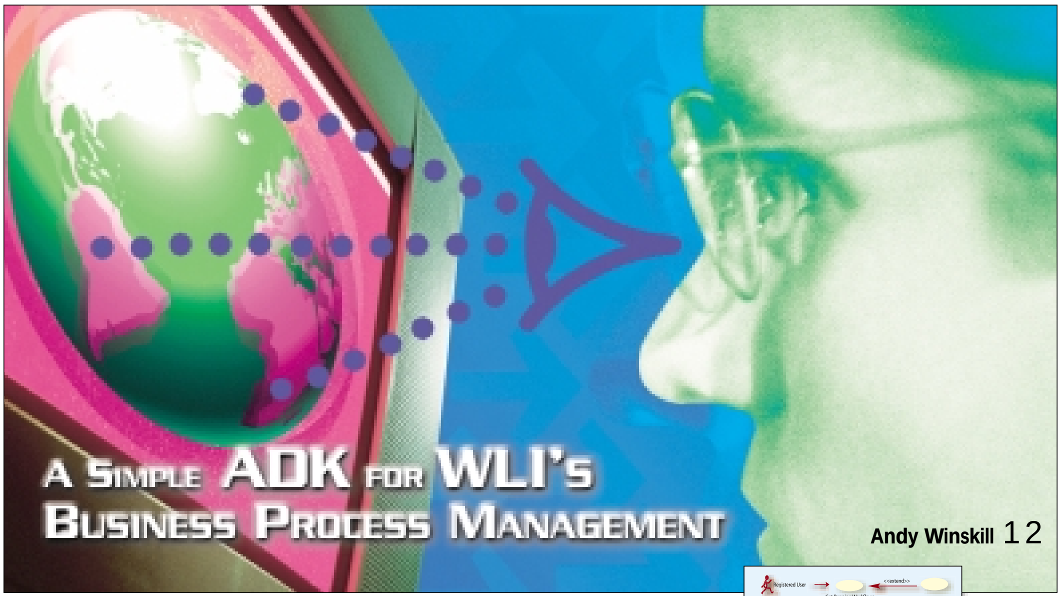
HEAR ADAM BOSWORTH & TYLER JEWELL AT WEB SERVICES EDGE

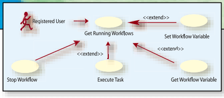
BEA WebLogic DEVELOPER'S JOURNAL

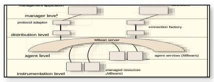
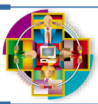
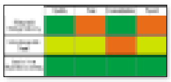
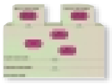
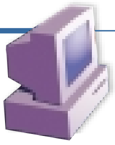
Volume:1 Issue:4

weblogicdevelopersjournal.com

SYS-CON MEDIA





BEA

www.developer.bea.com

BEA

www.developer.bea.com

Wily Technology

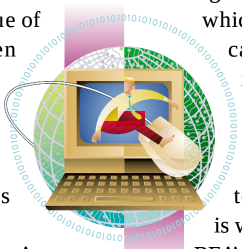
www.wilytech.com

BEA WebLogic Platform: Enabling Complex Web Services

BY JASON WESTRA
EDITOR-IN-CHIEF

Web services. Who needs them? You will. Indeed, I have. As a proof of concept for a wireless company, I wrote an application that allowed users to manage a fantasy football team from any WAP-enabled handheld. Users could set their lineup for the big day, or add and drop players from their roster. You know, the basics of running a fantasy team. The only glitch was integrating with the fantasy football service provider, who didn't have an open API to their back-end business processes and information. Had they provided a Web service that allowed me to retrieve player information and send updates, my development time would have decreased dramatically and the robustness of my application would have improved as well.

Why wasn't it an open Web service? In the past, I've been a skeptic about the value of Web services (until I needed one!). Even with all the Web services companies popping up these days, I still feel there are roadblocks ahead for wide adoption. Those I see center on the lack of adaption to alternative business processes and technical feasibility.



For instance, take the above example. A company offers a free service (e.g., fantasy football league) to millions of subscribers. A small fee is collected from members who purchase value-added services, but this revenue is nominal. How do you make money on this business model? I'll give you a hint. Did you see the Britney Spears commercials during the Super Bowl? You guessed it, advertising. Many companies offering "Web-serviceable" content on the Web are still reliant on advertising dollars. Web services promise to ease the trading of information. However, advertisers spending millions of dollars tend to frown on a technology that allows the circumvention of their promotions. Would you risk a multimillion-dollar advertising contract to build a Web service that opens your business up to the world, yet allows that world to filter out advertisements? The choice is hard to quantify, and I believe it's slowing the adoption of Web services today. My example is perhaps too focused on advertising-revenue–based firms, but illustrates my point. Businesses with established business processes and relationships are often unable to adapt their business models to be profitable using Web services. This lack of flexibility hinders adoption.

There are also technical hurdles preventing widespread adoption of Web services. One I'll cover today is the inability to perform complex transactions across multiple Web services hosted by numerous providers. To make Web services viable, the ability to perform asynchronous transactional communications is required. Some enablers include ebXML (Electronic Business eXtensible Markup Language) and BTP (Business Transaction Protocol), which allow businesses of any size to communicate across long-running business processes in a standard fashion. ebXML even provides semantics for advanced, asynchronous messages. However, the ability for providers to support asynchronous transactions is crucial to long-running, complex Web services. This is where the BEA WebLogic Platform shines. BEA's approach to Web services is to wrap EJBs and JMS destinations in SOAP wrappers. This approach uses the industry's strongest J2EE backbone, WebLogic Server, to support the asynchronous capabilities required to make Web services widely successful.

This issue of ***WLDJ*** features articles from Scott Dietzen, CTO of BEA, and Steve Chazin of Bowstreet. Scott elaborates on the impact of Web services on J2EE, while Steve talks about utilizing WebLogic Server to dynamically assemble applications with Web services. My confidence in Web services grows each time I hear about BEA's growing dedication to this technology.

Web services. Who needs 'em? If you don't already, you will, and BEA will provide the platform.

AUTHOR BIO...

Jason Westra is the editor-in-chief of WLDJ and the CTO of Evolution Hosting, a J2EE Web hosting firm. Jason has vast experience with the BEA WebLogic Server Platform and was a columnist for Java Developers Journal for two years, where he shared his WebLogic experiences with readers.

CONTACT: jason@sys-con.com

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

SYS-CON MEDIA

SQL Server isn't usually at the top of the list when it comes to selecting a database for J2EE development.

SQL Server and WLS

IT CAN BE A GOOD FIT

BY MICHAEL GENDELMAN

Organizations that have made a commitment to Java and J2EE are likely to be Unix shops, making them highly unlikely to choose SQL Server, which can only be hosted on the Windows platform. Furthermore, the front-end client tools that come packaged with SQL Server can only be used on Windows.

Even so, SQL Server can be a good fit for WebLogic Server projects. It implements all four ANSI standard transaction isolation levels – Read Uncommitted, Read Committed, Repeatable Read, and Serializable. Industry support is strong, with many vendors making JDBC drivers for SQL Server, including BEA and Microsoft. BEA includes special features in WLS just for SQL Server, which will be discussed later in the article. Microsoft's driver is still in beta, and can be downloaded at www.microsoft.com/sql/downloads/2000/jdbc.asp. A complete list of SQL Server JDBC drivers can found at http://industry.java.sun.com/products/jdbc/drivers.

WebLogic jDriver for MS SQL Server

BEA has a Type 4 SQL Server JDBC driver, which it licenses with WLS. A Type 4 driver should work on any platform because it doesn't require native libraries to be installed on the database client (in this case the WLS Application Server). Type 4 drivers are written entirely in Java, and communicate directly with the database server. Setting up the JDBC driver in WLS is quite easy, as the SQL Server driver is included in the weblogic.jar file. The only requirement is to set up the connection pool. You can even use BEA's SQL Server JDBC driver with standalone Java applications, as long as you include the weblogic.jar file and the license.bea file in the CLASSPATH. This is extremely useful when unit-testing code that performs database interaction. BEA's SQL Server JDBC driver is not XA-compliant, but can participate in distributed transactions. A distributed transaction, also referred to as a two-phase commit, is a transaction that spans two separate resources. All resources involved either commit or roll back the transaction together. WLS allows one non-XA–compliant resource manager to participate in distributed transactions.

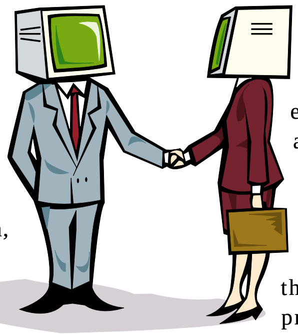
To set up the WebLogic jDriver for MS SQL Server within WLS, from the console select JDBC, then Connection Pools. Select "Configure a new JDBC Connection Pool". First fill in the URL. The first part of the URL will be "jdbc:weblogic:mssqlserver4:". Next add the name of the database, followed by an "@". Now set the server name, followed by a colon, and the port number:

```
jdbc:weblogic:mssqlserver4:DatabaseName@ServerName:1433
```

Next, set the "Driver Classname" to "weblogic.jdbc.mssqlserver4.Driver". Now set up the user name inside the properties. Enter "user=MyUserName". The WLS 6.1 console provides a special area to enter the password. If you're using an older version, just add "password=MyPassword" to the properties. You still need to configure the options on the "Connections" tab, and set up a TX DataSource, but that will be the same for all connection pools (see Figure 1).

Key Generation

SQL Server supports auto-increment fields, which makes key generation much easier and less complicated. Set up the primary key column as an auto-increment column. A counter will be incremented, and the new value will be placed in the primary key column of the newly inserted record. Only SQL Server can set the column's value. This provides the same key-generation mechanism to all applications that use the table, not just WLS applications. You can retrieve the value of the



AUTHOR BIO...

Michael Gendelman is a senior analyst with Viatech, Inc., a New Jersey consulting firm where he develops enterprise systems as a contractor for the U.S. Army. He has been developing distributive systems over the past five years utilizing DCOM, CORBA, and EJB, and is a Java Certified Programmer.

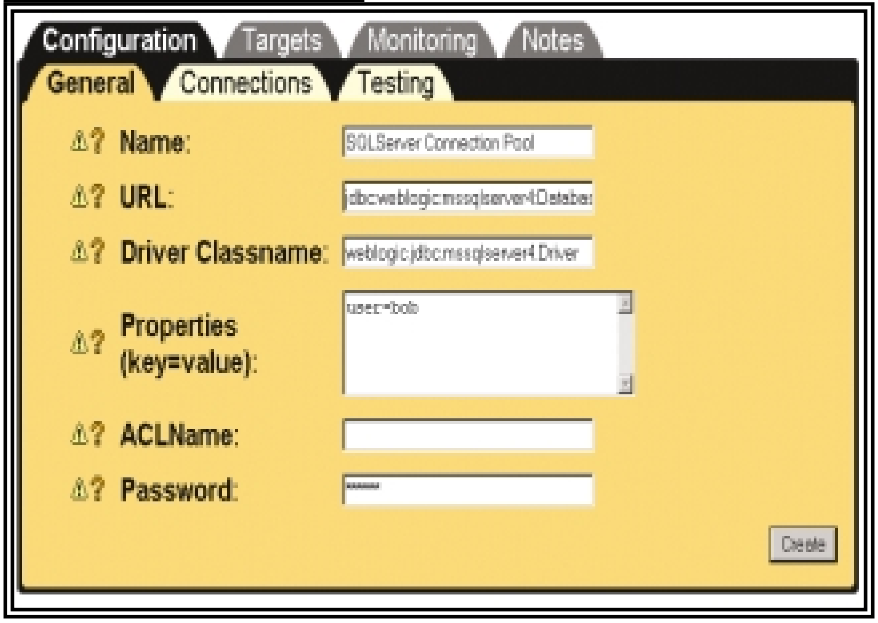
CONTACT...

mgendelman@yahoo.com

Sitraka

www.sitraka.com/jclass/wldj

FIGURE 1



Name: SQLServer Connection Pool
URL: jdbc:weblogic:mssqlserver4:Databa
Driver Classname: weblogic.jdbc.mssqlserver4.Driver
Properties (key=value): user=bob
ACLName:
Password: ******

WebLogic Portal J2EE architecture

new column by executing SELECT @@IDENTITY within the same transaction as the insert, as shown in Listing 1. If multiple inserts are executed within a single transaction, a very likely scenario, the identity value of the last record inserted will be retrieved by the SELECT @@IDENTITY.

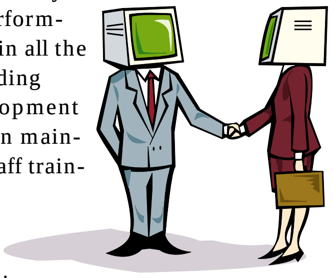
WebLogic Server has automated this process when using CMP (Container Managed Persistence) with SQL Server. Add the XML code in Listing 2 to your weblogic-cmp-rdbms-jar.xml file. The class mapped to the primary key field of the table must be of type Integer. And of course, the table's primary key must be set up as an auto-increment field. This feature is very nice, as it removes database-specific code. WLS has similar support for Oracle.

If you're using container-managed transactions, make sure to use Tx DataSource, not DataSource, when setting up your SQL Server Connection Pool. This is a good idea for many reasons, but especially important when using automatic key generation. The SQL statement that returns the key will fall outside the transaction and won't be able to return the key value if you use DataSource.

Conclusion

Any database with strong transactional support and a good JDBC driver can, at some level, support WLS. SQL Server meets both requirements, and in addition, has strong support from WLS. Organizations that have made a large commitment to SQL Server in the past will likely do well to stay with SQL Server. There may be a large number of production applications utilizing SQL Server, plus staff will be experienced at managing SQL Server in an enterprise environment.

The next question is what level of performance do you need from your database, and what is the most economical way to achieve that performance, factoring in all the variables, including licensing, development time, production maintenance, and staff training.



I've covered some of the issues surrounding SQL Server and WLS integration. Before making a choice you must still perform some due diligence. The choice comes down to whether SQL Server is capable of providing the level of service you need.

Listing 1

```
protected long getKeyValue(Connection con) throws SQLException {
   ResultSet rs = null;
   Statement stmt = null;
    try {
      stmt = con.createStatement();
      rs = stmt.executeQuery("SELECT @@IDENTITY");
      rs.next();
      return rs.getLong(1);
    }
    finally {
      try{ rs.close(); } catch (Exception e){}
      try{ stmt.close(); } catch (Exception e){}
      try { con.close(); } catch (Exception e){}
    }
  }
```

Listing 2

```
<weblogic-rdbms-jar>
<weblogic-rdbms-bean>
<automatic-key-generation>
<generator-type>SQL_SERVER</generator-type>
</automatic-key-generation>
</weblogic-rdbms-bean>
</weblogic-rdbms-jar>
```

Precise

www.precise.com/wldj

As we've discussed over the past few issues, JTA-style transactions provide a way for multiple data updates to be tied together so application logic can operate safely in the assumption that it will succeed or fail consistently, even in the face of technical failures along the road.

Transactions, where do they begin and end?

BY PETER HOLDITCH

There are, however, times when you do not want all the work you do to succeed or fail in one big lump.

Among the scenarios that may lead you to want to break work up into smaller chunks are:

- Progress/error logging
- Long-running operations
- Wide-spanning operations

I discussed the last two scenarios in a previous article (***WLDJ***, vol. 1, issue 2) – transactions hold locks on data. Holding locks for a long time may cause your application to freeze, unable to access data it needs. The wider the span of a transaction (measured in terms of data elements touched and hence locked), the more likely you are to run into deadlock situations, and the longer running the transactions will be – running right back into the previous bullet.

The first point, about logging, is more of a business logic–driven consideration. Many people use databases to record events in their systems. Take the example of an audit log. Imagine a banking application that logs all fund movements. Imagine also a teller trying to fraudulently withdraw $10,000 from some account. In order to track the fraud, the application users are likely to want to look through the audit log. If the fraudster had tried 1,000 withdrawals, then the auditors would want to see 1,000 pieces of evidence. They would be pretty ticked off if they only saw 900 records, since 100 withdrawals failed, and the writing of the log records rolled back with the failed withdrawals. Clearly, the log record cannot be written in the same transaction as the withdrawal.

Transactions: Where Do They Live?

When you write the code to start an XA transaction, it's associated with the thread of execution, and irrespective of what methods in whatever classes you call, any database connections that you get and use will inherit the transaction context from the thread. Thus, when the transaction is finally committed, all the work that you've caused to be done – whether or not it was in utility classes that you don't even know the contents of – will complete as a transactional atomic unit. Given that the transaction lives in the thread and is so pervasive, how do you break up the work done by your thread of control into multiple transactions?

Transactions, What A State!

The answer to this problem is that transactions can be in more states than simply active or inactive. They can also be suspended. A suspended transaction is still not complete – the timer controlling the timeouts etc. is still running, and can roll back of its own accord if the timeout is exceeded – but it's no longer associated with the current thread of execution.

–continued on page 26

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...

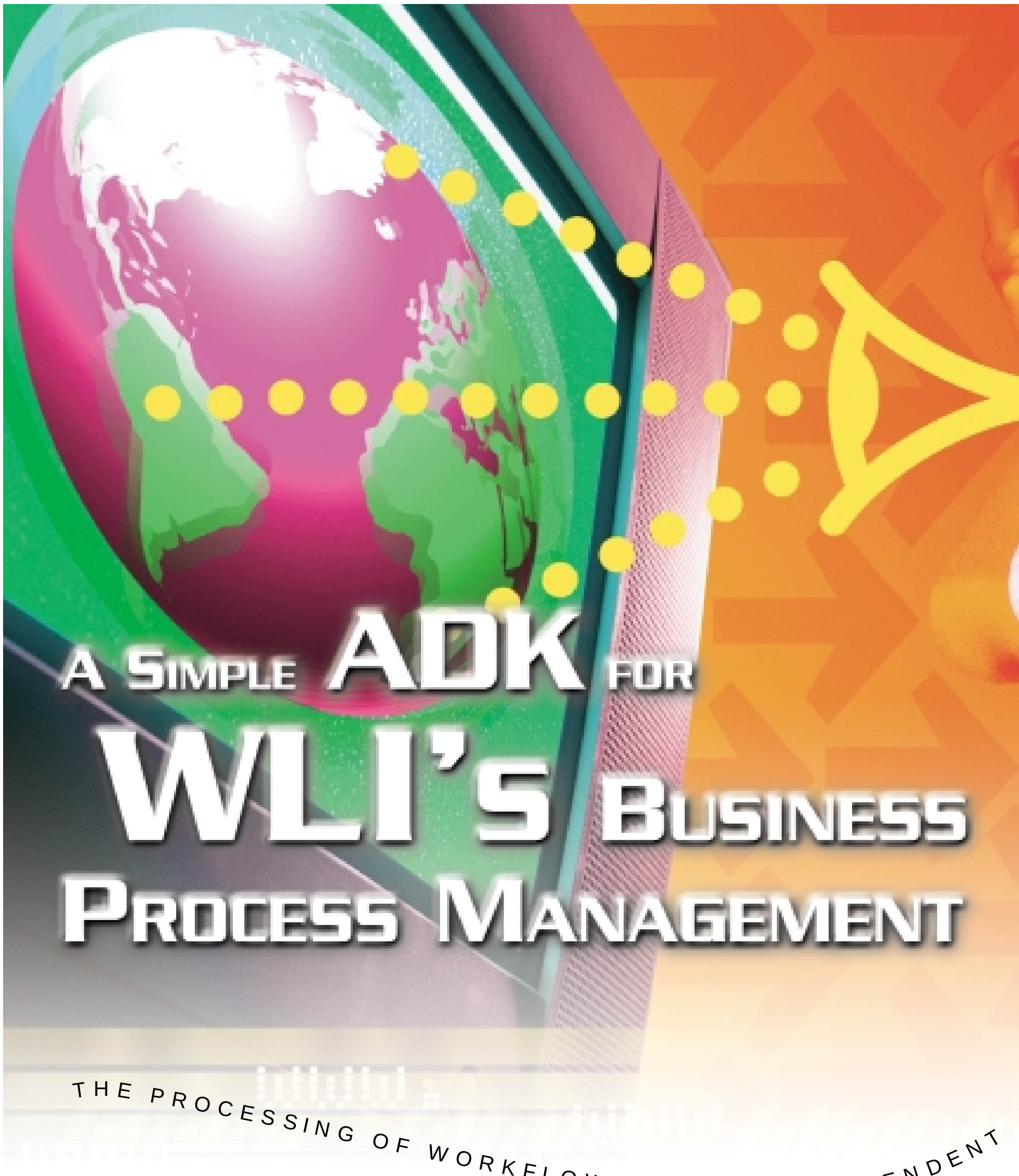
peter.holditch@bea.com

Intel

www.intel.com/ebusiness

A Simple ADK for WLI's Business Process Management

THE PROCESSING OF WORKFLOWS CAN BE INDEPENDENT



BY
ANDY WINSKILL

AUTHOR BIO...

Andy Winskill is principal consultant at Rosewoord Software Services Ltd, in the UK. Andy specializes in BEA and Rational software, and has more than 10 years of experience designing and constructing EAI and B2B applications. Rosewood Software Services is a BEA and Rational partner.

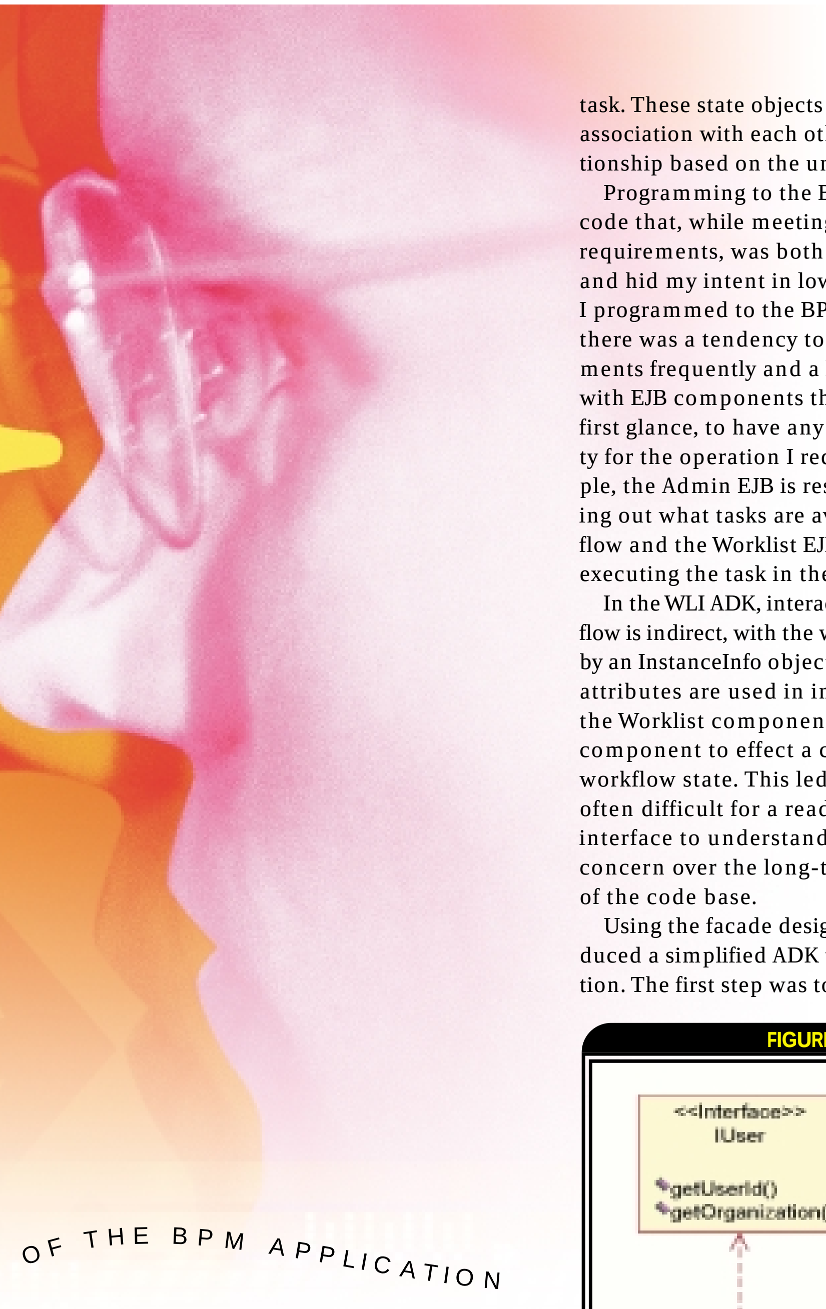
CONTACT...

andy.winskill@rosewoodsoftware.com

WebLogic Integration (WLI) consists of many application components, including a B2B application that manages business-to-business contracts during software conversation. To support the B2B application, WLI has a business process management (BPM) application for connecting business processes together. This used to be marketed as a discrete product, WebLogic Process Integrator (WLPI).

The BPM application's ADK requires the application developer to understand the architecture of the BPM application. The ADK has a number of components whose use depends on the state of the workflow and the operation that's required. Interaction with the BPM ADK's components and objects leads to an application developer coding to a low level interface.

This article outlines the design of a simple facade to the BPM ADK, providing a clean object interface that is, by intent, easy to use. It simplifies interaction with the



OF THE BPM APPLICATION

BPM engine and ensures that changes in the underlying BPM implementation won't affect the ADK adversely. Access to the BPM application is solely via the publicized WLPI interface.

Using the Simplified ADK

In creating this ADK I wanted to produce a simple object model that could be used within either a Java client or a JSP. Due to the nature of the BPM application there are many state objects. For example, an InstanceInfo object describes a running workflow and a TaskInfo object describes a task. These state objects have no direct association with each other, but have a relationship based on the underlying RDBMS.

Programming to the BPM ADK led to code that, while meeting the application requirements, was both difficult to read and hid my intent in low-level detail. When I programmed to the BPM ADK directly, there was a tendency to parse XML documents frequently and a lot of interaction with EJB components that didn't appear, at first glance, to have any direct responsibility for the operation I required. For example, the Admin EJB is responsible for finding out what tasks are available in a workflow and the Worklist EJB is responsible for executing the task in the workflow.

In the WLI ADK, interaction with the workflow is indirect, with the workflow described by an InstanceInfo object. The InstanceInfo attributes are used in interactions with the Worklist component or the Admin component to effect a change in the workflow state. This led to code that was often difficult for a reader new to the BPM interface to understand, which caused me concern over the long-term maintenance of the code base.

Using the facade design pattern, I produced a simplified ADK to the BPM application. The first step was to understand the main use cases and key entities that most BPM applications have. The use cases in many BPM clients are:

- Find out what workflows the user could start and then start one of them
- Execute a task (or mark it as done)
- Set or get a workflow variable's value
- Stop a workflow

The BPM application would then be responsible for interacting with other applications to satisfy the business process.

ADK Key Concepts

In most applications the key entities for interaction with the BPM interface were often found to be User, Workflow, Task, and Variable, as workflows are inherently stateful (see Figure 1).

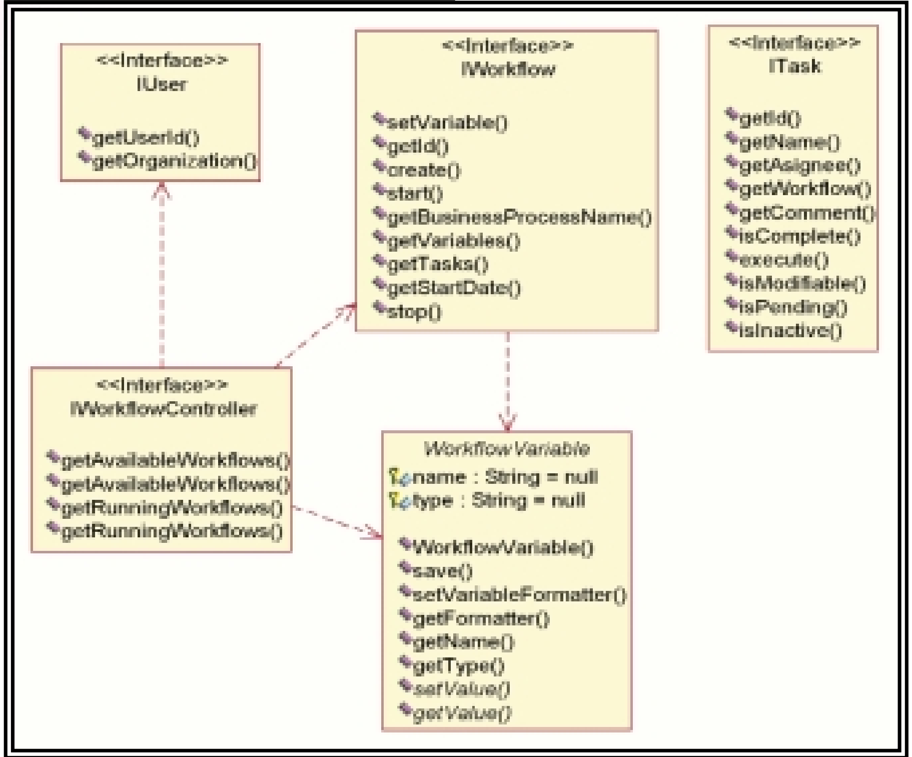
IUSER

The IUser is a simple interface. Its contract states that the user should be identifiable and have an organization. As I show below, the WLIUser class is the entry point to the BPM ADK.

IWORKFLOWCONTROLLER

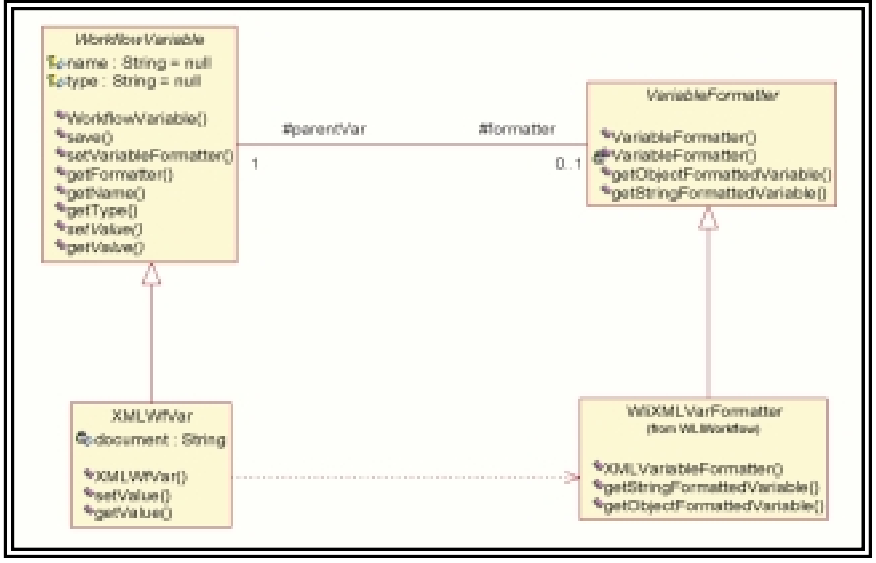
The IWorkflowController is a simple interface with two main overloaded methods, getAvailableWorkflows and getRun-

FIGURE 1



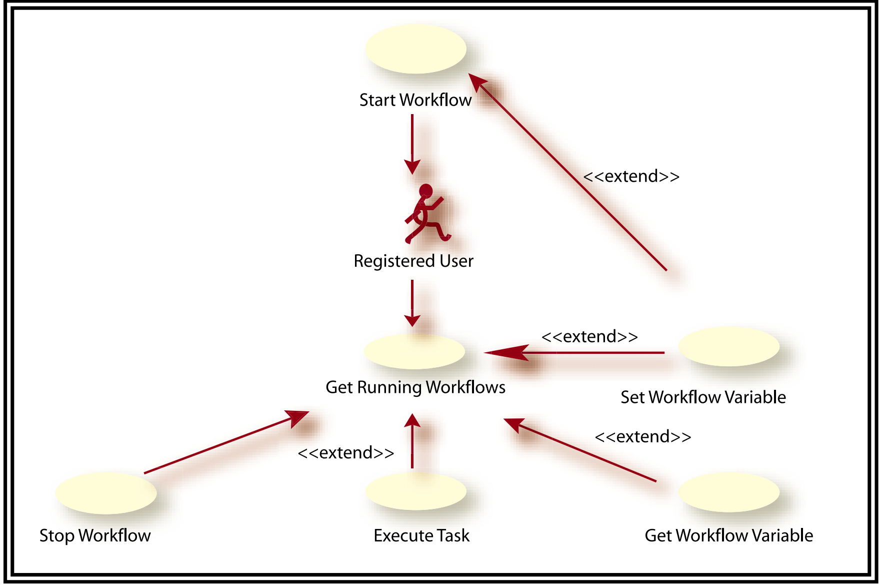
ADK key concepts

FIGURE 2



Workflow variables and variable formatters

FIGURE 3



The use-case model

ningWorkflows. I defined an available workflow as one that the user can start and a running workflow as one that has been started. I intended to make the controller an intelligent factory, removing the need to create a workflow instance directly. If the underlying BPM application changes, with this mechanism the ADK's clients would then be decoupled from the change.

IWORKFLOW

The IWorkflow interface says that a workflow can be started and stopped, variables set and read, and a set of tasks (ITask) can be accessed. It doesn't expose all the information that a WLI workflow can contain. It does, however, capture what I felt was the general essence of the workflow that I felt sufficient when exploring the ADK's design. Future implementations of the interface may expose more information.

In this ADK a workflow can have three main states: defined, running, and stopped. A defined workflow is defined within the BPM studio. It has tasks and may have workflow variables defined. Defined workflows can have instances created if the permissions of the workflow allow it. A workflow instance is a running workflow. Running workflows can have tasks waiting to be executed and may have workflow variables to set. A stopped workflow is a workflow instance that's been stopped. In this stopped state Workflow variables can be read, but their values can't be set.

VARIABLES

A workflow variable can have many types, from Java objects to XML documents. These are defined in the WLI documentation. This ADK is designed to work with most forms; however, only the XML variables are currently implemented. The problem was how to get variables into WLI. As a Java variable could have many classes, in addition to BPM applications having different or changing ways of setting a workflow variable, there was a design challenge. By applying the Mediator pattern it became simple to define a variable formatter to overcome the system boundary (see Figure 2).

TASKS

A workflow task is defined within the BPM studio. The Task is a collection of actions. This ADK does not interface with the actions directly, as this is the domain of the BPM application. My intent is to delegate the process logic to the BPM engine rather than have it hard-coded. The task interface defines operations to query the state of the task and to execute the task.

Use-Case Model

The process I adopted for this development was to implement the ADK in a use case–driven manner, the use cases providing the impetus to understanding how the ADK could be used. The application I produced was a simple command line interface to the workflow system. The command line application would be able to start and stop workflows, execute tasks in a workflow, and set and get workflow variables. The simple use-case model is shown in Figure 3.

Use-Case Realizations

I use a single use-case realization to describe the ADK design.

STARTING A WORKFLOW WITH VARIABLE

In this use case we need to start a workflow for a user see Figure 4. The workflow should have a value passed to it before starting. In WLI BPM this is where a workflow that has a mandatory input variable is defined.

Sitraka

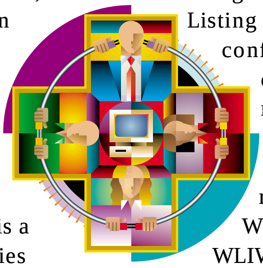
www.sitraka.com/jclass/wldj

The IWorkflowController creates a list of IWorkflow objects depending on whether the client requires a new workflow started or an existing workflow. Creation of WLIWorkflow objects is only via the IWorkflowController interface and the class WLIWorkflowController, shown in Figure 5, realizes this interface. The workflow controller is responsible for generating lists of workflows based on the user's permissions. How the permissions affect the starting of a workflow is dependent on the underlying workflow engine. A WLI user can belong to multiple organizations with one organization specified as the user's default organization. For this iteration of the ADK I decided that the WLIWorkflowController would only access workflows in the user's default organization.

An instance of WLIWorkflowController is normally created with a WLIUser object specified. The WLIUser class acts as a facade to the underlying WLIPrincipal remote interface and UserInfo class. It creates an association to the underlying UserInfo object. I decided to keep the IUser interface simple since it focuses on the use cases described above.

ADK CODE WALK

To illustrate how the ADK can be used, we'll walk through the pseudo code in Listing 1.



In line 1 a WLIUser object is created and the J2EE connection information is supplied, along with the organization that the user will belong to for this interaction. The J2EE connection information object is a data structure for holding all properties required to open the JNDI initial context. At line 2 the WLIWorkflowController is created for the user. Line 3 illustrates the use of the workflow controller to get the workflows that are available to the user. Lines 4 through 7 are interactions with the user. In line 8 I create a WorkflowVariable that's an instance of XMLWfVar (XML Workflow Variable) and in line 9 a value for the variable is set. Line 10 introduces the concept of the VariableFormatter, discussed below. The workflow variable is associated with the workflow in line 12. Finally, the workflow is started.

WLI Implementation

In this section I'll discuss the design of the ADK with regard to the underlying WLI implementation.

WLIUSER

The class WLIUser aggregates a reference to the WLPIPrincipal EJB, which is responsible for accessing the security principal of the caller. The call

```
theUserInfo = principal.getUserInfo(userId);
```

returns a state object of class UserInfo. The WLIUser object caches this state, and maintains the reference to the WLPIPrincipal. Access to the WLPIPrincipal cache is available to other classes in the Java package.

WLIWORKFLOWCONTROLLER

The WLIWorkflowController uses a reference to the Worklist EJB. For a call to WLIWorkflowController.getAvailableWorkflows(), the code in Listing 2 shows that the workflow list is first configured for the caller's organization. The call to Worklist::getStartableWorkflows returns a list of TemplateInfo objects that are used to create WLIWorkflows. The WLIWorkflowController passes a reference to the BPM's Admin EJB to the WLIWorkflow on construction of the WLIWorkflow. This EJB is used to control access to the workflow variables.

To access running WLI workflows the client would use the method WLIWorkflowController.getRunningWorkflows(). The code snippet below shows the mechanism used within the ADK. Access to running workflows in the WLI ADK is not via the Worklist EJB, but rather by the Admin EJB.

Listing 1: Starting a workflow that requires a workflow variable value (pseudocode)

```
1:  IUser user = new WLIUser(j2eeConnectInfo,"CDExpress");
2:  IWorkflowController wfControl = new WLIWorkflowController(user);
3:  WorkflowList wfList = wfControl.getAvailableWorkflows();
4:  displayWorkflows(wfList);
5:  int selection = getUserSelectionToStart();
6:  String varName = getVariableName();
7:  String varValue = getXMLValue();
8:  WorkflowVariable var = new XMLWfVar(varName);
9:  var.setValue(varValue);
10: var.setVariableFormatter(
    new XMLVariableFormatter((XMLWfVar) var));
11: IWorkflow wf = wfList.get(selection);
12: wf.setVariable(var);
13: wf.start();
```

Listing 2: Applying the Mediator pattern

```
theWorklist.setActiveOrganization(user.getOrganisation().getId());
// Get the workflows for the users organisation
List wflows = theWorklist.getStartableWorkflows(
                   user.getOrganisation().getId());
       // Build the workflow list
        for (int i=0; i < wflows.size(); i++)
        {
          TemplateInfo tmplInf = (TemplateInfo) wflows.get(i);
          IWorkflow wf = new WLIWorkflow(admin, theWorklist, tmplInf);
          wfList.addWorkflow(wf);
 }
```

```
    instanceList =
     admin.getTemplateInstances(wf.getId(),
        user.getOrganisation().getId(),
    true,from,to,0,0);
```

To find a running workflow, the ADK must first find workflows that the user can start. Once the list of workflows that can be started is created, the ADK queries the Admin EJB to find out which workflow templates have running instances. The simplified ADK hides the client from this complexity.

WLIWORKFLOW

The WLIWorkflow object is created with a reference to the Admin and Worklist EJBs. These EJBs are the main access route for workflows into the BPM application. If the Workflow is being created for a workflow that has not yet been started (i.e. in a *defined* state), the WLIWorkflow object must be created with a reference to the com.bea.wlpi.common.TemplateInfo object for the workflow. The BPM's TemplateInfo class is responsible for encapsulating simple information about a template's instance. The Worklist EJB in the WLIWorkflowController returns the TemplateInfo object.

Should the workflow be running, the WLIWorkflow object must be created with a reference to a com.bea.wlpi.common.InstanceInfo object. This contains state information about the workflow and is used by the WLIWorkflow object to access task information.

Starting a workflow in WLI is achieved in one of two ways. The mechanism depends on if the workflow has had variables associated with it. I wanted to hide this from my ADK's clients.

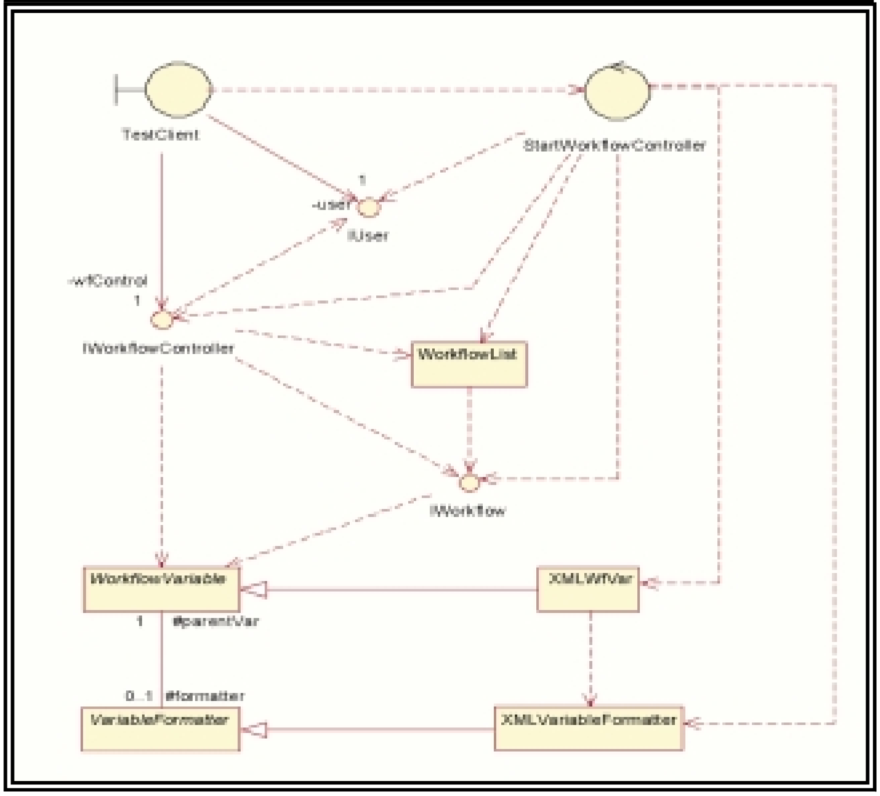
If we have a workflow in a defined (nonstarted) state, the WLIWorkflow object starts the workflow as follows:

```
wList.instantiateWorkflow(
   wList.getActiveOrganization(),
                                wfTemplate.getId());
```

The instantiateWorkflow() method in com.bea.wlpi.server.worklist.Worklist is used, and has the organization and the template identifier passed to it. To me, this mechanism doesn't manipulate workflows; rather, it relies on the client to track template identifiers and organizations. I wanted to hide this level of detail from the ADK's clients; treating the Workflow as an entity in the design allowed me to use the facade pattern to abstract the client away from the underlying WLI complexity. The importance of this is further underlined when setting variables in running workflows or manipulating tasks.

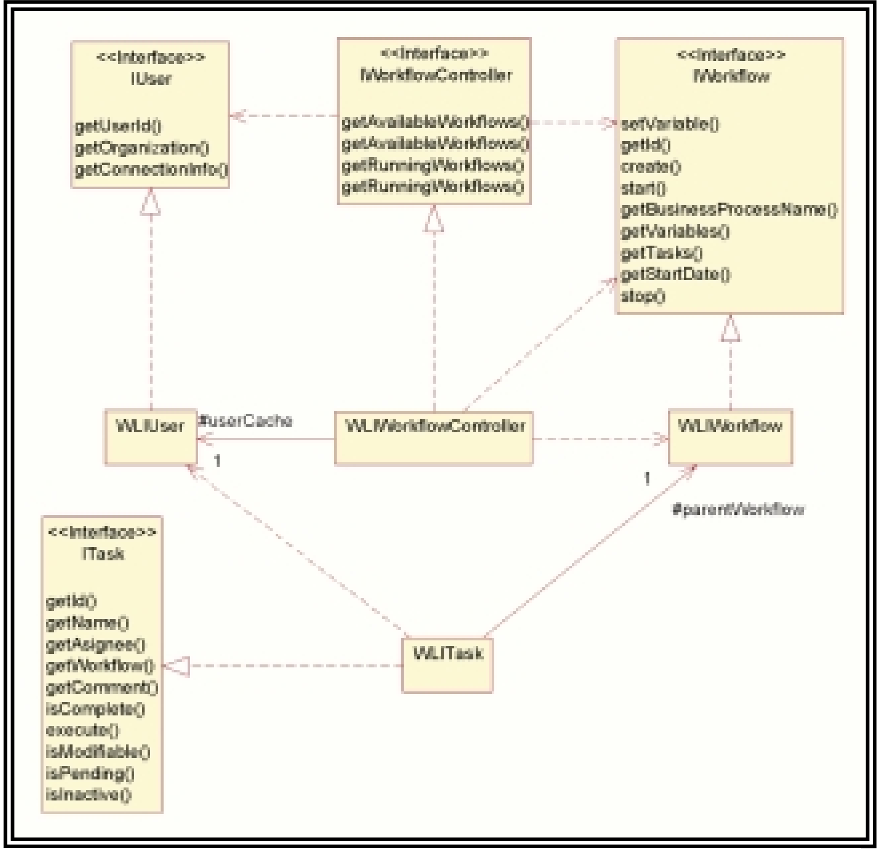
If the workflow has variables to be passed to it before starting, the ADK's clients need to have invoked setVariable on the WLIWorkflow object. The workflow object caches the variables in a HashMap, which is then passed to the overloaded instantiateWorkflow() method defined in com.bea.wlpi.server.worklist.Worklist.

FIGURE 4



Starting a workflow - view of participating classes (WLI Realizations omitted for clarity)

FIGURE 5



WLI realizations of key interfaces

Variables

The ADK's concept of Workflow Variables has already been introduced above. In Figure 2 I showed the class diagram for the workflow variables and variable formatters. A key design consideration was to decouple the ADK's clients from the underlying variable structure that a BPM engine required. In WLI's BPM application the variables are passed to the workflow instance via either the com.bea.wlpi.server.admin.Admin EJB or the com.bea.wlpi.server.worklist.Worklist EJB. The Worklist EJB is used if the workflow is to be started in the same call. If the workflow has already started, the Admin EJB is used.

In the Worklist EJB, passing a hashmap of the variable objects to the instantiateWorkflow() method sets the input variables. The keys to the hash map are used for the variable names; the object in the hash map is used for the variable value.

In the Admin interface, the method below can be used:

```
admin.setInstanceVariable
    (instanceInfo.getTemplateDefinitionId(),
           instanceInfo.getId(),
           var.getName(),

var.getFormatter().getObjectFormattedVariable());
```

I wanted the clients to be abstracted from the complexity required to support multiple variable formats. The Mediator pattern was an excellent solution for this required decoupling. Now the client can set a workflow variable on a workflow and the client doesn't need to be aware of the underlying mechanism that the BPM requires, or the format of the workflow variable. The client needs only to associate the correct variable formatter with the variable.

Tasks

In the WLI BPM, application tasks are where operations are performed. These may be manual actions that a user must undertake, or a programmatic action that allows the BPM to integrate existing application processes. If a task is accessed within a Java application, then the application is interrogating the state of the task to either monitor the progress of the workflow or use the task as an indicator that the application is required to undertake some manual action. If this isn't the case, then it's probably that there are better mechanisms to achieve the requirement, such as using JMS signals from the BPM.

"This led to code that was often difficult for a reader new to the BPM interface to understand, which caused me concern over the long-term maintenance of the code base."

In the ADK, tasks can have one of the following states: started, completed, or inactive. WLI has an overdue state which isn't currently supported. An inactive task is one that's defined in the BPM, but the preconditions for its starting haven't yet been met. A started task is one where all of its preconditions are met and the automatic processing of the task's actions has started. A completed task has completed all of its actions and been marked done. Marking a task as done can trigger additional processing within the task. For this reason, we execute a task within the ADK. This signifies that there may be processing yet to be done. Once the task is executed (all actions are performed) then the task is completed.

Access to a WLI workflow's tasks is via the com.bea.wlpi.server.admin.Admin EJB using the method:

```
List lst =
admin.getInstanceTasks(instanceInfo.getId());
```

This method returns a list of com.bea.wlpi.common.TaskInfo objects. As a workflow has tasks, I did not want to expose the Admin EJB to the ADK's clients. Using the IWorkflow::getTasks() method, the client can access tasks directly from the workflow object. This method will return a list of ITask objects. Once the client has the task object it can manipulate the task without having any dependency on the Admin EJB.

Executing the task requires an interaction with the com.bea.wlpi.server.worklist.Worklist EJB. The method used within the simple ADK is shown below:

```
wList.taskExecute(taskInf.getTemplateDefinitionId(),
                        taskInf.getInstanceId(),
                              taskInf.getTaskId());
```

Currently, the WLITask doesn't have any method for waiting for task completion. The WLI BPM application is designed to support tasks that are very long running, spanning days or months. I've stayed away from specifying this behavior in the task interface. Should ADK clients require this functionality, then I believe they should be aware of the implications!

Conclusion

I've outlined the design of a simple facade ADK to the WLI BPM application. The WLI BPM, while not very complex, requires that the developer be aware of the BPM architecture. This facade uses the key concepts of User, Workflow, Task, and Variable to make the processing of workflows independent of the BPM application. As the BPM application matures, the need for a façade like this may decrease. For now, this facade provides an abstraction from the underlying middleware.

Thought, Inc.

www.thoughtinc.com

WebLogic Platform 7.0

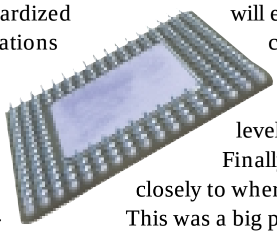
FLEXIBILITY FIT FOR ANY IT ENVIRONMENT

BY **SAM PULLARA**

DEVELOPMENT AND DEPLOYMENT. These are the foci of the WebLogic Platform 7.0 release. Don't get me wrong, it's not like we haven't been working on the container itself, we still have J2EE 1.3 compliance and some really high ECPerf numbers. We have, though, released with the product three tools that try to simplify the development, deployment, and administration of WebLogic Server applications.

The first tool, WebLogic Workshop, can be thought of as a Visual Basic–like tool for building Web services, complete with a debugger for those services. It's standard, too – underneath the Web service layer lies all the power of the J2EE stack that we've worked so hard to make fast, scalable, and reliable. Anyone who is familiar with WebLogic should bring up this tool and its tutorials, learn how to use it, and then show it to people who aren't yet convinced that Java is the way to go. You shouldn't have much trouble convincing them that this is an easy way to develop Web services.

For the developers out there who complained bitterly about losing the graphical deployment descriptor editor in 5.1, we've created a new tool called WebLogic Builder. This tool lets you edit deployment descriptors for J2EE application EARs, WARs, and EJB JARs in place and then redeploy those applications. It has built-in error checking and full coverage for all the descriptors. That should make it much easier and much less error-prone to deploy applications you've built or are porting to WebLogic from another system. It even auto-upgrades your existing older deployment descriptors to J2EE 1.3 so you don't have to change them by hand. With EJB 2.0 fully supported and standardized in this release, I think you'll find that all those complicated relations are much easier to manage inside Builder.



Maybe you're the kind of developer who scoffs at graphical tools and would really rather just use Emacs. Well, we have a productivity tool for you as well. WebLogic EJBGen lets you completely define your EJB implementation and its deployment information within a single file. Since the deployment information isn't separate from the implementation, it's much easier to make sure that all your relations are mapped and all your container-managed fields are hooked up. Using simple JavaDoc-like tags, you specify this information where it's needed, instead of trying to keep multiple files all in sync with each change.

And we haven't forgotten to include new enterprise-level features. One of the biggest additions (and my personal favorite, being a queuing kind of person) has to be the clustered queues and topics. Now you can build JMS-based applications without a single point of failure – without having to do all the hard work yourself. The distributed topics and queues do almost all the work for you. Very few, if any, code changes will be needed to take advantage of this feature. Now building that huge scalable chat server is just a few JMS calls away!

On the security side, we've also built in support for entitlements and the full extension of the security stack in the server. This very sophisticated security system lets you precisely control access to resources beyond simple ACLs. Now you can stop people from using parts of a Web application based on the status of their account, or allow them to make changes to their account and reports while disallowing access to every other account. This new security system really gives you much tighter control over your J2EE components than the broad, coarse-grain, method/URI-level security built in.

One of the hardest features to get just right, so everyone is happy, is the deployment step. We think we've covered all the bases here. With our new two-phase cluster deployment with the option of using shared disks or your own replication instead of our internal file replicator, everyone should be able to reliably deploy their applications the way they want. For instance, if you want all your clustered servers to pick up .jsp- and servlet-class changes automatically, you can configure your application to reside on a shared disk and specify that the application shouldn't be staged by WebLogic. On the other hand, you might be running a really tight ship, where files can only change when the application is redeployed. So you tell WebLogic to do the staging and it will ensure that an application isn't deployed until all the clustered servers are prepared and then the change happens all at once across the cluster. The staging options are changeable at both the server and application level, so you don't have to be locked into one configuration.

Finally, the top-level domain configuration is no longer tied closely to where the WebLogic server itself is installed on the machine. This was a big problem for people who wanted to put the server code on a read-only disk while keeping all their configuration data, logs, applications, etc. on another part of the disk. Simple things like this go a long way toward ensuring the flexibility to deploy the WebLogic Platform in any IT environment.

Phew! That's a lot for you to digest. However, I think you'll find that the BEA WebLogic Platform is expanding in a good way. We're building the most complete, scalable, secure, reliable platform for developing J2EE-based applications. That's our mission.

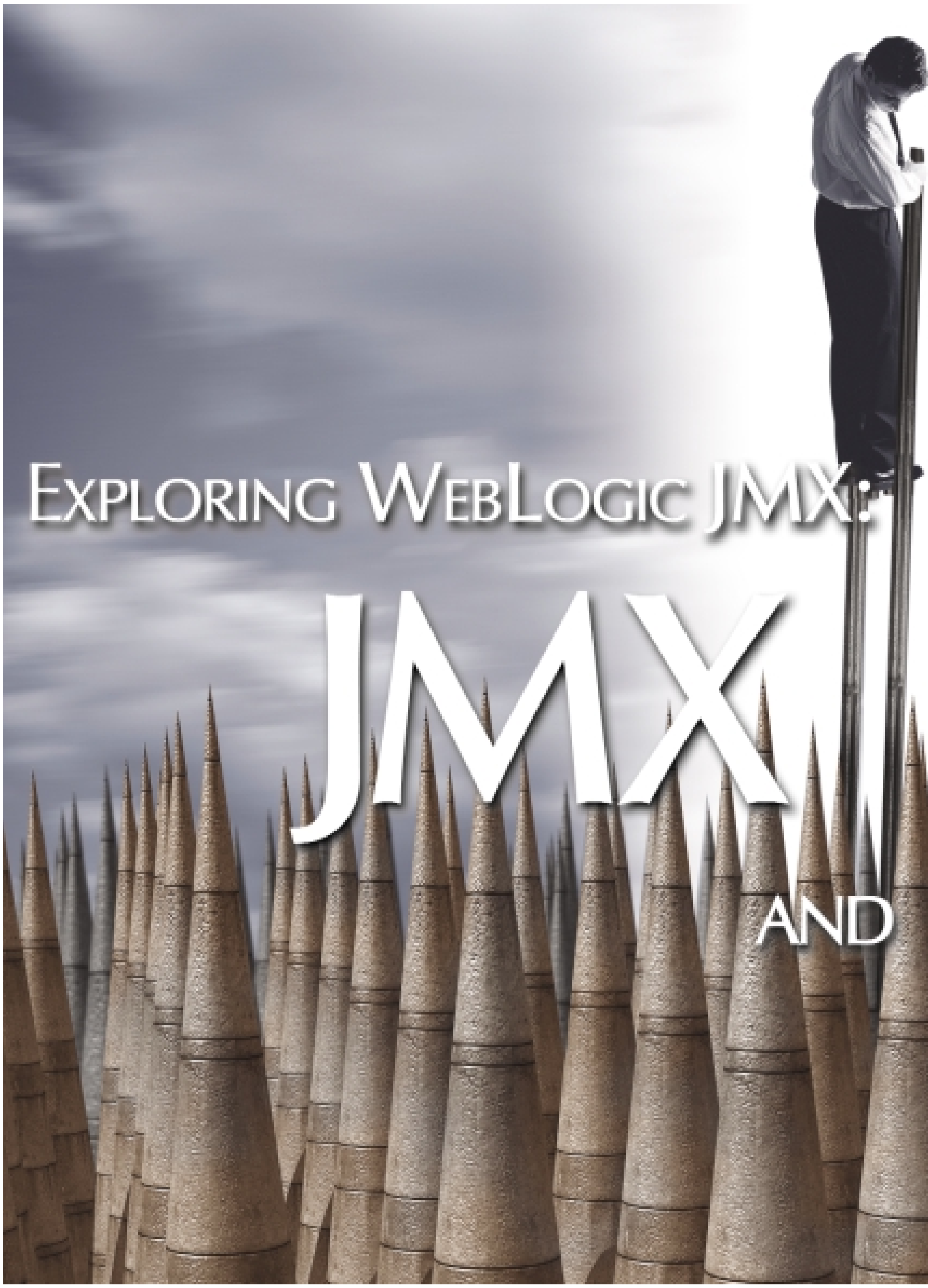
AUTHOR BIO...

Sam Pullara has been a software engineer at WebLogic since 1996 and has contributed to the architecture, design, and implementation of many aspects of the application server.

CONTACT: sam@sampullara.com

ReportMill

www.reportmill.com



Exploring WebLogic JMX: JMX and

BY
DAN MACKINNON

AUTHOR BIO...

Dan MacKinnon, a senior software engineer at WebGain Inc., was lead developer of the container-managed persistence integration for several releases of TopLink for WebLogic, and is currently working on EJB- and J2EE-based product initiatives. Dan holds a BSc and MSc in Mathematics from Dalhousie University, and a BCS from Carleton University.

CONTACT...

dan.mackinnon@webgain.com

J2EE is rapidly becoming an established platform for deploying long-running business-critical applications. As the number of J2EE applications grows and their importance increases, a standard way to manage J2EE servers and applications is becoming a key requirement.

Java Management Extensions (JMX) provide a pure Java management infrastructure that can be used for all Java-based software.

BEA WebLogic's support for JMX comes in not only providing a JMX implementation, but in basing all of its key administration and management features on this standard. The administration and management features of WebLogic are now extensible – open to both developers and third-party tool integrations.

There are obvious benefits in improved and extensible administration, but J2EE application developers still face several questions: What exactly does JMX provide and how is it likely to be used? What management features should developers build into J2EE applications? Which management facilities should the application server provide automatically? This first article in a two-part series on WebLogic JMX provides a brief overview of JMX and discusses some of its implications for enterprise Java developers.

JMX and Application Management

An application is *instrumented* if it provides a well-defined public interface that allows other applications to access its management or administrative functions. The management interface of an application does not generally resemble its public client interface. Instead, it is intended for privileged users, exposing data and functions for monitoring and administration. A component that has been instrumented to provide such a management interface is a *manageable resource*.

Manageable resources include devices such as printers, network routers, and telephone switches. More generally, any software component that resides on a server or in a distributed environment that is available to service requests is a good candidate for instrumentation. J2EE components fit well into this larger category. Management interfaces for such components include functions for tracking resource use, measuring client volumes, generating error logs, stopping and starting services, and load-balancing.

Developers (when tuning and debugging systems), system administrators (when monitoring applications and intervening when necessary), and other users employ management applications to access instrumented manageable resources. Management applications range from graphical resource-use monitors that run locally to Web-based administrative consoles that can be accessed remotely. Many of these tools use standard protocols to discover and display managed resources and their interfaces dynamically.

JMX is intended to be simple enough to allow developers to rapidly provide a management layer for either new or existing Java applications, and comprehensive enough to be adapted to other standard (non-Java) management protocols. The JMX specification, currently in its 1.0 release, is being developed through the Java Community Process. The JMX reference implementation is available from Sun Microsystems, which also provides a commercial JMX-based product, the Java Dynamic Management Kit (JDMK). A number of J2EE vendors now include JMX implementations in their application servers; WebLogic is one of the more comprehensive of these.

A specification for J2EE-specific management requirements based on JMX is currently under development, as are other specifications that describe how JMX should be adapted to other management protocols.

JMX Architecture

JMX presents a layered architecture, shown in Table 1 and Figure 1. At the two extremes of the JMX levels are the instrumentation level and the management level, representing managed resources and management applications. Between these are the agent and distribution levels, which provide the necessary protocols for applications in the management level to access the manageable resources of the instrumentation level.

Managed Beans, or MBeans, make up the instrumentation level. These objects represent the manageable resources of the system, and are the primary components of the JMX architecture.

An MBean Server is the most essential part of the agent level. It provides a means for creating, registering, and finding MBeans within the system.

The agent level includes several services that are also represented by MBeans.

The distribution level may provide remote access to the agent level through CORBA, RMI, or other means, and may also provide adapters for other protocols. It may provide security as well.

The management level consists of the external applications that can access the MBeans via the APIs exposed through the agent layer, using protocols provided by the distribution layer.

WebLogic goes beyond the JMX 1.0 specification to provide a distribution layer that maps logically onto the underlying agent level. Both local and remote users of JMX can access MBeans through the corresponding MBeanHome, a WebLogic-specific component that provides a client with access to MBeans. WebLogic's MBeanHome components offer some distribution facilities, as well as other features that make them easy to use.

Using JMX

SIMPLE INSTRUMENTATION

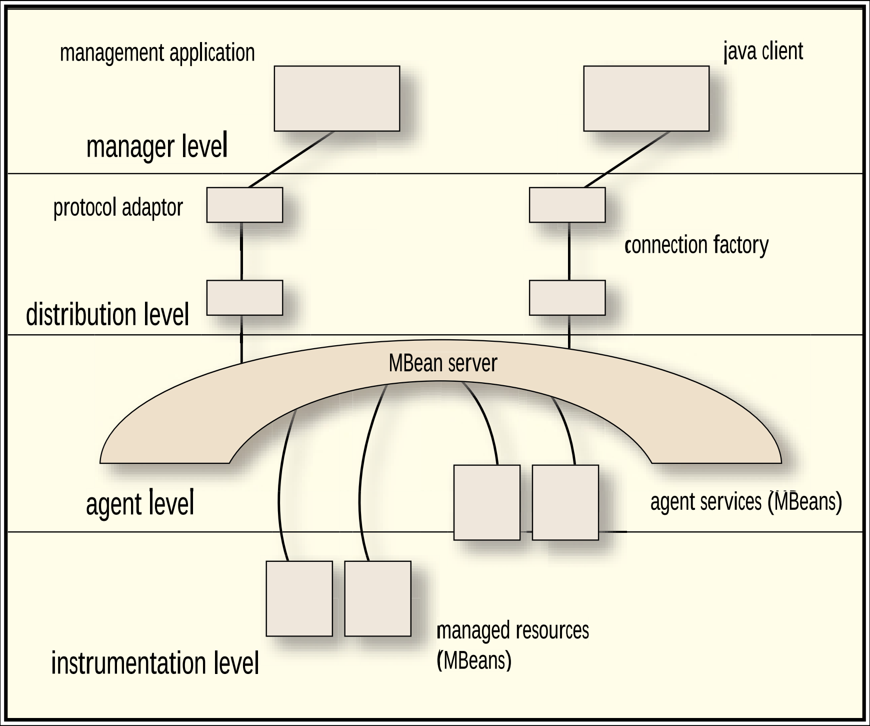
JMX's instrumentation level provides a variety of Managed Bean types for developers to implement depending on their requirements. It is important to note that MBeans bear no resemblance to Enterprise JavaBeans (EJBs). An MBean is like a regular JavaBean, having a well defined interface, complete with getters and setters.

TABLE 1

Architecture Level	Description	Key components
Instrumentation Level	Application components that provide a management interface and act as JMX manageable resources.	Managed Beans (MBeans), Notification classes and interfaces, metadata classes
Agent Level	Direct access to resources through their management interfaces. Implements an MBean Server for exposing the resources, and provides a number of generic services, implemented as MBeans.	MBean Server, MBean Server Factory, MBeans, Notification classes and interfaces, Metadata classes, MBean Monitors, Services
Distribution Level	Provides interfaces for JMX management applications to use in order to access the agent level and MBeans that have been registered with the agent level.	Protocol adapters, RMI, and CORBA interfaces for the agent layer
Manager Level	Applications that allow for the monitoring and administration of managed components. These may be tools that are specifically designed for a given type of application, or may provide a more generic interface.	Management applications

JMX architecture levels

FIGURE 1



JMX architecture levels

Unlike EJBs, they have no built-in transactional security or distribution facilities.

A *Standard MBean*, the most basic MBean, is not required to implement a special JMX interface or extend a particular abstract class. To be considered a Standard MBean, a class must simply implement an interface whose name matches that of the class with the additional MBean suffix. The MBean interface defines the management interface for the corresponding MBean class. A simple MyService MBean is shown in Listing 1 (code for this article may be found at www.wldj.com/sourcec.cfm).

Making the MyService MBean available within a JMX agent is straightforward. The examples in Listings 2 and 3 are based on the standard JMX interfaces and can be used with Sun's reference implementation.

Registration of an MBean with the agent layer is carried out using the MBeanServerFactory, MBeanServer, and ObjectName classes, all part of the standard JMX API, found in the javax.management package. The MBeanServer interface provides a rich API for registering and manipulating MBeans; essentially, all management of MBeans is carried out using this interface.

The MBeanServerFactory is a class that provides a broker for the various MBeanServer instances within a system. It is an implementation-neutral way of obtaining references to an MBeanServer. The API provided by the MBeanServerFactory allows for multiple servers to be active within the system.

An MBean is registered under a particular name that is used to uniquely and meaningfully identify it. For most MBeanServer operations, this name is represented by an ObjectName instance. It is a structured piece of data that complies with the JMX naming conventions. MBean names are given by a domain name, followed by a colon, and then by a sequence of one or more property-value pairs. Certain MBeanServer APIs support pattern-matching for MBean names, allowing users to query for a range of MBeans whose names conform to a given object name expression.

As shown in Listing 2, the basic registration process is quite simple. However, additional JMX facilities provide hooks for developers during the registration process, if required. For example, the MBean can implement an additional MBean Registration interface that will give the MBean-event callbacks during registration. These callbacks (preRegsiter, postRegister, deregister, and postDeregister) allow the MBean to obtain a reference to the MBeanServer, allowing the MBean to perform management actions on other MBeans. MBeans are used by invoking their methods through the MBeanServer interface (see Listing 3).

INTERMEDIATE AND ADVANCED JMX USE

Standard MBeans are simple, as is the basic

Simplex Knowledge Company

www.skc.com

API for manipulating the MBeanServer. However, JMX also offers functionality that goes far beyond basic use. Instead of Standard MBeans, implementers of MBeans can choose to build Dynamic MBeans or Model MBeans, and may make use of the JMX Notification framework. The agent level also has other features, including a number of built-in MBean monitors, and services for loading and relating MBeans.

Dynamic MBeans

Dynamic MBeans allow the management API of an MBean to be defined dynamically instead of through a predefined interface. For all MBeans, the agent layer discovers the management interface and exposes it through various APIs. For Standard MBeans, this interface is discovered by inspection of the MBean interface. For Dynamic MBeans, the agent level discovers the management interface by invoking particular methods on the MBean (see Listing 4).

Dynamic MBeans generally require more coding than Standard MBeans, but allow the management interface to change at runtime, and allow the MBean developer to expose more information (such as textual descriptions of the MBean and its operations) through the agent level.

The surprising aspect of DynamicMBeans is that they present essentially the same client view to their users as Standard MBeans. A Dynamic MBean's attributes are displayed and its actions are invoked through MBeanServer methods in the same way as the much simpler Standard MBeans.

Model MBeans

Although a specialized form of Dynamic MBeans, Model MBeans are completely different in how they are built and configured. Essentially, a Model MBean is a generic Dynamic MBean that can be loaded with an object that represents the manageable resource. A JMX Provider is required to provide Model MBean implementations that developers can configure to wrap their resources without creating any MBeans themselves. As a result, Model MBeans require far less coding than Dynamic MBeans. To build a Model MBean, the developer must

- Create an empty Model MBean,
- Provide a managed resource delegate against which all operations will be executed,
- Provide an MBeanInfo object that lists the allowed operations for the MBean,
- Register the Model MBean with the MBeanServer.

JMX Notifications

The JMX Notification framework allows MBeans to notify management applications of changes of state or other events that have occurred in the manageable resources of the system. Various subclasses of javax.management.Notification (which extends java.util.EventObject) can be used by MBeans that implement javax.management.NotificationBroadcaster to signal events to objects that have implemented the javax.management.NotificationListener interface.

MBEAN MONITORS

Instead of relying on MBeans to notify interested parties of changes in their state, management applications can actively track MBean attributes through MBean monitors provided by the agent level. A JMX agent implementation will provide various MBean Monitors. These are used to detect changes in specific MBean attributes. Several monitor types are provided with a JMX agent, including

- ***CounterMonitor:*** Checks the value of attributes that act as counters (these are integer types and have nonnegative values)
- ***StringMonitor:*** Detects changes in string values
- ***GaugeMonitor:*** Observes fluctuations in integer or floating point attributes within a range.

JMX METADATA

One of the key things that JMX provides is a way for managed resources to describe themselves to management applications. This allows management applications to be written independently of the systems they administer, without knowledge of the operations and attributes of the manageable components they will manipulate. The description of manageable resources is carried out through the JMX metadata data classes – MBeanInfo, MBeanOperationInfo, MBeanConstructorInfo, MBeanAttributeInfo, and so forth.

Without a metadata infrastructure, management applications would have to rely on the Java reflection API for this discovery process. By providing this framework, JMX avoids the difficulties associated with reflection and provides a standard means of self-description that allows developers to provide more information than is available through reflection.

Roles in JMX

One of the main questions with JMX is not how to use it – its architecture and API are straightforward – but when and why to use it. The potential users of JMX technology are best described through a set of illustrative roles (see Table 2).

Java developers will likely find themselves in the roles of application developer, while building manageable applications, and management solutions developer, while creating ways to better manage existing applications. The application developer develops Java applications that provide a runtime management interface. J2EE applications built with technologies such as JSPs, servlets, EJB, JMS, and so on are likely to be the most common manageable applications. This

–continued from page 10

Now, you're at liberty to start a new one, if you like (in our example above it's this transaction that would be responsible for the audit-log writing) This new transaction is quite independent of the old one – it can commit or abort as required by its own business semantics – and after whatever logic it represents has completed, the old transaction can be resumed and carry on where it left off (provided it didn't time out behind your back, as previously noted).

Anyone For Tennis?

In fact, for the real rocket scientist, a transaction can be suspended in one thread and resumed in a different one, which means that your code can play "transaction tennis," passing transaction objects back and forth as it pleases. This is, however, a very bad idea. For maximum reuse, code should leave the transactional state as it found it. If you're writing a class that is making all kinds of assumptions about the state of the current transaction, then the chances of it being useful in another future situation are pretty limited. It's a basic rule of encapsulation that you shouldn't expose your inner workings through your interfaces, and

transactions fall firmly into the category of inner workings. By all means, write code that suspends the current transaction (if any) before performing operations that need to be self-contained, so long as the transaction is resumed again afterwards. Because transactions are omnipresent in a thread, you need to make sure you don't code any nasty surprises for yourself. As we all know, there is no more difficult bug to track than one that involves the operating environment changing unexpectedly. That, after all, is why the Java language excommunicated the concept of the pointer.

THE LARGEST INTERNATIONAL

WEB SERVICES CONFERENCE & EXPO IN THE WORLD!

web services EDGE conference&expo

JUNE 24-27
JACOB JAVITS CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE CONVENTION CENTER
SAN JOSE, CA

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING
TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!

WEB SERVICES
SKILLS, STRATEGY, AND VISION

REGISTER ONLINE TODAY
FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT WWW.SYS-CON.COM

Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at *Web Services Edge East 2002*.

Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting-edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Keynote panel that will debate and discuss the realities and promise of Web Services.

A Sampling of Web Services-Focused Sessions

- PRACTICAL EXPERIENCES WITH WEB SERVICES AND J2EE
- STATE OF THE WEB SERVICES INDUSTRY
- THE ODD COUPLE: MAKING .NET AND J2EE WORK TOGETHER
- EXPLORING THE .NET MY SERVICES INITIATIVE
- STANDARDS WATCH
- GUARDING THE CASTLE: SECURITY & WEB SERVICES

SEAN RHODY
CONFERENCE TECH CHAIR
WEB SERVICES TRACK CHAIR
EDITOR-IN-CHIEF
WEB SERVICES JOURNAL

Featuring…

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 *i*-TECHNOLOGY PROFESSIONALS

Who Should Attend…

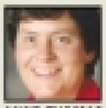
- DEVELOPERS, PROGRAMMERS, ENGINEERS
- *i*-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

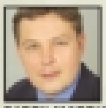
Hear these thought leaders in interactive, cutting-edge keynote addresses and panels…

JEAN FRANCOIS ABRAMATIC
SENIOR VP R&D, FORMER CHAIRMAN, W3C • ILOG

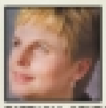
DAVE CHAPPELL
VP CHIEF TECHNOLOGY EVANGELIST
SONIC SOFTWARE

GREGG KIESSLING
CEO
SITRAKA


ANNE THOMAS MANES
CTO
SYSTINET


BARRY MORRIS
CEO
IONA

ERIC RUDDER
SENIOR VP DEVELOPER AND PLATFORM EVANGELISM
MICROSOFT


PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit information
CONTACT: MICHAEL PESICK
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645
201 802-3057
MICHAEL@SYS-CON.COM

SPONSORED BY:

IONA | END 2 ANYWHERE™
ADOS
bea
TogetherSoft
SilverStream
Actional
MERANT
Altaworks
PolarLake Enterprise Strength XML
ALTOVA xml spy

MEDIA SPONSORS

Federal Computer Week
Webservices.org
XML TIMES.com
Java Skyline
CE Advisor
WebServicesMall
XML-BAG
Wireless Business & Technology WIRE
JCert
wrox
OASIS
SD Times
JAVA Developer's Journal
wireless
XML-Journal
WebLogic
WebServices Journal
WebSphere
ColdFusion Developer's Journal

OWNED AND PRODUCED BY
SYS-CON MEDIA
SYS-CON EVENTS

role will use the instrumentation level of JMX.

Management solutions developers will create customized administration and monitoring systems that will function as components in the management layer of JMX. They will be accessing instrumented application components from the agent level through the distribution level. These developers may have to develop additional protocol adapters in the distribution level if they are integrating legacy management systems.

Management tool vendors may build management tools in Java, or may enhance existing tools to use JMX. This may include adapting existing management protocols to JMX.

For both the management solutions developer and the management tool vendor, the end-user of their work is the system administrator. This role uses these tools and solutions to manage the running applications – monitoring their behavior, running diagnostic checks, shutting down and starting up services, and doing manual load-balancing as required.

A standalone Java application may provide a complete JMX implementation, but the more common scenario is that the vendors of J2EE servers will also become JMX Providers. The JMX Provider provides an implementation of JMX either as a framework to be incorporated into standalone Java applications, or as part of an application server.

TABLE 2

Role	Description	JMX Technology used
Application Developer	Builds applications that include management capabilities.	• Uses instrumentation level, • Creates MBeans, • Accesses agent level to make MBeans available
Management Tool Vendor	Builds tools that can manage applications or integrate several applications behind a single management facade.	• Creates components in the manager level, • Accesses MBeans and agent layer through distribution level, • May also Create distribution level components
JMX Provider	Provides an environment for the deployment of managed applications. May provide JMX as part of a J2EE or other server environment.	• Provides the agent level, • May provide some or all of the distribution level, • May provide built-in management level.
Management Solutions Developer	Creates customized management solutions for individual applications or groups of applications.	• Creates components in the manager level, • Accesses MBeans and agent layer through distribution level, • May also create additional distribution-level components
System Administrator	Uses tools, utilities, and procedures provided by the Management Tool Vendor and the Management Solutions Developer.	• Should not access or be aware of JMX APIs directly, • Should use components provided by the manager level

Roles in JMX

JMX Use in J2EE

The inclusion of JMX in J2EE-based application servers will enable developers to:

- Provide application-specific management interfaces that can be used by in-house and third-party management tools
- Leverage existing application-server instrumentation to create custom reporting and monitoring applications
- Integrate third-party management tools into their J2EE applications.

Perhaps the most significant use of JMX technology by application developers will be in writing interfaces and reporting tools that make use of the instrumentation already provided by the application server. In other words, it is likely that developers will develop more applications at the JMX management layer than building MBeans themselves.

Ideally, integrating JMX into J2EE application servers will make the configuration and administration facilities of such servers extensible, allowing developers to create simplified administration consoles, integrate reporting facilities and statistics-gathering into the server, and tie server and application events to logging facilities or messaging channels.

The more instrumentation provided by the application server, the less need there will be to build JMX interfaces for individual J2EE application components. Instrumentation provided by the application server will be safer than anything built into individual components, and will provide a consistent management view across all deployed applications. Standard J2EE management facilities may include:

- Deployment, undeployment, and redeployment of components
- Server availability and cluster administration
- Security management
- Error logging
- Connector monitors (i.e,. JDBC connection tracking)

Clearly, developers will rely on the J2EE platform to supply more than just a JMX implementation. The platform will use JMX to provide access to built-in administration, management, and monitoring facilities, and ensure that applications deployed on the platform are preinstrumented and ready to be managed through app-

Hewlett-Packard

www.hp.com/go/infrastructure

server or third-party tools. WebLogic JMX does precisely this, offering a comprehensive set of prebuilt MBeans that provide access to all administration and monitoring features.

JMX Pitfalls

Overuse of JMX is a pitfall that should be avoided. It is not designed to be a general-purpose distributed object system. JMX use should be limited to those aspects of the system that are clearly management-related. Some other areas in which JMX users should proceed with care include:

1. ***Mixing JMX directly with EJB:*** Care must be taken to ensure that JMX code does not violate the security and transactional integrity of the EJBs. An EJB cannot also be an MBean for several reasons:
   - The EJB life cycle, which allows beans to be pooled, change identity, and be destroyed, does not match that of MBeans, which are treated like normal objects living within the virtual machine;
   - If an EJB acts as an MBean, the agent layer would have direct access to the bean-instance, bypassing the component interface and violating the transaction-and security-checking code provided by the container;
   - The EJB deployment, creation, and querying protocols do not match those provided by JMX.

   EJB components should not also be MBeans, nor should they expose their underlying bean instance directly to MBeans. It is possible, however, for MBeans to access EJBs through the standard EJB protocols. Unfortunately, MBeans would not be able to access EJBs through the new local interfaces provided by EJB 2.0 since these are restricted to use by other enterprise beans. Consequently, MBeans would have to access EJBs as remote clients, and the operations available to the MBeans would be the same as those exposed through the regular client interfaces.

“JMX is intended to be simple enough to allow developers to rapidly provide a management layer for either new or existing Java applications, and comprehensive enough to be adapted to other standard (non-Java) management protocols”

2. ***Distributed JMX use:*** Lack of details for distribution in the specification puts developers who are trying to create application server–neutral management solutions at a disadvantage. MBeanHomes make up for this by providing a straightforward distribution mechanism.
3. ***Relying on Standard J2EE management features:*** There is no official statement on what management facilities should be present in an application server. The existing management and administration facilities in WebLogic are a good example of the direction in which these specifications are likely to proceed.

Conclusions

JMX provides an incredibly simple and flexible way to make Java applications manageable, and a standard way to build powerful management applications. J2EE application servers have reached a level of maturity where management facilities are a requirement. As these two technologies evolve, developers of J2EE applications will see an increased need to make their applications JMX aware, and will see more JMX-based management tools appear in the application server marketplace. Although JMX is based on well proven management concepts and products, its use in application servers is in its infancy but as with all enterprise Java technology, it is bound to grow up quickly.

In part two of this series, we will look at WebLogic's JMX implementation, and discuss some concrete ways that it can be used to instrument J2EE applications.

References

- Sun Microsystems (July 2000). “Java Management Extensions (JMX) Instrumentation and Agent Specification (JSR 3).” http://jcp.org/jsr/detail/3.jsp.
- Sun Microsystems. “J2EE Management (JSR 77)”. http://jcp.org/jsr/detail/077.jsp.
- Sun Microsystems. “CORBA Adapter for JMX (JSR 70).” http://jcp.org/jsr/detail/070.jsp.
- Sun Microsystems. “WBEM Services Specification (JSR 48).” http://jcp.org/jsr/detail/48.jsp
- Sun Microsystems (March 2001). “Java Platform Enterprise Edition Specification, v1.3 (proposed final draft 3).” http://jcp.org/jsr/detail/3.jsp.
- Sun Microsystems. “IIOP Protocol Adapter for JMX Specification (JSR 70).” http://jcp.org/jsr/detail/070.jsp.
- Sun Microsystems (2001). “Java Platform Enterprise Edition Specification, v1.3 (proposed final draft 3).” http://jcp.org/jsr/detail/3.jsp.
- Sun Microsystems (April 2001). “Enterprise Java Beans Specification v 2.0 (public final draft 2).” http://java.sun.com/products/ejb/
- Sun Microsystems (June 1999). Java Management Extensions White Paper: “Dynamic Management for the Service Age; Revision 01.” http://java.sun.com/products/JavaManagement/doc.html
- BEA Systems (2001). “Using WebLogic Server JMX Services.” http://e-docs.bea.com/wls/docs61/jmx/index.html.

Altaworks

www.altaworks.com

Why Choose a CMP Architecture?

FIND STRENGTHS IN PERFORMANCE AND REDUCTION IN DEVELOPMENT BY LEVERAGING A HIGH-QUALITY CMP CONTAINER

BY TYLER JEWELL

Last month, I talked about the power of CMP entity EJBs and provided a number of scenarios where leveraging the CMP model would be preferable to developing stateless session EJBs that use JDBC or JDO.

AUTHOR BIO...

Tyler Jewell is BEA's director of technology evangelism and the coauthor of *Mastering Enterprise JavaBeans 2.0* and *Professional Java Server Programming (J2EE 1.3).* He writes widely on the Web on the subject of both J2EE and Web services.

CONTACT...

tyler@bea.com

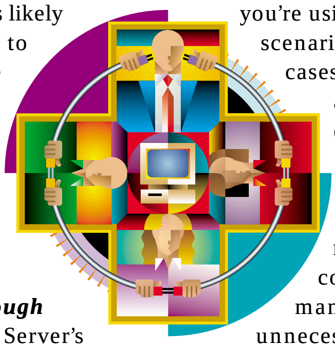
This month, I'll talk about the reasons for using a CMP architecture over a BMP one in entity EJBs.

Reasons to Use BMP with Entity EJBs

First, let's talk about the scenarios where BMP is appropriate for use in an entity EJB system. The biggest reason to use BMP over CMP is because what you want to accomplish cannot be done through BMP:

- ***Fields are accessed through stored procedures:*** If you work in an enterprise that regulates data access through stored procedures, a CMP engine won't know how to interact with the appropriate stored procedures. The entity EJB life cycle is very strict and it's likely that any stored procedure access to a database doesn't follow this life cycle. Also, even if the stored procedures followed the entity EJB life cycle, there would be no standard way of telling a CMP container which stored procedures to invoke during different points in the lifecycle.
- ***Persistent store is accessed through alternative methods:*** WebLogic Server's CMP engine requires the use of a JDBC driver to access the persistent store. If you want to use an alternative access technique such as JDO or a J2EE CA adapter, you need to use a BMP model.
- ***The bean's model requires data from multiple stores:*** WebLogic Server's CMP engine can only pull data from a single table. WebLogic Server 7.0 will support multi-table CMP access, but what do you do if the data in your bean model is from multiple databases? Although there are a number of significant performance reasons for not having a single entity EJB represent data from multiple stores, there are some scenarios where it's a requirement (especially when you try to integrate a legacy system with a new implementation). For these scenarios, a BMP bean should be used.
- ***The database requires nonstandard SQL:*** Databases differentiate themselves by adding proprietary extensions that enable better performance and robust functionality. For example, most databases offer a technique for doing automatic primary key generation; others offer specialized ways of doing optimistic locking at the database level. To take advantage of these features, the SQL submitted to a database usually has to be customized with proprietary extensions defined by the database vendor. CMP engines can differentiate themselves by supporting a wide range of these custom extensions. WebLogic Server 6.1 supports most primary key generation techniques and Oracle's serializable SQL extensions, but if your database requires further optimized SQL, you may need to use a BMP instead.

Why CMP?



BMP entity EJBs won't perform very well. If you're using a BMP EJB, chances are your scenario matches one of the oddball cases listed above. If you choose to use a BMP entity EJB when a CMP engine or a stateless session EJB with JDBC could be used, you're taking an unnecessary performance hit. A BMP entity EJB doesn't come without the overhead of container management, server management, and (sometimes) unnecessary life cycle migration. The pur-

pose of using a BMP is:

1. To take advantage of representing persistent data in a business object model in your system
2. To leverage the EJB deployment model, allowing for behavioral changes to be made by modifying XML, not code
3. Your data access falls into one of the scenarios listed in the first section.

But, if what you're trying to accomplish doesn't fit into one of these scenarios, and it is possible for a CMP container engine to functionally accomplish the data access you require, you should definitely be using CMP. There are a number of concrete reasons for using CMP:

- ***BMP entity EJBs can't take advantage of in-server locking models:*** These include pessimistic and optimistic in-server locking models. There are very few containers that support an in-server optimistic locking model, and it can be very powerful. Optimistic locking delays attempts to lock the underlying data in a database until the transaction begins the commit process. This is difficult to implement because the container needs to be intelligent and aware of any data modifications from the time the transaction started and the time the container places a lock on the data. If the data was modified in that time span by another process, this is called a collision, and the container needs to resolve it by rolling back the transaction or overwriting the data. Containers that support optimistic locking provide a more scalable implementation because data will be locked less frequently and the application server won't spend as many cycles blocking access to locked data.

  By having a container supporting pessimistic and optimistic locking models, a bean developer is freed from dealing with multi-request transactional issues, collision handling, and specialized SQL needed to make these locking scenarios occur. BMP entity EJBs cannot have a container implement these policies since it requires interaction with the underlying persistent store.

- ***BMP entity EJBs do not have EJB-QL support:*** EJB-QL is used to define find() methods and ejbSelect() methods. It allows a developer to quickly configure the way an entity EJB behaves without developing complicated JDBC or SQL code. Vendors, including WebLogic, may add enhancements to EJB-QL to allow developers to continually create more complicated queries. By dong this, the need for using optimized stored procedures diminishes and business logic moves further into the application server tier (which is where it belongs). Additionally, EJB-QL reduces the amount of code in an entity EJB, making it more maintainable, easier to support, and quicker to develop. BMP entity EJBs require find() methods to be implemented manually and, for the reasons listed above, should be avoided.

- ***BMP entity EJBs do not have any support for relationships:*** Your data has relationships with other data. It's a fact of life. If your data has relationships, then your object models in memory will also have relationships. Implementing relationships manually is very challenging. They can be directional and have multiplicity constraints applied to them. Also, many systems don't perform optimally because of the poor implementation of relationships. For example, if you have a one-to-many relationship of objects that are persistently stored, you might implement an aggressive loading model that brings a parent and all of its children into memory every time the parent is accessed. If the parent has 10,000 children, but those children objects are not needed on a regular basis, the system is being burdened with activities that could otherwise be avoided.

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altaworks	www.altaworks.com	603-598-2582	31
BEA	www.developer.bea.com	408-570-8000	2, 3
EnginData	www.engindata.com		36
Hewlett-Packard	www.hp.com/go/infrastructure	800-HPASKME ext. 246	29
Intel	www.intel.com/ebusiness	408-765-8080	11
JDJ Edge	www.sys-con.com	201-802-3069	55
Mongoose Technology	www.mongoosetech.com	281-461-0099	60
Performant	www.performant.com	866-773-6268	37
Precise	www.precise.com/wldj	800-310-4777	9
ReportMill	www.reportmill.com	214-513-1636	21
SilverStream	www.silverstream.com	888-823-9700	51
Simplex Knowledge Company	www.skc.com	845-620-3700	25, 57
Sitraka	www.sitraka.com/jclass/wldj	416-594-1026	7, 15
Thought, Inc.	www.thoughtinc.com	415-836-9199	19
WebLogic Developer's Journal	www.sys-con.com	800-513-7111	35
WebGain	www.webgain.com	888-822-3409	59
Web Services Edge	www.sys-con.com	201-802-3069	27
Web Services Edge World Tour	www.sys-con.com	201-802-3069	46, 47
Web Services Journal	www.wsj2.com	800-513-7111	33, 41
Web Services Resource CD	www.jdjstore.com	201-802-3012	42, 43
Wily Technology	www.wilytech.com	888-GET-WILY	4
XML Edge Conference	www.sys-con.com	201-802-3069	45
XML Journal	www.sys-con.com	800-513-7111	34

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.

NeXT MONTH

Focus: Migration

Migrating Sun J2EE Blueprints to WLS
Porting your applications to WLS 7.0

Exploring WebLogic JMX: JMS and J2EE, pt. 2
Concrete ways to instrument J2EE applications

Migrating to WebLogic Server 7.0
Moving up from versions 5.1 and 6.1

Plus

Intro to WebLogic 6.0 Certification
The first in a series of articles on preparing for the exam

Tips 'N Tricks
Mika Rinne looks at SQLJ

BEA WebLogic DEVELOPER'S JOURNAL

BMP entity EJB containers cannot support relationships because a container needs access to the underlying relationship management technology being used in the persistent store. In the case of relational databases, a container would need to access the primary and foreign key columns to properly manage the relationship. Using BMP, all persistent store access is managed by the bean itself, so the bean needs to manage the relationship manually. Since BMP relationships are nonstandard and very difficult to implement, you run the risk of building a poor-performing system. Also, WebLogic Server's CMP implementation provides aggressive and lazy loading of relationships that would be nearly impossible to implement using BMP.

"If the stored procedures followed the entity EJB life cycle, there would be no standard way of telling a CMP container"

- ***BMP cannot avoid the n+1 problem:*** One of the problems that plagued CMP engines with EJB 1.1 was the n+1 problem. This was caused by containers that required n+1 database queries and network invocations to load entity EJBs in a parent-child relationship. For example, if you had an Order entity EJB with 1,000 LineItem children entity EJBs in a one-to-many relationship, EJB 1.1 CMP engines would require 1,001 database queries to load all of the EJBs. This is crazy, since it's likely that all of the data was stored in only two tables.

  EJB 2.0 solved this problem for CMP engines by requiring entity EJBs in a relationship to use local interfaces. The container could be guaranteed that entity EJBs in a relationship were going to exist in the same virtual machine, and if a vendor required relationships to be in the same database, the CMP engine could continually optimize the n+1 scenario to be a 1+1 database hit scenario. This isn't the case for BMP containers. Since the container does not create the query, there is no way to "aggregate" all of the information in a standard way to pull in all of the child data all at once. The BMP engine will always have to iterate each of the children, forcing the n+1 problem to remain. One reason for using BMP in a relationship, however, is if you need a relationship that operates over remote interfaces. If the local interface restriction cannot be adhered to, using BMP may be an option.

EnginData Presents

2002 Developer Market Survey Reports

$2,995
Java Developer Market Survey 2002
$2,495
Web Services/XML Developer Market Survey 2002
$1,995
Wireless Developer Market Survey 2002

Our comprehensive reports offer insight and strategy to guide your most critical business decisions in today's fastest growing technologies...

✓ Establish your product and marketing strategy

✓ Understand your customer's needs

✓ Evaluate Technology & Trends

Preview and order reports at www.engindata.com

engindata.com

engindata RESEARCH

- ***BMP cannot do cascading deletes:*** For EJBs in a relationship, a CMP container can perform a cascading delete. This means that if a parent object is destroyed, the child objects in the relationship are automatically destroyed as well. Cascading deletes can be done by manually deleting each EJB one at a time or by using specialized database referential integrity mechanisms that automatically delete all of the data reachable through foreign keys when a row is deleted. The latter is almost always preferred for performance reasons.

  Since BMP entity EJB containers do not have access to the underlying data store, the EJB cannot take advantage of optimized database techniques for deleting a massive amount of data. This can have a huge performance and portability impact.

- ***BMP containers cannot do automatic primary key and table generation:*** One of the features in WebLogic Server's container implementation is the ability to create tables, set up relationships, and generate database-generated primary keys automatically. By having the container automatically create the appropriate tables and relationships when they don't exist, development time is shortened since a time-consuming mapping of the EJB to the database doesn't have to occur. Additionally, most databases offer the ability to automatically generate a universally unique primary key value. Entity EJBs can leverage the database-generated primary key values if the container can query the database and pull that information into the server.

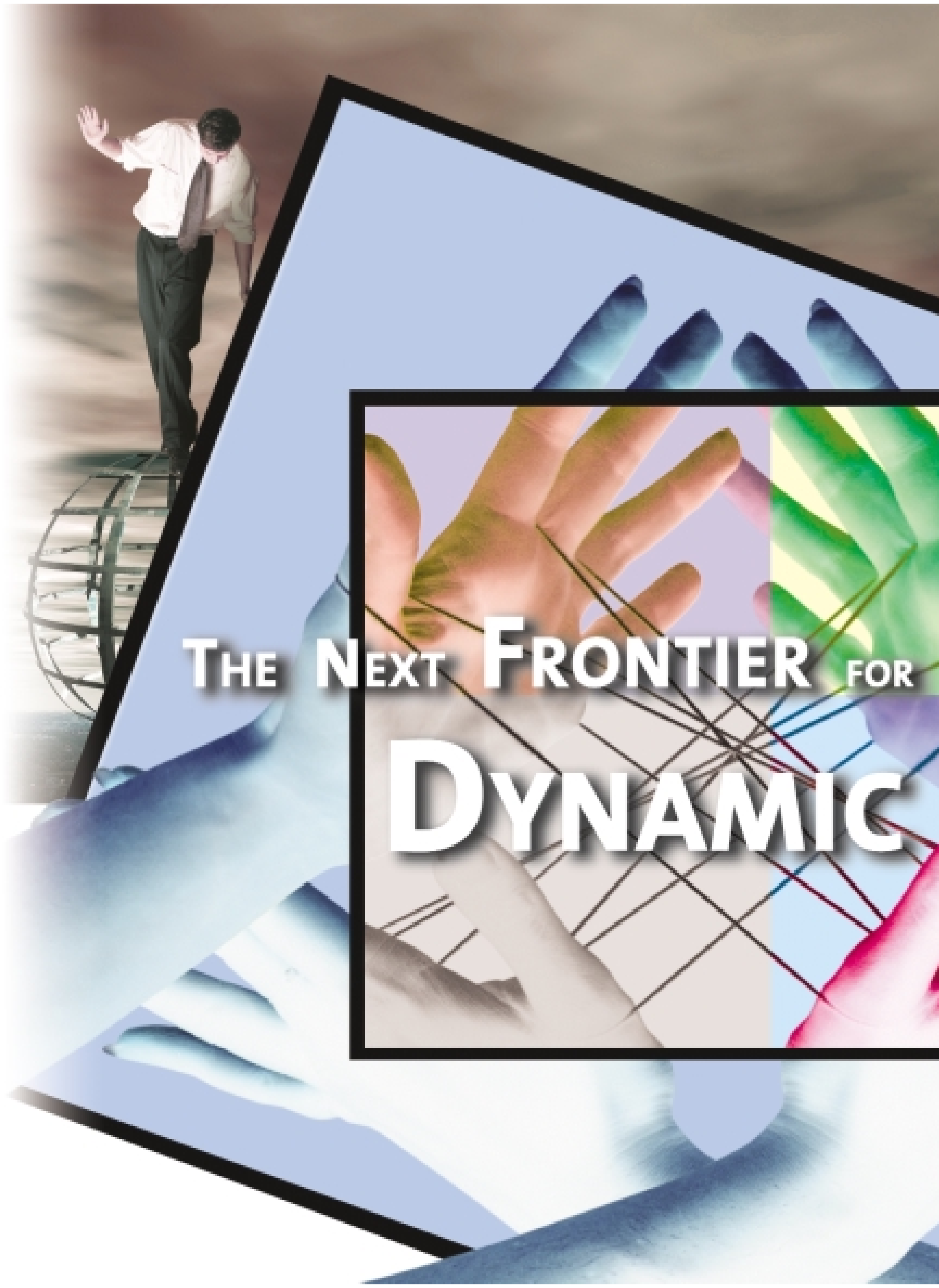
  It's impossible for a BMP container to create database tables or set up relationships in any way. Additionally, primary key generation techniques are proprietary based on the database being used, and cannot be done portably using a BMP entity EJB.

Conclusion

CMP entity EJBs are not used enough in production systems and BMP implementations are used too frequently for the wrong reasons. When you implement your next project, consider the strengths in performance and reduction in development that can be added by leveraging a high-quality CMP container as opposed to developing data access logic using a BMP architecture. I guarantee you'll be happy with the results.

Performant

www.performant.com



The Next Frontier for Dynamic

BY
STEVE CHAZIN

AUTHOR BIO...

Steve Chazin, director of product marketing at Bowstreet, Inc., is responsible for the worldwide marketing of software products based on the Bowstreet Business Web Factory - a system that automates the creation and maintenance of complex Web applications. Before joining Bowstreet Steve worked for Apple Computer and Raytheon.

CONTACT...

schazin@bowstreet.com

To understand the future of software, we need to understand the past, because once again we're seeing that history is repeating itself. This article looks at how the next frontier of software – the dynamic assembly of Web services into a new type of composite Web application – is following closely in the footsteps of the Industrial Revolution.

Web Services: Assembly

The Artisan & Cottage Industries

The year is 1386. You're on pins and needles today. For almost three years you've been waiting for Merac, the village's master craftsman, to finish the ceremonial coat of arms he's been forging for your only son's wedding. As one of the most highly decorated knights in feudal France and a descendant of a bloodline dating back nine generations, you have the resources and wealth to pass on such a masterpiece to Philip. The only thing that can ruin this day is the weather near your castle outside Dordogne. It's raining. Again.

You finally lay your eyes on the handicraft – and your heart jumps. It is exactly as you imagined it; perhaps even more superb in person than what you described to Merac so long ago. You remember why you selected this artisan in the first place: he has painstakingly transformed your vision into a unique work of art. Philip will be delighted – and of course the townspeople will admire it for generations.

In 1386, stories like this abound. Craftsmen labor for their lifetime, producing superbly crafted objects for a few privileged lords and landowners. Apprentices learn the tricks of the trade for many years before developing the skills they need to make a livelihood on their own. The quality of workmanship is superb and each object is a highly customized, handcrafted masterpiece – no two are ever alike. But because so few objects can be handmade to this level of perfection, the cost restricts ownership to only a few.

This system remained essentially unchanged until the 17th century, when England's economy was based on cottage industries. Workers would purchase raw goods from a marketplace and individually produce finished goods at their homes, or cottages. The Cottage industries still suffered from low worker productivity and a snail's pace of production. Goods were still high in price and exclusive only to the wealthiest people.

Ultimately, the need for efficiency and speed forced this system to change. In 1733, the demand for cotton cloth soared, but limited production capacity threatened England's economic might. The flying shuttle, invented by a weaver named John Kay, cut weaving time in half. Though not readily accepted at first (many inventions of his time were deliberately destroyed by the people who thought they were going to be displaced by machines), new inventions such as the spinning jenny and the spinning mule, the water-powered frame, the power loom, and the cotton gin all improved the manufacture of cotton goods by speeding up the process. The factory system of mass production made formerly expensive, handmade items (such as shoes and gloves) more affordable to lowe-class and less wealthy people – many of whom now earned their wages working in other factories. The quality and productivity of worker's lives improved dramatically.

Interchangeable Parts and the Assembly Line

In 1907, Henry Ford announced that the Ford Motor Company's goal was to create "a motor car for the great multitude." Before Ford's time, automobiles were expensive for the same reason that Lord Dordogne's wedding gift was – the car was custom-made by craftspeople skilled in producing just one part while other highly trained people assembled the parts by hand into the finished product. Ford looked for more efficient ways to produce his car in order to lower its price and make it more available to the masses.

His first principle was interchangeable parts. Ford required that each individual car part be made the same way every time. Now any valve would be able to fit any engine, any steering column would fit any chassis. Eventually, by producing a common specification for each part, other companies could develop parts that would instantly work between and within the assemblies of others. Ford improved the machinery and tools used to make these interchangeable parts so a lower skilled worker could operate them, replacing the skilled craftsperson that formerly painstakingly made each part by hand, providing jobs to legions – and soon unions – of workers.

Another Ford idea came together in Michigan in 1913 when the first moving assembly line for large-scale manufacturing was introduced. With this assembly process – inspired by grain-mill conveyor belts and the meat-packing houses of Chicago that Ford had visited – work moved to the workers rather than the other way around. Ford trained each worker to manually perform just one of the 84 steps required to assemble the Model T and arranged the work so that as one task was finished, another began with minimum delay.

These innovations allowed Ford to produce cars at a much higher rate than his competitors – which led to lower prices and higher profits through increased capacity. For the first time, complex products such as automobiles could be produced with good quality, in a much shorter amount of time and for a reasonable price. The only major drawback to Ford's process was his "one-size-fits-all" approach. Ford joked that his customers could have their Model T in "any color they wanted, as long as it was black." Amazingly, the Model T – first built in 1908 – kept the original design until the last one – the 15 millionth - rolled off the assembly line nearly 20 years later.

Dynamic Assembly Brings 'Just-in-Time' Mass-Customized Production

It wasn't until the latter part of the 20th century that mass customization – factory-built yet custom-made products – was eventually achieved. It started with a new invention Ford could never have imagined; computer-controlled robots operating on dynamic assembly lines. These robots could be programmed to perform highly repetitive or dangerous tasks with much greater precision than the humans they replaced, and to do so around the clock.

TABLE 1

	Quality	Cost	Customization	Speed
Artisans & Cottage Industry	🟩	🟥	🟩	🟥
Interchangeable Parts	🟨	🟨	🟥	🟨
Just in Time Dynamic Assembly	🟩	🟩	🟩	🟩

Dynamic assembly reduces costs and increases quality, speed, and customization

Technology companies like Cisco, Dell, and Apple now build many of their most profitable products only when a customer requests it. Such requests are received electronically – invariably at the firm's Internet e-commerce site – culminating in an electronic command that is dispatched to a dynamic assembly production line to build each unique product. "Turns" of inventory are now measured in minutes instead of weeks or months, and thousands of variations of high-quality products are the norm. Dynamic assembly of products on demand promises to eliminate overproduction of products that sit unsold on shelves, further reducing the cost of goods. So today's modern factories are a natural evolution of Ford's factories where "just-in-time" production techniques and fully automated factory floors dynamically assemble individual component parts into a multitude of customized products on demand.

An important byproduct of this dynamic assembly approach has recently emerged; complex products such as aircrafts are now designed and tested as digital prototypes entirely in the memory circuits of a computer. Each part – from the fuselage to the bolts that hold down each seat – is precisely designed, modified, tested and redesigned to near-perfection before any actual building occurs. The optimized digital information is finally transmitted to computerized plants for manufacturing and dynamic assembly – where the finished product exists for the first time in the physical world. Today's products are designed and built using software tools that act much like spreadsheets in the financial world. "What-if" analysis, thermal and stress analysis, and cost/weight/safety trade-offs are all done automatically by propagating even minor changes across thousands of related drawings. "Boutique" production runs are now possible in less than 24 hours.

But What Does All This Have to Do With Software?

Software has been silently following the same progression as the world of physical products. Like the artisans of old, software craftsmen would labor for years – sometimes in teams, sometimes alone – to create a single piece of software, their own work of art. Craftsmen like Dan Bricklin (VisiCalc), or Steve Jobs (MacOS) or Mark Andreeson (Mosaic/Netscape) would spend significant portions of their lives creating and then improving upon their software masterpieces. The search for the "killer app" became the preoccupation of users and investors. Software prowess was measured in millions of lines of code and the number years between major releases. An industry sprung up overnight around the need to produce better hand tools for developers – or to fill the holes in other software products. New companies emerged to produce compilers, spell checkers, debuggers, virus protection, and security products to name a few. This cottage industry still exists today.

Yet for more than 20 years we've quietly been in the second phase of the software industrial revolution where object-oriented programming and interchangeable parts – objects, "beans", components, and libraries with standardized interfaces – have become the norm. In many ways this trend is analogous to the concept of inventories of "off-the-shelf" interchangeable parts that people could use to build higher order assemblies. Like the experts focused on just one step in the assembly of the Model T, teams of programmers divide the application development workload, focus on their own competencies and manually assemble a variety of products from a known set of parts. But like those same factories of old, the existence of a huge inventory of parts doesn't solve the problem of building customized versions of each application for the needs of an ever more disparate and demanding audience. The lack of an automated, mass-production process has made economy of scale for Web applications an elusive goal.

Today's Challenges

So today's developers are swamped with demands to build a new type of application – using yesterday's tools. They need to build multiple, complex Web applications that provide access to business processes and data contained in various backend systems. In many cases developers have already deployed dozens of Web applications and now find they are unable to customize new application variations for various audiences and channels. Keeping up with the rampant demands of continuous change in business requirements is a constant battle. Suddenly, the need for efficiency and speed is forcing traditional approaches to Web application development, deployment and maintenance to change.

The Future of Web Services: Dynamic Assembly

So the stage is set for software's third phase, the dynamic assembly of complex applications on demand. The recent emergence of Web services – Web-accessible software applications with standardized or self-describing interfaces – brings new possibilities and new complexities to assembling Web applications from an ever-increasing supply of disparate component parts.

The need to assemble pieces of an application at runtime isn't new; the notion of breaking applications into pieces that get loaded together at runtime is common. WebLogic Application Server performs such tasks on EJBs and loosely coupled Web services in its runtime execution environment. But what is now required is an ability to make choices during the assembly process about which components should be used and how to customize the application's structure, functionality, behavior, and content out of such components. How can this feat be accomplished?

Developers have used many combinations of explicit instructions coupled with rules engines to build the appropriate assembly instructions for different use cases. But because developers have to manually construct all the variations of the rules, templates, and other pieces that drive the dynamic assembly process, they quickly get bogged down in maintaining a morass of hard-wired connections, if-then-else statements, and other explicit code-glue to make it all work.

The introduction of programmable robots on the modern assembly line enabled mass customization and incredibly short design-to-manufacture cycle times. Software factories, which mirror their physical world counterparts, have recently emerged that capture the application design process into discrete steps and use software robots to dynamically assemble components and Web services into highly customized composite Web applications on demand.

Bowstreet's aptly named Business Web Factory is a Web services development and assembly platform that automates the creation and maintenance of complex Web applications. Patent-pending robotic software agents called Builders enable programming tasks to be captured in software (for instance, generating the code that calls a SOAP/WSDL Web service). At runtime, specialized instructions invoke a sequence of Builders, who perform their various construction tasks during the "regeneration" of the Web application. Since that of the execution of the Builders during regeneration can be varied (much like computerized robots on a modern factory floor) through parametric inputs contained in profiles, this approach automatically generates a wide range of different Web application variations.

Business Web Factory routinely generates hundreds or even thousands of application variations, resulting in significant timesaving over successive development projects, and allows business users or the end users themselves to "order" new applications to be built on demand simply by creating new profiles. Like a modern factory, this approach reduces the need to maintain legions of separate applications (products sitting unsold on shelves), builds a call to a Web service (parts) on demand, and eliminates repetitive or boring programming tasks (robots) that take up too much development time and defocus the programmer from more important, revenue-producing development tasks.

This approach also inherits some of the same important benefits we saw in the CAD/CAM systems that drive today's modern factories. A single change can ripple instantly through tens, hundreds, or even thousands of applications – and because there aren't separate applications to constantly change, update and fix, the time and cost of application maintenance is greatly reduced. The developer is thus relieved of tedious, repetitive work associated with maintaining separate code bases while eliminating expensive regression testing. And for the first time, business people can control applications and respond to changes quickly and independently, allowing them to concentrate on creating competitive advantage for their firms, rather than helping IT manage a sprawling Web application franchise. "Boutique" production runs of highly customized and/or specialized Web applications can be done instantly.

So from the exquisite, expensive, one-of-a-kind products laboriously produced by an artisan to today's highly customized, high-quality products dynamically assembled at low cost and high speed (Figure 1), the software industry has closely followed the trend of the Industrial Revolution. We fully expect that history will record that the solution to the enormous cost and complexity of application development, deployment, and maintenance will be seen as having started with the dynamic assembly of Web services.

WebServices JOURNAL .NET J2EE XML

XML JOURNAL

THE FIRST & ONLY WEB SERVICES RESOURCE CD!

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN 400 EXCLUSIVE WEB SERVICES & XML ARTICLES

EDITED BY SEAN RHODY

$119 CD VALUE
FROM WEB SERVICES JOURNAL
web services EDGE conference & expo
$100 coupon inside

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

CD is edited by well-known WSJ Editor-in-Chief Sean Rhody and organized into more than 40 chapters containing more than 400 exclusive WSJ & XML-J articles.

Easy-to-navigate HTML format!

Bonus:

Full .PDF versions of every WSJ & XML-J published since the first issue

XML in Transit	XML Industry Insider	XML & Databases	UML
XML B2B	<e-BizML>	Electronic Data Interchange	Integration
Java & XML	XML & Business	Ubiquitous Computing	WSDL
The XML Files	XML Demystified	Information Management	Beginning Web Services
XML & WML	XML & E-Commerce	Objects & XML	Web Services Tips & Techiques
Voice XML	XML Middleware	XML Pros & Cons	Frameworks
SYS-CON Radio	XML Modeling	Java Servlets	Management
XML & XSLT	CORBA & XML	XML Filter	Security
XML & XSL	Data Transition		UDDI
XML & XHTML	XML @ Work		.NET
2B or Not 2B			
XML Script			

3 YEARS
25 ISSUES
400 ARTICLES
ONE CD

WebServices
XML JOURNAL
THE FIRST & ONLY WEB SERVICES RESOURCE CD!
WEB SERVICES RESOURCE CD
THE SECRETS OF THE WEB SERVICES MASTERS
INCLUDES EXCLUSIVE .NET ARTICLES
MORE THAN 400 EXCLUSIVE WEB SERVICES & XML ARTICLES
$119 CD VALUE
EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

Special Limited Time Price

Now Shipping

$79 + S&H

ONLINE ORDER AT JDJSTORE.COM

SAVE $40

www.JDJSTORE.com

OFFER EXPIRES JUNE 30, 2002

BEA eWorld 2002 Update

BY **JASON WESTRA**

The BEA eWorld conference was, in many ways, the same as every other conference I've attended. In other ways, it was quite different. The conference was held in the San Diego Convention Center, California, February 23-27, 2002. When I arrived, the hotel manager asked if I'd take a smoking room. "No," I replied. "What's changed?" I thought. But hey, this was a subtle sign that the conference was packed (actually, BEA eWorld recorded over 2000 attendees)!

The BEA eWorld conference had all the classics: t-shirts, conference bags, morning and afternoon keynotes from sponsors and BEA visionaries, morning and afternoon sessions that were both business and technical in nature, and the exhibit hall where more than 114 vendors touted their wares to interested and freeloading bystanders. Freeloaders, you know who you are! For those of you who could not attend or who just want to recall a great time, I've summarized the conference news, highlights, and information here just for you.

Keynote

Even after the dot-com crash, it's still cool to be on the edge of technology. I feel this way when working with BEA products. At the opening keynote, I felt it as well. On Monday morning, attendees flooded like ants into the hall amidst flashes from cameras, and techno music for the keynote address from the CEO of BEA, Alfred Chuang. Alfred made his appearance dressed in leather and riding an Italian chopper onto the stage – quite an entrance!

After dismounting and losing the leather, Alfred announced the release of BEA WebLogic 7.0, BEA WebLogic Platform, and BEA WebLogic Workshop (formerly code-named "Cajun"), along with ECperf results ranking WebLogic 47% faster than the next competitor, and, among other things, the acquisition of a Swedish company for it's Intel-specific JVM, JRockit. I must admit, it was interesting news.

THE LARGEST INTERNATIONAL

XML CONFERENCE & EXPO IN THE WORLD!

XML EDGE conference & expo

JUNE 24-27
JACOB JAVITS CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE CONVENTION CENTER
SAN JOSE, CA

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!

XML-NEXT G OF ENTERPRISE DEPLOYMENT

REGISTER ONLINE TODAY
FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT WWW.SYS-CON.COM

Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost effective Internet and intranet enabled business process integration.

Focus on XML during information-packed days while you enjoy an XML/Web Services Keynote panel, comprehensive conference sessions, and an unparalleled opportunity to exchange ideas with peers and industry leaders.

A Sampling of XML-Focused Sessions

- XML STANDARDS - AN OVERVIEW
- OASIS STANDARDS UPDATE
- UBL - A UNIVERSAL BUSINESS LANGUAGE
- BRINGING XML TO PKI
- LINK MANAGEMENT WITH XLINK
- PRACTICAL XSLT AND XPATH
- ENTERPRISE CONTENT MANAGEMENT WITH XML
- XML IN THE ENTERPRISE AND INTER-ENTERPRISE WORLD

NORBERT MIKULA
XML CHAIR
BOARD OF DIRECTORS, OASIS
INDUSTRY EDITOR
WEB SERVICES JOURNAL

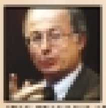
Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 *i*-TECHNOLOGY PROFESSIONALS

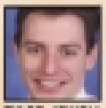
Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- *i*-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...

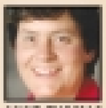


JEAN FRANCOIS ABRAMATIC
SENIOR VP R&D, FORMER CHAIRMAN, W3C • ILOG

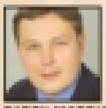


TYLER JEWELL
PRINCIPAL TECH. EVANGELIST • BEA

DAVID LITWACK
CEO
SILVERSTREAM

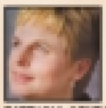


ANNE THOMAS MANES
CTO
SYSTINET



BARRY MORRIS
CEO
IONA

RICK ROSS
FOUNDER
JAVA LOBBY



PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit information

CONTACT: RICHARD ANDERSON
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645
201 802-3056
RICHARD@SYS-CON.COM

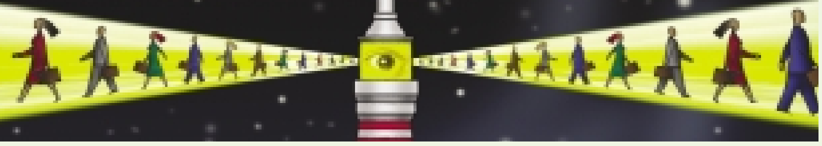
SPONSORED BY:

IONA END 2 ANYWHERE™
ADOS
bea
TogetherSoft
SilverStream
Actional
MERANT
Altaworks
PolarLake Enterprise Strength XML
ALTOVA xml spy

MEDIA SPONSORS

Federal Computer Week
Webservices.org
XML TIMES.com
Java Skyline
CF Advisor
WebServicesMall
XML.org
WIRE
JCert
wrox
OASIS
XML.org
SD Times
JAVA
wireless
XML
WebLogic
WebServices
WebSphere
COLDFUSION

OWNED AND PRODUCED BY
SYS-CON MEDIA
SYS-CON EVENTS



Sessions

The sessions this year were broad-ranging and classified into the following categories: business, general, and two levels of technical. Business sessions were obviously more focused on ROI and other management-related topics. Topics that all audiences might be interested in, such as the new BEA WebLogic Platform features, were categorized as general. The technical sessions were divided into tiers to help people attend sessions that were appropriate for their level of experience with Java and BEA products. This helped keep the J2EE architects out of the "EJB 101" sessions and vice versa.

What was exciting to see this year was more emphasis on Web services rather than just advanced features of J2EE and how WebLogic implements them. And, since the focus of BEA is now "simplification" of the J2EE environment to allow new developers to adopt BEA technology more quickly and easily, there were many sessions on the BEA WebLogic Workshop product.

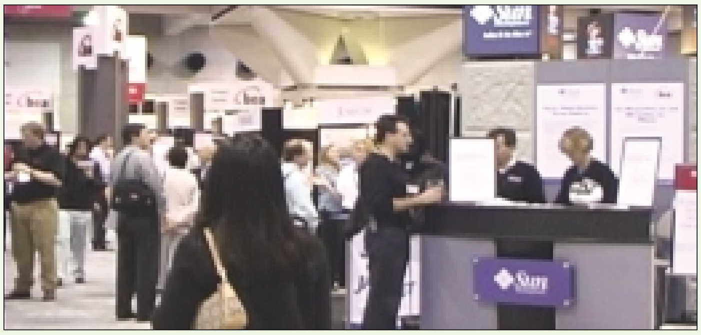
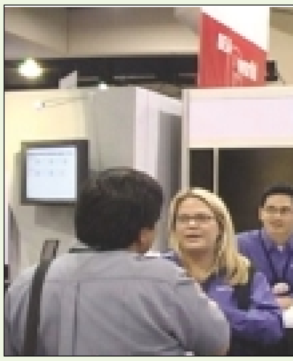


Exhibit Hall

The exhibit hall boasted over a hundred vendors, many of which I classified into the following categories: monitoring and/or performance products from companies like Sitraka and Resonate; content management and portal solutions from BEA (of course); divine, Documentum, FatWire, Interwoven, and Mongoose Technology; security products from veterans like RSA Security and Netegrity; and vendors such as AltoWeb, WakeSoft, Air2Web, and KANA that build on top of WebLogic.



RECEIVE $150
DISCOUNT OFF FULL CONFERENCE
WEB SERVICES EDGE REGISTRATION
web services EDGE conference&expo
2002WORLDTOUR
Learn How to Develop
SOAP Web Services NOW!
at a One-Day Tutorial...Coming to a City Near You!

How have the exhibitors changed from just a year ago? BEA and J2EE are maturing. You can see it in the types of vendors exhibiting at BEA eWorld 2002. Not too long ago, a battle for the application server market waged. Now, only a few stand strong, and companies like Silverstream and Persistence Software have changed their core focus from application servers to technology that integrates with BEA at traditional WebLogic soft spots like development tools and caching.

Also, the maturing of BEA and J2EE is evident in the number of products built on top of WebLogic, and in the number of vendors offering products for performance testing, monitoring, and security. In the past, who cared about security when you were still struggling to get your EJB deployed! There was nothing to load test and nothing to monitor. This is no longer the case. Applications are moving from the development stage to testing and production stages and there is a great need for these kinds of products.

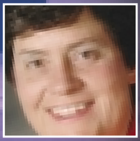
I had a chance to interview numerous companies exhibiting at the conference. Out of about 20 interviews, 8 or more were performance testing or monitoring vendors. As application servers consolidate, I'm sure these vendors will too. Until then, they're individually filling a much needed role in the life cycle of application development and deployment.

Overall, kudos to BEA for a great show! I look forward to attending next year in Orlando, Florida. If it's anything like this year's show, I recommend you go as well.

Jump-start your Web Services knowledge
Get ready for Web Services Edge East and West!

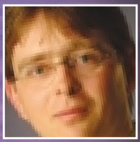
AIMED AT THE JAVA DEVELOPER COMMUNITY AND DESIGNED TO EQUIP ATTENDEES WITH ALL THE TOOLS AND INFORMATION TO BEGIN IMMEDIATELY CREATING, DEPLOYING, AND USING WEB SERVICES.

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASE TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI, AND XML, AND MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS, AND REMOTE REFERENCES.



PRESENTERS...

Anne Thomas Manes, Systinet CTO, is a widely recognized industry expert who has published extensively on Web Services and service-based computing. She is a participant on standards development efforts at JCP, W3C, and UDDI, and was recently listed among the Power 100 IT Leaders by Enterprise Systems, which praised her "uncanny ability to apply technology to create new solutions."



Zdenek Svoboda is a Lead Architect for Systinet's WASP Web Services platform and has worked for various companies designing and developing Java and XML-based products.

EXCLUSIVELY SPONSORED BY
systinet

NEW YORK (Doubletree Guest Suites) ....APRIL 19
SAN FRANCISCO, CA (Marriott San Francisco) ....MAY 2002

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE $495 REGISTRATION FEE.

Register at www.sys-con.com or Call 201 802-3069

SYS-CON i-TECHNOLOGY EDUCATION

If you've had the pleasure of working with BEA WebLogic Server for several years, you may know that RMI/T3 calls between different releases of the server used to be disallowed – to be buzzword - compliant, the releases were not "wire compatible."

RMI Interoperability – In a clasloader of its own

A NEW PERSPECTIVE ON JAVA CLASS LOADERS THAT MAKES WIRING YOUR APPLICATIONS TOGETHER EASIER

BY **PHILIP ASTON**

AUTHOR BIOS

Philip Aston is a senior consultant for BEA Professional Services, specializing in WebLogic Server. He also maintains "The Grinder," an open-source load testing tool at http://grinder.sourceforge.net.

CONTACT...

paston@bea.com

For example, you couldn't call an EJB deployed on WebLogic Server 4.5.1 from a WebLogic Server 5.1 application. This often meant that groups of applications had to be migrated together to new versions of WebLogic Server.

The lack of wire compatibility became more of a problem as users began to put increasingly sophisticated WebLogic Server configurations into production. Engineering stepped up to the mark and WebLogic Server 6.1 became the RMI/T3 "interoperability baseline" – interoperability between 6.1 and a number of future releases is designed into the product. New features such as Web services and enhanced RMI/IIOP also provide creative alternatives.

Not everyone has the luxury of green field sites using the latest WebLogic Server releases. In this article, I describe a scheme for sending JMS messages to both a WebLogic Server 5.1 application and a WebLogic Server 6.1 application from a single client. The approach I take is to use a special class loader that allows both versions of the RMI stack to be loaded into a single Java Virtual Machine (JVM).

A warning: you might need your propeller-head hats for some of this. I hope you come out the other side with an enhanced understanding of how class loaders work and how to use them to your advantage. The technique is not limited to addressing WebLogic Server interoperability problems; it can be applied wherever you want to load more than one version of a particular code base in a single JVM.

The full source code for this article can be found on the ***WLDJ*** Web site at www.sys-con.com/weblogic/sourcec.cfm.

So What's The Problem?

For a client to engage in RMI communication with a particular version of WebLogic Server, it must have the appropriate classes available. Usually, this means adding the WebLogic classes directory (for 5.1) or weblogic.jar (for 6.1) to the client's system classpath. The client uses these classes to communicate with the server and RMI stubs for application types are then retrieved from the server at runtime. Unfortunately, RMI classes and stubs from WebLogic Server 5.1 are not compatible with those from WebLogic Server 6.1; they can't be loaded together.

Java uses class loaders to control how classes are located and loaded into the virtual machine. Each class used by a program belongs to a class loader and, by default, a method uses its caller's class loader to load other classes. Class loaders are arranged in a hierarchy, with the Java boot, extension, and system class loaders at the top of the hierarchy. By convention, class loaders defer to their parents, meaning that when asked to load a particular class they give their parent class loader the option to load it instead. This is usually sensible; it minimizes the number of instances of a class that are loaded and allows parts of an application that use different class loaders to communicate through a set of common classes loaded by a parent class loader.

It's quite possible for a class with the same name to be loaded independently in two different class loaders and in this case the two instances are treated as distinct. By creating appropriate class loaders, you can partition parts of your application into separate "sandboxes" and ensure that classes one application component loads do not affect other components. This is what WebLogic Server 6.1 does to ensure that Web applications and enterprise applications are kept separate from one another. Refer to http://edocs.bea.com/wls/docs61/programming/packaging.html#1048725 for details. The WebLogic Server class loader also supports dynamic loading and unloading (hot deployment) of Web applications and enterprise applications. An interesting article on writing dynamic class loaders can be found at www.javageeks.com/Papers/ClassForName/index.html.

By now, I think you can see where this is going. We should be able to load the two versions of the WebLogic Server RMI classes in separate class loaders. Our client needs to access both JNDI and JMS services from the two servers. The services are implemented differently in the two server versions, and we'll need to load two sets of the classes and stubs. We can't load everything in isolation though; the JNDI and JMS interfaces must be loaded in a single class loader so we can compose a useful program that calls both servers. Figure 1 shows the class loader hierarchy we will use.

An alternative strategy would be to load one version of the RMI classes and stubs in the same class loader as the controlling code (the main method in our example). This would be the case if you were using the technique to call out from a servlet running in one version of WebLogic Server to an Enterprise JavaBean running in another. I chose to use a distinct class loader for each version of the RMI classes because the symmetry makes the code simpler to understand. However, the class loader implementation I provide allows you to take either approach.

The Isolating Class Loader

If you've been paying attention, you might be wondering about the previous paragraph. How would a WebLogic Server 6.1 class loader work if the WebLogic Server 5.1 RMI classes were loaded in the system classpath? When asked to load a particular class, wouldn't the WebLogic Server 6.1 class loader first call the system class loader (its parent), which would erroneously load a WebLogic Server 5.1 version of the class?

This would certainly be the case for a conventional class loader, which would always defer to its parent first, but I've created an unconventional class loader, IsolatingClassloader. Each IsolatingClassloader is constructed with a list of packages and classes that it should load, and a class path that specifies where to load them from. This method is called to load a class and is shown in Listing 1.

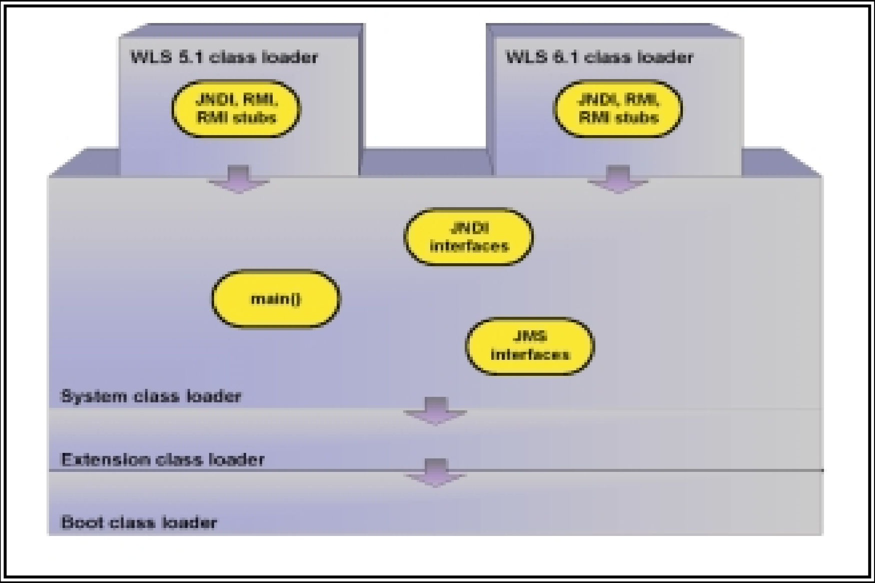
IsolatingClassLoader first determines whether the class name is one that should be loaded in isolation, and if so, attempts to discover and load the class and make it available to the JVM. The class loader asks its parent to load the class only if the class is one that shouldn't be isolated or an implementation cannot be found.

Putting It To Work, The JMS Client

The supplied example client has four command-line parameters, the classpath and JNDI provider URL for each of the servers. Listing 2 is an example of the client running against the two servers. Note that I added the WebLogic Server 5.1 weblogicaux.jar file to the system classpath. This contains the JNDI and JMS interface classes.

To call a particular version of the server, we first construct an instance of IsolatingClassLoader that will isolate the WebLogic implementation classes (package names beginning with weblogic and COM.rsa) and set an appropriate classpath. We can then use the code in Listing 3 to call either version of the server. All that changes from server to server is the isolating class loader instance and the JNDI provider URL.

FIGURE 1



Class loader hierarchy

I guess you're wondering what those setContextClassLoader calls are doing? These calls set a "context class loader," which is associated with the current thread. The Java JNDI and RMI interface classes use the context class loader to determine how to load the implementation classes.

I encourage you to download the source code to see just how elegant the class loader solution is. With the exception of the IsolatingClassLoader setup and the calls to setContextClassLoader, the application code uses the standard J2EE API's.

Bridging Interface Differences

By now you can appreciate the appropriate emerging pattern when using IsolatingClassLoader to call several different versions of an implementation. Interface code is loaded in a common class loader; implementation code that needs to be isolated is loaded in its own IsolatingClassLoader. But what if the interfaces themselves vary?

In fact, this is the case for our JMS example. WebLogic Server 5.1 uses JMS 1.0.1, whereas

WebLogic Server 6.1 uses JMS 1.0.2b. There are minor API differences in the handling of TopicConnections between the two versions of the JMS specification. Even if you don't create a TopicConnection, you'll run into this problem when you attempt to employ the IsolatingClassLoader technique to make WebLogic Server 5.1 JMS calls from a WebLogic Server 6.1 application. WebLogic Server 5.1 RMI performs strict type-checking that uses the stub interface and throws a ClassCastException if the local interface type does not exactly match the remote interface type. The mirror image of the scheme, with a WebLogic Server 6.1 IsolatingClassLoader embedded in a WebLogic Server 5.1 server, works fine because WebLogic Server 6.1 RMI uses dynamic proxies and is more relaxed about the API differences.

All is not lost. I've successfully extended the IsolatingClassLoader pattern to address this problem. I don't have room to cover the full solution here, but I will briefly describe it. I used a proxy object, residing in the common WebLogic Server 6.1 class loader, which maps JMS 1.0.2b calls to a neutral interface (a subset of the JMS 1.0.1 and 1.0.2b interfaces). A second proxy object, which is loaded by the WebLogic Server 5.1 IsolatingClassLoader, maps calls from the neutral interface to the JMS 1.0.1 interface. I managed to hide this implementation from application code by binding the entry points of the interfaces (connection factories) into the WebLogic Server 6.1 JNDI tree. Look out for this trick in the JMS Bridge that comes with the next release of WebLogic Server.

That's all for this month. I hope I've given you a different perspective on Java class loaders, and that you'll find it useful when wiring WebLogic Server applications together. Till the next time…

Listing 1

```
protected Class loadClass(String name,
                          boolean resolve)
    throws ClassNotFoundException
{
    if (shouldIsolate(name)) {
        Class c = findLoadedClass(name);

        if (c == null) {
            c = findClass(name);
        }

         if (resolve) {
            resolveClass(c);
        }

        return c;
    }

    return super.loadClass(name, resolve);
}
```

Listing 2

```
java -cp jmsinterop.jar;d:\wls5.1\lib\weblogicaux.jar
  com.wldj.paston.interop.DualWLSJMSClient
  d:\bea\wlserver6.1\lib\weblogic.jar
  t3://localhost:9001
  d:\wls5.1\sp10\lib\weblogic510sp10.jar;d:\wls5.1\classes
  t3://localhost:7001

Doing work with Server A:
   looking up queue connection factory
   initializing queue connection
   sending 'Testing'
   sent 'Testing'
   received 'Testing'
   closing objects
Done
Doing work with Server B:
   looking up queue connection factory
   initializing queue connection
   sending 'Testing'
   sent 'Testing'
   received 'Testing'
   closing objects
Done
```

Listing 3

```
final Thread thread = Thread.currentThread();

final ClassLoader oldContextClassLoader =
    thread.getContextClassLoader();

try {
    thread.setContextClassLoader(
        isolatingClassLoader);

    final Properties jndiProperties =
        new Properties();

    jndiProperties.put(
        Context.INITIAL_CONTEXT_FACTORY,
        initialContextFactory);

    jndiProperties.put(
        Context.PROVIDER_URL,
        providerURL);

     final Context jndiContext =
        new InitialContext(jndiProperties);

    final QueueConnectionFactory
        queueConnectionFactory =
        (QueueConnectionFactory)jndiContext.lookup(
            "javax.jms.QueueConnectionFactory");

    // … more JMS work …
}
finally {
     thread.setContextClassLoader(
         oldContextClassLoader);
}
```

SilverStream

www.silverstream.com

STANDARDS-BASED INTEGRATION: THE IMPACT OF WEB SERVICES & J2EE

BY
SCOTT DIETZEN

AUTHOR BIO...

Scott Dietzen has been working on Internet application infrastructure for the past decade. He is the CTO of BEA Systems, and oversees its technology strategy. Dietzen came to BEA via the acquisition of WebLogic, a pioneer in Java and Web application server technology. He holds a PhD in computer science from Carnegie Mellon University.

CONTACT...

sdietzen@bea.com

Standards can redefine a marketplace – consider the impact that SQL had on the relational database market. Standards can also create new markets – without HTML and HTTP, there would be no World Wide Web.

My thesis here is that Web services and Java 2 Enterprise Edition (J2EE) will have a similarly dramatic impact on application integration – advancing the industry from point-to-point integration solutions developed after the fact (which I call integration "in the small") toward standard application containers that are integration-enabled *a priori* (integration "in the large").

By application integration (or simply integration), I don't just mean Enterprise Application Integration (EAI), which happens on intranets (that is, behind the firewall). I also include business-to-business integration (B2B), wherein the applications from one company directly interconnect with the applications of a business partner across the Internet (or virtual private network).

In fact, EAI and B2B are already converging. Individual business units (BUs) increasingly manage their own IT infrastructure and applications. As a result, Web technologies are now broadly deployed on intranets to give personnel from one BU access to the data and services of another.

Therefore, just as the Web unites intranets and the Internet, we can expect eXtensible Markup Language (XML) and Web services (conventions for passing XML documents between applications) to become ubiquitous for both EAI and B2B. (*Note:* Web services are most easily defined relative to the Web: the Web was about connecting desktop browsers to documents and applications located anywhere on the network. Web services, then, is about extending that platform [HTTP, Secure Sockets Layer (SSL), and so on] – for application-to-application communications. Said differently, new applications are increasingly "Web ready" out of the box – i.e., designed to support a Web browser front-end. My claim is that future applications will also be "Web integration ready" – ready to plug into the emerging "service grid" that will become the basis for integration in the large.

Making the Business Case

Technology alone is never sufficient to transform an industry; there must also be a compelling business case. Today, large companies depend on tens of thousands of applications. Most of these applications operate in silos, interconnecting only with their close peers. At the same time, the rigors of competition are forcing businesses to specialize – to focus on only what they can do better than anyone else. But as businesses divest and outsource, they are forced to ever more closely integrate with their supply chains and distribution channels.

Integration "after the fact" is such a pain point that some application vendors now say that the only antidote is to purchase every business application from a single supplier so these applications are *preintegrated*. Call this "worst of breed!" First, experience to date would suggest that the promise of preintegration is more marketing hype than reality. Second, and far more important, the notion that any IT-intensive business could rely on a single application provider is absurd. What about verticalization? (No single vendor can have expertise in all industries.) What about the legacy? What about competitive differentiation (via home-grown software)? An IT-intensive business will be nearly as likely to get all of its applications from a single vendor as Boeing or Airbus would be to get the tens of thousands of components necessary to build an airplane (tires, seats, computers, hydraulics, raw aluminum, etc.) from a single manufacturer.

Integration Complexity

Addressing the complexity and cost of integration may well be an IT organization's single biggest need today, and the trends suggest things are only getting worse. Unfortunately, there are no panaceas—integration is and will remain an inherently hard problem.

In my experience, the lack of a *services-based* application architecture is the most typical and problematic barrier to application integration. Within many applications, the *business logic* is intermixed with the *presentation logic*. (*Note*: While the term "services-based architecture" may be relatively new, the concepts have been part of modular programming practice for more than two decades. Yet, many new Web applications similarly lack a services-based architecture: consider that many PERL or ColdFusion scripts intermix HTML page construction and SQL processing. J2EE encourages greater modularity by providing JSPs, JavaBeans, and EJBs to abstract page construction, presentation logic, and business logic, respectively.) Integrating such nonmodular applications generally requires an expensive reorganization of the code to separate out the reusable business logic services (think Web services) from the presentation logic.

In lieu of a costly rewrite, integration of nonmodular applications is often limited to *data sharing* (re-implementing the SQL calls instead of reusing the business service) or screen scraping (reusing the presentation logic as a business service). Data sharing in general refers to reusing the same schemas and tables across multiple applications – a good and well used practice. However, data sharing for application integration means exposing the data model of, say, an Enterprise Resource Planning (ERP) application to a Customer Relationship Management (CRM) system. This violates modular programming practice, increasing application complexity (since SQL queries and updates must be rederived across disparate applications) and fragility (since schema changes within the ERP system may break the CRM application). Screen scraping, on the other hand, does reuse the business logic, but is also fragile (since presentation changes may then break remote services). Screen scraping also makes for tedious programming.

Even for beautifully architected applications, however, integration is still hard (albeit less so). For example, *customer* in one application might refer to a business partner, while *customer* in another application means an end user. Even when one application's *customer* is semantically equivalent to another's, there's often an *impedance mismatch* between their respective representations – object versus relational, Java versus COBOL, ASCII versus EPSIDIC, two-line versus three-line addresses, and so on. Impedance mismatches can also be found in application structure – synchronous systems like today's Web applications behave differently from asynchronous (message-based) systems, and both differ from batch systems. Another pervasive example of impendance mismatch is the differences in the timeliness of data between an online transaction processing application and a decision-support application.

(Despite the marketing hype, XML and Web services do not resolve impedance mismatch between representations. XML simply defines a common alphabet for constructing words and documents; it's still up to the respective companies and industries to define the vocabularies [the mapping from XML syntax to business semantics] that enable conversations to take place. Some of these vocabularies will be defined "top down" by industry standards bodies and consortia. Other vocabularies will grow "bottom up" [much as natural languages have] as individual companies or small groups of companies specify the Web services interfaces for conducting business with them. XML-based EAI faces the same challenges, albeit within the boundaries of a single company. While tools that map from one vocabulary to another can ease the burden, defining [and resolving] vocabularies is the biggest barrier to achieving integration in the large. Infrastructure like Web services and J2EE adapters is easy by comparison.)

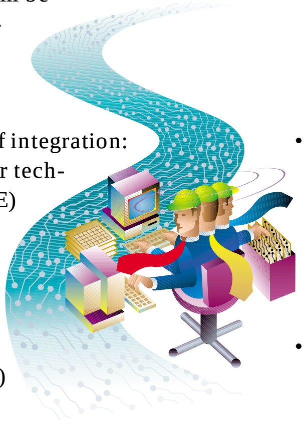
Integration Standards

Application architecture and impedance mismatch are inherent challenges to integration – challenges that are independent of the integration technologies. However, the lack of integration standards is also frustrating IT organizations today. Without appropriate standards, integration in the small will remain costly, and integration in the large, impossible. Today, instead of unifying standards, numerous small vendors offer proprietary integration platforms, making for a fragmented market that cannot protect investment. Collectively, the industry must replace these proprietary technologies with standards.

PROPRIETARY PROTOCOLS

The litmus test for a proprietary protocol is whether or not the same platform software has to run on both sides of the network. The Web analogy is compelling – without standardized HTML and HTTP, the ubiquitous interconnection of heterogeneous Web browsers and heterogeneous Web servers couldn't have happened. Proprietary protocols simply won't work for the scale of integration demanded on the Web. Some of the early B2B solutions were stalled in part by their dependence on proprietary protocols. The proliferation of Web integration standards will also help lower the cost barriers to entry – proprietary B2B protocols, like Electronic Data Interchange (EDI) before them, are simply too expensive to foster ubiquity.

(Beware: While XML is indeed a Web standard, XML document-passing conventions can still be highly proprietary. That's why the emerging family of XML/Web services standards – Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), Universal Description, Discovery, and Integration (UDDI), Electronic Business XML Initiative (ebXML), and so on – is so essential. Without such standards, users will be unable to mix and match integration solutions as they have Web technologies.)



PROPRIETARY ADAPTERS

Adapters solve the "last mile" problem of integration: they provide the mapping between a newer technology (such as Web services and Java/J2EE) and a "legacy" technology (such as COBOL/CICS). (*Note:* By using the term "legacy," I intend no insult. Becoming part of the legacy is the goal we set for all new software, and in a real sense, all production applications are part of the legacy.) Even with the emergence of XML and Web services, adapters remain essential, because little of today's legacy is going to be extended to directly support Web services. Moreover, adapters allow tighter integration with the legacy than Web services by, for example, allowing a single transaction or security context to be shared between the new and old applications.

Without a standard model for adapters, it's nearly impossible to get to critical mass. Consider the impact that standardizing on a common database adapter – Java Database Connection (JDBC) – had on Java's success. The J2EE Connector Architecture (CA) generalizes JDBC to define a universal model for Java adapters. J2EE CA eliminates the n^2 cost of each integration vendor having to "one off" their own proprietary adapter for each and every legacy system. More importantly, J2EE CA protects programming investment in integration solutions the same way JDBC protects investment in database applications.

PROPRIETARY APIS AND INFRASTRUCTURE

The *integration server* hosts the protocols and adapters that make up the "endpoints" of an integration solution. Integration is, of course, harder than simply mapping from one endpoint to another – which is why integration servers also provide sophisticated APIs for developing the "glue" necessary to tie together these endpoints. Unfortunately, within today's integration servers most of these APIs are proprietary. Since the cost of developing an integration solution is mostly in the glue programming, the biggest risk to integration projects is long-term dependency on proprietary APIs. These same proprietary APIs prevent ISVs and SIs from reusing their integration investment across different integration servers.

Following is some of the critical integration server glue that's ripe for standardization:

- ***Messaging***: Typically the most fundamental part of the integration server. The Java Messaging Service (JMS) standardizes asynchronous communications for Java applications with an API that supports the spectrum of messaging needs: store/forward versus publish/subscribe; hub-and-spoke versus peer-to-peer; unicast versus multicast; and so on. The programming of Java messaging applications is already converging on JMS the same way that database programming converged on JDBC. (In addition to a full JMS implementation, BEA WebLogic also offers the binding from JMS to J2EE adapters that enable asynchronous integration as well as synchronous.)
- ***Messaging and Web service components:*** Server-side component models dramatically reduce engineering complexity by moving infrastructure decisions out of the business logic and into deployment descriptors. In this way, the business logic that defines what should be done for a particular JMS message or Web service request can be greatly simplified with Enterprise JavaBeans (EJBs). J2EE message-driven EJBs make it easy to program the "sinks" for JMS messages, while WebLogic Server now automates the mapping from a Web services binding to an EJB session bean or JMS queue or topic.
- ***XML processing and transformation:*** XML and Web services have become so tightly bound to J2EE services (at least within WebLogic), that very efficient XML-standard parsing and transformation technology must be embedded within the platform. Tools that simplify mapping from one XML schema to another, as well as predefined XML transformations (such as mappings to EDI), should similarly be included within the integration server.
- ***Transactions:*** Transactions ensure truly guaranteed delivery via a two-phase commit protocol. Without transactionally-guaranteed delivery, there's always a window of vulnerability in which failures can cause a message to be dequeued without the corresponding database operation or other application processing having completed. The J2EE standard is the Java Transaction API (JTA), and is a prerequisite for J2EE CA.

  (*Note*: Two-phase commit is unlikely to be the best model for guaranteeing Web services. Under it, a single coordinator resolves the entire transaction, and IT organizations are justifiably reluctant to turn over the coordination of updates to their business-critical data to a business partner. Instead, more loosely coupled alternatives to the two-phase commit, such as the proposed Business Transaction Processing (BTP) standard, will be used to harden Web services. BTP uses sequencing and compensating actions (similar to transaction sagas) to ensure reliable operation for loosely coupled applications.)
- ***Security:*** Integration servers need to protect network communications (privacy, protection against tampering) as well as reliably identify participating applications (authentication) and ensure that each participant is only granted access to the appropriate services (authorization). With J2EE, platform security is provided by the Java Authentication and Authorization Service (JAAS) and supporting infrastructure like SSL. JAAS is a J2EE CA requirement, while SSL naturally extends to Web services. (*Note*: Nonrepudiation, the capability to establish the originator of a request after the fact, for example, to guarantee the authenticity of an order, is an area in which the standards are not yet finished.)
- ***Naming:*** Glue programming depends on naming and directory services. The Java Naming and Directory Interface (JNDI) is the Java API for manipulating local names (those in the address space) as well as for accessing the contents of a Lightweight Directory Access Protocol (LDAP) repository. This infrastructure now extends to Web services: WebLogic Server, for example, automatically generates and publishes a thin Java client consumer as well as WSDL (both on a Web URL and to a UDDI directory) to accommodate easy access to Web services.
- ***Business Process Management (BPM):*** BPM refers to the higher-level scripting languages and tools for composing workflows –

THE LARGEST INDEPENDENT

JAVA DEVELOPER CONFERENCE IN THE WORLD!

JDJ EDGE conference & expo

JUNE 24-27
JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!

JAVA IN JUNE
ESPECIALLY IN NEW YORK

REGISTER ONLINE TODAY
FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT WWW.SYS-CON.COM

Focus on Java

Java, now mainstream, is the dominant back-end technology upon which next-generation technologies are evolving.

Hear from the leading minds in Java how this essential technology offers robust solutions to i-technology professionals and senior IT/IS strategy decision makers.

The Java Fast Paths on June 24, a Java-focused CEO Keynote Panel, and comprehensive conference sessions will focus you on Java every information-packed day!

A Sampling of Java-Focused Sessions

- JAVA 1.4: WHAT'S NEW?
- BUILDING TRULY PORTABLE J2EE APPLICATIONS
- JAVA TOOLS FOR EXTREME PROGRAMMING
- BUILDING ASYNCHRONOUS APPLICATIONS USING JAVA MESSAGING
- .NET VS. J2EE
- J2EE: SETTING UP THE DEVELOPMENT ENVIRONMENT
- BUILDING WEB SERVICES WITH J2EE
- DETECTING, DIAGNOSING, AND OVERCOMING THE FIVE MOST COMMON J2EE APPLICATION PERFORMANCE OBSTACLES

ALAN WILLIAMSON
JAVA CHAIR • EDITOR-IN-CHIEF
JAVA DEVELOPER'S JOURNAL

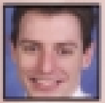
Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 i-TECHNOLOGY PROFESSIONALS

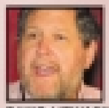
Who Should Attend...

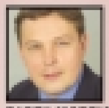
- DEVELOPERS, PROGRAMMERS, ENGINEERS
- i-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...


TYLER JEWELL
PRINCIPAL TECH. EVANGELIST • BEA

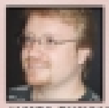
GREGG KIESSLING
CEO
SITRAKA


DAVID LITWACK
CEO
SILVERSTREAM


BARRY MORRIS
CEO
IONA

GREGG O'CONNOR
PRESIDENT
SONIC SOFTWARE

RICK ROSS
FOUNDER
JAVA LOBBY


JAMES DUNCAN DAVIDSON
FATHER OF ANT

For Exhibit information
CONTACT: MICHAEL PESICK
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645
201 802-3057
MICHAEL@SYS-CON.COM

SPONSORED BY:
IONA | END 2 ANYWHERE™
ADOS
bea
TogetherSoft
SilverStream
Actional
MERANT
Altaworks
PolarLake Enterprise Strength XML
ALTOVA

MEDIA SPONSORS
Federal Computer Week
Webservices.org
XML TIMES.com
Java Skyline
CF Advisor
WebServices Mall
XML.ORG
WIRE
JCert
wrox
OASIS
SD Times
JAVA DEVELOPER'S JOURNAL
wireless
XML JOURNAL
WebLogic
WebServices JOURNAL
WebSphere
COLDFUSION DEVELOPER'S JOURNAL

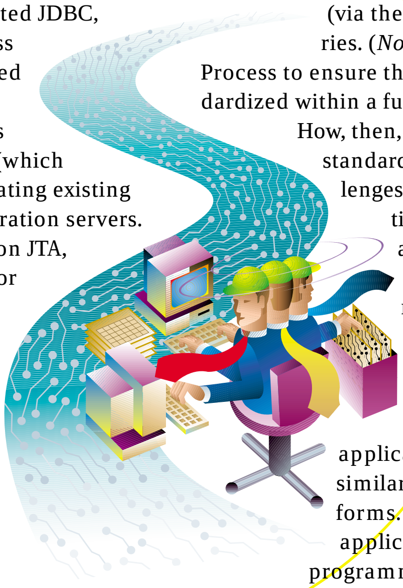
OWNED AND PRODUCED BY
SYS-CON MEDIA
SYS-CON EVENTS

computational sequences of tasks, exceptions, and so on. With WebLogic Integration, for example, workflows can be graphically programmed by dragging and dropping standard J2EE building blocks – EJBs, JMS messages, Web services, and so on.

- ***Performance, scalability, and reliability:*** Integration servers are as business-critical as the applications they interconnect. Delivering the deeply intertwined clustering technologies that transparently ensure quality of service – automated replication, load balancing, failover, caching, session protection, prioritization, and routing (including data- or content-dependent routing) – is both very hard and very expensive. So expensive that the cost of developing business-critical platform software (e.g., operating systems, databases, and application servers) is broadly amortized as the industry consolidates on a small number of long-term winners.
- ***Monitoring and management:*** Management consoles need to provide a comprehensive operational view into the interworkings of the integration server and its connections to various applications. Moreover, the integration server should offer extensible instrumentation so that collating a line of business view (orders, dollars, etc.) into the integration hub is as straightforward as possible.

Relevance of J2EE

No doubt the above list is incomplete (e.g., I omitted JDBC, which is needed for data sharing). But it nevertheless makes a compelling case that the future of Java-based integration solutions will be built on top of the J2EE platform! To date, J2EE-standard application servers have primarily been used to host new applications (which may access legacy systems), while the task of integrating existing applications has been left to more proprietary integration servers. But consider that J2EE adapters depend directly upon JTA, JAAS, JNDI, and clustering; and indirectly on JMS (for asynchronous invocation), the EJB container (for message-driven EJBs), XML services, and so on. Similarly, Web services depend directly on HTTP servlets, XML processing, and clustering, and indirectly on EJB, JMS, and so on. Add it up; the conclusion is that *virtually all of J2EE is essential to integration*, provided you believe in the need for standardization of adapters and Web services!



This should not prove too surprising. After all, why should integration programming differ so fundamentally from application programming? Why should expensive infrastructure software be built twice? Or purchased twice? Of course, the J2EE community has also been hard at work extending J2EE to make it a better fit for integration.

As a result, the industry is rapidly coalescing around the Java/J2EE platform and the .NET alternative from Microsoft. Both platforms have a compelling and largely-shared vision of Web services, a vision which is already being proven with direct interoperability testing. (*Note:* The BEA WebLogic server native SOAP implementation is interoperability-tested against Microsoft .NET, Micrsoft's SOAP Toolkit, GLUE, and IBM WebSphere [which is based on the Apache SOAP implementation]). Some of the additional value proposition for Java is, however, that

- Java/J2EE allows development and deployment across virtually all mainstream computing platforms (coverage is, of course, essential to integration).
- Java/J2EE APIs are standards that reflect the contributions of hundreds of companies within the Java community.
- Web services bindings can be generated transparently for existing Java/J2EE applications (programmers use what they already know), which is key for integration in the large.
- The J2EE CA is commercially viable today.

Indeed, software vendors such as PeopleSoft, Siebel, SAP, and others are already working with BEA to deliver standard J2EE adapters for their enterprise applications. In addition, leading systems integrators like Accenture, AMS, CSC, EDS, KPMG, and so on are collaborating with BEA to develop standards-based integration practices around this new framework. Of course, all of the major Java systems vendors – BEA, Sun, IBM, Bull, HP, Compaq, NEC, Nokia, Oracle, Unisys, and so on – are also working toward leveraging J2EE as a platform for integration. All this gives J2EE-based integration critical mass.

Conclusion

Of course, this new standard integration platform will take time to mature. Web services standards, in particular, continue to evolve: nonrepudiation, guaranteed delivery, and compensating actions are three key areas of standards activity. J2EE-compliant adapters are only now being built. Moreover, vendors like BEA have had to extend the J2EE CA with essential capabilities like bidirectional communications, support for asynchronous processing (via the Java Message Service), and metadata repositories. (*Note:* BEA is working with the Java Community Process to ensure that these extensions can be appropriately standardized within a future release of J2EE.)

How, then, should organizations best prepare themselves for standards-based integration? By treating integration challenges both tactically and strategically: tactically, by getting the job done with a mix of best-fit standard and proprietary technologies; strategically, by betting on the emerging standards. In particular, I recommend that you avoid long-term or large commitments to proprietary integration frameworks, at least until the transforming impact of standards weeds out the eventual losers.

For those that still have doubts about the applicability of J2EE, we recently witnessed a very similar transformation in application server platforms. Four years ago there were literally dozens of application servers promoting all sorts of proprietary programming models. We said then that J2EE would drive rapid consolidation in this market, and that those who ignored the impact of J2EE would waste their time and money. The majority of those proprietary platforms are now gone. Those that remain have generally redesigned their product around J2EE (or the emerging Windows-centric .NET alternative from Microsoft).

The smart money is on history repeating itself for Web-based application integration. The compelling need for standardization in Web services and Java APIs will drive this market. As with all technology transformations, right now there are compelling opportunities to gain competitive advantage, especially for independent software vendors and systems integrators focused on integration solutions.

Acknowledgements

I am grateful to numerous colleagues at BEA Systems for their thoughtful insights. I am particularly indebted to Rocky Stewart, Tom Mitchell, Pal Takacsi-Nagy, Ajay Patel, Benjamin Renaud, George Kassabgi, and Christina Grenier for their comments on this paper.

Simplex Knowledge Company

www.skc.com

News & Developments

Mongoose Technology Delivers Web Services and Portal Life Cycle Management

(Houston, TX) – Mongoose Technology has entered into an agreement with BEA Systems to build and maintain enterprise portals that incorporate collaboration Web services. Using PortalStudio Collaboration Web Services and BEA WebLogic Server, companies can incorporate powerful interaction management capabilities into their existing portals and Web sites to increase employee productivity, enhance partner effectiveness, and build customer loyalty. Mongoose Technology will also resell the WebLogic Server and include developer's licenses with PortalStudio.

A 30-day free trial is available at www.portalstudio.com. www.mongoosetech.com

AltoWeb Announces Expanded Platform Support and Improved Connectivity to Enterprise Systems

(San Diego CA) – AltoWeb, Inc. has announced release 2.8 of the AltoWeb Application Platform, with added support for the Linux and AIX operating systems as well as new data connectivity support for webMethods. With this release, AltoWeb offers organizations a more powerful enterprise application production platform.

The AltoWeb Application Platform can now integrate enterprise applications running on most common servers and network platforms supporting Java 2 Enterprise Edition (J2EE), including Windows, Solaris Linux, and AIX. www.altoweb.com

Wily Technology to Deliver Comprehensive Management Solution

(San Diego CA) – Wily Technology has announced a comprehensive performance management solution for WebLogic with the addition of Introscope PowerPack for BEA WebLogic Server to the Introscope family of products.

PowerPack for WebLogic works with Introscope and Introscope SQL Agent, Wily's database monitor, to provide a management solution for the WebLogic application environment. It combines Introscope's ability to monitor the performance of production J2EE components such as EJBs, JSPs, and servlets with measurements specific to WebLogic.

Developed with assistance from the BEA WebLogic Development Team, PowerPack for WebLogic is pre-configured to offer instant component-level monitoring of WebLogic Server, including JDBC Connection pools, Execute Threads Queue, HTTP sessions, SOAP Services, and security infrastructure. www.wilytech.com

WebGain Studio Adds J2EE Depth to BEA Workbench

(San Diego, CA) – WebGain, Inc., demonstrated at BEA eWorld that Web services designed in BEA's WebLogic Workbench can be incorporated as components into enterprise Java development projects. With WebGain Studio 7, Web services designed in BEA's WebLogic Workbench will be able to be imported and extended in WebGain Studio and deployed to the BEA WebLogic Server 7.0 Application Server. www.webgain.com

Mercury Interactive Supports WebLogic Server Apps Management

(Sunnyvale, CA) – Mercury Interactive Corporation has announced support for BEA WebLogic Server 7.0. With customized versions of Mercury Interactive's LoadRunner, ActiveTune, the first service designed specifically to tune the performance of production systems, and its Topaz® family of application performance management products, customers can test and manage the performance of their WebLogic applications to ensure optimal scalability and performance before deployment and ongoing availability in production. Using these solutions, organizations can accelerate their BEA WebLogic implementations, identify end-user performance and availability issues, and maximize application performance, often without investing in additional infrastructure.www-heva.mercuryinteractive.com

Rational Software Announces Support for WebLogic Server 6.1

(Lexington, MA) – Rational Software has announced support for BEA WebLogic Server 6.1 with Rational XDE Professional: Java Platform Edition and a new BEA WebLogic plug-in for the Rational Unified Process, simplifying and accelerating the development and deployment of Java and J2EE(TM) applications.

The new process eliminates the gap between software design and development in the Java environment. It offers the ability to deploy Java and J2EE applications directly from Rational XDE Professional to BEA WebLogic Server for execution. www.rational.com

Precise Delivers SmarTune Technology to Automate Performance Management

(San Diego) – Precise Software Solutions has unveiled version 2.0 of Precise/Indepth for BEA WebLogic, the J2EE application component of Precise i3, their comprehensive solution for application performance management. Precise/Indepth for BEA WebLogic helps to ensure optimal performance of J2EE applications as they are deployed into business-critical production systems on BEA WebLogic servers.

The new release delivers Precise's revolutionary new SmarTune technology, which automatically identifies the root causes of J2EE application performance problems and recommends the most appropriate corrective actions. When the performance of production J2EE applications degrades, it automatically alerts IT staff and provides intelligent analysis and recommended actions to resolve the issues before they turn into costly problems that adversely affect business continuity. www.precise.com

WebGain

www.webgain.com

Mongoose

Technology

www.mongoosetech.com