Related Articles

- MSDN Web Services Development Center: http://msdn.microsoft.com/webservices/
- XML, SOAP and Binary Data whitepaper on MSDN: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/infoset_whitepaper.asp
- The XML Files - A Quick Guide to XML Schema: http://msdn.microsoft.com/msdnmag/issues/02/04/xml/default.aspx
- Enterprise Interoperability: .NET and J2EE: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/dotnetinteroperability.asp
- Using Soap Faults: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service09172002.asp

Page_Load method of the Page1.cs.aspx. Each page has a PageConfiguration object stored in the Hashtable. The PageConfiguration object is aware of the current page, the previous, and next pages in the application navigation flow. If the next page is valid, the URL for the page is obtained and the NavigationController class redirects the application to the next page.

Listing 11 contains the entire code from the NavigationController class and the PageConfiguration class.

The NavigationController class in its current incarnation works well for a simple wizard-based application in which the navigation is relatively simple. To support more complex navigation, you'd need additional processing to delegate to a business class that provides complex validations, performs business calculations, and determines the next page that the application should navigate to based on business rules.

Summary

In this article I have discussed the integration of an ASP.NET client with a J2EE (WebLogic) middle-tier using Web Services. I discussed some of the issues that arise in this type of hybrid environment and how we addressed them. The application we developed using the technologies and techniques discussed here is now in production and the general feedback from the developers, the client, and users is very positive. I hope you found this article useful and, as always, I would love to hear from other developers and architects with ideas and suggestions based on their experiences.

Listing 1: SoapExceptionDetail

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefa
ult="unqualified">

   <!-- used in soap exception handling to define detail node -->

   <!-- global definitions -->
   <xs:element name="ClientErrorMessage" type="xs:string"/>
   <xs:element name="ServerErrorMessage" type="xs:string"/>
   <xs:element name="StackTrace" type="xs:string"/>
   <!-- Error element -->
   <xs:element name="Error">
         <xs:complexType>
                  <xs:sequence>
                           <xs:element ref="ClientErrorMessage"/>
                           <xs:element ref="ServerErrorMessage"/>
                           <xs:element ref="StackTrace"/>
                  </xs:sequence>
         </xs:complexType>
   </xs:element>
</xs:schema>
```

Lisiting 2: Soap Error Message

```
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <env:Header>
 </env:Header>
 <env:Body>
  <env:Fault    xmlns:fault="http://schemas.xmlsoap.org/soap/enve-
lope/">
   <faultcode>fault:Client</faultcode>
   <faultstring>This is a custom soap exception</faultstring>
   <faultactor>Client</faultactor>
   <detail>
    <Error>
     <ClientErrorMessage>This is a client error message.</ClientEr-
rorMessage>
     <ServerErrorMessage>This would be a server error message if the
fault type was 'server'.</ServerErrorMessage>
     <StackTrace>This is a stack trace.</StackTrace>
    </Error>
   </detail>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

Listing 3

```
// build and throw a client SOAP exception
    QName faultCode = new QName("http://schemas.xmlSOAP.org/SOAP/enve-
lope/", "Client");
    String faultString = "This is a custom SOAP exception";
    // set fault type to client to indicate the client operation caused
this exception
    String faultActor = "Client";
    throw SOAPUtil.buildSoapFaultException(faultCode, faultString,
faultActor,
                                            "This is a client error mes-
sage generated from the throwClientSoapException method in WebLogic.",
"This would be a server error message if the fault type was 'server'.",
"This is a stack trace.");
```

Listing 4

```
// define the detail element
    Detail detail = weblogic.webservice.util.FaultUtil.newDetail();

    try {
      // define the xml structure for the detail element
      Name name = SOAPFactory.newInstance().createName("Error", "",
SOAP_NAMESPACE);
      DetailEntry entry = detail.addDetailEntry(name);
      // add the values to the detail element      entry.addChildElemen
t("ClientErrorMessage").addTextNode(clientErrorMessage);
      entry.addChildElement("ServerErrorMessage").addTextNode(serverEr
rorMessage);
      entry.addChildElement("StackTrace").addTextNode(stackTrace);
    }
    catch (Exception ex) {
      Logger.getLogger("GenerateSoapException.class").warning("An
exception occurred in buildSoapFaultException of SOAPUtil. Exception
message is: " + ex.getMessage());
      throw new RuntimeException(ex);
    }

    // return the SoapFaultException
    return new SoapFaultException(faultCode, faultString, faultActor,
detail);
```

Listing 5: ASP.NET client SoapException code

```
private void bnSoapError_Click(object sender, System.EventArgs e)

{
weblogic.BusinessDelegate proxy = new weblogic.BusinessDelegate();
try
   {
   proxy.throwClientSoapException();
   }
catch (SoapException ex)
   {

   string message = "";
```

```
    //Create a DataSet instance
    DataSet dsError = new DataSet();

    //Create a DataView instance;
    DataView dataView = new DataView();

    //Get SoapExceptionDetail.xsd full path
    string schemaRelativePath = “../schemas/SoapExceptionDetail.xsd”;
    string schemaFullPath = HttpContext.Current.Server.MapPath(schemaRe
lativePath);

    //Get exception detail xml string
    string xmlString = ex.Detail.OuterXml;

    //Read the xml string into a StringReader object
    StringReader stringReader = new StringReader(xmlString);

    //Read the StringReader into the XmlReader object
    XmlReader xmlReader = new XmlTextReader(stringReader);

    //Read the Schema into the DataSet
    dsError.ReadXmlSchema(schemaFullPath);
    //Read the Xml into the DataSet
    dsError.ReadXml(xmlReader, XmlReadMode.IgnoreSchema);

    // Create an instance of  DataView object
    dataView = new DataView();

    // Specify the table
    dataView.Table = dsError.Tables[“Error”];
    // Get the first row
    DataRow dsErrorRow = dataView.Table.Rows[0];

    // Get the relevant Error message
    message = dsErrorRow[“ClientErrorMessage”].ToString();

    txtSoapClientMessage.Text = message;

}

}
```

Listing 6: ASP.NET code to perform a binary transfer

```
// check the extention of the file
if(!this.CheckFileExtensionIsValid(fileInput.Value))
{
    lblError.Text = “Only files with a .jpg, .jpeg and .gif extention
may be uploaded.”;
    return;
}

// Ceate the temporary path
    // DIRECTORY C:\TEMP MUST EXIST
    string imageFileName = System.IO.Path.GetFileName(fileInput.Value);
    string tempImagePath = “c:\\temp\\” + imageFileName;

    //Save it to the server as a temporary file before processing
    fileInput.PostedFile.SaveAs(tempImagePath);

    //Get the maximum image file size in bytes
    int maxImageSize = 2097152;

    //Do not allow images larger than the maximum size
    if(fileInput.PostedFile.ContentLength > maxImageSize )

    {
            lblError.Text = “Can not upload a file larger than “ + maxIma-
geSize + “ bytes”;
            return;
    }


    //Get file and convert it to an array of bytes
    System.Drawing.Image image = System.Drawing.Image.
FromFile(tempImagePath);
    MemoryStream memoryStream = new MemoryStream();

    switch(fileExtention)
    {
            case “gif”:
            {
                        image.Save(memoryStream, ImageFormat.Gif);
                        break;
            }
            case “jpg”:
            case “jpeg”:
            {
                        image.Save(memoryStream, ImageFormat.Jpeg);
                        break;
            }
    }

    long  length = memoryStream.Length;
    byte[] byteArray = new byte[length];
    byteArray = memoryStream.ToArray();

    // Discard Image
    if(image != null)
    {
            image.Dispose();
    }

    //Flush Memmory Stream
    if(memoryStream != null)
    {
            memoryStream.Flush();
    }

    // Delete the temp file
    File.Delete(tempImagePath);

    // send via web service to save
    weblogic.BusinessDelegate proxy = new weblogic.BusinessDelegate();
    bool successFlag = proxy.saveBinaryData(imageFileName, byteArray);
    if (successFlag)
    {
            lblError.Text = “The file was successfully uploaded.”;
    }
    else
    {
            lblError.Text = “The file was NOT successfully uploaded.”;
    }
}
```

Listing 7: WebLogic code for binary upload

```
public static boolean saveBinaryData(String fileName, byte[] fileData) {

    Logger.getLogger(“UploadImages.class”).info(“Entering saveBinary-
Data method. fileName = “ + fileName);

    boolean bResult = false;

    // for this example, we will store all images in the same directory
    // in a production environment, you may which to use subdirectories
based on client id
    // or a similar id

    // check that the directory exists, if not create it
    try {

      File attachmentDir = new File(imageDirectory);
      if (!attachmentDir.exists()) {
        // create directory
        boolean bDirCreated = attachmentDir.mkdirs();
        if (!bDirCreated) {
          throw new IOException(“Unable to create directory: “ + at-
tachmentDir.getAbsolutePath());
        }
      }

      // save the binary data to a file on the file system
      String fullFileName = imageDirectory + “\\” + fileName;
      File attachmentFile = new File(fullFileName);
      FileOutputStream fileStream = null;
      fileStream = new FileOutputStream(attachmentFile);
      fileStream.write(fileData);
      // close filestream
      fileStream.close();
      bResult = true;
    }
    catch (IOException ex) {
      Logger.getLogger(“UploadImages.class”).warning(“An IOException
occured in saveBinaryData. The message is: “ + ex.getMessage());
      throw new RuntimeException(ex);
    }
    return bResult;

  }
```

Listing 8: WebLogic code to return binary image data

```
public static byte[] getBinaryData(String fileName) {

    Logger.getLogger(“ViewImages.class”).info(“Entering viewApprais-
alAttachment method. fileName = “ + fileName);

    String imageDirectory = UploadImages.imageDirectory;

    // retrieve the file
    String fullFileName = imageDirectory + “\\” + fileName;
    File imageFile = new File(fullFileName);
    FileInputStream fileStream = null;
    try {
      fileStream = new FileInputStream(imageFile);
    }
    catch (FileNotFoundException ex) {
      Logger.getLogger(“ViewImages.class”).info(“FileNotFoundException
occured. fileName = “ + fileName + “. Message is: “ + ex.getMessage());
      throw new RuntimeException(ex);
    }

    int fileLength = (int) imageFile.length();
    byte[] fileData = new byte[fileLength];
    try {
      fileStream.read(fileData);
      // close filestream
      fileStream.close();
    }
    catch (IOException ex) {
      Logger.getLogger(“ViewImages.class”).info(“IOException in viewAp-
praisalAttachment occured. fullFileName = “ + fullFileName + “. Message
is: “ + ex.getMessage());
      throw new RuntimeException(ex);
    }

    return fileData;

  }
```

Listing 9: ASP.NET client code to display binary data as an image within Intenet Explorer

```
private void Page_Load(object sender, System.EventArgs e)
{

    object objFileName = Request.QueryString[“FileName”];
    if (!(objFileName == null || objFileName.ToString() == “”))
    {
          string fileName = objFileName.ToString();
          string fileExtension = fileName.Substring(fileName.IndexOf(“.”)+
1).ToLower();

          // call web service to obtain image array of bytes
          weblogic.BusinessDelegate proxy = new weblogic.BusinessDel-
egate();
          byte[] byteArray = proxy.getBinaryData(fileName);

          switch (fileExtension)
          {
                    case “jpeg”:
                    case “jpg”:
                    {
                              Response.ContentType = “image/JPEG”;
                              break;
                    }
                    case “gif”:
                    {
                              Response.ContentType = “image/gif”;
                              break;
                    }
          }

          Response.BinaryWrite(byteArray);

          }
    else
    {
          Response.Write(“Please specify a valid filename in the query
string”);
    }
}
```

Listing 10

```
// get page configuration
string currentPage = HttpContext.Current.Session[“CurrentPage”].
ToString();
Hashtable hashPageConfig = (Hashtable) HttpContext.Current.Application[
“PageConfig”];
PageConfiguration pageConfiguration = (PageConfiguration) hashPageConfig[c
urrentPage];

// redirect forward if not on the first page
if (pageConfiguration.NextPageName != “”)
{
string urlNext = NavigationController.getPageURL(pageConfiguration.
NextPageName);
HttpContext.Current.Session[“CurrentPage”] = pageConfiguration.NextPa-
geName;
    HttpContext.Current.Response.Redirect(urlNext);
}
```

Listing 11 and the source code for this article are available at weblogic.sys-con.com.

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Blog-n-Play.com	www.blog-n-play.com	201-802-3000	22
EV1 Servers	www.ev1servers.net	800-504-SURF	11
HP	http://devresource.hp.com/d2d.htm	800-752-0900	15
Information Storage & Security	www.issjournal.com	888-303-5282	35
Intersperse	www.intersperse.com	800-340-4553	2, 17
IT Solutions Guide	www.sys-con.com	201-802-3021	37
JavaOne	java.sun.com/javaone/sf.		21
JDJ	www.sys-con.com/jdj	888-303-5282	31
LinuxWorld Magazine	www.linuxworld.com	888-303-5282	45
Motive	www.motive.com/within1	512-531-2527	52
NetIQ	www.netiq.com/solutions/web	408-856-3000	51
Parasoft	www.parasoft.com/soaptest_wdj	626-256-3680	3
Quest Software	www.quest.com/wldj	949-754-8633	7
SandCherry	www.sandcherry.com	866-383-4500	19
SYS-CON e-newsletters	www.sys-con.com	888-303-5282	35
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	47
SYS-CON Reprints	www.sys-con.com	201-802-3026	22
Tangosol	www.tangosol.com	617-623-5782	13
Web Services Journal	www.wsj2.com	888-303-5252	23
Wily Technology	www.wilytech.com	888-GET-WILY	5
WLDJ	www.sys-con.com/weblogic/	888-303-5282	49

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

WebLogic Blog Round-Up

AROUND THE WEBLOGIC WORLD IN 80 BLOGS

By News Desk

If one of the true signs of a vibrant developer community is an active blogosphere surrounding a technology, then the WebLogic family of technologies certainly passes that test with flying colors. In case you're not yet active yourself, WLDJ brings you a comprehensive selection from some of the best known (and many of the not so well-known too). Don't forget that you can blog yourself now, too, at the WLDJ site – you can get started in just three minutes at http://blog-n-play.com.

Blog Topic: Advancements in Java Productivity

By Jim Revera
(http://dev2dev.bea.com/blog/jim_rivera/)

Lately I've been spending some time taking a look at Spring (www.springframework.org/). For those of you who are not familiar with it, Spring is a popular open source application framework that attempts to address many of the issues that make J2EE application development difficult. It provides a "lightweight container" that helps to simplify development, make testing easier, and encourages best practices to make applications more flexible and maintainable.

An interesting productivity study (www.xebia.com/oth_publications_java_with_spring_just_as_productive_as_4gl.html) was recently performed at the 2005 Dutch RAD Race that concluded developers can be just as productive using Spring and Java as they can be with a 4GL RAD tool. This is very encouraging for the Java community, which has long struggled with developer productivity issues. What I really like about Spring is that it provides a nice, consistent architectural approach without masking the strengths of the underlying platform. For example, when deploying a Spring application on WebLogic Server, you still have access to WLS distributed transactions, clustering, high-speed messaging, legacy integration, advanced Web services, and advanced application management functionality... not to mention the inherent performance and scalability associated with WebLogic Server.

It's really very similar to what BEA pioneered with the WebLogic Workshop framework, now the core of Apache Beehive (http://incubator.apache.org/beehive/). Both frameworks have the same high-level goals of improving developer productivity on J2EE, both can be deployed on multiple platforms, and both leverage the dependency injection pattern to simplify application code (Martin Fowler has written a nice article that defines *dependency injection*). Of course, both frameworks have their own strengths and weaknesses upon closer examination.

In fact, if you look at the write-up from the Dutch RAD Race you'll see that the biggest issues they ran into during the competition were in developing the Web user interface... hmmmm, perhaps next time they should consider using Beehive PageFlows (http://incubator.apache.org/beehive/pageflow/getting_started.html) in the web tier. :)

Blog Topic: Books I'm Currently Reading

By Vinny Carpenter
(www.j2eegeek.com/blog/archives/2004_08_01_j2eegeek_archive.html /)

As a recurring feature, I post a list of books that I

Author Bio:
WLDJ News Desk trawls the world of e-commerce technologies for news and innovations and presents IT professionals with updates on WebLogic-related technology trends, products, and services.

Contact:
wldjeditorial@sys-con.com

A LIMITED TIME SAVINGS OFFER FROM *SYS-CON MEDIA*

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

CALL TODAY! 888-303-5282

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

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

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

□ **LinuxWorld Magazine**

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

□ **JDJ**

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

□ **Web Services Journal**

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

□ **.NET Developer's Journal**

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

□ **Information Storage + Security Journal**

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

□ **Wireless Business & Technology**

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

□ **MX Developer's Journal**

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

□ **ColdFusion Developer's Journal**

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

□ **WebSphere Journal**

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

□ **PowerBuilder Developer's Journal**

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

□ **WLDJ**

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

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

3-Pack

Pick any 3 of our magazines and save up to $210.00 Pay only $99 for a 1 year subscription plus a FREE CD

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

6-Pack

Pick any 6 of our magazines and save up to $340.00 Pay only $199 for a 1 year subscription plus 2 FREE CDs

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

9-Pack

Pick 9 of our magazines and save up to $270.00 Pay only $399 for a 1 year subscription plus 3 FREE CDs

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

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

SYS-CON MEDIA

am currently reading. I am a voracious book collector and (usually) reader as well. With a new baby who's fewer than two months old, reading time is at a premium, and so I've built up quite a backlog. Books I am currently reading include:

- *Expert One-on-One J2EE Development without EJB* by Rod Johnson, Juergen Hoeller
- *Tiger: A Developer's Notebook* By David Flanagan, Brett McLaughlin
- Better, Faster, Lighter Java By Bruce A. Tate, Justin Gehtland
- *Programming Jakarta Struts,* 2nd Edition by Chuck Cavaness
- *Enterprise Service Bus* By Dave Chappell
- *Tapestry in Action* by Howard M. Lewis Ship

Blog Topic: Strategies for WebLogic Domain Configuration

By Prakash Malani

(http://dev2dev.bea.com/blog/pmalani/)

I have created a fun, dynamic, and interactive presentation on various strategies for domain configuration based on two articles that I wrote for *WLDJ*. When I was contemplating such an article, I predicted that the article would be about 1,500 words. After all the research, investigations, and implementing the different strategies, I ended up with 10,000-word article!

Due to space and time constraints, I had to condense all the information down to two separate articles.

For the presentation, I boiled down all of the information into a few slides. The primary slides are just a few bullet points on the pros and cons of each different strategy. Before discussing the pros and cons, I presented a brief overview of concepts and terminology.

The presentation was very interactive with the attendees participating with questions and comments. The attendees asked good questions that directly pertained to the challenges facing them.

The presentation is available (see *note below re: membership) here: http://groups.yahoo.com/group/bartssandbox/files/WebLogicDomainConfigurationOptions.pdf. Many of the attendees wanted to play with the different strategies. The example I used in the article as well as the in the presentation is available here: http://groups.yahoo.com/group/bartssandbox/files/Automate_SEM.zip.

The example is very good because even though it is simple, it leverages many different types of resources such as JDBC connection pool, JDBC datasource, JMS connection factory, JMS store, JMS server, and JMS destination. The example also includes JUnit tests to verify that the domain is created and configured properly. I encourage you to download the example and take the different strategies for a test drive.

Before the presentation, I had created a poll on what you were using for domain configuration (http://groups.yahoo.com/group/LABEAUG/message/290). The poll results concur with my informal surveys that most of you are using WebLogic Console for domain configuration. Domain configuration with the console is manual, error-prone, tedious, and repetitive. Therefore, I encourage you to consider one of the automated solutions, such as WebLogic Server Scripting Tool (WLST), instead of manual configuration with WebLogic Console.

Drop me a note about what strategies you have been successful with as well as what challenges you have encountered.

*Note: Free membership to bartsandbox [www.bartssandbox.com/] is required to access the presentation, as well as the source code example.

Blog Topic: XMLHttpRequest

By Tom Janofsky

(www.tomjanofsky.com/blog.html)

So I've been doing a bit of work with XMLHttpRequest lately to do dynamic updates on Web application pages behind the scenes and type-ahead drop downs. I thought this demo (http://www.paper-mountain.org/demos/live/) was a great example and I've made some modifications for better IE support (z-order with drop downs on the page, deal with IE caching GET requests, and some minor interaction bugs).

Pretty cool stuff really. I'll post those updates soon.

Blog Topic: WebLogic Workshop on Mac OS X (from Linux Installer)

By Simon Brown

(www.simongbrown.com/blog/2004/07/21/weblogic_workshop_on_mac_os_x_from_linux_installer.html)

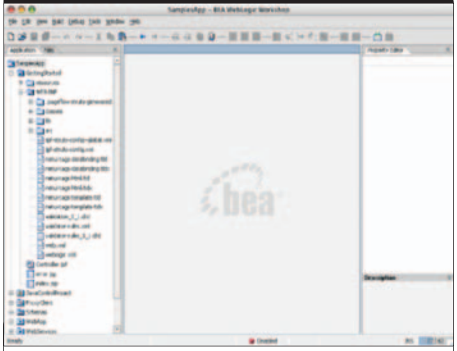
If you choose to install Workshop from the GUI installer, it gets installed underneath $BEA_HOME/weblogic81/workshop. In this directory is a script called Workshop.sh and this is used to run the tool. Unfortunately it doesn't run "as is" and some tweaking is needed. First of all, you'll need to "fix" your JDK (http://www.oreillynet.com/cs/user/view/cs_msg/31140) so that it looks like a typical install (with jre/bin/java and jre/lib/rt.jar). If you've ever looked at the Apple JDK then you'll know what I mean.

Next you need to edit that script and I've done this as follows. I basically changed the path to Java, some of the JVM options, and stopped output being redirected to dev/null (so I can see the stack traces when it blows up, which it hasn't yet).

```
java -Xmx256m -Xms64m -client
  -Djava.system.class.loader="workshop.core.
AppClassLoader"
  -cp "/Users/simon/bea/weblogic81/workshop/wlw-
ide.jar"
  workshop.core.Workshop
```

That's pretty much it. Run the script and Workshop will start up.

FIGURE 1



From Simon Brown's blog

Of course, since it's another Java application, WebLogic Builder also runs fine. I've loaded an application into these tools and I was able to deploy and redeploy from it just fine, editing the code and/or config as necessary. WebLogic on Mac is certainly a viable development platform for J2EE.

Blog Topic: Debug J2EE Apps Deployed on Weblogic with Eclipse

By Jeremy Whitlock

(http://dev2dev.bea.com/blog/jcscoobyrs/archive/2005/04/)

PREREQUISITES:

- Eclipse 3.0 – www.eclipse.org
- Weblogic 6.1/7.0/8.1 – www.dev2dev.com/wlserver
- Weblogic-Eclipse Plugin 1.1.1 – https://eclipse-plugin.projects.dev2dev.bea.com

GETTING STARTED

Once you have your environment properly setup (Eclipse installed, Weblogic-Eclipse plugin installed, and Weblogic installed with a working domain configured) we need to setup the Eclipse-Weblogic plugin.

Window > Preferences > Weblogic

- Select the "7.0 or higher" radio button for the Version
- Input your BEA Home
- Input your Weblogic Home
- Input your Domain Name
- Input your Domain Directory
- Input your Server Name
- Input your User capable of starting the server
- Input your Password
- Input your Hostname
- Input your Port

Window > Preferences > Weblogic > Classpath

- Input your required classpath entries. [Optional]

Window > Preferences > Weblogic > JavaVM Options

- Input your required Java VM Arguments. There are some required WebLogic arguments already predefined. [Optional]
- Input your JNI path. This is useful if you want to use the WebLogic native I/O libraries in $WL_HOME/server/bin. [Optional]

Window > Preferences > Weblogic > Project

- Input a reference to an existing Eclipse project with the source/classpath for your J2EE app that you have deployed to Weblogic. [Optional]

Some of the above mentioned as [Optional] are optional to make the plugin find your sources and such, but are not required for making the plugin start/stop WebLogic.

Now that we have everything setup, click the green triangle in the toolbar. This should start WebLogic. If you get errors they will show up in the Console and should be able to be fixed easily. Now deploy an application to WebLogic, if you haven't already, that has its corresponding source in an existing Eclipse project that you *should* have referenced in the WebLogic Preferences. Once it's deployed, open any of the .java files for the deployed application and put a breakpoint in the source by double clicking the gray bar to the left of your source document. You should be updated with a blue orb in the gray bar on the line you double clicked beside. Now run your J2EE app and when you hit that breakpoint, Eclipse should alert you and you can step through the J2EE application to debug. While brief, this should supply you with all you need to debug your J2EE apps deployed to WebLogic inside of Eclipse. If you need any help or run into problems let me know and we'll remedy them.

Blog Topic: WLShell for Quick Access to Data

By Mark Vaughn
(http://dev2dev.bea.com/blog/mvaughn25/archive/2005/04/)

Sure, both WLShell and WLST have a good bit of functionality. WLST may even have more, in some aspects. However, WLST is still lacking the GUI ability to graph resource usage in real time. I initially looked at WLShell as a tool to script and automate changes to WebLogic configurations. That would make migrations between environments and across clusters both more accurate and less time consuming. However, in the process of evaluating WLShell, I found the all-important "-g" option.

Sure, *"get /JVMRuntime/$SERVER/HeapFreeCurrent"* will get you instant access to information on your JVM Heap, and *"get -r 5 /JVMRuntime/$SERVER/HeapFreeCurrent"* will get the same information and update it every five seconds. But if it is hitting the fan and you need to know why, *"get -g /JVMRuntime/$SERVER/HeapFreeCurrent"* is the command you are looking for. This gives you a graphical representation of your JVM Heap, and updates it in real time.

Now that you have the ability to pop off a graphical window on MBean attributes, why not check on your JDBC Pools, your execute thread pools, your EJB stats, and your cluster stats? WLShell has a decent selection of commands that will allow you to read from shell prompts, throw in a few "if/then" statements, and drop it all into a quick "for" loop to get quick access to a wide array of data from multiple managed servers in a cluster.

This script (http://dev2dev.bea.com/blog/mvaughn25/archive/EXAMPLE.wlsh) is an example of a script I use to pull the data that I consider critical to troubleshooting application issues. If an application is misbehaving, I use this script to get all of the data in front of me. That lets me quickly determine where to focus my troubleshooting effort and lets me monitor the health of the application as we address the issues.

Blog Topic: WebLogic 8.1 SP3 Long Value Bind Problem

By Franz Garsombke
(http://www.jroller.com/page/fgarsombke/20050209)

After wasting about 5 hours on this problem, we have finally figured out the solution. The problem was that we had a database column of VARCHAR2 (4000) size.

Any time we would insert larger than 1333 characters we would get the following error:

```
(Hibernate operation): encountered SQLExcep-
tion [ORA-01461: can bind a LONG value only for
insert into a LONG column
]; nested exception is java.sql.BatchUpdateEx-
ception: ORA-01461: can bind a LONG value only
for insert into a LONG column
```

which, needless to say, is very misleading. The answer is that WL 8.1 SP3 comes with Oracle 10*g* drivers, and we are on an Oracle 9.x database. This link is how to fix this problem:
http://e-docs.bea.com/wls/docs81/jdbc/thirdparty.html#1050527.

> "Developers can be just as productive using Spring and Java as with a 4GL RAD tool"

oops.
N26MA
Forget something?

True application management starts from within.