SYS-CON

BUILDING A TOOLBAR FROM A MENU PAGE 54

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

JDJ

JDJ.SYS-CON.COM VOL.10 ISSUE:10

IN THIS ISSUE...

Deriving the Visitor Pattern
PAGE 26

Benefiting from Open Source Development
PAGE 32

JDBC – The Indispensable Component of Persistence Mechanisms
PAGE 44

Extreme MAKEOVER: ARCHITECTURE EDITION

Large-scale refactoring with Spring and Hibernate

RETAILERS PLEASE DISPLAY
UNTIL NOVEMBER 30, 2005
$5.99US $6.99CAN

PLUS...

- Bringing Together Eclipse 3.1, J2SE 5.0, and Tomcat 5.0
- Web Frameworks and IDE
- How to Provide Dynamic Security Permissions

Your potential. Our passion.™

Microsoft

JUNK MAIL.
MISSING THE TRAIN.
~~APPLICATION INTEGRATION.~~
HAIR LOSS.
BAD MUSIC.
PAPER CUTS.

WINDOWS SERVER SYSTEM WITH .NET MAKES APPLICATION INTEGRATION LESS PAINFUL.

The Microsoft® .NET Framework, an integral component of Windows Server System,™ is the development and execution environment that allows different components and applications to work together seamlessly. That means applications are easier to build, manage, deploy, and integrate. The .NET Framework uses industry standards such as XML and Web Services which allow enterprise applications to be connected to infrastructure of any kind, which removes many of the headaches of the integration process. Find out more about simplified application integration with Windows Server System and .NET: Simply get the Connected Systems Resource Kit at microsoft.com/connectedsystems

WHEN THINGS GET UNPLEASANT, JUST RELAX AND THINK OF EASIER APPLICATION INTEGRATION.

Microsoft® Windows Server System™

Subscriptions
For subscriptions and requests for bulk orders, please send your letters to Subscription Department:
888 303-5282
201 802-3012
subscribe@sys-con.com
Cover Price: $5.99/issue. Domestic: $69.99/yr. (12 Issues) Canada/Mexico: $99.99/yr. Overseas: $99.99/yr. (U.S. Banks or Money Orders) Back Issues: $10/ea. International $15/ea.

Editorial Offices
SYS-CON Media, 135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9638

Java Developer's Journal (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. Postmaster: Send address changes to: Java Developer's Journal, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

Pessimism Leads to Weakness, Optimism to Power

Jeremy Geelan

As I write this editorial, I am waiting, like hundreds and thousands of people worldwide, to hear whether (and if so, how) Sun and Google are going to be teaming up to take on Microsoft in its holiest of holy markets, the desktop.

Could such an alliance have been dreamed of just one year ago? The answer, of course, is "Yes!" Nothing has prevented Sergey Brin and Larry Page from nipping from Mountain View to San Jose long ago and dreaming up an imaginative combination of Google's brand recognition and Sun's desire to liberate desktop apps from the desktop and shift them to the network.

However, there is something about the prevailing mood of Q4 of 2005, just begun, that seems to be supremely conducive to a new kind of corporate symbiosis in which companies don't merely just gobble each other up, Oracle-PeopleSoft style, but genuinely seek to create various technology wholes that are greater than the sum of the technology parts.

We have Microsoft and JBoss teaming up to improve interoperability between the two companies' respective platforms, which is the closest yet Redmond has come to supping with open source, even if it is still with a long spoon. And now we have Sun and Google cozying up to one another, not for a merger or for an acquisition, but for something much more interesting: a game-changing strategy.

"Game-changing" is what a disruptive company like Google does best, and Sun for its part isn't a slouch when it comes to innovation, especially not since the arrival of Jonathan Schwartz with his charismatic (and highly unusual) combination of socially responsible entrepreneurial flair, business school acumen, and technological savvy.

The great American philosopher William James, leader of the philosophical movement known as Pragmatism, once wrote that "Pessimism leads to weakness, optimism to power." Optimism, in turn, can often be the crucible of inventiveness, of game-changing innovation, and Schwartz is, in these terms, an optimist par excellence, as are Brin and Page. Scott McNealy probably was an optimist, and still has power, but many in the industry sense some kind of weakness, the kind of weakness that most likely proceeds from pessimism, the same kind of pessimism that argues that technology has plateaued because the NASDAQ "is never again going to reach 5000."

Technology has never been healthier, but it takes a certain salt-water injection of optimism to see it, amid the growing worries (understandable worries) about the short-term economic effects of IT offshoring and the recurring grumbling that it's harder to make $10 million these days. So what if it is; how does that mean that technology is a sucked orange?

When Sun and Google make whoopee together, it's a clear sign that there's plenty of life yet in the desktop space; the rise and rise of Linux and open source show that on the server side things are just as competitive as ever; and the unstoppable surge of thin client and embedded technologies continues apace. So really truly, the only people making the argument that there's some kind of a technology slowdown are looking at stock prices, not at emerging technologies themselves.

Foremost among all emerging developments is something called "Web 2.0" – in which the network itself is the platform, spanning all connected devices. This is the paradigm that Schwartz's Sun, mark my words, will bestride (though not necessarily dominate Microsoft-style) in the next 10 years, much more than McNealy's Sun has prevailed over Web 1.0 in the last 10.

As the old joke has it, it's difficult to make predictions, particularly about the future, but some predictions are easier to make than others. Mine is that Sun is about to rise. Anyone with an interest in Web 1.0 will surely want to stick around to see just how Web 2.0 is going to shape up. It may take a while, but there will be some amazing new success stories, continuing more in the new-business mode of eBay, Google, and Amazon rather than the old-business mode of Microsoft and IBM.

Can Sun reinvent itself and shift from the latter category to the former? I believe, under Jonathan Schwartz's eventual (I would add, inevitable) leadership, that it can. Optimism, as James reminded us, leads not just to a more healthful outlook on business and on life.

It also leads to power.

Jeremy Geelan is group publisher of SYS-CON Media and is responsible for the development of new titles and technology portals for the firm. He regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas.

jeremy@sys-con.com

Altova® MapForce® 2006 – The premier data integration and Web services implementation tool.

ALTOVA® mapforce® 2006

Streamline Web services

Hook up with MapForce® 2006, and build Web services without writing any code

New in MapForce 2006:

- Drag-and-drop Web services implementation
- Advanced flat file parsing and integration
- Project-wide code generation
- Embedding in your applications via OLE / ActiveX

Altova® MapForce, the tool awarded for easy integration of XML, database, text, and EDI file formats, now also lets you implement Web services in a visual way. Simply drag connecting lines from information sources to targets and drop in data processing functions. MapForce converts data on-the-fly and auto-generates data mapping code in XSLT 1.0/2.0, XQuery, Java, C++, or C# for use in your data integration and Web services applications. Give your data direction!

Download MapForce® 2006 today: www.altova.com

Also available in the Altova XML Suite.

ALTOVA®

OCTOBER 2005 VOLUME:10 ISSUE:10

JDJ contents

JDJ Cover Story

Extreme Makeover: Architecture Edition
Large-scale refactoring with Spring and Hibernate
by Franz Garsombke
18

Features

Deriving the Visitor Pattern
by Nishanth Sastry

Benefiting from Open Source Development
by Sumitra Chary, et al.

FROM THE GROUP PUBLISHER
Pessimism Leads to Weakness, Optimism to Power
by Jeremy Geelan 3

VIEWPOINT
@ see Javadoc
by Xiaozhong Wang 6

GENERAL JAVA
Bringing Together Eclipse 3.1, J2SE 5.0, and Tomcat 5.0
Using Ant and JUnit
by Boris Minkin 8

YAKOV'S GAS STATION
Web Frameworks and IDE
Part Three
by Yakov Fain 30

GENERAL JAVA
How to Provide Dynamic Security Permissions
Two approaches
by Xiaozhong Wang 38

JAVA DATABASE
JDBC – The Indispensable Component of Persistence Mechanisms
It's critical for the success of the O/R infrastructure
by Jonathan Bruce 44

DESKTOP JAVA VIEWPOINT
The Usability Paradox
by Joe Winchester 48

TECHNIQUES
ArrayListModel
A convenient way to use a simple collection
by Phil Herold 50

JAVA DESKTOP
Building a Toolbar from a Menu
For better usability, versatility, and user friendliness
by Mauro Micalizzi 54

FEEDBACK
Letters to the Editors 58

JSR WATCH
2005 JCP EC Elections Are Under Way
Ratification ballot starts voting marathon
by Onno Kluyt 60

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. Postmaster: Send address changes to: JDJ, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

Viewpoint

Xiaozhong Wang
Guest Editor

@ see Javadoc

My dear wife has just started to learn to use Java in her work and asks me a lot of questions as she begins her journey in this wonderful language. To almost all her questions my answer is the same: "See Javadoc."

In addition to being a way of avoiding having to find or know the answer, it constantly surprises me what is available in the Javadoc and how useful a source of information it can be.

When I began to use Java in 1998 I received the pointer of the Java API Specification, or Javadoc, from a colleague. Despite the limitations of the JDK 1.1 Javadoc style when compared to Java 2, I immediately found that it was much more helpful than my other reference books. The books might help you get started on Java but, once you're serious, the Javadoc is something I know I can't live without.

The obvious benefit of using Javadoc is getting the most accurate and detailed description of Java APIs. Most likely the Javadoc is written by Java API implementers who know the behavior of the API inside-out. When the API changes, so does the Javadoc, whereas books, tutorials, and other sources lag behind. To know how a mehod works, asks the author who committed the doc and code together.

To ensure that different vendors' Java implementations uphold Java's fundamental tenant of "write once, run anywhere," Javadoc plays a part too. For any implementation to claim that it is compatible to a certain Java specification, it must pass the conformance test suite, sometimes called Technology Compatibility Kit (TCK). Each test in the TCK is strictly written according to the Javadoc, and nothing else, to make sure that the behavior of an implementation conforms. So if your application relies solely on behaviors described in Javadoc, you not only leverage the TCK effort but also have the maximum assurance that the application will work across different Java implementations. On the other hand, un-Javadoc'ed behaviors are likely implementation dependent and subject to change in future releases. If your application relies on any of these, the guarantee of "write once, run everywhere" may no longer hold.

Javadoc is a "living" document, as it is continuously being updated and maintained. At Sun, a lot of change requests of Javadoc are filed by development teams and customers during the development of a new release of a Java technology. These include requests for API semantic changes, behavior clarifications, or are as simple as grammar or spelling corrections. Regardless of how trivial it is, each change request to Javadoc is treated seriously. Some of them have to go through thorough reviews and approvals. Each new release of a Java technology is often accompanied by hundreds of Javadoc improvements, so by using Javadoc you leverage all these efforts.

Today's Javadoc is more complete than ever. The String.substring() story is just one example, but if you have other questions, such as "What is the behavior of this method if the parameter is null?" or "Does this method throw IllegalArgumentException or IndexOutOfBoundException if the index is invalid?", take a look at the Javadoc and seek the answers there.

Javadoc is by no means perfect. You may find that an API's description lacks certain aspects or is not clear enough. Should you feel that way, the developer who implements this API may have felt the same and chosen his own interpretation in the API's implementation. The result is different behaviors across different implementations. It is important therefore that, as an API consumer, you file Javadoc change requests at http://bugs.sun.com/services/bugreport/index.jsp. Remember, a more precise and definitive Javadoc will result in more conformant Java implementations, which in turn increase the chance that your applications can run across different platforms.

So what can you do with Javadoc? See it, rely on it, and improve it.

Xiaozhong Wang is a software engineer at Sun where he has solved some security problems in his TCK (Technology Compatibility Kit) work.

xiaozhong.wang@sun.com

SYS-CON MEDIA

President and CEO:
Fuat Kircaali *fuat@sys-con.com*
Vice President, Business Development:
Grisha Davida *grisha@sys-con.com*
Group Publisher:
Jeremy Geelan *jeremy@sys-con.com*

Advertising
Senior Vice President, Sales and Marketing:
Carmen Gonzalez *carmen@sys-con.com*
Vice President, Sales and Marketing:
Miles Silverman *miles@sys-con.com*
Advertising Sales Director:
Robyn Forma *robyn@sys-con.com*
National Sales and Marketing Manager:
Dennis Leavey *dennis@sys-con.com*
Advertising Sales Manager:
Megan Mussa *megan@sys-con.com*
Associate Sales Manager:
Kerry Mealia *kerry@sys-con.com*

Editorial
Executive Editor:
Nancy Valentine *nancy@sys-con.com*
Associate Editor:
Seta Papazian *seta@sys-con.com*

Production
Production Consultant:
Jim Morgan *jim@sys-con.com*
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*
Assistant Art Director:
Andrea Boden *andrea@sys-con.com*
Video Production:
Frank Moricco *frank@sys-con.com*

Web Services
Information Systems Consultant:
Robert Diamond *robert@sys-con.com*
Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Vincent Santaiti *vincent@sys-con.com*
Shawn Slaney *shawn@sys-con.com*

Accounting
Financial Analyst:
Joan LaRose *joan@sys-con.com*
Accounts Payable:
Betty White *betty@sys-con.com*
Accounts Receivable:
Gail Naples *gailn@sys-con.com*

SYS-CON Events
President, SYS-CON Events:
Grisha Davida *grisha@sys-con.com*
National Sales Manager:
Jim Hanchrow *jimh@sys-con.com*

Customer Relations
Circulation Service Coordinators:
Edna Earle Russell *edna@sys-con.com*
Linda Lipton *linda@sys-con.com*
JDJ Store Manager:
Brunilda Staropoli *bruni@sys-con.com*

The Developer Paradox:

No time to test your code? But long hours reworking it & resolving errors?

Check out Parasoft Jtest® 7.0
Automates Java testing and code analysis.
Lets you get your time back and deliver quality code with less effort.

- **Automated:**
  Automatically analyzes your code, applying over 500+ industry standard Java coding best practices that identify code constructs affecting performance, reliability and security.
  Automatically generates and executes JUnit test cases, exposing unexpected exceptions, boundary condition errors and memory leaks and evaluating your code's behavior.
  Groundbreaking test case "sniffer" automatically generates functional unit test cases by monitoring a running application and creating a full suite of test cases that serve as a "functional snapshot" against which new code changes can be tested.
- **Extendable:**
  Industry standard JUnit test case output make test cases portable, extendable and reusable.
  Graphical test case and object editors allow test cases to be easily extended to increase coverage or create custom test cases for verification of specific functionality.
- **Integrated:**
  Integrates seamlessly into development IDE's to support "test as you go" development, and ties into source control and build processes for full team development support.

To learn more about Parasoft Jtest or try it out, go to www.parasoft.com/JDJmagazine

PARASOFT® We make software work.™

Automated Software Error Prevention™

Bringing Together Eclipse 3.1, J2SE 5.0, and Tomcat 5.0

Using Ant and JUnit

by Boris Minkin

Eclipse is the most popular Open Source IDE on the Java market and the latest 3.1 release supports all the new language elements of J2SE 5.0.

In this article I'll show you see how to create a Web project that has Java classes located in different packages and how to use ANT to build this project and JUnit to test it. I assume that you have J2SE 5.0 installed and are familiar with Ant and JUnit.

Building Java 5.0 Applications with Eclipse

We'll build a servlet that will demo some new Java 5.0 features and do some basic tasks like creating a session to track user visits to multiple Web pages. This code can easily be extended to store a user ID in the session that will travel with her as the site is navigated. The value of the visit counter is stored in the session so multiple visits to the page will be counted (this includes the page refreshes after pressing the browser's "Reload" button). If the counter goes over five, the counter gets cleared.

Installing Tomcat

I've used Tomcat 5.0, which is available at http://jakarta.apache.org/tomcat/. Installation is a breeze, and it's smart enough to find your installation of J2SE 5.0. It also lets you test Java servlets easily on your own machine. Other that making you restart the server every time a new servlet is deployed, Tomcat is a great server for what we're doing and many other tasks.

Creating a New Project in Eclipse

To create our servlet, start by creating a new Java project by selecting the menus File, New, and Other. Then specify the name of the project, say, MyJavaProject, (make sure that J2SE 5.0 is selected as the default JRE), and click Finish. Actually, Eclipse lets you select a different version of the Java runtime at any time using the menus Windows, Preferences, Java > Installed JRE's. You can also set the compiler compliance level on the project level by selecting Java Compiler under the project's properties.

Since the standard J2SE 5.0 SDK doesn't support servlets, let's add the servlet.jar that comes with Tomcat. It's in its directory common\lib\servlet-api.jar. To add this external jar to an Eclipse project, right-click on MyJavaProject, select properties, Java Build Path, and under the Libraries tab add this external jar. Now our servlets will compile.

Eclipse by itself doesn't support debugging with Tomcat unless you add some custom Web development plug-ins to it, such as Web Tools Project (WTP). It's possible to debug Tomcat applications using Eclipse remote debugging capabilities. For more information see the Java Developer's Journal at http://java.sys-con.com/read/44918.htm.

Some of the New J2SE 5.0 Features

Let's create several supporting classes that will be used by our servlet and demonstrate some of the J2SE 5.0 capabilities. It's always a good idea to keep classes in separate packages based on functionality or some other criteria. Let's call the package "support." Just right-click on the project name, select "New Package," and enter the package name.

Autoboxing

Create a new class called Boxer that will support Java 5.0 enhanced boxing and unboxing operations. Right-click on the Java project, and select a New Class (un-check creation of the main method). Enter the source code of the class as shown in Listing 1 at the end of the article.

Just press Ctrl-S and the class is saved and compiled. As you can see, it has three methods for boxing, unboxing, and simplified adding to a collection.

Generics

Now create another class called Generic that shows how to eliminate the need for casting when retrieving elements from a collection by introducing generic collection types. You can specify the type of collection elements during its creation (see Listing 2).

Generic types let objects of the same class operate safely on objects of different types. For example, they provide compile-time assurances that a List<String> always contains Strings and a List<Integer> always contains Integers.

Eclipse can handle both generic and non-generic types:

- Generic types can be safely renamed.
- Type variables can be safely renamed.
- Generic methods can be safely extracted from or inlined into generic code.
- Code assist can automatically insert appropriate type parameters in parameterized types.

In addition, a new refactoring option has been added. "Infer Generic Type Arguments" can infer type parameters for every type reference in a class, a package, or an entire project.

Enhanced For Loop

The next useful feature of J2SE 5.0 provides support for the enhanced "for" loop. This spares us from creating an iterator, navigating it, and retrieving elements from it. Create this class in Eclipse in the same package support (see Listing 3).

Boris Minkin is a Divisional Vice President of a major financial corporation. He has more than 12 years of experience working in various areas of information technology and financial services. Boris is currently pursuing his Masters degree at Stevens Institute of Technology, New Jersey. His professional interests are in the Internet technology, service-oriented architecture, enterprise application architecture, multi-platform distributed applications, and relational database design.

bm@panix.com

Perforce.

The *fast* SCM system.

For developers who don't like to wait.

Tired of using a software configuration management system that *stops* you from checking in your files? Perforce SCM is different: fast and powerful, elegant and clean. Perforce works at your speed.

[Fast]

[Scalable]

[Distributed]

Perforce's lock on performance rests firmly on three pillars of design. A carefully keyed relational database ensures a rapid response time for small operations plus high throughput when the requests get big - millions of files big. An efficient streaming network protocol minimises the effects of latency and maximises the benefits of bandwidth. And an intelligent, server-centric data model keeps both the database and network performing at top speed.

It's your call. Do you want to work, or do you want to wait?

PERFORCE
SOFTWARE

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

Java printf Function

Now, C-lovers, you can use the printf function in Java so create one more class as in Listing 4.

Methods with Variable Number of Arguments (varargs)

Java 5.0 lets you create methods with a variable number of arguments as in Listing 5.

This method's signature is pretty similar to the public static String[] format(String str, Object[] args), however, it's more elegant to invoke since multiple arguments can be just passed on the command line rather than having to construct a whole new array.

Don't forget to create this class in the package support.

Static Imports

Allows importing static constants and methods, saves on extra mentioning of static classes (see Listing 6).

Creating a Servlet

Now that we've created all the supporting classes, let's create a servlet. First, we create a new package called servlet in our project.

Then, we can just create a SampleServlet class by selecting HttpServlet as a superclass in the Eclipse class creation window.

Servlets can handle all the HTTP protocol invocation types. GET and POST are the most common ones, through. Servlet code is in Listing 7.

As one can see from the code, the servlet first tries to create an Http session by calling request.getSession(true), which means that a new session will be created if one doesn't exist already, then we'll get the "sessiontest.counter" attribute from the session, which will be assigned 0 if it's null, increment it, and set it back to the session. The session will be invalidated (all attributes removed from it) when counter goes above five. We'll get an HTTP request header called "Cookie" and a reference to the HTTP response writer and pass them to the SampleProgram class (see Listing 8) that will perform the logic described below.

Creating a Test Class and JUnit Test Case

For our servlet to work, we'll create one more supporting class SampleProgram with a main method that can test all these supporting classes and will be eventually called by the servlet too.

As you can see, the program in Listing 8 uses all our classes. It can also be tested from the command line (without having to deploy a servlet in Tomcat) by creating a JUnit test case.

To create a JUnit test case in Eclipse, right-click on the class SampleProgram and select the menus New, Other, Junit, and Junit Test Case. It'll add the junit.jar to the classpath, if needed, and a window pops up asking how we want to create our test case.

Select setUp() and tearDown() methods to set up the test environment and tear it down when finished. Also select create main method and allow Swing UI so we can run our test visually. On the next screen select methods to test: main and testClasses.

Once finished, a SampleProgramTest class is generated and we can test it by selecting Run As – JUnit test from the context menu. JUnit will test to see if any exceptions occur and display them as errors. It will also display failures if any of our assertion tests fail. Below is the source code of our sample JUnit test (see Listing 9).

In both test methods, we've added method calls with arguments to make sure there are no exceptions. We also do an assertion that will definitely fail to demonstrate the JUnit failure detection capability:

```
junit.framework.ComparisonFailure: expected:<1>
but was:<2>
 at junit.framework.Assert.assertEquals(Assert.
java:81)
 at junit.framework.Assert.assertEquals(Assert.
java:87)
```

Coding Web.xml Deployment Descriptor

Web applications require a Web.xml deployment descriptor (see Listing 10), which should be put under the WEB-INF directory.

In our deployment descriptor, we've included our SampleServlet and the index.html as a welcome page.

Deploying Servlet Using ANT

Eclipse 3.1 comes with built-in ANT support so you can execute XML-based scripts to deploy your application in a specific application server. In our case, we're deploying under Tomcat 5.0 server, which also has built-in support for Ant for the following tasks:

- Deploying WAR files
- Reloading WAR files
- Undeploying WAR files

It makes deploying, redeploying, or undeploying an application easier. With Eclipse support for Ant, it's easy to create build an XML file with Ant tasks in it using Ant executor.

To take advantage of this support, we have to add Eclipse to the external catalina-ant.jar (which is located under the $TOMCAT\server\lib directory and contains support for Tomcat Ant tasks) to the runtime class-path before running Ant tasks. Select the menus Run, External Tools, and External Tools and double-click on the Ant-build.

Running Ant tasks is simple in Eclipse. Just right-click on your build file (you have to create a build.xml file containing your Ant tasks as in Listing 11) and select Run As – Ant Build, then select tasks to create-a war file, and deploy the application.

As you can see from Listing 11, there are four major tasks in the build:

- Creating the WAR file
- Deploying the WAR file to Tomcat
- Reloading the application
- Undeploying the WAR file from Tomcat

The file also contains variables used by these tasks. The notable ones are:

- ***"build"*** – where to build a WAR file
- ***"path"*** – the context root of the Web application, in this case "myapp"
- ***"url"*** – the url of the Tomcat administrative application (which comes with Tomcat and provides an interface for deploying/undeploying applications)
- ***username*** – the user name for the Tomcat admin application (in our case it's "admin")
- ***password*** – the password for the Tomcat admin application (in our case it's "admin")

The file also contains various "taskdef" entries that point to the Java classes inside catalina-ant.jar that define the execution of the particular tasks.

Using the Eclipse interface (right-click on build.xml file and select Run As – Ant Build…), these tasks can be executed one at a time or all together. Note that "Deploying the WAR file to the Tomcat" task is dependent on "Creating WAR file" task, so execute the "create-war" task first, then "deploy" task.

It's time for us to start the Tomcat server and deploy our application. In Windows, simply go to Control Panel – Services and start the Apache Tomcat service. In Unix, you can execute the $TOMCAT\bin\startup.sh script.

Once it's deployed (the ANT deployment task has been completed), just point your browser at http://localhost:8080/myapp/SampleServlet/ and you should see the output as in Figure 1.

THE MOST CUTTING EDGE DEVELOPMENT IN REPORTING SOFTWARE GIVES USERS SOME MUCH-DESIRED FREEDOM.

THE ONLY REPORTING SOFTWARE THAT LETS USERS DESIGN THEIR OWN REPORTS IN MSWORD. YES, REALLY.

Free Yourself from the Task of Report Design

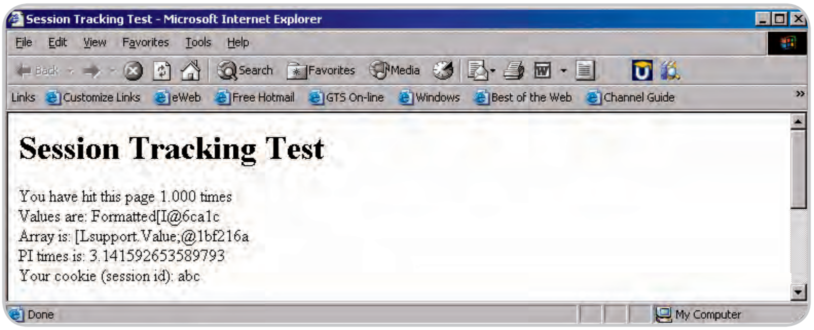
At last! There's a way for developers to rid themselves of the daunting and time consuming task of trying to design report templates that satisfy business owners and managers. That's because Windward Reports enables them to design their own reports using popular, easy-to-use word processing programs like Microsoft Word – so there is no new software to learn.

Point. Click. Done.

Windward Reports utilizes Microsoft Word, the word processing software program that virtually everyone knows (and frankly, it works with just about every other word processing program, as well.) All a business owner needs to do is open up Word's deep, rich library of design templates and configure the design that they want. So designing a report is now as easy as creating any MSWord document. Your data just flows in and completes the picture, creating reports that will dazzle and impress. It's really that easy.

Find Out for Yourself with a Free, No-Obligation Demo.
Go to www.windwardreports.com. You won't believe your eyes.

WindwardReports
advanced J2EE/.NET report generation


Session Tracking Test - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Links Customize Links eWeb Free Hotmail GTS On-line Windows Best of the Web Channel Guide
Done
My Computer

Session Tracking Test

```
You have hit this page 1,000 times
Values are: Formatted[I@6ca1c
Array is: [Lsupport.Value;@1bf216a
PI times is: 3.141592653589793
Your cookie (session id): abc
```

Figure 1 The servlet's output

Conclusion

In this article, I've shown you some of the new J2SE 5.0 elements and covered the creation of a Web application using simple tools available in Eclipse 3.1 and Tomcat 5.0, such as Java wizards, JUnit, and Ant. There are more advanced tools that can streamline and automate this development. One such extension to Eclipse is called Web Tools Project WTP (http://www.eclipse.org/Webtools), which I'll cover in a future article.

Listing 1: Class Boxer

```
package support;

import java.util.Collection;

/** This class supports a conversion between primitive types and
wrapper objects (and vice-versa) that's needed when adding primi-
tives to a collection
*/

public class Boxer {

 // boxes int type into the Integer object
 // note that no wrapping or class casting is needed
 public static Integer box(int i) {
  Integer obj = i;
  return obj;
 }


 // unboxes primitive type from the Integer object
 // note again that no wrapping is needed
 public static int unbox(Integer obj) {
  int i = (int)obj;
  return i;

 }

 // adds int primitive directly to collection
 // with auto-wrapping
 public static void add(Collection col, int i) {
  col.add(i);
 }
}
```

Listing 2: Class Generic

```
// Shows an example of using Java generics
package support;

import java.util.List;
import java.util.LinkedList;

public class Generic {
 // Return Generic List<Integer> object with one int eleemnt
 public static List<Integer> createGenericListWith(int g) {
  List<Integer> l = new LinkedList<Integer>();
  l.add(g); // Using autoboxing
  Integer i = l.iterator().next();
  return l;
 }
}
```

Listing 3: Class ForLoop

```
package support;
```

```
import java.util.Collection;
import java.util.ArrayList;
/**
 * Provides support for enhanced for loop
 *
 */

public class ForLoop {

 // convert the Collection of Integers to String collection
 public ArrayList<String> convertToString(Collection<Integer> c) {
  ArrayList<String> arr = new ArrayList<String>();
  for (Integer i : c) arr.add(i.toString());
  return arr;
 }
}
```

Listing 4: Using printf

```
package support;
import java.io.PrintWriter;
public class Printf {
 // print the number with decimal points
 public static void printWithDecimals(PrintWriter out, double q) {
  // Print the number with 3 decimal places.
  out.printf ("%5.3f %n", q);
 }
}
```

Listing 5: Variable arguments

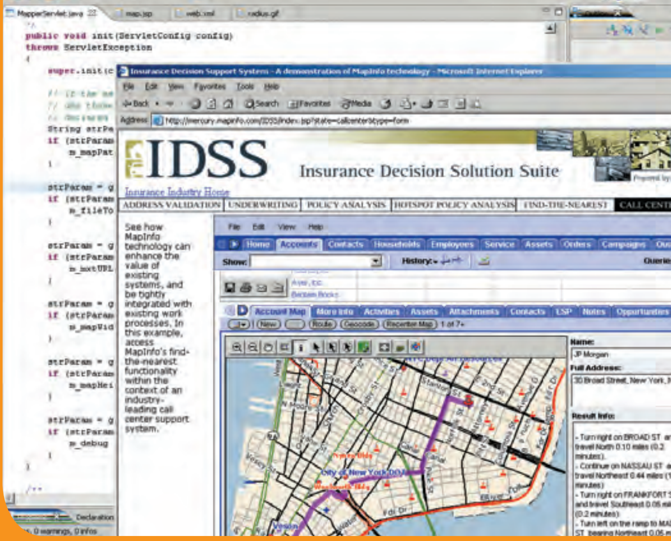
```
package support;
/**
 * Using a method with variable arguments
 */

public class Varargs {
 // method with variable number of arguments
 public static void format(java.io.PrintWriter out, String str,
Object... args) {
  for(int i=0; i<args.length; i++) {
   out.println("Formatted" + (args[i]));
  }
 }
}
```

Listing 6: Static Imports

```
package support;
/** Demonstrates static import capability
 */
import static java.lang.Math.*;
public class StaticImport {
 public static double pitimes(double d){
```

Listings continued on page 14


MapperServlet.java
map.jsp
web.xml
radius.gif
Insurance Decision Support System - A demonstration of MapInfo technology - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://mercury.mapinfo.com/IDSS/index.jsp?state=callcenter&type=form
IDSS
Insurance Decision Solution Suite
Insurance Industry Home
ADDRESS VALIDATION
UNDERWRITING
POLICY ANALYSIS
HOTSPOT POLICY ANALYSIS
FIND-THE-NEAREST
Home Accounts Contacts Households Employees Service Assets Orders Campaigns
Show:
History:
Account Map More Info Activities Assets Attachments Contacts LSP Notes Opportunities
New Route Geocode Recenter Map 1 of 7+
Name:
JP Morgan
Full Address:
30 Broad Street, New York,
Result Info:
Declaration
0 warnings, 0 infos

```
public void init(ServletConfig config)
throws ServletException
{
    super.init(c
    String strPa
    if (strParam
        m_mapPat
    }

    strParam = g
    if (strParam
        m_fileTo
    }

    strParam = g
    if (strParam
        m_extURL
    }

    strParam = g
    if (strParam
        m_mapWid
    }

    strParam = g
    if (strParam
        m_mapHei
    }

    strParam = g
    if (strParam
        m_debug
    }
}
```

See how MapInfo technology can enhance the value of existing systems, and be tightly integrated with existing work processes. In this example, access MapInfo's find-the-nearest functionality within the context of an industry-leading call center support system.

SUPERCHARGE YOUR APPS WITH THE POWER OF LOCATION INTELLIGENCE

100% Java SDK enables Location Intelligence through web services

Create desktop & web UIs using Swing, JSP and JSF Components

Integration with Eclipse, NetBeans, IntelliJ and more

Create applications for:

- Web-based store location finders
- Analyzing where revenue comes from – and where it doesn't
- Visualizing where your customers are
- Managing assets such as cell towers, vehicles and ATMs

Try it and see for yourself. **Visit www.mapinfo.com/sdk**
Learn more about the MapInfo Location Platform for Java, access whitepapers and download free SDKs.

MapInfo®
Be Location Intelligent™

Contact us at sales@mapinfo.com

```
  return d * PI;
 }
 public static double absoluteValue(double g) {
  return abs(g);
 }
}
```

Listing 7: Sampleservlet class

```
package servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import static support.Boxer.*;


/**
A servlet can create a session, which means that visits to multiple
pages within a site can be tracked. The counter's value is stored in
the session, so multiple visits to the page can be counted. If the
counter goes over five, it's cleared.
 */

public class SampleServlet extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse resp)

              throws ServletException, IOException {

 // Get the Session object from the request parameter (create it if
necessary)

 boolean create = true;
 HttpSession session = req.getSession(create);

// Get the session data value (null value means counter has never
been set)

Integer ival = (Integer)session.getAttribute ("sessiontest.counter");

 // convert the Integer object to an int primitive type
 int num = unbox(ival);
 num++;

 // convert the int primitive type to an Integer and set it as a
session attribute
 session.setAttribute ("sessiontest.counter", box(num));
 if (num > 5) {
   session.invalidate();
 }

 String cookie = req.getHeader("Cookie");


          // Write the output HTML page - just echo the counter's
value

      resp.setContentType("text/html");
      PrintWriter out = resp.getWriter();

      SampleProgram.performLogic(out, num, cookie);
 }
}
```

Listing 8: SampleProgram class

```
package servlet;

import static support.ForLoop.convertToString;
import static support.Generic.createGenericListWith;
import static support.Printf.printWithDecimals;
import static support.StaticImport.pitimes;
import static support.Varargs.format;

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;


/**
 * Sample program to be used by servlet
 *
 */
public class TestProgram {

 /**
  * To be called from the command line if needed
  */
 public static void main(String[] args) {

  if (args.length < 2) throw new RuntimeException("Usage: java serv-
let.SampleLogic number string");
  int num = Integer.parseInt(args[0]);
  String cookie = args[1];
  PrintWriter out = new PrintWriter(System.out, true);
  performLogic(out, num, cookie);

 }

 /**
  * Performs all the necessary logic to display proper output
  */
 public static void performLogic(PrintWriter out, int num, String
cookie) {

      List<Integer> list = createGenericListWith(num);
   ArrayList<String> alist = convertToString(list);

      out.println(„<html>");
      out.println(„<head><title>Session Tracking Test</title></
head>");
      out.println(„<body>");
      out.println(„<h1>Session Tracking Test</h1>");
      out.print („You have hit this page „);

      // example of using printf function
      printWithDecimals(out, num);
      out.println(„ times" + „<br>");

      String fmt = „{1,number,$'#',##}";
      out.println(„Values are: „);

      int[] intArr = new int[5];
      for(int i=0;i<5;i++) intArr[i] = i;
      format(out, fmt, intArr);
      out.println(„<br>");
      out.println(„Array is: „ + intArr + „<br>");
      out.println(„PI times is: „ + pitimes(num) + „<br>");

      out.println („Your cookie (session id): „ + cookie);
      out.println(„</body></html>");
 }
}
```

Listing 9: JUnit Test Program

```
package servlet;

import java.io.PrintWriter;

import junit.framework.TestCase;
```

Listings continued on page 16

Pure *Java*
Pure *Excel*
Power for users
Control for IT

Pure *Paradise*

Formula One e.Spreadsheet Engine

API-driven, embedded, 100% pure-Java, scalable spreadsheet toolset

Spreadsheets play an essential role in the business world. And now, they can also perform a vital function in your Java applications. How? With the Formula One e.Spreadsheet Engine, an API-driven, 100% pure Java toolset for embedding Excel spreadsheet functionality in your applications.

Excel-enable your Java applications

Use a state-of-the-art graphical development environment to build fully-formatted Excel report templates that include formulas, functions, colors, merged cells, even charts. It's fast, easy and effective.

Embed live, Excel-compatible data grids

Include interactive Excel forms and reports in your Java applications. Let users manipulate them using their spreadsheet skills and then commit the changes to databases or applications – or save them as an XLS file on their desktops.

Automate complex calculations and rules

Embed Excel spreadsheets that automate complex calculations and business rules on J2EE servers. Stop translating spreadsheet logic into Java code today, and start leveraging the development skills of your organization's spreadsheet experts.

Read and write server-based Excel spreadsheets

Give your users server-based spreadsheets populated with up-to-the-minute information directly from data-bases, XML files and enterprise applications. Get control of runaway data warehouses and spreadmarts now.

Make spreadsheets a part of your strategies

Visit us today at **www.reportingengines.com** to request a **free trial** of the e.Spreadsheet Engine. We'll show you how to make Excel spreadsheets a vital and productive part of your enterprise computing strategies.

ACTUATE
ReportingEngines™

www.reportingengines.com
sales@reportingengines.com
888-884-8665 + 1-913-851-2200

FREE TRIALS, DEMOS AND SAMPLE CODE

```
public class SampleProgramTest extends TestCase {

 public static void main(String[] args) {
  junit.textui.TestRunner.run(SampleProgramTest.class);
 }

 protected void setUp() throws Exception {
  super.setUp();
 }

 protected void tearDown() throws Exception {
  super.tearDown();
 }

 /*
  * Test method for 'servlet.SampleProgram.main(String[])'
  */
 public void testMain() {
  // TODO Auto-generated method stub
  SampleProgram.main(new String[]{"1", "abc"});
  assertEquals("1", "2");

 }

 /*
  * Test method for 'servlet.SampleProgram.performLogic(PrintWrit
er, int, String)'
  */
 public void testPerformLogic() {
  // TODO Auto-generated method stub
  PrintWriter out = new PrintWriter(System.out, true);
  SampleProgram.performLogic(out, 1, "abc");
 }
}
```

Listing 10: The servlet's deployment descriptor Web.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/dtd/Web-app_2_3.dtd">
<Web-app id="WebApp">
 <display-name>MyApp</display-name>

 <servlet>
  <servlet-name>SampleServlet</servlet-name>
  <display-name>SampleServlet</display-name>
  <description>Sample Servlet</description>
  <servlet-class>servlet.SampleServlet</servlet-class>
 </servlet>

 <servlet-mapping>
  <servlet-name>SampleServlet</servlet-name>
  <url-pattern>/SampleServlet</url-pattern>
 </servlet-mapping>

 <welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
 </welcome-file-list>
</Web-app>
```

Listing 11: ANT deployment script build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="MyJavaProject" default="create-war" basedir=".">

  <!-- Configure the directory where Tomcat 5.0 is -->
  <property name="local-tomcat" value="C:/Program Files/Apache
Software Foundation/Tomcat 5.0"/>
   <!-- Configure the directory into which the Web application
is built -->
  <property name="build"    value="${basedir}/build"/>

  <!-- Configure the context path for this application -->
  <property name="path"     value="/myapp"/>

  <!-- Configure properties to access the Manager application -
->
  <property name="url"      value="http://localhost:8080/man-
ager/html"/>
  <property name="username" value="admin"/>
  <property name="password" value="admin"/>

  <!-- Configure the custom Ant tasks for the Manager applica-
tion -->
  <taskdef name="deploy"    classname="org.apache.catalina.ant.
DeployTask"/>
  <taskdef name="list"      classname="org.apache.catalina.ant.
ListTask"/>
  <taskdef name="reload"    classname="org.apache.catalina.ant.
ReloadTask"/>
  <taskdef name="resources" classname="org.apache.catalina.ant.
ResourcesTask"/>
  <taskdef name="roles"     classname="org.apache.catalina.ant.
RolesTask"/>
  <taskdef name="start"     classname="org.apache.catalina.ant.
StartTask"/>
  <taskdef name="stop"      classname="org.apache.catalina.ant.
StopTask"/>
  <taskdef name="undeploy"  classname="org.apache.catalina.ant.
UndeployTask"/>

  <!-- Executable Targets -->
 <target name= "create-war">
    <property name= "war-base" value="${local-tomcat}/Webapps/
myapp/"/>

  <war destfile="${build}/myapp.war" Webxml="${basedir}/WEB-INF/
Web.xml">
       <fileset dir="${basedir}/html"/>
    <classes dir= "${basedir}/WEB-INF/classes">
      <exclude name="index.html"/>
      <exclude name="Web.xml"/>
      <exclude name="myapp.war"/>
     </classes>
   </war>

  </target>

  <target name="deploy" description="Install Web application"
          depends="create-war">
    <deploy url="${url}" username="${username}"
password="${password}"
             path="${path}" war="${build}${path}.war"/>
  </target>

  <target name="reload" description="Reload Web application"
          depends="create-war">
    <reload  url="${url}" username="${username}"
password="${password}"
             path="${path}"/>
  </target>

  <target name="undeploy" description="Remove Web application">
    <undeploy url="${url}" username="${username}"
password="${password}"
             path="${path}"/>
  </target>

</project>
```

Innovations by InterSystems

Rapid development with robust objects

Lightning speed with a multidimensional engine

Easy database administration

Massive scalability on minimal hardware

Database For Web-Based Applications.

Caché, the multidimensional database from InterSystems, can automatically project data and logic in a number of Web-centric forms, such as XML, Web Services, Java, and EJB. These unique capabilities make Caché ideal for rapidly developing Web-based applications.

Caché is the first database to seamlessly combine robust objects and robust SQL, thus eliminating object-relational mapping. Its post-relational technology delivers lightning-fast transaction processing, real-time analytics, and massive scalability on minimal hardware. It requires little administration, and incorporates a rapid application development environment.

These innovations mean faster time-to-market, lower cost of operations, and higher application performance. We back these claims with this money-back guarantee: *Buy Caché for new application development, and for up to one year you can return the license for a full refund if you are unhappy for any reason.** Caché is available for Unix, Linux, Windows, Mac OS X, and OpenVMS – and it's deployed on more than 100,000 systems ranging from two to over 50,000 users. We are InterSystems, a global software company with a track record of innovation for more than 25 years.

InterSystems
CACHÉ™

Try an innovative database for free: Download a fully functional, non-expiring copy of Caché, or request it on CD, at **www.InterSystems.com/Cache11P**

* Read about our money-back guarantee at the web page shown above.

Feature

Extreme Makeover: Architecture Edition

Large-scale refactoring with Spring and Hibernate

by Franz Garsombke

In the past few years there has been a proliferation of frameworks that allow for lighter, faster, and loosely coupled Java projects. These frameworks not only let you decouple your Java project from the application server for unit testing, they also allow for more agile refactoring, testing, and design techniques. This article will focus on telling the story of a large-scale refactoring effort implementing Spring and Hibernate as the underlying infrastructure tools. For those living under an abacus Spring is a J2EE framework built to handle many of the plumbing issues on a typical J2EE application. Hibernate is a popular Open Source Java object/relational persistence framework.

Bad EJB, No Donut

The last few years has produced a legion of J2EE applications tightly coupled to everyone's two favorite saviors, the Enterprise Java Bean (EJB) and the application container. Developers have tied their proverbial hands by coupling their code directly to these two Java mainstays. Many J2EE applications have been over-engineered and deemed overly complex by relying on these mainstays with few architectural best-practice guidelines to follow. Articles have been written and battle lines drawn around why and how this unfortunate series of events came to fruition.

Fortunately, there are now tools that provide ways to enable loose coupling and promote "simpler" architectures. Loose coupling encourages things like unit testing, test-driven development (TDD), and dependency injection. TDD is an approach to development that combines test-first development and constant refactoring. Dependency injection lets you inject dependent objects into your wired Java classes at runtime.

The First Step Is Admitting You Have A Problem

Everyone has probably experienced one or more applications built using the paradigm "'the more J2EE components the better." One of these applications was delivered to my doorstep for an extreme architecture makeover. The application owner's loose requirements were as follows:

- Scalable and reliable
- Flexible while resilient enough to support product changes
- Easy to integrate and test
- Loosely coupled so that disparate groups can work on separate components while simultaneously working on the application as a whole
- Enabling high developer productivity

Now, I understand that these requirements are ubiquitous requests for most applications and are almost assumed out of the gate. They can almost be seen as "infrastructure" requirements. Still, they're requirements and the right tools need to be used to satisfy them. Putting on my architect hat we came up with five problem areas that were symptoms of why these requirements aren't being met. From these symptoms we derived their root causes and solutions. Table 1 is a matrix showing our findings.

As you can see all of these issues had nothing to do with how the application dealt with business logic. They were solely infrastructure problems that had to be resolved for the application to reach a higher level of software maturity. Follow along as we see what an extreme makeover entails.

Patient Diagnosis – Condition Serious

The application needing the makeover wasn't small and continued to grow every day. A high-calorie diet of EJBs, deployment descriptors and other various J2EE infrastructure files was the norm. It had over 1,600 Java files, 50 entity beans, and over 20 stateless beans (that served mainly as entry points into the application for transactional boundaries). The application's main functions were order provisioning and workflow. It played an integral part in a service-oriented architecture (SOA) that encompassed over 25 Web Services. The software build and deployment cycle to the application server took over 15 minutes. I guess you could call it obese.

There were zero out-of-container unit tests. The only existing tests were written using Cactus, which is a unit-testing framework for executing server-side code. Loosely translated that means that by choosing tools like Cactus your code has to be deployed to the application server to be tested. I personally feel that tools like Cactus enable or even promote bad habits like not having tests that can be run straight from your integrated development environment (IDE).

After reviewing all of the symptoms, a diet of Spring and Hibernate was prescribed. These technologies were primar-

Franz Garsombke has been developing and architecting enterprise software solutions in Colorado for the last ten years. Franz is a huge proponent of open source frameworks and is passionate about developing and delivering quality applications. He has been published in many industry publications and spoken at Java user groups and conferences. He is proud to be the co-founder of a Java Bean mapping framework (http://dozer.sourceforge.net).

fgarsombke@yahoo.com

ily chosen for their ability to solve given symptoms. The following is a diary of the patient's miraculous transformation from emergency room to recovery room. Each symptom was typically remedied in one software iteration. The refactoring project took about five iterations over a period of five months, with between 2–4 infrastructure developers working on it at any given time. As the refactoring took place, the business logic developers went full steam ahead sprinkling functionality goodness across the application. The refactoring team's core focus was solely on the infrastructure. A good analogy is that the engines were changed out on the airplane while it was still in flight.

Remember the Food Pyramid for Healthy Living

Symptom:	Developers were interfering with each other
Root Cause:	Poor code structure and no separation of concerns
Solution:	Modularize the code base and adopt a layered architecture

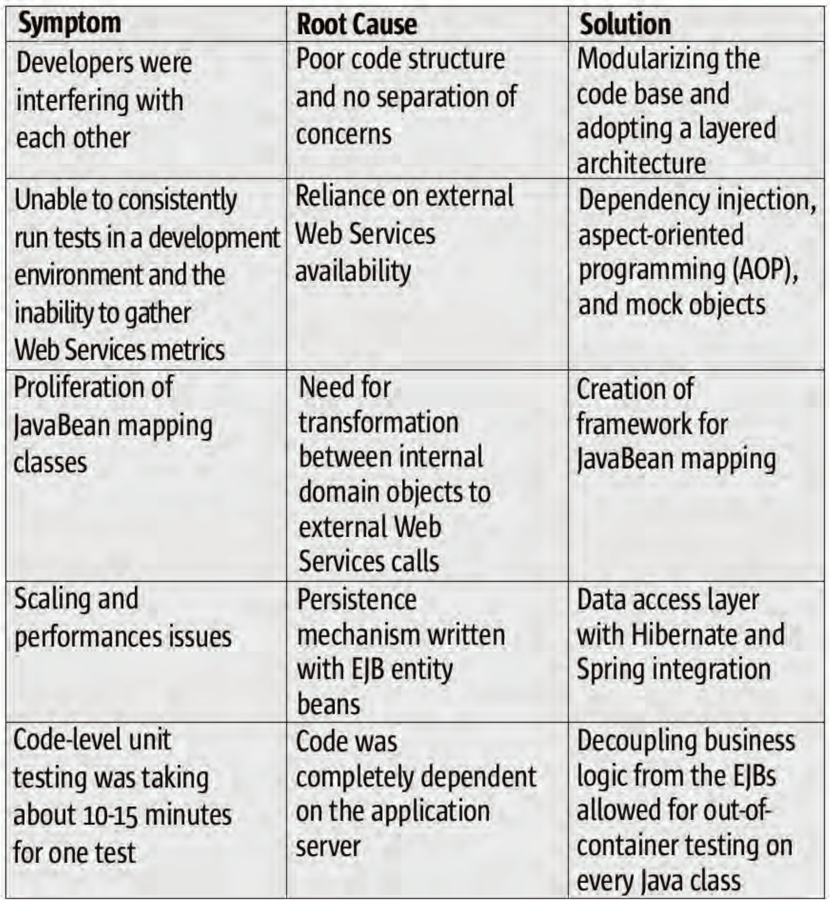
Having a tightly coupled code base can wreak havoc on developer productivity. Interaction complexity is extremely high and one code change can potentially affect multiple areas of the code. The application can be classified as brittle and development typically takes much longer than in a layered architecture. Building layers in your architecture will lead to a well-defined separation of concerns allowing each layer to be understood as a coherent whole. Interfaces are typically used as the contract between each layer. By implementing the Interface pattern your application can now substitute each layer with an alternative implementation. The benefits of this paradigm will be discussed in greater detail in the next symptom. The first step taken to modularizing the code base was to break the project into five distinct sub-projects. These sub-projects were compiled according to their dependencies. This enforced that each layer only knew about the ones below it. Figure 1 offers a before and after picture of the logical code base.

Notice that each distinct functional area dictates where the layer boundaries are. These are not revolutionary concepts, just good pragmatic architectural decisions. A loosely coupled architecture makes adopting an Inversion of Control (IOC) container like Spring much easier than not. The next symptom's solution would be hard if not impossible to implement without a layered architecture.

Take Your Injections

Symptom:	Unable to consistently run tests in a development environment or gather Web Services metrics.
Root Cause:	Reliance on external Web Services availability
Solution:	Dependency injection, aspect-oriented programming (AOP), and mock objects

With the advent of SOA came the troublesome deal of unit testing. As if testing wasn't hard enough, you're now reliant on someone else's application to execute unit tests. Another problem with a SOA is the inability to audit your service calls without intrusive metrics-gathering code throughout your application. The first order of business was to create an external provider layer with well-defined interfaces. Figure 2 shows the use of the façade pattern for all of the external Web Service calls done by the application. Classes in the external provider layer implement an interface and are injected into plain old Java object (POJO) business services.



Symptom	Root Cause	Solution
Developers were interfering with each other	Poor code structure and no separation of concerns	Modularizing the code base and adopting a layered architecture
Unable to consistently run tests in a development environment and the inability to gather Web Services metrics	Reliance on external Web Services availability	Dependency injection, aspect-oriented programming (AOP), and mock objects
Proliferation of JavaBean mapping classes	Need for transformation between internal domain objects to external Web Services calls	Creation of framework for JavaBean mapping
Scaling and performances issues	Persistence mechanism written with EJB entity beans	Data access layer with Hibernate and Spring integration
Code-level unit testing was taking about 10-15 minutes for one test	Code was completely dependent on the application server	Decoupling business logic from the EJBs allowed for out-of-container testing on every Java class

Table 1 Solution Matrix

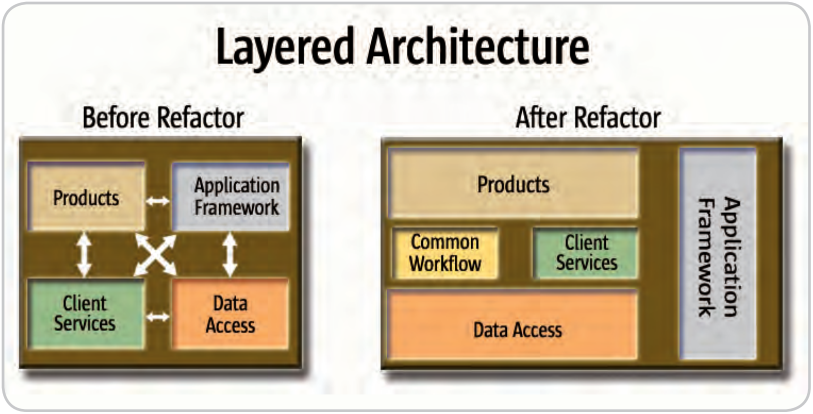


Figure 1 Building layers

One of the things that Spring excels at is being an IOC container. Spring uses a declarative approach to move an object's dependencies into an XML configuration file.

```
<!-- Spring wired Business Service bean -->
<bean id="businessServiceTarget" class="example.
BusinessServiceImpl">
  <property name="provisionWebService"><ref bean="provisionWebServ
ice"/></property>
  <property name="orderDAO"><ref bean="orderDAO"/></property>
</bean>
```

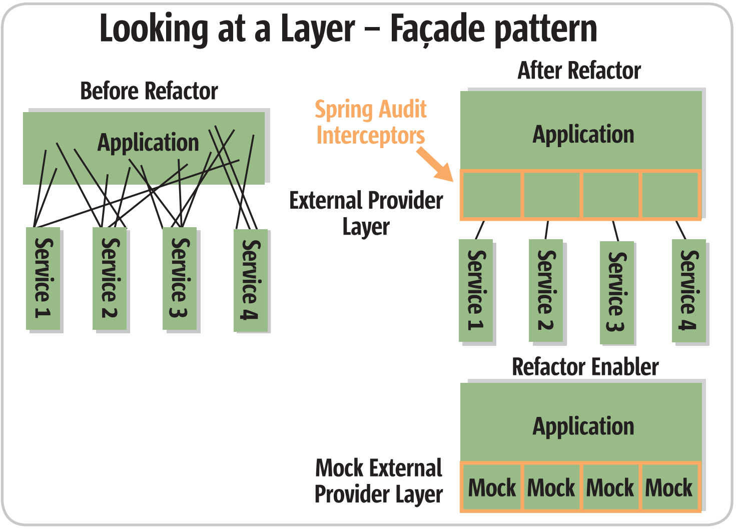


Figure 2 Façade pattern

This example shows how a POJO business service class would be defined in a Spring configuration file. This bean is injected with two other beans called *provisionWebService* and *orderDAO*. The injected *provisionWebService* bean implements the interface *ProvisionIF*. By having public *getter()* and *setter()* methods on our business service we can literally change the implementation of the *provisionWebService* attribute at runtime. As long as we inject another class that implements the *ProvisionIF* interface our business service bean will be content. This same pattern would apply to the order data access object (DAO) injected bean.

```
public class BusinessServiceImpl implements BusinessServiceIF {
  private ProvisionIF provisionWebService;
  public ProvisionIF getProvisionWebService();
  public void setProvisionWebService(ProvisionIF provisionWebSer-
vice);

  private OrderDAOIF orderDAO;
  public OrderDAOIF getOrderDAO();
  public void setOrderDAO(OrderDAOIF orderDAO);

}
```

Unit testing our business service class just got a lot easier. Now we don't have to rely on the provisioning Web Service to be available for testing. We can use Spring to inject a stub class that implements the *ProvisionIF* interface and returns canned data. This class has to be a concrete implementation and physically present as a Java class. If we don't want a proliferation of stub classes we can always choose a mocking framework like EasyMock. EasyMock provides mock objects for interfaces and generates them on-the-fly using Java's dynamic proxy mechanism. Tools like dependency injection and mock objects are a perfect fit for TDD since application code should be written so that modules are testable in isolation. These two paradigms for testing are shown graphically in Figure 2.

We have unit-tested our code, but how do we capture live metrics data in a production environment without writing copious amounts of auditing logic? SOA has enabled distributed architectures, but with that comes the headache of uniformly capturing input/output data, storing error messages, measuring request/response times, and correlating message calls. AOP aids developers in handling cross-cutting concerns such as security, auditing, logging, and transaction management. An 'advice' in Spring is an action taken by the AOP framework at a particular point during the execution of a program. The most typical types of 'advice' used in Spring are "around," "before," "after," and "throws." Spring has modeled these types as interceptors that are done using a chaining mechanism. Spring lets you attach N number of interceptors to a wired bean. In the example in Listing 1 we've attached an "around" audit interceptor to our bean that performs actions before and after the method invocation. This interceptor is applied declaratively. Spring uses dynamic proxies for AOP proxies but can also use CGLIB proxies if it's necessary to proxy a class rather than an interface.

Our example shows our business service bean dynamically proxied with a custom audit interceptor and a Hibernate interceptor provided by Spring. The audit interceptor is just another Spring wired bean that implements a *MethodInterceptor* interface. The audit interceptor will insert audit information into an auditing database before and after our business service method is invoked. This auditing information is now an integral part of capturing application metrics and managers distribute reports on a daily basis. The Hibernate interceptor is a class provided by Spring that handles Hibernate session management.

It's Dozerin' Time

Symptom:	Proliferation of Java Bean mapping classes
Root Cause:	Need for transformation between internal do main objects to external Web Service calls
Solution:	Creation of a framework for Java Bean mapping

A side effect of an SOA is the passing of domain objects between different systems. Typically, you won't want internal domain objects exposed externally and won't allow for external domain objects to bleed into your system. Creating a mapping framework is useful in a layered architecture where you're creating layers of abstraction by encapsulating changes to particular data objects versus propagating these objects to other layers (i.e., external service data objects, data transfer objects, or internal service data objects). Most programmers will develop some sort of mapping framework and spend countless hours and thousands of lines of code mapping to and from their many transfer objects.

One of the many positive side effects of the refactoring project was the birth of an Open Source project called *dozer* (http://dozer.sourceforge.net). Dozer was extracted from the refactored code base and made generic enough to handle most real-world mapping scenarios. Dozer supports Java Bean transformation, conversion, simple property mapping, complex type mapping, bidirectional mapping, and implicit/explicit mapping as well as recursive mapping.

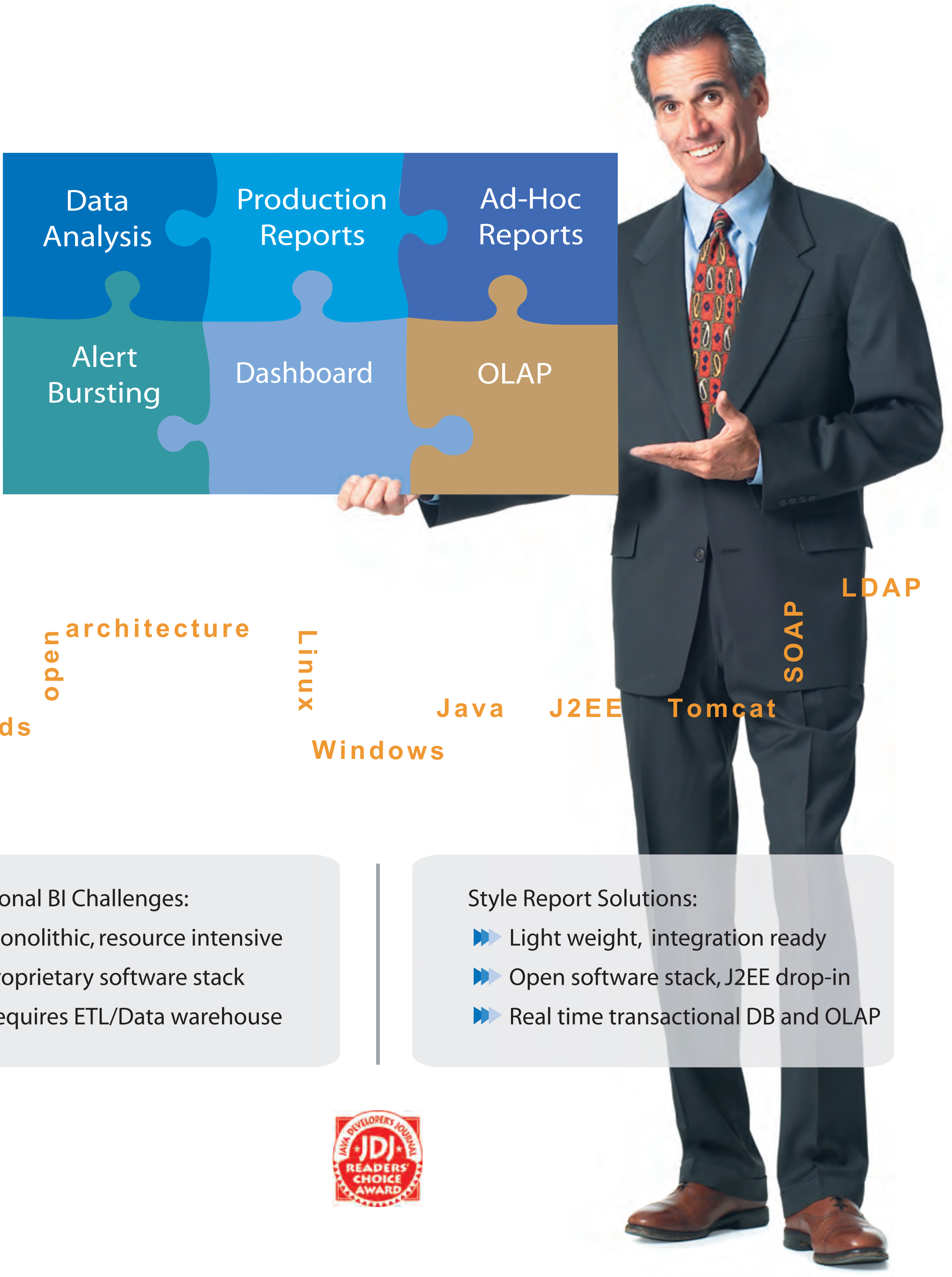
Hibernate Through the Cold Winter Nights

Symptom:	Scaling and performance issues
Root Cause:	Persistence mechanism written with EJB entity beans
Solution:	Data access layer with Hibernate and Spring integration

InetSoft
open standards innovation

Style Report 7

Light Weight, Integration Ready Enterprise Reporting & Analysis



Traditional BI Challenges:

- Monolithic, resource intensive
- Proprietary software stack
- Requires ETL/Data warehouse

Style Report Solutions:

- Light weight, integration ready
- Open software stack, J2EE drop-in
- Real time transactional DB and OLAP

JAVA DEVELOPER'S JOURNAL JDJ READERS' CHOICE AWARD

For more information and to download a free evaluation copy .... www.inetsoft.com/jdj

There have been many articles written about the reasons why we shouldn't use entity beans. Some of these reasons are performance issues, an immature query language, and the inability to unit-test and use entity beans as POJOs. Hibernate solves all of these issues and more. Before implementing Hibernate we created a technology-agnostic data access object (DAO) layer. This simply means that if there was an even newer, shinier persistence technology created tomorrow we could replace Hibernate with a minimal amount of refactoring. Some of the immediate benefits of using Hibernate are:

- The entire DAO layer can be tested without an application server
- First- and second-tier caching
- Support for association, inheritance, polymorphism, composition, and the Java collections framework

Spring provides integration with Hibernate for DAO implementation support, resource management, and transaction management. Spring provides a Hibernate interceptor class that manages the Hibernate session throughout a transaction by using Spring's AOP proxying. As a developer you just need to apply that interceptor to any of your Spring wired beans. Spring also lets you define a Hibernate session factory object. An example is presented in Listing 2.

Notice that the session factory is injected with a bean called *dataSource*. The *dataSource* bean can be injected with any class that implements the javax.sql.DataSource interface. Spring has the capability of loading different context files based on different testing scenarios. If you're running tests in an application container the *dataSource* bean can be defined as a container-based JNDI object.

```
<!-- In-Container implementation of a DataSource -->
<bean id="dataSource" class="org.springframework.jndi.
JndiObjectFactoryBean">
  <property name="jndiName"><value>jndi_datasource</value></prop-
erty>
</bean>
```

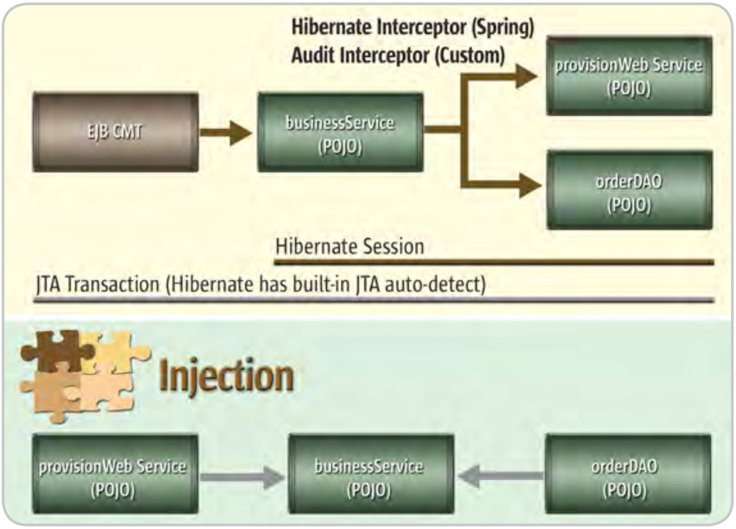

Hibernate Interceptor (Spring)
Audit Interceptor (Custom)
EJB CMT
businessService (POJO)
provisionWeb Service (POJO)
orderDAO (POJO)
Hibernate Session
JTA Transaction (Hibernate has built-in JTA auto-detect)
Injection
provisionWeb Service (POJO)
businessService (POJO)
orderDAO (POJO)

Figure 3 Business services with EJBs

If you're running tests out of the container the *dataSource* bean can be a simple JDBC connection.

```
<!-- Out of container implementation of a DataSource -->
<bean id="dataSource" class="org.springframework.jdbc.datasource.
SingleConnectionDataSource">
  <property name="driverClassName"><value>someDriverName</value></
property>
  <property name="url"><value>myUrl</value></property>
  <property name="username"><value>user</value></property>
  <property name="password"><value>password</value></property>
</bean>
```

By looking at this one example the power of dependency injection becomes truly apparent. One slight configuration change and we're one step closer to running our entire suite of unit tests without an application server.

It's Springtime, Get 'Outside' and Enjoy the Weather

Symptom:	Code-level unit testing was taking about 10-15 minutes for one test
Root Cause:	Code was completely dependent on the application server
Solution:	Decoupling business logic from the EJBs allowed for out-of-container testing on every Java class

At this stage of the game the prognosis is getting better for our refactored application. We've successfully layered our architecture, applied dependency injection to move our object's dependencies into a configuration file, used AOP to audit our Web Service calls, used a Java Bean mapping framework, and replaced all the entity beans with Hibernate. One last step has to be implemented before our entire suite of unit tests can be tested out- of-container. Before Spring, EJBs were seen as the panacea for infrastructure details like security and transaction management. Most developers chose stateless session beans to handle all of their container-managed transactions. They also probably chose to store their business logic in these same beans, thus coupling them to the container with no chance of writing simple unit tests. Figure 3 offers a solution to this problem.

All of the EJB business logic has been moved into a Spring wired bean called *BusinessService,* which is a POJO. The EJB simply retrieves the bean out of the Spring context and calls business methods on it. In this particular example the *BusinessService* bean is injected with two other Spring wired beans. It's also wrapped with a custom audit interceptor as well as a Spring Hibernate interceptor. The EJB still manages the JTA transaction while the Hibernate interceptor does the Hibernate session management. The only reason that stateless EJBs weren't discarded altogether was for their RMI remoting capabilities. In future iterations we plan to remove the stateless EJBs altogether. Spring has transaction interceptors that can simulate the transactional work of a container-managed EJB.

Unit testing was made painless by using Spring's *HibernateTransactionManager* class. Using that class we were able to have our JUnit test cases simulate the transac-

keep your server problems...at bay. Get freedom for more in life

No more endless hours of sorting out server problems. AutoPilot ™ from Arcturus Technologies, automatically does that for you.

It's an affordable new concept that eliminates your worries about WebLogic upkeep and maintenance. AutoPilot ™ is so powerful and efficient that not only does it monitor, analyze and optimize your WebLogic for maximum performance, it also advises you about how to cure problems and avoid costly failures. It does this all automatically, giving you the freedom to put back what you have been missing in life.

- Automatically optimizes WebLogic server, saving you time and effort.
- Detects problems automatically.
- Gives advice from a dynamic knowledge base.
- Provides WebLogic system state for analysis in the event of a catastrophic failure.
- Optimizes existing J2EE resources.
- Increases the life, effectiveness, and quality of existing resources by reducing or eliminating problem areas.
- Improves resource communication and response time.
- Enhances server capacity to handle more loads and transactions.
- Provides advanced system monitoring.
- Furnishes customizable reports.
- Generates email alerts.
- Cost Effective at only $999 per IP address, regardless the number of CPUs

Want to know more & place an order ?
Log on to **www.arcturustech.com**

AUTOPILOT™
AUTO OPTIMIZER

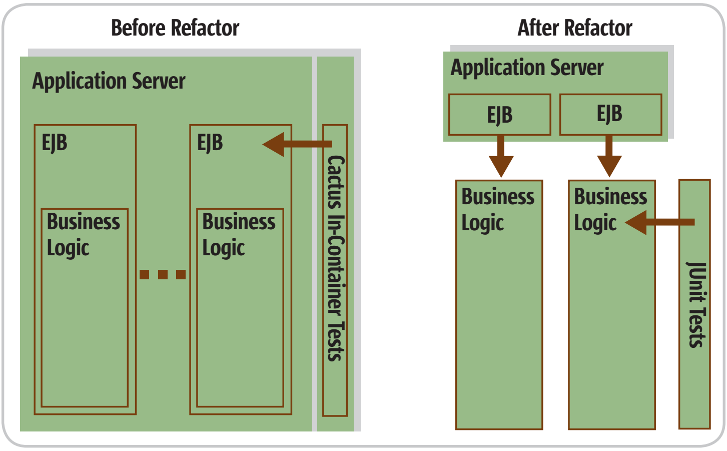


Figure 4 Decoupling code from the container

tion management and Hibernate session management capabilities performed in the container. Getting full transaction and Hibernate session support outside of the container lets your IDE mimic testing inside the container. Figure 4 graphically represents the unit-testing transformation.

You Reap What You Sow

At the end of the refactoring project many tangible results had been achieved:

- Greater visibility and control in an SOA by using Spring's simple AOP mechanism for capturing auditing metrics
- The entire code base can be tested outside of the container using Spring's dependency injection model and Hibernate for the persistence layer
- Code is much more maintainable with a loosely coupled layered architecture
- Refactoring is much easier with a baseline of out-of-container unit tests in place
- Dependency injection helps immensely in mocking out external service layer calls
- Testing becomes much easier, meaning it's more likely to happen

On any refactoring project you really do reap what you sow. Refactoring has an incredible return on investment in the short- and long-term future of your application. A word of caution before using any 'new' technologies: create a prototype application of what you want your 'end-state' architecture to look like. This prototype should be the proving grounds for all of the gory technical details and answer many hypothetical questions at two in the morning. Without our prototype, this refactoring project would have failed.

Summary – Prognosis Good

Applications that choose frameworks that allow for loose coupling can start to shed some of their J2EE baggage and leave a lot of the plumbing to tools like Spring. The days of being tightly coupled to your application server for unit testing are quickly slipping away. By using a pragmatic approach to Open Source tool selection many rewards will be instantly achieved. Applications will be more flexible and developer productivity and efficiency should increase dramatically. Not only do these tools help in decoupling, but they also tend to make applications a lot simpler by removing dependencies on J2EE containers. After having the pleasure of working on a loosely coupled flexible architecture you will wonder how any work got done before.

References

- *AOP:* http://www.springframework.org/docs/reference/aop.html
- *EasyMock:* http://www.easymock.org/
- *Dependency Injection:* http://www.martinfowler.com/articles/injection.html
- *Dozer:* http://dozer.sourceforge.net
- *Hibernate:* http://www.hibernate.org/
- *Spring:* http://www.springframework.org/
- *Test Driven Development:* http://www.agiledata.org/essays/tdd.html

Listing 1

```
<bean id="businessService"
class="org.springframework.aop.framework.
ProxyFactoryBean">
  <property name="proxyInterfaces"><value>example.
BusinessServiceIF</value></property>
  <!-- this property determines which bean to dynami-
cally proxy -->
  <property name="target"><ref local="businiessServiceTa
rget"/></property>
  <property name="interceptorNames">
    <list>
      <value>auditInterceptor</value> <!-- Custom
Interceptor -->
      <value>hibernateInterceptor</value> <!--
Interceptor provided by Spring -->
    </list>
  </property>
</bean>

<bean id="auditInterceptor" class="example.
AuditInterceptor">
 <property name="auditDAO">
   <ref bean="auditDAO"/>
 </property>
</bean>

<bean id="hibernateInterceptor" class=" org.springframe-
work.orm.hibernate.HibernateInterceptor
"/>
```

Listing 2

```
<!-- Hibernate Session Factory -->
<bean id="sessionFactory" class="org.springframework.orm.
hibernate.LocalSessionFactoryBean">
  <property name="dataSource"><ref bean="dataSource"/></
property>
  <!-- Hibernate mapping files -->
  <property name="mappingResources">
    <list>
      <value>example/Audit.hbm.xml</value>
     </list>
  </property>
</bean>
```

Powering Eclipse to the next level...

NitroX™

PROFESSIONAL TOOLS FOR ECLIPSE™

JSP | Struts | JSF

Professional development for JSP, Struts & JSF
Simultaneous visual/source Struts & JSF configuration
Code completion for all levels
Automatic validation & error checking for all levels
Debugging support for all levels
AppXRay, AppXaminer, AppXnavigator

FREE JSP EDITOR!

download at: www.m7.com/power

M7®

Top Reviews Winner: InfoWorld (8.3), CRN ★★★★★ , SDMagazine ★★★★

Deriving the Visitor Pattern

A review and discussion

by Nishanth Sastry

Like most other self-respecting developers I had also read the GoF book, including the section on the visitor pattern. However, when a colleague came over to me with a question, I could not initially justify the complexity of the example code I saw in the book. What follows is a discussion of why the visitor pattern is the way it is.

Brief Review of the Pattern

The definitive description of the pattern is in the GoF book *Design Patterns*, Chapter 5 (pp 331-344)(see References section). The Wikipedia has a concise and good description, which formed the basis for my brief review here. The visitor pattern is classified as a Behavioral pattern, so the thing to notice is the way in which the classes and objects interact and distribute responsibility. A typical application of this pattern occurs in the following scenario: we have a number of elements in an object structure (common structures include trees & lists) and we want to perform a bunch of disparate operations (e.g. printing or cloning each element) on the elements of the structure.

The visitor pattern is a way of separating the operation from the object structure and a way of collecting together the different implementations of an operation for different kinds of elements in the object structure. A Visitor class is created which knows how to perform a particular operation on the different kinds of elements in the object structure. Each type of element in the structure defines an accept() method that can accept any kind of Visitor. The visitor is passed to each element in the structure in turn, by calling its accept() method and the Visitor then performs the operation on the visited element. One important consequence of this separation of object structure and operation is that we can later add a new operation (a new kind of Visitor) without having to modify the element classes of the object structure.

Each type of Visitor defines several visit()methods, one for each kind of element. The basic insight is that the precise set of instructions to execute (i.e. the method or function to call) depends on the *run-time* types of both the Visitor & the visited element. Java only lets us call different methods based on the run-time type of one object (via virtual functions), so the pattern advocates a clever solution: The second dependency on the type of element visited is first resolved by polymorphically calling the accept() method of the visited element. accept() then resolves the first dependency by turning around and polymorphically calling the visit()method for its class.

An Example

Before this description gets too confusing, let us study the pattern in the context of a concrete problem: Let us say we need to traverse a list *collecting* node-specific information. The list has two kinds of nodes, say, Red and Black, which needed to be processed differently. It seems like an ideal application for the visitor pattern. Listing 1 shows the code. (All code samples in this article use a J2SE 5.0 compatible compiler.)

To me and my colleague, this initially seemed like an overly complex solution for a simple problem. NodeVisitor.doVisit() calls into the Node's accept methods, which simply delegates back into NodeVisitor. Furthermore, the accept() methods of RedNode and BlackNode are almost identical. Finally, notice that if we now add a GreenNode class, we need to add a new visitGreen() method to the NodeVisitor class and re-compile it (not to speak of the almost redundant implementation of accept() in the GreenNode class). Ugh! This does not seem kosher by any OO standard.

The Need for the accept() Methods

Novice armchair Java developers might ask why we can't do something simpler, like Listing 2, for example, without touching the Node interface, or the classes RedNode and BlackNode which implement it.

Listing 2 has two significant differences from the previous. First, there is no redundant method (namely accept()) for each node type to implement. Second, we use function name overloading for the visit() implementations, thus enabling the "clever" foreach loop, which iterates over each node and calls the appropriate overloaded version of visit() depending on the type of the current element. With this, we hope to contain all the visiting logic within NodeVisitor.

Alas, real developers have a more difficult job than armchair developers! If you are using a language like Java or C++, an overloaded function name like visit() has to get resolved at compile time. Thus line 6.iii will not compile because none of the visit() methods provided in NodeVisitor know how to accept a generic "Node" as argument.

For line 6.iii to work the way we want it to, the decision on what operation needs to be performed has to be delayed until we can determine at runtime the type of the node *n* being examined in the current iteration of the for-each loop.

Nishanth Sastry is a software developer at IBM working on the WebSphere Portal & Workplace family of products. He has a Masters degree in Computer Science from the University of Texas at Austin. He enjoys well-written code & fall in New England, among other things. He lives in Concord, Mass.

nishanth_sastry@us.ibm.com

Traditional OO languages (Java, C++ etc) provide us with one standard tool for delaying function resolution until run-time: virtual functions. Thus, in Listing 1, 6.iii is modified to a virtual function call n.accept(nv). So the actual function that gets called is decided at run-time. The version called then delegates work by invoking the right version of NodeVisitor.visit().

So Why Not Just Use Plain Vanilla Inheritance?

The explanation I just gave is good, but not good enough. I can almost hear you ask: why doesn't accept() do the work itself? Why does it have to delegate back to NodeVisitor? There are three reasons:

1. ***Accumulating state:*** If you read the problem I presented closely, you will notice that I specified a need to *collect* node-specific information. Since the doVisit passes the same NodeVisitor instance to each accept(), the visitor can be used to accumulate state across the different Node objects. For example, say you have an Employee HR application where the Red nodes represent employees, the Black nodes represent managers, visitRed() calculates the pay raises for programmers, and visitBlack the pay raises for managers. The NodeVisitor nv could print a report of the total increase in salary expense at the end of the for loop.
2. ***Supporting more than one visitor (the need for double dispatch):*** Say the next version of your Employee HR application needs to add a new HRPolicyVisitor that checks for compliance with some HR policy and the implementation is different for managers and programmers.

   To accommodate both the types of Visitors, we introduce an additional layer of indirection – an abstract EmployeeNodeVisitor interface with virtual visitXXX() functions for each type of element to visit, namely visitProgrammer() & visitManager(). The old PayRaiseVisitor and the new HRPolicyVisitor both implement EmployeeNodeVisitor. The decision on which version of visit() gets called now gets determined by a two-step process. The first step is as before. The node type of the visited element n in the foreach loop determines which version of the virtual function accept() gets called. In the second step, the type of the EmployeeVisitor passed in to accept() determines the (virtual function) version of visitXXX() called. The source files that come with this article (These can be downloaded at **???**) show the skeleton of this implementation. Figure 1 illustrates the sequence of calls from both doPayHike(), which uses a PayRaiseVisitor to raise the pay of each employee, and doEnforcePolicy() which uses a HRPolicyVisitor to check HR policy compliance.

   This technique, where the types of two objects are used to select the operation invoked is known as *double dispatch*. By contrast, *single dispatch* uses the type of one object to select the operation invoked. One known implementation of single dispatch is virtual functions. Since Java and C++ support only this form of single dispatch, the pattern simulates double dispatch by using single dispatch twice!
3. ***Separation of concerns:*** A concern is any focus of interest in a program. A classic tenet of good software design is that the different concerns of a program must be broken down into separate modules that have little or no overlap. In the Employee HR program , visitProgrammer and visitManager of a particular visitor have more commonality than the two visitProgrammers of the different visitors or the two visitManagers of the different visitors. In fact, the methods in a given visitor may even share state information as described in 1 above. This makes the Visitor pattern a good way to organize code by separation of concerns.

Notice also that as a consequence of this way of organizing code, it is extremely easy to add a new visitor operation, but adding a new kind of node requires adding a new visitXXX method to *all* the Visitors.

If none of the above three reasons apply, you would be better off not delegating the work of accept() back to a separate visitXXX() method.i.e. plain vanilla inheritance would be more

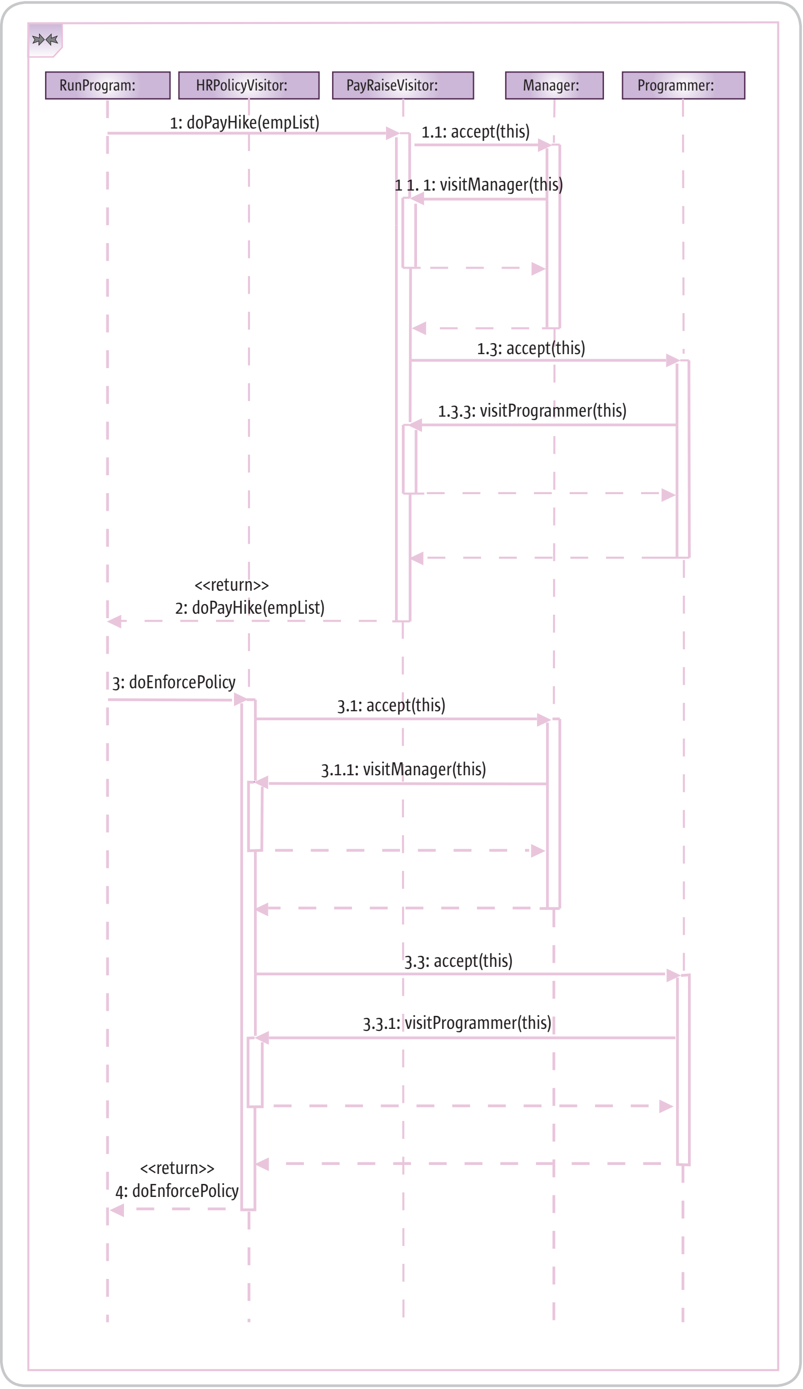


Figure 1 Sequence diagram of an application of the Visitor pattern with two visitors & two kinds of nodes to visit

appropriate than an application of the Visitor pattern. On the other hand, if *any* of the above reasons apply, the Visitor pattern would be a good solution for you.

But This Still Does Not Preclude Overloading the visit() Methods...

You might still have one lingering question about Listing 1: Why can't we use function name overloading instead of the different visit<<NodeType>>() methods (as in Listing 3)?

The short answer is that nothing prevents you from doing this; Listing 3 is just as correct as Listing 1 For the last word, however, I will have to defer to the GoF, who write the following in a footnote:

We could use function overloading to give these operations the simple name, like Visit, since the operations are already differentiated by the parameter they're passed. There are pros and cons to such overloading. On the one hand, it reinforces the fact that each operation involves the same analysis, albeit on a different argument. On the other hand, that might make what's going on at the call site less obvious to someone reading the code. It really boils down to whether you believe function overloading is good or not [in this situation].

Conclusion

In this article we reviewed the Visitor pattern and "derived" it from an armchair sketch of the functionality we wanted: the ability to accumulate state over elements of an object structure, the separation of the operations from the object structure, and the ability to add new operations without recompiling the element types. These requirements called for a "double dispatch"; i.e. the precise method to call for "visiting" each element in the structure depended on two runtime types: the type of Visitor and the type of the visited element. The Visitor pattern was shown to be a way to simulate double dispatch using virtual functions, a form of single dispatch.

References

- Gamma, et al. *Design Patterns: Elements of Reusable Object-Oriented Software*, 1995, Addison-Wesley, Reading, MA.
- Wikipedia contributors, "Visitor pattern," *Wikipedia: The Free Encyclopedia*, http://en.wikipedia.org/wiki/Visitor_pattern (accessed Aug 19, 2005)

Listing 1: A visitor for Red & Black nodes in a list

```
class NodeVisitor {
    public void visitRed(RedNode n) {
 // do red node-specific things...
    }
    public void visitBlack(BlackNode n) {
 // do black node-specific things...
    }

    public static void doVisit(List<Node> list) {
 NodeVisitor nv = new NodeVisitor();
 for (Node n:list) {
     n.accept(nv);
 }
    }
}


interface Node {
 void accept(NodeVisitor visitor);
}

class RedNode implements Node {
    //... other methods
    public void accept(NodeVisitor visitor) {
 visitor.visitRed(this);
    }
}

class BlackNode implements Node {
    //... other methods
    public void accept(NodeVisitor visitor) {
 visitor.visitBlack(this);
    }
}
```

Listing 2: "Simpler" (erroneous) implementation of NodeVisitor

```
class NodeVisitor {
public void visit(RedNode n) {
           // do red node-specific things...
}
public void visit(BlackNode n) {
          // do black node-specific things...
}
public static void doVisit(List<Node> list) {
NodeVisitor nv = new NodeVisitor();
for (Node n:list) {
nv.visit(n);
}
}
}
```

Listing 3: Listing 1 with overloaded visit methods

```
class NodeVisitor {
    public void visit(RedNode n) {
 // do red node-specific things...
    }
    public void visit(BlackNode n) {
 // do black node-specific things...
    }

    public static void doVisit(List<Node> list) {
 NodeVisitor nv = new NodeVisitor();
 for (Node n:list) {
     n.accept(nv);
 }
    }
}


interface Node {
 void accept(NodeVisitor visitor);
}

class RedNode implements Node {
    //... other methods
    public void accept(NodeVisitor visitor) {
 visitor.visit(this);
    }
}

class BlackNode implements Node {
    //... other methods
    public void accept(NodeVisitor visitor) {
 visitor.visit(this);
    }
}
```

That's what building your own reporting solution from scratch feels like!
POW!!!

Relax! *Intelli*VIEW comes to your rescue.

Why knock yourself out trying to build your own reporting solution when you have IntelliVIEW - the hassle-free way to providing reporting within your applications. IntelliVIEW saves you the trouble of defocusing from your core competencies with these unique advantages

- Seamless integration with your applications
- Scalable & flexible licensing
- 24x7 support
- 'Stitched-to-fit' for your unique reporting needs
- Lightweight & robust reporting framework
- Hassle-free & interactive reporting

Call toll-free 1-866-99IVIEW or
visit http://www.intelliview.com/jdj

PRODUCT OF Synaptris

3031 Tisch Way, Suite 1002, San Jose, CA 95128. USA

Web Frameworks and IDE

Part Three

by Yakov Fain

In the first two articles of this series (see http://java.sys-con.com/read/108260.htm and http://java.sys-con.com/read/124664.htm), I started thinking aloud about automating my gas station using various Java-related technologies. This time, I'm trying to figure out what IDE and Web framework to use.

How Many Java Web Frameworks Does Mankind Need?

Being a consultant in my previous life, I worked on different projects for various clients. Each time I joined a project I had to learn a new Java technology that promised to make my life easier. Here it comes again! Now I need to select a Web framework. The good part is that literally all of them are free (is it the right word? I need to do some more reading on all these public licenses).

Since a gas station is the best place for networking, I started to ask drivers/programmers to recommend a good Java Web framework. By the way, while pumping gas, you can notice things you've never known before. For example, work visa holders from India usually drive Toyotas and Hondas. Eventually, they switch to something BMWish. Anyway, I was able to collect more than 50 (!) names of Java Web frameworks that are available today. Are we serious? I understand the benefits of the free market and competition, but isn't it a little too much? One Mercedes driver from China told me that their government restricts the number of kids per family. Should we penalize anyone who's even thinking of creating yet another Java Web framework?

Let me use my proven way of finding the right software. Some people just throw the dice, but I'd rather go to dice.com, which is a major IT job search portal in the U.S. Just start entering the names of the frameworks one by one, perform the search, and write down the counts of ads that look for people with such skills. The leaders are Struts: 1350, Spring: 377, and JSF: 230. For those who don't like this approach, I'll mention a couple of other frameworks that people like to blog about: Tapestry, Cocoon, and Wicket.

Car drivers also form two major groups: pragmatic people drive Lexus, Toyotas, and Hondas. But there are people who will never betray Mercedes, Jaguar, or Land Rover even though they break at least once a year.

IDE

People who work at my gas station came to America with no money. When they need to get their first TV set, they don't buy it, but pick it up off the streets. That's right, when people in our neighborhood need to get rid of an old TV, they just take it out on the street, and the garbage truck picks it up…unless my guys do it sooner. My Pakistani guy revealed a secret: if the TV set is broken, people who put it out cut the power cord off. This way you won't waste time bringing it home. Nice!

These days it works the same way with commercial software. When a company has a dead-end product, it gives it away to the open source community. The only difference is that people take their TVs out of the house quietly, while the software vendors make a loud noise about their donations. First you read a title like "BEA has contributed WebLogic Workshop IDE to the Beehive Apache project." How sweet! But a little later, "BEA decided to join the Eclipse platform and build another version of Workshop as a plugin with some functionality that exists in its current version." That's right, just dump unwanted software onto the open source community. They have plenty of handy people there, who might take it apart and use some nice ideas like the use of annotations in the pre-Java 5 era. I worked with WebLogic Workshop; it's slow, uses proprietary constructs, and, if you start a J2EE project with it, you're stuck: there is no way to port it to any other environment.

Technically, there are only two free Java IDEs to consider: NetBeans (Sun Microsystems, Inc.) and Eclipse (The Rest of the World, Inc.). Based on all the reviews and demos I've seen, they both are robust and provide similar functionality. I tried to read blog postings to pick one of these, but the bloggers usually fight over particular features, like "Refactoring in NetBeans is more powerful than in Eclipse." So? My gut feeling tells me that I should go with the Eclipse IDE, which has an open architecture and an abundance of online documentation and support. If I had several thousand dollars, I could have paid for a fat report that would analyze all available Java IDEs, but I'm sure on its last page it would have just one word: Eclipse. Even Borland gave up: their first Eclipse-based product called Peloton will be available in the first half of 2006.

Actually, to be fair, I need to admit that there are people who despise everyone who's using anything other than vi or Emacs, but let's not even go there. We need to play it safe.

Help

Thank you to all who responded to the first two articles and provided some good ideas. I'm planning to make my next column a discussion of your feedback. As always, I'm asking for your suggestions in automating of my small business. Just share your thoughts at the online version of this article at http://java.sys-con.com/read/136518.htm.

Yakov Fain is a J2EE architect and creator of seminars "Weekend with Experts" (www.weekendwithexperts.com). He is the author of the best-selling book *The Java Tutorial for the Real World* and an e-book *Java Programming for Kids, Parents and Grandparents*. Yakov also authored several chapters for *Java 2 Enterprise Edition 1.4 Bible*.

yakovfain@sys-con.com

Complex JAVA J2EE Hosting made easy.

WebAppCabaret sm

http://www.webappcabaret.com/jdj.jsp
1.866.256.7973

JAVA J2EE-Ready Managed Dedicated Hosting Plans:

Xeon I

SAMEDAY SETUP

Dual 2.8 GHz Xeons
2GB RAM
Dual 73GB SCSI
1U Server
Firewall
Linux
Monitoring
NGASI Manager

$279 *monthly*

Pentium 4 I

SAMEDAY SETUP

FREE SETUP

2.4 GHz P4
2GB RAM
Dual 80GB ATA
1U Server
Firewall
Linux
Monitoring
NGASI Manager

$199 *monthly*

2nd month FREE

4Balance I

1 Database Server and 2 Application Servers connected to 1 dedicated load balancing device. Dual Xeons. High-Availability.

$1724 *monthly*

At **WebAppCabaret** we specialize in **JAVA J2EE Hosting**, featuring **managed dedicated servers** preloaded with most open source JAVA technologies.

PRELOADED WITH:
JDK1.4 . JDK1.5 . Tomcat . JBoss . Struts . ANT . Spring . Hibernate
Apache . MySQL . PostgreSQL . Portals . CRM . CMS . Blogs . Frameworks
All easily manage via a web based control panel.

Details:
-All Servers installed with the latest Enterprise Linux
-Firewall Protection
-Up to 60 GB daily on site backup included at no extra charge per server.
-Database on site backup every 2 hours
-Daily off site database backup
-A spare server is always available in case one of the server goes down
-Intrusion detection.
-24x7 Server and application monitoring with automatic self healing
-The Latest Bug fixes and Security updates.
-Tier 1 Data Center. 100% Network Uptime Guarantee
-Guaranteed Reliability backed by industry-leading Service Level Agreements

Log on now at **http://www.webappcabaret.com/jdj.jsp** or call today at **1.866.256.7973**

NGASI POWERED

WebAppCabaret sm

JAVA J2EE Hosting

Benefiting from Open Source Development

The goal: cross-platform Java development

by Sumitra Chary, Christian Donner, Jim Lamoureaux, Ilia Papas, and Dita Vyslouzil

In a market that is defined by today's tight IT budgets, saving on software licenses can mean the difference between financial failure and success for a software development project. While our corporate clients use commercial-grade application servers, we sometimes find ourselves in a situation where there are no funds for developer licenses of these commercial application servers. Out of necessity, we developed and implemented a process that allows for development on top of an open source stack, while production delivery relies on a commercial application server.

Initial concerns that implementation differences and the different runtime environments would lead to issue-prone deployments turned out to be unjustified. While different application servers do indeed show incompatibilities, we found that we were able to avoid common pitfalls through preparation and disciplined coding. In this article, we will explain what it takes to develop complex Web applications with Eclipse and Tomcat and to deploy these applications to a WebSphere-based production environment.

Introduction

It all started when a client requested a solution for the WebSphere application server platform, but did not want to cover the cost of WebSphere Studio licenses for the development team. We looked for alternatives and found one in Eclipse and Tomcat.

The team initially feared that the different implementation of core functionalities provided by application server containers would create application portability issues. The main areas of concern included transaction management, security, and application deployment.

Because we used IBM's Tivoli Access Manager and WebSEAL Reverse Proxy in production, but relied on Tomcat's built-in authentication in development, there was concern that having only a subset of the target security infrastructure available in development would limit our ability to build a security service layer for Tivoli.

These risks had to be addressed and dealt with. At that time it seemed that the cost of doing so would outweigh the potential savings from software licenses. However strong this concern was, it was difficult to convey it to a client who was eager to start the project, and so we embarked on the open source endeavor.

Sumitra Chary is a senior software engineer at Molecular. Her career has spanned both academic and commercial worlds. These have included software systems for X-ray observatory missions, network management, marketing automation, and enterprise Web applications.

schary@moecular.com

Developing with Eclipse and Tomcat

Once properly configured, Eclipse can be a powerful hub for developing your application. It can automatically generate content and code such as class header comments, implementations of functions from interfaces, variable getters and setters, and more. These time-saving tools, along with the multitude of available plug-ins (e.g., for Tomcat, VSS, and Struts) allowed us to spend less time performing repetitive tasks and more time actually developing.

We created a project in Eclipse with its root reflecting the root of our Web application, which would later be packaged into a WAR (Web Application Archive), then an EAR (Enterprise Archive), along with the required application configuration files, for deployment to WebSphere. This root directory was located within the "webapps" directory of our Tomcat installation, which is the default directory that Tomcat allocates for Web applications.

IBM WebSphere	5.0.2	Websphere Application Server
MS SQL Server	2000	The client licensed Microsoft's database
IBM JDK	1.3.1	Required, the application does not work with Sun's JDK
Jakarta Struts	1.2.4	Open Source MVC Framework
Spring	1.1.1	Open Source Framework
JTDS	1.0	Open Source SQL Server JDBC driver
Tivoli Access Manager	5.1	Tivoli Access Manager for eBusiness (TAMeb).
IBM Tivoli Directory Server	5.2	LDAP Server
IIS	5.0	Microsoft's Internet Information Service

Table 1 Development Stack

Eclipse	3.0.1	Open Source IDE
Jakarta Tomcat	5.0.28	Open Source Application Server
Jakarta Struts	1.2.4	Open Source MVC Framework
Eclipse VSS Plugin	1.6.0	Source control plugin for Eclipse and MS Visual Source Safe
Eclipse Tomcat Plugin	3.0	Start, stop and recycle Tomcat from within the IDE
Jakarta Log4j	1.2.8	Open Source logging and tracing framework
MS SQL Server	2000	This is optional. Developers can use a shared database server instead of deploying the database locally.
IBM JDK	1.3.1	Required, the application does not work with Sun's JDK
Spring	1.1.1	Java Framework
JTDS	1.0	Open Source SQL Server JDBC driver

Table 2 Production Stack

Although the Tomcat plug-in for Eclipse does not add any new functionality to either product, it greatly eases the integration of the two and saves time by consolidating common tasks in one place and reducing the need for multitasking. Debugging in Eclipse is fairly robust, allowing the user to step through code and to evaluate expressions on the fly. The JDK we were using (IBM 1.3.1) does not support hot-replacing of classes, but new code is loaded on an application server restart, which does not take much time.

It should be mentioned that Tomcat does not support Enterprise beans. We decided against Enterprise beans because the Spring framework provides similar features without the platform dependencies.

The Microsoft Visual SourceSafe plug-in integrates well into the Eclipse interface, allowing for comments on both checkout and check-in. It also provides a report of all files checked out within the project, the owner, and what actions are being performed on them. The only gripe is that when checking-in files, it does not remember the checkout comment, so it must be reentered manually.

There are a few aspects to take into consideration when bridging the gap between the development and production environments. User authentication, handled by Tivoli Acess Manager in production, was handled by the *tomcat-users.xml* file located in the *config* directory. Roles, users, and passwords are recorded in this file. Through the use of configuration files and Ant, we were able to easily change server locations and credentials, as well as any other variables that may need to change when code is moved between environments. Tomcat tends to be much more forgiving when it comes to parsing configuration files such as the *web.xml* and tag library definitions, whereas WebSphere will either load the application in a crippled state or not at all. The *dtd*s must be adhered to in order to avoid this issue.

Production Environment

The production environment was a load-balanced configuration of two application servers and several other servers hosting the security environment (Tivoli Access Manager) and the database (see Tables 1 and 2 and Figure 1).

Multiple Environments

In most software development projects, to support the life cycle of the application, there are multiple environments into which the code must be deployed (see Figure 2).

When the application is deployed from one environment to another, various things need to change, such as database data source information and LDAP server information. We used Ant's property filtering capability to generate runtime resource files, such as properties files and Spring application context files, with the correct information appropriate to each environment.

We recommend the following steps to make this work:

1. Define a *deploy.host* property and assign a value according to the hostname of the target deployment environment

Christian Donner is a senior consultant and technical architect at Molecular. He is a Certified Sun Enterprise Architect for Java 2 and devotes much of his career to helping clients integrate complex Web applications with their grown corporate IT infrastructures. Christian has 20 years of experience in software development

cdonner@molecular.com

*Dynamic***PDF**™ Merger v3.0
Our flexible and highly efficient class library for manipulating and adding new content to existing PDFs is available natively for Java
- Intuitive object model
- PDF Manipulation (Merging & Splitting)
- Document Stamping
- Page placing, rotating, scaling and clipping
- Form-filling, merging and flattening
- Personalizing Content
- Use existing PDF pages as templates
- Seamless integration with the Generator API

*Dynamic***PDF**™ Generator v3.0
Our popular and highly efficient class library for real time PDF creation is available natively for Java
- Intuitive object model
- Unicode and CJK font support
- Font embedding and subsetting
- PDF encryption ◆ 18 bar code symbologies
- Custom page element API ◆ HTML text formatting
- Flexible document templating

Try our free Community Edition!

*Dynamic***PDF**
www.dynamicpdf.com

*Dynamic***PDF**™ components will revolutionize the way your enterprise applications and websites handle printable output. *Dynamic***PDF**™ is available natively for Java.

ceTesoftware
INFINITE POSSIBILITIES
800.631.5006 +1 301.656.1331

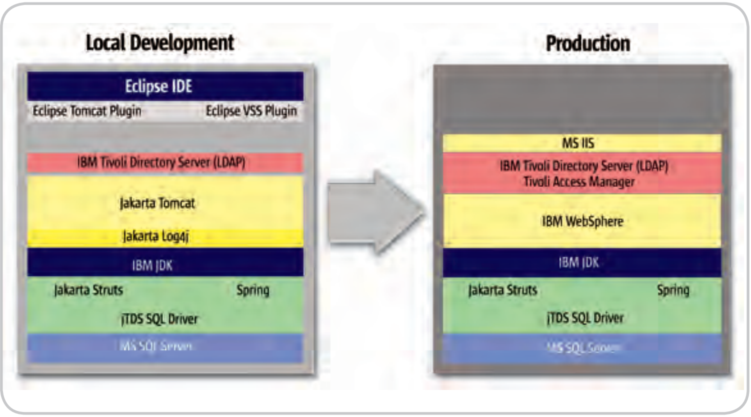


Figure 1 Development and Production Stack

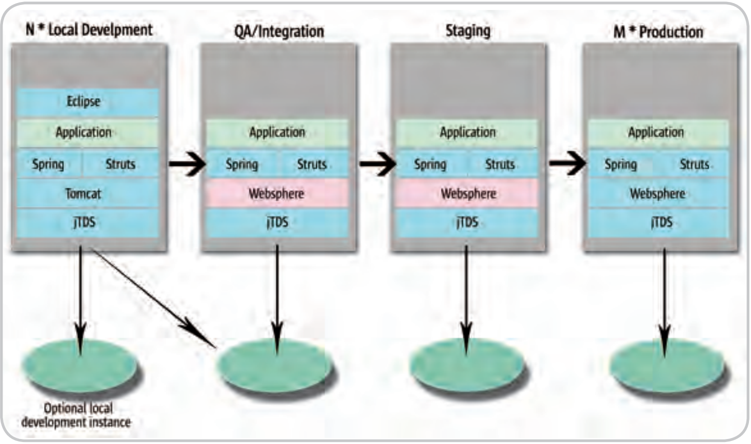


Figure 2 Multiple Environments

2. Create a separate properties file for each host with environment-specific values. For example, JDBC property definitions for *serverA* might be defined in *serverA.properties* as follows:

```
#
# Database overrides
#
jdbc.driver.classname = net.sourceforge.jtds.jdbc.Driver
jdbc.driver.type      = jtds
jdbc.server.type      = sqlserver
jdbc.server.port      = 1433
jdbc.server.host      = db01
jdbc.username          = db01-user
jdbc.password          = db01-pwd
```

3. Create Ant filter token definitions that use the environment-specific properties:

```
<filterset id="project.filter.tokens">
  <!-- DB Service(s) -->
  <filter token="JDBC.DRIVER.CLASSNAME" value="${jdbc.driver.class-
name}"/>
  <filter token="JDBC.DRIVER.TYPE"      value="${jdbc.driver.
type}"/>
  <filter token="JDBC.SERVER.TYPE"      value="${jdbc.server.
type}"/>
  <filter token="JDBC.SERVER.HOST"      value="${jdbc.server.
host}"/>
  <filter token="JDBC.SERVER.PORT"      value="${jdbc.server.
port}"/>
  <filter token="JDBC.USERNAME"         value="${jdbc.username}"/>
  <filter token="JDBC.PASSWORD"         value="${jdbc.password}"/>
</filterset>
```

4. Create a properties file containing the filter tokens. Ant will substitute actual values for the tokens:

```
jdbc.driverClassName = @JDBC.DRIVER.CLASSNAME@
jdbc.url = jdbc:@JDBC.DRIVER.TYPE@:@JDBC.SERVER.TYPE@://@JDBC.
SERVER.HOST@:@JDBC.SERVER.PORT@
jdbc.username = @JDBC.USERNAME@
jdbc.password = @JDBC.PASSWORD@
```

5. Place a *copy* task in some target that invokes the filter token substitution (<filterset>):

```
<target name="copy-files" depends="">
  <!-- Copy, with overwrite, properties and xml files
    -  so that configuration changes via Ant build properties
    -  will always be picked up.
  -->
  <copy todir="${web.build.dir}" overwrite="yes">
    <fileset dir="${web.src.dir}">
      <include name="**/*.properties" />
      <include name="**/*.xml" />
    </fileset>
    <filterset refid="project.filter.tokens" />
  </copy>
</target>
```

When the application is packaged, it looks in (among other places) *${web.build.dir}* for files to include in the Web application archive (WAR). There, it will find the generated runtime resources with environment-specific values.

Spring

The Spring Framework was very useful in allowing us to develop our application in a container-agnostic fashion. We took advantage of several of the many features of Spring.

1. ***Service Location***

We used Spring application contexts for the integration with Struts, for deployments to Tomcat and WebSphere, in standalone utility applications, and even in JUnit tests. Spring allowed us to standardize how our service objects were found and initialized across all uses of those objects in a compelling way.

2. ***Bean Life Cycle and Dependency Management***

By using Spring's application contexts, we successfully avoided stateless session beans that would have caused deployment issues across containers (not to mention the fact that Tomcat would not have readily supported EJBs).

Jim Lamoureaux is a senior consultant and software architect at Molecular. His interests include object-oriented design and implementation, programming languages, and software process. Jim is a Sun Certified Programmer for the Java 2 Platform. He currently lives in Southern New Hampshire.

jim@molecular.com

Configuring Ant for Deployments Between Different Application Servers

We used Ant (Ant 1.6+) to manage configuration, builds, and deployments from local development environments to the integration server, from there to the staging server, and finally to production. The ant scripts needed to handle two main server differences:

1. The *WEB-INF/lib* directory had to be populated with any JARs not provided by the application server. Specifically, our Tomcat environment required the optional JDBC 2.0 Package while WebSphere already came with the necessary classes installed.
2. The security-* elements of the Web deployment descriptor (web.xml) needed to include security-role definitions for deployments to Tomcat. In WebSphere, the security roles were defined at the enterprise application level (application.xml).

The solution was to treat any environment dependencies through parameters and to create configuration files that contained all settings for a server type. We laid the groundwork by explicitly providing a value for the *server.type* Ant property:

```
<!-- Server Type property-override customizations (if any) -->
<property name="server.type.config.file" location="${build.modules.home}/
deployment/servertypes/${server.type}.properties"/>
<echo message="server.type.config.file=${server.type.config.file}"/>
<property file="${server.type.config.file}"/>
```

Having a separate properties-file for each server type was helpful, because it made the deployment process agnostic of the type of server that we deployed to. The main property set in each of these files was *deploy.tomcat* or *deploy.websphere* (essentially *deploy.server-type*). Having these properties allowed us to configure the *build-war* macro according to the server type to handle the inclusion/exclusion of the JDBC 2.0 optional package (see Listing 1).

Only one of the war-* targets is being called depending upon whether the *deploy.websphere* property is defined or not. This results in a macro definition of *build-war*, which has been configured for the target server.

Similarly simply, the appropriate definitions for the security-* elements of the web.xml are handled according to the value of *server.type*.

```
<!-- Copy the environment-specific version of the web-security.xml XDoclet
merge file -->
<target name="web-security-websphere" if="deploy.websphere">
  <copy file="${web.merge.dir}/was-web-security.xml"
      tofile="${web.merge.dir}/web-security.xml" overwrite="yes"/>
</target>
<target name="web-security-tomcat" unless="deploy.websphere">
  <copy file="${web.merge.dir}/tomcat-web-security.xml"
      tofile="${web.merge.dir}/web-security.xml" overwrite="yes"/>
</target>
```

The targets *web-security-tomcat* and *web-security-websphere* are then named as dependencies in other targets that use the XDoclet *webdoclet* task (which uses the *web-security.xml* deployment descriptor snippet).

Listing 1: Ant macro for building a WAR file

```
<!-- Call the build-war macro that is defined by the dependencies
-->
<target name="package-web"
     depends="webdoclet,war-tomcat,war-websphere">
  <build-war/>
</target>

<!-- Setup the build-war macro for a tomcat deploy -->
<target name="war-tomcat" depends="" unless="deploy.websphere">
  <macrodef name="build-war">
    <sequential>
      <war destfile="${web.dist.dir}/${web.war}"
             webxml="${web.build.dir}/WEB-INF/web.xml"
           compress="true">
        <fileset dir="${web.build.dir}"    excludes="**/web.xml" />
        <webinf  dir="${struts.dir}"       includes="validator.
xml,*.dtd" />
        <lib     dir="${cfmx.dir}"          includes="*.jar" />
        <lib     dir="${commons-lang.dir}" includes="*.jar" />
        <lib     dir="${dist.dir}"          includes="${dist.name}"
/>
        <lib     dir="${jstl.lib.dir}"     includes="*.jar" />
        <lib     dir="${struts.dir}"       includes="*.jar" />
        <lib    file="${commons-dbcp.jar}"/>
        <lib    file="${commons-pool.jar}"/>
        <lib    file="${log4j.jar}" />
        <lib    file="${spring.jar}" />
        <lib    file="${jdbc.jar}"/>
        <lib    file="${jtds.jar}"/>
      </war>
    </sequential>
  </macrodef>
</target>

<!-- Setup the build-war macro for a WebSphere deploy -->
<target name="war-websphere" depends="" if="deploy.websphere">
  <macrodef name="build-war">
    <sequential>
      <war destfile="${web.dist.dir}/${web.war}"
             webxml="${web.build.dir}/WEB-INF/web.xml"
           compress="true">
        <fileset dir="${web.build.dir}"    excludes="**/web.xml" />
        <webinf  dir="${struts.dir}"       includes="validator.
          xml, *.dtd" />
        <lib     dir="${commons-lang.dir}" includes="*.jar" />
        <lib     dir="${dist.dir}"          includes="${dist.name}"
          />
        <lib     dir="${jstl.lib.dir}"     includes="*.jar" />
        <lib     dir="${struts.dir}"       includes="*.jar" />
        <lib    file="${commons-dbcp.jar}"/>
        <lib    file="${commons-pool.jar}"/>
        <lib    file="${log4j.jar}" />
        <lib    file="${spring.jar}" />
        <lib    file="${jtds.jar}"/>
      </war>
    </sequential>
  </macrodef>
</target>
```

3. ***JDBC Template Code***

The Spring JDBC APIs allowed for facile database coding – much cleaner code and standardization along the lines of connection management and exception handling.

4. ***Flexible Data Sources***

The Spring model of using beans to wire together dependent objects allowed us to use extra-container data sources. This came with the benefit of standardized usage of data sources across our runtime scenarios – no fiddling around with container-specific data source configuration.

Jakarta Commons Logging API

We used the Jakarta Commons Logging API from the beginning. It provides a very useful abstraction of typical logging needs while supplying useful hooks for plugging in various logging services such as Log4j, the Java Logging API, etc. WebSphere even provides a gateway to its own tracing facility. The ws-commons-logging.jar in the lib directory off the WebSphere installation root directory allows for logging of classes to be controlled via the WebSphere Administrative Console – as long as those classes were coded to use the Jakarta Commons Logging API.

Commons Logging allowed us to configure which plugin to use (e.g., Log4J in a Tomcat environment,

Tivoli Access Manager

The production security configuration followed the recommendations for Tivoli implementations published by IBM. The setup consisted of two WebSEAL servers, two Web/application servers, one policy server, and a master/replica LDAP configuration. The application servers hosted all of the applications with WebSEAL tying to each application through an IP/Port specific junction (a "junction" is a resource mapping and defines the true location of a URI). This necessitates multiple network cards in the WebSEAL machines in order to support multiple host addresses that are on the standard Web port.

Each production WebSEAL instance had numerous junctions configured to the multiple applications. The configuration was also set up for failover by ensuring that the server UUID configured in the junctions matched on each machine; therefore cookies for session fail-over could be picked up by either WebSEAL instance.

Choosing to install the Authorization Server on each application server created policy server redundancy. The authorization servers act as a replica of Policy server information. As a default, when the authorization server is installed, the application server does not hit the policy server directly in most cases because it obtains authorization information directly from the authorization server. The only time the policy server is reached is for any account updates. All these settings can be found in a configuration file (webseald.conf). Choosing to follow the authorization server route ensures application availability in case the policy server is down – it's a more economical method for fail-over than a master/replica policy server configuration.

WebSphere logging in that environment) and – via its default implementation that simply writes to the console – to trace unit test code without the need to configure or enable a logging service. In addition, we were able to completely turn off logging via configuration files. (This is done by placing a file called commons-logging.properties on the classpath with the line org.apache.commons.logging.Log=org.apache.commons.logging.impl.NoOpLog in it). In fact, this was the standard configuration for running our unit tests, which were run as part of every build. Of course, if a unit test failed, logging could be turned on again as a diagnostic tactic by setting org.apache.commons.logging.Log=org.apache.commons.logging.impl.SimpleLog.

What About the Risks?

After a year of real-life experience of developing several applications and performing multiple production deployments with this configuration, we feel that developing on Tomcat and deploying to WebSphere is a low-risk strategy. Once the environments were set up and the deployment process automated, there were very few problems. Spring provided the necessary container capabilities that we needed in a portable way. We were able to take advantage of Spring's bean management, service locator, data source, and JDBC abstractions.

Differences in the security infrastructure were overcome by using Tomcat's built-in features and by providing stub code in the service layer of the development environment that simulated the presence of TAM. We were able to use the same LDAP server in development (with Tomcat security) and in QA (with Tivoli Access Manager security).

Over time, the cost-saving aspect of cross-platform development became less important in favor of other advantages that were initially not anticipated. The lightweight development environment turned out to be a great advantage, and being forced to layer the application architecture to achieve isolation from the container produced cleaner and better maintainable application code – something that reduced the overall project risks, not increased them.

Summary

It takes a good amount of planning to develop on Tomcat and successfully deploy to a WebSphere environment. Open source frameworks, such as Spring and Struts, can be used to shield an application from platform-dependent implementation details. Ant is a handy tool that facilitates cross-platform deployments. Special consideration is required to handle application security across different platforms. Coding guidelines designed to avoid platform-dependencies must be followed rigorously.

With all these things in mind, cross-platform Java development is a rewarding goal, because your resulting application will be cleaner, easier to maintain, and can provide a real cost advantage.

Resources and Links

- *JDBC package for Tomcat with JVM 1.3.1:* http://java.sun.com/products/jdbc/articles/package2.html
- *IBM WebSphere:* http://www.ibm.com/developerworks/websphere
- *IBM Tivoli Access Manager:* http://www.ibm.com/developerworks/tivoli
- *Struts:* http://struts.apache.org
- *Spring Framework:* http://www.springframework.org
- *Commons Logging:* http://jakarta.apache.org/commons/logging
- *jTDS JDBC Driver:* http://jtds.sourceforge.net/
- *Info Center for Tivoli – with related replication/failover configurations:* http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.itame.doc_5.1/toc.xml
- *Great detailed intro to Tivoli Access Manager – must read for anyone considering TAM:* http://www.redbooks.ibm.com/redpapers/pdfs/redp3677.pdf

Ilia Papas is a software engineer at Molecular. He has been working with web applications for five years and has interests in the design and implementation of enterprise applications using a variety of technologies. He currently lives in the Boston area.

ipapas@molecular.com

Dita Vyslouzil is a Consultant and Technical Architect in the Engineering group at Molecular in Watertown. She has been in software development for 7 years, concentrating in transactional web applications.

dvyslouzil@molecular.com

"Once properly configured, Eclipse can be a powerful hub for developing your application"

Think Crystal is a good reporting solution for your Java application? Think again.

>>> Think JReport®

Deliver Intuitive, Easy-to-Understand Reports

- Distribute interactive reports via any Web browser with JReport's 100% DHTML client; no ActiveX or other client downloads needed to support filter, sort, search and drill.
- Reduce the report maintenance burden on developers since users can customize their reports.
- Cascading parameters provide users with run-time flexibility.

Seamlessly Integrate with Your Java Infrastructure

- 100% pure Java architecture, J2EE-compliant and a Swing-based report design tool to leverage your existing Integrated Development Environment (IDE).
- APIs exposed to support integration into your application.
- Modular and re-usable components enable you to embed customized reporting in your application.

Robust Security for Enterprise Deployment

- Security permissions controlled by groups, roles, realms, or users to row-, column- and cell-level for each report.
- Synchronization with external sources allows you to leverage existing security methods like LDAP, Active Directory, Lotus Domino and Novell Directory Server.
- JReport Security API for complete integration with existing security systems.

Support Enterprise Service Level Requirements

- Clustering, failover and load balancing.
- Scales from single-CPU to large, multi-CPU and clustered server environments.
- Report deployment and management with on-demand report viewing, scheduling and version control.
- Export reports from a single template into a variety of formats including DHTML, HTML, PDF, Excel, RTF and CSV.

Forcing a Windows-based reporting tool into your J2EE environment is like putting a square peg into a round hole. You can do it, but why go through the pain for something that doesn't fit.

Only JReport® is a 100% pure Java-based enterprise reporting solution that meets all of your end-user requirements today and provides a stable, standards-based J2EE platform for the future.

See for yourself how easily you can embed our enterprise-class reporting engine into your Java application.

Download a FREE, fully functional copy of JReport at www.jinfonet.com/jp10.htm or call (301) 838-5560.

JReport®
JINFONET SOFTWARE

How to Provide Dynamic Security Permissions

Two approaches

by Xiaozhong Wang

For various reasons, an application may install a security manager. Usually it does so to guard against malicious third-party code either installed or dynamically downloaded at runtime. If the application uses RMI APIs, it's even required by a Java specification that a security manager be installed, otherwise the classloader will not download any classes from remote locations.

The most convenient security manager to use is java.lang.SecurityManager. Once installed, it will work with security policy to control the security permissions granted to different protection domains. For simplicity, it will be referred to as SecurityManager for the rest of this article.

The security policy is statically initialized at application start-up. For Sun's JDK, the security policy is defined in a security policy file. Naturally, this initial security policy cannot be changed at runtime once it's loaded with the application.

What if you want the security permissions to change at runtime? For example, you have a list of hosts from which the socket connection requests should not be accepted by the security manager. This list keeps changing when the application is running and you don't want to shut the application down to make the latest list effective. Or you feel that the expressions allowed in the security policy file are not enough for your application. Sure, it allows wildcards like "*", but you need something more dynamic and powerful, like a regular expression. What can you do?

Before any solution is proposed, let's take a look at how security permissions are managed normally. First, create a security policy that defines a set of security permissions granted to one or more protection domains, then install java.lang.SecurityManager at the start of your application. When the application calls a security-sensitive API, the API first checks with the SecurityManager to determine whether certain operations are allowed. The SecurityManager calls AccessContoller.checkPermission() method, which in turn consults the security policy when making security permission decisions.

It's not difficult to find out from the above that three components work together to provide security permissions – a security manager, a security policy, and the AccessContoller. AccessController is a final class and cannot be dynamically set with the system, so there's nothing we can do about it. SecurityManager and Policy, on the other hand, are extendable and can be set with the system.

It seems there are two approaches – writing your own security manager or writing your own security policy.

Writing Your Own Security Manager

If you take a look at SecurityManager APIs, the bulk of them are two checkPermission() methods and some check*Operation*() methods, where *Operation* is an action like Connect, Listen, SetFactory, etc. If the security permission is granted, these methods simply return without doing anything. Otherwise, they throw SecurityException to indicate that the related security permission is denied. To dynamically control the behavior, just override one or more such APIs. If a method is not overridden, leave the behavior to SecurityManager and essentially the security policy to decide.

So far this seems easy. Is that so? Let's find out with a simple example. In this example, you want to control which properties can be accessed by overriding the checkPropertyAccess(String key) API. It's assumed that the list of accessible properties keeps changing and you get a fresh list each time checkPropertyAccess(String key) is invoked (see Listing 1).

You don't expect to get a security exception because we allow access to "user.home". By the way, if you use a security policy file that grants PropertyPermission to access "user.home" and "user.dir" and install a SecurityManager, TestProperty prints out the value of "user.home" just as expected.

If you run TestProperty with MySecurityManager in Sun's JDK 1.4.2, it prints out the following:

```
Exception in thread "main" java.lang.Excep
tionInInitializerError
 at java.lang.System.setSecurityManager0(Sy
```

Xiaozhong Wang is a software engineer at Sun where he has solved some security problems in his TCK (Technology Compatibility Kit) work.

xiaozhong.wang@sun.com

"An application may need a security manager that is less restrictive than the initial security policy at certain times"

Your Mission: Rescue Business Logic from the Black Hole of Static Spreadsheets...

Your Solution: Spreadsheet Automation with ExtenXLS

Astronaut Training Greatly Reduced!

With the familiar look and feel of Excel, ExtenXLS eliminates the learning curve imposed on your end-users by other reporting tools. ExtenXLS unlocks the business logic stored in static spreadsheets throughout your organization, saving time & money.

Put a Rocket Under the Hood!

ExtenXLS makes recreating Excel formulas in Java obsolete. Avoid errors and save time by converting existing spreadsheets to Java WorkBook objects. Update formulas without recompiling, set the value of Cells, and execute formulas all in **100% Java code**.

Escape the Gravitational Pull of Obsolete Reporting!

Output to customized **HTML** for maximum compatibility, or to **XML** for further processing. Keep your users happy with native Excel output which preserves the **VB macros**, **images**, **charts** and other features that transform live data into actionable reports. You can even embed a familiar spreadsheet component in your **Swing applications**. You have now achieved escape velocity.

Beam ExtenXLS Down Now!

Focus your tractor beam on **www.extentech.com/jdj** and take your free 30-day evaluation copy for a test flight.

"ExtenXLS makes it possible to re-use spreadsheets representing a tremendous investment in time and business logic and to 'connect' those spreadsheets to various SQL data sources to provide access to usable data in a way never before possible."

- Project Leader, AT&T

"We use ExtenXLS for 'pushing' accounting data into an Excel template. A key customer wanted Web access to Excel format reports. We tried .NET but had problems with preserving charts and formatting. ExtenXLS provides a flexible, cost effective solution that gives us the ability to stay ahead of user requirements".

- Project Leader, John J. McMullen Company

EXTENXLS 4™
JAVA | XLS REPORTING TOOLKIT

extenTECH™
Call Us: 415-759-5292

```
stem.java:243)
 at java.lang.System.setSecurityManager(S
ystem.java:212)
 at TestProperty.main(TestProperty.
java:5)
Caused by: java.lang.SecurityException:
Not allowed!
 at MySecurityManager.checkPropertyAccess
(MySecurityManager.java:9)
 at java.lang.System.getProperty(System.
java:573)
 at java.lang.Integer.getInteger(Integer.
java:814)
 at java.lang.Integer.getInteger(Integer.
java:731)
 at sun.security.action.GetIntegerAction.
run(GetIntegerAction.java:90)
 at java.security.AccessController.
doPrivileged(Native Method)
 at sun.net.InetAddressCachePolicy.<clini
t>(InetAddressCachePolicy.java:94)
 ... 3 more
```

The exception is thrown from System.setSecurityManager() and is caused by a read of property "sun.net.inetaddr.ttl", which is totally unrelated to our code. Seems like you just shot yourself in the foot, yet you don't know where the bullet came from.

Actually it's not important to know where the check comes from, but it is important to note that the security exception is caused by an AccessController.doPrivileged() call.

When we try the TestProperty application with the standard SecurityManager and security policy, AccessController.doPrivileged doesn't throw a security exception. This is because SecurityManager.checkPropertyAccess() delegates to checkPermission(), which in turn calls AccessController.checkPermission(). AccessController knows how to handle privileged code blocks. When it sees a privileged code block and the associated protection domain has the required permission, it returns without further checking callers of the privileged code block on the call stack. In our case, the privileged code block is in the sun.net.InetAddressCachePolicy, which is from the system domain that has all the permissions.

Let's go back to MySecurityManager. There is no way for it to know whether a call is from a privileged code block or the information about the call stack. It grants and denies the same set of permissions to all protection domains, even if the protection domain is the system domain where all permissions should be granted. That's where the problem comes from.

For more details regarding AccessController, I encourage you to check out the JavaDoc for the AccessController and security documentation at http://java.sun.com/j2se/1.4.2/docs/guide/security/spec/security-spec.doc4.html#20389.

It's important to note that MySecurityManager tends to be more restrictive than SecurityManager (or the initial security policy) by specifically disallowing access to most of the properties. On the other hand, an application may need a security manager that is less restrictive than the initial security policy at certain times. In this case, override a SecurityManager's check method in the following manner:

1. Specifically allow an action by directly returning from the method when a condition is met.
2. Otherwise delegate to the same check*Operation*() method in its super class to get the default behavior controlled by the initial security policy.

Listing 2 is an example of a security manager that is less restrictive than SecurityManager. The nice thing about a less restrictive security manager is that it can correctly handle a privileged code block as it calls super.checkPropertyAccess(), which eventually calls AccessController.checkPermission(). But that's not the end of the story. If you use this security manager with our TestProperty application, you still get strange SecurityException:

```
Exception in thread "main" java.lang.
ExceptionInInitializerError
 at java.lang.System.setSecurityManager0(
System.java:243)
 at java.lang.System.setSecurityManager(S
ystem.java:212)
 at TestProperty.main(TestProperty.
java:5)
Caused by: java.security.
AccessControlException: access denied
(java.util.PropertyPermission sun.net.
inetaddr.ttl read)
 at java.security.AccessControlContext.
checkPermission(AccessControlContext.
java:269)
 at java.security.AccessController.checkP
ermission(AccessController.java:401)
 at java.lang.SecurityManager.checkPermis
sion(SecurityManager.java:524)
 at java.lang.SecurityManager.checkProper
tyAccess(SecurityManager.java:1276)
 at MySecurityManager.checkPropertyAccess
(MySecurityManager.java:9)
 at java.lang.System.getProperty(System.
java:573)
 at java.lang.Integer.getInteger(Integer.
java:814)
 at java.lang.Integer.getInteger(Integer.
java:731)
 at sun.security.action.GetIntegerAction.
run(GetIntegerAction.java:90)
 at java.security.AccessController.
doPrivileged(Native Method)
 at sun.net.InetAddressCachePolicy.<clini
t>(InetAddressCachePolicy.java:94)
 ... 3 more
```

Guess what? On the call stack, MySecurityManager.checkPropertyAccess() is the caller of AccessController.checkPermission(). And MySecurityManager is in a protection domain that does *not* have all permissions like system domain does. In our case, the default security policy does not grant the required permission (java.util.PropertyPermission sun.net.inetaddr.ttl read) to MySecurityManager's

"Basically there is no way to let a more restrictive security manager properly handle a privileged code block" protection domain. AccessController immediately throws the exception without further checking the call stack.

A quick remedy for this is to put MySecurityManager under a trusted classpath, where the classes are granted all granted permissions. In Sun's JDK, it can be achieved by using -Xbootclasspath/a: command line operation. Once this is applied, the application happily prints out the value of system property "user.home".

What have we learned? Basically there is no way to let a more restrictive security manager properly handle a privileged code block. Given that there are a lot of privileged code blocks in a Java implementation and possibly in applications, you could find a lot strange behaviors that are transparent to you under a normal security environment. It's generally not preferable to use such a security manager with your application. On the other hand, it is possible to use a less restrictive security manager with your application but not without minor deployment overhead.

Another piece of advice: check*Operation*() methods are JDK 1.1 style APIs and are less preferable than checkPermission() methods, which were introduced in Java 2. According to the SecurityManager specification, all calls to check*Operation*() methods delegate to checkPermission() methods with an appropriate permission object as an argument. Overriding checkPermission() will provide the same functionality as overriding check*Operation*(). For simplicity, Listing 2 does not override the checkPermission() methods. This will only work if the application or the underlying Java API implementation uses check*Operation*() APIs. The specification does not forbid the direct use of checkPermission() methods (in which case it makes overriding check*Operation*() methods useless) in applications and new Java API implementations. Therefore, it's more efficient and much safer to override checkPermission() methods instead.

Writing Your Own Security Policy

We just mentioned that it's not preferable to use a security manager that is more restrictive than SecurityManager. But what can we do if we need more restrictive permissions? The answer: write your own security policy that extends java.security.Policy and sets it with the system. This option has become available since Sun's JDK 1.4 as it starts to support security policy set at runtime after loading the initial security policy.

Let's start by looking at the Policy specification. There are two abstract methods in Policy:

```
getPermissions(CodeSource codesource)
refresh()
```

It seems we only need to provide the implementation to these methods. However, in practice, getPermissions(ProtectionDomain domain) and implies(ProtectionDomain domain, Permission permission) should also be overridden to avoid the inconsistency caused by a particular caching strategy in a Java implementation. For example, in Sun's JDK, the implies() method sometimes returns the result from an internal cache, which may contain stale permissions granted to a particular protection domain, without calling the getPermissions() methods to get updated security permissions.

Listing 3 provides an example of a policy that provides dynamic security permissions for accessing certain properties. Please note that in order to implement getPermissions (CodeSource codesource), a customized implementation of PermissionCollection has to be available as well.

Apparently, this implementation is far more complicated than the security manager approach. However, it gives you more control of the runtime security permissions – they can be more restrictive or less restrictive than the ones specified in the initial security policy.

The application successfully prints out the property value for "user.home". If you want to print out the value for the property "user.name", it throws a SecurityException as expected:

```
Exception in thread "main" java.security.AccessControlException:
access denied (java.util.PropertyPermission user.name read)
 at java.security.AccessControlContext.checkPermission(AccessControl
Context.java:269)
 at java.security.AccessController.checkPermission(AccessController.
java:401)
 at java.lang.SecurityManager.checkPermission(SecurityManager.
java:524)
 at java.lang.SecurityManager.checkPropertyAccess(SecurityManager.
java:1276)
 at java.lang.System.getProperty(System.java:573)
 at TestPolicy.main(TestPolicy.java:72)
```

We've got problems with your name on them.

At Google, we process the world's information and make it accessible to the world's population. As you might imagine, this task poses considerable challenges. Maybe you can help.

We're looking for experienced software engineers with superb design and implementation skills and expertise in the following areas:

- high-performance distributed systems
- operating systems
- data mining
- information retrieval
- machine learning
- and/or related areas

If you have a proven track record based on cutting-edge research and/or large-scale systems development in these areas, we have brain-bursting projects with your name on them in Mountain View, Santa Monica, New York, Bangalore, Hyderabad, Zurich and Tokyo.

Ready for the challenge of a lifetime?
Visit us at http://www.google.com/jdj for information. EOE

Google

There are two important things to note when you write your own policy:

1. Avoid using any APIs that are checked by SecurityManager in the methods of your policy implementation. In a worst case scenario, you may end up having a indefinite loop and stack overflow. If you really have to use such APIs, wrap them with a privileged code block and put your policy class under a trusted classpath.
2. Even though you have overridden all possible methods in the policy, you still can't prevent the AccessController from using the internal policy cache. One way to minimize the "bad" impact of the cache is to force a refresh of the cache by setting the policy again whenever the security permissions change. For example, you can set up a listener for the change of security permissions. When it happens, the listener takes the action to reset the policy with the system. However, if there are other threads running when the policy is reset and initialized, the threads may get incorrect security permissions from the policy simply because the policy is still in initialization. You may observe strange behaviors in your application if the security permissions change frequently.

Conclusion

To sum up, you can either override the SecurityManager or Policy to provide dynamic runtime security permissions. The security manager approach is suitable if the runtime security permissions tend to be less restrictive than the initial security policy. The security policy approach is suitable for either more restrictive or less restrictive security permissions, but you need to watch for and take extra steps to prevent unexpected behavior due to a policy cache in the underlying Java implementation. Your own policy may not work well if the security permissions change frequently for a multi-threaded application.

Listing 1

```
import java.security.*;

public class TestProperty {
    public static void main(String[] args) throws Exception {
        System.setSecurityManager(new MySecurityManager());

        System.out.println(System.getProperty("user.home"));
    }
}

public class MySecurityManager extends SecurityManager {
    public void checkPropertyAccess(String key) {
        String[] allowed = getAllowedProperties();
        for (int i = 0; i < allowed.length; i++) {
            if (key.equals(allowed[i])) {
                return;
            }
        }
        throw new SecurityException("Not allowed!");
    }

    private String[] getAllowedProperties() {
        return new String[] {"user.home", "user.dir"};
    }
}
```

Listing 2

```
public class MySecurityManager extends SecurityManager {
    public void checkPropertyAccess(String key) {
        String[] allowed = getAllowedProperties();
        for (int i = 0; i < allowed.length; i++) {
            if (key.equals(allowed[i])) {
                return;
            }
        }
        super.checkPropertyAccess(key);
    }
    // This method is supposed to return different things
    // at different times
    private String[] getAllowedProperties() {
        return new String[] {"user.home", "user.dir"};
    }
}
```

Listing 3

```
import java.security.*;
import java.util.*;

public class TestPolicy extends Policy {
    private static CodeSource appCodeSource;
    private static PermissionCollection permissions;
    private static Permissions allPermissions;

    // assume that TestPolicy and the application are from the
same code source
    static {
        appCodeSource = TestPolicy.class.getProtectionDomain().
getCodeSource();
        permissions = new MyPermissionCollection();
        allPermissions = new Permissions();
        allPermissions.add(new AllPermission());
    }

    static class MyPermissionCollection extends
PermissionCollection {
        public void add(Permission permission) {
        }

        public boolean implies(Permission permission) {
            if (permission instanceof PropertyPermission) {
                return getAllowedPropertyPermissions().
implies(permission);
            }
            // we allow all other permissions
            return true;
        }
        public Enumeration elements() {
            return new Enumeration() {
                public boolean hasMoreElements() {
                    return false;
                }

                public Object nextElement() {
                    return null;
                }
            };
        }

        // This method is supposed to return dynamic results
        private Permissions getAllowedPropertyPermissions()
{
            Permissions perms = new Permissions();
            perms.add(new PropertyPermission(“user.dir”,
“read”));
            perms.add(new PropertyPermission(“user.home”,
“read”));
            return perms;
        }
    };

    public PermissionCollection getPermissions(CodeSource
codesource) {
        if (appCodeSource.equals(codesource)) {
            return permissions;
        }
        return allPermissions;
    }

    public boolean implies(ProtectionDomain domain,
Permission permission) {
        return getPermissions(domain.getCodeSource()).
implies(permission);
    }

    public PermissionCollection getPermissions(ProtectionDom
ain domain) {
        return getPermissions(domain.getCodeSource());
    }

    public void refresh() {
    }

    public static void main(String[] args) throws Exception
{
        Policy.setPolicy(new TestPolicy());
        System.setSecurityManager(new SecurityManager());

        System.out.println(System.getProperty(“user.home”));
    }
}
```

The Power Is Here

and It’s Stronger Than Ever

Oracle Development Tools User Group

Plug Into

- *Java Development*
- *Business Intelligence*
- *Data Warehousing*
- *Web Development*
- *PL/SQL*
- *Methodology, Modeling, and Analysis*
- *All tools that work with an Oracle Database*

Still using Designer or Oracle Forms and Reports? The ODTUG membership network provides you with the support you need.

www.odtug.com

UPCOMING EVENTS

Desktop Conference 2006
You’re Virtually There! – February 21-23

ODTUG NOW! 2006 – June 2006

Sponsor of Oracle PL/SQL Programming Conference
November 2-3, 2005

© 2005 Oracle Development Tools User Group

JDBC –The Indispensable Component of Persistence Mechanisms

It's critical for the success of the O/R infrastructure

by Jonathan Bruce

Few serious database applications are considered enterprise-worthy without a core database engine backed by a normalized and optimized relational database architecture. Traditionally, such database applications rely on SQL statements to retrieve and update data in the back-end data source.

This well-established model continues to have strategic importance and is the basis for the continued growth of object/relational mapping and its associated persistence mechanisms. What is object/relational mapping (O/R mapping)? It's a programming technique that links a relational database to object-oriented language concepts. O/R mapping mechanisms allow a developer to create Java objects using an object-oriented perspective, eliminating the impedance mismatch that exists between object and relational models.

O/R mapping is a compelling solution for you, as a Java developer, because you can concentrate on building the application while leaving the data persistence details to the O/R mapping mechanism.

As a Java developer, you have a variety of choices when it comes to Java-based O/R mapping mechanisms. Three communities or organizations are most active in the Java O/R persistence world: open source communities; standards-based organizations, and commercial ventures.

Open source includes prominent implementations such as Hibernate and the Spring Framework. Standards-based implementations are technologies such as EJB 3.0 (JSR-220) and JDO (JSR-244). Commercial implementations of note include Oracle's TopLink. In this article, we take a look at the open source and standards-based options to understand the specific nuances that you should consider when selecting an O/R mapping mechanism for your environment.

Regardless of the O/R mapping mechanism, they all leverage JDBC to communicate with the underlying relational database. Although it is possible to use an O/R mapping mechanism to access non-relational data sources, the vast majority of applications leverage relational databases. With that in mind, you need to carefully consider each layer in the software stack to ensure an optimal O/R persistence design. As you'll see, each O/R mapping mechanism has a singular dependency on the JDBC driver to communicate data to and from the database in a highly efficient manner. If a suboptimal JDBC driver is used, gains in developer efficiencies are not complemented by a fast-performing, highly scalable application. Selecting the best-performing, most reliable JDBC driver is essential for building and deploying your applications on any O/R mapping mechanism.

Figure 1 shows a representation of the various O/R mapping mechanisms and how they relate to the application code and relational data sources. This clearly illustrates the critical role fulfilled by the JDBC driver since it is used as the foundation for each of these O/R mapping mechanisms. The efficiency of the JDBC driver has a profound effect on the performance, scalability, and reliability of the applications. Each O/R mapping mechanism is fully dependent on the performance of the JDBC driver, regardless of the design of the O/R mapping mechanism API exposed to the application code, and regardless of the SQL-based optimizations the O/R mapping mechanism is able to achieve.

One of the most hotly debated and passionate discussions in the software industry is which O/R mapping mechanism deserves to be the dominant choice. This discussion has left the Java community polarized. As this debate rages on, developers and architects should select the O/R mapping mechanism that best fits the needs of their applications. This goes for each component in the architecture stack, including selecting the best JDBC driver in order to realize application success.

Before we delve into the details of O/R mapping mechanisms, it's important to acknowledge that O/R technologies do not alleviate the need for applications that are built directly on top of the JDBC

Jonathan Bruce is program manager at DataDirect Technologies. He has led and participated in four JSRs and enjoys helping Java and .NET developers take advantage of the benefits XQuery offers when working with XML and a variety of databases. Recently relocated from San Fransisco to North Carolina, Jonathan spends his weekends running, sailing, and traveling.

Jonathan.Bruce@datadirect.com

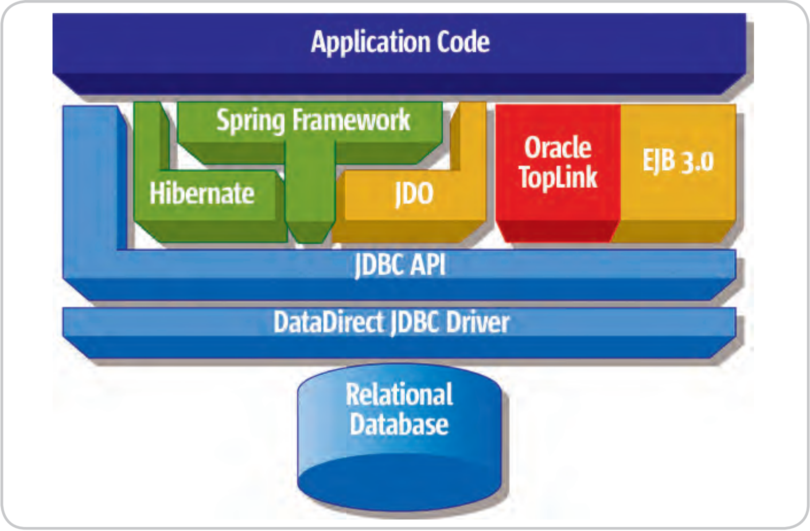


Figure 1

(JSR 054 and JSR 221) and JDBC RowSet Implementations (JSR 114). In many cases, these technologies will continue to be an optimal solution and will continue to be useful for many practical purposes. JavaServer Faces, JSP, and JSP Tab Libraries have considerable support for directly binding Web interface widgets to underlying relational structures, allowing developers to quickly and easily develop applications. A number of ease-of-development enhancements are being made that will allow organizations to continue to leverage the investments that they have made in these important technologies.

To navigate our way through the stormy waters of O/R mapping mechanisms, let's take a look at the most active open source and standards-based implementations. As we step through these mechanisms, I will describe their high-level features and highlight how many of the common features of these sometime competing technologies can be used in a complementary fashion.

Open Source: Hibernate and Spring

Hibernate

Hibernate is arguably considered the leading open source O/R mapping mechanism. Its success can be attributed to a vibrant developer community, which is widely considered to be a critical life source for any successful and prominent open source project. Hibernate provides a well-designed API that permits full persistence within the Java Object model, encompassing support for collections, inheritance, and polymorphism. For developer's considering O/R persistence for the first time, Hibernate provides the option to access the JDBC API directly, allowing greater flexibility since you can leverage the full power of SQL. Hibernate's lead developers regard this as an important asset, especially for those organizations that have made significant investments in relational database technologies.

Hibernate 2.0 is a production ready, fairly mature solution that provides the base capabilities required for O/R mapping. With the Hibernate 3.x releases, the APIs and the overall look and feel of the framework is evolving to look more like the APIs documented in the EJB 3.0 specification. The most significant contribution that Hibernate has made to EJB 3.0 design is the EntityManager, which manages the life cycle of entities or objects. This core functional component, originally inspired by Hibernate, has been incorporated into the EJB 3.0 specification and will soon become one of the standard interfaces for the upcoming Java EE 5.0 platform.

With the convergence in the overall design of Hibernate and EJB 3.0, some would argue that the future of Hibernate is bleak. After all, many of the improved interfaces, annotations, and ease-of-development considerations included in EJB 3.0 are the direct result of three iterations of the Hibernate product. I would argue that by aligning the Hibernate interfaces with EJB 3.0, Hibernate has nicely positioned itself for wide-range adoption outside of the Java EE 5.0 standard. In addition, it remains in a position to innovate and introduce bleeding-edge enhancements outside the context of a standards body.

In the following example, let's look at how to configure Hibernate with the DataDirect Connect *for* JDBC driver for the Microsoft SQL Server database. As with most JDBC-enabled applications, Hibernate permits configuration with a JDBC Connection or DataSource. First, locate the $HIBERNATE_HOME/etc/hibernate.properties file and insert the following configuration details:

```
hibernate.connection.driver_class =
 com.ddtek.jdbc.sqlserver.SQLServerDriver
hibernate.connection.url =
 jdbc:datadirect:sqlserver://server_name:1433
hibernate.connection.username = sa
hibernate.connection.password = sa
```

Next, Hibernate needs to be configured to use the correct SQL dialect. In this case, we need to configure Hibernate to function with SQL Server.

```
hibernate.dialect org.hibernate.dialect.SQLServerDialect
```

Your application can easily establish a connection to this new established data source with the following code.

```
SessionFactory sessions = cfg.buildSessionFactory();
Session session = sessions.openSession();
```

Spring Framework

The Spring Framework is a relatively recent innovation, but it already enjoys significant developer support for a variety of reasons. It addresses some of the core complaints associated with working with J2EE by absolving and abstracting the developer from the intricacies of J2EE. Spring's layered architecture is modularized, which permits you to select the components that best meets the application requirements. As described in this article, you can leverage the JDBC capabilities provided by Spring while ignoring the other layers it provides.

COMMON CONTROLS www.common-controls.com

The Java™ Presentation framework for J2EE™ Web applications

Based on:
- Java™
- Servlets™
- Java Serverpages™
- and Struts

For Free!

Get your free trial version – www.common-controls.com
See the common controls in action – go for the Online Demo!

Contains the most common control elements which are required for the development of J2EE™ applications with rich HTML frontends like:

Lists | Trees | Tabfolders | Menu | Forms | BreadCrumbs | Calendar | Colorpicker

www.common-controls.com

If you are considering O/R persistence on top of JDBC, the Spring Framework Data Access Objects (DAO) deserves a close look. The DAO consistent layer allows you to gracefully assemble object persistence directly on top of JDBC, Hibernate, or even Oracle TopLink. Spring's concept of DAO is grounded on the principle of Plain Old Java Objects, often referred to as POJOs. POJOs allow you to think in terms of Java objects while allowing the underlying persistence mechanisms to manage the application object life cycle.

Spring's DAO architecture has carefully considered JDBC and has delivered an alternative to the native JDBC that some consider difficult to manage. Particular emphasis has been given to improving error handling when dealing with underlying relational data sources, as significant improvements have been made with respect to the depth of exceptions that can be reported against any data source. These efforts have not gone unnoticed by the JDBC specification team as they are quietly being adopted in the JDBC 4.0 exception hierarchy.

The Spring Framework provides JDBC abstraction using four packages that provide the core data source object and associated support. As with any modern coding practice, Spring requires you to follow a set of design patterns. In this case, bean definition files, written in XML, are used extensively and are the primary way to configure applications using Spring.

In the following example, we use a bean definition file to configure a J2EE DataSource using JDBC DriverManager class:

```
<beans>
  <bean id="myDataSource"
    class="org.apache.commons.dbcp.
BasicDataSource"
    destroy-method="close">
  <property name="driverClassName"
    value="com.ddtek.jdbc.sqlserver.
SQLServerDriver" />
  <property name="url"
    value="jdbc:datadirect:sqlserver://<HOST>/
mydb"/>
  <property name="sa" value="sa" />
  </bean>
:
</beans>
```

We can also do this using a JDBC programmatic approach:

```
DriverManagerDataSource dataSource =
    new DriverManagerDataSource();
dataSource.setDriverClassName(
    "com.ddtek.jdbc.sqlserver.
SQLServerDriver");
dataSource.setUrl(
    "jdbc:datadirect:sqlserver://<HOST>:/
test");
dataSource.setUsername( "sa" );
dataSource.setPassword( "sa" );
```

With Spring, it is possible to layer the DAO layer on top of JDBC or an O/R Persistence mechanism such as Hibernate. Building Spring-enabled applications with Hibernate is an increasingly popular model since developers can realize increased productivity gains. With Spring's modular and layered approach, you can easily mix and match your preferred approach to data access. Spring provides JdbcDaoSupport, HibernateDaoSupport, and JdoDaoSupport classes for use of their DAO model on top of either the data access or persistence layer.

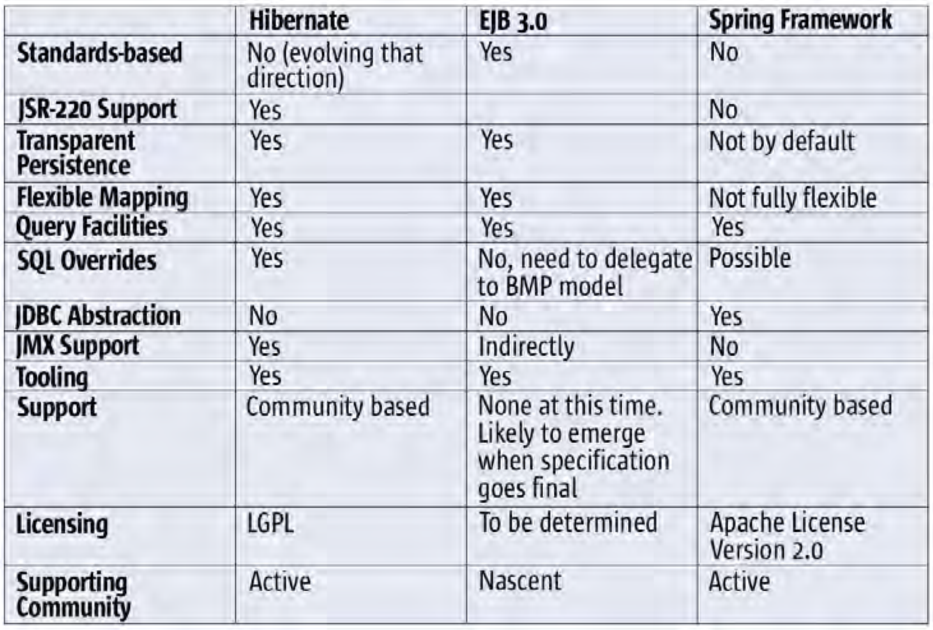
Standards Based: EJB 3.0

EJB 3.0

EJB 3.0 is a standard-based effort that has been underway since the launch of a number of JSRs sponsored by Sun Microsystems that are aimed at simplifying the J2EE platform. EJB 3.0 was originally focused on resolving many of the difficulties developers experienced with session beans and message-driven beans (MDB). Although not a foundational goal, the EJB 3.0 expert group felt it was important to establish a consistent persistence layer in EJB in an effort to quell ongoing debates about the J2EE's standard persistence model. By soliciting input directly from the O/R Mapping industry, EJB 3.0 has evolved significantly from its original specification goals. At the time of writing, EJB 3.0 Public Review draft provides a unifying Persistence layer that is the result of cooperative efforts from lead developers of Hibernate, JDO, Oracle TopLink, and others that have a vested interest in this space.

This cooperative effort is excellent news for developers and architects alike since EJB 3.0 promises to combine the best design efforts of Hibernate, JDO, and Oracle TopLink. This effort will result in the delivery of an O/R model that will become the standard for future J2EE-based development. Furthermore, EJB 3.0's persistence layer specification will be immediately recognizable to anyone who has previously worked with the established O/R mapping mechanisms. The concept of the EntityManager is almost identical to a similar idea provided by Hibernate. And, the provision of Named queries, Callback listeners, and O/R Mapping types have all been heavily influenced by the existing O/R persistence market leaders.

Although EJB 3.0 provides a standard set of interfaces, the configuration for each implementation will likely vary to some degree. In the following example, we look at how Oracle's EJB 3.0 implementation is configured with the DataDirect Connect *for* JDBC driver for the Oracle database.



	Hibernate	EJB 3.0	Spring Framework
Standards-based	No (evolving that direction)	Yes	No
JSR-220 Support	Yes		No
Transparent Persistence	Yes	Yes	Not by default
Flexible Mapping	Yes	Yes	Not fully flexible
Query Facilities	Yes	Yes	Yes
SQL Overrides	Yes	No, need to delegate to BMP model	Possible
JDBC Abstraction	No	No	Yes
JMX Support	Yes	Indirectly	No
Tooling	Yes	Yes	Yes
Support	Community based	None at this time. Likely to emerge when specification goes final	Community based
Licensing	LGPL	To be determined	Apache License Version 2.0
Supporting Community	Active	Nascent	Active

You can obtain a copy of Oracle's 10.1.3 J2EE Container (OC4J) from http://www.oracle.com/technology/software/products/ias/preview.html, and the DataDirect Connect *for* JDBC Driver for the Oracle database from http://www.datadirect.com/products/jdbc/index.ssp. Then, copy the driver JAR files to this location:

```
cp 'base.jar utils.jar and oracle.jar'
 $ORACLE_HOME/j2ee/home/applib
```

Next, create a DataSource in data-sources.xml located in the following directory:

```
 $ORACLE_HOME/j2ee/home/config
```

For an Oracle database, configure the DataDirect driver as a managed data source:

```
<managed-data-source
  name="OracleDataSource"
  jndi-name="jdbc/DDOracleDS"
  description="Managed DataSource">
 connection-pool-name="myConnectionPool"/>
```

Include the following data that defines a connection pool.

```
<connection-pool
  name="myConnectionPool"
  min-connections="10"
  max-connections="30"
  inactivity-timeout="30">
    <connection-factory
        factory-class="com.ddtek.jdbcx.oracle.OracleDataSource"
        user="scott"
        password="tiger"
        url="jdbc:datadirect:oracle://<HOST>:1521"/>
    </connection-factory>
</connection-pool>
```

Persistence Mechanisms – Comparison

Having discussed the various backgrounds of today's market-leading O/R mapping mechanisms, there are many considerations that you need to keep in mind when making your choice. Table 1 summarizes the key technical, business, and support capabilities that you will want to consider when choosing a Persistence framework.

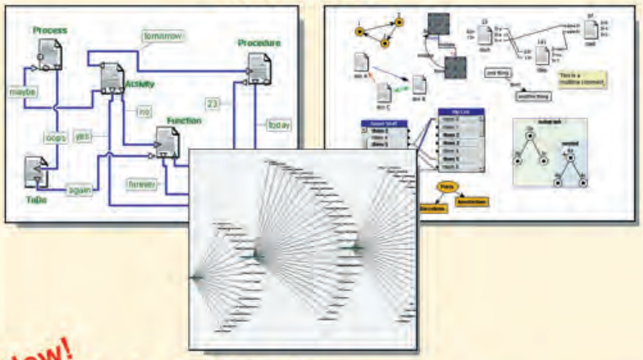
JDBC – The Critical Link

O/R technologies excel at providing an object-oriented view of relational data while eliminating the development effort relating to the persistence model. Since the O/R mapping mechanisms generate efficient JDBC calls to access the database, some people argue that the relative importance of the JDBC driver has been reduced. But, as with any architecture, the overall efficiency of the application will be greatly affected by the weakest layer in the application stack. Regardless of the JDBC code that is generated, the O/R mapping mechanisms lack the ability to control how the drivers interact with the database. At the end of the day, the efficiency of the application is in large part dependent on the ability of the JDBC driver to securely move data between the application and the database with the greatest level of performance, scalability, and reliability. The ability of the driver to efficiently communicate with the wire-level protocol exposed by the database and to efficiently manage packet transmission as it relates to fetching data and performing updates is fundamental to the application. Although there are many factors to consider when selecting a JDBC driver (interoperability, standards leadership, technical support, etc.), selecting the best possible JDBC driver based on performance, scalability, and reliability is key to realizing the benefits of applications that are based on an O/R framework.

Conclusion

O/R mapping mechanisms offer a compelling model for many developers looking to overcome the impedance mismatch associated with object and relational data constructs. O/R technologies improve developer productivity by eliminating the development effort associated with managing persistence, allowing the developer to focus on the application's business logic. Although there are many factors to consider when selecting an O/R technology, the recent trend to reconcile the O/R mapping APIs should prove beneficial to all development organizations. While some organizations are motivated by the developer productivity gains associated with using an O/R mapping mechanism, it is extremely important to the overall success of the application to leverage best-of-breed components at each layer in the architecture stack. Leveraging an industry-leading JDBC driver that provides the performance, scalability, reliability, and security that is necessary in today's applications is critical to the overall success of the O/R infrastructure.

Build Incredible Interactive Diagrams with JGo™



New! *JGo for SWT/Eclipse*
JGo Instruments for meters, dials, gauges

Create custom interactive diagrams, network editors, workflows, flowcharts, and design tools. For web servers or local applications. Designed to be easy to use and very extensible.

- ***Fully functional evaluation kit***
- ***No runtime fees***
- ***Full source code***
- ***Excellent support***

Northwoods SOFTWARE CORPORATION

Free evaluation at:
www.nwoods.com/go
800-434-9820 or 603-886-9173

Joe Winchester
Desktop Java Editor

The Usability **Paradox**

The world's first office computer, known as LEO, was created in the 1950s by Lyons, the British tea-shop giant. Its aim was to replace the thousands of clerks who did the billing, invoicing, and stocktaking, and also tracked the supply and demand of sticky buns and cups of tea that the public were consuming. Its success lay not in the technology it employed, but because it made the company more efficient by streamlining what was previously a very labor-intensive business process. It benefited Lyons, which cut costs and had more control of corporate information, and it also benefited the thirsty public who had enough cakes, sandwiches, and cups of tea to see them through their seaside weekends, rain or shine.

In much the same way that early machines automated tasks such as harvesting crops or weaving cotton, LEO was successful because it was more than just an electronic filing cabinet – it had integrated itself into the DNA of the corporation and freed up employees from manual labor.

The 1960s to 1990s was not such a cakewalk for IT, however, and the Nobel Prize–winning economist Robert Solow summarized this period: "We see computers everywhere except in the productivity statistics." He was basing his observation on the statistic that while IT spending grew in every decade since the 1960s, productivity growth slowed. In the 1990s the amount of money spent on new computer hardware alone was over $750 billion, and since none of this seemed to make companies more efficient, the expression – *The Productivity Paradox* – was coined. Despite all the money that was being spent, there was no real return on investment.

Solow did the industry a disservice as he had painted a picture of inefficiency. What has occurred since is that many corporations view IT not as an opportunity to create revenue, but as an overhead that departments have to incur and as such it should be minimized.

Such an attitude worries me deeply as the line between saving costs for the business and providing a poor customer experience is a thin one. Two examples of places where this line is wafer-thin are voice response systems and browser apps that front end legacy applications.

Anyone who has telephoned a company to deal with a request and had to navigate touch-tone options in vain knows the frustration and poor service it provides. Most companies spend a lot of money on their office's reception

area; plush furniture, nice lighting, and welcoming smiles greet customers as they walk into the business. A voice response system, however, is the virtual equivalent of a company's reception area as it creates the first impression and is the waiting lounge until you can see the person you've called to visit. While companies implement cost savings by outsourcing help desks to far-flung time zones and attempting to put their customers many touch-tone menus away from the real people left on their help-desk support staff, they are doing the equivalent of decorating the entrance hall to their corporate offices with uncomfortable chairs, shabby carpets, and impersonal service.

The Web has had a phenomenal effect on companies and how they can interact with their customers, but for many industries I fear that all that has occurred is they have front ended their batch systems and exposed inherent business weaknesses and flaws. Most of the computing universe runs on batch systems that were conceived and built in the last millennium, where nightly jobs compute numbers, move data, send messages, and print reports. Front ending this with a browser so customers can interact with their data is more efficient both for the company and the user; however, if it suffers from inherent legacy business inefficiencies, then it's no more than lipstick on a mainframe. A colleague of mine suffered this recently when on Friday they cancelled a payment that was due to be made the following Monday, only to find it had occurred anyway. The final explanation given was that three days notice was required because Monday's transactions were processed over the weekend and the job to do this started on Friday night. Listening to the story I had visions of an IT department in a deep subbasement somewhere with armies of oompah loompahs stoking a Heath Robinson Series II computer with currant buns while they drank cups of lukewarm tea.

Is the problem that IT is forever suffering from the poor return on investment that they suffered in the latter half of the last century? That it will forever be viewed as a cost center where only the minimum functionality is enough rather than a revenue-generating opportunity? Successful e-businesses understand that IT is the blood supply of their company and invest hugely in being able to deal with a world where customers exist in, travel to, and relocate around all corners of the globe and quality service must be provided 24 hours a day. For companies whose boardroom goal is to report quarterly results that boost shareholder value based on profit and loss figures, is the only way to do this to shave overhead and cut costs and investment? To become more productive and shake Solow's aphorism, are IT departments focused on keeping the economists and accountants happy, while delivering a poor usability experience for customers and hurting the company where it matters most – the satisfaction of their users?

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

joewinchester@sys-con.com

ArrayListModel

A convenient way to use a simple collection

by Phil Herold

This article presents a data model based on a Collection implementation that can be used with Swing components JList and JComboBox. It also discusses a method to use these same concepts in constructing the user interface of an application.

Overview

Java Collections are indispensable for building any application, whether GUI or non-GUI. And the ArrayList class is a heavyweight in the java.util package. In a GUI application, the user often must choose items from a list, which can be presented in a variety of forms (drop down combo, list box, etc.). For example, the Java Swing components JList and JComboBox each have list data models – ListModel and ComboBoxListModel, respectively. Both components will react to changes in these models in keeping with the Model-View-Controller paradigm. However, neither of these models are based on a Collection, and therefore lack many of the convenient methods that the Collection interface provides. In addition, lists shown in an application are often populated from external sources (like a database) that return a Collection. Unfortunately, none of the Collection classes in Java broadcast changes to their contents, which is necessary for a user interface component to react to a true MVC data model.

The solution presented in this article to this dichotomy is simple – a subclass of the ArrayList Collection class that implements the ListModel Swing interface, the ArrayListModel class. A subclass, ArrayListComboBoxModel, which extends ArrayListModel implementing the ComboBoxModel model (which is itself a subclass of ListModel), is also presented.

Phil Herold is a Java architect with over 24 years experience in software engineering. He has been working with Java client technologies since 1996.

Phil.Herold@sas.com

Details

The ArrayListModel class implementation is very straightforward. You can probably already imagine what the methods look like. All ArrayList methods that modify the underlying collection are augmented. The superclass ArrayList method is invoked, and then the ArrayListModel publishes the underlying collection changes to all ListDataListeners that have been added (as part of satisfying the ListModel interface contract). Listing 1 shows the implementation of the add(), remove() and set() ArrayListModel methods.

Each method calls a corresponding fire method, which notifies any ListDataListeners what exactly in the collection has changed (see Listing 2).

Other methods in ArrayListModel work very similarly. For example, the clear() method calls the superclass method, and then fires a ListDataEvent signaling that all items from the collection (model) were removed. The ListDataEvent that is sent to each ListDataListener completely describes the collection elements (actually the indices) that have been added, removed, or modified. Because ArrayListModel implements the ListModel interface, the collection can be directly assigned as the data model of a JList. The ArrayListComboBoxModel class extends

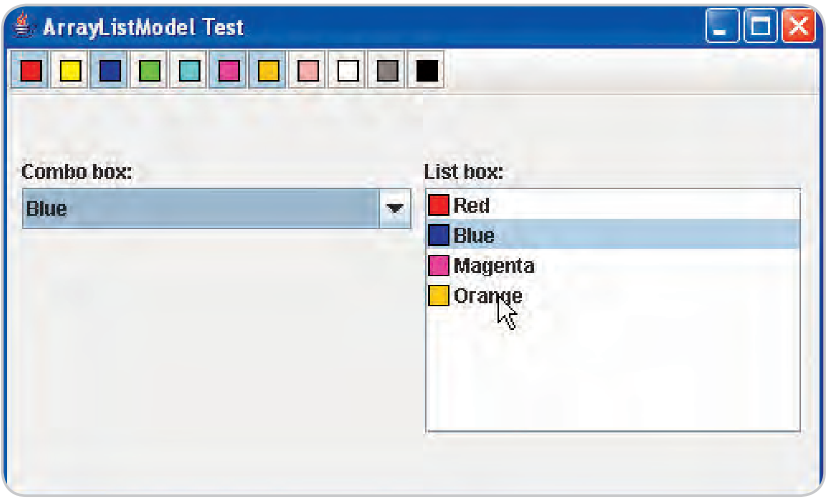


Figure 1

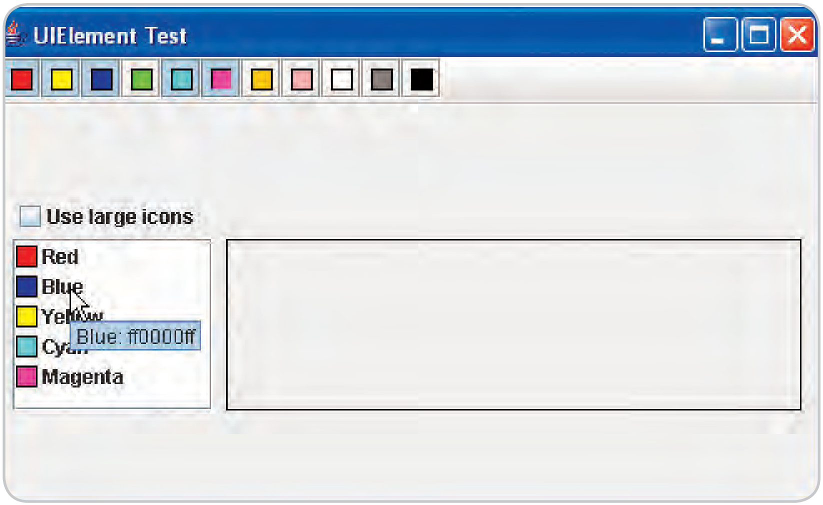


Figure 2

ArrayListModel implementing the two additional methods in the ComboBoxModel interface: getSelectedItem() and setSelectedItem().

To see these two classes in action, download the provided code and run the ArrayListModelTest class (see Figure 1). This is a Swing application consisting of a JToolBar, a JComboBox, and a JList. There are two data models used. One is an ArrayListModel that holds all of the data items, and the other is an empty ArrayListComboBoxModel used by both the JComboBox and JList components as their data models.

Figure 1 shows the contents of the combo and list boxes after the red, blue, magenta, and orange toolbar toggle buttons, respectively, have been pressed. Pressing a toggle button on the toolbar adds an item to the data model, while un-pressing it removes the corresponding data item from the model. The data model items are each an instance of a ColorItem, an internally defined class that consists of a name and an icon property (see Listing 3).

Listing 4 shows the method in the test program that constructs the JToolBar. The method has two arguments – an Iterator of ColorItem objects and the list to manipulate on toggle button press/un-press. The list parameter is actually the ArrayListComboBoxModel, but this method simply knows it as a generic List. Note that the ActionListener that is added to each toggle button is very trivial: it simply adds or removes the data item from the list as appropriate. The visual components (JList and JComboBox) attached to the single ArrayListComboBoxModel are automatically updated. A ListCellRenderer is used to render the ColorItem icon in the JList.

You can probably think of other uses for a Collection that announces changes to itself, and not necessarily in a GUI scenario. In that case, you might object to the fact that ArrayListModel and ArrayListComboBoxModel are connected to Java Swing, both in the interfaces they implement and the ListDataEvent/ListDataListener classes they consume. One possible solution to this would be to extend ArrayList as I have done but define your own event model and listener interface that would be more generic and not Swing biased. You could then use this class in a GUI application as I've shown by writing the necessary adapters to implement the ListModel and ComboBoxModel interfaces.

ListModels in Your UI

Have you ever considered all of the list-like elements in a GUI application? Many of the user interface constructs in your application can be thought of as lists of user interface elements that potentially have to be manipulated (items are added, ordered, and removed). Examples are menus, toolbars (their buttons), tabbed panes, internal frames of an MDI application, and potentially other custom components. And quite often you need to associate an icon, a tool tip, and some other visual component with each element.

Listing 5 shows an interface, UIElement, which defines these properties. Run the UIElementTest application that is provided as part of the code for this article. The application looks similar to ArrayListModelTest and has many of the same concepts (see Figures 2 and 3). The data model consists of UIElement objects that are obtained from a UIElement-Factory. (The factory returns instances of an inner class, our ColorItem object from before that now implements the UIElement interface.) The application knows nothing about

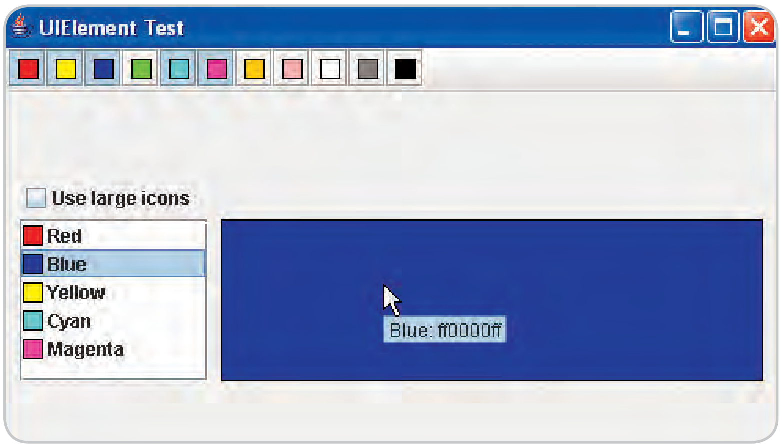


Figure 3

the underlying visuals, just that they implement the UIElement interface. The JList knows it can show icons from the elements and a tool tip for each element. The JList is populated by selecting toggle buttons on the toolbar as above (see Figure 2).

The toolbar is constructed in a method that is nearly identical to the one shown in Listing 4, except the ColorItem parameterized type is replaced by UIElement. When an element in the list is selected, the associated visual component is obtained from the UIElement in the list and shown in the line-bordered JPanel on the right (see Figure 3).

Note the use of tool tips in both the list and panel, which are provided by the getDescription() method of the UIEle-

RightOrder, Inc. Santa Clara, CA. Software Engineer
Perform research, design and develop computer software systems. Take a lead role in development team. Be a hands-on developer of proprietary server components & caching subsystems. Must have a B.S. in computer science or equivalent, 7 yrs + SW development of back-end servers & GUI applications in a multi-threaded environment and be fluent in Java, Visual C++, SQL, PL/SQL & SQL Tuning. Must have DBA level experience with Oracle 8/8i/9i/10g . M-F, 40 hr/wk. Send resume to: RightOrder, 404 Saratoga Ave. #200, Santa Clara, CA 95050, Attn: Ginger Burrell

RightOrder, Inc. Santa Clara, CA Senior Software Engineer
Analyze and create software requirements and design, develop and test software systems. Requires excellent design skills & hands-on development experience with a strong background in electronic commerce oriented systems, back-end servers & GUI applications. Requires proficiency at both high and low-level design with a demonstrated ability to plan and develop complex server systems. The candidate must have a BA in computer science or similar degree and preferred MBA in Technology Management. At least 7 years experience programming in OO languages and at least 5 years experience in Java development, Web logic 6.0 on Windows, XML, TogetherJ, JBuilder, InstallAnywhere, Winrunner, Startteam, and J2EE. M-F, 40 hr/wk. Send resume to: RightOrder, 404 Saratoga Ave. #200, Santa Clara, CA 95050, Attn: Ginger Burrell.

RightOrder

RightOrder, Inc.
404 Saratoga Avenue
Santa Clara, CA 95050
www.rightorder.com

ment data items. The ListCellRenderer for the JList reacts to the state of the *Use large icons* check box, calling the appropriate UIElement get*Icon() method. A simple change to my UIElementFactory could be coded to return a completely different implementation, but the rest of the application code would not have to be touched.

As you can see, we've implemented a simple list selection method that populates another part of the application upon selection. I would bet that most Swing developers have done something very similar in one or more applications they've worked on. And we've implemented a fairly rich user interface, with icons and tool tips using a single data model, with minimal code.

In my next article, I will present more of this list-based/UIElement framework for constructing an application's user interface. We'll start with an AbstractUIElement that will serve as the base class for our user interface data model. I'll also introduce the UIElementListModel class and its view counterpart, the UIElementListView interface (with an AbstractListView base class implementation).

Conclusion

In this article I have described a convenient way to use a simple collection (List) as the data model for JList and JComboBox Swing components. These components react to changes in the underlying collection, remaining synchronized with the data in the model. I have also introduced the beginning of a small framework that can assist in constructing a Swing application.

Listing 1

```
public void add(int index, E element) {
 super.add(index, e);
 fireIntervalAdded(index, index);
}

public Object remove(int index) {
 Object obj = super.remove(index);
 if (obj != null) {
  fireIntervalRemoved(index, index);
 }
 return obj;
}

public E set(int index, E element) {
 element = super.set(index, element);
 fireIntervalUpdated(index, index);
 return element;
}
```

Listing 2

```
protected void fireIntervalAdded(int firstIndex, int lastIn-
dex) {
 if (listDataListeners != null) {
ListDataEvent e = new ListDataEvent(this, ListDataEvent.
INTERVAL_ADDED, firstIndex, lastIndex);
  for (ListDataListener listener: listDataListeners) {
    listener.intervalAdded(e);
  }
}
}

protected void fireIntervalRemoved(int firstIndex, int last-
Index) {
 if (listDataListeners != null) {
ListDataEvent e = new ListDataEvent(this, ListDataEvent.
INTERVAL_REMOVED, firstIndex, lastIndex);
  for (ListDataListener listener: listDataListeners) {
    listener.intervalRemoved(e);
  }
}
}

protected void fireIntervalUpdated(int firstIndex, int last-
Index) {
 if (listDataListeners != null) {
ListDataEvent e = new ListDataEvent(this, ListDataEvent.
CONTENTS_CHANGED, firstIndex, lastIndex);
  for (ListDataListener listener: listDataListeners) {
    listener.contentsChanged(e);
  }
 }
}
```

Listing 3

```
private static class ColorItem {
 private String name;
 private ColorIcon icon;
 public ColorItem(String name, Color color) {
  this.name = name;
  icon = new ColorIcon(color);
 }
 public String toString() {
  return name;
 }
 public Icon getIcon() {
  return icon;
 }
}
```

Listing 4

```
private static JToolBar getToolBar(Iterator<ColorItem>
  colorItems,
final List<ColorItem> list) {
 JToolBar toolBar = new JToolBar();
 toolBar.setFloatable(false);
 while (colorItems.hasNext()) {
  final ColorItem item = colorItems.next();
  final JToggleButton toggle =
new ToggleButton(item.getIcon());
  toolBar.add(toggle);
  toggle.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    if (toggle.isSelected()) {
     list.add(item);
    } else {
     list.remove(item);
    }
   }
  });
 }
 return toolBar;
}
```

Listing 5

```
public interface UIElement {
 Object getItem();
 String getDescription();
 Icon   getSmallIcon();
 Icon   getLargeIcon();
 JComponent getComponent();
}
```

JAVA DEVELOPER'S JOURNAL Advertiser Index

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	4
Arcturus Technologies	www.arcturustech.com	703-822-4582	23
ceTe Software	www.dynamicpdf.com	800-631-5006	33
Common Controls	www.common-controls.com	+49 (0) 6151/13 6 31-0	45
ExtenTech	www.extentech.com/jdj	415-759-5292	39
Google	www.google.com/jdj	650-253-0000	41
IBM	www.ibm.com/middleware/tools		Cover IV
InetSoft	www.inetsoft.com/jdj	888-216-2353	21
Information Storage & Security Journal	www.issjournal.com	888-303-5282	53
InterSystems	www.intersystems.com/cache11p	617-621-0600	17
IT Solutions Guide	www.itsolutions.sys-con.com	888-303-5282	49
Jinfonet Software	www.jinfonet.com/jp10	301-838-5560	37
M7	www.m7.com/power	866-770-9770	25
MapInfo	www.mapinfo.com/sdk	800-268-3282	13
Microsoft	microsoft.com/connectedsystems		Cover II
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	47
ODTUG	www.odtug.com		43
Parasoft Corporation	www.parasoft.com/jdjmagazine	888-305-0041	7
Perforce Software	www.perforce.com	510-864-7400	9
ReportingEngines	www.reportingengines.com	888-884-8665	15
RightOrder, Inc.	www.rightorder.com	408-983-4000	51
Smart Data Processing, Inc.	www.weekendwithexperts.com	732-598-4027	59
Software FX	www.softwarefx.com	800-392-4278	Cover III
Synaptris	www.intelliview.com/jdj	866-99IVIEW	29
SYS-CON Website	www.sys-con.com	888-303-5282	61
WebAppCabaret	www.webappcabaret.com/jdj.jsp	866-256-7973	31
Windward Studios, Inc.	www.windwardreports.com	303-499-2544	11

Building a Toolbar from a Menu

by Mauro Micalizzi

For better usability, versatility, and user friendliness

As users of software applications, we commonly deal with the Graphical User Interface, which, in most of cases, contains the following main elements:

1. *A menubar*, which includes all the available commands and options (we'll call them functionalities)
2. *A toolbar*, which is a container for a subset of the most common and useful functionalities (typically a subset of "shortcuts" for the above mentioned commands and options)
3. *A working space* that is a kind of "container" panel, where the user can type, draw, or do anything related to the application

In this article I'll discuss the relationship between the menu and the toolbar from both sides: the application user and the application programmer.

In the role of developer of an application, I have to design a complete and efficient Graphical User Interface for the application function calls; a common choice is to provide the application with both a menubar and a toolbar.

First I'll start to design the menubar and then provide the GUI with a toolbar; however, I realize that part of the toolbar work was already done at the menubar design time. My question is: Can I save development time and work programming the GUI toolbar by using what it is already done for the menubar?

For example, suppose my text editor–like application supports clipboard operations. I have to insert the menu items "cut," "copy," and "paste" into the "Edit" menu, providing icon, accelerator key, and tooltips text for each of them. Then I have to associate what is commonly called "action," that is the code that implements the functionality (for "cut" it could be to delete the selected text and put it in the clipboard). Finally I have to manage the enabled or disabled status of each item (for example, disabling paste if clipboard is empty and enabling cut and copy if some text is selected). Of course, all these things are required for a complete and useful, user-friendly and good-looking menu.

Actually I would like to do something more, for instance, providing a toolbar for my application. First, I add three buttons with the same cut, copy, and paste icons (no text for them, according to the Look-and-Feel Design Guidelines), specifying the same tooltip text used for the menu items (note that I do exactly what I did before for the menubar). Then I link the buttons to the code that implements their functionalities (I repeat my actions again). At the end I write the code to disable

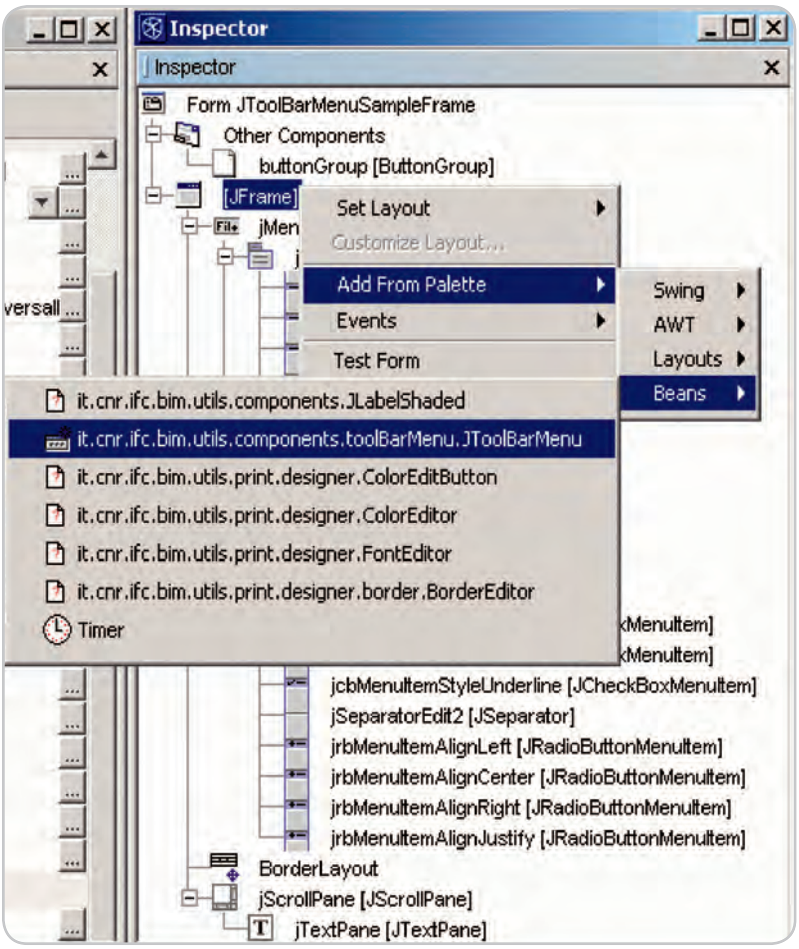

Inspector
Inspector
Form JToolBarMenuSampleFrame
Other Components
buttonGroup [ButtonGroup]
[JFrame]
jMen
Set Layout
Customize Layout...
Add From Palette
Events
Test Form
Swing
AWT
Layouts
Beans
it.cnr.ifc.bim.utils.components.JLabelShaded
it.cnr.ifc.bim.utils.components.toolBarMenu.JToolBarMenu
it.cnr.ifc.bim.utils.print.designer.ColorEditButton
it.cnr.ifc.bim.utils.print.designer.ColorEditor
it.cnr.ifc.bim.utils.print.designer.FontEditor
it.cnr.ifc.bim.utils.print.designer.border.BorderEditor
Timer
versall
xMenuItem]
xMenuItem]
jcbMenuItemStyleUnderline [JCheckBoxMenuItem]
jSeparatorEdit2 [JSeparator]
jrbMenuItemAlignLeft [JRadioButtonMenuItem]
jrbMenuItemAlignCenter [JRadioButtonMenuItem]
jrbMenuItemAlignRight [JRadioButtonMenuItem]
jrbMenuItemAlignJustify [JRadioButtonMenuItem]
BorderLayout
jScrollPane [JScrollPane]
jTextPane [JTextPane]

Figure 1 Adding JToolBarMenu object into my GUI by choosing from customized component list in NetBeans 3.6

Mauro Micalizzi, a researcher, has been involved in software developing for 25 years, and the Java language for seven. He is currently working on GUI, signal processing, and printing framework. Mauro has a degree in computer science.

mica@ifc.cnr.it

and enable the buttons, following the same criteria used for the menu items. In conclusion, I do the same things twice! And everybody knows that doing something twice is not a matter of laziness; this may easily introduce unexpected errors and cause undesired effects (such as a menu item disabled and the corresponding button enabled), especially when the application grows up and new commands or options are added.

I concluded that my efforts could be better employed in designing and coding a good menubar, leaving the development of the corresponding toolbar to some automatic procedure.

This kind of utility can be very useful for applications that have many functionalities and require frequent maintenance. To my knowledge, even sophisticated GUI design editors (like form editors included in the most common IDEs such as Eclipse or NetBeans) don't offer enough support for this aspect and they don't provide anything that keeps linked menu items and buttons.

Solution

The idea for solving this problem comes from the Action class (which is included in the java.swing package); this class offers an interface that extends the ActionListener and can be used in cases where the same functionality could be accessed by several controls (just like menu items and buttons).

The Action interface allows you to define, in a single place:

- The actionPerformed method defined by the ActionListener interface. This method is called when the user activates the control (for example, when he or she selects a menu item or presses a button). This method contains the code to implement the functionality.
- The text describing each functionality; these strings can be used to set the text in a menu item or to display the flyover text for a button or a menu item (tooltip text).
- The icon that depicts the functionality.
- The enabled/disabled state of the functionality.
- The accelerator key used to quickly access the functionality.
- The mnemonic key used to "navigate" by keyboard through a set of Actions.

Certain containers, including menus and toolbars, know how to manage an Action object. In detail they can achieve information from the Action to:

- Create the component appropriate for the container (a button for the tool bar, a menu item for the menu)
- Get the suitable information to render the container (texts, icons, enabled/disabled state)
- Notify the change of state
- Activate the functionality

Implementation and Use

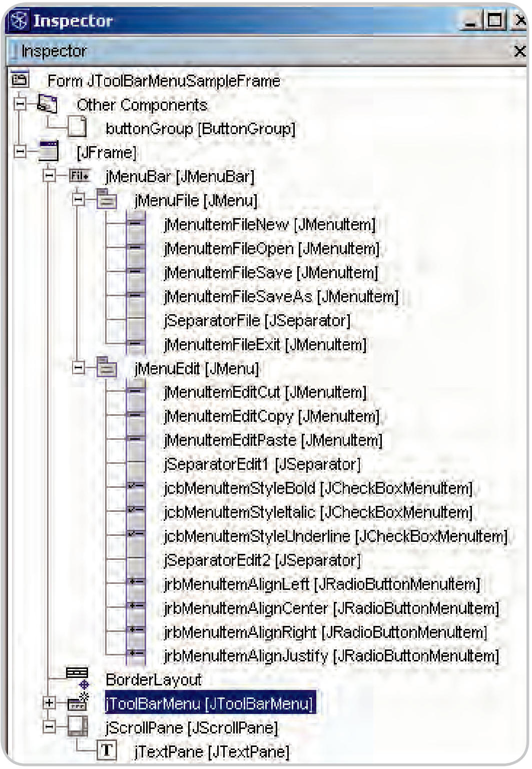
JToolBarMenu is an extension of the JToolBar from the javax.swing package (see sample use in Figures 1 and 2). It is initialized with a JMenuBar object, that is, the menubar the toolbar refers to.

There is no limitation on the way the menubar can be built: JToolBarMenu always shows a rational set of buttons derived from the menubar structure. Of course, if the menubar is built in a chaotic form, JToolBarMenu will be not so intuitive and tidy.

The set of buttons showed in the JToolBarMenu follows the same tree structure as the menubar. In fact, as root, there is a menubar, which has attached some menus (primary branches), and each of these branches contains next menu items (leaves) and/or some other submenu (other branches) and recursively going on.

It was developed as an algorithm to arrange, under all possible conditions and in the proper way, any input menubar. This algorithm consists of the following steps:

1. It scans all menus present in the menubar. For each menu that is not empty, a separator is added to the toolbar.
2. For each item present in the menu:
   - If the item is a separator, a separator is obviously added.
   - If the item is a submenu, the algorithm is recursively applied to the item again.
   - if the item is a JRadioButtonMenuItem (a menu item with exclusive selection), it's collected in a vector until there is no more contiguous JRadioButtonMenuItems. When the collection is ready, a set of JToggleButton, each one corresponding to the item in the vector, is added and preceded by a new separator.



```
Inspector
Inspector
Form JToolBarMenuSampleFrame
  Other Components
    buttonGroup [ButtonGroup]
  [JFrame]
    jMenuBar [JMenuBar]
      jMenuFile [JMenu]
        jMenuItemFileNew [JMenuItem]
        jMenuItemFileOpen [JMenuItem]
        jMenuItemFileSave [JMenuItem]
        jMenuItemFileSaveAs [JMenuItem]
        jSeparatorFile [JSeparator]
        jMenuItemFileExit [JMenuItem]
      jMenuEdit [JMenu]
        jMenuItemEditCut [JMenuItem]
        jMenuItemEditCopy [JMenuItem]
        jMenuItemEditPaste [JMenuItem]
        jSeparatorEdit1 [JSeparator]
        jcbMenuItemStyleBold [JCheckBoxMenuItem]
        jcbMenuItemStyleItalic [JCheckBoxMenuItem]
        jcbMenuItemStyleUnderline [JCheckBoxMenuItem]
        jSeparatorEdit2 [JSeparator]
        jrbMenuItemAlignLeft [JRadioButtonMenuItem]
        jrbMenuItemAlignCenter [JRadioButtonMenuItem]
        jrbMenuItemAlignRight [JRadioButtonMenuItem]
        jrbMenuItemAlignJustify [JRadioButtonMenuItem]
    BorderLayout
    jToolBarMenu [JToolBarMenu]
    jScrollPane [JScrollPane]
      jTextPane [JTextPane]
```

Figure 2 A JToolBarMenu has been added to menu bar of sample application used in this article

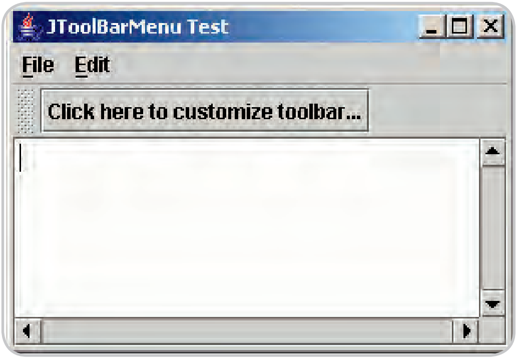


Figure 3 Sample application with no configuration file. A big button appears to invite user to customize his toolbar

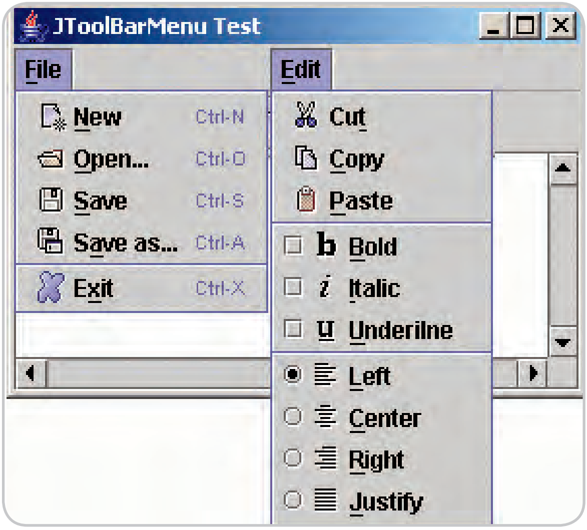


Figure 4 The 2 menus in menubar of sample application.

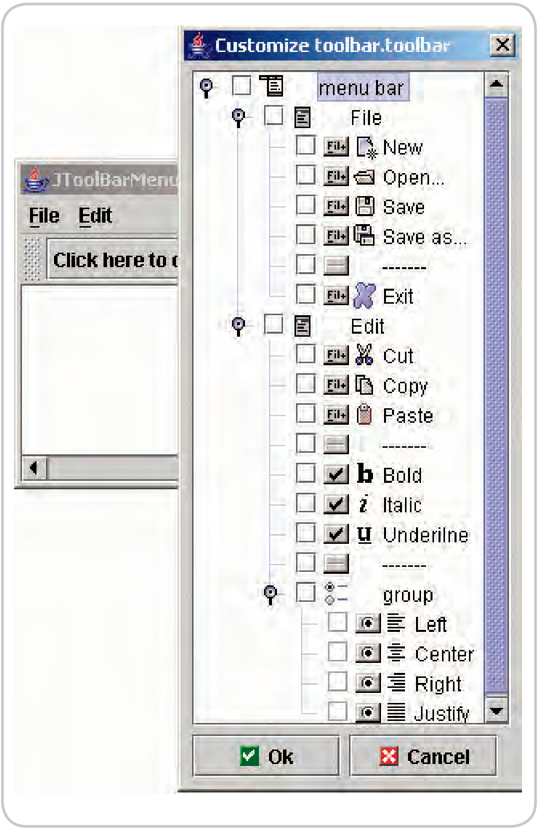


Figure 5 After clicking on "Customize" button, a dialog displaying the menubar structure in a JTree will appear. Both icons are visible: item type icon and specific icon too

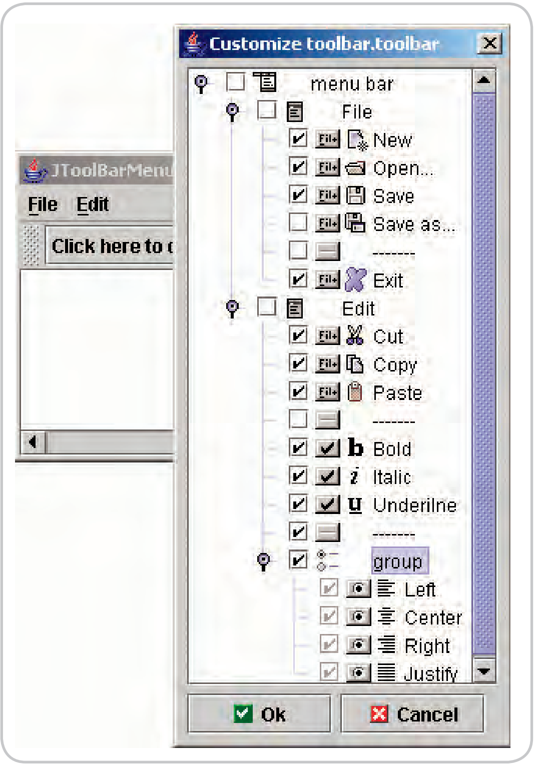


Figure 6 Some items are edited (i.e. selected). Note selection of alignment group and its disabled items

- If the item is a JCheckBoxMenuItem (menu item without an exclusive selection), a JToggleButton is added.
- Otherwise the item is simply a JMenuItem, thus a JButton is added.

This mechanism ensures consistency with the menubar structure and an attractive display because the role of the separators, which are always put in a right and rational position, result in a user-friendly layout.

JToolBarMenu can be initialized with any other component you would like to add to the toolbar. To do this, a component array can be passed to the JToolBarMenu and added at the beginning, each one divided by a separator.

JToolBarMenu is associated with a configuration file (a simple properties file), which is used to store the sequence of preferred menu items that have to be added to the toolbar for subsequent rebuilding.

Within the configuration file, each menu item is identified by a property; if its value is empty, the item will not be present in the toolbar, otherwise, it will be added.

If the configuration file is empty (see Figure 3) or absent, the toolbar will contain only one button, inviting the user to customize the toolbar. In this case the user should press the "Customize" button to select the preferred functionalities he or she would like to have in the toolbar; a dialog box will appear displaying a JTree that represents the structure of the menubar given as input (see Figures 4 and 5). Each tree element (except root, which identifies the menubar) is editable, meaning that it has a checkbox to specify whether that element should be included in the toolbar (see Figure 6).

As described previously, the grouping of JRadioButtonMenuItems is detected too. A group node is created for them and added to the tree; the JRadioButtonMenuItems appear as group node leaves.

The JRadioButtonMenuItems that are grouped together are initially disabled. They can be selected (or unselected) all together by clicking on their parent group node. The general rule is: by clicking on a leaf element this will be toggled, selected/unselected; by clicking on a node, all its descendant items will be selected/unselected.

For a better view, the tree elements show a double icon: the first one identifies the type of the item (one of the following: menu bar, menu or submenu, radio button, check box, menu item), the second one is the menu item icon provided by the menu designer.

After the first call, customization of the dialog box can be accessed by pressing the special button that's last in the set of buttons in the toolbar (see Figures 7–9). This action will open a popup menu that shows the option "Customize...". It is advisable to provide the JToolBarMenu application with a predefined configuration file so the user doesn't have to customize the menu on the first run. The most important part of JToolBarMenu is the construction of the Action objects, which keeps the menu items and toolbar buttons linked together.

If a menu item was already built with an action, a new JButton is created by directly passing the menu item action (using the getAction method). Otherwise a new and complete Action object must be constructed, getting the following information from the menu item:

- Action name, from the menu item text.
- Action command key, from the menu item action command.
- Action Mnemonic key, from the menu item mnemonic key.
- Action icon, from the menu icon, if any, otherwise an "empty" icon (that is, a 100% transparent icon) will be set. Note that frequently used menu items should always have an icon, if not, an empty button will be created that will be anonymous and hardly distinguishable among others, especially if there are some empty buttons.
- Action short description, from the menu item tooltip text.
- Action enable state, from the menu item enable state.

ActionListener objects, if any, are also taken from a menu item. This is important for two reasons: to notify a menu item when a button is pressed so that its functionality can be activated, and to set the selected state when a button is pressed or a menu item is selected. A Change-Listener object is created to reciprocally set the enable state for both the components.

JToolBarMenu is sensitive to the resizing of the frame; when the frame width is reduced, not enough space could be available for displaying all buttons in the toolbar. The solution is to hide the minimum number of buttons that precede the "customize" button so that it will be always visible. An ellipsis ("…") is replaced instead of the hidden buttons. This strategy is also used when, after customizing the toolbar, the user has selected too many buttons to fit the assigned width.

Limitations

Currently there is only one known limitation: the algorithm that's used to represent the menu bar in a JTree may not correctly interpret how to group JRadioButtonMenuItems depending on the menu design.

For example, consider the Alignment menu composed by left, center, right, and justify in this order:

1. If you put a menu separator between center and right, the algorithm will create two groups instead of one, allowing operations like pressing left and right at the same time;
2. If you don't insert a separator (or a submenu or something that breaks the continuity in the menu) before left and after justify, the algorithm will extend the grouping to others that will eventually precede or follow JRadioButtonMenuItems and will produce malfunctions. If your menubar contains this kind of menu, your design is probably not in agreement with common GUI guidelines.

Conclusion

This article presented a simple, powerful, and useful tool for an integrated menubar and toolbar GUI design and implementation; this tool matches the needs of both the programmer (because he doesn't have to duplicate code) and the final user (because she can arrange the toolbar, adding or deleting buttons according to her needs). Using it yields some benefits in terms of better usability, versatility, and user friendliness. No relevant overhead has been encountered using JToolBarMenu.

Programmers can insert this component among the customized components in their IDE and use it whenever they want.

Software code was tested within our needs and developed using the Java 2 Platform, Standard Edition, v 1.4.2. The source code is available at http://jdj.sys-con.com.

References

- Sun Microsystems Inc. (2003). Java 2 Platform, Standard Edition, v 1.4.2, API Specification: http://java.sun.com/j2se/1.4.2/docs/api/
- Sun Microsystems Inc. "Java look and feel Graphics Repository": http://java.sun.com/developer/techDocs/hi/repository/
- Sun Microsystems Inc (1999). "The Java Look and Feel Design Guidelines": http://java.sun.com/products/jlf/ed1/dg/index.htm
- Walrath, K.; Campione, M.; Huml, A.; and Zakhour, S. (2004). *The JFC Swing Tutorial: A Guide to Constructing GUIs, Second Edition*. Addison Wesley Professional: http://java.sun.com/docs/books/tutorial/uiswing/components/

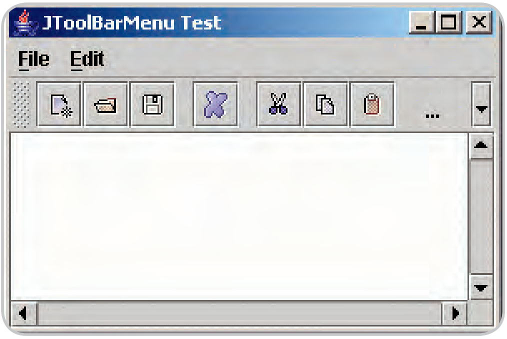


Figure 7 After customizing toolbar, selected buttons that fit frame width are displayed. Ellpsis substitute hidden buttons

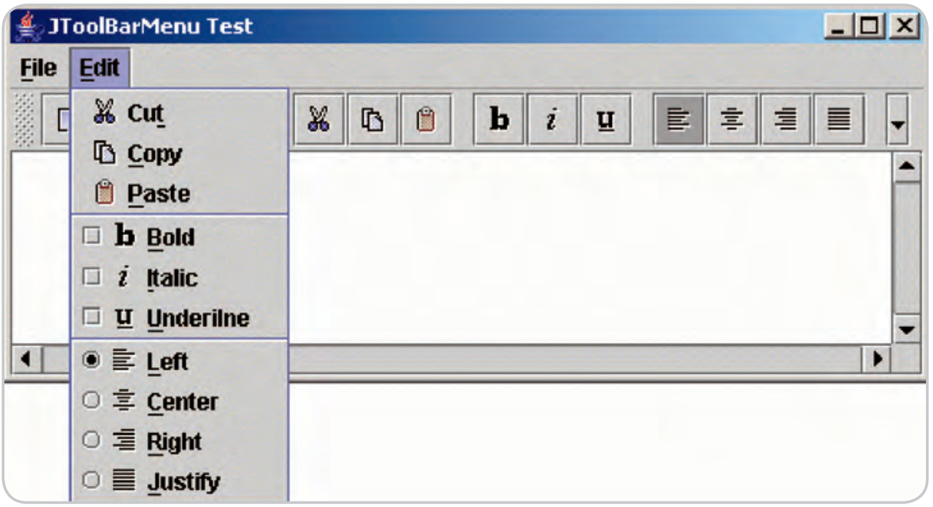


Figure 8 Now frame has been resized to display all selected buttons. Note that left alignment is selected both in menu item and button

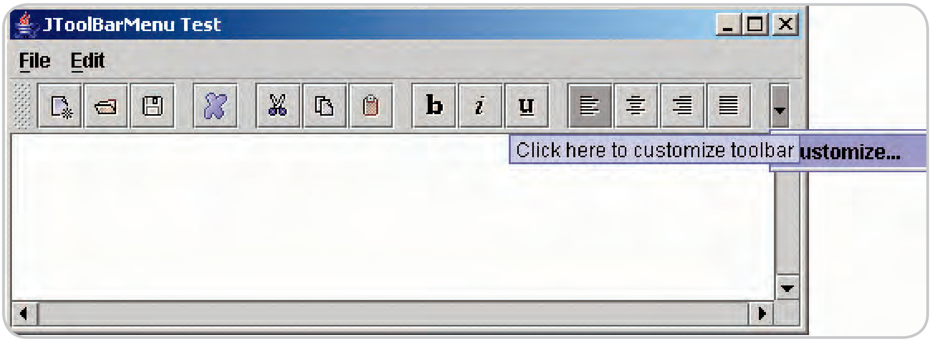


Figure 9 Click on "Customize" button to change your toolbar layout again.

Letters to the Editors

C# versus Java

["C#: Is the Party Over?" by Calvin Austin, Vol. 10, issue 8]

I attribute the lukewarm success of C# to two major factors:

1. It is not a cross-platform solution. (Mono notwithstanding; to be really cross-platform, MS would have to be producing versions of the runtime for both Linux and Mac, and keeping those up-to-date.)
2. There is no easy way to use it to create rich clients. This is just stupidity on MS's part, since the CLR has a security sandbox similar to Java's. In fact, the very first version of .NET would run applets (they called them "Windows Forms" applications) without difficulty. But then they locked down the security settings, making it next to impossible to deploy rich clients in heterogeneous networks.

The result is that .NET is just a substitute for C++, not a substitute for Java. And given the huge C++ code base people working on Windows software are starting with, it makes no sense for them to switch.

–Joshua Smith

Who cares about cross-platform? It has nothing to do with how great C# really is. Much better than Java, hands down.

I wouldn't refer to applets as "rich." They were cool back in the '90s, but are useless today. Why would MS duplicate an old technology like applets? DHTML, Flash, and even AJAX are more than enough "richness," and all you need.

C# is absolutely a replacement for Java, and a good one at that. The adoption rate of C# is steadily increasing as developers discover the better features and syntax of C#. And don't forget that VB is still a hugely popular language and developers don't switch languages over night.

-Mark Stewart

In the area my company serves – 3D visualization – the other client technologies you mention don't come close to providing the richness required. Have a look at this: www.kaon.com/3DCatalog.html.

That is a Java 1.1 applet. It works on all browsers, all operating systems.

As for the question, "who cares about cross-platform," the answer is pretty much every software architect, CTO, and CIO. The issue isn't being able to run your app on multiple platforms (which is nice to have, but often not a requirement). Rather, it is having the freedom to change platforms when you need to. You might start out deployed on Windows Servers and then, for cost reasons, decide to switch to Linux (or vice-versa, depending on who's TCO numbers you believe). Being able to do that without having to recode your applications is crucial.

–Joshua Smith

Calvin Austin replies:

If I had to name my #1 reason why C# has failed to live up to the expectations for it back in 2000 is it was created for all the wrong reasons.

Ruby, PHP, and Python, in my opinion, were all designed to solve a developer's problem and not a company's problem. C# came out of a time when there were concerns with Visual Basic migration, C++ defections, and the growth of open source frameworks.

As developers we are consistently reminded that C# and .NET are to be treated as one and the same. Why can't C# stand on its own merits for once, or is it a language that is only useful with .NET?

Working with Open Source

["J2SE and Open Source – Living Together in Perfect Harmony" by Joe Winchester, Vol. 10, issue 8]

I agree with James Gosling's concern – there will always be holes in any specifications.

If there is only one single implementation, such undocumented behaviors can still be utilized after some preliminary investigations; but if more than one implementation is used by the majority of end users, these undocumented portions could have different behavior.

It may sound as if we shouldn't have used any unspecified "features" in a specification, but they are often the only workaround for certain tasks to date, due to the deficiency in the spec. Only by having a single major implementation can we do away with these problems easier, if at all.

–Alex

Joe Winchester replies:

But there are already different versions of the JRE created by different vendors for their particular hardware platforms. Sun does not create the Z-OS or iSeries JRE. What keeps these together is the TCK and given that Java now works cross-platform, all that is different is that there can be an open source version of J2SE. James' comments were more about the fact that open source J2SE would create forking, and this is something that I agree with that we don't want. What I don't fully understand is why forking is more likely with open source that occurred before with different JREs being created commercially.

Solving Small Business Problems

["Small Business Solutions" by Yakov Fain, Vol. 10, issue 7]

Seems like a few assumptions in Yakov Fain's article were a little off base.

- Web services: I prefer NetBeans, since from my experience it's a lot farther ahead than Eclipse nowadays.
- Database: MySQL is free, quick, and easy for this type of thing. Hibernate is also a good choice.

- Front end: Check out DWR for AJAX (www.getahead.ltd.uk/dwr/). Also, "Swing is not there yet; SWT looks better." Have you actually looked at either of them? The only thing that SWT does well is Eclipse. Take a look at the SWT mailing lists or the bug lists, and you'll notice there are all kinds of problems. Swing is mature, stable, and ready to go –- use it. Also, why are you asking JDJ readers about .NET? I'm sure many of them have used it but really…

Using Tomcat, Hibernate, MySQL, a free operating system, and OS JMS products you should be able to do this project with a minimal budget.

–Branden

Dude! Get a copy of QuickBooks and the QB Point-of-Sale system and be done with it! Then, if you need to integrate applications, there's a great SDK that lets you do this.

If you need to get a programming fix once you have QB in place, see my Web site at http://qbopen.com or go to http://developer.intuit.com.

Now, I didn't write QuickBooks, and don't really wish to have to defend it, but if you have to sell gas and need an immediate off-the-shelf solution in your price range, then go to Best Buy and get the QBPOS bundle, which also comes with a Dell computer.

–Bill McCuistion

Yakov Fain replies:

I like Bill's comment!

Most likely QuickBooks can solve most of my needs, but…

1. I don't want to be the same as any other gas station owner on the block.
2. I don't want to cut my ties with the Java world. If my gas business won't become profitable in a year, I'll sell it to a QuickBooks user and return to coding in Java, and my skills will be up to date.
3. I want to understand the world of the open source software and my next column [in this issue] will have an interview with one of my famous customers who happens to be a best selling author in this field.
4. By having this column, we can discuss real-world problems that small guys face as opposed to talking all the time from the perspective of large-scale enterprises.

A Slick Workaround

["FrameResizer" by Phil Herold, Vol. 10, issue 8]

Brilliant!!! I've been writing fancy Swing apps for a long time and never even considered that there was a way around the gray-screen-as-you-resize problem. Very slick, very thorough. An icon for this was meticulously created by Jonathan Simon here: http://today.java.net/pub/a/today/2005/06/07/pixelpushing.html.

—Michael Bushe

Java and Tag-Based Languages

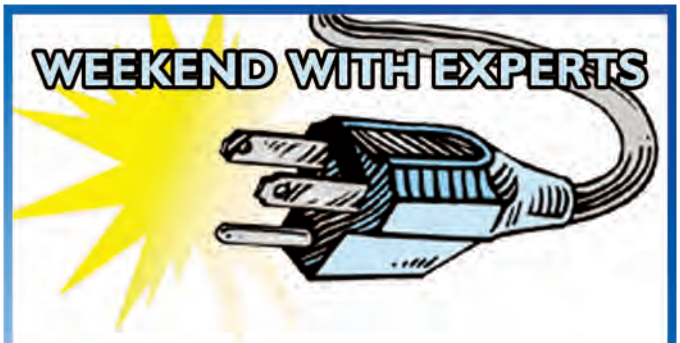
["Back to Two Tiers and Plain JSP" by Brian Russell, Vol. 10, issue 7]

Good article. It's worth noting that there is another free solution out there that marries the best of Java and the simplicity of tag-based languages. While ColdFusion CF is Java under the covers, New Atlanta makes a competing product (Blue Dragon) that follows the CF language model (tag-based) and for which the low-end version is free. That version has about 80 plus percent of the features and certainly all that is needed if what is in view is moving from a site that is HTML to dynamic content. It is available in both Windows and Linux incarnations and is quite powerful. The major cost of a Web site over time is maintenance and CF is a very "friendly" alternative to JSP. It can be readily understood by those coming primarily from HTML as it uses the same tag paradigm. Like ColdFusion, it uses Java as the core engine so it can easily be extended to incorporate in Java anything needed that doesn't come natively. It's worth checking out if you are looking at low cost (free is hard to beat). I've ported a lot of code between commercial CF, which we use at work, and Blue Dragon, which I use on my own sites with complete portability. The Macromedia folks are always upping the ante by adding new things but CF has been around for so long that the core things used to build dynamic sites are commodity features shared by both products and very nearly identical in function. About the only thing required is adjustments in the SQL if you use a different underlying DB, but that's true everywhere.

–Don Babcock

One thing you failed to mention when using JSP is the need for an application server. PHP on the other hand uses just the HTTP Server on a system with PHP installed. I have been a J2EE architect and developer for many years and, recently, when trying to get something that's professional looking up and running, it's easier to forget about the app server and use something like PHP. Just my two cents.

WEEKEND WITH EXPERTS



Get a Java injection on the go!

Spend a weekend and have a lunch with experts in an intimate learning environment in the city near you.

No salesmen allowed. Join our technical discussions and hands-on workshops.

The next Roadshow is in Philadelphia on December 10 and 11.

www.weekendwithexperts.com

2005 JCP EC Elections Are Under Way

by Onno Kluyt

Ratification ballot starts voting marathon

It's that time of the year again when the JCP is in election mode and an update about it is more than timely. At writing time, the ratification ballot was just posted (September 27) and this year's nominees are BEA, SAP, and SAS for the Java SE/EE Executive Committee (EC) and Nokia, IBM, and Philips for the Java ME EC (https://www.jcpelection2005.org/jcp/ratification_ballot).

Before I share with you the nominees' qualifications, just a reminder that the JCP Elections process starts with member nominations by the JCP Program Management Office (PMO) for the vacant Ratified Seats of the two JCP ECs. Nominations are made with due regard for balanced community and regional representation. For more details, here are a few pointers to useful election information resources. You'll find a 2005 JCP Elections primer on jcp.org at http://jcp.org/en/whatsnew/elections and an overview at PriceWaterhouse Coopers, the annual online central of the JCP elections, at http://www.jcpelection2005.org/jcp/overview.

And the ratified ballot nominees for JCP SE/EE EC are…!

BEA Systems, Inc.

BEA's elections qualifications card mentions the company's leadership in enterprise infrastructure software, its contribution to providing standards-based platforms to accelerate the secure flow of information and services. BEA product lines – WebLogic, Tuxedo, JRockit, and the new AquaLogic family of Service Infrastructure – help customers reduce IT complexity and successfully deploy service-oriented architectures to improve business agility and efficiency.

BEA WebLogic Server, a leader in the application server market segment, has been a supporter of the Java EE Platform and its latest release is certified for J2EE 1.4. The WebLogic family of Java products are providers of Java technologies and a continued source of innovation within the Java community and have played a significant role in the success of Java in the enterprise market.

BEA has served on the Executive Committee for J2SE/J2EE since its inception and continues to be a strong supporter of Java and Java standardization. BEA has been a diligent voter in progressing new JSRs through the process and a significant contributor to improvements in both the JSPA and the JCP process document. The company also supported numerous JCP activities at JavaOne and other venues. BEA also holds leadership positions in other standards and open source organizations, which enable BEA to be even more effective on the JCP EC.

BEA has successfully completed three JSRs as Spec Lead: Java Rules Engine API (JSR 94), Streaming API for XML (JSR 173), and Web Services Metadata for the Java Platform (JSR 181). They participate in over 20 JSR Expert Groups, including JAX-WS (JSR 224), EJB3 (JSR 220), JDBC 4.0 (JSR 221), Content Repository for Java Technology (JSR 170), and Java EE5 (JSR 244).

SAP

With 12 million users and 96,400 installations, SAP is a large inter-enterprise software company and the third-largest independent software supplier overall. With this experience, SAP is positioned to guide Java to become the foundation for a next-generation service composition platform that can rapidly deliver applications that support those business processes that are most critical for a company.

SAP has been a member of the SE/EE Executive Committee since 2002 and has actively worked with other leaders in the Java industry to guide the future success of the Java platform. SAP has proposed ways to make Java simpler and easier to use by reducing the number of technical choices that Java developers face. SAP has been most active in JSRs related to the Web services space such as JAX-WS 2.0, Java Business Integration (JBI), and the SAP-led API for Web Services Policies (JSR 265). Also, SAP has facilitated the dialogue with communities outside of the JCP such as the Eclipse Foundation.

SAS

SAS provides a new generation of business intelligence software and services that create enterprise intelligence. SAS also develops solutions that are used at about 40,000 sites – including 96 of the top 100 companies on the FORTUNE Global 500 – to develop more profitable relationships with customers and suppliers; to enable better, more accurate, and informed decisions; and to drive organizations forward. SAS integrates data warehousing, analytics, and traditional BI applications to create intelligence from massive amounts of data.

Java technology is a critical underpinning of the SAS Enterprise Intelligence Platform. Because its customers desire to run SAS software on a wide array of platforms and operating systems, Java is a natural fit for SAS. Java technology provides SAS with a solid foundation upon which to build multi-platform solutions that meet key customer requirements. The company delivers a broad range of Java-based software, from general-purpose analytical and reporting software to targeted, industry-specific, vertical-market business intelligence applications. This software is delivered to both desktop clients and the mid-tier.

Due to the multi-platform nature of SAS software and their use of Java technology, Java standards and interoperability are critical components to successful development and deployment of their software offerings. SAS believes that encouraging and supporting the development of such standards is an important part of working with the Java community. As a result, SAS has been a Java licensee partner since 1997 and has been actively involved in the Java Community Process for much of that time.

Onno Kluyt is director of the JCP Program at Sun Microsystems and Chair of the JCP.

onno@jcp.org

SAS has participated in approximately 25 Java Specification Requests spanning both J2SE and J2EE technology, including such diverse JSRs as the J2SE umbrella JSRs (JSRs 59, 176, and 270), Java OLAP Interface (JSR 69), Data Mining (JSR 73), JSP 2.0 and Servlet 2.4 (JSRs 152 and 154), and the Portlet specification (JSR 168).

SAS believes that it brings a unique, end-user focus to the JCP Executive Committee. As a Java platform consumer rather than a platform provider, the company has an understanding of the challenges and opportunities related to deploying end-user Java applications in large-scale, enterprise environments. By participating on the JCP Executive Committee, SAS intends to collaborate with the broader Java community in helping to strengthen the Java platform and effectively drive Java standards forward.

And the ratified ballot nominees for JCP ME EC are…

IBM

IBM has been a contributor to the Java Community since 1996, bringing its breadth of resources and deep understanding of Java technologies to bear on almost every aspect of the platform's evolution. Since 1998, IBM has been a key participant and leader on numerous JSRs and has been a contributing member on both Executive Committees since their inception in 2000. IBM is heavily investing in the success of Java technology – providing Java SE ports for more than a dozen environments, delivering over 300 Java-compatible products to market, and creating the Java EE implementation in the form of WebSphere Application Server. In the Java ME space, IBM provides VM ports to more than 20 environments together with support for a broad range of configurations, profiles, and JSRs. IBM's representatives, whether leading or participating in Expert Groups and Executive Committees, will continue to use their expertise and technical resources to make contributions for the betterment of the Java Community and to help guide the Java platform's evolution and development.

IBM is Spec Lead on three J2ME JSRs, and participates in 17 J2ME JSR Expert Groups.

Nokia

Nokia has made a long-term commitment to Java technology. The company had already been participating in the development of the Java Community Process before the establishment of the current process and the Executive Committees in 2000. Since then Nokia has been actively cooperating with other EC companies to further the JCP process. Nokia's goal has been, and is, to help the JCP become more of an open standards organization; at the same time, however, maintaining/securing its effectiveness and increasing Java technology competitiveness and acceptability to better serve the business of the members. For example, Nokia has made proposals (and used similar practices itself) to the EC, which will increase the predictability of the licensing terms of the Java specifications, and add acceptability and accessibility of the technology and create new business opportunities around Java technology.

In addition to the EC work, Nokia is also a strong contributor to the practical JSR EGs in bringing new functionality to Java technology to better enable the development of new applications and services.

Nokia is currently Specification Lead (or co-lead) in more than 15 JSRs in the ME area and actively contributing to more than 15 other JSRs. Nokia-led JSRs encompass a wide area of necessary technologies for mobile devices, such as 3D and 2D graphics (JSR 184 Mobile 3G API, JSR 226 Mobile 2D SVG API), multimedia capabilities (JSR 135 Mobile Media API, JSR 234 Mobile Multimedia Supplements), UI customization (JSR 258), sensors (JSR 256) and near field communication (JSR 257), and Mobile Service Broadcasting (JSR 272).

Nokia has also been working hard to standardize and develop a more advanced Java platform for future mobile devices, which will enable a service-oriented, modular, scalable, and extendable architecture based on CDC, enhanced with operational management (JSR 232 Mobile Operational Management) that allows remote management and operation of the devices.

Nokia has also felt a responsibility to increase Java technology competitiveness by making an effort to defragment the Java technology marketplace. This work is successfully run in close cooperation with main mobile device suppliers, mobile operators, and other interest groups in the Nokia (co-led with Vodafone) Mobile Service Architecture JSRs – JSR 248 and JSR 249. There the parties will create a mobile service architecture and platform definition for the high-volume wireless handsets to enable more Java technology–based applications and services business to the developers and service providers.

Nokia's latest JSR (co-led with Sun Microsystems) is working to harmonize the XML parsing API offering in the ME and allow for a high-level service connection to Web services (JSR 279 Service Connection API for ME, JSR 280 XML API for ME).

Nokia continues to be a valuable member of the JCP EC and has shown an excellent track record in the active EC work, in the practical development of the Java technology in the EGs, and in their ability to work together with other JCP members for the good of the Java technology.

Royal Philips Electronics

Royal Philips Electronics is one of the biggest electronics companies and Europe's largest. The main divisions in the company are consumer products, lighting, semiconductors, and medical systems. Philips became interested in the JCP as it became apparent that Java technology would be built into a number of the products made by Philips.

As a diverse company, Philips brings a wider perspective to the ME EC than the mobile phone–centric approach of many EC members. Much of the Philips interest in Java has been in the area of entertainment, initially digital television receivers and more recently Blu-ray disc. Philips also has interests in Java in mobile phones, in smart cards, and in embedded consumer devices, such as intelligent remote controls.

Philips became a member of the JCP program in 2000 and a member of the JCP EC for J2ME when the JCP was first formalized, also in 2000. Philips is a member of the JCP Experts Groups including those for JSR 62 Personal Profile Specification, JSR 68 J2ME Platform Specification, JSR 118 Mobile Information Device Profile 2.0, JSR 121 Application Isolation API Specification, JSR 129 Personal Basis Profile Specification, JSR 135 Mobile Media API, JSR 179 Location API for J2ME, JSR 216 Personal Profile 1.1, JSR 217 Personal Basis Profile 1.1, JSR 218 Connected Device Configuration (CDC) 1.1, JSR 219 Foundation Profile 1.1, JSR 242 Digital Set Top Box Profile – "On Ramp to OCAP," JSR 257 Contactless Communication API, and JSR 272 Mobile Broadcast Service API for Handheld Terminals.

I'll be back with the final results of the 2005 JCP Elections in next month's column. Meanwhile, if you are a JCP member, make sure you mark the following important election dates in your calendar: Ratified Voting, October 4–17; Open Nominations, October 18–31; and Elections Ballot, November 1–14. Don't forget to vote!

Software FX
Bringing Your Data, Business & Strategy Into Focus.

New! version 6.2
Now Includes Maps!

All New Flavors... Same Great Taste of Java!

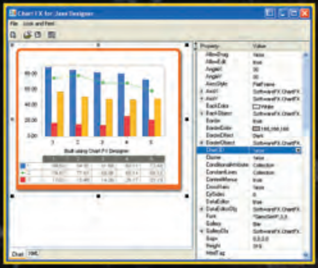


Chart FX for Java Designer

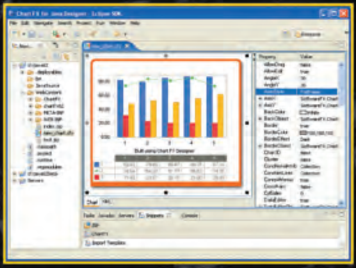


Chart FX for Java - Eclipse

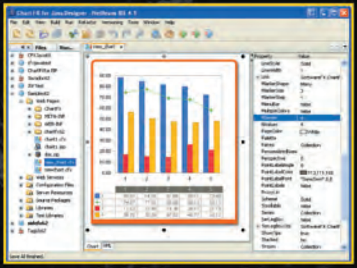


Chart FX for Java - NetBeans

Chart FX for Java 6.2

Now offers seamless integration within Eclipse and NetBeans, Chart FX for Java 6.2 brings New features include Smart & dynamic axis labeling, conditional marker and axis attributes, extended URL linking, multiple chart panes and robust annotation objects. One of the most dynamic new features is the inclusion of the Chart FX Maps Extension as a permanent fixture within Chart FX for Java. ➤

Chart FX

US: (800) 392-4278 • UK: +44 (0) 8700 272 200 • Check our website for a Reseller near you! **www.softwarefx.com**

IBM RATIONAL PRESENTS

YOU ★VS★ THE INCREDIBLE SHRINKING DEADLINE

MAIN ATTRACTIONS

KNOCKOUT INNOVATION

INTEGRATED DEVELOPMENT TOOLS SUPPORTING ASSET-BASED DEVELOPMENT ★ BASED ON ECLIPSE™ ★ RUNS ACROSS MULTIPLE PLATFORMS INCLUDING LINUX®

POWER TO CREATE BETTER SOFTWARE FASTER

IBM MIDDLEWARE. POWERFUL. PROVEN. FIGHT BACK AT WWW.IBM.COM/MIDDLEWARE/TOOLS

AND DOWNLOAD TRIAL VERSIONS OF RATIONAL SOFTWARE MODELER & RATIONAL SOFTWARE ARCHITECT