SYS-CON

MODEL-BASED TESTING FOR JAVA APPLICATIONS PAGE 6

No. 1 *i*-Technology Magazine in the World

JDJ

JDJ.SYS-CON.COM VOL.12 ISSUE:6

COMING TO NEW YORK CITY! SEE PAGES 29 and 31

3-DAY EVENT!

11th International 2007 SOAWORLD™ CONFERENCE & EXPO

PLUS

2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

www.SOAWorld2007.com

2007 VIRTUALIZATION CONFERENCE+EXPO www.virtualizationconference.com

Introduction to Maven 2

A promising application development lifecycle management framework

RETAILERS PLEASE DISPLAY UNTIL AUGUST 31, 2007

$5.99US $6.99CAN

07>

0 09281 01751 6

PLUS...

Java API for XML Web Services (JAX-WS)

The Evolution of Java

Securing Tomcat Database Passwords

Altova® XMLSpy® 2007 – The industry standard XML development environment.

ALTOVA® xmlspy® 2007

Immerse yourself in

XML intelligence

Dive into XMLSpy® 2007, and take XML development to greater depths.

Spied in Version 2007 Release 3:

- Deep integration with IBM DB2 9: pureXML data server
- Support for new MS Office document format – Open XML
- New database query window supporting SQL, SQL/XML, and XQuery
- New intelligent CSS editor with context-sensitive entry helpers

Altova® XMLSpy, the industry standard XML development environment, is vital for modeling, editing, transforming, and debugging XML applications. Breathe life into your plans with the world's leading XML editor, the original graphical schema designer, a code generator, file converters, debuggers, profilers, full database integration, support for XSLT, XQuery, WSDL, SOAP, and a trove of treasured XML utilities and usability aides. Become a markup mastermind! **Download XMLSpy® 2007 today: www.altova.com**

Join Altova at Microsoft TechEd, Orlando **Booth # 908**

XMLSpy is also available as part of the value-packed Altova MissionKit™ software bundle.

ALTOVA®

Conference Presentations, Magic Shows, and the Five-Ring Circus

Joe Winchester

Having attended two conferences in the past three weeks and seen untold presentations, I've come to the conclusion that irrespective of the subject matter, each presenter invariably falls back on the same technique to impress the audience: to rely on the skills of a conjurer or circus ringmaster as they try to captivate, amaze, and hoodwink their audience.

The Magic Show

Magicians rely on a basic technique to dazzle and fool their audience. They set up the promise of something really difficult – "I'm going to make this rabbit disappear" – and then go ahead and perform the trick right in front of our eyes. The act can be repeated with silk handkerchiefs, coins, chopping people in half, but whatever the prop it basically involves doing something seemingly impossible. Having captured the attention of the audience, the conjurer then elaborates on the original trick by doing something to extend the theme and pushes the envelope to a further level of disbelief. This could involve removing the apparently lost rabbit from inside someone's hat, or finding a missing object inside an impenetrable empty box that was locked at the start of the act. The basic formula is to do something that looks difficult, and then surpass the act with a variant that uses the same props yet seems more impossible, the goal being to make the audience enthusiastically clap while exclaiming to themselves, "I don't know how he did that, pure magic!"

The Technical Presentation

At conferences the presenter typically has one hour to win over the audience and usually warms them up with a few jokes before launching into demo number one. This involves some kind of GUI appearing from an IDE. It doesn't really matter what the GUI looks like at this stage or for that matter what technology is being used, the key thing is that a few lines of code can be turned into a running GUI. There might be some applause, but this is just in anticipation of greater things to come. The presenter will point out a few flaws in the GUI, return to the IDE to make a couple of changes, push the save button, and wait. It's allowable for the presenter to hit some kind of refresh button on the GUI to have the change reflected, although more kudos is scored if this isn't necessary and the runtime update occurs without any obvious intervention.

Assuming nothing has crashed, the presenter is in full stride and after a few slides to tease and set the scene for the finale, a more difficult and risky change is made to the IDE code. It's possible the GUI was based on some kind of metadata such as a database schema or WSDL file, in which case this input definition will be swapped out. Doesn't really matter what rug is pulled, except that for this feat the expectation is for something more impressive than the first couple of acts. After the presenter nervously pastes in some magic line of code or takes a suspicious menu option hastily added for the demo, the original GUI is transformed. Like a frog turning into a princess, the GUI is more beautiful, has more color, and possibly some sound is played to hopefully send the audience into a chorus of applause.

The Five-Ring Keynote

In the late 1800s the troupe of Barnum & Bailey improved on the art of previous circus shows by having multiple acts simultaneously performing side by side in what became known as the "three-ring circus." Conference keynotes now seem to recognize this as the way to keep the show moving along. You now find multiple back-to-back podiums with different teams of engineers showboating their particular polemic using the same code formula – execute, change, re-execute with the GUI looking better, and so forth. To accompany this, an emcee introduces the engineer and adds narrative in the form of rhetorical and rehearsed questions: "That's great Thor, but it looks pretty ordinary, what can you do about that for us?" to which the engineer is cued to add some magic lines of code, too fast for the audience to see, question, or even appreciate, as the end game is just to refresh the GUI with gratuitous use of color, animation, and, for bonus marks, sound effects. After the applause the engineer takes his seat and the next act is called onto the stage.

–continued on page 34

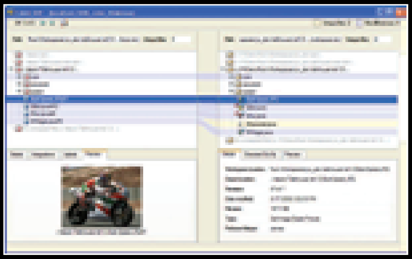
Joe Winchester is a software developer working on WebSphere development tools for IBM in Hursley, UK.

joewinchester@sys-con.com

Perforce | *Fast Software Configuration Management*

Introducing Folder Diff,

a productivity feature of Perforce SCM.



Perforce Folder Diff

Folder Diff is an interactive, side-by-side display for comparing the state of any two groups of files.

Use Folder Diff to quickly determine the differences between files in folders, branches, labels, or your local disk. This is especially useful when performing complex code merges.

And when you've been working offline, Folder Diff makes it a snap to reconcile and catch up with the Perforce Server when you get back online.

Folder Diff is just one of the many productivity tools that come with the Perforce SCM System.

PERFORCE SOFTWARE

Download a free copy of Perforce, **no questions asked, from www.perforce.com. Free technical support is available throughout your evaluation.**

JUNE 2007 VOLUME:12 ISSUE:6

JDJ contents

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. Periodicals postage rates are paid at Woodcliff Lake, NJ 07677 and additional mailing offices. Postmaster: Send address changes to: JDJ, SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

Model-Based Testing for Java Applications

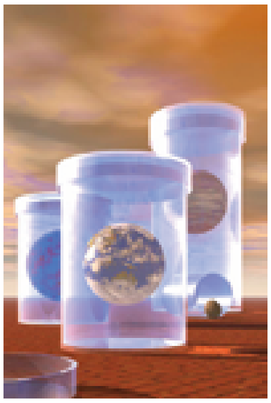
A supplement to automated testing

by Bill Hayduk

Software testing – whether it's automated, manual, or model-based – is a systemic method of discovering variances between how a program was expected to perform and how it actually behaves in use. Every application should be tested to ensure it is both usable and functional – that it doesn't contain bugs or flaws.

Model-based testing has stirred up a significant amount of interest over the past couple of years. For some development and testing teams it's certainly a test method worth exploring but be aware that model-based testing may be a great supplement to the automated testing you already do, but it's really an adjunct and not a replacement for standard automated testing.

A properly constructed automated software quality assurance test aims to discover all of the ways that every feature in a piece of software can be used, and how these features will interact with all other features, exposing the flaws and bugs that lurk in the programming code. Standard tests accomplish this by running a pre-determined series of code examinations. Model-based testing, in contrast, uses algorithms to determine all of the usage paths for an application, pares down that number for maximum coverage and minimal testing, and then generates various test cases to try the application against.



Skilled software testers labor long hours to figure out all of the possible ways a user might interact with an application. But users are almost guaranteed to surprise programmers by figuring out some innovative and exciting way to use a program, ones cause results that may not have been predictable to a logical mind.

Model-based testing can be compared to having many users, all of whom are quite happy to test your application 24/7 for weeks on end, each one interacting with the application in a distinctive way. And because model-based testing generates a broad sampling of test cases, it avoids the fallacies that are potentially inherent in repeating the same narrow run of tests. This can generate a false sense of security as subsequent runs will naturally produce ever smaller numbers of found flaws in programming code that may actually harbor serious problems. That's why model-based testing can sometimes catch flaws that standard tests might miss.

As we all know once you put an application onto the Web, users will inevitably be probing it for flaws or accidentally stumble across whatever bugs managed to evade your software testing process. Spotting and correcting these problems will save money, time, and aggravation. Think of model-based testing as a good way to broaden your quality assurance procedures.

Inherent Issues Concerning Model-Based Testing

Before you even begin to seriously consider bringing model-based testing into your development process it's important to note that this testing method does not suit every application or organization's needs.

As we've discussed, model-based testing examines an application's actual behavior in response to actions. These actions are generated from inputs created specifically for the application that is undergoing testing. To develop the inputs that the tool will use to generate test cases we have to understand, in depth, exactly what the application that we're testing should optimally do, how it should respond to its users, and perhaps how it should interact with other applications. We then need to clearly describe that expected behavior to the testing tool. The process of creating inputs for the testing tool is a highly critical process, since all the results that follow will necessarily be based on those inputs. If the inputs are imperfect, if they don't follow the proper syntax (and so aren't properly processed by the tool) or are incomplete, then your testing process will be flawed.

–continued on page 32

Bill Hayduk is founder, president, and director of professional services at RTTS (www.rtts.com). Over the past 15 years, Bill has successfully implemented large-scale automation projects at many Fortune 500 firms. He has managed projects in most verticals, including banking, brokerage, multimedia, ISVs, government, telecommunications, healthcare, education, pharmaceutical, and insurance.

bhayduk@rttsweb.com

SYS-CON MEDIA

President and CEO:
Fuat Kircaali *fuat@sys-con.com*

President and COO:
Carmen Gonzalez *carmen@sys-con.com*

Group Publisher:
Roger Strukhoff *roger@sys-con.com*

Advertising

Advertising Sales Director:
Megan Mussa *megan@sys-con.com*

Associate Sales Manager:
Corinna Melcon *corinna@sys-con.com*

Events

Events Manager:
Lauren Orsi *lauren@sys-con.com*

Events Associate:
Sharmonique Shade *sharmonique@sys-con.com*

Editorial

Executive Editor:
Nancy Valentine *nancy@sys-con.com*

Production

Lead Designer:
Tami Lima *tami@sys-con.com*

Art Director:
Alex Botero *alex@sys-con.com*

Associate Art Directors:
Abraham Addo *abraham@sys-con.com*
Louis F. Cuffari *louis@sys-con.com*

Web Services

Vice President, Information Systems:
Bruno Decaudin *bruno@sys-con.com*

Information Systems Consultant:
Robert Diamond *robert@sys-con.com*

Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Richard Walter *richard@sys-con.com*

Accounting

Financial Analyst:
Joan LaRose *joan@sys-con.com*

Accounts Payable:
Betty White *betty@sys-con.com*

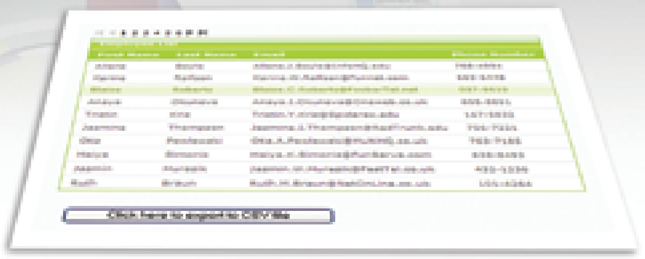
Customer Relations

Circulation Service Coordinator:
Edna Earle Russell *edna@sys-con.com*
Alicia Nolan *alicia@sys-con.com*

"I bring meaning and comprehension to my corporate data using advanced visualization tools."

- Application Developer

WebChart™ - 3D Overlay Donut Chart



WebGrid™ Export to CSV Format

NetAdvantage® for JSF 2007 Volume 1

Consistent Multi-Platform User Experience

WebCharting – Add high impact 2D/3D visualizations with over 20 different chart types and views

Improved Data Reporting – Easily export data within a grid to any application that supports the import of CSV such as spreadsheets and databases

Application Performance – Leverage our AJAX framework to turbo-charge your Web applications

Accessibility – Support of US Code Section 508 across all Infragistics controls

learn more: infragistics.com/jsf

Infragistics®

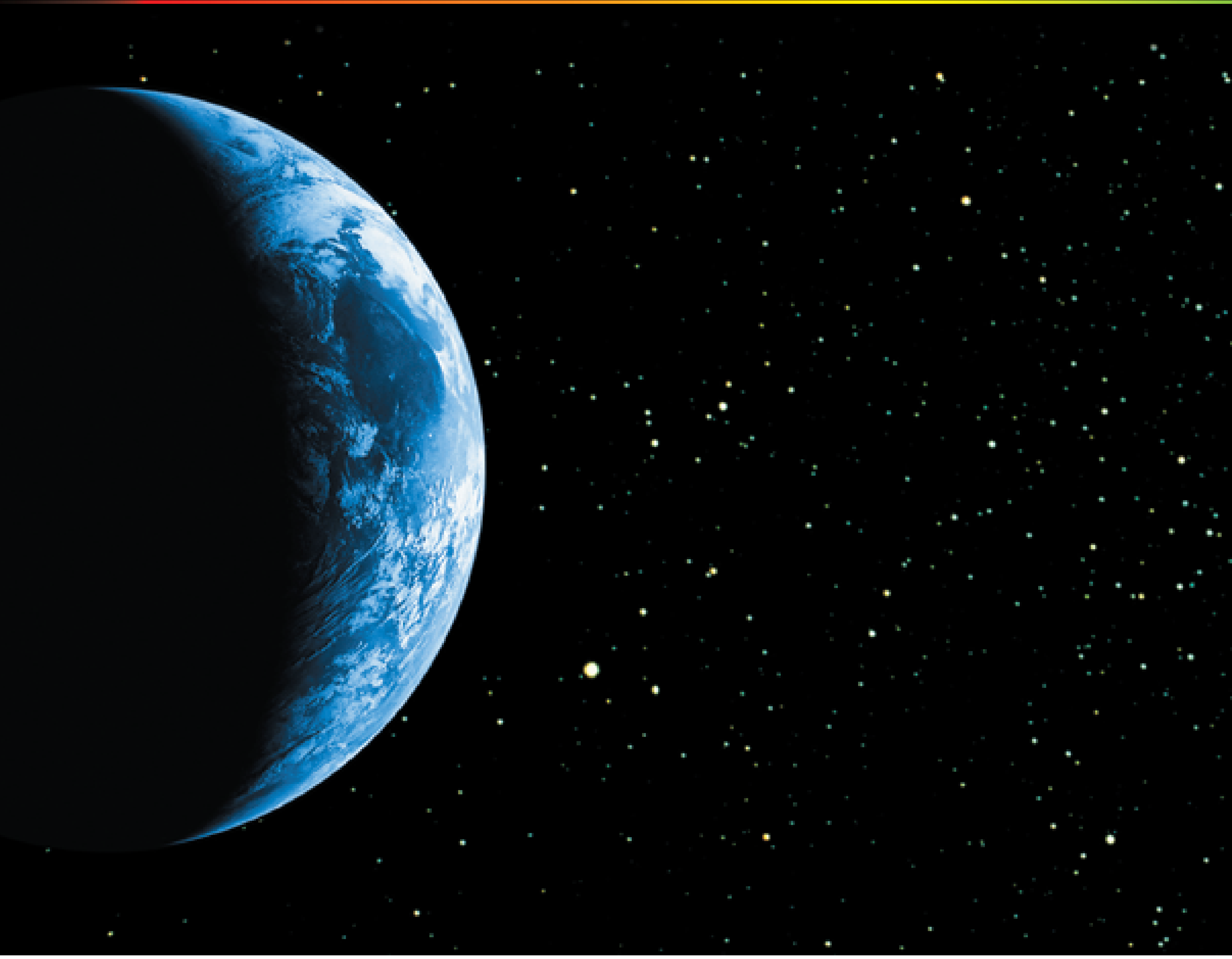
Powering The Presentation Layer

Infragistics Sales - 800 231 8588

Infragistics Europe Sales - +44 (0) 800 298 9055

WINDOWS® FORMS ASP.NET WPF JSF

grids scheduling charting toolbars navigation menus listbars trees tabs explorer bars editors



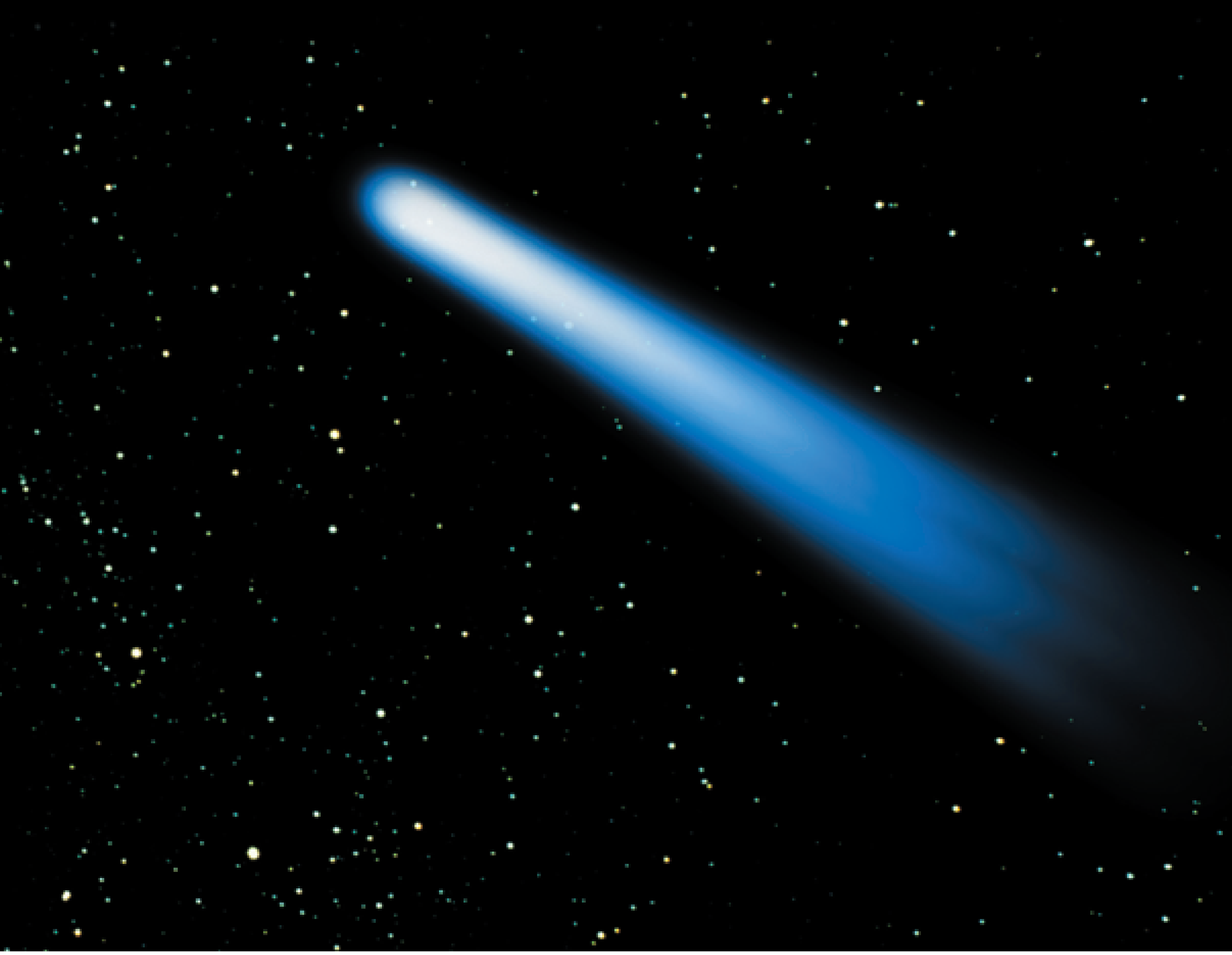
RARE OCCURRENCE.

For a limited time, upgrade to Crystal Reports® XI for only $99. Create brilliant reports in minutes and sp
focus on what you do best – core application coding. A great price with this depth of features is a rare o

- .NET, Java™ and COM SDKs
- Unlimited free runtime for internal corporate use
- Includes Crystal Reports Server – embed report management services
- Includes crystalreports.
- Unlimited installation-re

Act fast. Go to www.businessobjects.com/rareoccurrence or call 1-888

Business Objects™



eed report integration so you can
ccurrence, indeed.

com – share reports over the web
lated support

3-333-6007 today.

NOW
$99
UPGRADE
or
$395 NEW

Spring and Java EE 5

Part 2: Simplicity and power combined

by Debu Panda

In the first part of this article you learned how Java EE 5 has simplified enterprise application development by adopting a POJO programming model and making use of Java 5 metadata annotations. You also discovered how the Spring Framework version 2.0 integrates with Java Persistence API (JPA) and makes it simple to use from enterprise Java applications.

In this second part, you will learn how you can integrate the Spring Framework with EJB 3 components and to exploit Spring's declarative transaction features with Java EE applications. Finally we'll discover how Java EE application servers provide seamless management of Spring components from their JMX-enabled management consoles.

Using Spring and EJB 3

As we discussed in Part 1 of this article, EJB 3 greatly simplifies development of EJB components. Spring's Pitchfork project implements part of the EJB 3 specification, for example, enabling use of @Stateless or @Resource annotations with Spring beans. It also lets developers use @Stateless annotations in a web container.

You can mix and match EJB 3 and Spring components to leverage the power of both frameworks. For example, you can combine such features as statefulness, pooling, clustering, declarative security, Web Service feature, and message-driven beans of EJB 3 with AOP, POJO injection , template-based data, and resource access support provided by Spring.

Injecting Session Beans

You may remember from our earlier discussion that Java EE 5 lacks support for POJO injection. If you want to use an EJB from a helper class of your Web or EJB module then injection isn't supported and unfortunately you have to resort to JNDI lookup. Note that application server vendors will provide proprietary extensions to support POJO injection.

However you can use Spring to simplify and use its powerful dependency injection mechanism to inject an instance of a session bean. This will help your applications to port between application servers. Let us dive down and see an example.

The code examples that follow are taken from my recently published book, *EJB 3 in Action,* published by Manning Publications.

Assume that you have a session bean named ItemManager that you will use in the ItemServiceBean, which is a Spring POJO, as shown below:

```
public class ItemServiceBean implements
ItemService {

  private ItemManager itemManager;
  public ItemServiceBean() {
  }

  // Setter injection of ItemManagerEJB
  public void setItemManager(ItemManager
   itemManager) {
   this.itemManager = itemManager;
  }

  public Long addItem(String title, String
   description,
     Double initialPrice, String sellerId) {
  Item item = itemManager.addItem(title,
   description,
     initialPrice, sellerId);
  return item.getItemId();
  }
}
```

As you can see, we are using setter injection to inject an instance of ItemManager EJB object and invoke a method on the EJB.

By now you must be wondering where the actual magic happens? We're not doing a JNDI lookup and not using the @EJB injection (that we discussed in Part 1) to inject the session bean object. The real magic occurs through wiring in the EJB through the Spring configuration.

Let's assume that the Spring Bean uses a remote business interface of ItemManager EJB and retrieves it using a Spring 2.0 simplified jee-jndi lookup as follows:

```
<jee:jndi-lookup id = "itemManager" jndi-
name = "ejb/ItemManager"
resource-ref = "true"/>
```

Note that we are using setter injection in ItemServiceBean to inject an instance of ItemManager EJB and we must wire the ItemManager EJB as follows:

```
<bean id = "itemService"
      class = "actionbazaar.buslogic.
ItemServiceBean">
  <property name = "itemManager" ref =
"itemManager"/>
</bean>
```

Remember from our discussion in part 1 of the article that EJB 3 Session beans are POJOs and interfaces are POJI. So there's no difference between invoking EJB3 session beans with either a remote or local interface if your Spring beans and EJBs are collocated in the same container, and the configuration is identical for both local and remote session beans. If your Session bean is in a remote container you'll have to provide the JNDI properties such has provider URL, principal, and credentials to invoke the remote bean.

Spring-Enabled Session beans

We 'll examine another interesting case of integration where you can leverage the power of Spring in an EJB 3-based application. You can use Spring in your session beans (both stateless and stateful) and message-driven beans (MDB). I'll demonstrate the use of Spring beans from an EJB 3 stateless session bean. If you've used Spring with EJB 2 you may remember that the framework provides several factory classes for such integration. You can use those abstract classes with EJB 3 session beans to enable access to the Spring bean factory. As these factory classes require they have to implement the onEjbCreate() method in your EJB 3 bean class to access a Spring bean.

Below is the same EJB 3 example (PlaceBid EJB) transformed into a Spring-

Debu Panda, lead author of the recently published EJB 3 in Action (Manning Publications), is a senior principal product manager on the Oracle Application Server development team. He maintains an active blog on enterprise Java at http://debupanda.com.

debabrata.panda@oracle.com

enabled stateless session bean. Here the PlaceBid EJB acts as a façade and delegates the actual business logic to the PlaceBidServiceBean.

```
@Stateless(name = "PlaceBid")
public class PlaceBidBean extends AbstractS
tatelessSessionBean
implements PlaceBid {

    private BidServiceBean bidService;
    public PlaceBidBean() {
    }

    protected void onEjbCreate() {
      bidService =
       (BidServiceBean) getBeanFactory().
          getBean("bidService");
    }

   public Long addBid(String userId, Long
    itemId, Double bidPrice) {
      return bidService.addBid(userId,
      itemId, bidPrice);
    }
}
```

Now let's explore how the Spring bean factory is created and how the Spring configuration is provided.

When an EJB instance is created (when a client invokes an EJB), the onEjbCreate method is invoked automatically. A JNDI lookup is done to obtain the path for the bean factory by using an environment entry named ejb/BeanFactoryPath. So you have to define it in the EJB deployment descriptor for the EJB:

```
<session>
  <display-name>PlaceBid</display-name>
  <ejb-name>PlaceBid</ejb-name>
  <env-entry>
    <env-entry-name>ejb/BeanFactoryPath</
     env-entry-name>
    <env-entry-type>java.lang.String</env-
     entry-type>
    <env-entry-value>/actionBazaar-service.
     xml</env-entry-value>
  </env-entry>
</session>
```

Although EJB 3 made deployment descriptor optional there are a few cases where you still have to use it. If you're a heavy Spring user you live and breath XML configurations so you have nothing to complain about! In our example we've set the env-entry-value for the ejb/BeanFactoryPath environment variable at /actionBazaar-service.xml. So you have to package the EJB classes, Spring classes, and Spring configuration file into your ejb-jar package.

This way you can use the declarative security, transaction, remoteability, and Web Service features of EJB and combine that with POJO injection, AOP, simplified data, and resource access of Spring. Next we'll see a powerful combination of MDB and Spring JmsTemplate.

JmsTemplate and Message-Driven Beans

Message-driven Beans are simple to develop and can be configured as a message consumer. MDBs support several performance features such as pooling and may either be used with a JMS-compliant provider or an EIS using a JCA 1.5-compliant connector. Below is a simple MDB configured against a JMS provider.

```
@MessageDriven(activationConfig = {
        @ActivationConfigProperty(propert
yName =
"destinationName", propertyValue =
"jms/OrderBillingQueue"),
        @ActivationConfigProperty
        (propertyName =
            "destinationType",
            propertyValue =
             "javax.jms.Queue") })
public class OrderBillingMDB implements
MessageListener {
    public void onMessage(Message
     message) {

    }
```

Building JMS producer with Java EE 5 is still a pretty complex task in spite of its injection support. Spring makes building a message producer application very simple as evident in the following code:

```
public void sendMessage() {
    try {
       ..
     JMSSender jmsSender =

 (JMSSender)appContext.getBean("jmsSender");
     jmsSender.sendMesage();

   } catch(Exception e) {
        e.printStackTrace();
    }
}
```

Looks pretty simple! You must be wondering: what is the connection factory and destination configured that this client is sending a message to? The magic is done in background by the Spring configuration as specified below:

```
<bean id="connectionFactory">
       class="org.springframework.jndi.Jndi
            ObjectFactoryBean">
    <property name="jndiName">
        <value>jms/QueueConnectionFactory</
           value>
    </property>
</bean>
 <bean id="jmsDestination"
    class="org.springframework.jndi.
JndiObjectFactoryBean">

    <property name="jndiName">
        <value>jms/OrderBillingQueue</value>
    </property>
</bean>
 <bean id="jmsTemplate"
       class="org.springframework.jms.core.
JmsTemplate102">
    <property name="connectionFactory">
        <ref bean="connectionFactory"/>
    </property>
    <property name="defaultDestination">
        <ref bean="jmsDestination"/>
    </property>

</bean>
<bean id="jmsSender" class="oracle.fusion.
  demo.OrderService">
    <property name="jmsTemplate">
        <ref bean="jmsTemplate"/>
    </property>
</bean>
```

"The real magic occurs through wiring in the EJB through the Spring configuration"

If you look at this code, the JMS producer code is very simple, however, you have to write quite a bit of XML configuration. It's your choice!

While building enterprise applications you probably want your business operations to be transactional. Next we'll see how you can use Spring declarative transactions and how Java EE containers provide the integration of Spring with their Transaction Manager.

Spring-Declarative TX and Integration with Application Server TM

Java EE provides a robust platform for building and deploying a transactional enterprise application. Java EE supports both programmatic and declarative transactions and unfortunately you can use declarative transactions only with EJB. Spring has robust support for declarative transactions and you can use the scheme with any POJO, adding declarative transaction capability to the Web container.

Let's assume that you want to define a declarative transaction for a method in a Spring bean. You have the following code:

```
@Transactional(propagation = Propagation.
REQUIRES_NEW, isolation = Isolation.READ_
COMMITTED)
public Long addBid(Bid bid) {
..
}
```

When the addBid() method is invoked, Spring will automatically start a new transaction per our definition. The integration between Spring and the application server's transaction manager is what makes that possible.

Java EE containers such as Oracle Containers for Java EE (OC4J) provide integration between the transaction manager and Spring so that declarative transactions can be used. You can enable such integration with a simple setting:

```
!-- enable the configuration of transac-
tional behavior based on annotations -->
<tx:annotation-driven/>
<bean id="transactionManager" class="org.
springframework.transaction.jta.OC4JJtaTrans
actionManager">
```

The Manageability of Spring Components

While Spring components are easier to build and deploy in an application server, managing them may pose maintainability issues since Spring is a "foreign" framework. Application servers such as Oracle Application Server address this with interesting integration for managing Spring components.

For example, a Spring bean can be exposed in the MBean browser of Oracle's Application Server Control by making some minimal configuration changes in the Spring application context configuration. For example, we can register the Spring beans with the following Spring configuration:

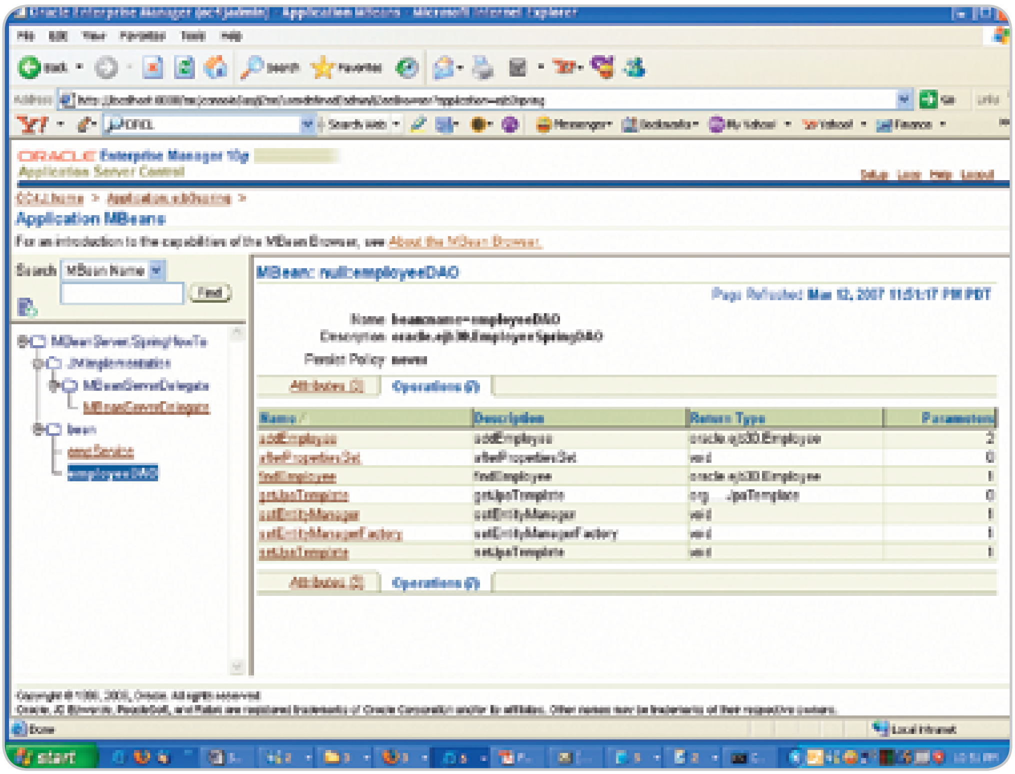
```
<bean id="howToMBeanServer"  class="org.
springframework.jmx.support.
MBeanServerFactoryBean">
 <property name="defaultDomain"
value="SpringHowTo" />
</bean>

<bean id="exporter" class="org.springframe-
work.jmx.export.MBeanExporter">
 <property name="beans">
   <map>
     <entry key="bean:name=empService"
value-ref="empService" />
     <entry key="bean:name=employeeDAO"
value-ref="employeeDAO" />
   </map>
 </property>
 <property name="server"
ref="howToMBeanServer" />
</bean>
```

The Spring beans appear as application-defined Managed Beans (MBean) in the MBean browser of application server's management console. e.g., OracleApplication Server Control's MBean browser as shown in Figure 1. Thus managing Spring components is a snap!

You can perform MBean operations on the Spring beans deployed as a part of your application

Conclusion

This concludes the second installment of the two-part article. In this article, we provided a series of examples to illustrate how Java EE 5 developers can exploit some common integration points with the Spring Framework. Specifically, we demonstrated how EJB 3.0 and JPA combine naturally with Spring, as well as demonstrated how Spring-based applications can utilize high-end Java EE container services such as database access and the JMS messaging infrastructure. Finally you learned how application servers provide management capability for Spring beans.


ORACLE Enterprise Manager 10g
Application Server Control

OC4J:home > Application:ejb3spring >

Application MBeans

For an introduction to the capabilities of the MBean Browser, see About the MBean Browser.

MBean: null:employeeDAO

Page Refreshed Mar 12, 2007 11:51:17 PM PDT

Name: beanname=employeeDAO
Description: oracle.ejb30.EmployeeSpringDAO
Persist Policy: never

Name	Description	Return Type	Parameters
addEmployee	addEmployee	oracle.ejb30.Employee	2
afterPropertiesSet	afterPropertiesSet	void	0
findEmployee	findEmployee	oracle.ejb30.Employee	1
getJpaTemplate	getJpaTemplate	org....JpaTemplate	0
setEntityManager	setEntityManager	void	1
setEntityManagerFactory	setEntityManagerFactory	void	1
setJpaTemplate	setJpaTemplate	void	1

Figure 1 Oracle application server control

Help is here.

At last!

- *Lightweight*
- *Java™ browser SDK*
- *Based upon Mozilla™*
- *Faithful robust rendering*

Swing Edition also features:

- *Pure Swing lightweight rendering*
- *Web Standards compliance*
- *W3C DOM*
- *Easy to embed*
- *Java Applet and plugin support*
- *Extensive API*

All WebRenderer products support the following standards: HTML, XHTML, CSS 1 & 2, JavaScript (inc AJAX), DOM, HTTP, HTTPS, XML, XSL, plugins etc. with faithful and always predictable rendering. Check out our other products:

***PLUS** Desktop Edition*

- Supports *Internet Explorer™*, *Mozilla™* & *Safari™* spawns
- Fast and robust SDK
- Micro edition available
- Native font support
- Faithful, predictable rendering

***PLUS** Server Edition*

- Server optimized and scalable
- J2EE support
- Leading server-side Java browser
- Render to thumbnail image
- Headless print support SDK

JadeLiquid® Software

DOWNLOAD FREE 30 DAY TRIALS NOW! WEBRENDERER.COM

Java API for XML Web Services (JAX-WS)

Creating a JAX-WS 2.0 Web Service in WebLogic Server 10

by Deepak Vohra and Ajay Vohra

WebLogic Server 10 Technology Preview supports JEE 5. A feature of JEE 5 is the Java API for XML Web Services (JAX-WS) used to create Web Services and Web Service clients. WebLogic Server 10 provides the jwsc task to create the Web Service artifacts and the clientgen task to create the artifacts for Web Service clients. In this article we'll create an example JAX-WS 2.0 Web Service in WebLogic Server 10 Technology Preview.

JAX-WS is an API to create Web applications and Web Services using the XML-based Web Services functionality. To create a Web Service first create a Service Endpoint Implementation (SEI) class. The implementation class is annotated with javax.jws.WebService annotation. The implementation class must not be abstract or final, and must contain a default public constructor.

A Web Service provides operations that are made available to Web Service clients. So add business methods to the Web Service class. The business methods are public and must not be static or final and may be annotated with the javax.jws.WebMethod annotation. By default all public methods are made available as Web Service operations.

Preliminary Setup

Download and install the *WebLogic Server 10* Technical Preview. Download and install *Apache Ant*, which is required to run Ant tasks to create Web Service and client classes. Download and install *JDK 5* if not already installed. It's required to run Apache Ant.

Using the WebLogic server Configuration Wizard create a WebLogic domain, base_domain.

Creating a Web Service

We'll create a JAX-WS 2.0 Web Service. First, set the WebLogic server environment by running the setDomainEnv command script.

```
C:\BEA\user_projects\domains\base_domain\bin> setDomainEnv
```

Next, create a project directory in the C:\BEA\user_projects directory with the mkdir command. The directory names should be separated with backslashes instead of forward slashes for the mkdir command to run.

```
C:\BEA\user_projects> mkdir webservices\hello_webservice
```

Create an src directory under the project directory that contains subdirectories corresponding to the package name of the Java Web Service class.

```
C:\BEA\user_projects\webservices\hello_webservice>mkdir src\webser-
vices\hello_webservice
```

Create a Java Web Service (JWS) file for implementing a JAX-WS Web Service. Annotate the Web Service implementation class HelloWsImpl with the @WebService annotation. JSR 181: Web Services Metadata for the Java Platform defines the standard annotations that can be used in a Java Web Service. The javax.jws.WebService annotation specifies that a class implements a Web Service. The attributes that may be specified in the javax.jws.WebService annotation are discussed in Table 1. All of them are optional.

Add a Web Service method that returns a Hello message. By default all public methods are exposed as Web Service operations. Web Service methods may be annotated with the @WebMethod annotation. Attributes operationName and action may be specified in the WebMethod annotation. The operationName attribute specifies the name of the operation as mapped to the wsdl:operation element in the WSDL. Default value is the method name. For SOAP bindings, the action attribute maps to the SoapAction header in the SOAP messages.

Attribute	Description
name	Specifies the name of the Web Service. Maps to the wsdl:portType element in the WSDL file. Default value is the unqualified name of the Web Service implementation class.
targetNamespace	Specifies the XML namespace for the XML and WSDL elements generated from the Web Service.
serviceName	Specifies the service name of the Web Service. Maps to the wsdl:service element of the WSDL file. Default value is the Web Service implementation class name appended with "Service."
wsdlLocation	Specifies URL of a WSDL file. If specified a WSDL isn't generated for the Web Service.
endpointInterface	Specifies a service endpoint interface class.

Table 1 WebService Annotation Attributes

Deepak Vohra is a Sun Certified Java 1.4 Programmer and a Web developer.

dvohra09@yahoo.com

Ajay Vohra is a senior solutions architect with DataSynapse Inc.

ajay_vohra@yahoo.com

“JAX-WS is an API to create Web applications and Web Services using the XML-based Web Services functionality”

The HelloWsImpl Web Service implementation class is listed below.

```
package webservices.hello_webservice;
import javax.jws.WebService;

@WebService()
public class HelloWsImpl {

    public String hello(String name) {

        return «Hello «+name +» Welcome to Web Services!»;

     }

}
```

Copy the HelloWsImpl.java class to the C:\BEA\user_projects\webservices\hello_webservice\src\webservices\hello_webservice directory.

WebLogic Server 10 provides WebLogic Web Services Ant tasks to create Web Services and client classes. The jwsc Ant task takes an annotated JWS file as input and generates all the artifacts required to create a WebLogic Web Service. The generated files include the following.

1. Java Source files that implement a standard JSR-921 Web Service, such as the Service Endpoint Interface (SEI). For a JWS class HelloWsImpl an SEI HelloWsImplPortType.java gets created.
2. Standard and WebLogic-specific deployment descriptors. The standard webservices.xml deployment descriptor and the JAX-RPC mapping files get created. The WebLogic-specific Web Services deployment descriptor weblogic-webservices.xml also gets created.
3. The WSDL file that describes the Web Service.
4. The XML Schema representation of any Java user-defined types used as parameters or return values of Web Service methods.

A jws sub-element of the jwsc element specifies a JWS file. By default jwsc generates a JAX-RPC 1.1 Web Service. To generate a JAX-WS 2.0 Web Service specify the type attribute of the jws element as type=”JAXWS”.

Subsequent to generating the Web Service artifacts, jwsc compiles the JWS and Java files and packages the generated artifacts and classes into a Web application WAR file. Jwsc also creates an enterprise application directory structure. The Web Service may be deployed as a WAR file or packaged into an EAR file and deployed. Jwsc generates a WAR file corresponding to each jws element that’s a direct sub-element of the jwsc element. JWS files may be grouped by adding the jws elements to a module element, which is a direct sub-element of the jwsc element. If a module element is specified only one WAR file is generated.

In addition to the standard attributes some WebLogic specific attributes may be specified in the jwsc ant task. Only the srcdir and destdir attributes are required. The WebLogic specific jwsc attributes are discussed in Table 2.

A jws element specifies a JWS file to compile. A jws element may be specified as a direct sub-element of the jwsc element or be included in a module element, which is a direct sub-element of the jwsc element. The only required attribute of the jws element is file, which specifies the JWS file. Some of the other attributes that may be specified in the jws element are discussed in Table 3.

Next, create a build.xml file to generate the artifacts for a Web Service from the example JWS file. To the build.xml file add a taskdef element for the jwsc task that specifies the class for the jwsc task. Add a target to build the Web Service. Specify a jwsc task with srcdir as src and destdir as output/HelloWsEar. Using a jws element specify the

Attribute	Description
applicationXml	Specifies the name and path of the application.xml deployment descriptor. If the file already exists, the file is updated with Web Services. Jwsc also creates or updates the corresponding weblogic-application.xml file. By default jwsc creates or updates the META-INF/application file in the directory specified by destdir.
destdir	Specifies the output directory in which the Web Service artifacts are generated.
destEncoding	Specifies the character encoding of the output files, XML files, and deployment descriptors. Default encoding is UTF-8.
dotNetStyle	Specifies that the jwsc task generate a .NET-style Web Service. Applies to JAX-RPC Web Services.
keepGenerated	Specifies if artifacts be regenerated if previously generated. The default value is no, which implies that the artifacts be regenerated and the previously generated artifacts aren’t kept.
sourcepath	Specifies the pathname of the top-level directory that contains Java files referenced by the JWS file/s.
srcdir	Specifies the top-level directory containing the JWS file. The srcdir directory contains sub-directories corresponding to the JWS file package name.
srcEncoding	Specifies the character encoding of the input files. The default value is the character encoding set by the jvm.

Table 2 WebLogic-specific jwsc Attributes

Attribute	Description
contextPath	Specifies the context root of the Web Service. The default value is the same as the JWS file name.
file	Specifies the JWS file to compile. The JWS file is looked for in the srcdir directory.
name	Specifies the name of the WAR file including the .war extension. The default value is the JWS file name.
type	Specifies the type of Web Service. Valid values are JAXWS and JAXRPC. The default value is JAXRPC.
wsdlOnly	Specifies that only WSDL be generated. Applies only if jws is a direct sub-element of the jwsc element. The default value is false.

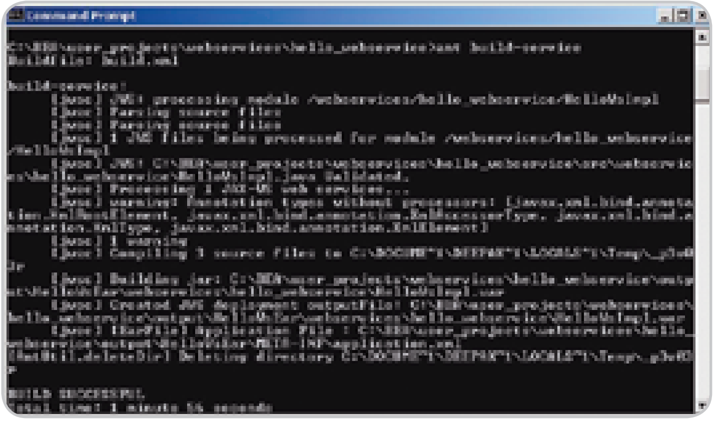
Table 3 jws Attributes

"Subsequent to generating the Web Service artifacts, jwsc compiles the JWS and Java files and packages the generated artifacts and classes into a Web application WAR file"

JWS file to be compiled. Specify the type attribute of the jws element as JAXWS as the Web Service is a JAX-WS 2.0 Web Service. The build.xml file is listed below.

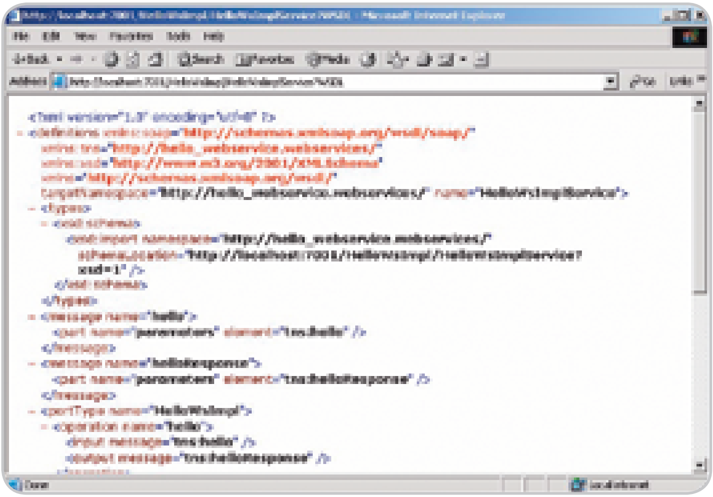
```
<project name="webservices-hello" default="all">
    <taskdef name="jwsc"    classname="weblogic.wsee.tools.anttasks.
JwscTask" />
        <target name="build-service">     <jwsc  srcdir="src"
                                destdir="output/HelloWsEar">
        <jws file="webservices/hello_webservice/HelloWsImpl.java"
type="JAXWS"       />
</jwsc></target></project>
```

Next, we shall run the build-service target to generate the Web Service artifacts. Run the build-service target with the following command.



```
C:\BEA\user_projects\webservices\hello_webservice>ant build-service
Buildfile: build.xml

build-service:
     [jwsc] JWS: processing module /webservices/hello_webservice/HelloWsImpl
     [jwsc] Parsing source files
     [jwsc] Parsing source files
     [jwsc] 1 JWS files being processed for module /webservices/hello_webservice
/HelloWsImpl
     [jwsc] JWS: C:\BEA\user_projects\webservices\hello_webservice\src\webservic
es\hello_webservice\HelloWsImpl.java Validated.
     [jwsc] Processing 1 JAX-WS web services...
     [jwsc] warning: Annotation types without processors: [javax.xml.bind.annota
tion.XmlRootElement, javax.xml.bind.annotation.XmlAccessorType, javax.xml.bind.a
nnotation.XmlType, javax.xml.bind.annotation.XmlElement]
     [jwsc] 1 warning
     [jwsc] Compiling 3 source files to C:\DOCUME~1\DEEPAK~1\LOCALS~1\Temp\_[illegible]
[illegible]
     [jwsc] Building jar: C:\BEA\user_projects\webservices\hello_webservice\outp
ut\HelloWsEar\webservices\hello_webservice\HelloWsImpl.war
     [jwsc] Created JWS deployment outputFile: C:\BEA\user_projects\webservices\
hello_webservice\output\HelloWsEar\webservices\hello_webservice\HelloWsImpl.war
     [jwsc] [EarFile] Application File : C:\BEA\user_projects\webservices\hello_
webservice\output\HelloWsEar\META-INF\application.xml
[AntUtil.deleteDir] Deleting directory C:\DOCUME~1\DEEPAK~1\LOCALS~1\Temp\_[illegible]
[illegible]

BUILD SUCCESSFUL
Total time: 1 minute 56 seconds
```

Figure 1 Output from the ant build-service Command



```
  <?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://hello_webservice.webservices/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://hello_webservice.webservices/" name="HelloWsImplService">
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://hello_webservice.webservices/"
          schemaLocation="http://localhost:7001/HelloWsImpl/HelloWsImplService?
          xsd=1" />
      </xsd:schema>
    </types>
  - <message name="hello">
      <part name="parameters" element="tns:hello" />
    </message>
  - <message name="helloResponse">
      <part name="parameters" element="tns:helloResponse" />
    </message>
  - <portType name="HelloWsImpl">
    - <operation name="hello">
        <input message="tns:hello" />
        <output message="tns:helloResponse" />
```

Figure 2

```
C:\BEA\user_projects\webservices\hello_webservice>ant build-service
```

The artifacts for a Web Service get generated and packaged into a WAR file HelloWsImpl.war in the directory C:\BEA\user_projects\webservices\hello_webservice\output\HelloWsEar\webservices\hello_webservice. An exploded directory structure for an EAR application including the application.xml and weblogic-application.xml also get generated. The output from the ant command is shown in Figure 1.

To deploy the Web Service copy the WAR file HelloWsImpl.war to the C:\BEA\user_projects\domains\base_domain\autodeploy directory. Start WebLogic with the command script C:\BEA\user_projects\domains\base_domain\bin\startWebLogic.

When the Web application is deployed the application server and the JAX-WS runtime generate the WSDL file and any additional artifacts required to invoke the Web Service from a client. The WSDL may be accessed with URL http://localhost:7001/HelloWsImpl/HelloWsImplService?WSDL.

Creating a Client

In this section we'll create a JAX-RPC Java client for the Web Service created in the previous section. We'll use the clientgen Ant task to generate the client component files. First, create a project directory for the client application.

```
C:\BEA\user_projects> mkdir \webservices\simple_client
```

Create an src directory under the project directory. The subdirectories of the src directory should correspond to the package name of the Java client class, which we'll create next

```
C:\BEA\user_projects\webservices\simple_client> mkdir src\webservices\
jaxws\client
```

Create a Java client application Main.java in package webservices.jaxws.client. In the Java client application create an instance of the HelloWsImplService service.

```
HelloWsImplService service = new HelloWsImplService();
```

Obtain a proxy to the service from the service using the getHelloWsImplPort() method.

```
HelloWsImpl port = service.getHelloWsImplPort();
```

Invoke the hello(String) method of the service.

```
result = port.hello("Deepak");
```

ICEfaces - Ajax for Java EE

SECURITY
ACCESSIBILITY
SCALABILITY
AJAX PUSH
JSF

PROFESSIONAL OPEN SOURCE SOFTWARE

Join us at **www.ICEfaces.org**

The Most Active Enterprise Ajax Community for Java Developers

Rich User Experience

Create a new class of collaborative and dynamic enterprise applications. Unleash the unique power of Ajax Push Technology to deliver server-initiated, instantaneous presentation updates. Easily migrate existing JSP-based and traditional client-server applications to RIAs.

Standards-Based

Develop and deploy scalable RIAs in pure Java using an integrated Ajax framework complete with rich JSF-based components. Harness the power of Ajax and leverage the entire standards-based Java EE ecosystem of familiar tools and runtime environments.

Performance, Scale and Security

Deploy rich enterprise applications with advanced Ajax connection management for maximum application reliability. Seamlessly scale applications across clustered Java EE servers. Extend the existing web security model and avoid transferring sensitive business logic and data to the client browser.

ICESOFT
TECHNOLOGIES INC.

Toll Free 1.877.263.3822
www.icesoft.com

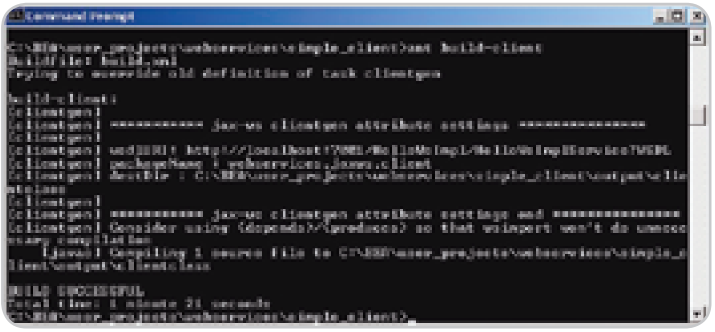
"WebLogic Server 10 Technical Preview supports JEE 5 and the JAX-WS 2.0 Web Services"

The Java client application Main.java class is listed below.

```
package webservices.jaxws.client;

public class Main {
   public static void main(String[] args) {
      HelloWsImplService service = new HelloWsImplService();
      HelloWsImpl port = service.getHelloWsImplPort();
     String result = null;     result = port.hello("Deepak");
     System.out.println(result);
  }
}
```

Next, we'll generate the client application artifacts required to invoke the Web Service using the clientgen task. The clientgen task generates the following artifacts.

1. The client-side copy of the WSDL file.



```
C:\BEA\user_projects\webservices\simple_client>ant build-client
Buildfile: build.xml
Trying to override old definition of task clientgen

build-client:
[clientgen]
[clientgen] ************ jax-ws clientgen attribute settings ****************
[clientgen]
[clientgen] wsdlURI: http://localhost:7001/HelloWsImpl/HelloWsImplService?WSDL
[clientgen] packageName : webservices.jaxws.client
[clientgen] destDir : C:\BEA\user_projects\webservices\simple_client\output\clie
ntclass
[clientgen]
[clientgen] ************ jax-ws clientgen attribute settings end ****************
[clientgen] Consider using <depends>/<produces> so that wsimport won't do unnece
ssary compilation
    [javac] Compiling 1 source file to C:\BEA\user_projects\webservices\simple_c
lient\output\clientclass

BUILD SUCCESSFUL
Total time: 1 minute 21 seconds
C:\BEA\user_projects\webservices\simple_client>_
```

Figure 3 The output from the build-client target

Attribute	Description
destDir	Specifies the directory in which clientgen generates the client application artifacts. Either destDir or destFile should be specified.
destFile	Specifies JAR file or exploded directory in which clientgen generates the artifacts. Clientgen also compiles all Java classes.
failonerror	Specifies whether clientgen should continue executing if an error occurs. The default value is True.
generateAsyncMethods	Specifies if clientgen should include methods in the generated stubs that may be used to invoke the Web Service operations asynchronously. The default value is True.
packageName	Package name into which the client interfaces and Stub files get generated. The default value is determined from the targetNamespace of the WSDL file.
serviceName	Specifies the service name in the WSDL file for which client artifacts get generated. Required if the WSDL file contains more than one <service> element. By default the only <service> element in the WSDL file is used.
type	Specifies the type of Web Service JAX-RPC 1.1 or JAX-WS 2.0. Valid values are JAXRPC or JAXWS. The default value is JAXRPC.
wsdl	Specifies the URL of the WSDL that describes the Web Service. Required attribute.

Table 4 clientgen Attributes

2. The Java source code for the Stub and Service interface implementations for the Web Service.
3. Java classes for any user-defined XML Schema data types defined in the WSDL file.
4. JAX-RPC deployment descriptor that describes the mapping between the Java data types and the corresponding XML Schema types in the WSDL file.

The only required attribute of the clientgen task is one of destDir or destFile and wsdl. Some of the commonly used attributes of the clientgen task are discussed in Table 4.

Create a build.xml file in the C:\BEA\user_projects\webservices\simple_client directory. Specify the class name for the clientgen task with the taskdef element.

```
<taskdef name="clientgen"
classname="weblogic.wsee.tools.anttasks.ClientGenTask" />
```

Add a target build-client to the build.xml file to generate the client artifacts. Add a clientgen element to the target element. Specify type as JAXWS, destDir as output/clientclass, packageName as webservices.jaxws.client, and wsdl as http://localhost:7001/HelloWsImpl/HelloWsImplService?WSDL.

```
<clientgen type="JAXWS"
wsdl="http://localhost:7001/HelloWsImpl/HelloWsImplService?WSDL"
destDir="output/clientclass" packageName="webservices.jaxws.client"/>
```

Add javac elements to the target element to compile the client artifact Java classes and the Main.java client Java application. The build.xml file is listed below.

```
<project name="webservices-simple_client" default="all">  <taskdef
name="clientgen"
classname="weblogic.wsee.tools.anttasks.ClientGenTask" />
<target name="build-client">    <clientgen type="JAXWS"
wsdl="http://localhost:7001/HelloWsImpl/HelloWsImplService?WSDL"
destDir="output/clientclass" packageName="webservices.jaxws.client"/>
<javac
srcdir="output/clientclass" destdir="output/clientclass"
includes="**/*.java"/> <javac
srcdir="src" destdir="output/clientclass"
includes="webservices/jaxws/client/*.java"/></target>
</project>
```

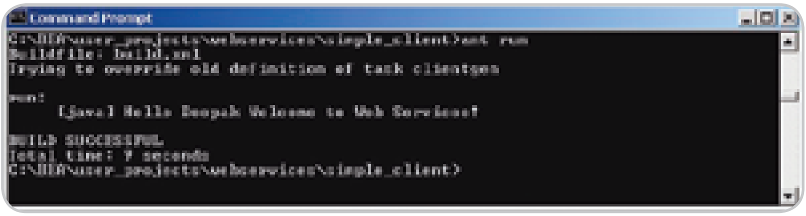
Run the build-client target to generate the client artifacts and compile the Java classes.

```
C:\BEA\user_projects\webservices\simple_client> ant build-client
```

The client artifacts get generated and the Java classes get compiled. The output from the build-client target is shown in Figure 3.

Next, add a target to the build.xml file to run the client Java application Main.java. Add a target run. Also add a path element to specify the classpath to run the Main.java application. The classpath should include the output/clientclass directory that contains the client artifacts and the system property java.class.path. The modified build.xml file is shown below.

```
<project name="webservices-simple_client" default="all">  <taskdef
name="clientgen"
classname="weblogic.wsee.tools.anttasks.ClientGenTask" />
<target name="build-client">    <clientgen type="JAXWS"
wsdl="http://localhost:7001/HelloWsImpl/HelloWsImplService?WSDL"
destDir="output/clientclass" packageName="webservices.jaxws.client"/>
<javac
srcdir="output/clientclass" destdir="output/clientclass"
includes="**/*.java"/> <javac
 srcdir="src" destdir="output/clientclass"
includes="webservices/jaxws/client/*.java"/></target>
<path id="client.class.path">
<pathelement path="output/clientclass"/>  <pathelement
path="${java.class.path}"/>    </path>  <target name="run" >
<java fork="true"
classname="webservices.jaxws.client.Main"
    failonerror="true" >
<classpath refid="client.class.path"/>  </java>  </target>
</project>
```



```
C:\BEA\user_projects\webservices\simple_client>ant run
Buildfile: build.xml
Trying to override old definition of task clientgen

run:
     [java] Hello Deepak Welcome to Web Services!

BUILD SUCCESSFUL
Total time: 7 seconds
C:\BEA\user_projects\webservices\simple_client>
```

Figure 4 WSDL

Next, run the target run.

The Web Service operation hello() gets invoked with a String argument and the result returned by the Web Service gets output as shown in Figure 4.

Conclusion

WebLogic Server 10 Technical Preview supports JEE 5 and the JAX-WS 2.0 Web Services. WebLogic server 10 also provides Ant tasks jwsc and clientgen to generate the Web Service artifacts and client artifacts.

THE | POWER | TO CREATE | REALITY™

Make Java Fly!

DiSTI, the global leader in human machine interface development, announces GL Studio for Java.

GL Studio for Java delivers state-of-the-art, 3D, fully interactive virtual reality—*as reusable Java components.* Whether you are a web developer, a simulation-based courseware designer, or need powerful tools to create maintenance training solutions, GL Studio for Java gives you the rich, dynamic content you need to reach entirely new heights of realism and interaction.

GL Studio for Java...
Giving *you* the power to create reality.

For a free trial download,
visit ***www.simulation.com/jdj***

DiSTI

www.simulation.com

11486 Corporate Blvd. Suite 190, Orlando, FL 32817 • PH: 407-206-3390

Introduction to Maven 2

A promising application development lifecycle management framework

by Murali Kashaboina and
Geeth Narayanan

Maven is a promising application development lifecycle management framework coming from Apache's armory of open source tools. Maven was originally developed as a framework to manage and mitigate the complexities of building the Jakarta Turbine project and soon became a core entity of the Apache Software Foundation project.

Without a uniform application development lifecycle management framework, different development teams would create their own build frameworks with varying flavors and complexity and this tendency would only proliferate as more and more new projects get developed. The creation of different build approaches for different projects would lead to build system disparity lacking reuse of build logic that would impede developers in moving easily between the projects because every time a developer starts working on a different project, the developer would spend too much time understanding the prevalent build system, its configuration, and usage instead of focusing on the core components.

This type of perplexity was particularly felt in the open source community. There was a definite need for a standardized project development lifecycle management system. The advent of Maven as part of the Jakarta Turbine project was the perfect remedy for the old malady.

As the name suggests, Maven is a connoisseur of build process. It encapsulates years of project development lifecycle management knowledge and tremendously simplifies the build process by extensively reusing build logic and eliminating most of the grunt work typical of the usual application development process. Ever since, Maven has been extensively used in the open source community for building projects and in the process was enhanced and extended to bring it to its current mature state. Maven, currently at version 2, has become the de facto build system in many open source initiatives and is being adopted by many software development organizations.

Development teams usually would have a plethora of challenges and concerns during typical application project development. The following are a few such examples:

- What should the project directory structure be?
- How should source, test source, libraries, configuration, documentation, and report directories be laid out?
- Where should the dependent library Jars be downloaded from?
- What versions of library Jars should be used for the project?
- What about the other Jars that the project library Jars depend on internally? Where should such Jars be downloaded from? What versions of such Jars should be used? Is there an easy way to know the compatible versions?
- What is the best way to resolve dependent library Jar version conflicts?
- Where should the library Jars be located?
- How should the project keep up with the latest versions of dependent Jar files?
- How should the compile time, runtime, and testing time classpath libraries be separated?
- How should compile time, runtime, and testing time resources be separated?
- Is there an easy way to execute all test cases during the build process and immediately evaluate percentage test coverage?
- Is there an easy way to test code quality during the build process?
- Is there a way to integrate code profiling during the build process?
- Is there a way to run integration tests during the build process? How can continuous integration be achieved by developing custom build scripts?
- How should the build scripts be designed for different project building tasks?
- Where should build scripts be located in the overall project directory structure?
- Should there be a dedicated resource to maintain the build scripts while the project is being developed?
- How can consistent company-wide Jar libraries be maintained?
- Should new team members learn the custom build process?
- Should each project have its own inconsistent and typically non-standard build process?

Maven thoroughly addresses such concerns by providing a common project build model that can be reused by all software projects. Maven abstracts the project structure and contents by following a set of guiding principles such as "convention over configuration," "declarative execution of development lifecycle process," "reuse of build logic across projects," and "logical organization of project dependencies."

Murali Kashaboina is a lead architect at Ecommerce Technology, United Airlines, Inc. He has more than 10 years of enterprise software development experience utilizing a broad range of technologies, including JEE, CORBA, Tuxedo, and Web services. Murali previously published articles in WLDJ and SilverStream Developer Center. He has master's degree in mechanical engineering from the University of Dayton, Ohio.

murali.kashaboina@united.com

The key benefit of this approach is that developers will follow one consistent build lifecycle management process without having to reinvent such processes time and again. Ultimately this will make developers to become more productive, agile, disciplined, and focused on the work at hand rather than spending time and effort doing grunt work understanding, developing, and configuring yet another non-standard build system.

Standard Conventions Used by Maven

Maven goes by the notion of "convention over configuration." Some of the common concerns when building a project are project directory structure, directory naming conventions, and the build output. For example, the directory structure of a Web application project will be slightly different from that of an EJB application project. Similarly the output of a Web application project is typically a WAR file while that of an EJB application is a JAR file. However, for a specific project type, the typical requirements in terms of directory layout and naming conventions are almost the same. Without a unified framework such as Maven, developers would typically spend time configuring such nitty-gritty details as setting up directories for source, resources, test case source, testing time resources, classes, and project dependencies. Moreover, developers will have to spend a good chunk of time creating build scripts such as ANT scripts to execute build tasks according to the project layout. This entire endeavor ends up being chaotic and in a large-scale project it can lead to a maintenance nightmare demanding dedicated resources just to focus on such build aspects.

Maven inculcates three main conventions to address common concerns:

1. **Projects of the same project type will have one common standard project directory structure:** At project creation, Maven uses a standard project directory layout for source files, resources, test case source files, test resources, configuration files, output files, reports, and documentation. In almost all cases, this standard project directory layout is sufficient to carry out development tasks. However, a custom directory structure can also be configured by overriding Maven's defaults. This override is not generally recommended unless there's a compelling reason and will deviate from Maven's best practice propositions.
2. **Every project results in one primary artifact of specific type:** Every project in Maven will result in one primary output file known as an artifact. For example, a Maven project containing a mathematical utility API will yield a JAR file containing compiled utility classes. The output JAR file is the resulting artifact of that project. Some other common artifact types are WAR, EAR, and RAR. Each artifact in Maven is uniquely designated by three Maven coordinates; artifact Id is the actual name of the artifact, group Id is the name of the group the artifact belongs to and the artifact version. This convention helps tremendously while resolving dependencies because every dependency in Maven is an artifact and so every dependency can be uniquely identified. This convention enables developers to think in terms of modularization at the code base level so that each project module yields one specific artifact specializing in one area of concern. This type of modularization encourages maximum reusability with different projects can now depend on only one functionally specialized and distinct artifact without having to include multiple disparate artifacts that may contain pieces of the required functionality.
3. **Use of standard naming conventions:** Maven uses standard names for project directories and output files. For example, Maven creates a standard 'projectDirectory/src/main/java' directory for all Java source files and 'projectDirectory/src/test/java' directory for all Java test case source files. Similarly, while creating a project artifact, Maven follows a standard naming convention such as 'artifactName-version.' An artifact version is typically represented in a standard format of 'MMM.mmm.bbb-qqqqqqq-nn' where 'MMM' is the major version number, 'mmm' is the minor version number, 'bbb' is the bug fix number,' 'qqqqqqq' is an optional qualifier, and 'nn' is an optional build number. Such naming conventions offer immediate clarity and in the case of artifacts, enable cohesive and consistent organization of dependencies using their respective artifact coordinates.

Overview of the Maven 2 Working Model

Apart from the standardization of project structure and contents using conventions, Maven follows a set of in-built core competencies in pursuit of simplifying the project development build lifecycle.

- **Definition of build lifecycle phases:** At Maven's core is the definition of build lifecycle phases. It's a fact that almost all software projects follow a series of steps during software building typically varying the number of steps and their sequencing. Maven standardizes these steps in pre-defined lifecycle phases and drives the build process through each phase in a specific logical sequencing order. Each phase can perform one or more actions or goals relevant to that lifecycle phase. Figure 1 captures Maven's phases and the logical order in which they are invoked.

The phase-specific actions and goals are accomplished by configuring Maven plug-ins. A Maven plug-in is a software extension that can execute one or more tasks. Each task is known as a Mojo and each Mojo tries to accomplish a certain build goal as part of executing a specific task.

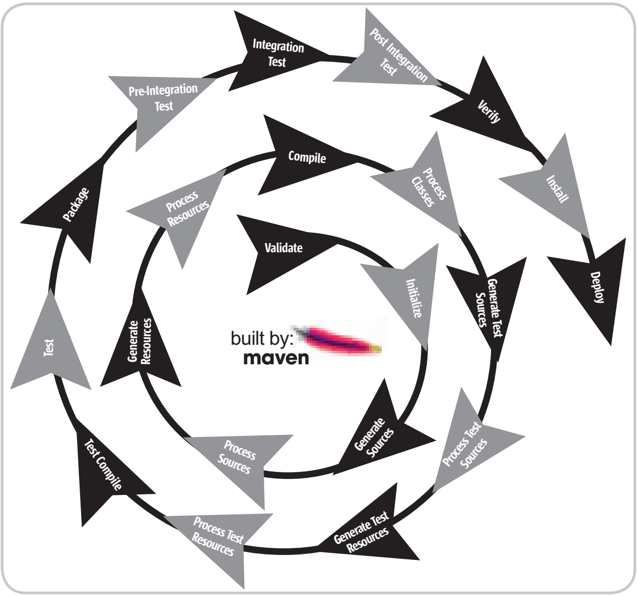


Figure 1

Geeth Narayanan is a senior architect at Ecommerce Technology, United Airlines, Inc. He has 10 years of experience in the IT industry, specializing in solutions using Java EE technologies. Geeth has master's degree in electrical engineering from the University of Toledo, Ohio.

geethakrishn.narayanan@united.com

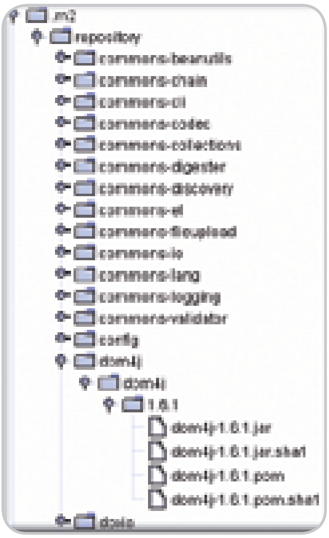
For example, the JAR Mojo from Maven JAR plug-in creates a JAR file during the packaging phase of the Maven build lifecycle. With this kind of build choreography in place, different projects can take advantage of different combinations of phases with appropriate plug-ins bound to them to accomplish project specific build lifecycle.

- **Declarative execution and Project Object Model:** Everything seems to be simplified using Maven because the Maven engine is driven in a declarative fashion. Project-specific metadata is declared in a descriptor file known as the Maven Project Object Model a k a POM. The project's POM file forms the cornerstone for executing Maven's build process. POM contains all the information required about the project such as project packaging type, output artifact coordinates, dependencies and dependency management information, and plug-in configuration information. Listing 1 is a sample POM file for a Maven project aimed at building a RAR file.

The sample POM in Listing 1 is for a JCA RAR project that will result in a 1.0 version of the JCA adapter. The sample POM states that the JCA project is dependent on two other artifacts, 'geronimo-spec-j2ee-connector-1.0-M1.jar' and 'junit-3.8.1.jar.' This POM also configures two plug-ins, 'maven-jar-plugin' and 'maven-rar-plugin,' and binds them to the 'package' phase of the Maven build lifecycle.

The sample POM in Listing 1 seems very simple in its configuration because in principle Maven shields the actual project POM from knowing all the other nitty-gritty default information by capturing such information in a Super-POM. When executing the build cycle, Maven implicitly associates the parent Super-POM with the project POM and thereby aggregates the overall project information needed to carry out the build process. Per se, every project POM has a parent Super-POM and so the project POM inherits lot of default information from the parent Super-POM. For more information on POM file contents, see http://maven.apache.org/pom.html.

- **Logical organization of dependencies and dynamic dependency resolution:** As shown in the sample POM, all project dependencies and plug-ins are declared in the POM file. In the Maven world, each dependency and each plug-in is an artifact that can be uniquely identified using its coordinates namely group Id, artifact Id, and version number. As far as developers are concerned, their responsibility in terms of defining the project dependencies ends with the declaration of the dependencies in the POM file. Developers only deal with the declaration of logical dependencies without either physically downloading the dependencies to the project directory or setting the classpath to point to some location where dependent JARs are available. Everything else in terms of resolving the dependencies and setting dependent artifacts in the build classpath is transparently handled by Maven using its in-built dynamic dependency resolution mechanisms.



```
.m2
  repository
    commons-beanutils
    commons-chain
    commons-cli
    commons-codec
    commons-collections
    commons-digester
    commons-discovery
    commons-el
    commons-fileupload
    commons-io
    commons-lang
    commons-logging
    commons-validator
    config
    dom4j
      dom4j
        1.6.1
          dom4j-1.6.1.jar
          dom4j-1.6.1.jar.sha1
          dom4j-1.6.1.pom
          dom4j-1.6.1.pom.sha1
    doxia
```

Figure 2

All artifacts are stored in Maven repositories. There are two kinds of Maven repositories – local and remote. Each repository stores artifacts in a logical structure using artifact coordinates. When Maven is installed, a local repository is typically created in the user home directory and is located at ~/.m2/repository. Figure 2 is an example local repository directory structure.

In the local repository example, the dom4j artifact is stored in a logical fashion using its group Id 'dom4j', artifact Id 'dom4j,' and version number '1.6.1.' In the dom4j/dom4j/1.6.1 directory, the actual artifact JAR file along with Maven's POM file is stored.

The Maven 2 remote repository is located at http://repo1.maven.org/maven2 and is a central repository for all kinds of open source artifacts. There are some other World Wide Web-accessible repositories such as the Codehaus repository located at http://repository.codehaus.org/ that contains artifacts for upcoming projects such as JRuby. A development organization can have its own remote repositories located in the company's intranet to store company-specific project artifacts. Figure 3 shows the repository layout for the central Maven repository.

By default, Maven uses http://repo1.maven.org/maven2 as the central remote repository at the time of dependency resolution. However, for a given project, additional remote repositories can be specified in the POM file as shown in Listing 2.

At the time of the build process, Maven resolves the dependencies specified in the POM file in the following order:

1. Maven checks if the dependency artifact is present in the local repository using artifact coordinates. For example, the following POM snippet includes version 1.6.1 of dom4j as a dependency. To resolve this dependency, Maven will check if '~/.m2/repository/dom4j/dom4j/1.6.1/dom4j-1.6.1.jar' exists. Typically Maven will look for the dependency in the local repository using the location path built using the artifact coordinates – '~/.m2/repository/groupId/artifactId/version/artifactId-version.extenstion' where the extension by default is '.jar' unless the artifact type is specified in the dependency declaration using a '<type>' element.

```
<dependency>
                    <groupId>dom4j</groupId>
                    <artifactId>dom4j</artifactId>
                    <version>1.6.1</version>
                    <scope>runtime</scope>
</dependency>
```

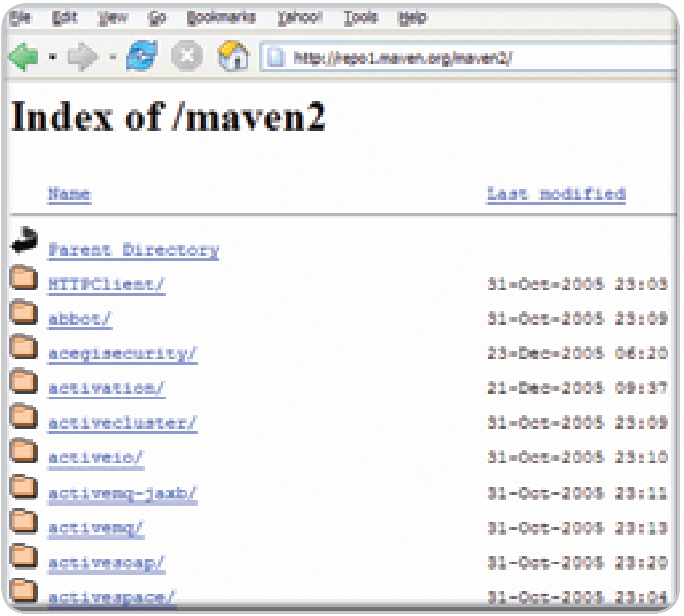
   If the group Id contains qualifiers separated by a period as in a company domain, Maven will include the qualifiers in the location path for the dependency artifact. For example, the location path '~/.m2/repository/com/ibatis/ibatis-sqlmap/1.3.1/ ibatis-sqlmap-1.3.1.jar' will be used to check if version 1.3.1 of the 'ibatis-sqlmap' artifact that has a group id of 'com.ibatis' is present in the local repository.

2. If a dependency artifact isn't found in the local repository, Maven will attempt to download the artifact from the remote repository to the local repository. As in step 1, Maven will build the path using the artifact coordinates and the URL for the remote repository. For example, Maven will use the http://repo1.maven.org/maven2/dom4j/dom4j/1.6.1/dom4j-1.6.1.jar URL path to download version 1.6.1 of the dom4j artifact from the central Maven 2 artifact repository. The remote lookup for the artifact will start with the first configured remote repository in the POM. If the lookup fails with a '404 Not Found,' Maven will search for the artifact in the next configured remote repository. The purpose of downloading the artifacts to the local repository is to make the process more efficient by not having to look up remotely time and again. Thus the local repository acts as a local cache for all the artifacts and one place to go to for all artifact shopping. Note that Maven will also download other information such as the POM file for the dependency.

3. If the dependency can't be resolved in any of the repositories, Maven will error out the build process.
4. Once the dependency is resolved by downloading the artifact, if necessary, Maven will include that artifact in one of the Maven classpaths based on the scope of the artifact specified in the dependency declaration in the POM.
   a. **compile**: A compile-scope dependency is available in all phases. This is the default value for a dependency.
   b. **provided**: A provided dependency is needed at the time of compiling the application but need not be part of the deployment because such a dependency may be made available by other means during runtime such as an application server providing necessary container-specific classes etc.
   c. **runtime**: A runtime-scope dependency isn't needed for compilation but is needed at the time of execution.
   d. **test**: A test-scope dependency is needed for compiling and running test cases.
5. Once the primary dependency is resolved, Maven will attempt to resolve all other artifacts that the primary dependency may internally depend on. This is commonly referred to as Maven 2 transitive dependency resolution. To achieve this, Maven will read the primary dependency's POM file found in the local repository alongside the artifact's JAR file, identify its dependencies, and go about resolving those dependencies using the same approach by repeating the steps from #1.

Conclusion

In Part 1 of this article, we introduced Maven through a quick tour of some of its core internals without delving into details of how to exercise Maven. We think that without understanding the basics first, it will be hard to sail into the nitty-gritty practicalities. In Part 2, we will get into the details of installing Maven, configuring the Maven plug-in in Eclipse, and providing a practical example that illustrates how to accomplish typical development tasks and how Maven simplifies the process, making development much more productive.



Index of /maven2

```
    Name                         Last modified

    Parent Directory
    HTTPClient/                  31-Oct-2005 23:03
    abbot/                       31-Oct-2005 23:09
    acegisecurity/               23-Dec-2005 06:20
    activation/                  21-Dec-2005 09:37
    activecluster/               31-Oct-2005 23:09
    activeio/                    31-Oct-2005 23:10
    activemq-jaxb/               31-Oct-2005 23:11
    activemq/                    31-Oct-2005 23:13
    activesoap/                  31-Oct-2005 23:20
    activespace/                 31-Oct-2005 23:04
```

Figure 3

Listing 1

```
<project>
     <modelVersion>4.0.0</modelVersion>
     <groupId>com.somecompany.apapters</groupId>
     <artifactId>jcaAdapter</artifactId>
     <version>1.0</version>
     <packaging>rar</packaging>
     <dependencies>
          <dependency>
               <groupId>junit</groupId>
               <artifactId>junit</artifactId>
                <version>3.8.1</version>
                <scope>test</scope>
          </dependency>
          <dependency>
               <groupId>geronimo-spec</groupId>
              <artifactId>geronimo-spec-j2ee-
connector</artifactId>
              <version>1.0-M1</version>
              <scope>provided</scope>
         </dependency>
     </dependencies>
     <build>
         <plugins>
              <plugin>

<groupId>org.apache.maven.plugins</groupId>
                   <artifactId>maven-jar-plugin</artifactId>
                   <executions>
                          <execution>
                           <id>jarCreation</id>
                           <phase>package</phase>
                           <goals>
                              <goal>jar</goal>
                           </goals>
                         </execution>
                  </executions>
            </plugin>
<plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-rar-plugin</artifactId>
                 <executions>
                       <execution>
                             <id>rarCreation</id>
                             <phase>package</phase>
                             <goals>

                                    <goal>rar</goal>
                              </goals>
                              <configuration>
                                     <includeJar>true</includeJar>
                               </configuration>
                        </execution>
                    </executions>
               </plugin>
            </plugins>
         </build>
    </project>
```

Listing 2

```
<repositories>
         <repository>
                  <snapshots>
                           <enabled>false</enabled>
                  </snapshots>
                  <id>central</id>
                  <name>Maven Central Repository</name>
                  <url>http://repo1.maven.org/maven2</url>
         </repository>
         <repository>
                  <id>codehaus </id>
                  <name> Codehaus Maven Repository</name>
                  <url>http://repository.codehaus.org/</url>
         </repository>
         <repository>
                  <id>xyzCompany</id>
                  <name>XYZ Company Maven Repository</name>
                  <url>http://intranet.xyz.com/repository/</url>
         </repository>
</repositories>
```

The Evolution of Java

Interviews with Mike Milinkovich and Bill Roth

Interview by Joe Winchester

Mike Milinkovich, executive director of the Eclipse Foundation, has been kind enough to answer some questions for Java Developer's Journal. Rather than rattle off the usual ones about the name, about why Swing wasn't used, or how much influence IBM still has, Mike has fielded questions on some more current and topical subjects, as well as given us his insights onto the future. Thanks for taking the time to talk to us Mike.

JDJ: *The Eclipse Foundation recently joined the Java Community Process. Can you tell us how this is going and what you expect to get out of this, as well as give to the JCP?*
Mike Milinkovich: Yes, we recently joined the JCP, as we also joined the OSGi Alliance and OMG. The reason for joining these organizations is that the Eclipse community relies heavily on the standards that are produced by these standards organization, so we wanted to show our support.

As for the JCP specifically, we are planning to contribute in a couple of different areas, the most immediate example being the use of the Eclipse Equinox code as the reference implementation for JSR291.

JDJ: *At this year's EclipseCon I felt that the amount of interest in RCP had surpassed the amount of interest in the actual IDE. Do you think this is the case, and if so does this change the dynamics of Eclipse's strategy and direction to become more of a general-purpose application platform and less of a development environment?*
Milinkovich: Yes, I agree the Eclipse community and the industry as a whole has moved toward viewing Eclipse as an application platform. We are seeing a lot of interest and adoption of RCP and also projects such as Equinox, RAP, and Higgins. However, this is not new as we have had a conscious strategy to move Eclipse beyond just being a Java IDE for several years now. I believe what we are seeing is quite simply that a number of the newer projects within the Eclipse community are becoming more mature and are enjoying greater interest and adoption as a result. The vision for Eclipse has always been about being a complete platform for software development and I think we are well on the way.

However, I do continue to see a lot of interest in Eclipse as a tools platform. We have new projects for providing IDEs for dynamic languages such as Ruby and PHP. The AJAX Toolkit Framework (ATF) is attracting a lot of interest as a tooling platform for AJAX developers. CDT, our C/C++ IDE, has great momentum as being the C/C++ IDE for embedded and Linux development. In my opinion, Mylar is one of the most innovative new developer technologies to come about in a long time.

Therefore, I don't really see a large change in strategy or direction; I see it more as a journey and evolution. This is what makes Eclipse such a vibrant and interesting community.

JDJ: *JSR 291 ratified the OSGi Java module mechanism used by Eclipse to become part of the Java language specification. Can you see the same occurring with SWT?*
Milinkovich: I haven't seen any interest from the community in putting SWT into the JCP process. It is not that I don't think it would be an interesting idea but someone would have to step up to spend the considerable amount of time required to take it through the process.

JDJ: *How do you manage the relationship between the Eclipse board members, some of whom are fierce competitors in the commercial marketplace, yet need to collaborate for the good of the platform?*
Milinkovich: Interestingly, this has been mostly a non-issue to date. I've been really very happy with how collegial and effective the Board of the Eclipse Foundation has been.

That said, I think you have already identified the answer; there is a common need for a strong, stable platform for building all sorts of different software. Eclipse is providing this platform and it really becomes a unifying force at the Board and through-out the community. The other thing is that the Eclipse governance model is proving to be very good at managing the different interests that participate in Eclipse. All the organizations have an equal say at the
Board level and the principles of meritocracy and transparency help resolve a lot of the issues within the projects.

Joe Winchester is a software developer working on WebSphere development tools for IBM in Hursley, UK.

joewinchester@sys-con.com

"The vision for Eclipse has always been about being a complete platform for software development and I think we are well on the way"

–Mike Milinkovich

JDJ: *What excites you most about what is going on with Eclipse at the moment?*
Milinkovich: Europa, our next release, is going to be pretty exciting. We have over 22 projects lined up for the annual release train, so a lot of new stuff is getting ready for release. Our annual release trains are very important for the entire Eclipse community. First of all, they are a real testament to the committer and project community's ability to deliver on a predictable schedule. Second, they are hugely important to our adopter community as they use the projects to deliver their commercial products or open source projects.

I think the growing involvement of Eclipse in the world of Equinox-based OSGi runtimes – such as the EclipseLink persistence project recently proposed by Oracle – is cool as well. Over the next 12–18 months we are going to see a lot of new stuff being built with Equinox.

JDJ: *What worries you most about what is going on with Eclipse at the moment?*
Milinkovich: I'm not sure that "worried" is the right word, but I would like to see vertical market frameworks (e.g., banking frameworks, health care frameworks) being developed as Eclipse open source projects. This will require enterprises to become more involved with contributing to open source projects and it is something that I think will take time. We have a great start with the Eclipse Open Healthcare Framework (OHF) but we need to do more to encourage large enterprises to begin collaborating in open source projects.

JDJ: *NetBeans seems to be gaining a lot of traction at the moment, especially with some of the emerging and BRIC markets. What is Eclipse doing in this space to keep up?*
Milinkovich: This is a funny question since we are finding the BRIC countries to be our highest growth areas. Our downloads from China are just exploding; Evans Data recently reported our usage grew 30% in India and 20% in Brazil. It is nice to hear that NetBeans is doing well but I certainly don't feel like we need to keep up. In fact, I believe we are leading in the BRIC countries.

JDJ: *There is talk that Eclipse 4.0 will be a major rethink of the platform and how it is put together. Can you share with us what's going on here?*
Milinkovich: To be honest, everything about Eclipse 4.0 is conjecture at this point. EclipseCon was the first time the idea had even been discussed by the community and not only are there no concrete plans yet for 4.0, there isn't even yet a firm commitment that it is going to happen. Of course, I think it will, but it is definitely too early for me to speculate what may or may not be in it.

What is going on at this point is the beginning of a process – a community process to decide whether doing a major new release – quite possibly with some API breakages – is the right thing to do for our broad ecosystem.

JDJ: *If you could do Eclipse all over again, what would you do differently and why?*
Milinkovich: First, please remember, I did not "do" Eclipse. A lot of people spent a considerable amount of time and energy establishing Eclipse long before I appeared on the scene in 2004. I have just been lucky enough to be involved in the implementation of the vision they set out.

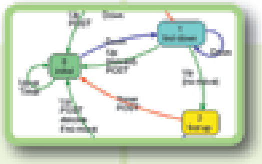
Things are working remarkably well, so I am not sure I would change anything. It is my belief Eclipse has become a place where many organizations can collaborate and innovate on interesting new technology. I truly believe Eclipse has the right technology, architecture, and governance to make it an important platform for future software innovation for many years to come.

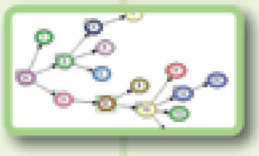
• • •

Bill Roth has had a distinguished career in Java, formerly being the JEE spec lead at Sun Microsystems where he worked on some of the early EJB specifications. In a previous life, Bill was a contributing editor to Java Developer's Journal. He maintains a lively blog and is currently vice president of the Workshop Business Unit at BEA. Bill has kindly agreed to answer some questions on BEA, Java, and all things interesting to do with the two.

Build interactive diagrams easier than you ever imagined

Create custom interactive diagrams, network editors, workflows, flowcharts and design tools. For web servers or local applications. It's easy-to-use and very flexible. We're the first developer of diagramming components and still the best. Find out for yourself: download our FREE fully functional evaluation kit, with full support at www.nwoods.com.





FREE Download With Full Support!

GoDiagram

Interactive diagram components

www.nwoods.com

JDJ: *Can you tell us a little about what's going on at BEA at the moment?*
Bill Roth: There is a lot going on in Java right now at BEA. First off, we are in the final stages of delivering the next generation of BEA Workshop. This will allow the construction of blended applications, the mixing of commercial source and open source. This will allow developers to take advantage of the best of both worlds, in the way they want. I call it our "Burger King" strategy, where developers can "Have it Your Way."

We're also working on an exciting new vision on how all the roles in IT can work together. This effort, WorkSpace 360, is a vision for unifying the various participants across the SOA life cycle.

WorkSpace 360 is intended to break down the communication barriers that exist between the various participants in the SOA life cycle: business analysts, architects, developers, and IT ops. It provides the ability for the various participants to share information and assets among each other in a seamless, governed manner. This is enabled through a series of tools, communication capabilities, and views tailored to the individual stakeholders. At the core of WorkSpace 360 is a centralized metadata repository that serves as a central source of record for enabling the seamless flow of information across the different stages of the SOA life cycle.

JDJ: *Recently there was some ruckus in the press about BEA failing to announce third-quarter 2006, which meant a possible delisting of their stock. What happened here, and is there any long-term damage?*
Roth: We are among the over 200 publicly traded companies who are reviewing their options granting practices. BEA has been, since the beginning of the investigation, working collaboratively with the SEC and the stock exchange, and we continue to keep them in the loop every step of the way as we make progress. From everything we have seen, NASDAQ has worked constructively to avoid delisting when the companies involved are working in good faith to resolve their issues and get financial statements back on file as quickly as possible. We believe we fall into this category.

JDJ: *For JRockit, do you see yourself as being able to compete commercially with Sun, whose JVM is freely available?*
Roth: Of course we can. Not only is JRockit freely available, but has been shown to be 24–28% faster than the leading JVM. We'll compete with Sun's JVM any day of the week.

JDJ: *The app server market is becoming largely commoditized with open source projects like JBoss, Tomcat, or Geronimo. How can WebLogic compete in this space and remain relevant?*
Roth: Your question is based on an erroneous assumption. While certain segments of the application server market are indeed commoditizing, saying the entire market is "largely commoditized" is overreaching. If this were the case, our WebLogic business would be shrinking, and it most certainly is not.

JDJ: *Is BEA aligned very strongly to Java as a server-side programming model, or are you embracing things like PHP?*
Roth: I have blogged about this in the past. While we're committed to the J2EE programming model, it's clear that developers are looking at other technologies. We have a number of projects where we're working with next-generation dynamic languages like PHP and Ruby, so we can be ready as the enterprise developers begin adopting them to build their applications.

JDJ: *What do you think of Web 2.0 and all the AJAX excitement? Is this something you're tooling for and adopting in WebLogic?*
Roth: AJAX is an exciting new area of great technology, but it is also an area of great chaos as well. There are way too many AJAX frameworks and no standards that are clearly emerging. Also, no one has clearly articulated a declarative, standards-based XAML-like way of defining the UI for this technology. As such, our strategy is to work with a small number of vendors in the short term to deliver value to our developers, and then keep an eye on the standardization process.

JDJ: *What excites you most about what is going on with BEA at the moment?*
Roth: What excites me the most about working at BEA is the pace of innovation and our plans for the next five years. When you couple the success of AquaLogic along with our vision for Workspace 360 and SOA 360, I am even more convinced than before that BEA is on the right track.

JDJ: *What excites you most about what is going on with Java at the moment?*
Roth: Same answer, really. The pace of innovation in the area of open source software and frameworks is really exciting. We're also seeing a shift in user requests to JSF from Struts, but it also appears that Struts 2.0 is picking up steam as well. What's truly unique is that the bulk of the innovation these days appears to be happening outside of the Java Community Process, and I view this as a good thing. The Java Community is growing organically in ways that are impossible to predict.

"WorkSpace 360 is intended to break down the communication barriers that exist between the various participants in the SOA life cycle: business analysts, architects, developers, and IT ops"

–Bill Roth

CALL FOR PAPERS NOW OPEN!

Web 2.0 and Rich Internet Apps

September 24-26, 2007

Santa Clara Convention Center
Hyatt Regency Silicon Valley
Santa Clara, CA

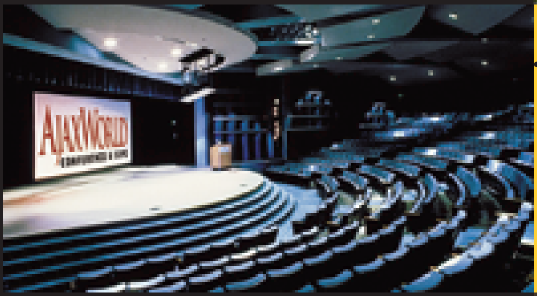
To Register
Call 201-802-3020 or
Visit www.AJAXWorld.com

Hurry! Limited Seating
This Conference Will Sell-Out!

Providing developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

Hyatt Regency Silicon Valley
Santa Clara, CA

On September 24-26, 2007 over 1,000 developers, architects, IT managers, and software professionals of every stripe will be converging in Santa Clara, CA to attend the West coast AJAXWorld Conference & Expo -- the most comprehensive meeting on the most significant technology subjects of recent times: AJAX, Rich Internet Apps & Web 2.0.



Join Over 3,600 Early AJAX Adopters Who Attended the #1 Web 2.0 Event in the World!

Experience AJAX, RIA, and Web 2.0 Knowledge and Best Practices at AJAXWorld Conference and Expo 2007

Delegates will hear first-hand from the creators, innovators, leaders and early adopters of AJAX, and a slew of leading vendors will provide sneak-peeks of the very latest frameworks, tools, products and applications during the conference, which will have over 100 sessions and presentations given by 125 different speakers - the largest AJAX-focused speaker faculty ever assembled in one place at one time.

AJAXWorld Conference & Expo 2007 West will be jump-started with a full one-day "AJAX University Boot Camp," followed by the Main Conference and Expo over the following 2 days.

Highlights Include:

- ***100+ conference speakers*** *selected from among the world's leading AJAX technologists and practitioners, including high-level IT executives, industry thought leaders, VCs and analysts*
- ***Topical, Interactive "Power Panels"*** *simulcast on SYS-CON.TV*
- ***Demos from 30+ vendors*** *both in General Session and in the Exhibit Hall/AJAX Showcase*
- ***Pre-Conference "AJAXWorld University Bootcamp"*** *for those wanting an all-day, hands-on encounter with AJAX*
- ***Leading-edge sessions*** *on issues like Offline AJAX, Mobile AJAX, Enterprise AJAX, AJAX Security, Desktop AJAX, Semantic Mash-Ups, Next-Generation SaaS, JavaScript & XML, Google Web Toolkit, Adobe Apollo, Adobe Flex, AJAX RIA GUIs, and Appropriate vs Inappropriate Use of AJAX.*

Early Bird Price: $1,495
(Register Before May 25, 2007)

SYS-CON EVENTS
For more great events visit www.EVENTS.SYS-CON.com

VISIT WWW.AJAXWORLD.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

Securing Tomcat Database Passwords

How encrypting the username and password can be implemented

by Michael J. Remijan

Tomcat is a great reference implementation of the Java EE specification and is intended for desktop use by developers who are starting to learn about Java EE or those who work on enterprise applications and need an EE server for development. However because Tomcat is free it finds its way into production environments. In this environment there are features of Tomcat that don't pass security audit reviews. One of these features is the use of clear text passwords in the server.xml file to create data sources. The purpose of this article is to show how encryption of the username and password can be implemented thus closing a potential security vulnerability.

Configuring a container managed data source with Tomcat is easy and well documented under the "JDBC DataSources" section of Tomcat's documentation (this article uses Tomcat 5.0.28). The data source configuration information is stored in TOMCAT/conf/server.xml. The resource is defined using the <Resource/> tag.

```
<Resource
  name="jdbc/TestDB"
  auth="Container"
  type="javax.sql.DataSource"
/>
```

The name attribute defines where the resource is bound in JNDI. The auth attribute will have either the value Application or Container. Container means Tomcat will provide the username and password to connect to the resource whereas Application means the application will provide them. Container is specified because the username and password will be entered in server.xml. The type attribute is the fully qualified name of the class returned when the resource is looked up using JNDI.

Next the resource needs to be configured. This is the purpose of the <ResourceParams/> tag. For a JDBC resource, a basic configuration is presented in Listing 1.

The name attribute has the value jdbc/TestDB, which must match a <Resource/> tag. The "factory" name/value pair tells Tomcat the fully qualified name of the class that implements javax.naming.spi.ObjectFactory. This class is responsible for returning objects of the type defined by the type attribute of the <Resource/> tag. The rest of the name/value pairs of <parameter/> tags are passed to the "factory" in a javax.naming.Reference object.

Finally, the application has to be configured to use this data source. This is done by referencing the data source in /META-INF/context.xml of the application's WAR. A <ResourceLink/> tag is added to context.xml.

```
<ResourceLink
  global="jdbc/TestDB"
  name="jdbc/SpiderMan"
  type="javax.sql.DataSource"
/>
```

The type attribute is the fully qualified name of the class returned when the resource is looked up using JNDI. The global attribute must match a <Resource/> tag in server.xml. The name attribute defines where the resource is bound in JNDI. From an application's point of view, the location specified by the name attribute is relative to java:comp/env. Each application can have its own conventions for JNDI lookup names and the <ResourceLink/> tag is the bridge between how an application does a resource lookup and where the resource is actually located. It's good practice to have the application use an atypical lookup name. This makes your application more portable to other EE servers because it's not accidentally bound to the conventions of one EE server's resource placement.

For development environments or personal use this configuration is acceptable, however, in a production environment the clear text username and password in <ResourceParams/> is a security vulnerability. If a production server is compromised, the intruder would have easy access to production data. With privacy concerns and Sarbanes-Oxley requirements the harder it is for an intruder to gain access to such data the better.

While Yahooing! I was surprised to discover I couldn't find any instructions addressing this issue. Most search results related to debating if securing the clear text username and password is even necessary. The general consensus seemed to be this security is not necessary for two reasons. One, if an intruder compromises a production server in a way that would allow read access to server.xml then the clear text username and password is the least security concern. Two, Tomcat is a reference implementation so it shouldn't be used in a production environment. Although these are lively debates and can spark great topics of conversation they do not address the issue. What I need is a way to get rid of the clear text username and password.

I will present three possible solutions to this issue. The first two are custom solutions involving application updates while the third takes advantage of Tomcat's built-in extensibility, requires no application updates and so is a much more attractive solution.

The first solution is to bypass Tomcat altogether and not configure a data source. With no data source there's no clear text username and password in server.xml so the problem is solved. However applications still need database connections. With no container-managed database pool each application will have to do its own pooling. This brings up all sorts of problems. Applications will need knowledge of the database and how to access it, unnecessary plumbing code is needed to manage the pool, data access objects will no longer be coded to EE specs making them less portable, no built-in or plug-in transaction management resulting in more plumbing code, an ever-growing number of connections as more applications are developed, performance degradation during high-traffic periods as connections become scarce, support overhead from managing many individual configurations, and the list goes on. It's possible each of these problems can be solved but as you solve them your code moves further and further from the EE specs. Because of all these problems, this is not an attractive solution.

The second solution is to move the responsibility for database authentication from Tomcat to the application. Recall the auth attribute of the <Resource/> tag. If the value is Application, the application is responsible for providing the username and password. The Tomcat administrator can delete the

Michael J. Remijan is a Senior Application Developer at Monsanto in St. Louis, MO. He designs and develops mission critical EE applications. Michael has a BS in Mathematics/Computer Science from the University of Illinois, MBA in Technology Mangement from the University of Phoenix. Sun Certified Web Component Developer. Pursuant Masters in Computer Science.

mjremijan@yahoo.com

“Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises
— Yafim Natis, Gartner Analyst

3-DAY EVENT!

SOAWorld 2007 Plus Enterprise OpenSource Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services	Open Source
› AJAX and SOA	› Open Source Business Models
› Web 2.0	› Open Source ESB
› Universal SOA	› OpenAjax Alliance
› Protecting Web Services	› SaaS and Open Source
› Troubleshooting SOA	› Spring, Hibernate and Eclipse
› Governance	› Seam
› Open-Source SOA	› Open Source Penetration
› XBRL	› Monetizing Open Source
› Service Virtualization	› Open Source Databases
	› AMQP
	› Open Source Middleware

June 25-27, 2007
Roosevelt Hotel / New York City

Register Online! www.SOAWorld2007.com

SOAEOSCONFERENCE.SYS-CON.COM
REGISTER ONLINE TODAY
SAVE $200!
(HURRY FOR EARLY-BIRD DISCOUNT)

11th International 2007 SOAWORLD™ CONFERENCE & EXPO

2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-26, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

BROUGHT TO YOU BY:

SOA Web Services Journal focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.

Enerprise Open Source Magazine EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Exhibit and Sponsorship Info:
Call 201-802-3020 or email events@sys-con.com

username and password parameters from server.xml and the Developer would code the username and password into the application. Typically the first time JNDI is used to look up a DataSource is when the username and password are set. The code in Listing 2 gives an idea of what this would look like for Tomcat. Although the example doesn't use any kind of encryption it's not difficult to plug in an encryption strategy. This solution also has problems. The application is now responsible for either knowing the username and password or knowing the encryption strategy. The application also needs to know the real class implementing DataSource (in the case of Commons DBCP) so the username and password can be provided. This could make the application not portable to another EE server. So this is not an attractive solution either.

The third solution is to take advantage of Tomcat's built-in extensibility. Inside the <ResourceParams/> tag Tomcat offers the ability to specify the fully qualified name of the Java class that implements the javax.naming.spi.ObjectFactory interface. This class is responsible for returning implementations of javax.sql.DataSource. By default Tomcat uses org.apache.commons.dbcp.BasicDataSourceFactory and this class requires <ResourceParams/> to have clear text username and password values. So why not create a class that extends BasicDataSourceFactory and is configured with an encrypted password instead? For the purposes of this article I'll use a simple Base64 encoding but the concept can be easily extended to any other method.

The source code for BasicDataSourceFactory has two important methods. They are:

```
public Object
  getObjectInstance(
  Object o
, Name n
, Context c
, Hashtable h)
```

```
public static DataSource
  createDataSource(
Properties p
)
```

Tomcat creates an instance of BasicDataSourceFactory by calling its no-argument constructor. When a DataSource is needed the getObjectInstance(…) method is called. The BasicDataSourceFactory class implements this method in the following way. First it typecasts the Object parameter to Reference. Then all of the name/value pairs are removed from the Reference object and set in a Properties object. Finally, the Properties object is passed to the createDataSource(…) method where it's assumed the username and password exist in the Properties object as clear text.

To secure Tomcat database usernames and passwords, create a new class named EncryptedDataSourceFactory that extends BasicDataSourceFactory. This new class is going to override the getObjectInstance(…) method. The new method is implemented in the following way. First it will remove the encrypted username and passwords from the Reference object. Next, it will decrypt them. Then it will put the decrypted values into the Reference object. Finally it will call the getObjectInstance(…) method of the super class so it can take care of creating the DataSource. See the source code in Listing 3. (Listing 3 can be downloaded from the online version of this article at http://java.sys-con.com)

In the source code the "username" string of the setUsername() method and the "password" string of the setPassword() method refer to the <name/> value of:

```
<parameter>
  <name>username</name>
  <value>postgres_user</value>
 </parameter>
 <parameter>
  <name>password</name>
   <value>postgres_pass</value>
 </parameter>
```

These strings have corresponding constants in the BasicDataSourceFactory class but the constants are declared private so can't be used. The find() method throws an exception if not found because decryption will fail if there's nothing to decrypt. The decrypt() method uses a Base64 decoder that isn't secure but adequately demonstrates the concept. Finally the replace() method removes the encrypted values and puts in the decrypted values. When getObjectInstance(…) of the super class is called, it has the clear text passwords and is completely unaware that the values in server.xml are actually encrypted.

Using EncryptedDataSourceFactory is simple. First drop the org.moss.jdj.jdbc-yyyy.mm.dd.x.jar file into TOMCAT/server/lib. Next, get a Base64 encoding of the username and password. A simple Yahoo! search of "online base64 encoder" will find sites that will do it. Finally, edit server.xml and replace the username and password values with their corresponding Base64 equivalents and set the factory value to org.moss.jdj.dbcp.EncryptedDataSourceFactory. Start up Tomcat and see it in action.

Using Tomcat's built-in extensibility like this is an attractive solution. It fulfills security audit requirements by removing clear text usernames and passwords but it also lets applications be coded to EE specs. Using this solution doesn't put any unnecessary or unwanted responsibility in applications. Applications can be developed to fulfill business requirements and don't have to worry about plumbing code like initializing or shutting down a database connection pool properly, maintaining a custom transaction system to roll back on errors, or implementing an encryption strategy. Plus, when the time comes to move away from Tomcat the applications will be ready. The code in Listing 3 will be thrown away but small amounts of throwaway code is much better than the effort needed to go back and update applications.

Listing 1

```
<ResourceParams name="jdbc/TestDB">
 <parameter>
   <name>factory</name>
   <value>
org.apache.commons.dbcp.BasicDataSourceFactory
   </value>
 </parameter>
 <parameter>
  <name>url</name>
  <value>jdbc:postgresql://localhost/db</value>
 </parameter>
 <parameter>
  <name>driverClassName</name>
  <value>org.postgresql.Driver</value>
 </parameter>
 <parameter>
  <name>username</name>
  <value>postgres_user</value>
 </parameter>
 <parameter>
  <name>password</name>
  <value>postgres_pass</value>
 </parameter>
</ResourceParams>
```

Listing 2

```
DataSource
  ds = (DataSource)ctx.lookup("java:comp/env/jdbc/SpiderMan");

org.apache.tomcat.dbcp.dbcp.BasicDataSource
  bds = (org.apache.tomcat.dbcp.dbcp.BasicDataSource)ds;

bds.setUsername("MyUsername");
bds.setPassword("MyPassword");

return bds.getConnection();
```

2007 VIRTUALIZATION CONFERENCE + EXPO
www.virtualizationconference.com

Register Now! EARLY-BIRD SAVINGS! $200

ROOSEVELT HOTEL
NEWYORK CITY
June 25-27, 2007

Virtualization: Solutions for the Enterprise.

Delivering The #1 i-Technology Educational and Networking Event of the Year!

As today's CIOs and IT managers rise to the challenge of using their enterprise computing infrastructure as cost-effectively as possible while remaining responsive in supporting new business initiatives and flexible in adapting to organizational changes, Virtualization has become more and more important.

A fundamental technological innovation that allows skilled IT managers to deploy creative solutions to such business challenges, Virtualization is a methodology whose time has come. The fast-emerging age of Grid Computing is enabling the virtualization of distributed computing, of IT resources such as storage, bandwidth, and CPU cycles.

But Virtualization can also apply to a range of system layers, including hardware-level virtualization, operating system level virtualization, and high-level language virtual machines.

Register Online!
www.VirtualizationConference.com

THE FIRST MAJOR VIRTUALIZATION EVENT IN THE WORLD!

HURRY, FOR EXHIBITOR AND SPONSORSHIP OPPORTUNITIES
CALL 201-802-3020

— BROUGHT TO YOU BY —
ENTERPRISE OpenSource MAGAZINE SOAWORLD MAGAZINE

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

–continued from page 6

Model-Based Testing for Java Applications

by Bill Hayduk

An issue of model-based testing is the skill set of the people using this approach. Classical manual testers and/or automated scripters will have a difficult time creating the framework needed to execute this kind of testing. Resources with in-depth knowledge of fully functioning programming languages (such as Java) are needed to code this approach. Either testers with exceptional programming skills or developers are needed to implement the model-based testing approach.

It's also difficult and time-consuming to define the correct inputs. Testing and development teams that have complete documentation to draw from will have an easier time developing inputs. Without such documentation, creating proper inputs can be more trouble than it's worth. One side benefit of model-based testing can be that it's an impetus for creating good documentation and coding guidelines. Establishing in-house guidelines for writing code enables others to easily check, work with, maintain, and reuse code. While each coder may prefer his or her own methods, personal programming idiosyncrasies, particularly of the undocumented sort, can baffle other developers – and perhaps even the original developer when they attempt to rework code at a later date.

Standardizing the development process doesn't mean it becomes stagnant. Java programmers have a wealth of best practices information to draw from, created and shared by members of the development community. Regarding security flaws, since most vulnerability attacks take advantage of common and well-known mistakes in programming code, complying with best practice development rules is a useful way to avoid potential problems. Each problem that's prevented by complying with best practice rules is one less problem that the team needs to identify, analyze, and correct later in the development process (or that will impact the deployed application.)

One of the benefits of model-based testing can be more thoughtful programming, since programmers and/or testers must understand in depth how an application is actually intended to work in order to develop model-based tests. Model-based testing can also hone more of an awareness of how actions interact and what effects these interactions have on code – simply thinking through the process of how to test an application can result in better coding.

All that said, only you can determine whether model-based testing is a concept that's likely to suit your project or company's culture. If you or your team already has a tendency to work under tight deployment deadlines, adding model-based testing to the mix may create unnecessary stress.

Input Issues

A good way to get started with model-based testing with your Java applications is to pick a small, easy-to-describe subset of your application to test. Choose five to 10 primary states, generate inputs for them, and run this basic test for at least a few hours – preferably a few days – to see what problems the test turns up. Remember that at this point you're primarily testing and refining your input-creation skills, even if the tests do turn up some bugs that had been hidden and especially if they don't it would be incorrect to assume you have the model-based testing process all worked out. Continue expanding the complexity of the subsets you test until you're confident that your inputs are correct, at that point you can begin to trust your test results.

When first starting to develop your inputs, you'll want to think about all the ways you intend the application to be used, and all the ways it can potentially be used and how you might attempt to abuse the application if you wanted to crack it. And since a big part of a Web-based application's success will be judged by its users on performance issues, consider testing scalability and reliability as well as looking for security flaws.

Thankfully model-based testing doesn't require programmers/testers to predict and fully define every single possible (or bizarre) interaction a user and an application might conjure up. Instead you will define the states, actions, and transitions for your application. So why did you have to think through how someone might use or abuse your application? Because having both a big picture and detailed view of how your application should and could act will help you define those inputs. Once you've defined them, the tool will generate the test cases.

Next you'll need to set initial test objectives. When you're just beginning to work with model-based testing this won't be an issue, since you'll be running very limited tests on small subsets of your application. But if you decide to incorporate this tool into your test process you'll need to clarify objectives before you begin testing. Since there are potentially an infinite number of tests that can be run in model-based testing, the best practice approach is to risk-prioritize the work. Decide which areas are likely to be at highest risk and then proceed to develop objectives for the test in descending order of perceived risk. As you run the tests, analyze the results and fix bugs, you'll continue to modify the model until you've met your testing objectives. Make sure to document what you've discovered and how the problem should be or was addressed.

In general, model-based testing can be a good way to augment the testing process that you already use. You'll want to put your application through other testing processes, obviously, with Java you typically need to ascertain how the application works in tandem with a Web browser, or how it functions in an embedded mobile system. You'll likely want to run scalability tests to see how the application holds up under heavy use. There are, as you've no doubt noticed, dozens of excellent tools – many of them open source – and verification programs available for Java application testing. Make use of whatever suits your application and its intended use. If you're developing an application for a client you may want to confer with them to see if they require or would prefer that you/your team performs other specific analysis of the application during the testing process. If your application is intended for the mobile market you may want to have it tested under the Unified Testing Criteria (www.javaverified.com), a set of mobile industry-approved automated tests that check Application Launch, Functionality, Operation, User Interface, Security, Network, and Localization abilities, so that you can use the "Java Powered" logo in your marketing.

Model-based testing is not a marvelous magic bullet that will quickly and effortlessly spot every single flaw in a program, nor will it necessarily produce results more economically or effectively than standard automated tests. Its ability to generate non-repetitive, real-world user tests is obviously of real value particularly in projects that tend to be more intricate such as real-time and embedded applications, assuming that good inputs can be produced.

JAVA DEVELOPER'S JOURNAL Advertiser Index

Advertiser	URL	Phone	Page
AJAXWorld Conference & Expo	www.ajaxworld.com	201-802-3022	27
Altova	www.altova.com	978-816-1600	Cover II
Business Objects	www.businessobjects.com/rareoccurrence	888-333-6007	8-9
DiSTI	www.simulation.com/jdj	407-206-3390	19
ICEsoft	www.icesoft.com	877-263-3822	17
Infragistics	www.infragistics.com/jsf	800-231-8588	7
InterSystems	www.intersystems.com/ja1p		Cover IV
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	33
JadeLiquid	www.webrenderer.com		13
Northwoods Software Corp.	www.nwoods.com	800-434-9820	25
Perforce Software	www.perforce.com		4
SOA and EOS 2007 Conference & Expo	www.soaworld2007.com	201-802-3020	29
Software FX	www.softwarefx.com	561-999-8888	Cover III
Virtualization Conference & Expo 2007	www.virtualizationconference.com	201-802-3020	31

The JCP and the OpenJDK Community

Onno Kluyt

At the JavaOne conference earlier in May, Sun launched the OpenJDK project (http://openjdk.java.net). The OpenJDK project is Sun's Java SE implementation under the GPL license. While portions of the project, such as the compiler and Hotspot, were released at an earlier time, at the JavaOne conference all the class libraries and other source code that together making up JDK7 were launched. At the same time, Sun also announced an interim Governance Board for OpenJDK. This board is comprised of two people from Sun and three from the community. Mark Reinhold (chief engineer for Java SE) and Simon Phipps (Sun's Chief Open Source Officer) both from Sun, and Dalibor Topic (of GNU Classpath and Kaffe.org fame), Doug Lea (professor at SUNY), and Fabiane Nardon (CTO of VIDATIS and JavaTools community manager) have graciously accepted invitations to serve on the interim board. The key task before them is to write a constitution for the OpenJDK community, have it ratified by the membership, and then put their seats up for re-election.

An oft-asked question during the conference was about any relationship between this Governance Board and the JCP Executive Committees and also between projects at OpenJDK and the JCP's JSRs. It occurred to me that others, in addition to the conference attendees, would have these questions and so I would like to demystify this here.

Onno Kluyt is director of the Community Growth group at Sun Microsystems and the Chair of the JCP.

onno@jcp.org

The role of the JCP was and is for the industry to develop and agree upon Java specifications. The OpenJDK project is Sun's open source project for their implementation of the Java SE and related specifications as they are produced by the JCP. The JCP Executive Committee (EC) votes to approve proposals for standardization. The OpenJDK Governance Board is the body of ultimate resolution for OpenJDK and the maintainer of the OpenJDK Constitution.

The OpenJDK Constitution will describe how the OpenJDK community makes decisions, who is considered a member, how projects are started, how contributions are made, and how these are accepted. In short, it will describe how the OpenJDK community operates. The (interim) Governance Board does not make technical decisions. That is done by the projects and groups in the OpenJDK community. The Governance Board does not decide on Java specifications. The JSR expert groups in the JCP work to define the draft and final Java specifications. The JCP Executive Committee then votes to formally approve these specifications.

The JCP's Executive Committees are intended to have broad industry and geographical representation. The EC members formal voting on JSR proposals thus lends industry credibility to the standardization efforts of the Java community. The OpenJDK community is a meritocracy in which developers collaborate with Sun engineers on the development of the JDK. As new participants in this community gain experience and reputation, their responsibilities and abilities will also increase, for example, taking up the role of committer or as project lead for a particular port of the JDK code base. I would expect that this rise in fame and responsibility may then also lead such individuals to be elected by the membership to the Governance Board.

While the OpenJDK Governance Board and the JCP Executive Committees have very distinct, and non-overlapping, roles I do expect there to be practical interaction and collaboration between the two communities. For example, I expect that most Java SE Spec Leads (i.e., leaders of JSRs that feed into SE or JDK releases) to be project leaders and code committers in the OpenJDK community, and probably a similar overlap between many key expert group members in the JCP and code contributors to the OpenJDK community. I also expect that the openness and transparency of the JDK source code under the GPL license will lead developers to provide feedback on the (draft) specifications that the code is based on, which will have a positive quality and progress impact on the JSRs in the JCP.

The interim Governance Board is just starting to draft the Constitution, and rough estimates are it will take 6–9 months to complete the effort and have the document ratified. In the meantime the OpenJDK community is open for business. Sun has established a set of guidelines for the community on how to start projects and make contributions while the interim board does its work. You can find links to the work of the board, these guidelines, and the OpenJDK Charter on the OpenJDK Web site at http://openjdk.java.net.

Conference Presentations, Magic Shows, and the Five-Ring Circus

by Joe Winchester, *Desktop Java Editor*

–continued from page 3

During this five-ring podium act it matters little what alphabet soup of technology is being showcased or whether it's server- or client-side scripting, just whether or not the demo works and whether the IDE jocks can keep the magic alive. Often the emcee on stage is a senior development manager who outranks the engineer who, while trying to stoke the IDE into life, will remark on how his pay review is coming up or, when it unfortunately bombs, how he's lost his bonus. The clear analogy here is to the monkey who is desperately trying to perform some tricks for his master, the organ grinder. Only if the audience is amused will the monkey be rewarded with peanuts.

Back to Basics

Conferences are costly to attend, difficult to put on, and involve thousands of people travelling thousands of miles on either their own, or their company's expense. I love going for the people I meet, the discussions I have, and the concentration of like-minded technical talent in the same venue for a few days. However, I'd like to see the whole presentation format go way back to basics and rely less on being a venue for travelling technology salesmen and have instead education sessions that have more in common with a good physics lesson in a high school classroom than a Las Vegas smoke and mirrors conjuring act. The conference presenters are often the top of their class in terms of intellectual talent and ideas, and are at the cutting edge of implementing, deploying, or understanding technology. Let's try in the future to get the best from their presence, rather than resort to having them perform circus tricks on stage.

TITLE UNIFICATION FOR ECLIPSE & NETBEANS INTEGRATION

☆SHOWDOWN☆

JOE "CHART FX" CODER

vs

DATA VISUALIZATION

A FULL 6.2 ROUNDS OF CHAMPIONSHIP HEAVYWEIGHT DATA VISUALIZATION

Chart FX
Undisputed Champion

BOX OFFICE
www.chartfx.com

A KNOCKOUT COMBINATION OF CHARTS, GAUGES & MAPS FOR JAVA

BROUGHT TO YOU BY
SoftwareFX

Call (561) 999-8888 for more information or download a trial version at www.softwarefx.com

Innovations by InterSystems

Make Java Applications More Valuable

Embed the world's fastest object database.
A golden opportunity to make Java applications richer.

When you embed Caché in your applications, they become more valuable. Caché dramatically improves speed and scalability while decreasing hardware and administration requirements. This innovative object database runs SQL queries faster than relational databases. And with InterSystems' JALAPEÑO™ technology for Java developers, Caché eliminates object-relational mapping. Which means Caché doesn't just speed up the *performance* of applications, it also accelerates their development. Caché is available for Unix, Linux, Windows, Mac OS X, and OpenVMS – and it is deployed in more than 100,000 systems ranging from two to over 50,000 users. Embed our innovations, enrich your applications.

InterSystems CACHÉ®

Download a free, fully functional, no-time-limit copy of Caché, or request it on CD, at **InterSystems.com/Ja1P**