SYS-CON MEDIA

PERFORMANCE • SCALABILITY • MARSHALLING

BEA WebLogic DEVELOPER'S JOURNAL

DECEMBER 2003 - Volume:2 Issue:12

weblogicdevelopersjournal.com

International Conference & Expo
edge 2004 EAST
FEBRUARY 24-26, 2004
HYNES CONVENTION CENTER, BOSTON, MA
see page 33 for details

BPM Offline Viewer

Michael Havey 6

FROM THE EDITOR
Can you Log In Now?
by Joe Mitchko
page 4

TRANSACTION MANAGEMENT
Four Legs Good, Two Legs Bad?
by Peter Holditch
page 38

PERFORMANCE
Performance Improvement in a J2EE Application
by GVB Subrahmanyam & Shankar Itchapurapu
page 40

FROM THE OFFICE OF THE CTO
Data Center Automation
by Benjamin Renaud
page 44

NEWS & DEVELOPMENTS
page 50

PERFORMANCE: **Smoothing the Hand-Off to Production** Strategies for bridging the expertise gap — 12 Hugh Docherty

SCALABILITY: **The Cost of Marshalling** Not as expensive as you might think — 16 Peter Zadrozny

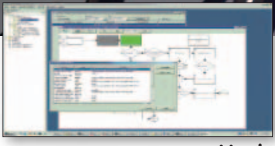
INTEGRATION: **Introduction to ebXML** Making collaboration easier — 18 Komal Mangtani

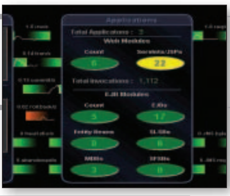
MANAGEMENT: **Application Management with WebLogic Server for Developers** PART 3 JMX fundamentals and tool support — 26 Robert Patrick, Vadim Rosenberg

SERVER: **conf2admin** Automate script creation for your WebLogic Server domain configuration — 32 Mike Jasnowski

PRODUCT REVIEW: **Application Management with NetIQ AppManager Suite** Show how your investment dollars are earning their keep — 36 Laurence Moroney

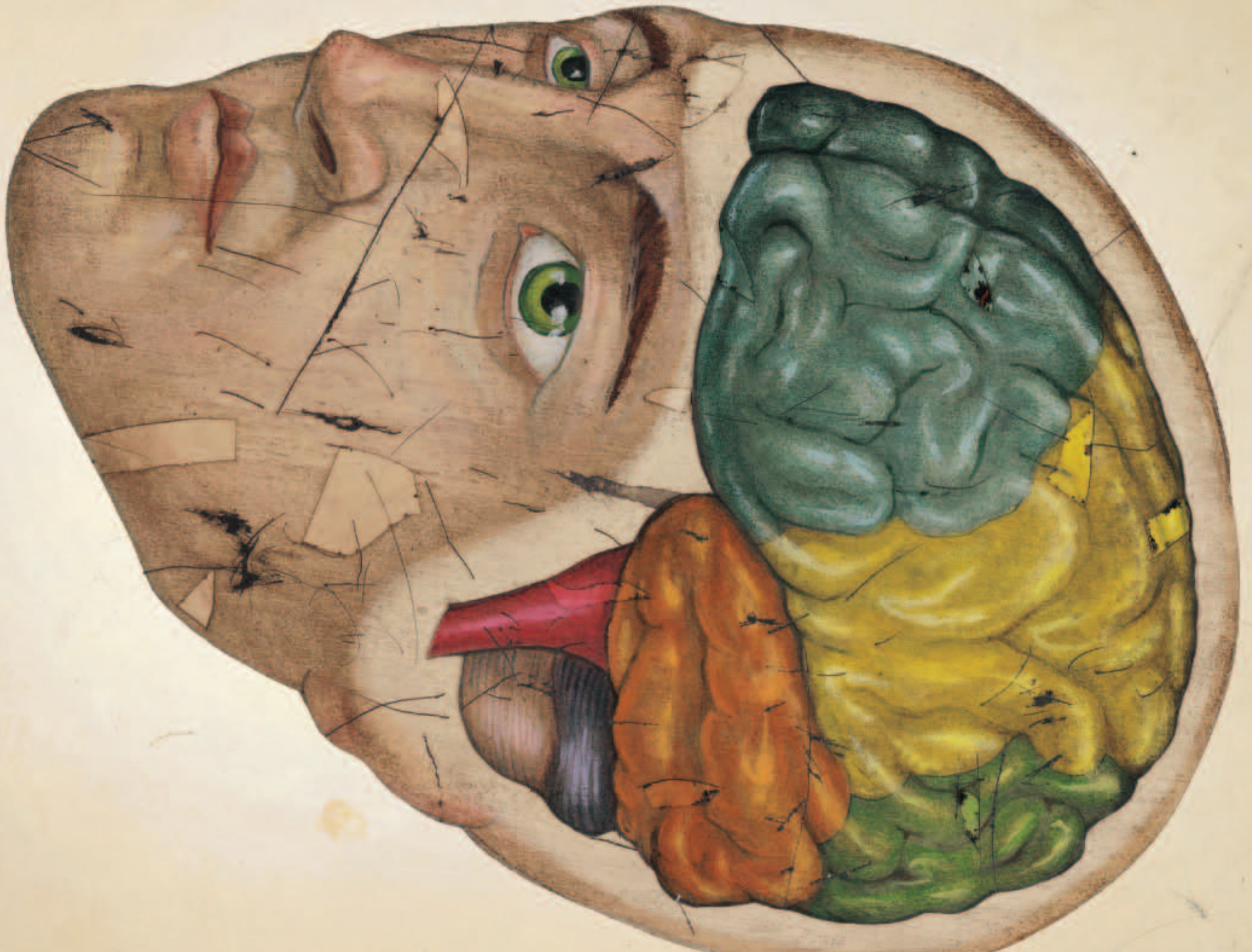
WORKSHOP: **WebLogic Workshop 8.1 IDE Fundamentals** Speed up your development and testing — 46 Anbarasu Krishnaswamy





RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2004
$8.99US $9.99CAN
12>
0 09281 03424 7

SYS-CON MEDIA



DO YOU HAVE A BRAIN?

CEREBRUM: CONTROLS THOUGHT.

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

MEDULLA OBLONGATA: CONTROLS INVOLUNTARY FUNCTIONS.

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

BROCA'S AREA: CONTROLS LANGUAGE.

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

PITUITARY GLAND: CONTROLS HORMONES.

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

YOU HAVE A BRAIN.

So why spend so much time on mindless coding?

Download Weblogic Workshop Now.

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

bea®

Can You Log In Now? Good!

BY JOE MITCHKO

A colleague of mine, who is a senior architectural specialist, recently finished a short, three-week consulting engagement with several other performance-test engineers to determine why it was taking several minutes on average for users to log in to a financial funds management system. After poking and prodding through the system, and sifting through architectural and design documents, the team determined that there wasn't any one particular performance bottleneck that was the root cause of the problem. The short of it was that the system was just doing too much – from retrieving hoards of account information from the legacy back end to setting up user session objects. No smoking gun here, the login process was running about as fast as it could. Ah, there is nothing that I can think of in this business that is so often neglected during the development process than the subject of performance. We are so often transfixed on making the darn thing work, leaving the performance issues for postmortem work – that is, something for the performance testers to figure out. Often we can get away with this thinking, and try to tune a system up after we get it functionally certified. In the case mentioned above, the problem goes a lot deeper and is not easily solved without redesign.

Designing a system so that it operates within the specified service-level agreements requires, first of all, good service-level agreements (SLAs). The SLA must spell out exactly what is expected regarding user response times and maximum concurrent users, to name a few. A good SLA also requires that someone familiar with the technology establish realistic values in this regard, so that expectations are within technical reason.

The next step requires the coordinated effort of all involved on the project, from system and data architects to each and every developer on the team – all must be conscious of performance, and the consequences of what they are doing (or not doing) in that regard.

As an example, a few years ago I worked on a Java development team with several ex-Netscape engineers who had a lot of experience in the area of performance and were highly sensitized to getting it right early in the game. I remember their initial hesitancy to develop the system using Java, primarily because of its interpretive nature and high overhead versus going with native C++. In the end, Java was chosen primarily so that it would be compatible with the remaining WebLogic J2EE architecture – but every line was scrutinized. They realized the implications of ignoring this stuff early on.

In this issue of ***WLDJ***, we focus on performance-related issues as well as a few scalability options for the WebLogic platform. The two kind of go hand-in-hand; you can always solve performance issues by scaling up – the old throwing-hardware-at-the-problem solution. But this leaves little room for when you need to scale up for more legitimate reasons, such as increased transaction activity.

I hope that what you read in this issue will be beneficial when it comes to solving some of the real performance issues you may encounter in the field.

On another note, what comes in a handsome black metal case and provides you with everything you need to stay up-to-date as a WebLogic developer? Why, it's the BEA dev2dev subscription of course. On a quarterly basis, you will receive software updates (including a non-expiring development license), technical support, developer resources, product betas, and partner content. And the part I like best: you get a complimentary subscription to *WLDJ*! Basically, everything you need to stay up-to-date. For further information, go to http://dev2dev.bea.com/subscriptions.

AUTHOR BIO...

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

CONTACT: joe@sys-con.com

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

WORLDWIDE NEWSSTAND DISTRIBUTION
Curtis Circulation Company. New Milford, NJ

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

SYS-CON MEDIA

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

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

Not any more.

Get Wily.™

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

wily
technology

BPM Offline Viewer

A BEA WEBLOGIC INTEGRATION 7 TOOL

BY **MICHAEL HAVEY**

AUTHOR BIO

Michael Havey is a BEA consultant with eight years of industry experience, mostly with application integration. He is also a father, poet, philosopher, and sports fan.

CONTACT...

mhavey@bea.com

Developers of workflow-based applications with the Business Process Modeler (BPM) component of BEA WebLogic Integration Version 7 use a powerful, feature-rich, graphical editor, called Studio, to design workflow templates and to monitor the progress and state of runtime instances of templates.

As Figure 1 shows, Studio is an online tool: it calls the WebLogic Integration server application to retrieve information about workflows or to commit changes. This information is ultimately stored in a database; internally, much of what the WLI application does is converse with the database.

But what if the server goes down? Studio is rendered inoperative. Developers and other users can no longer monitor workflows; no such tools are provided with WebLogic Integration. The obvious solution is to look directly in the database for relevant workflow information. But the Integration database is unintelligible to most users: even if you understand the table structure, you still have to contend with the fact that most of the critical information in the tables is hidden in binary fields, which only WebLogic Integration knows how to interpret. You cannot glean much from exploring the database in your favorite SQL editor.

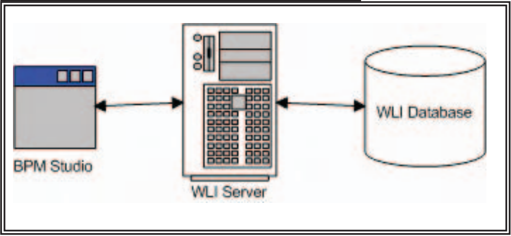
The solution is a new tool, called BPM Offline Viewer, which queries the database directly and unearths the critical information, thanks to its inside knowledge of the layout of WebLogic Integration data (see Figure 2).

Why Is an Offline Viewer Important?

When BPM is used in an enterprise project, it is often one of the most important pieces of the architecture. Workflow, the key entity of BPM, is often referred to as a "choreographer" of human and external system actors, coordinating their activities in a formal business process, and making something coherent from it all. There are plenty of stakeholders who care about, and want to be able to track, workflows in flight. If the server is down, the stakeholders still need to query workflows. The data is important because:

1. The information is valuable in its own right from a business point of view, and should be visible whether or not the server is up.
2. If the server has crashed, the information might provide clues to why, like a "black box"

FIGURE 1



Data flow in BPM: Studio, WebLogic Integration server, database

for a doomed aircraft.

3. If the server is unstable and cannot be restarted, knowing the state of the workflows enables the business to continue the processes by alternate means, such as manually.

About BPM Data

Of the many tables that constitute the BEA WebLogic Integration data model, three hold all of the information pertinent to workflow templates and instances (see Figure 3).

The *design* of a workflow (i.e., the logical sequence of steps developed in BPM Studio) is held in the tables TEMPLATEDEFINITION and TEMPLATE. A template, identified by a unique name and ID (NAME and TEMPLATEID fields, respectively), can have zero or more template definitions. Normally a template has one definition, but having multiple definitions is useful for versioning. For example, a template might have two definitions, only one of which is active. If there are any instances of the inactive definition, they will run to completion under that version, but new instances will run against the active definition. The actual "source code" of the workflow is held in the DATA field of the template definition table, though it is entangled with other control data. The DATA field is a binary type (a LONG RAW type in Oracle) containing serialized Java objects, one of which is an XML document representing the source code.

Runtime instances of a workflow are captured in the INSTANCE table. Most of the important state information (such as the values of workflow variables and the state of workflow tasks) is contained in the DATA field, which again is a binary stream of serialized Java objects.

Knowing how to read and interpret the contents of these binary data fields requires knowing how they were assembled in the first place – in other words knowing how, in the bowels of WebLogic Integration data is persisted to the database. Without divulging any product secrets, Listing 1 shows how, generically, to read one of these fields.

The data, stored in memory as a byte array, is fed into a standard Java ObjectInputStream (Listing 1, line 9). Objects – representing template definition source code, instance variables, runtime task state and other information – are then read from the stream one by one (lines 12–15). Of course, the objects are of no use unless you know what they are and what they are for. You must know:

1. How many objects are in the stream
2. The order of objects in the stream
3. What each object is

Code like this is at the heart of the BPM offline viewer tool.

Tools

The offline viewer consists of three tools:

- ***Template Viewer:*** A command-line tool that retrieves and prints information about a workflow template and each of its definitions. The inputs are database connection parameters (JDBC driver name, URL, user name, and password) and either the template ID or template name. The output is an XML document, which can be fed through the report generator to create an HTML report.
- ***Instance Viewer:*** A command-line tool that retrieves and prints information about an instance of a workflow template. The inputs are database connection parameters and the instance ID. The output, again, is an XML document, which is the basis for a report.
- ***Report Generator:*** A command-line tool that generates an HTML report for one of the viewers above. The inputs are the file names of an XSL and XML document. The output is an HTML document, which can be shown in a browser.

Template Viewer Tool

The template viewer tool prints out information about a workflow template and each of its definitions. In BPM Studio, the equivalent information is presented on a design canvas, as shown in Figure 4.

The XML generated by the tool is lengthy and tedious, but when fed through the report generator it is useful and informative (see Figure 5).

The template viewer tool is called with the following command line:

```
java com.bea.tools.OfflineTemplateDetail $DBDRIV-
ER $DBURL $DBUID $DBPWD -t $TID $FILE
```

The parameters are:

- ***DBDRIVER:*** Class name of JDBC driver used to connect to WebLogic Integration database
- ***DBURL:*** JDBC URL of WebLogic Integration database
- ***DBUID:*** User ID to connect to the database
- ***DBPWD:*** Password to connect to the database
- ***TID:*** Template ID that you are interested in.
- ***FILE:*** Name of output file to write XML document to

For example, Listing 2 will gather information about template 101002 and output the results to 101002.xml.

The XML document output by the tool contains the following information:

- Template ID
- Template Name
- Template Definition ID
- Effective and expiry dates of template definition
- Template Definition XML, which is the source code of the definition in XML form

Instance Viewer Tool

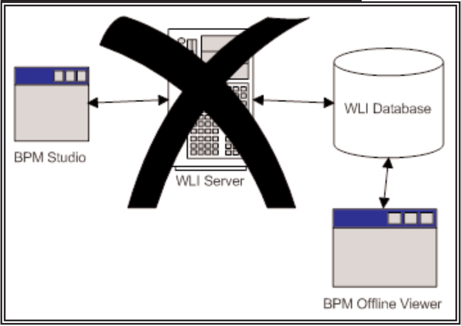
The instance viewer tool prints out information about an instance of a workflow template, notably the values of its variables and the status of its tasks. Figure 6 shows BPM Studio's online view of a workflow instance:

The tool generates XML that is, as in the case of the template viewer, overwhelmingly detailed. But the report generator saves the day with the HTML view shown in Figure 7.

The instance viewer tool is called with the following command line:

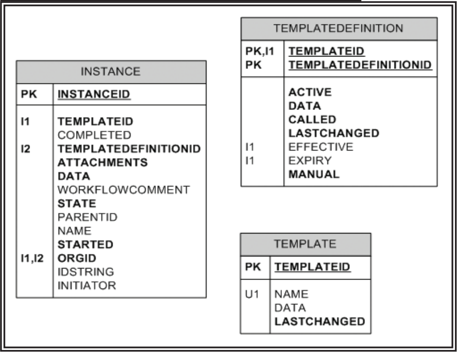
```
java com.bea.tools.OfflineWorkflowDetail $DBDRIV-
ER $DBURL $DBUID $DBPWD $WFID $FILE
```

FIGURE 2



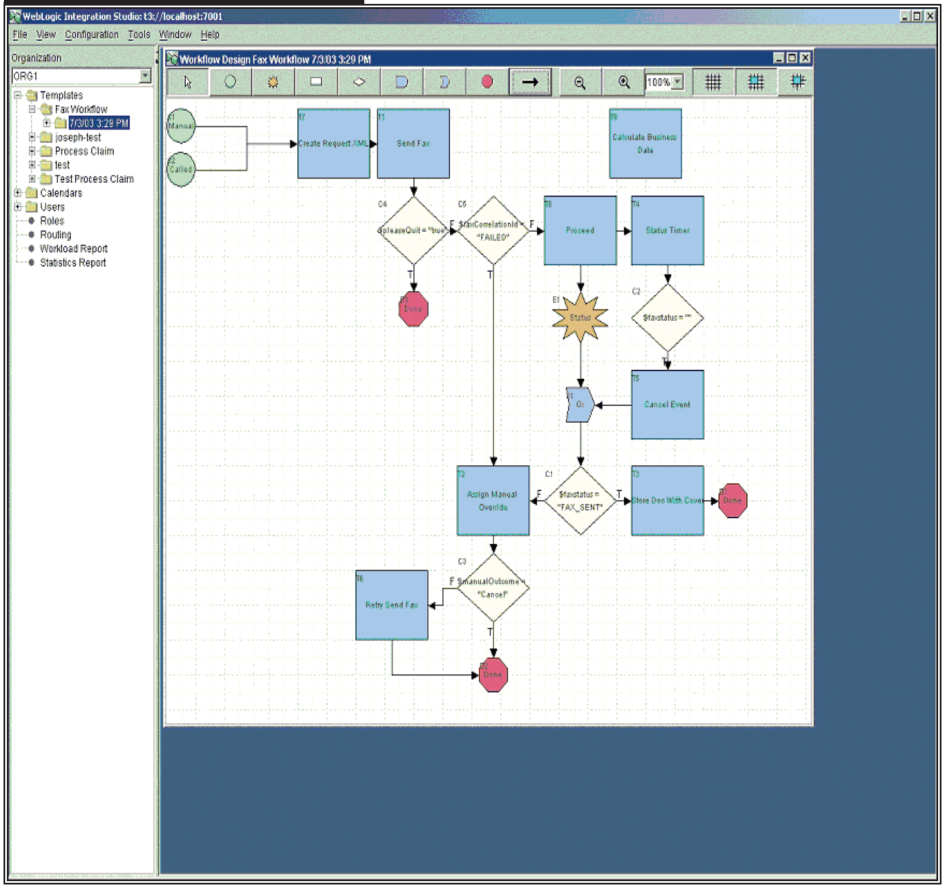
BPM Offline Viewer hits database directly

FIGURE 3



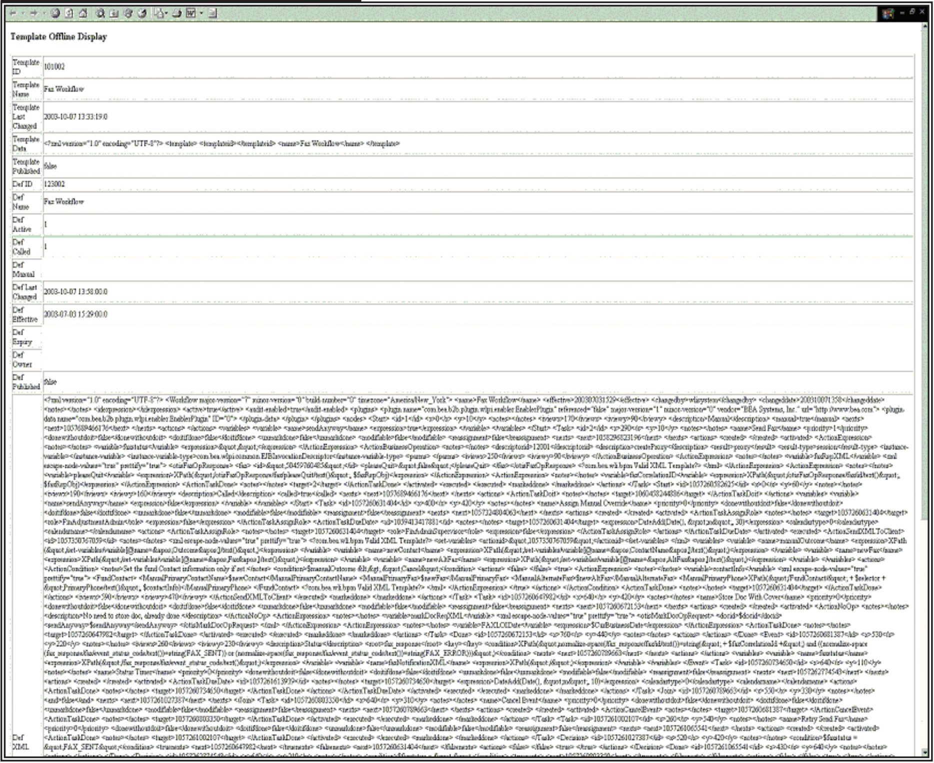
BPM tables

FIGURE 4



Workflow template in BPM Studio

FIGURE 5



Template Offline Display

Template ID	101002
Template Name	Fax Workflow
Template Last Changed	2003-10-07 13:33:19.0
Template Data	<?xml version="1.0" encoding="UTF-8"?> <template> <templateid></templateid> <name>Fax Workflow</name> </template>
Template Published	false
Def ID	123002
Def Name	Fax Workflow
Def Active	1
Def Called	1
Def Manual	
Def Last Changed	2003-10-07 13:58:00.0
Def Effective	2003-07-03 15:29:00.0
Def Expiry	
Def Owner	
Def Published	false
Def XML	<?xml version="1.0" encoding="UTF-8"?> <Workflow major-version="7" minor-version="0" build-number="0" timezone="America/New_York"> <name>Fax Workflow</name> <effective>2003070 [illegible]

Template Report

The parameters are the following:

- ***DBDRIVER:*** Class name of JDBC driver used to connect to the BEA WebLogic Integration database
- ***DBURL:*** JDBC URL of the WebLogic Integration database
- ***DBUID:*** User ID to connect to the database
- ***DBPWD:*** Password to connect to the database
- ***WFID:*** Workflow instance ID that you are interested in
- ***FILE:*** Name of output file to write the XML document to

For example, Listing 3 will gather information about workflow instance 1155003 and output the results to 1155003.xml.

The XML document output by the tool contains the following information:

- Template ID
- Template Definition ID
- Name of the template
- Start and completion time/date of instance
- Status of each task: Task ID, start and completion date/time, assignee, and other data
- Workflow variable summary: Name, type, value, and other information, for each variable

Report Generator

The report generator takes the XML document created by the template or workflow tool and applies an XSL style sheet to it to produce an HTML document that can be viewed in the browser. Default style sheets are provided for template and workflow report types or users can substitute their own style sheet. The report generator is called as follows:

```
java com.bea.tools.OfflineReportGenerator $XML
$XSL $HTML
```

The parameters are:

- Name of XML file produced by the template or instance viewer tool
- Name of the XSL file to generate the report. Use "WorkflowOffline.xsl" for instances, and "TemplateOffline.xsl" for templates
- Name of the HTML file to be generated by applying the XSL to the XML

For example, the following code snippet generates the HTML report "101002.html" for template 101002:

```
java com.bea.tools.OfflineReportGenerator
101002.xml TemplateOffline.xsl 101002.html
```

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

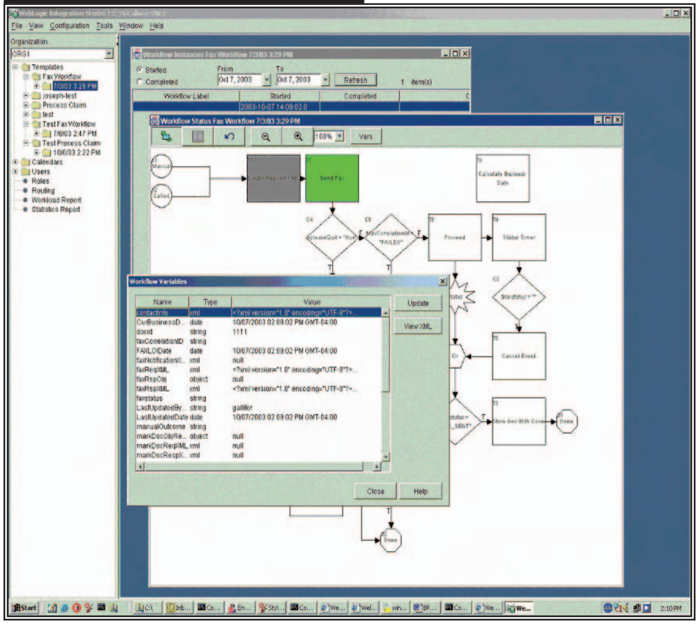
The wild blue yonder of operational monitoring and management is extremely unforgiving. Which means that going live with the monitoring software you used in development is a great way to go dead—quickly! You simply can't support operations if your staff is drowning in details provided by development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

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

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

netiQ®
Work Smarter®

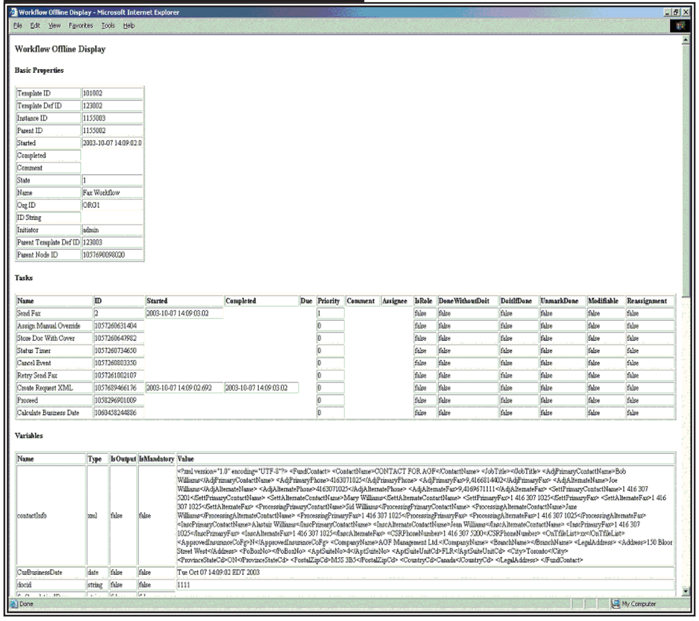
FIGURE 6



Name	Type	Value
contactInfo	xml	<?xml version="1.0" encoding="UTF-8"?>...
CurBusinessD...	date	10/07/2003 02:09:02 PM GMT-04:00
docid	string	1111
faxCorrelationID	string	
FAXLOIDate	date	10/07/2003 02:09:02 PM GMT-04:00
faxNotificationX...	xml	null
faxReqXML	xml	<?xml version="1.0" encoding="UTF-8"?>...
faxRspObj	object	null
faxRspXML	xml	<?xml version="1.0" encoding="UTF-8"?>...
faxstatus	string	
LastUpdatedBy...	string	gallifer
LastUpdatedDate	date	10/07/2003 02:09:02 PM GMT-04:00
manualOutcome	string	
markDocObjRe...	object	null
markDocReqXML	xml	null
markDocRespX...	xml	null

BPM Studio Instance Monitoring

FIGURE 7


Workflow Offline Display - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Done
My Computer

Workflow Offline Display

Basic Properties

Template ID	101002
Template Def ID	123002
Instance ID	1155003
Parent ID	1155002
Started	2003-10-07 14:09:02.0
Completed	
Comment	
State	1
Name	Fax Workflow
Org ID	ORG1
ID String	
Initiator	admin
Parent Template Def ID	123003
Parent Node ID	1057690098020

Tasks

Name	ID	Started	Completed	Due	Priority	Comment	Assignee	IsRole	DoneWithoutDoit	DoitIfDone	UnmarkDone	Modifiable	Reassignment
Send Fax	2	2003-10-07 14:09:03.02			1			false	false	false	false	false	false
Assign Manual Override	1057260631404				0			false	false	false	false	false	false
Store Doc With Cover	1057260647982				0			false	false	false	false	false	false
Status Timer	1057260734650				0			false	false	false	false	false	false
Cancel Event	1057260803350				0			false	false	false	false	false	false
Retry Send Fax	1057261002107				0			false	false	false	false	false	false
Create Request XML	1057689466176	2003-10-07 14:09:02.692	2003-10-07 14:09:03.02		0			false	false	false	false	false	false
Proceed	1058296901009				0			false	false	false	false	false	false
Calculate Business Date	1060458244886				0			false	false	false	false	false	false

Variables

Name	Type	IsOutput	IsMandatory	Value
contactInfo	xml	false	false	<?xml version="1.0" encoding="UTF-8"?> <FundContact> <ContactName>CONTACT FOR AGF</ContactName> <JobTitle></JobTitle> <AdjPrimaryContactName>Bob Williams</AdjPrimaryContactName> <AdjPrimaryPhone>4163071025</AdjPrimaryPhone> <AdjPrimaryFax>9,4166814402</AdjPrimaryFax> <AdjAlternateName>Joe Williams</AdjAlternateName> <AdjAlternatePhone>4163071025</AdjAlternatePhone> <AdjAlternateFax>9,4169671111</AdjAlternateFax> <SettPrimaryContactName>1 416 307 5201</SettPrimaryContactName> <SettAlternateContactName>Mary Williams</SettAlternateContactName> <SettPrimaryFax>1 416 307 1025</SettPrimaryFax> <SettAlternateFax>1 416 307 1025</SettAlternateFax> <ProcessingPrimaryContactName>Sid Williams</ProcessingPrimaryContactName> <ProcessingAlternateContactName>Jane Williams</ProcessingAlternateContactName> <ProcessingPrimaryFax>1 416 307 1025</ProcessingPrimaryFax> <ProcessingAlternateFax>1 416 307 1025</ProcessingAlternateFax> <InscPrimaryContactName>Alastair Williams</InscPrimaryContactName> <InscAlternateContactName>Jean Williams</InscAlternateContactName> <InscPrimaryFax>1 416 307 1025</InscPrimaryFax> <InscAlternateFax>1 416 307 1025</InscAlternateFax> <CSRPhoneNumber>1 416 307 5200</CSRPhoneNumber> <OnTfileList>xx</OnTfileList> <ApprovedInsuranceCoFg>N</ApprovedInsuranceCoFg> <CompanyName>AGF Management Ltd.</CompanyName> <BranchName></BranchName> <LegalAddress> <Address>150 Bloor Street West</Address> <PoBoxNo></PoBoxNo> <AptSuiteNo>4</AptSuiteNo> <AptSuiteUnitCd>FLR</AptSuiteUnitCd> <City>Toronto</City> <ProvinceStateCd>ON</ProvinceStateCd> <PostalZipCd>M5S 3B5</PostalZipCd> <CountryCd>Canada</CountryCd> </LegalAddress> </FundContact>
CurBusinessDate	date	false	false	Tue Oct 07 14:09:02 EDT 2003
docid	string	false	false	1111

Workflow Report

The snippet generates the HTML report “1155003.html” for instance 1155003:

```
java com.bea.tools.OfflineReportGenerator 1155003.xml WorkflowOffline.xsl
1155003.html
```

Limitations

This offline tool has only been tested on BEA WebLogic Integration 7. Its magic is in its ability to read hidden binary data in the WebLogic Integration database, but the structure of this data changes from release to release. Minor modifications to the tool would allow it to run properly against earlier versions of WLI (1.x, 2.x), which are similar in implementation to release 7.

WebLogic Integration 8.1

As for the latest version of BEA WebLogic Integration (release 8.1), it is so dissimilar to release 7.0 that it is almost a different product. In WebLogic Integration 8.1, template definitions are designed in the offline BEA Weblogic Workshop design tool rather than the online BPM Studio. WebLogic Workshop eliminates the need for the offline template viewer tool. As for instance data, WebLogic Integration 8.1 introduces a new data model and the concept of runtime versus archived data, neither of which the current implementation of the instance viewer is designed to handle.

Summary

The BPM Offline Viewer is valuable to users of the BPM subsystem of BEA WebLogic Integration 7.0. It extends the capabilities of this popular product by providing offline viewing and monitoring. Considering the importance and centrality of BPM on most projects in which it is used, offline capabilities are critical.

Listing 1: Reading a hidden binary data field

```
void readHiddenField(byte data[])
{
   ByteArrayInputStream bis = null;
   ObjectInputStream ois = null;
   try
   {
      // open Java object stream
      bis = new ByteArrayInputStream(data);
      ois = new ObjectInputStream(bis);

      // read each of the objects
      Object o1 = ois.readObject();
      Object o2 = ois.readObject();
      …
      Object oN = ois.readObject();
   }
   finally
   {
      // close the stream
      if (ois != null) try { ois.close(); } catch(Exception x) {}
      if (bis != null) try { bis.close(); } catch(Exception x) {}
   }
}
```

Listing 2: Script to generate report

```
DBDRIVER=oracle.jdbc.driver.OracleDriver
DBURL=jdbc:oracle:thin:@(description=(address=(host=dbhost.bea.com
(protocol=tcp)(port=1500))(connect_data=(sid=wlidb)))
DBUID=wliuser
DBPWD=wliuser
TID=101002
FILE=101002.xml
java com.bea.tools.OfflineTemplateDetail $DBDRIVER $DBURL $DBUID $DBPWD
-t $TID $FILE
```

Listing 3 Script to generate workflow instance report

```
DBDRIVER=oracle.jdbc.driver.OracleDriver
DBURL=jdbc:oracle:thin:@(description=(address=(host=dbhost.bea.com
(protocol=tcp)(port=1500))(connect_data=(sid=wlidb)))
DBUID=wliuser
DBPWD=wliuser
WFID=1155003
FILE=1155003.xml
java com.bea.tools.OfflineWorkflowDetail $DBDRIVER $DBURL $DBUID $DBPWD
$WFID $FILE
```

Lowering the cost and complexity of developing secure, manageable enterprise-class Web services with BEA WebLogic Workshop™ and Confluent for BEA

With Confluent for BEA, Web services developed in BEA WebLogic Workshop™ 8.1 are automatically monitored, managed and secured at development time. Confluent's Control is bundled with WebLogic Workshop 8.1.

Real-time monitoring
No additional coding required, turn on instrumentation for a Web service, view all conversations as they execute, and drill down to views of individual operations and controls

Policy specification and enforcement
Define operational policies including security, logging and monitoring for Web services within BEA WebLogic Workshop 8.1

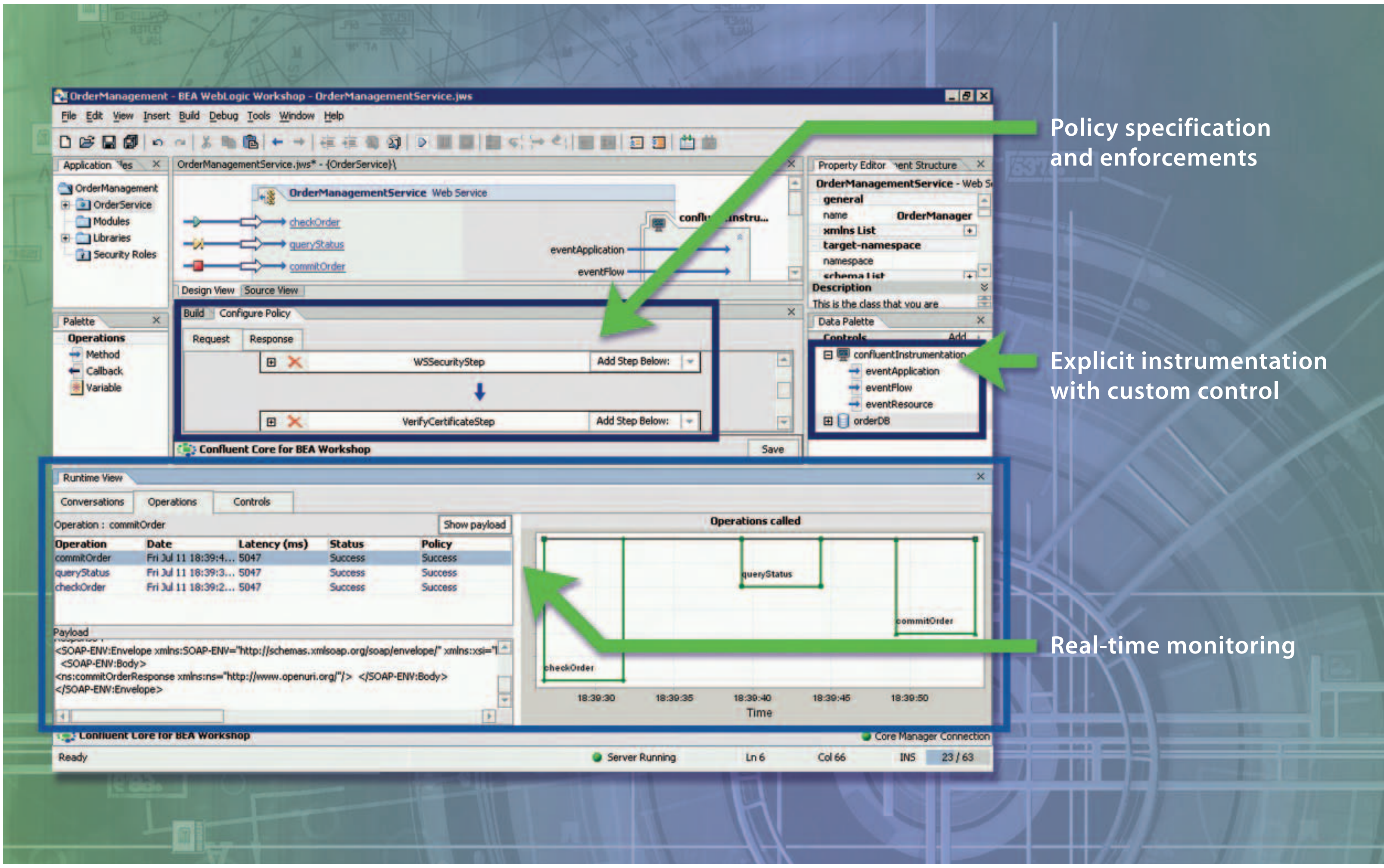
Explicit instrumentation with custom controls
Use Confluent Control to explicitly call operations such as monitoring and security from Web services

BENEFITS:

If you're a Developer—focus on application logic not infrastructure services

If you're an Architect—consistently implement security, change and other operational policies

If you're an Operations Manager—efficiently monitor, manage and evolve increasingly distributed applications



Operation	Date	Latency (ms)	Status	Policy
commitOrder	Fri Jul 11 18:39:4...	5047	Success	Success
queryStatus	Fri Jul 11 18:39:3...	5047	Success	Success
checkOrder	Fri Jul 11 18:39:2...	5047	Success	Success

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="
 <SOAP-ENV:Body>
<ns:commitOrderResponse xmlns:ns="http://www.openuri.org/"/>  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Confluent Software

Download Confluent for BEA at:
http://www.confluentsoftware.com/solutions/download/beacib.html

For more information on additional Confluent Web services management solutions, visit: **http://www.confluentsoftware.com**

BY **HUGH DOCHERTY**

AUTHOR BIO...

Hugh Docherty is a product manager for Quest Software's J2EE performance management solutions. Hugh has extensive experience managing development of Web-based financial applications, on top of 10 years of experience building and supporting Web applications.

CONTACT...

hugh.docherty@quest.com

It is late Monday afternoon and your application is finally going into production. After a year of development and months of QA, it will be live first thing Tuesday morning. The next several days will be critical as customers interact with the new system for the first time.

It would be nice to think developers can kick back, or even get started on new development, but that's not going to happen. Unexpected performance degradation and errors will likely crop up, and application support staff won't be able to address them alone.

Right now, J2EE applications management in production is a mixed bag. Some companies have an experienced WebLogic administrator or specialized application support staff. But many companies don't, with production management falling to some combination of general IT application support staff and developers (with developers shouldering more of the burden because they generally have more J2EE experience). It's not an effective or efficient production management strategy. Companies need to grow the expertise of application support staff so they can handle most simple production issues without requiring assistance from developers.

Performance Assurance Before Production

As a developer, you do all you can to ensure that your application will run well in production. During development, as you design and implement the application's functionality, you also consider performance goals. Time and resources are a challenge in every project. And while functional testing usually takes precedence over performance testing in development and QA, the benefits of discovering performance issues at this stage should not be overlooked. Allocating time and resources to test performance pays for itself as you avoid downtime and maintain customer loyalty. Every development and test plan should identify performance criteria that need to be met prior to production.

But profiling your application's performance in the lab is not the same as subjecting it to real user load. In the lab, you generally don't have the resources for a truly comprehensive stress test, so you test the application on a small cluster of machines handling a simulated user load. This is helpful to identify transactional and system-wide bottlenecks that are in need of attention, and to design how the application will behave when it is overloaded or unable to access some part of the system. For example, if a data source is not available (resulting in some kind of "Error 500") can you inform the user that a piece of data is not available and allow them to proceed in another direction instead of simply halting abruptly?

However, performance testing during development and QA, while important, will not address every possible performance problem. In production, your application will be subjected to loads and usage patterns that you could not anticipate. Your application support team needs to be prepared for the unexpected.

The Expertise Gap

Once the application is ready for production, responsibility for the application should ideally be handed over to a WebLogic administrator or the IT application support staff. Unfortunately, many companies don't have experienced BEA WebLogic Server administrators, so developers are pressed into service to fill the gap.

The reasons for this expertise gap are logical enough when you think about it. In the early days of a new business application platform, it's the developers who have had the most experience working with it. Only when many applications are in production does the application support personnel have enough experience to build their own expertise.

The application support team has two goals: to ensure that application availability is maintained based on established service levels, and to plan for the application's continued availability in the future as required by the business. We'll focus on the first goal here, but be aware that capacity planning is a future activity that requires the expertise application support teams need to develop.

Support teams need to ramp up quickly and have a process in place to deal with the crisis of the day. The expertise they need to develop is quite different from that of application developers – it's more about tuning the application server environment, understanding the resource dependencies of J2EE, and configuring the application servers.

Strategies for Handing Off to Production

Developers can either passively wait to be drawn into firefighting minor performance issues, or play a proactive role in helping the application support team learn to maintain the availability of the application. The amount of support required varies with the size and structure of the team, and the depth to which they are expected to diagnose problems. As WebLogic administration staff gain experience and expertise, they will in time become the experts in running your application as optimally as possible in production.

As part of the transition into production, the development team should produce a support document describing the high-level functionality and dependencies of the application – an overview of the application from a systems administration perspective, if you will. This starting point should give even inexperienced support staff a place to start learning about the application. It should document each tier of the application and external dependencies, and include a list of key error log messages.

While your code resides solely on the BEA WebLogic Server, many other servers provide data and functionality to your application. The support document should outline all Web servers, components developed outside your team, and all back-end databases used by the application for data and customer authentication. A diagram depicting the source of the data and the type of communication used between your system and the external data source would be useful. A good description of all the moving parts in your J2EE environment will help to get them up to speed quickly.

To ensure a stable environment, the WebLogic administrator should become the gatekeeper for all application and configuration changes made to the WebLogic Servers in production. To prepare the WebLogic administrator to manage the server, developers should update the support document with a list of applications that need to be deployed to each server, expected response times, and a list of hotspots and potential bottlenecks for the server and applications.

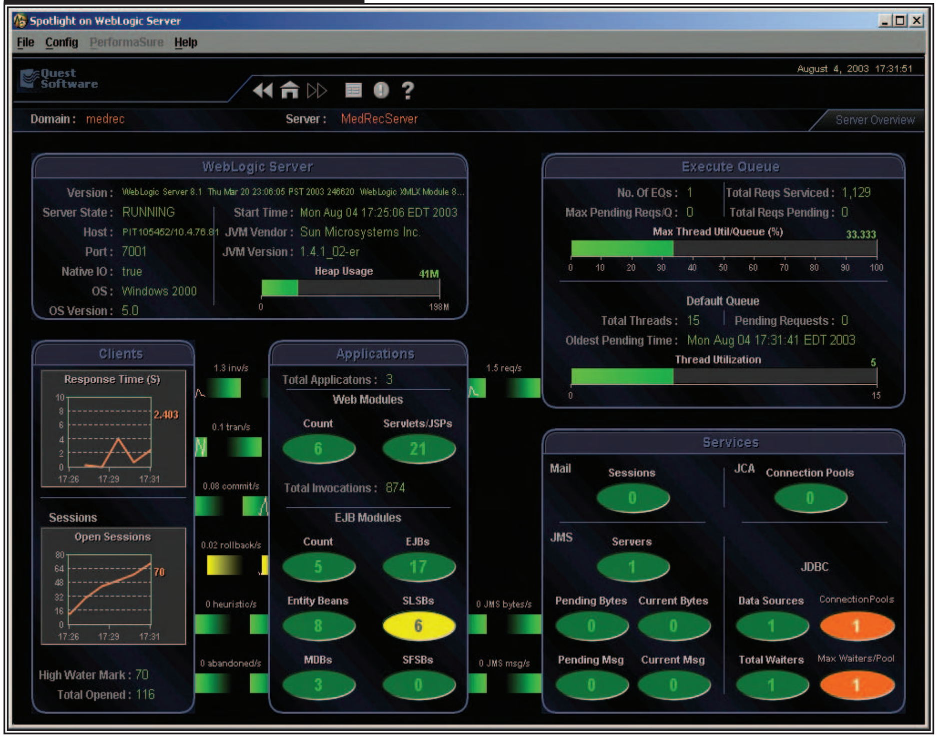
Once provided with this roadmap, the application support team needs to be able to see how it all works in practice. This is one area in which tools can be beneficial.

Tools to Inform and Educate

As a developer, you have many tools available to help you create and test a large application – basic tools like editors, debuggers, and source control tools; and tools that enhance productivity and enable you to do your job better, such as profilers, memory debuggers, and automated testing tools. Application support teams need their own specialized tools to configure, monitor, and diagnose problems with the application in production. Ideally, such tools not only make for more efficient management, but also educate and help develop expertise.

Tools exist for anything from 24x7 application monitoring to deep problem diagnostics tools for code and database issues. But in terms of helping application support teams develop their J2EE expertise, a good

FIGURE 1



Spotlight on WebLogic Server

File Config PerformaSure Help

August 4, 2003 17:31:51

Domain : medrec Server : MedRecServer

WebLogic Server

Version : WebLogic Server 8.1 Thu Mar 20 23:06:05 PST 2003 246620 WebLogic XMLX Module 8...
Server State : RUNNING
Host : PIT105452/10.4.76.81
Port : 7001
Native IO : true
OS : Windows 2000
OS Version : 5.0
Start Time : Mon Aug 04 17:25:06 EDT 2003
JVM Vendor : Sun Microsystems Inc.
JVM Version : 1.4.1_02-er
Heap Usage 41M (0–198M)

Execute Queue

No. Of EQs : 1
Max Pending Reqs/Q : 0
Total Reqs Serviced : 1,129
Total Reqs Pending : 0
Max Thread Util/Queue (%) 33.333
Default Queue
Total Threads : 15
Pending Requests : 0
Oldest Pending Time : Mon Aug 04 17:31:41 EDT 2003
Thread Utilization 5 (0–15)

Clients

Response Time (S) 2.403
Sessions
Open Sessions 70
High Water Mark : 70
Total Opened : 116

Applications

Total Applicatons : 3
Web Modules — Count 6, Servlets/JSPs 21
Total Invocations : 874
EJB Modules — Count 5, EJBs 17
Entity Beans 8, SLSBs 6
MDBs 3, SFSBs 0

1.3 inv/s; 0.1 tran/s; 0.08 commit/s; 0.02 rollback/s; 0 heuristic/s; 0 abandoned/s; 1.5 req/s; 0 JMS bytes/s; 0 JMS msg/s

Services

Mail — Sessions 0
JCA — Connection Pools 0
JMS — Servers 1
Pending Bytes 0, Current Bytes 0
Pending Msg 0, Current Msg 0
JDBC — Data Sources 1, ConnectionPools 1
Total Waiters 1, Max Waiters/Pool 1

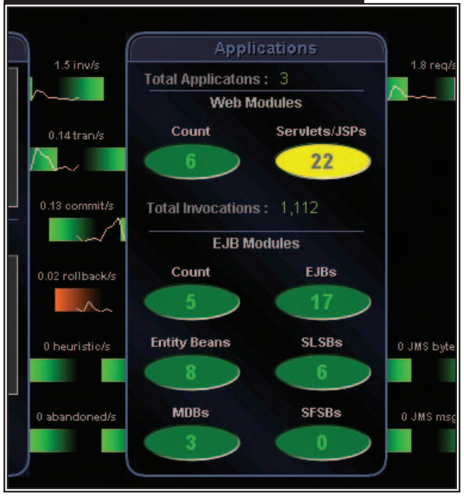
Server Overview

Intuitive real-time management tool for BEA WebLogic Server

place to start is in the area of WebLogic Server management.

The BEA WebLogic Server cluster is really the front door to the application. It runs the code, pools access to the back-end databases, and manages and coordinates use of the shared resources of the system. To help manage the WebLogic Servers, application support should choose a tool that presents information in a well-organized, real-time interface; highlights trouble areas; and provides insight into the cause of problems (see Figure 1).

FIGURE 2



Applications

Total Applicatons : 3
Web Modules — Count 6, Servlets/JSPs 22
Total Invocations : 1,112
EJB Modules — Count 5, EJBs 17
Entity Beans 8, SLSBs 6
MDBs 3, SFSBs 0

1.5 inv/s; 0.14 tran/s; 0.13 commit/s; 0.02 rollback/s; 0 heuristic/s; 0 abandoned/s; 1.8 req/s; 0 JMS byte; 0 JMS msg

At-a-glance view of application performance

Tools that provide a domain overview help isolate whether performance issues are specific to one server, or spread across the entire cluster. With the status of each WebLogic Server, the clusters, and each deployed application, the administrator can quickly determine the extent of the performance outage. This initial triage allows the WebLogic administrator to focus on the problem quickly and provides an initial idea about the impact of the situation on customers.

Presentation of information is a key factor in enabling inexperienced administrators to determine the health of their J2EE systems. There are hundreds of metrics available inside WebLogic Server, each a piece of the performance puzzle. Individually these metrics aren't that useful, but when you are able to visualize the relationships, the performance picture comes into focus, and diagnosing the problem is possible. Tools like the BEA WebLogic Console and command-line interfaces that allow you to browse through the metrics are good if you know what the problem is and may even help WebLogic experts find problems. However, while most WebLogic administrators will rely on the console to make changes to a server configuration, they typically need a more intuitive tool to help find which configuration parameter actually needs to be adjusted.

The best way to bring the WebLogic administrator up to speed on the internals of the WebLogic Application Server is to provide them with a real-time performance view of it. Learning an application and finding performance problems is much easier if you can see how the application and WebLogic Server interact. To see the resources usage fluctuate with the process flow between WebLogic components as customer traffic varies is a powerful educational tool. Once visual alarms are added to the picture, the WebLogic administrator is able to detect and diagnose many problems instead of calling development for assistance. Visual alarms should highlight bottlenecks and indicate where the admin should dive deeper into the application or WebLogic server. A picture is worth a thousand words and a real-time performance picture will save hours of investigation.

Another useful capability to look for in an application server management tool is built-in expert advice. This information enables the WebLogic administrator to make informed decisions about the problem and options for resolution. Development teams may still need to be consulted before the administrator takes corrective actions but the availability of context-sensitive advice will improve the decision-making process within the application support team. The right diagnostic tool is critical whether you are a WebLogic veteran or just starting to manage J2EE applications, and integrated expert advice will improve the confidence of the veterans and allow some tasks to be offloaded to junior staff.

Conclusion

Running J2EE applications in production takes expertise that isn't widely available yet. Developers are the most qualified to diagnose performance and scalability issues at the moment, but instead of fighting fires in production, they should be concentrating on helping application support staff develop the expertise they need to manage their applications.

Simple application documentation is one strategy that will help support teams maintain the availability of the application in production. And to plan future capacity, empirical data collected from the production application is needed. For day-to-day management (and education), a real-time performance viewing tool is ideal, one that can show the live application in operation and help gather data for capacity planning.

FileNet Enterprise Content Management

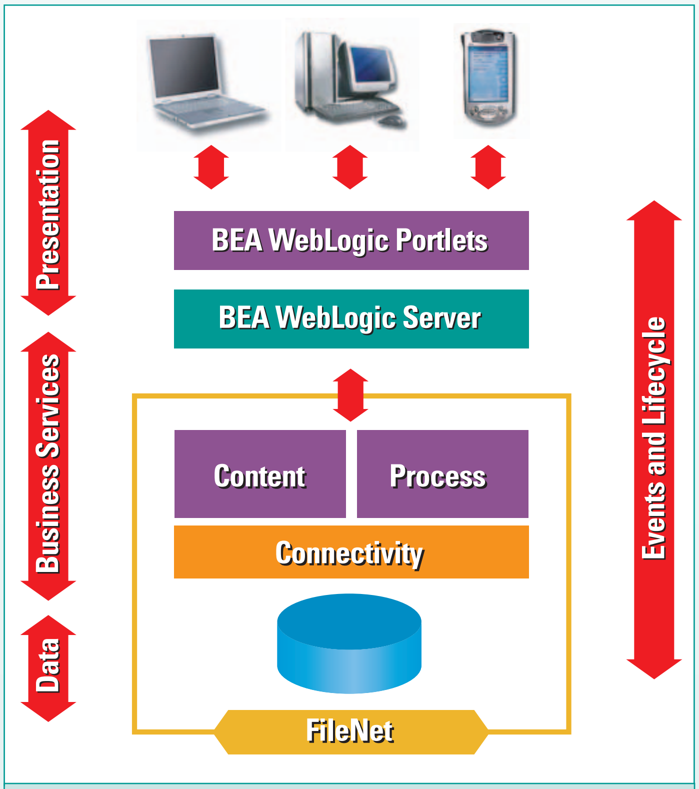
SOLUTIONS LEVERAGE BEA WEBLOGIC ENTERPRISE PLATFORM™

While content and process management is key to improved decision-making, increased access to information, and lower operating costs, developers often struggle with finding scalable content and process management capabilities that are easily integrated into existing infrastructure.

To this end, FileNet Corporation, the leading provider of Enterprise Content Management (ECM) solutions, has teamed with BEA to deliver a powerful and highly integrated solution for easy access to content and business process management functionality through the BEA WebLogic Server™ and WebLogic Portal™ platform.

FileNet's industry-leading ECM technology extends the capabilities of BEA's J2EE development platform to make content easily accessible throughout the enterprise. FileNet P8, FileNet's market-leading ECM architecture, uniquely combines content process and connectivity to help organizations to make more informed decisions in less time. FileNet P8 includes the FileNet Web Content Management Suite, a comprehensive J2EE application for creating and managing websites.

BEA WebLogic Server features a highly scalable architecture optimized for a wide variety of hardware platforms and operating systems and provides a unified and simplified management infrastructure for FileNet applications. Developers can leverage this with FileNet's content and process capabilities as a foundation for portal and Web initiatives.



Leveraging the BEA WebLogic platform, FileNet ECM makes content easily accessible to laptops, desktops, and PDAs. Additionally, through the unification of content, process, and connectivity, FileNet ECM "activates" content to speed critical business decision-making.

Additionally, BEA WebLogic Portal provides users with a personalized, single point of access to FileNet's powerful capabilities to manage and control content. FileNet portlets specifically optimized for BEA WebLogic Portal include:

- User Queue Portlet – Provides the ability to respond to workflow tasks and check the status of workflow processes
- Public Queue Portlet – Offers a single access point where multiple portal users can respond to workflow requests.
- Browse Portlet – Allows portal users access to business content within FileNet repositories.
- Authoring Portlet – Provides users a centralized area to actively manage content within FileNet repositories, allowing users to contribute, check-in and publish content to web applications.
- Quick Search Portlet – Offers single-field search capability combining text and content criteria.

The portlets are built on standard Java Server Page (JSP) technology. This allows developers to use the portlets out of the box, or as a foundation for custom-built portlets.

The combination of the BEA WebLogic Enterprise Platform and FileNet ECM technologies provides key benefits:

- **Streamlined Integration and Management**
  Through standards-based integration, the BEA WebLogic Enterprise Platform offers easy deployment, reduced portal integration costs, and the ability to leverage FileNet's industry-leading FileNet P8 architecture, which offers enterprise-level scalability and flexibility to handle the most demanding content and process challenges.

- **Maximum Customizability and Extensibility**
  FileNet portlets are easily customized to fit specific application requirements. FileNet's P8 architecture provides a framework to manage Web publishing challenges and provide greater process control and consistency across the enterprise.

- **Active Content**
  The ability to access critical content from within the BEA WebLogic Portal activates content and empowers business partners, suppliers and employees to respond immediately to events that set critical processes in motion, and drive them quickly to completion. This enables organizations to make more informed business decisions in less time.

See how you can leverage your BEA WebLogic Enterprise Platform with enterprise content management solutions from FileNet at the FileNet booth at BEA's eWorld Conference, or by calling 1-800-FileNet, or outside the U.S. at 512-434-5935.

FILENET®

The Cost of Marshalling

NOT AS EXPENSIVE AS YOU MIGHT THINK

BY **PETER ZADROZNY**

AUTHOR BIO

Peter Zadrozny is the founding editor of *WebLogic Developer's Journal* as well as the author of the book J2EE Performance Testing.

CONTACT...

z@bea.com

For those of us who are always looking to optimize our code and improve performance by squeezing out a few milliseconds here and there, marshalling is one of those areas that you expect to be so bloated that you would think you could improve performance many times if you could get your hands on it.

This article will explore this area by running a set of experiments to actually understand the cost of marshalling in a typical J2EE setting.

Our experiments are based on a very simple test application. A servlet prepares an object, serializes it, and sends it to an EJB. The EJB deserializes the object, serializes it again, and sends it back to the servlet, which in turn deserializes it. Quite simple as you can see; however, it touches on the basic points of a J2EE application server, the Web container, and the EJB container.

The test application has a handful of parameters. The first one allows us to specify if the object will be passed by reference or by value. The second parameter allows us to specify the type of object we want to use for the tests. We used only an array or a map of strings. The following parameter allows us to specify the size of the object, that is, the number of items in the array or map of strings.

Initial tests on our old Pentium III laptops showed us that the clocks of our machines (and operating systems) could not measure the response time of a round-trip of the application using an array of 1,000 items as it was far less than 1 millisecond. This did not meet our initial expectations that marshalling was expensive. Because of this we added another parameter (*n*) that allows us to define the number of times the servlet calls the EJB. After various tests we found that using *n*=1,000 was the best compromise for all the tests to obtain timing values manageable by the system clocks.

Our tests were conducted on an Intel Shasta computer with four CPUs (Xeon with HT at 2GHz) and 4GB of memory running Red Hat Linux AS 2.1. The application server was BEA WebLogic Server 8.1 running out of the box (all defaults) as an administration server. The only change was to enable HTTP tunnelling so that we could investigate the use of various protocols between server instances. The JVM used is JRockit 8.1 with the following parameters: -Xgc:parallel –Xms:1024m –Xmx:1024m.

The test application was implemented in a servlet so it could be conveniently invoked by altering query string parameters; for example, http://xeon:7001/marshalling/marshalling?n=1000&size=1&callBy=value&type=array.

Every test was run 15 times and the data presented is the average of the last 5 runs. Table 1 presents the results in nanoseconds for an array with sizes going from 1 to 10,000 items.

As expected, passing by reference is very cheap, almost negligible. Passing by value is also quite cheap. Just consider that an array with 10,000 items was passed back and forth between the servlet and the EJB in only 173 nanoseconds per round-trip!

Using a map of strings (see Table 2) becomes more expensive, especially when passing by value, to the point that our laziness prompts us not to run tests using a map of strings with 10,000 characters as it takes too long for our threshold of patience (especially because we run every single test 15 times).

Once again, passing by reference is negligible. However, passing string maps by value is much more expensive than an array, but not as expensive as we would have expected. A map of strings with 1,000 characters will take 7.6 milliseconds to pass by value from the servlet to the EJB and back. That is two serializations and two deserializations.

In running these tests we observed that the maximum CPU usage for the array tests was 2%, while for the string map tests it was 15%. Obviously working with strings maps is more expensive both in response time and CPU usage.

Since not everybody has the good fortune of working with such hardware and OS, we ran the same tests on a Compaq DL380 with two CPUs (Pentium III at 933MHz) and 1GB of memory running Windows 2000 Professional. The pass by reference tests had zero response time. We think this is because of the granularity of the H/W and OS clocks, so we will only show the response times for pass by value (see Table 3).

As you can see, hardware does make a difference. The cost of marshalling the map of strings by value is roughly double that of our previous tests. We also observed that the CPU usage on these tests was about 20% for the array tests and 50% for the string tests.

Obviously you want to use pass by reference when you know that your application is running on the same instance of WebLogic. Good thing WebLogic will automatically convert at runtime all pass by value to pass by reference when in the same .ear file.

With these results in hand, our curiosity moved us to look at how expensive it is to marshal between two computers over a network. We added a new parameter to our test application, which allows us to specify the location of the EJB. If not used, it assumes the same JVM.

Our next set of tests was exactly the same as the previous ones, but the difference was that we had the Web container in the Pentium III–based machine (P3) and the EJB container in the Xeon-based computer (Xeon). The network was isolated and traffic was generated only by the tests (see Table 4).

TABLE 1

ARRAY	BY REFERENCE	BY VALUE
1	3	5
10	4	6
100	4	7
1000	4	23
10000	4	173

Results in nanoseconds

TABLE 2

MAP OF STRINGS	BY REFERENCE	BY VALUE
1	4	75
10	4	139
100	4	788
1000	4	7607

Using a map of strings

TABLE 3

BY VALUE	ARRAY	MAP OF STRINGS
1	0	119
10	0	231
100	31	1441
1000	31	14975
10000	366	N/T

Response for pass by value

TABLE 4

ARRAY	BY REFERENCE	BY VALUE
1	1250	1206
10	1216	1256
100	1300	1269
1000	2422	2494
1000	20607	20784

Passing between two computers

TABLE 5

Map of Strings	By Reference	By Value
1	1450	1397
10	1347	1409
100	3028	3119
1000	21719	21956

Map of strings between two computers

As you can see, the results are substantially more expensive than when running in the same JVM. Also, there is really no difference between passing by reference and by value. In both cases the information has to be passed back and forth between both computers, so it has to pass the values. The results for the map of strings are similar in nature (see Table 5).

However, we can't really say that things are that bad; after all, we are marshalling a map of strings of 1,000 items in about 22 milliseconds back and forth between two computers.

During these tests we observed that the maximum network usage was 40% of the 100 Mbps. The maximum CPU usage for the array tests was 15% on P3 and that of Xeon was 5%. For the string tests it was 25% on P3 and 5% on Xeon.

At this point we realized that we had the choice of transport mechanism. Namely, we could choose T3, the highly optimized RMI wire-protocol of WebLogic, or T3 tunnelled within HTTP.

We ran the same set of tests, but now using T3, and the results showed that for these tests plain T3 was faster than tunnelling it within HTTP. The difference is anywhere between 10% and 50%. In general, we observed that as the number of items in the object increase, the difference decreases. That is, the overhead of HTTP tunnelling is larger for smaller messages.

Another issue that became a concern was that the two machines used in these experiments were not the same, and that the direction of the tests could make a difference. The tests so far had been done by having the Web container in P3 and the EJB container on Xeon. So, would there be a difference in response times if we changed the direction and had the Web container in Xeon and the EJB container in P3?

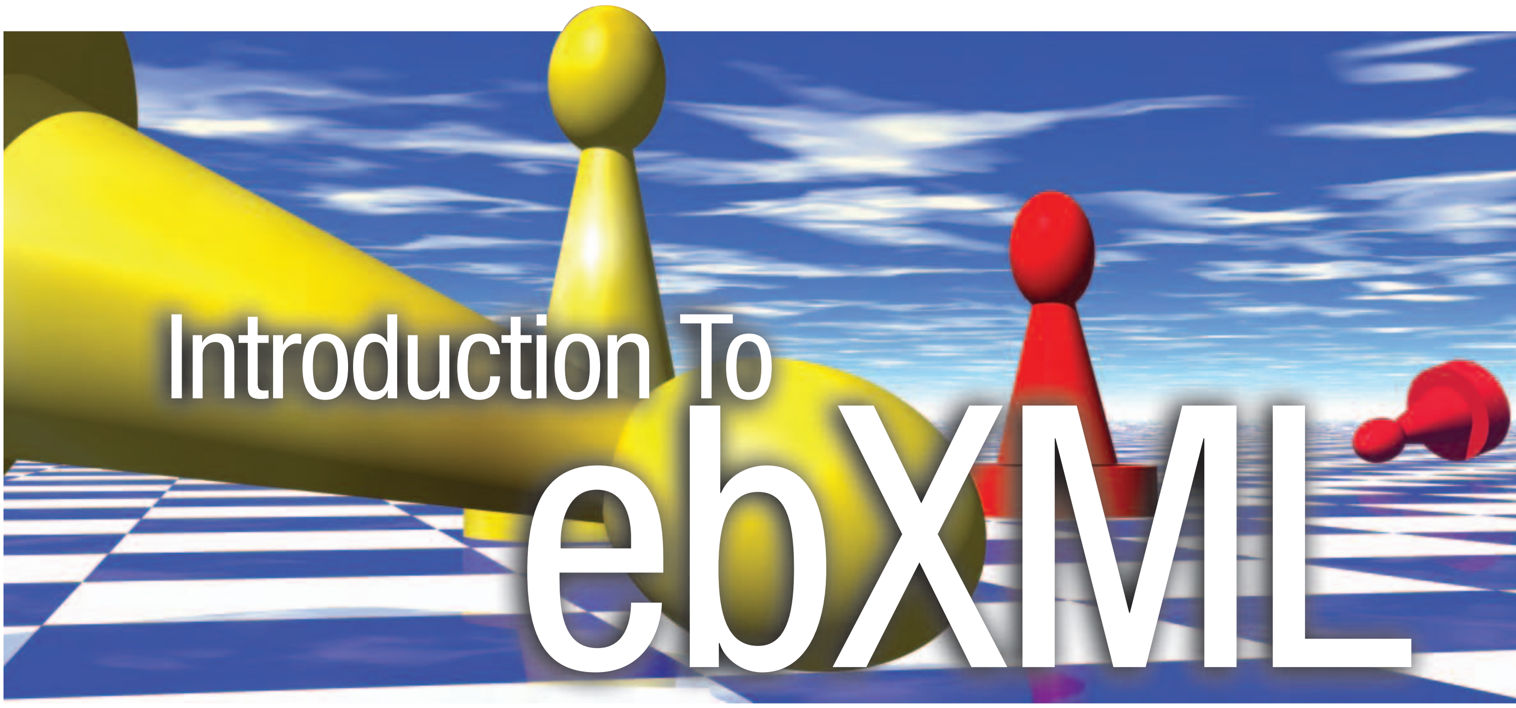
We again ran all the tests changing the direction and noticed that having the Web container in Xeon was slightly faster. The difference was between 1% and 10%, which in general can be considered within the margin of error.

Based on the results of all of these tests, we can conclude that the cost of marshalling is not as expensive as most of us thought it would be. Passing objects by reference when in the same .ear file is the most efficient, and this is how WebLogic handles it internally.

When you have various instances of the BEA WebLogic Server running on different computers, there seems to be no difference between using pass by value and pass by reference. Finally, we observed that plain T3 is more efficient than tunnelling it within HTTP, although the difference tends to decrease as the number of items in the object grows.

Acknowledgments

I want to thank Phil Aston for writing the test application and his wise comments. Special thanks go to Intel for lending us the Shasta computer for running these tests.



Introduction To ebXML

MAKING COLLABORATION EASIER

BY **KOMAL MANGTANI**

AUTHOR BIO

Komal Mangtani is a senior software engineer working on BEA's WebLogic Integration product. She implemented ebXML Business Protocol for BEA WebLogic Integration 8.1 and is a member of the ebXML committee in OASIS. Komal works in the area of high availability for the product.

CONTACT...

mkomal@bea.com

With today's increasing demand for businesses to communicate with each other, business-to-business (B2B) integration holds the key to successful e-commerce collaboration.

The first problem that companies generally face when interacting with each other is how to transfer information that is understood by the organizations involved. Standards, therefore, are an important contributor in helping different partners to interoperate. Many standards have arisen to solve integration problems. While each of them have addressed this issue in different ways, ebXML-Messaging has gained market traction as one of the mature protocols that has resolved reliability and security issues around integration.

ebXML-Messaging is an integration standard from OASIS that provides an XML-based infrastructure to enable consistent, secure, and interoperable message exchange. This specification is part of a series of specifications published by OASIS to define an end-to-end B2B framework. It encapsulates the notion of B2Bi conversations and enforces rules of engagement agreed upon by involved parties. It is transport independent and extends SOAP Messages with Attachments to enable the exchange of business messages containing payloads of any format. Further, it defines transport-level acknowledgements and error messages, eliminating the need to explicitly model them in business processes.

BEA WebLogic Integration 8.1 leverages these features of ebXML to provide a highly productive environment for implementing effective trading partner integration strategies by implementing the ebXML 1.0 and 2.0 Messaging Specification that is integrated with BEA WebLogic Workshop. It simplifies the entire process of designing B2B choreography by eliminating the need to configure the Trading Partner Repository upfront and manages conversations with different trading partners seamlessly. Apart for this, it provides some very useful features for reliable messaging, error handling, and message tracking.

In this article, I look at ebXML Conversations and how they are mapped to an ebXML Control and modelled in business processes. I'll provide some insight into conversation management with WebLogic Integration 8.1 and look at the trading partner configurations required for it. Finally I'll look at the product features mentioned above.

ebXML Conversation

ebXML conversation is at the heart of trading partner integration. It carries the context for messages being exchanged by trading partners. Each ebXML message carries in its envelope an ebXML Service Name and Conversation ID that identifies the conversation it belongs to. The message exchanges represent business activities occurring as part of that conversation and are called ebXML Actions. An ebXML conversation always involves an initiator and a participant. The initiator trading partner may instantiate several conversations with different trading partners as part of a single business process. ebXML control is used to model each of these conversations for the initiator. The participant, on the other hand, participates only in one conversation and therefore the business process itself represents an ebXML conversation on its end.

This concept can be clearly understood using an order-processing scenario. Let's assume an integration scenario between BEA and Avitek, a computer manufacturing company. BEA places purchase orders with Avitek to buy computers. Avitek, on receiving the purchase order from BEA, responds with an invoice. BEA thus becomes the initiator of this conversation while Avitek acts as the participant. BEA and Avitek agree on OrderProcessing as their conversation name, which will have two business activities, processOrder and processInvoice.

ebXML Control

The initiator business process may have multiple ebXML controls and/or ebXML control instances based on the number of conversations it needs to carry. In our case, BEA starts a single conversation with Avitek. Hence we need only one ebXML control and control instance. An ebXML control can be created using the Control Wizard seen in Figure 1. The control name specified here, OrderControl, is used as a TPM Service name as we will see later. An important aspect of the control is the annotations it will carry. An ebXML control has an ebXML service name, From ID, To ID, and ebXML action mode annotations (see Listing 1). The production values for the From and To IDs should be duns' number or actual IDs for BEA and Avitek. It may very well be that these trading partners are not yet configured in the trading partner repository. The workflow designer may need to test the choreography to ensure the correctness of the business process logic without getting into the actual details of configuring the TPM Repository. To avoid the hassle of going back and forth from the WebLogic Workshop IDE to the WebLogic Integration Console, the developer may use the out-of-the-box TPM data available with the product. Another use case can be when the same business process is used as part of various integration scenarios. The processes are made agnostic of trading partners by using the to-selector, from-selector control instance annotations that take in XQuery Selectors as their value.

Once the ebXML Control is created, we modify the method names in the control to reflect the ebXML action name decided by BEA and Avitek. The method to send the order thus becomes processOrder and the method to receive the invoice is named processInvoice (see Listing 1).

A control thus created can be shared across different business processes. The design question that generally arises is, "when can I reuse a previously created ebXML control and when should a new one be created?" The answer is simple. The initiator trading partner can reuse the control if it needs to carry out another conversation with the same participant using the same reliable messaging and security settings. If the reliability and/or security settings required are different while conversing with the same trading partner, a new ebXML control should be created.

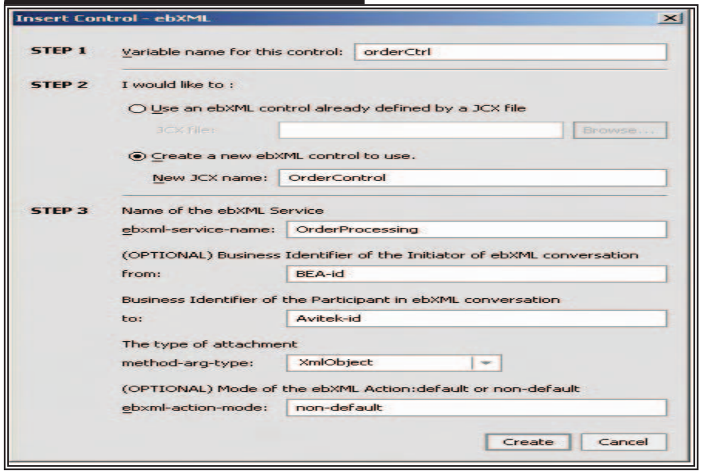
Modelling an ebXML Conversation for the Initiator

The OrderControl created above will now be used to model conversation in BEA's buyer business process. The initiator process can be started via a Message Broker or simple ClientRequest (see Listing 2). Once the start node is modelled, we use OrderControl to model an ebXML conversation. Methods of the control are displayed at the right bottom corner in the design view in the WebLogic Workshop IDE (see Figure 2). As decided by BEA and Avitek beforehand, BEA will send an order to Avitek; hence, we drop the processOrder method in the process. As part of the same conversation, Avitek will respond with the invoice. We drop the processInvoice method in BEA's business process to receive this invoice. Any extra logic for processing received invoices can be implemented in this method. With this step, BEA's business process is complete.

Modelling an ebXML Conversation for the Participant

As I mentioned earlier, the business process for the participant by itself represents an ebXML conversation. Hence we see an ebXML service name as a JPD annotation in Listing 3 that has snippets from Avitek's business process. As decided by two parties, OrderProcessing is specified as an ebXML service name. The participant process is always started by an ebXML message received from the initiator. The Java method name associated with this node represents the action name of the ebXML message to be received and is in our case called processOrder. Now Avitek needs to respond back to BEA with an invoice after fetching it from the file system. This response has to be part of the same ebXML conversation and is modelled using the client callback mechanism. By using a ClientCallback to send a response, the responsibility of ensuring that the message is sent as part of the same conversation is delegated by the workflow designer to the underlying message service handler (MSH). This way of modelling an ebXML participant falls inline with the Web Services Provider model wherein the provider of the service may send an asynchronous response to the consumer of the service using the WebServices Callback

FIGURE 1



Insert Control – ebXML

STEP 1 Variable name for this control: orderCtrl

STEP 2 I would like to :

( ) Use an ebXML control already defined by a JCX file

JCX file: Browse...

(●) Create a new ebXML control to use.

New JCX name: OrderControl

STEP 3 Name of the ebXML Service

ebxml-service-name: OrderProcessing

(OPTIONAL) Business Identifier of the Initiator of ebXML conversation

from: BEA-id

Business Identifier of the Participant in ebXML conversation

to: Avitek-id

The type of attachment

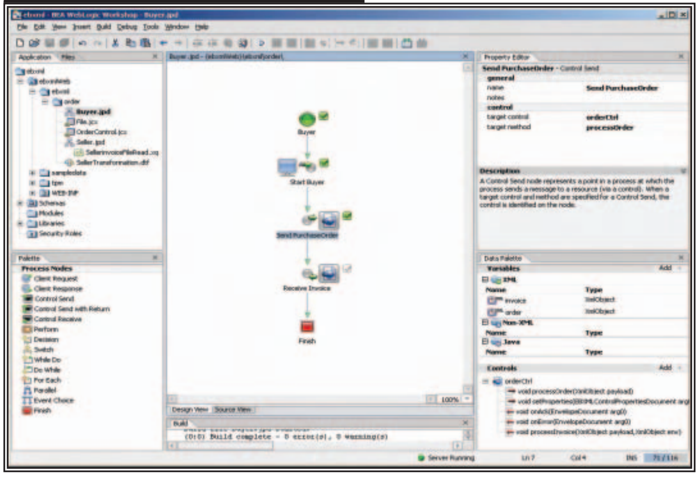
method-arg-type: XmlObject

(OPTIONAL) Mode of the ebXML Action:default or non-default

ebxml-action-mode: non-default

ebXML Control Wizard

FIGURE 2



Property Editor

Send PurchaseOrder - Control Send

general
name: Send PurchaseOrder
notes:
control
target control: orderCtrl
target method: processOrder

Description

A Control Send node represents a point in a process at which the process sends a message to a resource (via a control). When a target control and method are specified for a Control Send, the control is identified on the node.

Data Palette

Variables

XML

Name	Type
invoice	XmlObject
order	XmlObject

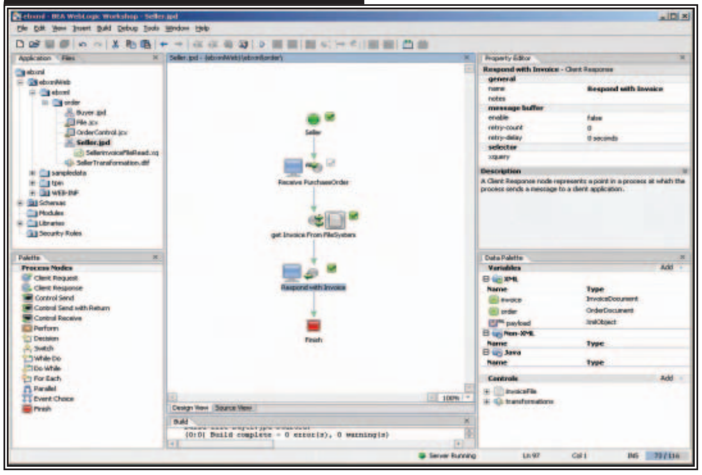
Controls

orderCtrl
void processOrder(XmlObject payload)
void setProperties(EBXMLControlPropertiesDocument arg0)
void onAck(EnvelopeDocument arg0)
void onError(EnvelopeDocument arg0)
void processInvoice(XmlObject payload,XmlObject env)

```
(0:0) Build complete - 0 error(s), 0 warning(s)
```

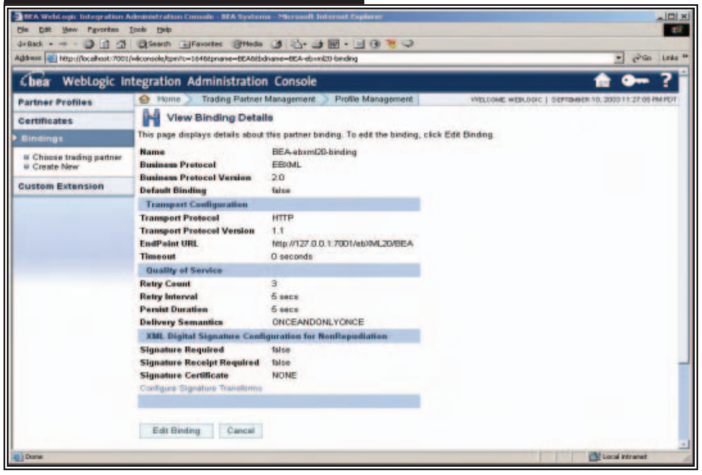
Initiator business process

FIGURE 3



Participant business process

FIGURE 4



WebLogic Integration Administration Console

View Binding Details

This page displays details about this partner binding. To edit the binding, click Edit Binding.

Name	BEA-ebxml20-binding
Business Protocol	EBXML
Business Protocol Version	2.0
Default Binding	false
Transport Configuration	
Transport Protocol	HTTP
Transport Protocol Version	1.1
EndPoint URL	http://127.0.0.1:7001/ebXML20/BEA
Timeout	0 seconds
Quality of Service	
Retry Count	3
Retry Interval	5 secs
Persist Duration	5 secs
Delivery Semantics	ONCEANDONLYONCE
XML Digital Signature Configuration for NonRepudiation	
Signature Required	false
Signature Receipt Required	false
Signature Certificate	NONE

Configure Signature Transforms

Edit Binding Cancel

ebXML Protocol Binding

mechanism. The Java method in the callback interface becomes the ebXML action name and is named processInvoice as decided by BEA and Avitek earlier. Figure 3 shows the design view of this JPD.

Behind the Scenes

Once the business processes for participant and initiator trading partners are designed, orchestration between these processes is carried on by starting the BEA initiator process from a browser. The processOrder method is invoked as part of BEA's process execution. It invokes the ebXML MSH internally and passes the payloads (received as method arguments) to the MSH. The MSH creates a new conversation and encapsulates its ID in an ebXML SOAP envelope along with the service name OrderProcessing and action name processOrder. It creates an ebXML message using this envelope and adds payloads as SOAP Attachments. The ebXML message created is sent to the trading partner identified by to-id in the control annotation. The reliability and security semantics for sending a message are derived from BEA's profile in the Trading Partner Repository while the transport endpoint is derived from to-id Avitek's profile.

On Avitek's end, the receiving MSH extracts the ebXML service name, action name, and conversation ID from the incoming message to identify the conversation this message belongs to. If the message indicates a new conversation, it creates a new instance of a business process using the service name to identify the process type. It uses an ebXML action name to invoke the right node in the process. Thus, the processOrder method of Avitek is invoked and user code defined in this method is executed.

As Avitek's process executes further, it reaches the ClientCallback node that is supposed to respond to BEA. The ClientCallback method processInvoice internally calls sending ebXML-MSH, which creates an ebXML envelope encapsulating the same conversation ID it received from BEA's message to indicate that the response is part of the same conversation. It also encapsulates the processInvoice as the action name and OrderProcessing as the ebXML service name in the outgoing message.

The message eventually reaches BEA, which uses the conversation ID from the incoming message to identify the related conversation. The system thus guarantees messages sent and received via an ebXML control instance are part of a single ebXML conversation.

The conversation ID also helps to identify the business process instance the message should berouted to. Finally, an ebXML Action name is used to route the message to BEA's processInvoice ControlCallback node. User code specified in the processInvoice method is executed next.

Configuring the Trading Partner Repository

At this point, the business processes for BEA and Avitek are designed and tested using out-of-the-box trading partners. To run these business processes in production, we need to configure profiles for BEA and Avitek. This configuration can be done via the WebLogic Integration Console or using the offline Bulkloader tool. Listing 4 displays TPM data that can be directly uploaded for our example. As seen in the listing, the trading partner configuration is now made easy with only two root elements: trading partner and service. The trading partner element has protocol binding as child element. A trading partner may have one or more protocol binding per protocol name and version. Figure 4 displays the BEA-ebxml20-binding protocol binding. As seen, it contains Protocol information, transport endpoints, reliable messaging parameters and a security configuration for BEA.

The other root element, service, has one or more service profiles, which tie together the trading partners that need to converse. It is configured differently for the initiator and participant (see Figure 5). The participant trading partner Avitek has a business process name as the service name and the service type is "PROCESS". The initiator trading partner BEA has an ebXML control name specified as the service name and the service type is "SERVICECONTROL".

The service ebXML.order.OrderControl in Listing 4 indicates that BEA, when interacting with Avitek, uses the BEA-ebxml20-binding binding. BEA communicates with Avitek using ebXML 2.0 protocol and OnceAndOnlyOnce reliable messaging mode. Similarly the service /ebxmlWeb/ebxml/order/Seller.jpd is used by Avitek when interacting with BEA.

Messaging Reliably

BEA WebLogic Integration 8.1 supports four reliable messaging modes for ebXML 2.0 Point-to-Point Messaging: Once and Only Once, Atmost Once, Atleast Once, and BestEffort. For ebXML 1.0 messaging, it provides two Reliable Messaging modes: BestEffort and Once and Only Once.

- ***BestEffort:*** The fastest mode as it does not care for acknowledgement or duplicates.
- ***Atleast Once:*** Acknowledgement is requested but duplicate messages aren't checked.
- ***Atmost Once:*** Duplicates are checked but the message is not acknowledged.

As Java development evolves, so does JProbe®

JProbe®

Find the cause of J2EE code performance, memory and threading problems faster than ever before with Quest JProbe. New investigative features for finding memory problems combined with dramatic performance improvements mean even quicker resolution of problems in your application, servlet, JSP and EJB code.

JProbe Suite

JProbe Profiler

JProbe Memory Debugger

JProbe Threadalyzer

JProbe Coverage

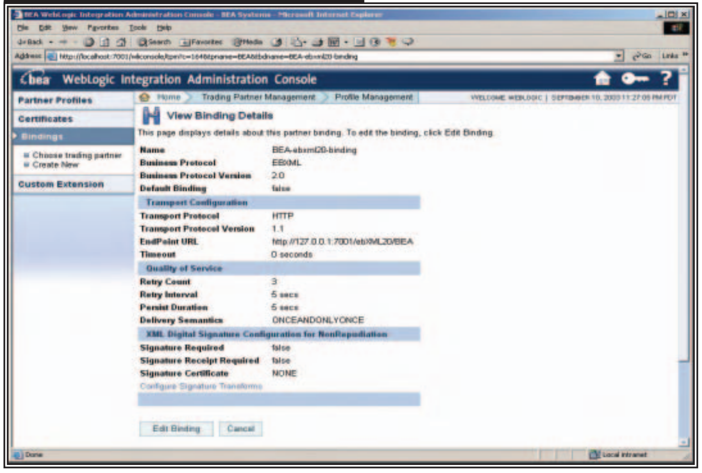
Part of the **Quest Performance Management Suite for the J2EE Platform**

QUEST SOFTWARE®

For more info and a free eval, visit:

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

FIGURE 5


BEA WebLogic Integration Administration Console - BEA Systems - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/wliconsole/tpm?c=1646&tpname=BEA&bdname=BEA-ebxml20-binding
bea WebLogic Integration Administration Console
Partner Profiles
Certificates
Bindings
Choose trading partner
Create New
Custom Extension
Home
Trading Partner Management
Profile Management
WELCOME WEBLOGIC | SEPTEMBER 10, 2003 11:27:05 PM PDT
Edit Binding
Cancel
Done
Local intranet

View Binding Details

This page displays details about this partner binding. To edit the binding, click Edit Binding.

Name	BEA-ebxml20-binding
Business Protocol	EBXML
Business Protocol Version	2.0
Default Binding	false
Transport Configuration	
Transport Protocol	HTTP
Transport Protocol Version	1.1
EndPoint URL	http://127.0.0.1:7001/ebXML20/BEA
Timeout	0 seconds
Quality of Service	
Retry Count	3
Retry Interval	5 secs
Persist Duration	5 secs
Delivery Semantics	ONCEANDONLYONCE
XML Digital Signature Configuration for NonRepudiation	
Signature Required	false
Signature Receipt Required	false
Signature Certificate	NONE

Configure Signature Transforms

TPM Services

- ***Once and Only Once:*** The most reliable mode as the message is acknowledged and the duplicates are checked.

When BEA sends a purchase order, Avitek's MSH sends a low-level acknowledgement that can be viewed in JPD. BEA, as the initiator, can simply model the acknowledgement for viewing by dragging/dropping the onAck ebXML control method. Avitek, as the participant, does not have the ebXML control. It uses the ClientRequest node and system-level method onAck to model acknowledgments.

Modelling Error Messages

It may happen that the message received by a peer-trading partner is erroneous. For example, Avitek cannot parse the purchase order sent by BEA. In that case, Avitek sends an error message to BEA. This message can be modelled in BEA's business process for viewing by dragging/dropping the onError ebXML control method. An ErrorList indicating actual errors can then be extracted from the ebXML envelope that is modelled as an argument to onError method.

Exposing ebXML Envelope

The SOAP-based ebXML Envelope sent with the ebXML message contains important elements like From and To Party; Message ID, used to track messages; ebXML Conversation ID, used to track the conversations; ErrorList, sent with the error message; and Signature, to name a few of them. With BEA WebLogic Integration 8.1, the ebXML envelope can be viewed in the business process as org.xmlsoap.schemas.soap.envelope.EnvelopeDocument XBean by specifying the annotation on the method used to receive the message in question. Listing 3 displays the @jpd:ebxml-method envelope method annotation used by Avitek's jpd while Listing 1 displays the control annotation @jc:ebxml-method envelope="{env}" used by BEA's jpd. The method argument specified in these annotations is the one that will contain the message envelope when the corresponding receive node is invoked.

The ebXML envelope for the outgoing message can be exposed in the business process by specifying XmlObject as the return type of method used to send the message in question. Each element of ebXML envelope can further be extracted and mapped to Java objects using the data transformation tool embedded in the WebLogic Workshop IDE. The schema for ebXML envelope required to view its elements for mapping is available as part of the system schemas that come with WebLogic Integration 8.1.

Message Tracking

BEA and Avitek can monitor messages they have sent and received and their status using the WebLogic Integration Console. Different levels of message tracking can be set in the TPM Repository at the root level. For granular control, it can also be set at the service profile level. There are three levels of message tracking: ALL, Metadata, and NONE. With the tracking level set to ALL, the contents as well as the Metadata of all the messages exchanged are tracked and displayed in the WebLogic Integration Console.

The metadata of the message includes a conversation ID related to message exchange; a business process ID, an action name representing the message exchange; a time when the message was sent/received; a message status for whether the message was successfully sent or received; a status description; the size of the message, action, transport protocol; and so on. With message tracking set to metadata, the contents are not tracked. The tracking level should be set as required. Using the highest level adversely impacts performance.

References

- *BEA WebLogic Integration 8.1:* http://commerce.bea.com/index.jsp
- *BEA WebLogic Integration Documentation*: http://e-docs.bea.com/workshop/docs81/doc/en/core/index.html
- *ebXML Tutorial:* An extensive tutorial to create various B2B Integration Scenarios with ebXML: http://dev2dev.bea.com/codelibrary/code/tptutorial.jsp
- *ebXML-MSH 1.0 Specification:* www.ebxml.org/specs/ebMS.doc
- *ebXML-MSH 2.0 Specification:* www.ebxml.org/specs/ebMS2.pdf
- *SOAP1.1 Specification:* www.w3.org/TR/2000/NOTE-SOAP-20000508
- *SOAP with Attachments specification:* www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211

Listing 1

```
package ebxml.order;

import com.bea.control.EBXMLControl;
import com.bea.data.RawData;
import com.bea.xml.XmlObject;
import com.bea.data.MessageAttachment;

/**
 * Defines a new ebXML control.
 * @jc:ebxml from="BEA-id" to="Avitek-id" ebxml-service-
name="OrderProcessing" ebxml-action-mode="non-default"
 */
public interface OrderControl extends EBXMLControl,
com.bea.control.ControlExtension
{

  /**
   * Send ebXML message to participant
   * @param payload ebxml message payload(s)
   *
   */
   public void processOrder(XmlObject payload);


    interface Callback extends EBXMLControl.Callback
    {

      /**
       * Receive ebXML message from participant
       * @param payload ebxml message payload(s)
       * @jc:ebxml-method envelope="{env}"
       */
       public void processInvoice(XmlObject payload, XmlObject env);

    }
}
```

Listing 2

```
package ebxml.order;
```

www.altova.com

xmlspy®
2004

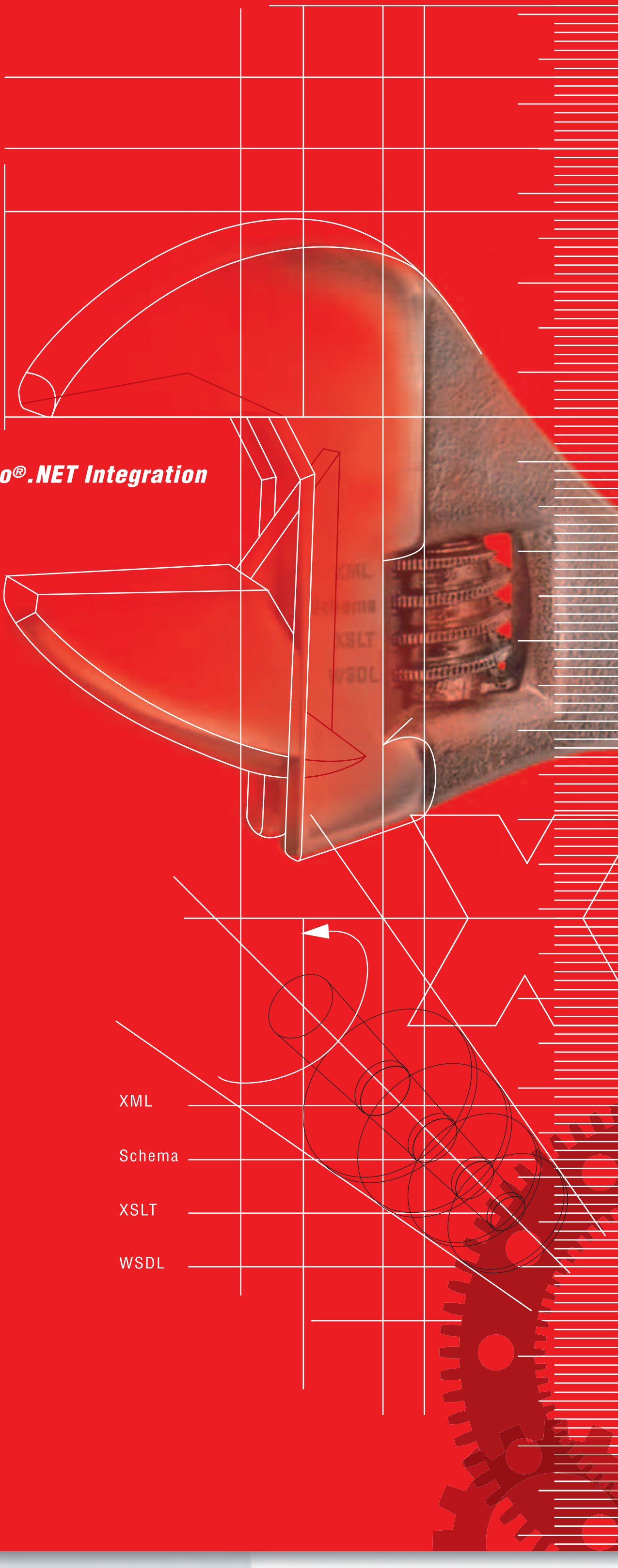
Microsoft® Visual Studio®.NET Integration

NEW **_XML-to-XML Mapping_**

XML Differencing

improved **_XML Schema/WSDL Editor_**

XSLT Debugger _and_ **_Code Generator_**



Build advanced XML and Web services applications with XMLSPY 2004, the industry leading XML Development Environment! XMLSPY provides intuitive editing views and powerful XML utilities to model, edit, transform, and debug all XML technologies, quickly and easily, eliminating unwanted bugs and headaches. XMLSPY is the right tool for any XML development task, and the ultimate productivity enhancer for Microsoft .NET, J2EE, and database developers. ***Download a free trial now!***

ALTOVA®
www.altova.com

```
import ...
/**
 * @jpd:process process::
 * <process name="Buyer">
 *   <clientRequest name="Start Buyer" method="startBuyer"/>
 *   <controlSend name="Send PurchaseOrder"
method="orderCtrlProcessOrder"/>
 *   <controlReceive name="Receive Invoice"
method="orderCtrl_processInvoice"/>
 * </process>::
 */
public class Buyer implements com.bea.jpd.ProcessDefinition
{
    ...

    /**
     * @common:control
     */
    private ebxml.order.OrderControl orderCtrl;

    public void startBuyer(com.bea.tutorial.order.OrderDocument x0)
    {
        this.order = x0;
    }

    public void orderCtrlProcessOrder() throws Exception
    {
        orderCtrl.processOrder(this.order);
    }

    public void orderCtrl_processInvoice(XmlObject payload, XmlObject
env)
    {
        this.invoice = payload;
    }

}
```

Listing 3

```
package ebxml.order;
import ...
/**
 * @jpd:process process::
 * <process name="Seller">
 *   <clientRequest name="Receive PurchaseOrder" method="processOrder"/>
 *   <controlSend name="get Invoice From FileSystem"
method="invoiceFileRead"/>
 *   <clientCallback name="Respond with Invoice"
method="clientResponseCallbackHandler"/>
 * </process>::
 *
 * binding="ebxml"
 *
 * @jpd:ebxml protocol-name="ebxml" ebxml-service-name="OrderProcessing"
ebxml-action-mode="non-default"
 */
public class Seller implements com.bea.jpd.ProcessDefinition
{

 ......

    /**
     * @jpd:ebxml-method envelope="{env}"
     */
    public void processOrder(com.bea.tutorial.order.OrderDocument x0,
XmlObject env )
    {
        ....
    }



    public void clientResponseCallbackHandler()
    {

        callback.processInvoice(this.invoice);

    }


    public void invoiceFileRead() throws Exception
    {
        ....
    }

    public interface Callback
    {

        void processInvoice(com.bea.tutorial.invoice.InvoiceDocument
x0);
    }

}
```

Listing 4

```
<?xml version="1.0" encoding="UTF-8"?>
<trading-partner-management
    xmlns="http://www.bea.com/2003/03/wli/tpm"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xsi:schemaLocation="http://www.bea.com/2003/03/wli/tpm TPM.xsd"
    message-tracking-default="ALL"
    message-trace="false"
    secure-audit-logging="false"
    test-mode="true">
    <trading-partner
        name="Avitek"
        status="ENABLED"
        type="LOCAL"
        is-default="false"
        business-id="Avitek-id">
        <ebxml-binding
            name="Avitek-ebxml20-binding"
            business-protocol-name="EBXML"
            business-protocol-version="2.0"
            is-default="false"
            is-signature-required="false"
            is-receipt-signature-required="false"
            delivery-semantics="ONCEANDONLYONCE"
            retries="3"
            retry-interval="5 seconds"
            persist-duration="5 seconds">
            <transport
                protocol="http"
                protocol-version="1.1"
                endpoint="http://127.0.0.1:7001/ebXML20/Avitek"
                timeout="0 seconds">
            </transport>
        </ebxml-binding>
    </trading-partner>
    <trading-partner
        name="BEA"
        status="ENABLED"
        type="LOCAL"
        is-default="false"
        business-id="BEA-id">
        <signature-certificate
            name="BEA-sig"
            password-alias="BEA-sig">
        </signature-certificate>
        <ebxml-binding
            name="BEA-ebxml20-binding"
            business-protocol-name="EBXML"
            business-protocol-version="2.0"
            is-default="false"
            is-signature-required="false"
            is-receipt-signature-required="false"
            delivery-semantics="ONCEANDONLYONCE"
            retries="3"
            retry-interval="5 seconds"
            persist-duration="5 seconds">
            <transport
                protocol="http"
                protocol-version="1.1"
                endpoint="http://127.0.0.1:7001/ebXML20/BEA"
                timeout="0 seconds">
            </transport>
        </ebxml-binding>
    </trading-partner>
    <service
        name="/ebxmlWeb/ebxml/order/Seller.jpd"
        service-type="PROCESS"
        business-protocol="EBXML">
        <service-profile
            local-trading-partner="Avitek"
            local-binding="Avitek-ebxml20-binding"
            external-trading-partner="BEA"
            external-binding="BEA-ebxml20-binding"
            status="ENABLED"
            message-tracking="ALL"/>
    </service>
    <service
        name="ebxml.order.OrderControl"
        service-type="SERVICECONTROL"
        business-protocol="EBXML">
        <service-profile
            local-trading-partner="BEA"
            local-binding="BEA-ebxml20-binding"
            external-trading-partner="Avitek"
            external-binding="Avitek-ebxml20-binding"
            status="ENABLED"
            message-tracking="ALL"/>
    </service>
</trading-partner-management>
```

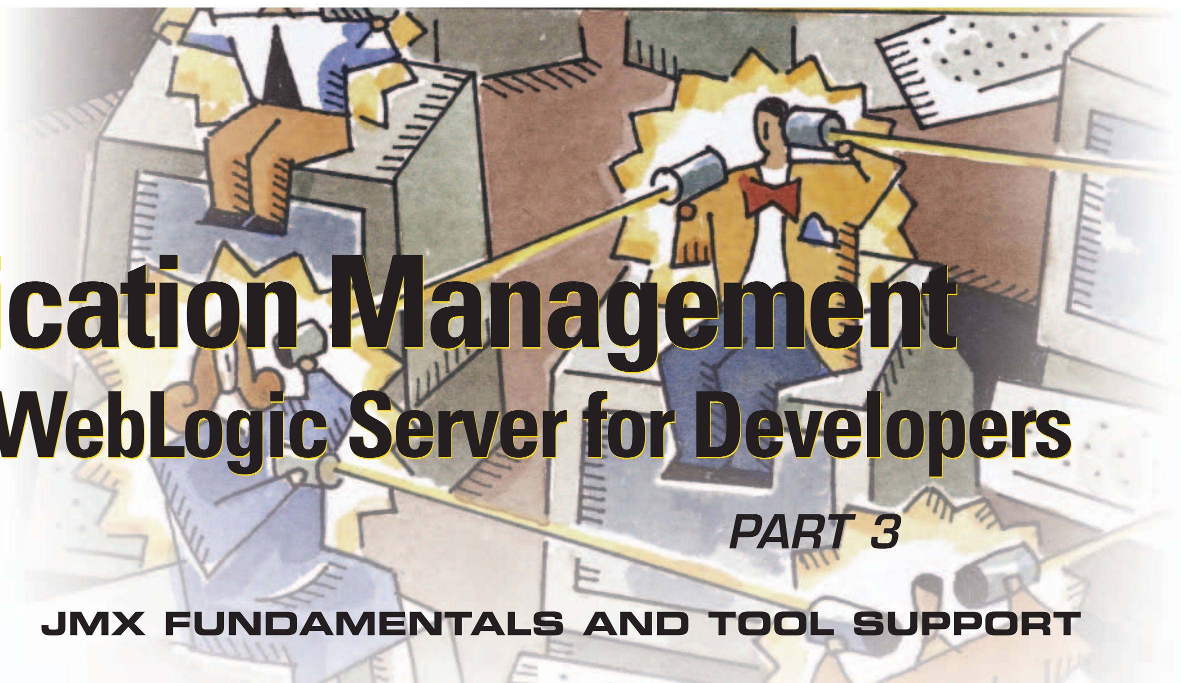
It’s all coming together.

You want Linux. Running your applications. With systems management and support you can trust. You want Red Hat Enterprise Linux: Seven platforms. Runs software you already use from BEA. Systems management via Red Hat Network. Backed by Red Hat.

www.redhat.com or call 1-888-2REDHAT x45011

redhat.

bea STAR Partner



Application Management with WebLogic Server for Developers

PART 3

JMX FUNDAMENTALS AND TOOL SUPPORT

BY **ROBERT PATRICK & VADIM ROSENBERG**

AUTHOR BIOS...

Robert Patrick is a director of technology in BEA's CTO Office and coauthor of the book Mastering BEA WebLogic Server: Best Practices for Building and Deploying J2EE Applications.

Vadim Rosenberg is the product marketing manager for BEA WebLogic Server. Before joining, Vadim spent 13 years in business software engineering, most recently at Compaq Computers.

CONTACT...

vadimr@bea.com

Robert.patrick@bea.com

This article is the third in a series on BEA WebLogic Server administration and management for developers.

The first installment (***WLDJ***, Vol. 2, issue 10) focused on WebLogic Server administration concepts and terminology, and the graphical tools for packaging an application and setting up and configuring a WebLogic Server domain. In the second article (***WLDJ***, Vol. 2, issue 11), we focused on application deployment, runtime management, and monitoring facilities available with WebLogic Server that did not require knowledge of JMX.

In this article, we'll discuss the basic concepts and terminology of JMX and the BEA WebLogic Server 8.1 JMX infrastructure. We will also show you how to use JMX-specific tools that come with WebLogic Server 8.1.

JMX Concepts and Terminology

BEA WebLogic Server 8.1 implements the Java Management Extensions (JMX) version 1.0 specification. This specification consists of two parts: the instrumentation and the agent specifications.

In the instrumentation level, JMX provides a specification for implementing manageable resources. *Manageable resources* can be anything from a single Java object to an entire application, from some sort of device to the user of an application, or anything else that is implemented in such a way that it can be managed by a JMX-compliant application. A manageable resource is implemented to allow JMX-compliant applications to manage them by defining one or more managed beans, better known as *MBeans*.

A JMX agent directly controls the manageable resources and makes them available to JMX-compliant applications. Agents typically live on the same machine as the resources that they control. An agent provides a registry of managed objects that the agent exposes to JMX-compliant applications; this registry is known as an *MBean Server*. In addition, the agent exposes an additional set of management services for loading classes dynamically, monitoring changes to MBean attributes, creating timers, and defining relationships between MBeans.

Managed Beans

An MBean is a Java class that provides management capabilities for controlling and monitoring a resource. These management capabilities are exposed through the MBean's *management interface*, which may be defined either statically at compile time, or dynamically at runtime. JMX calls an MBean that defines its management interface statically, a *standard MBean* and one that does it dynamically, a *dynamic MBean*.

A standard MBean is a Java class that exposes its management interface through a Java interface that the class implements. The interface class name must be derived from the implementation class name by adding the MBean suffix to the end of the implementation class name. For example, a standard MBean class called com.example.MyDevice would expose its management interface through an interface class called com.example.MyDeviceMBean.

MBeans expose attributes and operations. An attribute is always exposed through the MBean's management interface with getter and/or setter methods using the standard JavaBean naming conventions. The access to the specific attribute is indicated through the management interface by the presence, or lack thereof, of a getter or setter method. For example, a read-only attribute would not have a corresponding setter method exposed in the MBean's management interface.

An operation is any method that does not con-form to the conventions of an attribute's getter or setter method. A JMX management program uses introspection of the MBean's management interface to determine the management functionality the MBean exposes.

MBeans support notification. The JMX notification model is based on the Java event model. A *notification broadcaster* is an MBean that emits notifications. Management applications and others register their interest in these notifications with the broadcaster. Notifications are delivered to registered objects through their implementation of the NotificationListener interface.

The payload of a notification is sent using a Notification object, which contains a type, sequence number, timestamp, message, and any associated data. Notification types are expressed as a string using dot notation. For example, a type name for a notification emitted when a new device is brought online might be com.example.mydevice.start. Listeners can also define a *notification filter* that limits the types of notifications a listener receives. We will talk more about JMX notification in another article.

A dynamic MBean is a Java class that implements the javax.management.DynamicMBean interface, shown here:

```
public interface DynamicMBean
{
    Object getAttribute(String attribute);
    AttributeList getAttributes(String[] attrib-
utes);
    MBeanInfo getMBeanInfo();
    Object invoke(String actionName, Object[]
params,
                  String[] signature)
    void setAttribute(Attribute attribute);
    AttributeList setAttributes(AttributeList
attributes) ;
}
```

Dynamic MBeans are more flexible than standard MBeans. This is because their management interface is exposed through the return value of the getMBeanInfo() method rather than a statically defined interface class. Management applications are expected to use the information in the MBeanInfo class returned to determine the set of attributes and operations that can be manipulated. All subsequent attribute access or operation invocations are performed using the methods defined on the DynamicMBean interface.

The JMX specification also defines two additional types of MBeans: open MBeans and model MBeans. An *open MBean* is a dynamic MBean whose management interface is defined in terms of a limited set of specified Java types. The idea is that by restricting the types an MBean's management interface exposes, management applications can be packaged to include all of the required classes needed to manage a set of applications without requiring dynamic class loading of application-specific classes by the management application.

A *model MBean* is a dynamic MBean that provides a configurable, generic template that allows anyone to quickly instrument any resource. A JMX agent provides the implementation for the Model MBean that a resource provider can instantiate, configures the management interface it exposes, specifies the mapping between the management interface and the method that the agent should call, and registers it with the JMX agent. The idea is to significantly reduce the programming burden for adding manageability to a resource.

Now that you understand the basic MBean concepts, let's talk about JMX agents.

JMX Agents

A *JMX agent* is a Java component that acts as the intermediary between a management application and a set of MBeans. The agent is composed of an MBean server, a set of MBeans representing the managed resources, and a set of agent services. In addition, an agent typically includes some mechanism for providing remote access to support communication between the managed resources and management applications. The JMX 1.0 specification suggests that this support is achieved by the JMX agent providing one or more connectors and/or protocol adapters; exactly how these connectors or protocol adapters work is not specified.

MBean Server

The MBean server is the central registry for MBeans. All management operations on MBeans must go through the MBean server. It is a Java object that implements the javax.management.MBeanServer interface. If you look at the MBeanServer interface, part of which is shown in Listing 1, you will quickly see that the methods closely resemble those defined by the DynamicMBean interface. This allows the MBean server to provide a uniform interface to a management application regardless of the type of MBean that the resource developer chose to implement.

Notice that each method takes an ObjectName as its first argument. A management application uses the object name to specify the MBean, or set of MBeans, on which method invocation applies. Object names consist of two parts: a domain name and an unordered set of key properties. The canonical representation of the object name is a string with the following syntax:

```
[domainName]:property=value[,property=value]*
```

For example, the canonical name for the MBean that represents a WebLogic Server instance might be:

```
mydomain:Name=myserver,Type=Server
```

Object names also include support for specifying a set of MBeans. By specifying only a subset of information, JMX allows the agent to find any MBeans that match the supplied information. For example, specifying the object name mydomain:Type=Server would cause the agent to match all MBeans in mydomain of type Server.

Object names support implicit domain names and wildcards. To use an implicit domain name, simply omit the domain name from the object name and the agent will automatically apply the name to the default domain. For example, the object name :Name=myserver,Type=Server is the same as the example above if the default domain is mydomain. Object names can also contain the standard file globbing characters:

- * – matches any character sequence
- ? – matches any single character

While object names provide JMX with a great deal of flexibility, they also add complexity for users who are simply trying to determine the correct set of properties needed to identify a particular MBean. We will use object names later in our discussion of WebLogic JMX client tools.

Agent Services

JMX agents must also provide a set of management services:

- Dynamic class loading
- Monitoring
- Timers
- Relations

The dynamic class loading service, also known as the *m-let service*, allows a management application to instantiate and register MBeans with the MBean server by specifying the remote URL where the MBean implementations are available. (A discussion of the m-let service is beyond the scope of this article. See Chapter 8 of the JMX 1.0 specification for more information.)

The monitoring service allows a management application to tell the agent to observe the value of an MBean attribute at periodic intervals and to notify the management application when the attribute's value satisfies a particular condition. JMX defines three types of monitors for specifying the conditions in which the management application wants to be notified. They are:

- ***Counter:*** Observes integer-valued attributes for when they meet or exceed the configured value
- ***String:*** Observes string-valued attributes for when the attribute value either matches or differs from the configured value
- ***Gauge:*** Observes numeric-valued attributes for when they meet or exceed a high threshold value and/or meet or fall below a low threshold value

The timer service allows a management application to have the agent send out notifications at specific dates and times or repeatedly at a constant interval. We will talk more about the monitoring and timer services in another article.

The relation service allows a management application to define relationships between MBeans. Once defined, the agent is responsible for checking the validity of any existing relationships after any management operations are performed. Any operation that causes a relationship's values to change will cause the agent to generate a notification. (A detailed discussion of the relation service is beyond the scope of this article. See Chapter 11 of the JMX 1.0 specification for more information.)

Now that you have an understanding of the basic concepts of JMX, let's briefly discuss BEA WebLogic Server 8.1's JMX implementation.

WebLogic Server 8.1 JMX

BEA WebLogic Server 8.1 implements the JMX 1.0 specification. Almost every WebLogic Server resource is instrumented using MBeans, and any JMX-compliant management application can plug in and communicate with WebLogic Server's JMX agent implementation.

While this JMX-compliant interface is sufficient for any management task at hand, the fact is that the introspective and loosely typed nature of the MBeanServer interface makes writing JMX management programs to automate simple management tasks more difficult than we would like. As such, WebLogic Server also exposes its own strongly typed MBean interfaces as an alternative to the one defined by JMX. Through these interfaces, you can perform the same set of operations as you can through the standard JMX MBeanServer interface but do so using the more typical strongly-typed Java programming model that provides a much better compile-time checking of your application code. We will go into the details of each of these two approaches at another time.

WebLogic Server 8.1 defines three primary types of MBeans that serve different purposes:

- ***Configuration:*** Represents the domain's configuration information for its resources
- ***Runtime:*** Located on the same server as the managed resource that they represent, these MBeans, provide access to the runtime state of the resource
- ***Security:*** Represents the security providers for WebLogic Server

Configuration MBeans are further classified as:

- ***Administration:*** The configuration MBeans located on the domain's administration server. To modify a domain's configuration, you need to modify the appropriate administration MBeans on the admin server.
- ***Local Configuration:*** Local replicas of the administration MBeans that the individual server uses for performance reasons. You should never modify a local configuration Mbean.

To find a description of all of the WebLogic Server 8.1 MBeans, see the BEA WebLogic Server 8.1 Javadocs for the weblogic.management.configuration, weblogic.management.runtime, and weblogic.management.security.* packages at http://edocs.bea.com/wls/docs81/javadocs/index.html.

WebLogic JMX Client Tools

BEA WebLogic Server 8.1 includes two client tools for accessing its underlying JMX MBeans. The first tool is the weblogic.Admin utility discussed in our last article. WebLogic Server 8.1 also includes the wlconfig Ant task that allows you to easily write scripts to create or modify a domain's configuration. With both of these tools, you must understand the object naming scheme that WebLogic Server uses.

WebLogic Server uses the standard JMX canonical representation for object names. Most object names include the Name and Type properties and many require a Location property.

The Type property value is determined by the class name of the MBean and whether or not the WebLogic MBean is an administration, local configuration, or runtime MBean. For administration MBean types, the Type value is derived by simply removing the MBean suffix from the configuration MBean's interface class name. For example, MBeans that implement the weblogic.management.configuration.JDBCConnectionPoolMBean class have a Type value of JDBCConnectionPool.

Runtime MBeans have a similar scheme so that the Type for MBeans that implements the weblogic.management.runtime.JDBCConnectionPoolRuntimeMBean interface is JDBCConnectionPoolRuntime. To specify the Type for a local configuration MBean, simply add the Config suffix to the Type of the equivalent administration MBean. Following our previous examples, the local configuration MBean's Type value would be JDBCConnectionPoolConfig.

Both runtime and local configuration MBeans require you to specify the Location property in their object names. The value of this property is the name of the WebLogic Server instance where the MBean of interest resides. This location information is required since many other servers in the domain may have an MBean with the same Name and Type property values.

Now that you understand the object naming scheme, let's look at each of the JMX client tools and see how you can use them to automate basic administration tasks.

weblogic.Admin

The weblogic.Admin utility supports the following JMX commands:

- ***CREATE:*** Creates an instance of an administration MBean
- ***DELETE:*** Deletes an instance of an administration MBean
- ***GET:*** Displays the object name and properties of an MBean
- ***INVOKE:*** Executes the method on the targeted MBean
- ***QUERY:*** Searches for MBeans whose object names match a pattern
- ***SET:*** Sets the value of an configuration MBean's property

There is really no way for us to explain all possible uses of these commands – there are just too many of them to fit here. We'll provide just a few examples for you to get a feel for what you can do.

To create a new JMS connection factory, use the CREATE command:

```
java weblogic.Admin -url t3://AdminHost:7001
     -username weblogic -password weblogic
     CREATE -name jms/MyConnectionFactory
            -type JMSConnectionFactory
```

To set the JNDI name of your new connection factory, use the SET command. However, the SET command requires that you specify the MBean's object name. If you're unsure about the exact object name, use the GET command to get all of the MBeans of the JMSConnectionFactory type:

Cyanea

How Can You Sink Your Application Overhead?

With Cyanea/One™, your company can ensure that the resource consumption of its J2EE applications stays below desired thresholds. Through Cyanea/One's *Performance Analysis and Reporting (PAR)* engine, you can quickly baseline an application and trend its performance and resource utilization over time. Automatically identify hotspots within the code and take corrective action. Be instantly notified when key threshold conditions are approaching. Improve service levels and customer satisfaction. Accomplish all these without touching your code or even requiring knowledge of your application.

...Keeping **Your Applications Under PAR.**

Find PAR at:
www.cyanea.com/wldj/underpar.html
1-877-CYANEA8 | sales@cyanea.com

The World's Leading Java Resource!

JAVA DEVELOPER'S JOURNAL

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

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

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

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

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

JDJEDGE CONFERENCE & EXPO OCTOBER SAN JO
JAVA DEVELO JOUR
The World's Lea
September
JAVADEVELOPERSJOURNAL.COM
FULL CONFERENCE PROGRAM
WEB SERVICES EDGE
CONFERENCE: October 1-3, 2002
EXPO: October 2-3, 2002
INSIDE PAGE 75
From the Editor Alan Williamson pg. 7
J2EE Editorial Ajit Sagar pg. 9
J2SE Editorial Jason Bell pg. 34
J2ME Editorial Jason R. Briggs pg. 52
Career Opportunities Bill Baloglu & Billy Palmieri pg. 104
Cubist Threads Blair Wyman pg. 106
RETAILERS PLEASE DISPLAY UNTIL NOVEMBER 30, 2002
SYS-CON MEDIA
Whole House Aud
written by Bill Ray page 60
EJB Techniques: Primary Keys and Containe Managed Persistence in EJB 2.0 *Dealing with EJBs*
Feature: Introduction to Session Management *Create a better experience for your Web site vis*
J2EE Projects: Delivering a J2EE Application Suite *A case study*
Industry Commentary: Combating the 'Object Crisis' *The foundation for Java proficiency*
Java Techniques: Java Design *Using interfaces and abstract classes to create flexible code*
Feature: Creating a Custom Launcher *Say goodbye to java.lang.NoClassDefFound*
Java & Bluetooth: Wireless J2ME Apps *Share and collaborate in a wireless network*
Product Review: FULCRUM Professional Edition 1.1 *by AccelTree*
First Look: IntelliJ IDEA 3.0 *by JetBrains, Inc.*

Offer subject to change without notice

```
java weblogic.Admin -url t3://AdminHost:7001
     -username weblogic -password weblogic
     GET -type JMSConnectionFactory
```

The MBean name is shown in the output below:

```
{MBeanName="medrec:Name=jms/MyConnectionFactory,T
ype=JMSConnectionFactory"{AcknowledgePolicy=All}{
AllowCloseInOnMessage=false}...
```

Now, use the SET command to set the JNDI name:

```
java weblogic.Admin -url t3://AdminHost:7001
     -username weblogic -password weblogic
     SET -mbean
"medrec:Name=jms/MyConnectionFactory,
                    Type=JMSConnectionFactory"
         -property JNDIName
jms/MyConnectionFactory
```

The last thing to do is to deploy your new connection factory to the MedRecServer. Use the SET command again, but since the setTargets() method (on the JMSConnectionPoolMBean interface) takes an array of MBeans as the argument, you must use the object name for the MedRecServer's Server MBean:

```
java weblogic.Admin -url t3://AdminHost:7001
     -username weblogic -password weblogic
     SET -mbean
"medrec:Name=jms/MyConnectionFactory,
                    Type=JMSConnectionFactory"
         -property Targets
"medrec:Name=MedRecServer,
                                   Type=Server"
```

To add a new user, use the INVOKE command to invoke the createUser() method on the appropriate authenticator's UserEditorMBean interface (assuming your authenticator implements this interface). To find the object name of your authenticator's security MBean, use the QUERY command:

```
java weblogic.Admin -url t3://AdminHost:7001
     -username weblogic -password weblogic
     QUERY -pattern "Security:*"
```

The MBean name of interest is shown here:

```
{MBeanName="Security:Name=myrealmDefaultAuthentic
ator" {ControlFlag=SUFFICIENT}...
{MBeanName="Security:Name=myrealmMedRecSampleAuth
enticator"{ControlFlag=SUFFICIENT}...
```

Since the MedRecSampleAuthenticator does not implement the UserEditorMBean interface, you must add the user to the DefaultAuthenticator using the INVOKE command:

```
java weblogic.Admin -url t3://AdminHost:7001
     -username weblogic -password weblogic
     INVOKE -mbean
"Security:Name=myrealmDefaultAuthenticator"
            -method createUser rpatrick pass-
word "Robert Patrick"
```

To delete our example JMS connection factory, use the DELETE command:

```
java weblogic.Admin -url t3://AdminHost:7001
     -username weblogic -password weblogic
     DELETE -mbean
"medrec:Name=jms/MyConnectionFactory,
                         Type=JMSConnectionFactory"
```

For more information on the weblogic.Admin commands for manipulating WebLogic MBeans, see http://edocs.bea.com/wls/docs81/admin_ref/cli.html#1240048.

wlconfig Ant task

The wlconfig Ant task provides a mechanism by which you can connect to your domain's admin server and query, create, or change the attributes of the domain's administration MBeans. BEA WebLogic Server 8.1 also provides four other Ant tasks: wlcompile, wlappc, wlserver, and wldeploy. These tasks allow you to fully automate the end-to-end process of creating and configuring your domain as well as compiling, packaging, and deploying your application. To use wlconfig, you need to write an Ant script. For more information on Ant, see http://ant.apache.org.

To connect to our MedRec administration server, we would use the following wlconfig tag in our Ant script:

```
<wlconfig url="t3://AdminHost:7001"
    username="weblogic" password="weblogic">
  ...
</wlconfig>
```

wlconfig supports five nested tags with similar meanings to the equivalent weblogic.Admin commands covered earlier: create, delete, get, query, and set. We will provide just a few examples here to give you a feel for what you can do.

Listing 2 is an example of the use of wlconfig to modify your domain's configuration. We use the query tag to get a reference to the MedRecServer's Server MBean and store it in a property called medrecserver for use later in the script. Next, we use the create tag to create a JMS file store and store a reference to the newly created MBean in the myfilestore property. We use a nested set tag to specify the JMS file store's directory attribute. Then we use the create tag again to create a JMS server. Notice that the nested set tags use the properties we previously created to supply the appropriate MBeans when setting the attribute values of the JMS server's MBean.

– contunued on page 48

Listing 1

```
public interface MBeanServer
{
    ...
    Object getAttribute(ObjectName name, String
attribute);
    AttributeList getAttributes(ObjectName
name,
                                String[]
attributes);
    String getDefaultDomain();
    ...
    MBeanInfo getMBeanInfo(ObjectName name);
    ...
    Object invoke(ObjectName name, String
operationName,
                  Object[] params, String[]
signature);
    ...
    void setAttribute(ObjectName name,
Attribute attribute);
    AttributeList setAttributes(ObjectName
name,

AttributeList attributes);
    ...
}
```

Listing 2

```
<wlconfig url="t3://AdminHost:7001"
       username="weblogic" password="weblog-
ic">
  <query domain="medrec" type="Server"
         name="MedRecServer" property="medrec-
server"/>
  <create type="MyJMSFileStore"
          name="MyJMSFileStore"
property="myfilestore">
    <set attribute="Directory" value="./jms-
store"/>
  </create>
  <create type="JMSServer" name="MyJMSServer">
    <set attribute="Store" value="${myfile-
store}"/>
    <set attribute="Targets" value="${medrec-
server}"/>
    <create type="JMSQueue" name="MyQueue">
      <set attribute="JNDIName"
value="jms/MyQueue"/>
    </create>
  </create>
</wlconfig>
```

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

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.

☐ **MX Developer's Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $167 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $29.99 /	Save: $60
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $137 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $40
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $127 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $30
Digital Edition - One Year (12)	You Pay: $19.99	

☐ **Linux World Magazine**

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

☐ **Java Developer's Journal**

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

☐ **Web Services Journal**

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

☐ **.NET Developer's Journal**

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

☐ **XML-Journal**

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

☐ **WebLogic Developer's Journal**

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

☐ **ColdFusion Developer's Journal**

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

☐ **Wireless Business & Technology**

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

☐ **WebSphere Developer's Journal**

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

☐ **PowerBuilder Developer's Journal**

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

3-Pack

Pick any 3 of our magazines and save up to $275^00

Pay only $175 for a 1 year subscription plus a FREE CD

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

6-Pack

Pick any 6 of our magazines and save up to $350^00

Pay only $395 for a 1 year subscription plus 2 FREE CDs

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

9-Pack

Pick 9 of our magazines and save up to $400^00

Pay only $495 for a 1 year subscription plus 3 FREE CDs

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

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

SYS-CON MEDIA

SERVER

Enterprise software applications are complex, but almost certainly more complex is the underlying software that provides services and resources to these applications. There are different types of software that fall into the latter category, one of those being a Java application server, which of course for this article is the BEA WebLogic Application Server.

conf2admin

AUTOMATE SCRIPT CREATION FOR YOUR WEBLOGIC SERVER DOMAIN CONFIGURATION

BY MIKE JASNOWSKI

In this article I'll examine the use of scripting languages with WebLogic SErver and use code generation to facilitate the creation of scripts.

Benefits of Scripting Languages

To be more productive, system administrators create scripts that enable them to rapidly execute repetitive tasks. These scripts can execute tasks for configuration and control. In the case of WebLogic Server, these scripts are used to do things like start and stop servers, or configure some aspect of a server.

BEA WebLogic Server has different types of scripting languages available today, two of which are shipped with WebLogic Server and one of which is an open-source program called WLShell. The two languages that ship with WebLogic Server are weblogic.Admin and WLAnt. The weblogic.Admin utility interprets commands that enable you to accomplish a variety of administrative tasks. The syntax of the language weblogic.Admin understands is proprietary in nature. The WLAnt language is a set of Ant tasks that accomplish tasks similar to those of weblogic.Admin. The WLShell language, while accomplishing similar tasks, is another syntax to learn, but adds an interactive shell for executing commands and interacting with the domain.

As with any programming language, there is a learning curve before you become comfortable with it. This learning process encompasses the language fundamentals and how to effectively use the language to your best advantage. After you learn the language fundamentals, you must then learn how to apply this to the WebLogic Server domain. Let's briefly look at how to apply each of these scripting languages to a WebLogic Server domain entity, the server.

WebLogic Server Domain

The BEA WebLogic Server has many components that provide runtime services and resources to applications running in it. Some of the components you'll find in a WebLogic Server domain are servers, clusters, JDBC Connection Pools, JMS servers, and JMS Queues to name a few. Each of these scripting languages is applied differently to these objects. For example, writing a script to create a WebLogic Server, along with some properties, for each of the languages would look like Listing 1. (*Note:* Some statements have been omitted for brevity.)

As you can see, each language has a different syntax for doing the same task, but each conceptually works in the same way, and for different types of WebLogic Server entities. Writing and maintaining these scripts can in some cases be a time-consuming process, keeping in mind that you have to be familiar with the syntax. In some cases, you might use scripts to help in the creation of an entirely new domain. Writing such a large script would be a time-consuming and potentially error-prone process.

We can, however, enhance the process of script writing using code-generation, which enables you to:

- Rapidly create reusable management code snippets
- Reduce the learning curve of one of the languages
- Reduce the learning curve of applying the language to the WebLogic Server domain
- Provide the ability to generate scripts in the language of your choice

Introducing conf2admin

conf2admin is a utility that processes a config.xml and produces output in the form of a scripting language. The current implementation includes support for weblogic.Admin, but Ant and WLShell are planned. The primary features of conf2admin are:

- Runs against offline configuration
- Generates scripts in the scripting language of your choice
- Generates an undo file

AUTHOR BIO

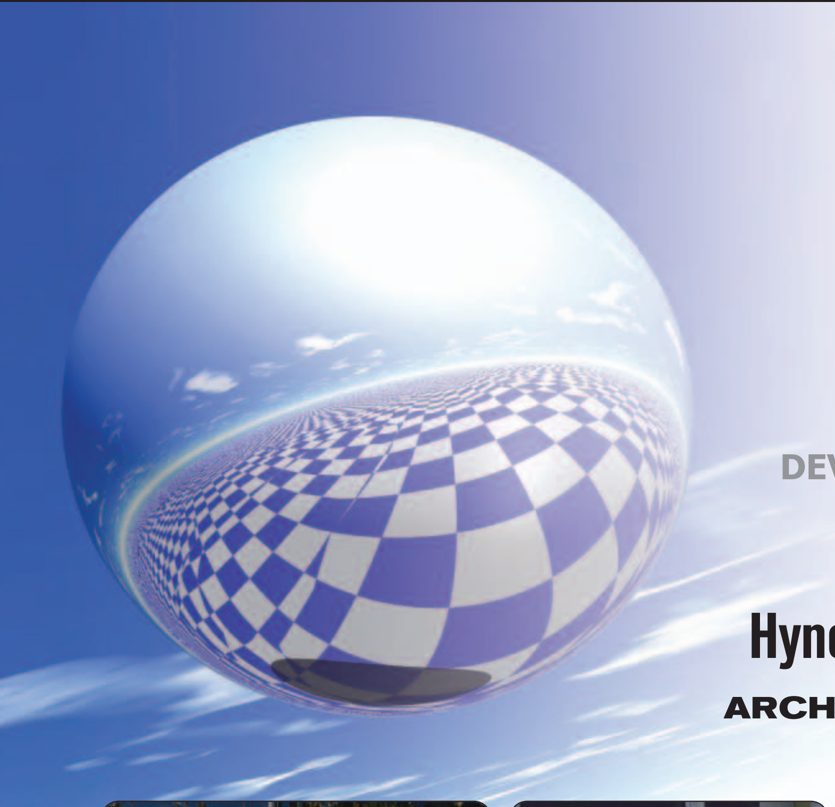
Mike Jasnowski is a senior software engineer on the BEA WebLogic Server Administration Console team. He has been involved in development for almost 20 years and in many industries. Mike is a contributing author to several books and author of JMX Programming (Wiley).

CONTACT...

mjasnows@bea.com

4TH Annual

International Developer Conference & Expo



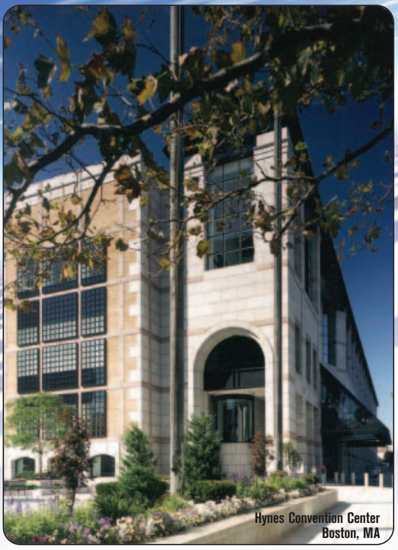
edge 2004 EAST

DEVELOPMENT TECHNOLOGIES EXCHANGE

February 24-26, 2004

Hynes Convention Center, Boston, MA

ARCHITECTING JAVA, .NET, WEB SERVICES, MX, OPEN SOURCE, AND XML



PROVIDING DEVELOPERS THE LATEST IN:

- Application Integration
- Desktop Java
- Mobility
- ASP.NET
- VS.NET
- JavaServer Faces
- SOA
- Interoperability
- Rich Internet Applications

Preregister for ***FREE*** full-day tutorials featuring the hottest Java, .NET, XML, Web Services, and Security issues. (ADVANCED REGISTRATION REQUIRED)

FREE Exhibition Floor (ADVANCED REGISTRATION REQUIRED)
Meet with industry-leading companies showcasing the newest products and services.

- Product Demonstrations Theater
- Opening Night Reception
- Internet Café
- Case Study Forum

Register By December 19, 2003 SAVE Up To $400

MX ® WebSphere XML Microsoft .net

For more information
visit www.sys-con.com/edge2004east
call 201 802-3069
e-mail events@sys-con.com

Over 100 participating companies will display and demonstrate over 300 developer products and solutions.

Over 3,000 C-Level Executives, Systems Integrators, System Architects, Developers, and Project Managers will attend the conference and expo.

Over 100 sessions – training, certifications, seminars, case studies, and panel discussions – promise to deliver real-world benefits, the industry pulse, and proven strategies.

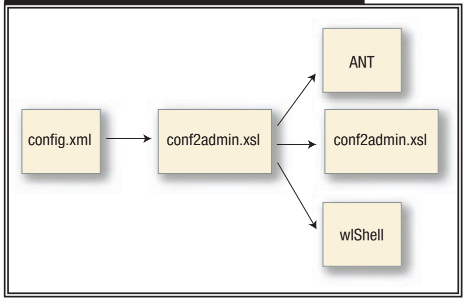
Full-Day Tutorials Targeting
• Java • .NET • XML • Web Services
• Open Source

Conference program available online!
www.sys-con.com/edge2004east

- Generates a script for deploying applications in the domain
- Excludes certain WebLogic Server entities from output

The BEA WebLogic Server domain configuration is an XML format file that describes the configuration for servers, clusters, and applications to name a few. The conf2admin utility works by executing an XSLT transformation against the config.xml document, and has XSL templates that produce the commands necessary for creating and setting values on WebLogic Server domain entities.

FIGURE 1



conf2admin concept

Figure 1 illustrates the general concept of conf2admin. Before we continue, let's take a brief look at the core technology behind conf2admin, the Extensible Stylesheet Language Transformations or XSLT.

XSLT

XSLT is an XML-based transformation language that works by matching templates against markup-like elements and attributes in an XML document. The templates in conf2admin match against elements in the BEA WebLogic Server domain configuration. The output from these templates is the script code that you will execute using weblogic.Admin or some other script interpreter. The following code snippet shows what a template looks like.

```
<xsl:template match="//Server">

  <!—specify contents of template here ‡

</xsl:template>
```

XSLT is a good choice for working with an XML-based configuration. We can get document parsing for free from the XML parser, and a parser for the templates. Writing an XSLT template is pretty straightforward. I won't cover the XSLT specification in depth, but will focus on the sections of the stylesheet relevant to the code generation.

Applying conf2admin to Medrec

We'll take the conf2admin utility and run it against the WebLogic Server MedRec samples domain configuration. The MedRec sample ships with BEA WebLogic Server and includes a variety of WebLogic Server resources. Depending on which release of BEA WebLogic Server you have, your MedRec domain configuration might be slightly different. We'll run the utility and focus on the MedRecServer and JDBCConnectionPool resources that will be generated. Those resources are represented by entries in the domain configuration that look like Listing 2.

When you run the conf2admin utility against the MedRec config.xml, the resulting file will be filled with many lines of generated code; the two snippets for our example deal with the server and the JDBCConnectionPool (see Listing 3)

From the output you can see that a command for CREATE was generated that will create the server. There were also four SET commands created, each representing the set for an attribute on that Server. Listing 4 lists similar results for the JDBCConnectionPool.

The conf2admin utility matches against elements in the configuration, in this case the server and JDBCConnectionPool elements, and instantiates the template for emitting CREATE commands. The stylesheet does not name the individual elements it looks for; it merely matches against whatever elements it finds. Next, within the template that emits the CREATE commands, conf2admin begins processing attributes and emitting commands to SET properties on the newly created server or JDBCConnectionPool entities. This flow of processing continues until the entire document has been processed.

The conf2admin utility will also emit some comment blocks, which indicate the source location in the config.xml that the generated code came from. So in the example of the MedRecServer, the comment block looks like this:

```
#==============================================
#==
#== Creating MBean of type Server
#==
#== Origin: /[1]/Domain[1]/Server[1]
#==
#==============================================
```

The origin comment indicates the location in the document expressed as an XPath expression. The numbers in brackets are the indices of the element since in some cases there can be several of a particular type.

Once the output is generated, you can use that file as input into the weblogic.Admin interpreter in batch mode. The documentation for conf2admin describes how to configure it to generate commands for stand-alone interpretation of script commands. The default is to generate commands to be used in batch mode of weblogic.Admin.

"XSLT is a good choice for working with an XML-based configuration"

Summary

In this article I looked at the scripting languages that are available for BEA WebLogic Server and how you can use code generation to enable quick creation of management scripts. While seemingly powerful, there are some limitations to using XSLT to generate code. It does not include a friendly interface for looping to reprocess parts; instead, you must recursively call a template until you are finished processing. There is also no real concept of flow control within a template.

Links

The following links may provide you with more details about the technologies and concepts discussed here.

- *conf2admin on dev2dev:* http://dev2dev.bea.com/resourcelibrary/utilitiestools/adminmgmt.jsp
- *WLShell:* www.wlshell.com
- *WebLogic Admin:* http://e-docs.bea.com/wls/docs81/admin_ref/cli.html
- *WebLogic Server Ant tasks:* http://edocs.bea.com/wls/docs81/admin_ref/ant_tasks.html
- *XML 1.0 (Second Edition):* www.w3.org/TR/REC-xml
- *XPath 1.0:* www.w3.org/TR/xpath
- *XSLT 1.0:* www.w3.org/TR/xslt

Listing 1

```
For WLShell:
mkdir /Server/MedRecServer
cd /Server/MedRecServer
set JavaCompiler “javac”
set ListenPort 7001
set IIOPEnabled “false”
set InstrumentStackTraceEnabled “false”

For WLAnt:
-<create name="MedRecServer" type="Server">
  <set attribute="JavaCompiler" value="javac" />
  <set attribute="ListenPort" value="7001" />
  <set attribute="IIOPEnabled" value="false" />
  <set attribute="InstrumentStackTraceEnabled" value="false" />
  </create>

For weblogic.Admin:

CREATE -mbean mydomain:Name=MedRecServer,Type=Server
SET -mbean mydomain:Name=MedRecServer,Type=Server -property JavaCompiler
"javac"
SET -mbean mydomain:Name=MedRecServer,Type=Server -property ListenPort
"7001"
SET -mbean mydomain:Name=MedRecServer,Type=Server -property IIOPEnabled
"false"
SET -mbean mydomain:Name=MedRecServer,Type=Server -property
InstrumentStackTraceEnabled "false"
```

Listing 2

```
<Server JavaCompiler="javac" ListenPort="7001" Name="MedRecServer"
IIOPEnabled="false" InstrumentStackTraceEnabled="false">
  <ExecuteQueue Name="default" ThreadCount="15" />
  <SSL Name="MedRecServer" Enabled="true" ListenPort="7002" />
  </Server>

<JDBCConnectionPool CapacityIncrement="1"
DriverName="com.pointbase.jdbc.jdbcUniversalDriver" InitialCapacity="1"
MaxCapacity="10" Name="MedRecPool" Password="MedRec"
Properties="user=MedRec" RefreshMinutes="0" ShrinkPeriodMinutes="15"
ShrinkingEnabled="true" Targets="MedRecServer"
TestConnectionsOnRelease="false" TestConnectionsOnReserve="false"
URL="jdbc:pointbase:server://localhost/demo" />
```

Listing 3

```
CREATE -mbean mydomain:Name=MedRecServer,Type=Server
SET -mbean mydomain:Name=MedRecServer,Type=Server -property JavaCompiler
"javac"
SET -mbean mydomain:Name=MedRecServer,Type=Server -property ListenPort
"7001"
SET -mbean mydomain:Name=MedRecServer,Type=Server -property IIOPEnabled
"false"
SET -mbean mydomain:Name=MedRecServer,Type=Server -property
InstrumentStackTraceEnabled "false"
```

Listing 4

```
CREATE -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
CapacityIncrement "1"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
DriverName "com.pointbase.jdbc.jdbcUniversalDriver"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
InitialCapacity "1"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
MaxCapacity "10"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
Password "foobar"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
Properties "user=MedRec"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
RefreshMinutes "0"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
ShrinkPeriodMinutes "15"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
ShrinkingEnabled "true"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
TestConnectionsOnRelease "false"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property
TestConnectionsOnReserve "false"
SET -mbean mydomain:Name=MedRecPool,Type=JDBCConnectionPool -property URL
"jdbc:pointbase:server://localhost/demo"
```

Application Management with NetIQ AppManager Suite

SHOW HOW YOUR INVESTMENT DOLLARS ARE EARNING THEIR KEEP

Reviewed by
LAURENCE MARONEY

As budgets are shrinking at the same pace that requirements are growing, there is a squeeze on enterprises to show value for the dollars spent on expensive software and hardware for running applications. It is clear that software such as application servers, while offering more and more features, also costs more and more, and that the accountability factor is fast becoming a priority for information professionals. However, as the complexity of typical systems continues to increase, finding the strategy to optimize bang for the buck grows exponentially in difficulty.

Enter an exploding area of applications that once may have been deemed a luxury, but is now considered paramount for the successful operation of your enterprise.

In this space NetIQ brings to the table the amazingly sophisticated and intricate AppManager Suite. It is a complex tool, but it miraculously layers the complexity of managing large systems. As a result you get an easy learning curve to get you in quick, but the slightly disturbing feeling that you are sitting on top of a Saturn 5 rocket, with the power just waiting to be unleashed.

The NetIQ AppManager Suite is a collection of applications that are used for managing the performance and availability of a number of operating systems and applications. The basic suite is used to manage operating systems such as Windows, Unix, and Linux, and add-on modules are available to monitor Web servers or application servers that aren't included in the base release. The BEA WebLogic module is one of these, but before you can delve into the array of tools provided to monitor WebLogic, you will need to gain familiarity with the main AppManager environment, and with the core user interface – the Operator Console.

Installation

Installing the AppManager Suite is a complicated process but has an excellent, user-friendly, and well-designed installation program that takes most of the pain out of it. Figure 1 shows a screen capture of the first stage.

The setup application gives you plenty of options as to which components to install. For the purposes of this review I installed everything, but there are far too many options to go into detail here on all of them.

SQL Server 2000 is required, and you will be asked to provide the passwords as part of the installation.

Running through this process will install the base AppManager Suite, which includes tools to monitor and manage a vast array of applications from IIS to Apache to Oracle to Lotus.

Operation

The AppManager Operator Console is the heart of the management system. It allows you to configure and control the execution of Knowledge Scripts (more on these later) on the systems and applications that you manage. From here you set up your management jobs, configurable to be regular or once-off, and manage the data that is generated by running these jobs.Once it is installed you are presented with a plethora of applications, but the core of the system is the Operator Console, shown in Figure 2.

This console can be broken down into the toolbar at the top, the treeview pane along the center left, the Knowledge Script pane along the right, and the events and status list along the bottom.

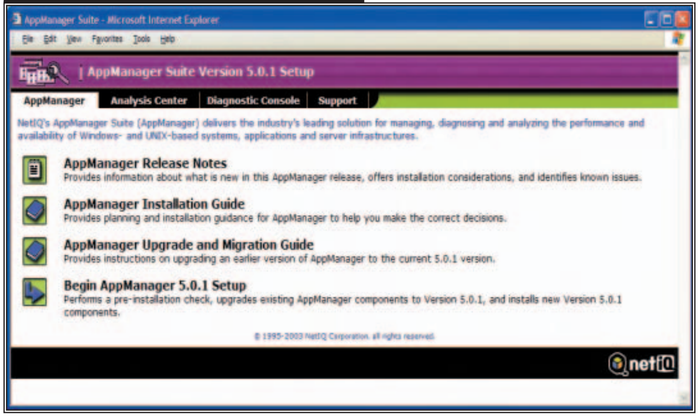
Knowledge Scripts are the workhorses of the application. They are prepackaged monitoring functions that NetIQ uses to maintain your system. One function of these scripts is to tie AppManager to the computers and applications that you want to manage using the discovery process. Another is to manage the systems and applications themselves. Once resources are discovered, the process of managing the system is as simple as dragging the Script from the

CORPORATE INFORMATION

NetIQ
3553 North First St.
San Jose, California 95134
Phone: 1-408-856-3000
Fax: 1-408-273-0578
Sales: 1 888-323-6768
E-mail: info@netiq.com
Web: www.netiq.com

Product:
www.netiq.com/products/am/default.asp

FIGURE 1


AppManager Suite - Microsoft Internet Explorer

File Edit View Favorites Tools Help

| AppManager Suite Version 5.0.1 Setup

AppManager | Analysis Center | Diagnostic Console | Support

NetIQ's AppManager Suite (AppManager) delivers the industry's leading solution for managing, diagnosing and analyzing the performance and availability of Windows- and UNIX-based systems, applications and server infrastructures.

AppManager Release Notes
Provides information about what is new in this AppManager release, offers installation considerations, and identifies known issues.

AppManager Installation Guide
Provides planning and installation guidance for AppManager to help you make the correct decisions.

AppManager Upgrade and Migration Guide
Provides instructions on upgrading an earlier version of AppManager to the current 5.0.1 version.

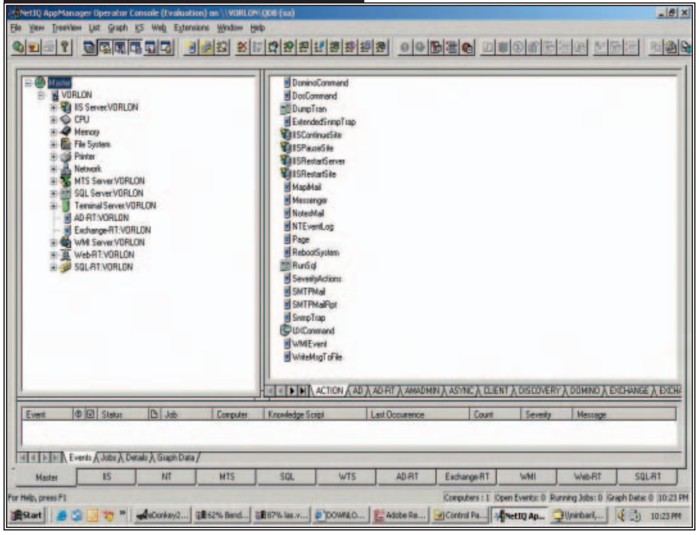
Begin AppManager 5.0.1 Setup
Performs a pre-installation check, upgrades existing AppManager components to Version 5.0.1, and installs new Version 5.0.1 components.

© 1995-2003 NetIQ Corporation. all rights reserved.

netIQ

Installation Dialog

FIGURE 2



NetIQ AppManager Console

Knowledge Script pane and dropping it on the correspon-ding system or application in the treeview pane.

The list pane at the bottom of the screen is a familiar-looking area where you can check on the status of the jobs that you are running. A job starts when a Knowledge Script is applied to an entity. For example, if you want to stress test your Web server, you would drag the appropriate knowledge script and drop it on the correct icon for your Web server. It is as simple as that.

As tasks run over time, they form a data stream. These will appear on the Graph Data tab on the list pane. You can then drag these onto the Graph Pane (accessible from the View menu) to plot a pretty 3D picture of the appropriate statistics. The visualization tool is very powerful, and very important – as it helps you tweak trouble areas in your system by getting a good look at where and when problems occurred. It's a lot easier than sifting through log files!

Managing BEA WebLogic with NetIQ AppManager

NetIQ AppManager supports all versions of BEA WebLogic after 6.0. The add-on consists of a suite of Knowledge Scripts tailored for managing BEA WebLogic servers.

The suite is incredibly comphrehensive, containing more than 70 Knowledge Scripts tailored specifically for the WebLogic application server.

One thing to note is that AppManager is not a log analyzer tool – it collects the information for you in real time via an agent that is installed on the target server. A nice advantage of this is that you don't have to stop the application server to get a snapshot of where you are performance-wise at any time, nor do you have to worry about losing data as log files expunge older data.

AppManager is designed to manage the server itself (i.e., Apache or IIS). It is not a testing tool for specific applications but a management and monitoring tool for the hardware and the services that run on the hardware. For example, in the case of IIS there exist Knowledge Scripts to manage things like count the ASPRequests or the number of FTPBytes that IIS is handling

BEA WebLogic is a J2EE application server, so it supports many core functionalities described in the J2EE specification. As a result, the suite of Knowledge Scripts provided cater towards J2EE functionality, but not solely so.

For example, BEA WebLogic 8.1 introduced a new Java VM – JRockit – which BEA claims outperforms other virtual machines. Knowledge Scripts written specifically for JRockit allow you to put this claim to the test!

Other entities that have knowledge Scripts available are all forms of EJBs, JDBCConnections, JMS, JTA, transactions, and, of course, the health and status of the app server. Scripts and jobs need not be tied to a single app server and can be spread across clusters.

The Verdict

NetIQ has a great product on their hands. The extensibility aspect makes it a great solution for enterprises with a multitude of different platforms, meaning they don't have to spread their thin budgets even thinner on different performance tools for each platform. In addition, you can create new Knowledge Scripts or modify the existing ones to optimize for your needs.

On the weak side, while their WebLogic tools are very powerful, they are not as user friendly to install as the rest of the application. The instructions with the download tell you that you need to install a patch prior to installing the main WebLogic pack, but the patch isn't part of the download and you have to try your best to find it on NetIQ's Web site. If you aren't a customer, and are simply using the 30-day trial, then most of the online support isn't available to you and sometimes finding the patch or a solution to your problems can be difficult.

If application and platform management are high on your to-do list, you could do a lot worse than to download the 30-day preview and give it a test-drive.

AUTHOR BIO

Laurence Moroney is a senior architect with a major financial services group in New York City. He has over 10 years' experience in multiple fields and disciplines, having developed applications for such diverse environments as casinos and airports.

CONTACT...

lmoroney@philotic.com

Four Legs Good, Two Legs Bad?

STICKING TO THE BASIS IS NOT A BAD IDEA

BY PETER HOLDITCH

The buzz in the industry these days is all about service-oriented architecture. One of the key benefits that this brings is loose coupling between systems, which in turn improves the agility of the overall architecture – if systems are unaware of each other's implementation details, then as things change over time the function of the IT infrastructure as a whole can be preserved without constantly having to revisit and tweak hundreds of touch points between systems, which has been a bugbear of composite systems in the past.

I have started to hear of people who, having heard the industry mantra, are starting to say (and worse, write reports saying) that for this reason, any distributed transaction is therefore necessarily a bad thing, since an atomic transaction is inherently a tightly coupled thing. After spitting some choice words in the general direction of these people, I thought of George Orwell's book *Animal Farm* – hence the title of the article – since that comparison made something closer to a printable article than the expletives, from which I'll spare you.

For those that haven't read *Animal Farm*, the plot is that the animals on a farm decide that they want self-rule and set about overthrowing the farmer and setting up a collective. Their rallying cry for this insurrection is "Four legs good, two legs bad" and duly united, the animals take over the running of the farm.

All very interesting, I hear the yawn go up, but what on earth is he rambling on about now? Well, like all mantras that get off the starting grid there is a lot of fundamental truth in the loosely coupled, coarse-grained school of design. In fact, it's not really anything new. Ever since mainframe systems were built with the screen-handling logic and the business logic in distinct modules, this has been the kind of thing considered "best practice." In fact, classical two-tier client/server systems have arguably failed to deliver the durability of the mainframe applications primarily because their construction, which usually simply involves taking database rows and plunking them onto a user interface canvas, actively discourages the coarse-grained, loosely coupled mode of design. Many, or even most, two-tier systems built in the '80s are now creaking badly at the seams, or have simply been decommissioned.

That's all well and good, but to take the leap from that to the position that absolutely anything that isn't loosely coupled is bad is foolish at best. You don't have to simply take my word for it; if this was the case, then why would the Web services pioneers (BEA among them) have taken the trouble to define the WS-Transaction specification, and moreover given it the power to propagate tightly coupled, two-phase commit transactions between multiple Web service endpoints? And even more compelling perhaps (depending on your background!), why is BEA's non-Java application server, Tuxedo – which arguably provides the original service-oriented architecture for open systems – often considered to be synonymous with xa, transactions, and two-phase commit?

The answer of course, as always, is that there are no absolutes when it comes to software design – there are many best practices that should be considered, and many design patterns to choose from. Many of these will conflict with each other. This does not mean that some are wrong and others right; it simply shows that there is no such thing as a perfect solution to a problem. But there is a set of solutions (imagine them plotted as points on some pair of axes) to the problem and some subset of these solutions will be in close proximity to the defined problem and constraints (also plotted in this same, very hard to define scale for axis).

It is this reality that yields the observation that building software is easy, but designing software well is hard. The skill is not in writing the lines of code that implement the solution (especially if the code is in a language like Java that does much to protect the innocent from a set of problems that commonly beset lower-level languages such as memory management) but in constructing an architecture from within which these lines of code can deliver against the business requirements, and allow flexibility in the face of these requirements changing over time (as they doubtless will). So, to the point: transactions are very much an architecture-level consideration (or they should be); they are a very useful tool in the architectural arsenal but a lethal weapon in the hands of someone charged with a detailed unit level implementation task.

–continue on page 42

AUTHOR BIO

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

CONTACT...

peter.holditch@bea.com

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

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

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

SYS-CON MEDIA

MX developer's journal

Performance Improvement in a J2EE Application

KILL THE BUGS THAT SAP TIME AND PRODUCTIVITY

BY **GVB SUBRAHMANYAM & SHANKAR ITCHAPURAPU**

AUTHOR BIOS...

GVB Subrahmanyam and Shankar Itchapurapu are consultants with CitiGroup Technologies. Dr. Subrahmanyam holds an M.Tech and a Ph.D. from IIT Kharagpur and MS software Systems from BITS Pilani, India. Mr. Itchapurapu holds a master's degree in computer applications.

CONTACT...

Subrahmanyam.vb.gampa@citigroup.com

Java is hot. Just nine years old, it has become one of the leading development environments in the world. Millions of programmers and thousands of companies use it, and half of all IT managers expect to deploy J2EE applications this year.

But Java's popularity hasn't necessarily made it easy for the growing population of Java code jockeys. Ever-shortening production cycles have kept the heat on programmers, who increasingly work in large teams to meet production milestones. And every day those teams come face-to-face with an immutable law of software development: the more code you write, the more bugs you'll get – bugs that cost time and sap quality and performance from applications.

This article looks at performance tuning and optimization of memory usage of a J2EE application. Our setup uses the BEA WebLogic Application Server. We will consider the following:

- The problem domain
- Tuning the Java Virtual Machine
- HTTP session management
- Tuning the application server
- Coding standards: laying ground rules for the future

The Problem Domain

We have a J2EE application with the following setup:

- BEA WebLogic 6.1 Service Pack 5 as application/Web server.
- Some popular RDBMS. This has no effect on our discussion.
- Model I Web Architecture.
- Eight stateless EJBs and six stateful EJBs.
- HTTP session holding references to stateful EJBs.
- Database Connection Pool with initial size 2 and maximum size 10.
- Around 120 servlets in the Web tier.
- XML/XSLT–based architecture.

The Problems

There was a memory issue with the application. When the server was started, the memory usage was around 7–8% of the total physical memory available. As the days went by and the application went into wider use, the memory usage grew to nearly 49–53% (over a 7–10 day period).

If the users logged off from their session by clicking the "Log off" button in the left-hand side menu, the application removed all the stateful beans from the server, but if a user just closed the browser window it didn't remove those beans and stayed in the container until the application server was restarted. As this continued, the number of EJB instances in memory shot up to 400 and above.

When BEA WebLogic Server loads more than ~400 EJBs, the Hotspot VM throws an OutOfMemory Exception. This occurs even though there appears to be more memory available.

Tuning the Java Virtual Machine

The Hotspot Virtual Machine was throwing an OutOfMemory Exception while trying to allocate PermGeneration space. The Hotspot VM uses different sections of memory. The permanent generation section is used for storing classes, methods, and symbols used by running Java objects. The initial size of the permanent generation section is 1 MB and the maximum size was 32 MB prior to 1.3.1 and 64 MB.

To get around this, we can set up the permGeneration space through a JVM switch with the following command line:

```
java -server -XX:MaxPermSize=128M.
```

Note that increasing the max perm size only delays a failure. It's up to your application to properly clean the unused objects. Also, the **XX** options are not supported across all JVMs.

HTTP Session Management

When the user closes the browser without logging off from the session, the EJBs in the user's session will not be garbage collected. This is the primary reason for having too many EJBs in memory. To get around this, the HTTP session management has to look into all possible combinations. We can set up a default session timeout period in web.xml (Web application deployment descriptor) as follows:

```
<session-config>
        <session-timeout>x</session-timeout>
<session-config>
```

With this setting, the user's session will be automatically deactivated after **x** minutes of inactivity.

The other way is to code the session management using the following code when creating HTTP session

```
HttpSession session=new HttpSession ();
session.setmaxinactiveinternal(int
timeoutSeconds);
```

This code will invalidate the user's session of timeoutSeconds of inactivity.

Note: If you do both of these steps, the value in the servlet code will override the value set up in the web.xml.

The only difference between these two methods is that the second one takes seconds as the parameter while the <session-timeout> tag value takes minutes as the argument.

Normally, when the session is invalidated the logoff servlet/JSP will have code to remove references to all objects/object graphs referenced by the particular session. But when the user just closes the browser button there is no way our logoff servlet/JSP will be called. In this scenario, even though the session is invalidated, the enclosed objects/object graphs will still be there. When the garbage collector tries to garbage collect this session, it will also have to garbage collect all of these enclosed objects. When we have large objects (objects with larger references/data), we can also use the HTTPSessionListener interface to perform some clean-up action.

javax.servlet.http.HTTPSessionListener Interface

This interface declares the following two call-back methods:

```
Public void sessionCreated(HttpSessionEvent
event);
Public void sessionDestroyed(HttpSessionEven
event);
```

These methods are called before a session is created/destroyed.

We can have a listener class that implements this interface and use these callback methods to control how sessions are created/destroyed. We need to register our listener class in the web.xml as follows:

```
<listener>
        <listener-
class>MySessionListener</listener-class>
</listener>
```

The advantage of using listener class along with the session timeout parameter in the web.xml file is that we will have more control of session management. If the session has large objects, then before the garbage collecter cleans up the objects, its time slice may disappear. In this case, it needs to wait for the next slice to clean up the objects.

Note: It is important that we design the application so that there is a single entry point. We need to start a new HTTP session in this class. All the remaining pages should check for the existence of HTTP session and for an error page when the session is null (Session Expired). This enables centralized control of HTTP sessions.

Tuning the Application Server

BEA WebLogic offers a number of parameters that we can set to optimize the bean pool size, including setting the initial size of the stateless bean pool. Some of the features include:

- ***Use the <initial-bean-pool-size> to set it to minimum possible value for stateless session beans:*** By default, the value is 1000. As stateless session beans can be shared between concurrent users, it is better to set this value to a very minimum. The correct value for this tag depends upon the concurrent users and peak load situation of the application.

- ***Use the <max-beans-in-free-pool> tag to set the maximum bean pool size for the stateful session beans:*** The default value for this has no limit. The value set for this tag greatly affects the activation/passivation mechanism of BEA WebLogic Server. The appropriate value for this tag depends on the application's traffic. When the number of beans in this pool has reached the threshold and a request for a new bean instance has come, the WebLogic Server will pick one or more beans for this pool for passivation. The algorithm for picking up the bean instance for passivation is either LRU (Least Recently Used) or NRU (Not Recently Used).

  If max-beans-in-cache is reached and EJBs in the cache are not being used, WebLogic Server passivates some of those beans. This occurs even if the unused beans have not reached their idle-timeout-seconds limit. If max-beans-in-cache is reached and all EJBs in the cache are being used by clients, WebLogic Server throws a CacheFullException.

- ***Use the <idle-timeout-seconds> tag to set the time the WebLogic Server will wait before passivating an idle stateful session bean instance:*** The WebLogic Server will wait for the same amount of time before removing the bean from the swap space. Care should be taken in setting this value because once the passivated instance is removed from the swap space, there is no way to retrieve the state of the bean again.

For example, consider the following setting:

"Every day those teams come face-to-face with an immutable law of software development: the more code you write, the more bugs you'll get"

```
<idle-timeout-seconds>1200</idle-timeout-sec-
onds>
```

The idle bean instances will be passivated after 20 minutes of inactivity. After another 20 minutes, the bean instance will be removed from the disk also. Now let's say at the 41st minute the user has called a method on the bean instance. The BEA WebLogic Server will throw the error shown in Listing 1.

WebLogic 6.1 Service Pack 5 provides a useful tag to get around this. The tag is described as follows:

<!-- The stateful session beans that are passivated to the disk will stay alive for this many seconds. After this interval, the passivated beans will be removed from the disk.

Used in: stateful-session-cache

Since: WebLogic Server 6.1 sp5

Default value: 600
-->
<! ELEMENT session-timeout-seconds (#PCDATA)>

If we set the value for this <session-timeout-seconds> we can control the time the passivated beans will be removed from the disk. We can use this tag and set it to a very value so that we will always have our stateful EJBs (either in memory or on the disk). This will completely eliminate the bean deleted error.

Using WebLogic Managed Servers

BEA WebLogic allows you to create one or more servers within a single domain. One server will be the admin server; all other servers will be managed servers in the sense that they are managed by the admin server. An application, when put into production, should not be deployed in the admin server. The advantage of using managed servers is that we can start and stop them from the admin console. Thus even if the server holding the application stops responding (for any reason), we still have a chance to restart the server from the admin console.

Coding Standards: Laying Down the Rules for the Future

Managing a production system is even more difficult when the environment in which the system will run has not been taken into account in the design phase. System re-engineering is a simple task when careful consideration of the environment, boundaries, and context of the application is taken care of during the design phase. Design is an abstract definition of execution path. Solutions will be more scalable when the development is done by taking into account every minute detail of the design. There will be no hard and fast rules for system development as this depends on the particular problem domain in hand. But, there are some axioms that always help.

- ***Have a clear idea about supplementary classes like String and StringBuffer:*** Use the appropriate class in appropriate situations. For example, using String class to build a long SQL query is inefficient and overloads the JVM string pool.
- ***Clean up legacy objects like Hashtable as soon as you're done with them***.
- ***Call remove() method on EJB references explicitly:*** This will release the bean instance to the bean pool and reduce the number of bean instances created by the container.
- ***Manage database connections carefully:*** Have a close() method called on them as soon as you're done with it. Don't open the connection as a first line in the class. Open the connection only when you need it.
- ***Understand the business requirements*** and the context in which your code executes before you write a single line of code.

To sum up… "Create Objects as late as possible and remove them as early as possible".

Listing 1

```
: Bean has been deleted.
    at
weblogic.ejb20.swap.DiskSwap.read(DiskSwap.java
:156) at
weblogic.ejb20.manager.StatefulSessionManager.g
etBean(StatefulSession Manager.java:242) at
weblogic.ejb20.manager.StatefulSessionManager.p
reInvoke(StatefulSessionManager.java:313)
    at
weblogic.ejb20.internal.BaseEJBLocalObject.preI
nvoke(BaseEJBLocalObject.java:113)
    at
weblogic.ejb20.internal.StatefulEJBLocalObject.
preInvoke(StatefulEJBLocalObject.java:126)
```

Four Legs Good, Two Legs Bad?

–continued from page 38

All this (or something very similar) has led to the emergence of frameworks as the key element of reusability. In general, frameworks are bought and sold readily (application servers themselves are, after all, just runtime frameworks) but a market for business componentry is harder to detect. Frameworks encapsulate best practices in areas such as transactional behavior whilst not rigidly imposing them, in case (as is bound to happen from time to time) the default assumptions that a framework must necessarily make do not fit some corner use case.

In BEA terms, the new idea today is the BEA WebLogic Workshop runtime framework. This is the key piece of reusable value – it is positioned at a high-enough level that developers are insulated from the nitty-gritty of considerations such as transactions and empowered to write their unit-level Java code in comparative safety. The framework exhibits some sensible transactional behaviors for the common case – each Web service invocation executes in a single transaction, a process flow executes in a single transaction unless it has to wait for an external event, and so on. The really exciting potential is for the emergence of a market in business components – useful units of business functionality whose use can be orchestrated readily because all of them are relying on the same set of core assumptions in the framework, and yes… sometimes the interactions between the components will be transactional, and sometimes not.

So, back to *Animal Farm*. In the end the pigs assume responsibility for the farm over the mass of animals, eventually learning to walk on two legs and rewriting the slogan as "four legs good, two legs better." I'll leave you to draw your own conclusions about how that ironic twist applies to transactions, frameworks, and software design, while noting that if you slice the whole technology stack through from low-level things like a transaction manager itself embedded in the core of the application server, atop which sits the WebLogic Workshop framework, above which are layered the higher-level presentation capabilities of the portal, the process orchestration capabilities of the Integration suite, and the data aggregation capabilities of Liquid Data, you are free to choose the appropriate number of legs to suit any occasion.

More next month as soon as I can undo these buckles behind my back…

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com	978-816-1600	23
BEA Systems	http://dev2dev.bea.com/medulla	800-817-4BEA	2,3
Confluent Software	http://www.confluentsoftware.com	408-523-9000	11
Cyanea	www.cyanea.com/wldj/underpar.html	877-CYANEA8	29
FileNet	www.filenet.com	512-434-5935	15
Java Developer's Journal	www.javadevelopersjournal.com	888-303-5282	29
Mercury Interactive	www.mercuryinteractive.com/optimizej2EE	800-831-8911	52
MX Developer's Journal	www.sys-con.com/mx/subscription.cfm	888-303-5282	39
NetIQ	www.netiq.com/solutions/web	408-856-3000	9
PowerBuilder Developer's Journal	www.sys-con.com/pbdj	800-303-5282	35
Quest Software	http://java.quest.com/jprobe/wldj	800-663-4723	21
Quest Software	http://java.quest.com/jcsv/wldj	800-663-4723	51
RedHat	www.redhat.com	888-2REDHAT x45011	25
SYS-CON Media Database	www.edithroman.com,	800-223-2194,	49
SYS-CON Media Database	www.epostdirect.com	800-409-4443	49
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	31
SYS-CON Reprints	www.sys-con.com	201-802-3026	45
Web Services Edge East 2004	www.sys-con.com	201-802-3069	33
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	43
Wily Technology	www.wilytech.com	888-GET-WILY	5

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

LOOK WHAT'S COMING NEXT MONTH

EMALL: Building an Integration Application

A secure, mission-critical application was built in no time to allow hundreds of vendors to compete for federal business. It integrated with systems that are geographically separated and based on a diverse range of distributed architectures and data communication technologies.

State Machines and Workflow

Process-oriented enterprise applications abound, and workflow toolsets, such as WebLogic Integration's BPM, are viable and popular frameworks to develop them. However, the state machine approach is also valid, and has particular advantages when coupled with BPM.

Do You Know What You Run?

Knowing exactly what you run will allow you to speed up that inescapable initial exchange of information with your BEA support engineer and allow them to get past configuration and into the technical depth of the actual issue.

Admin Automation with wlshell

A look at how to write wlshell scripts to automate administrative tasks for configuration and monitoring. This approach provides clear advantages over other methods, improving productivity and legibility, and reducing the chances of committing configuration mistakes.

Configuring WebLogic 8.1 JDBC Connectivity

Following the procedure in this tutorial a WebLogic developer should be able to configure a WebLogic 8.1 server with an Oracle 8.1 database. The general concepts explained here will be easily transferable to other relational databases

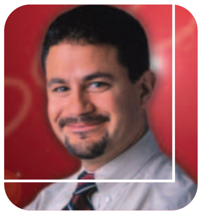
BEA WebLogic DEVELOPER'S JOURNAL

Automation is coming to a data center near you. It promises to cut costs, speed up deployment, ease problem diagnostics, and protect your applications against man-made and natural disasters.

Data Center Automation

GRID FOR THE ENTERPRISE

BY BENJAMIN RENAUD



AUTHOR BIO

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

CONTACT...

br@bea.com

Data center automation is hot. This year alone Veritas bought Jareva, IBM bought ThinkDynamics, and Sun nabbed first infrastructure pioneer TerraSpring, then CenterRun. Oracle has made Grid the central marketing theme for Oracle 10*i*, promptly renamed Oracle 10*g*. HP has announced a comprehensive strategy, dubbed the Adaptive Enterprise, which promises to make the job of deploying applications, bringing servers up and down, and tracking data center configurations considerably easier. OpsWare, a nimble startup in the space has been working to productize its powerful ASP management software and make it available to you.

But why talk about data-center automation here? Isn't that outside the scope of WebLogic and the Web platform? The answer is that the definition of an application is not as clear-cut as we like to think. You can think of it to go from the Web server to the database. Or you can think of it to go from the user's browser all the way back to the legacy system, with the vast Internet, a few routers, a load balancer or two, a Web server, and a couple of application servers in the middle. In other words, you can easily include the entire data center in the conceptualization of the application (or applications). With so much activity around data-center automation, we thought we'd stop and consider the implications for your applications and for the Web platform.

The first step to data-center automation is inventory. What exactly is in my data center? For many IT managers that can be a daunting task: thousands of machines, running dozens of operating systems, with heterogeneous configurations. They have different clock speeds, RAM, disk capacity, number of network interfaces, or connectivity to the network. These servers have different support hardware, such as UPSs, cable configuration, etc. They have OS versions, patch levels, and installed utilities, not to mention applications. Creating an inventory of everything in the data center is a Herculean task. And since different groups often have access to different machines, and can remotely change the software or quietly upgrade or shift around resources in the data center, keeping track of it all is beyond the ability of even mythical Greek gods, let alone most IT managers.

To tackle this problem vendors are taking different tacks. Most have a model-driven approach, which invariably uses XML to describe the data center. HP's Utility Data Center (UDC) product uses FML to describe everything in the data center. Motive, a leader in service management, has comprehensive modeling capabilities. Most interestingly, OpsWare and its chairman, Marc Andreessen of Netscape fame, just announced an effort to standardize the language used to describe all the elements in the data center: DCML, or Data Center Markup Language. This is an ambitious effort: in the words of Andreessen himself, DCML "will do for the data center what TCP/IP did for networks and HTML did for Web content." Trying to rally everyone behind a standard won't be easy however. Some of the big system players like IBM and Sun envision end-to-data centers built exclusively on their technology, which wouldn't require a standard – standards are for interoperation, after all. But for its debut DCML has signed up some 25 companies, including heavyweights like BEA, CA, EDS, and Mercury Interactive.

Having a language to describe all that is in your data center is only the first step, however. Then comes the actual inventory. Some players like HP expect the inventory to be taken once, and to change little from that point on. Others, like Veritas, put a heavy emphasis on automated discovery, sparing the IT staff some heavy lifting and reducing the likelihood of errors. The manager for a relatively large IT center who was using an automated resource discovery tool confided: "What we didn't know about our infrastructure was staggering. We found out so much." Understandably he wished to remain anonymous.

With a robust model of the infrastructure many things become possible. One is configuration management and change tracking. How many times have you heard "it was working last week.

Something must have changed." Yes, but what? Configuration management tools can help answer that question and pinpoint problems more quickly. They keep track of patches, upgrades, changes in network configurations, and other changes.

Another thing that becomes possible, at least with some of these solutions, is to say: "I want to bring up 10 blades, with a minimum clockspeed of 2GHz, a minimum RAM of 2GB, and a minimum of 2 network interfaces. I want to load RedHat Enterprise Advanced Server at the latest patch level, load the WebLogic platform, deploy my WebLogic application. And don't forget my OpenView monitoring module." Veritas' OpForce product can do exactly that, taking a bare-metal box (a Dell blade or a high-end Sun server) and provisioning it into a fully functional BEA WebLogic Platform server. HP's UDC and OpsWare's software can do the same. While they can employ different techniques to get to the same point, from disk blasts to dynamic software install, the end result is the same. The remote management and configuration isn't limited to servers or the WebLogic Platform. OpForce, for example, can configure load balancers and databases, while UDC manages UPS units attached to servers.

Once you can remotely and programatically bring servers up and down, a new world of possibilities awaits you. One is on-demand provisioning: instead of provisioning an application for its peak possible load, allocate just the right amount of computing power for that application at any given time. A classic example would be a bank that runs a promotional campaign for one of its product lines, say mortgage refinancing. The resulting rise in usage for the application that processes these requests could be handled by going to a free pool of servers that can be dynamically provisioned to handle the additional load. If later the company decides to launch a free-checking ad campaign, the same servers can easily be retargeted to handle that.

And this can be completely automated. Given the speed at which information travels and the very nature of the Internet, load peaks are unpredictable. And while load can go up as the result of a well-planned ad campaign, it can also rise in response to a suddenly popular blog or an unscheduled mention on a television program. It can result from a natural or man-made disaster – any unexpected rise or fall in demand. Response time becomes critical. We can imagine monitoring systems detecting increasing log and immediately reacting, shifting resources where they are needed dynamically and without human intervention.

This is the vision painted by many of the players in this space, and the demos they give are impressive. As with any cool new technology though, it will take some time to mature. Many of these solutions are still too slow to respond effectively to unexpected changes in load. The languages used to describe resources, including DCML, still need to be expanded to many devices, and the software to manage these devices has yet to be written.

But the promise, and the need, is real. And we can dream of even cooler applications for the future. Imagine that you are writing an application in BEA WebLogic Workshop. You have some expectations about where, and how, it's going to be deployed. You have a pretty good idea of how many tiers, how much RAM, and what kind of database back end will be needed. Today the best you can do is to document this somewhere, and hope that 1) it won't be lost and 2) the deployment team will read and understand it. What if, while you wrote your application, you could capture these requirements formally, in a language like DCML, and package them with the application. At deployment time a utility application could match available resources with the required model from your application, and come up with a suggested configuration for the BEA WebLogic Platform and any supporting components like a database or load balancer. The utility framework could also flag any discrepancies between what it needs and where it is deployed.

Until then, there will be many developments in the utility computing space. Prices have already begun to fall from dizzying heights, and they will fall further. A standard will emerge, either DCML or another. One thing that will not change is the heterogeneity of data centers and the dislike of end-users for proprietary approaches. The transition to a utility model will be progressive, starting with application automation and progressively spreading to the entire data center.

WebLogic Workshop 8.1 IDE Fundamentals

SPEED UP YOUR DEVELOPMENT AND TESTING

BY **ANBARASU KRISHNASWAMY**

As a software architect and developer, I've used a number of IDEs for my J2EE development. I have my priority list of features that I look for in all the IDEs but I wasn't able to find one that gave me everything I wanted.

If an IDE satisfied some of my needs, it completely ignored my other needs. The BEA WebLogic Workshop 8.1 IDE, however, delivers most of the features I've been looking for. Now I'm a happy WebLogic Workshop user and I want to share some of the best features with you.

AUTHOR BIO

Anbarasu Krishnaswamy is a senior principal consultant with BEA professional services. He has about 10 years of experience with BEA products and is a Sun-certified Java programmer and BEA-certified WebLogic Server developer. He holds a master's degree in computer science and engineering and a bachelor's degree in electrical and electronics engineering.

CONTACT...

anbarasu@bea.com

IDE Features

One of the strengths of BEA WebLogic Workshop is its ability to do pretty much all the development using this tool. It is a one-stop shop for J2EE development. The following list gives you an idea of what you can do with it. Please note that this is not the complete list:

- Java source
- Web services
- JSPs
- HTML
- CSS
- EJBs
- Page flows
- Java process definitions
- Java controls
- Schemas and XMLBeans
- Ant integration
- Configurable external tools
- IDE debugging
- Test form capability
- Integration with BEA WebLogic server 8.1

In this article I'll focus on the basic IDE features and HTML/JSP editing and deployment capabilities of BEA WebLogic Workshop 8.1.

The Editor

You create and build an *application* in the WebLogic Workshop editor. The Workshop IDE operates on one application at a time. The application tab presents a tree view of the application and the files tab presents an alphabetical, filtered list of all files in the application.

The Edit Pane displays either a source view or a design view appropriate for the file being edited. When you select an entity in the Edit Pane, the entity's properties are displayed in the Property Editor. The Property Editor displays the properties for the currently selected entity in the Edit Pane.

The Document Structure Pane displays a structured view of the methods and fields in the class or interface being edited.

The Output Pane displays various forms of information that are generated when you run or debug the application.

The Server Status area indicates whether the associated instance of BEA WebLogic Server is currently running.

The Status Bar displays messages indicating the state of the IDE.

The Data Palette displays a structural list view of the methods or attributes of controls and tags. As you drag items from the Data Palette to the Edit Pane, code is inserted in the current entity to access the item you dragged. The palette contains often-used items you may wish to add to the entity in the Edit Pane. You may drag an item from the palette to the Edit Pane to create a new item of that type.

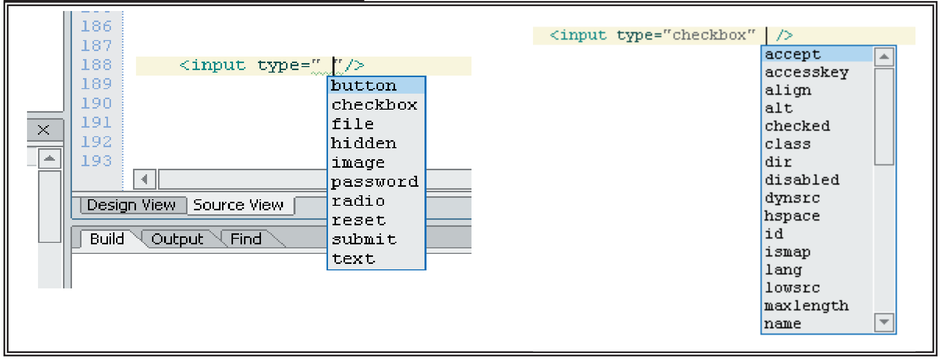
Applications

An application in BEA WebLogic Workshop is the collection of all resources and components that are deployed as a unit to an instance of BEA WebLogic Server. It is also the top-level unit of work that you manipulate with the WebLogic Workshop IDE. In the IDE, you may have at most one application open at a time. A WebLogic Workshop application is a J2EE Enterprise Application and ultimately produces a J2EE Enterprise Application Archive (EAR) file. An application may contain projects, libraries, and modules.

Projects

Projects are a project group's related files that comprise a component of an application. WebLogic Workshop supports a number of project types, including Web projects, Web services

FIGURE 1



```
<input type="" />
button
checkbox
file
hidden
image
password
radio
reset
submit
text
```

```
<input type="checkbox" />
accept
accesskey
align
alt
checked
class
dir
disabled
dynsrc
hspace
id
ismap
lang
lowsrc
maxlength
name
```

Code completion for HTML

projects, Control projects, EJB projects, Java projects, Schema projects, portal projects, and integration projects. You may also create custom project templates to meet the needs of your organization.

File Types

BEA WebLogic Workshop 8.1 supports a number of file types, including EJB, JCS (Java Control Source), JCX (Java Control Extensions), JSP, JWS (Java Web Service), JPD (Java Process Definition), JPF (Java Page Flow), WSDL, DTF (Data Transformation Format), and a number of portal file types. The WebLogic Workshop IDE adapts itself in several ways to the type of file being edited.

Libraries and Modules

In addition to projects, each BEA WebLogic Workshop application also contains two folders named Libraries and Modules. These folders can contain compiled Java code that you want to be available to the components of the application. The products of the various project types are automatically placed in the Libraries or Modules folder as appropriate. For example, the JAR file containing the XMLBeans Java classes derived from the XML Schemas in a Schema project is automatically placed in the Libraries folder. However, you may also copy additional JAR files directly into the Libraries or Modules folders to make them available to the rest of your application.

Testing and Debugging

You can use the BEA WebLogic Workshop integrated debugger to debug your application. The debugger allows you to set break points, step through your code line-by-line, view local variables, set watches on variables, and view the call stack and exception information. If you're developing against a remote server, you can also debug against that remote server. Break points and other debugging information will be stored on the local machine.

In order to debug an application, you must have a way to exercise the application as a real client would. WebLogic Workshop includes a test browser in which you may test Web services and Web applications. When you run a Web service, the test browser automatically loads Test View, a tool for exercising Web services.

HTML/JSP Editor

In the Workshop JSP/HTML editor you can create a Web project or import an existing Web application and create JSP files in that project. Web projects are typically used to expose enterprise application logic via a user interface. The user interface is constructed from Java Server Pages (JSPs), which are Web pages that can interact with server resources to produce dynamic content. BEA WebLogic Workshop defines Java Page Flows that define and contain the logic required to connect multiple JSPs. Web services are typically used to expose enterprise application logic to applications (as opposed to users). Individual Web service interfaces are published via Web Services Description Language (WSDL) files.

Each Web project or Web service project ultimately produces a J2EE Web Application Archive (WAR) file. Note that the name of a project becomes part of the public URL of each resource located within that project. You should choose your project names with that in mind.

BEA WebLogic Workshop offers the following features that all Web developers need in their IDE.

Editing Scriptlets

When you write Java code in a scriptlet, the editor shows all the public members and methods in the class. Any syntax errors are indicated using a red squiggle as you edit the file. The scriptlets also inherit the IDE settings for Java code, displayed using appropriate syntax coloring. Any errors are displayed as a little red or green line on the scroll bar and you can go to the specific error line by clicking on the scroll bar line.

Editing HTML

A palette window displays the HTML controls like Table, Image, and Hyperlink that can be dragged and dropped on the design view. The document structure window shows the layout of the document that makes it easy to debug and navigate to a certain section in the page. In the source view, WebLogic Workshop offers code completion for HTML tags. This works not only for the property names of the tags but also for property values (see Figure 1).

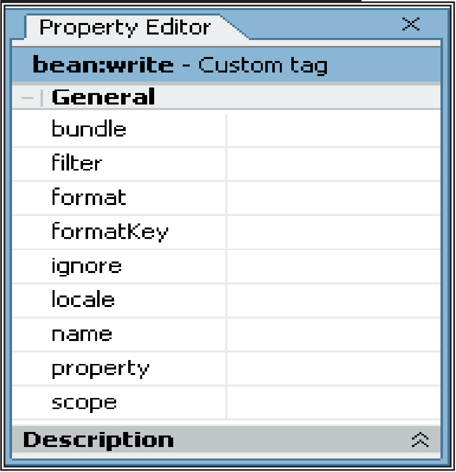
Using Tag Libraries

Any tag library configured in the web.xml is automatically recognized by the IDE and organizes all the tags in the palette window. This gives you the ability to drag and drop any tag into the design view or source view. WebLogic Workshop does a perfect job by placing the tag with any required properties. When you select a tag, the properties can also be edited conveniently using the property window. Figure 2 shows the Property Editor for the struts bean:write tag.

NetUI Tag Library

NetUI is a powerful BEA tag library that covers HTML, data binding, and templates. NetUI tags wrap struts tags, so they offer features like form beans, forwards, global forwards, etc. These tags are used extensively in page flow files.

FIGURE 2



Property Editor

bean:write - Custom tag

General

bundle	
filter	
format	
formatKey	
ignore	
locale	
name	
property	
scope	

Description

Property Editor

Syntax Coloring

BEA WebLogic Workshop 8.1 comes with a fully customizable editor. You can customize almost anything, including the comments, annotations, keywords, and scriptlets.

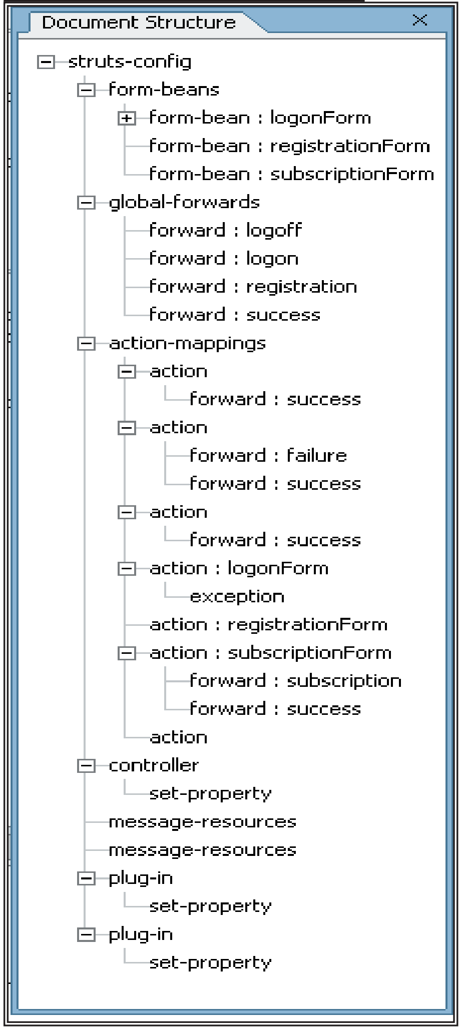
Document Structure

The document structure panel displays a tree view of the document for easy navigation. Document structure is also useful when editing XML files, for example an ant build.xml or Struts struts-config.xml file. It displays the tags with the names specified in a name parameter. Figure 3 shows the document structure panel for the struts-config.xml file in the Jakarta struts example.

Testing and Debugging

Any WebLogic Workshop–enabled Web application can be debugged using the IDE by setting break points. You can run and test any JSP from the editor using the test form feature.

FIGURE 3



```
Document Structure
⊟ struts-config
   ⊟ form-beans
      ⊞ form-bean : logonForm
        form-bean : registrationForm
        form-bean : subscriptionForm
   ⊟ global-forwards
        forward : logoff
        forward : logon
        forward : registration
        forward : success
   ⊟ action-mappings
      ⊟ action
           forward : success
      ⊟ action
           forward : failure
           forward : success
      ⊟ action
           forward : success
      ⊟ action : logonForm
           exception
        action : registrationForm
      ⊟ action : subscriptionForm
           forward : subscription
           forward : success
        action
   ⊟ controller
        set-property
     message-resources
     message-resources
   ⊟ plug-in
        set-property
   ⊟ plug-in
        set-property
```

Document structure panel for Struts configuration file

External Tools

WebLogic Workshop's external tools are a powerful feature to extend the IDE to suit your needs. Workshop provides substitution strings to customize the development environment. In a typical development environment, you develop the sources in a work area and copy them to the deployment area for unit testing. One of the challenges I faced before was manually copying the JSPs to the appropriate directory for testing. With WebLogic Workshop I can create an external tool task that can copy any file or a directory that I point at to the deployment directory through the substitution strings that give you access to the current file, directory, etc.

Following are some of the substitution strings available in WebLogic Workshop

- ***${file.path}:*** Full path of current file
- ***${file.name}:*** File name of current file
- ***${file.dir}:*** Directory of current file
- ***${project.dir}***: Root of current project
- ***${application.dir}:*** Root of the open application

To create a tool, go to Tools->IDE Properties->Tools and click "New Tool". Enter the tool name, command, and directory from which the command should be executed. The command may include the substitution strings listed above. For example your Copy JSP command could be:

```
xcopy  /s ${file.path} <Deployment
Directory>\webapp\jsp
```

You can also specify any script or batch file to execute. When I change any JSP, I run this external tool to copy just this JSP to the deployment directory.

If you want to see the output of the command in the IDE, check the "Capture Output" check box. If you want to see the output in a separate command window, check the "Use interactive shell" option.

Using Ant with WebLogic Workshop

BEA WebLogic Workshop projects can be built with the IDE build or using a customized ant build. You can export the IDE build to an ant build file by going to Tools ->Project Properties->Project->Build and clicking "Export to Ant file". You can also use this file to begin and customize your build process.

To use your own ant build process, go to Tools->Project Properties->Project->Build and select "Use Ant build". Click the "Browse" button in "Ant Settings" and choose the Ant build XML file. Then select the build target in the drop-down box. It lists all the available tasks in the build XML file.

Conclusion

BEA WebLogic Workshop 8.1 is a great IDE for developing J2EE applications and improves productivity by a great magnitude.

This article described the basics of the IDE and showed the features of the JSP/HTML editor. We also looked at how to use ant to build your application and how to create external tools to speed up development and testing. If you haven't used BEA WebLogic Workshop, I encourage you to look at it for help in developing your J2EE applications.

– continued from page 30

APPLICATION MANAGEMENT WITH WEBLOGIC SERVER FOR DEVELOPERS

Finally, we use a nested create tag inside the JMS server's create tag to create a JMS queue that is part of our newly created JMS server. This nesting structure is used to specify the MBean upon which the action is to be applied (in the case of set commands) or to specify the relationship between MBeans (in the case of the create commands).

For more information about the wlconfig Ant task, see http://edocs.bea.com/wls/docs81/admin_ref/ant_tasks.html.

Summary

This article provides an overview of JMX concepts and terminology and a discussion of the BEA WebLogic Server 8.1 JMX implementation. The article ended with a discussion of the WebLogic JMX client tools and how to use them to create, view, and modify the WebLogic Server MBeans. Our next article will dive into the Java APIs for building custom JMX programs. A later article will discuss creating custom MBeans and extending the Admin Console to display them.

SYS-CON MEDIA

304,187 of the World's Foremost IT Professionals

DIRECT MAIL, EMAIL OR MULTI-CHANNEL

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

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

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

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

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

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

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

COLDFUSION Developer's Journal
Windows Server 2003 and CFMX 6.1

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

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

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

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

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

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

e-POSTDIRECT® Your iMarketing Source

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

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

List Brokerage & Management
EDITH ROMAN
Data Processing • Email Marketing

News & Developments

Wily Technology announces Introscope 4.2

(Brisbane, CA) – Wily Technology, a leader in Enterprise Java application management, has released Introscope Version 4.2, the newest edition of Wily's solution for monitoring, managing, and improving enterprise Java applications.

Using patented, low-overhead technology, Introscope enables IT teams to monitor and manage live production applications and their operating environment 24x7. It allows them to isolate bottlenecks throughout the entire application, whether in production applications, application servers, or back-end systems, all the way down to individual servlets, JSPs, EJBs, classes, methods, and more.

Introscope 4.2 is part of Wily 4, Wily Technology's Java application management solution. Wily 4 facilitates interdepartmental communication by bridging the functional silos in IT organizations, allowing problems to be identified and solved quickly. www.wilytech.com

Accredited Home Lenders Selects BEA and Blue Titan

(San Jose, CA) – BEA Systems, Inc., the world's leading application infrastructure software company, has announced that Accredited Home Lenders has selected the BEA WebLogic Platform 8.1 and Blue Titan Network Director as the foundation for its information technology infrastructure and next-generation customer portal. The solution, implemented by WellFound Technology, is designed specifically for the mortgage industry and can allow Accredited Home Lenders to model and deliver a standards-based and adaptive application for mortgage processing.

Accredited Home Lenders will deploy a multichannel customer portal that models their mortgage process from origination through account servicing to help mortgage brokers as they sell non-prime residential mortgage loans. Using WebLogic Platform 8.1, Accredited Home Lenders will have the flexibility to access critical back=end systems and leverage existing systems and IT assets.

Blue Titan Network Director 2.0 will tie the solution together into a services-oriented architecture by delivering message-based security, reliability, and life-cycle management. The combination of BEA's and Blue Titan's technology will allow Accredited Home Lenders to build service-based applications that can enhance control, performance, and manageability. www.bea.com, www.bluetitan.com

Appian Announces Strategic Alliance with BEA Systems

(Vienna, VA) – Appian Corporation, a provider of real-time, enterprise Web solutions, today announced a strategic alliance with BEA Systems, Inc., to provide joint technology solutions to government and commercial customers.

Appian and BEA are working to help state and federal government customers, as well as commercial customers, better manage their information. The alliance will give Appian access to BEA WebLogic standards-based infrastructure software that Appian will use to build and optimize its application server–independent software for public sector and commercial projects. Appian and BEA will serve as joint marketing partners and solution providers to prospective and existing customers. www.appiancorp.com, www.bea.com

eLinear Wins Order for New Kila Security Solution

(Houston) – eLinear, Inc. has announced the sale of the kila security solution to an international company with $400 million in revenue.

ELinear's client required a solution that would work with Bea Weblogic's security framework and will provide a single sign-on from the user's desktop. This solution will enable the users to directly enter the application, once they are logged into the network on the workstation/desktop. Kila security solution automatically uses the security credentials from the desktop and authenticates and/or authorizes the user against the corporate Microsoft Active Directory and communicates with the BEA WebLogic security framework on a Unix environment. www.elinear.com

Salesforce.com Announces sforce Toolkit for Borland JBuilder X

(San Jose, CA) – Salesforce.com, a provider of software-as-service, has released the salesforce.com sforce Toolkit for the Borland JBuilder X development environment. Sforce is an on-demand application platform that allows enterprises and solution developers to customize, integrate, and extend salesforce.com to create CRM solutions that address their specific business needs. The sforce Toolkit, planned for shipment with Borland JBuilder X Enterprise and Borland Enterprise Studio 7 for Java, will assist developers with Borland Java solutions to rapidly build custom-tailored software-as-service CRM solutions with lower cost and less complexity than software-only alternatives.

The sforce toolkit for JBuilder X is included free of charge in every copy of JBuilder X Enterprise. www.sforce.com

BEA Releases Preview of Streaming API for Java (StAX)

(San Jose, CA) – BEA Systems, Inc., has announced the public availability of a preview release of the Streaming API for Java (StAX), a new Java API designed to improve developer productivity and performance by making it easier to incorporate XML into Java.

StAX is designed to overcome many of the disadvantages of former methods by providing the efficiency of streaming APIs and the control of tree-based APIs. This method represents a next generation of APIs – pull parsing. Unlike other event-based approaches, developers using StAX for XML document parsing can pull events, rather than handling events in callback, which can enable them to stop processing, skip ahead, or get subsections and, ultimately, help to gain precise control, thereby saving time and reducing overall development costs. http://dev2dev.bea.com/technologies/stax/index.jsp

Generate dynamic rich content in your server-side Java applications

JClass® ServerViews

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

JClass ServerChart

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

JClass ServerReport

Generate high-quality Adobe PDF for a global audience.

QUEST SOFTWARE®

Evaluate and experience JClass today – visit:

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

Optimize J2EE.

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

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

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

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

It's the Business Technology Optimization revolution.

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

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

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

Get Optimized.™

MERCURY INTERACTIVE