SYS-CON

JAVA GoF BEHAVIORAL DESIGN PATTERNS PAGE 44

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

JDJ

JDJ.SYS-CON.COM VOL.10 ISSUE:11

IN THIS ISSUE...

DEVELOPING INTELLIGENT WEB APPLICATIONS WITH AJAX

PART 2

A peek into modern technologies for browser-based applications

RETAILERS PLEASE DISPLAY UNTIL DECEMBER 31, 2005

$5.99US $6.99CAN

PLUS...

Innovations by InterSystems

Rapid development with robust objects

Lightning speed with a multidimensional engine

Easy database administration

Massive scalability on minimal hardware

Multidimensional Database Enables Rapid Development.

Caché is the first multidimensional database for transaction processing and real-time analytics. Its post-relational technology combines robust objects and robust SQL, thus eliminating object-relational mapping. It delivers massive scalability on minimal hardware, 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/Cache12P**

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

Disturbing the Lexicon

Jeremy Geelan

If successful trade expos are a good barometer of the market place (and they are), then things are going very well indeed with the homegrown category of apps named by Macromedia (soon to become Adobe), namely RIAs.

Whether "rich Internet applications" are going to be dominated by Adobe in the future, just as Macromedia has dominated them in the past is open to question. There are many pretenders to the throne, not least those in the AJAX camp.

One of the first hurdles for their coinage, "RIAs," to jump though is that of passing into common currency. Developers are definitely using the term, but they don't seem to be taking to it as fast as they took to, say, "Web services" or "podcasts" or "social software." Which started me thinking: To what extent are the winners and losers in the game of personal and enterprise software those who first and foremost win The Name Game?

It's nothing new, of course. Douglas Coupland wrote, in *Microserfs*, that "everybody's trying to find a word that expresses more bigness than the mere word 'supermodel'" and instanced attempts like 'hypermodel,' 'gigamodel,' and 'megamodel' – none of which, so far as I am aware, has caught on.

That was originally published in WIRED 2.01, though, back in the mid-'90s. Ten years on, I am happy to report that humankind is no longer suffering from any such inability to come up with a word bigger than "supermodel." Of course it's the technology sector that has come to our rescue, just when it was maybe looking like this word-poverty was reflecting some kind of collective "inability to deal with the crushing weight of history we've created for ourselves as a species" (Coupland's – marvelous – words again).

The rescue-words I have in mind? '"Terabyte" and 'gigabyte," for starters. PCs containing a terabyte or more of storage space have recently become possible using combinations of high-capacity mass-market hard drives. As of about mid-2005, common commercial hard drives exceeded 400 gigabytes in size, so storage capacity totaling a terabyte or more can be reached using as few as two or three hard disks, at a street cost of as little as $450, down from over $1000 in 2003.

I would go so far as to say that any company who dares to disturb the lexicon (for want of any better way of describing it) sufficiently early in the product release cycle is 50 times as likely to catch on, whether it be with VCs or the general public. It's why the iPod Nano got off to such a roaring start in spite of its subsequent, erm, screen difficulties; and why Nick Kamen's invention, codenamed "Ginger" or "IT" (neither of them exactly dictionary-busters), was doomed from the moment it was unveiled as the even less lexicon-disturbing "Segway Human Transporter."

This is not overly scientific, I hear you say. Yet I fear you may immediately remember what I'm saying when next any of the following terms trip off your tongue: "Java SE 6," "Java EE 5," and "Java ME."

Josephine Baker once wrote: "I was learning the importance of names – having them, making them – but at the same time I sensed the dangers. Recognition was followed by oblivion, a yawning maw whose victims disappeared without trace."

Baker, who died in 1975, never lived to see the birth of Java. I do hope that no yawning maw ever swallows it up only 15 years later just because those in charge of naming and renaming it were afraid to sufficiently disturb the lexicon.

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® XMLSpy® 2006 – The industry standard XML development environment.

ALTOVA® xmlspy® 2006

Bring your development plans to light

Sneak a peek at XMLSpy® 2006, and see how essential it is to master XML.

New in XMLSpy 2006:

- Schema-aware XSLT 2.0 support
- Schema-aware XQuery support
- Updated platform integration for Microsoft® Visual Studio® .NET 2005
- Updated platform integration for Eclipse 3.1

Altova® XMLSpy, the industry standard XML development environment, is indispensable for modeling, editing, debugging and transforming all XML-related technologies. Illuminate your strategy with advanced standards compliance, extended platform integration, and enlightened usability aides. Use XMLSpy to structure XML Schemas and devise XML documents, then automatically generate runtime code from schemas in multiple programming languages. Become a markup mastermind!

Download XMLSpy® 2006 today: www.altova.com

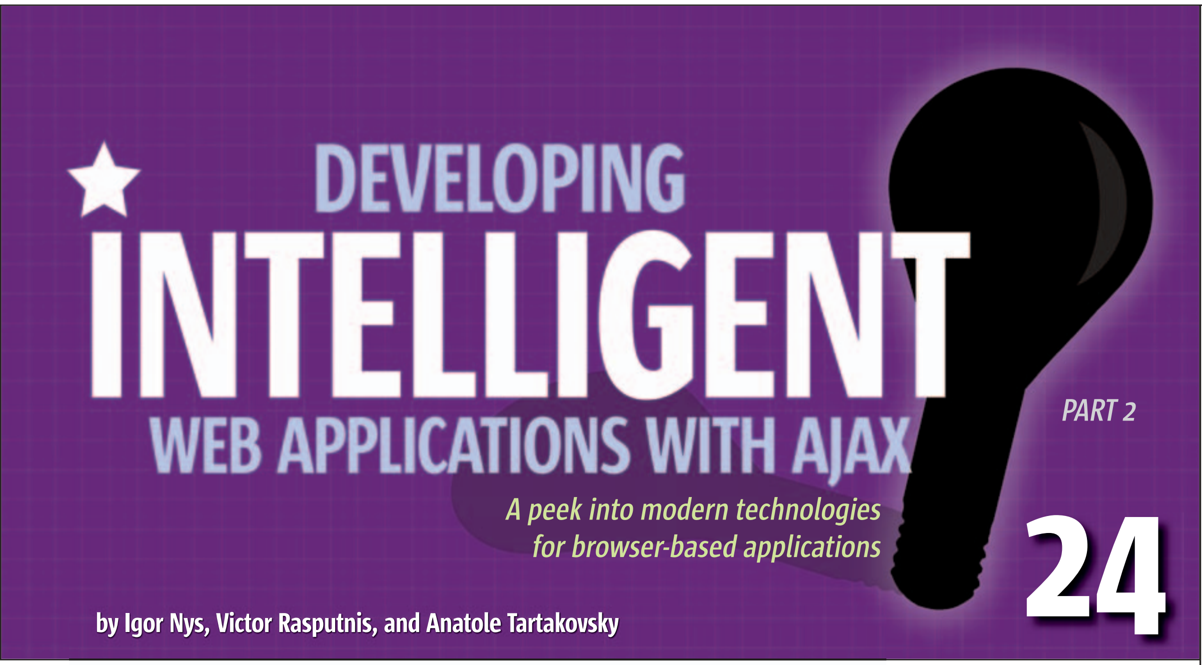
Also available in the Altova XML Suite.

ALTOVA®

NOVEMBER 2005 VOLUME:10 ISSUE:11

JDJ contents

JDJ Cover Story


DEVELOPING
INTELLIGENT
WEB APPLICATIONS WITH AJAX
PART 2
A peek into modern technologies for browser-based applications
24
by Igor Nys, Victor Rasputnis, and Anatole Tartakovsky

Features

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.

Yakov Fain
Contributing Editor

Arranged Java Marriages

As per Wikipedia, "*an arranged marriage is a marriage in which the marital partners are chosen by others based on considerations other than the pre-existing mutual attraction of the partners.*"

This definition comes to mind when I see how large IT organizations prearrange "marriages" between Java application developers and architects. I'd like to discuss potential issues between architects and developers and, to avoid confusion, I'll keep quoting Wikipedia in *italic font*.

The Honeymoon

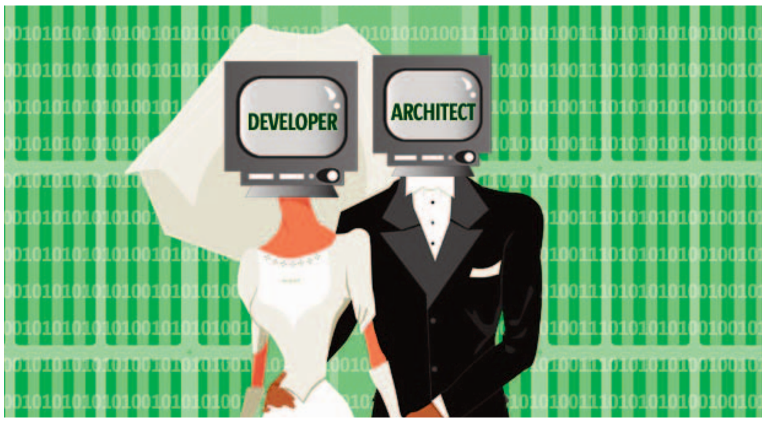
As soon as your IT department grows to more than a half of a dozen Java developers, the leader of the pack (the architect) suggests centralized creation of reusable components. This is an easy sell: your group is agile and still not too large and, if one of these components needs to be changed, the architect is right on the premises and he or she works for you and on your schedule. Developers know on which shelf a singleton object resides and where the date transformation utilities are located, and they are happily reusing them as the need arises. At this stage we can call relations between developers and architects consensual.

"Architecture groups often turn themselves into small kingdoms where mere Java mortals are not allowed"

The Family Life (Corporate Politics)

Time goes by, the economy is on the rise, and older Java species bring in the younger ones. The population increases. Management moves the architects from several application development units into a new group where they can increase reusability of the objects and frameworks across the enterprise.

–continued on page 8



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

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™

–continued from page 6

When a new development project begins, you (the application developer) are told that you must use only the objects and frameworks recommended by the architecture group. Basically, you don't have a choice.

Noble families, especially reigning families, long used arranged marriage to consolidate their strengths and to join their kingdoms. The parents, who often arrange the marriages, are trusted to make a match that is in the best interest of their children; though there are times when the choosers select a match that serves their interest and not the couple's.

Yes, your architects create new components and frameworks, but don't they have to compete with outside third-party vendors? If Jakarta Commons has a generic pool object, why are you not allowed to use it in your project and have to use the homegrown pool instead?

Arranged marriages can also be very flexible. In one scenario, the parents introduce their son or daughter to several potential mates, while giving two the final decision, given some time.

Here's a typical conversation over the morning coffee:

- Darling, I need a generic Java class that would run SQL queries that are given in an XML file.
- No problem, honey. Now I'm working on a very exciting project: a global logger that will allow reading of any log file on any specified corporate workstation. But I'll definitely look into your request next month.
- But I have my deadlines… Remember, you promised that my wish would be your order…

Architecture groups often turn themselves into small kingdoms where mere Java mortals are not allowed (they might have picked up this infection after multiple unsafe relations with Oracle DBAs). Their main business is now the evaluation and purchase of the third-party tools and the introduction of new software layers between these tools and business applications. They know how to talk the talk, and the CIO rests assured that everything goes well. Meanwhile, experienced application developers start to quietly develop their own components to meet their deadlines. Their weak attempts to offer these completed useful components back to the architecture group are not always well received.

Proponents of arranged marriages claim that arranged marriages are more successful than other marriages. They hold that the spouses in an arranged marriage begin without any expectations from each other, and that as the relationship matures, a greater understanding between the two develops.

The Family Budget

Who pays the architects' salaries? The architects usually cut slices from the approved budgets for business application development. I am absolutely not against such deals as long as the architects don't forget who makes their living. They can really save the firm's money by suggesting solutions leading to efficient utilization of existing server licenses, idling hardware, use of open source products, parallel computing, performing code reviews, mentoring of junior developers, delivering technical training (not the one that exists in the approved list of courses), and suggesting best practices that are immediately applicable to business systems.

Arranged marriages operate on the notion that marriages are primarily an economic union or a means to have children.

Unfortunately, not every marriage produces children.

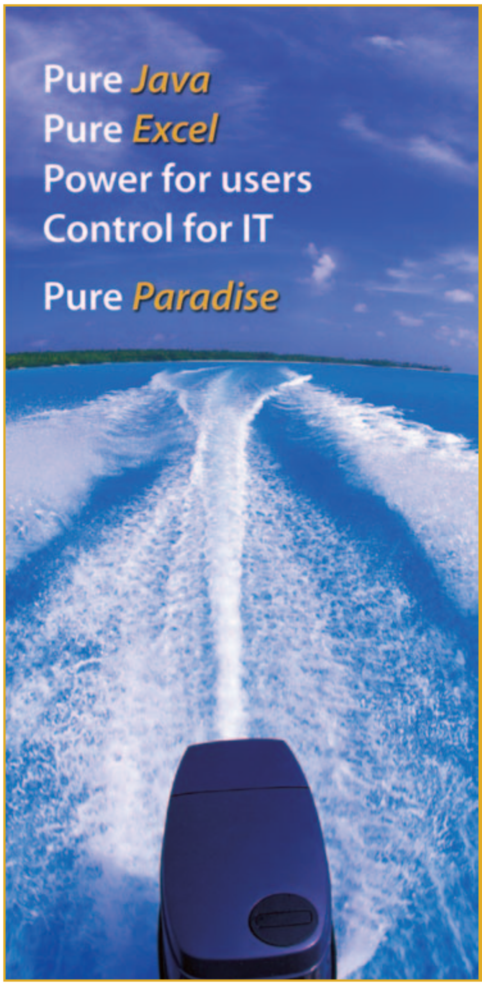
Divorce Is Not an Option

It has also been said that in some cultures where divorce is forbidden or uncommon, arranged marriage would work out nicely because both husband and wife would accept the marriage, producing their best efforts to make it a success instead of breaking up at the slightest conflict.

Needless to say that application developers must also put their best foot forward and stop blaming architects when something doesn't work right. The chances are that you didn't spend enough time learning how to use these components. Maybe they're not that bad?

Oh well, it's time to take a shower, go to bed, and have relations with my spouse…oops…I meant to say it's time to go to a meeting with the Java architecture group.

"If Jakarta Commons has a generic pool object, why are you not allowed to use it in your project and have to use the homegrown pool?"



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
- Embed live, Excel-compatible data grids
- Automate complex calculations and rules
- Read and write server-based Excel spreadsheets

FREE TRIALS, DEMOS AND SAMPLE CODE

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.

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

ACTUATE ReportingEngines

ACTUATE BIRT

ECLIPSE REPORTING YOU CAN DEPEND ON

Introducing Actuate BIRT, the Business Intelligence and Reporting Tools you need for your Eclipse applications. With a single click, you install Actuate BIRT and all related plug-ins, and then you can use BIRT to:

- Guide you through all of the steps of developing a report
- Create reports with full language support
- Manage and test your data sources and queries
- Create report designs with an object-based, drag-and-drop, WYSIWYG editor
- Edit properties for standard and custom report items

With Actuate BIRT, your organization gets the license indemnification, technical support and software maintenance it needs to make BIRT a reliable part of its Java strategy. Try a free download of Actuate BIRT today at **www.actuate.com/birt** or call us at 800-884-8665.

ACTUATE BIRT.
ALL THE POWER OF OPEN-SOURCE REPORTING WITH NONE OF THE RISK.

Scripting Languages Ease Development and Administration

by Tugdual Grall

The ideal solution for automating repeated tasks and creating extensible applications

As developers or system administrators, we all have common tasks that must be replicated again and again. A few examples might be creating a new feature test, populating a test database with data, copying data from a file system to a database, or creating new user accounts…. You get the idea.

Scripting languages share a number of characteristics that make them ideal for managing such redundant tasks:

- Their built-in abstract data types let them easily manipulate lists, arrays, or other composite structures. Also, the languages are often extended with regular expression support and built-in sorting capabilities.
- They support dynamic typing.
- They offer automated memory and exception management.
- They don't need a compiler and can be run directly from a command line or interpreter – meaning you can just write and run.

Because of these characteristics, scripting languages require less code than system-level languages such as Java, C, and C++. Does this mean that the Java language is obsolete and useless and will be replaced by scripting languages? Not at all. Just as shell scripts written in Perl and Python didn't replace C, Java and scripting languages are, in fact, complementary.

Java is more than a simple language. It contains many low-level features, including networking, input-output, XML manipulation, and threading, that have contributed to its success. It's also an enterprise-ready solution based on the services provided by the J2EE platform, including persistence, transaction management, security, Web services, and open connectivity using J2EE Connector Architecture.

Java scripting languages extend the benefits of scripting development with the power of Java – with the goal of making Java development easier and more productive.

Java Scripting Language Offerings

Over the past few years, multiple Java scripting solutions have emerged, and many are encountering good adoption in the developer community. Most of today's Java scripting languages are based on open source projects that either port an existing scripting language to Java or simplify Java to offer a more high-level development approach.

Table 1 lists some of the most popular languages built around Java.

Scripting in Action

If you're not familiar with scripting languages, you might be skeptical about the benefits they provide. Let's look at some simple examples that illustrate these benefits.

Listing 1 provides a small Java program that processes a list of strings and prints those that are shorter than six characters.

Here's the same program written as a Groovy script:

```
list = ["BeanShell","Groovy","Jacl","JRuby"
,"JudoScript","Jython","Rhino"]
shorts = list.findAll { it.size() >= 6 }
shorts.each { println it }
```

As you can see, using Groovy dramatically reduces the number of lines of code that you have to write. It's also easier to read.

In addition to a very concise syntax, most scripting languages provide high-level constructs for dealing with most of the components that current systems use. For example, many provide APIs for manipulating XML documents, simplifying database operations, or creating user interfaces. Some languages such as BeanShell, Groovy, and JudoScript also have native integration with Apache Ant to facilitate build and test tasks, as discussed in the next section.

"Java scripting languages extend the benefits of scripting development with the power of Java"

Tugdual Grall is a principal product manager for J2EE and Web services at Oracle. He joined Oracle in January 1999 initially with Oracle France in consulting, and, since April 2002, he has worked with Oracle Application Server product management. His current areas of focus include J2EE and Web services with Oracle Containers for J2EE (OC4J)

tugdual.grall@oracle.com

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

Language	Description	Web Site
BeanShell	Java Source interpreter	www.beanshell.org
Groovy	Java-like language that comiles as Java classes.	groovy.codehaus.org
Jacl	Java implementation of the Tcl scripting language.	tcljava.sourceforge.net
JRuby	Java implementation of the Ruby scripting language.	jruby.sourceforge.net
JudoScript	Javascript-like language that runs in a JVM.	www.judoscript.com/
Jython	Java implementation of the Python scripting language.	www.jython.org/
Rhino	Java implementation of the Javascript interpreter. This will be built into the next release of Java (Java 6/Mustang).	www.mozilla.org/rhino/

Table 1

Scripting Languages and Apache Ant

Using Ant to automate tasks such as building and testing development projects has become almost standard practice in the Java development cycle. However, there are often times when using Ant for complex tasks isn't feasible. In such cases, it may be easier to write a quick script using Groovy, JudoScript, or BeanShell and embed it in the Ant build.xml file.

The following example shows how BeanShell instructions can be embedded in a build.xml file:

```
<project name="antBeanShell"
default="execScript" basedir=".">
    <target name=" execScript ">
        <!-- Run script from a file -->
        <script language="beanshell"
src="demo.bsh"/>
        <!-- Run script in-line -->
        <script language="beanshell">&lt;!
[CDATA[
            for(int i=0; i<10; i++ )
                print( "i="+i );
        ]]&gt;</script>
    </target>
</project>
```

You can also replicate Ant scripting using Groovy. In this case, you wouldn't use XML to manipulate the different tasks; instead, you'd use a real scripting language, which would allow better constructs such as loops and logical branching. You can even mix Ant and Groovy operations in a single script file.

Where Should You Use Scripting Languages?

As we've seen, scripts are well suited to automating redundant tasks such as building and testing applications. They can also be useful when you need to create quick prototypes for a specific feature.

Although some developers often question its use, you can use scripts to build complex applications – or at least use them inside complex applications or systems. Outside of the Java world, developers have used languages such as PHP, Perl, Python, and Tcl to create multiple Web sites or as part of complex integration projects.

The story is much the same with regard to Java scripting languages. While you might not be able to completely build a complex application using one of the scripting languages I've mentioned, these technologies can help you create more agile solutions.

As a developer, you may often have to add new business logic to your application due to constantly changing requirements. However, you may also need your application to be extensible so that advanced users or third-party developers can plug in their own modules.

Administrators and developers can use scripts to connect to a Java application (such as J2SE or J2EE) using JMX so they can configure and deploy it. This is another example where scripts leverage a low-level Java API to easily perform complex tasks.

For such uses, scripting languages are ideal. Already, some Web applications, such as Portal or Wiki, let developers add new features by publishing their own scripts to the product. Desktop applications based on Swing, SWT, or even native languages also offer extensibility based on scripts.

Another example is the service-oriented architecture world, where the goal is to interconnect various services. Here, using scripting technologies to provide the plumbing in a loosely coupled manner is especially practical, since scripts are easy to change and completely dynamic.

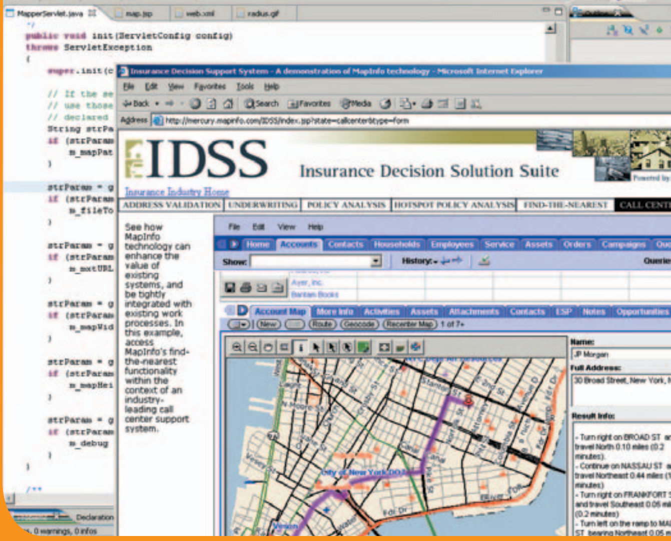
Wrapping It Up

Java scripting languages can provide the ideal solution for Java developers or system administrators who need to automate repeated tasks or create extensible applications, while still leveraging the power of Java and the J2EE platform. Along with reducing code, scripting languages integrate well with commonly used tools such as Ant. The dynamic nature of scripts also makes them ideal to use when you're adding features to existing products.

Another sign of the importance of scripting in current development projects is that scripting technologies will be directly integrated into Java; Java 6 (code name Mustang) will contain a JavaScript interpreter based on Rhino. Also, the JSR 223, Scripting for the Java Platform, is a JCP specification that defines a low-level API to expose a scripting language engine to the Java platform. The current reference implementation integrates PHP, JavaScript, and Groovy.

Listing 1

```
  public class Sample1 {
  public static void main( String[] args
     ) {
  List list = new java.util.ArrayList();
      list.add( "BeanShell" );
      list.add( "Groovy" );
      list.add( "Jacl" );
      list.add( "JRuby" );
      list.add( "JudoScript" );
      list.add( "Jython" );
      list.add( "Rhino" );
      Filter filter = new Filter();
      List shorts = filter.filterLonger
        Than( list, 6 )
      for ( String item : shorts )  {
        System.out.println( item ); }
  }
  public List filterShorterThan( List
        list, int length )  {
  List result = new ArrayList();
  for ( String item : list )  {
  if ( item.length() >= length ) {
    result.add( item ); }
      }
      return result;
    }
}
```


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
File Edit View Help
Home Accounts Contacts Households Employees Service Assets Orders Campaigns
Show:
History:
Ayer, Inc.
Bantam Books
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

    // If the se
    // use those
    // declared
    String strPa
    if (strParam
        m_mapPat
    }

    strParam = g
    if (strParam
        m_fileTo
    }

    strParam = g
    if (strParam
        m_mxtURL
    }

    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.

- Turn right on BROAD ST and travel North 0.10 miles (0.2 minutes).
- Continue on NASSAU ST and travel Northeast 0.44 miles (1 minutes)
- Turn right on FRANKFORT ST and travel Southeast 0.05 miles (0.2 minutes)
- Turn left on the ramp to MA ST bearing Northeast 0.05 m

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

Migrating J2EE Applications from Development to Production

by Michael Baum

Is it becoming more complex?

Decades ago, when we were all computing on mainframes, the application stack was pretty simple. Programs were all running in core memory on the same machine as the operating system and the data store. There was typically one transactional log for activity and one for errors – not too many places to look for evidence of what had gone wrong and why.

Then the application stack started to grow up. Client software communicated over a network to a server running a more featured operating system and an RDBMS (relational database management system). The biggest challenge for developers was debugging those fat clients – typically by looking at something like Windows event viewer logs. System administrators were generally busy responding to alerts from SNMP traps and rummaging through network device logs or sniffing TCP traffic looking for the source of network failures.

Fast forward to the year 2005. Now our compute environments are distributed architectures based on a multitude of open source and proprietary components.

Each application requires many tiers of technology. Physical data center and logical application stack complexity is exploding. It's not uncommon for a mission-critical application in a single deployment to require a hundred or more physical machines and devices.

Michael Baum is chief executive splunker at Splunk. During his career, he has been building and managing large distributed infrastructures, most recently at Yahoo and Infoseek. His focus now is applying many of the innovations and affordances that grew out of the Web to the challenges of managing the modern data center. You can read his blog at www.splunk.com/blogs/thebaum

What Does This Mean for Java Developers? Mountains of Data...

The significance of all of these new dependencies for the typical enterprise application is that managing the modern stack requires the ability to understand large amounts of evidential data produced by all these different tiers, components, and technologies. This is especially true for J2EE applications.

A recent survey by JBoss of their customers revealed that the typical J2EE application gets deployed in four environments on its way to going live: development, QA, staging, and production. Each migration becomes increasingly difficult, time-consuming, and expensive. Each environment introduces new troubleshooting tasks and involves more people and resources for resolution. Code defects, configuration errors, resource conflicts, and multi-tier dependencies are key sources of the pain.

Developing, deploying, and managing J2EE applications today means being able to understand how Web servers, application servers, databases, storage systems, security, and networking all work together. At the Web tier we're greping Apache and IIS logs. Our J2EE application servers are generating JMS, log4j, custom Java logging, and JMX. If we want to figure out persistence dependencies, the database tier requires an understanding of JDBC (Java Database Connectivity) exceptions, redo or audit tables, slow query information, and replication status.

When we look at an application server, a Web server, or a database in a medium-sized enterprise data center, it could be generating (depending on the verbosity level) anywhere from 10 to 100 megabytes of data per day in its logs. If you add up all the subcomponents of the typical enterprise data center, log data can account for upwards of a terabyte of data in a single day.

Unfortunately, there's a tremendous amount of disparity between different types of log file data. An Apache log, for example, has little, if anything, in common with a MySQL slow query log or a JBoss log4j server log. However, entries from different sources of evidence do typically have a timestamp and an IP address from the host generating the evidence. Occasionally there will be one or two common values like a session ID, thread, or process number, or, in the case of a database, a SELECT statement. These are the kinds of clues we need to look for to figure out what is actually happening at runtime within the application stack.

Let's look at a few of the environments that a J2EE application gets deployed in on its way to going live. For each, I'll show a common breakdown that Java developers are experiencing today.

Development and QA Environments

Deployment of a J2EE development environment for building and debugging Web-based applications requires at least three tiers of the application stack – a Web server, an application server, and a database. Initial installation and configuration, debugging container interaction problems, or optimizing code for performance are often troublesome tasks that require access to multiple sources of evidence, including Web server access and error

“Developing, deploying, and managing J2EE applications today means being able to understand how Web servers, application servers, databases, storage systems, security, and networking all work together”

logs, application server logs, the application logs themselves, EJB and JMS activity, JVM state data through JMX, JDBC exceptions, and database Redo or activity logs.

Consider the following example of debugging a Java container-managed persistence problem. This is a very common situation where, after deploying an application in a development environment, we decided to upgrade the database to one that resembles more of what we'll use in production when our application goes live. Many development environments by default come bundled with a simple database. In our case, we're using Hypersonic, which comes bundled with JBoss, and we want to switch it out for MySQL. We've redeployed our application after switching from Hypersonic to MySQL, and the MySQL JDBC library is getting loaded, but we get an exception when trying to run the application.

```
[root@localhost 2005-10-12 06:55:20]# ant
–f jboss-build.xml run-cmp
Buildfile: jboss-build.xml
run-cmp:
   [java] Caught an exception:
   [java] java.rmi.Server Exception:
RemoteException occurred in server thread;
nested exception is:…
```

The rest of the exception was not helpful so we decided to look through the JBoss log4j server for entries with a severity level of “error.” Narrowing the search to the same time-frame as the exception helps minimize the number of errors on which we need to focus. In our case, we found the following error indicating that our EJB could not be instantiated, but it doesn't tell us why.

```
2005-10-12 06:55:29,111 ERROR [org.jboss.
ejb.plugins.LogInterceptor] EJBException
in method: public abstract void roster.
Roster.createLeague(util.LeagueDetails)
throws java.rmi.RemoteException: javax.ejb.
EJBException: Could not instantiate bean;
CaughtByException is:
*Could not instantiate bean
*at roster.RosterBean.
createLeague(RosterBean.java:439)…
```

Again, the rest of the log entry was not really helpful in this situation, so we decided to search the log for “debug”-level entries for our RosterBean around the same time-frame as the EJB error. We discovered that a state file was missing. Aha! The application server was probably holding on to the old application deployment state file, so we deleted the state files and restarted the application server.

```
2005-10-12 06:55:29, 159 DEBUG [org.
jboss.ejb.plugins.AbstractInstanceCache]
Activation failure javax.ejb.EJBException:
Could not activate; failed to restore
state; CausedByException is:
*/opt/live_jboss/jboss-4.0.2/server/alltmp/
sessions/RosterBean-eepnx16n-1m/eepo2fs6-
1p.ser (No such file or directory)
*at org.jboss.rjb.plugins.StatefulSessionFi
lePersistenceManager.activateSessions(Statef
ulSessionFilePersistenceManager)…
```

Moving to Staging Environments

Once a J2EE application has been developed and debugged, it typically gets moved into a staging environment that more accurately represents a model of the destination production systems. Staging an application involves not only developers but also system administrators who are responsible for setting up and managing the staging environment. Because the environment can be significantly more involved than development, production migration issues often include IP address and port binding issues, incorrect database and service access permissions, and wrong Java class paths and/or versions. Correcting these types of problems now requires a more detailed understanding of the database tier and system configurations. A knowledge of networking and security also becomes very important.

Scaling Up in Production Environments

Provided your production environment is not too dissimilar from your staging environment, most production issues typically relate to troubleshooting hard-to-find defects, threshold conditions, latent dependency issues, and resource conflicts. Unfortunately, these are the types of complex system problems that often take hours and sometimes days to track down. These are the types of problems that can suck up a lot of an IT organization's time and money. In fact, most production environments running a modern application stack have become too difficult for help desk and first-line support staff to assist with troubleshooting. So the initial evidence gathering and analysis falls to system administrators (the ones who understand physical complexity) and developers (the ones who understand logical complexity).

We've all been through the reported incidents of application availability problems resulting in the multi-tier “blame game” drill. Investigating the chain of evidence from a Web server to the application server to the database across a network can be very painful. There are many sources of IT data and it often takes two or more domain experts to understand the data.

In this example, we see a series of events from three tiers that needs to be located to track the trail of a single Web browser posting data into a database. The transaction in question is a $200 ATM withdrawal from account ID 5005.

First, from the Apache access log, we're able to find the following by looking through the logs for the account ID:

```
127.0.0.1 - 200 [18/Oct/2005:08:53:31 -
0800] “GET /bank/atm?accountId=5005&opera
tion=0 HTTP/1.1” 200 1987 “http://local-
host:8080/bank/main” “Mozilla/5.0 (X11; U;
Linux i686; en-US; rv:1.7.8) Gecko/20050524
Fedora/1.0.4-4 Firefox/1.0.4”
```

```
127.0.0.1 - 200 [18/Oct/2005:08:53:40 -
0800] “POST /bank/atmAck HTTP/1.1” 200 1148
“http://localhost:8080/bank/atm?accountId
=5005&operation=0” “Mozilla/5.0 (X11; U;
Linux i686; en-US; rv:1.7.8) Gecko/20050524
Fedora/1.0.4-4 Firefox/1.0.4”
```

Next, from the application server logs, we are able to find the trail of events processing the application logic. We find the first event by looking at the account ID and follow the events in time to the remainder of the trail:

```
2005-10-18 08:53:39,793 INFO [STDOUT]
Debug: 5005

2005-10-18 08:53:39,793 INFO [STDOUT]
Debug: Setting operation to: 0

2005-10-18 08:53:39,793 INFO [STDOUT]
Debug: Setting amount to: 220.00

2005-10-18 08:53:39,793 INFO [STDOUT]
Debug: Setting account id to: 5005

2005-10-18 08:53:39,783 INFO [STDOUT]
Debug: /atmAck
```

Finally, from the database, we see the connection to the SQL statements with the account ID and EJB name:

```
/*C10*/SET AUTOCOMMIT FALSE
DELETE FROM NEXT_ID WHERE BEANNAME='tx'
INSERT INTO NEXT_ID VALUES('tx',137)
COMMIT
/*C6*/INSERT INTO TX
VALUES('137',NULL,'2005-10-18
08:53:39.931',-220.00,7108.28,'ATM
Withdrawal')
DELETE FROM ACCOUNT WHERE ACCOUNT_ID='5005'
INSERT INTO TX VALUES('137','5005','2005-
10-18 08:53:39.931',-220.00,7108.28,'ATM
Withdrawal')
COMMIT
```

In this simple example we had enough associative data information to follow a series of cascading events from one tier to the next; however, it's usually not this easy. Most multi-tier systems don't have this kind of identifier threading through each tier of technology. More advanced troubleshooting techniques, such as searching for time-based correlations or transitive data connections, are usually required. Obviously, knowing your data and knowing what to look for helps. Having automated tools that can quickly search through multiple data sources is a good idea too.

Look What's Coming Next

Someone once told me that data centers are like roach motels…stuff comes in but it never comes out. As Java and J2EE technologies become even more mission-critical, Web services become more widely deployed, and new layers like virtualization arrive, look out!

At the recent JBoss World conference in Barcelona, I met several developers working on the next generation of large-scale, truly mission-critical applications. Samuel Stammbach from SkyGuide in Switzerland is building the world's first air traffic control system based on J2EE. Ron Eller from HP talked about the real-time reticketing application they've built for Continental Airlines to provide instant changes to tickets and the ability to change a ticket online. Jean-Marie Lapeyre is the director of project Copernic, a 10-year program with a 1B Euro budget to build a simplified online tax system for 55M citizens and 3M businesses in France. Copernic has 4,000 Linux servers and a committed goal to solve every problem within 48 hours. The Copernic Web site had 5B hits for the current tax season during a two-month period and 3.8M returns were filed online last year.

All of these developers are looking to Web services and service-oriented architectures to reduce complexity through the reuse of software components and easier integrations between J2EE containers, services, and legacy systems. According to Gartner Group, we're all buying into the vision. Gartner estimates by 2008, 80% of software development projects will be based on service-oriented architectures and, by 2006, more than 60% of the $500B market for IT services will be based on Web services standards and technologies. Looks like we'll all be adding Web services requests, WS*, and WSTransaction logs to the pile of stuff we already need to regularly sift through.

Another obvious trend to watch is the continued popularity of open source, which is also changing the characteristics of the modern application stack. No longer do the third-party tools and components we use come from a small number of vendors. More than two-thirds of enterprises are using open source technologies and there are more than 100,000 open source projects on SourceForge.net. All this choice is great, but at what cost?

What's the next technology to jump on the stack? It's called virtualization. Some say virtualization will be the solution to the explosion of physical machines and devices while others believe it will only create additional logical complexity, rendering it unmanageable. Time will tell, but for sure all that virtualization state information and activity logs will need to be added to the mountain of IT data we need to grok.

It's been 10 years since the birth of Java, and 4.5 million Java developers would agree that J2EE has become the standard for enterprise applications. Now we're all spending an increasing amount of time and effort trying to keep our applications running as the "application stack" has become more complex. Analyst firms like IDC have sized the total expenditure on managing data centers upwards of $100B. I'd be interested to know how much of that is consumed by the human labor element of troubleshooting J2EE applications.

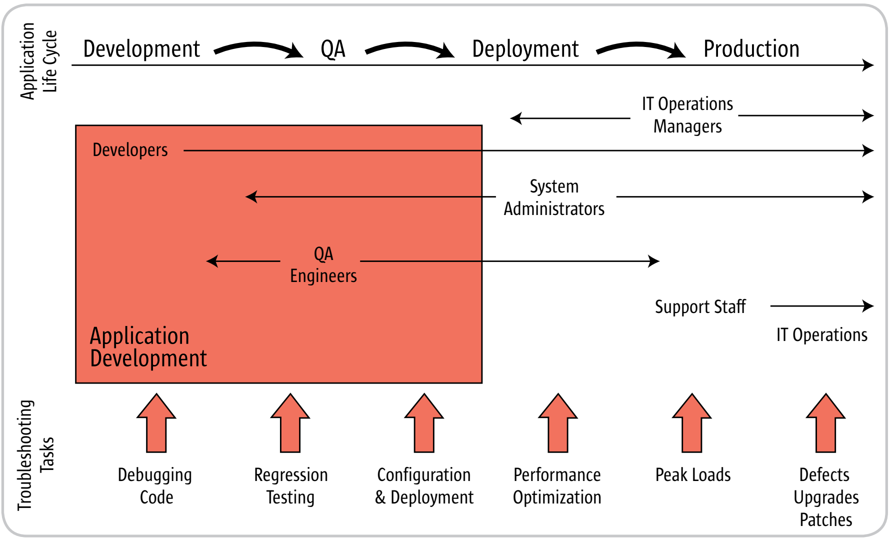


Figure 1 Phases of Application Migration

keep your J2EE problems...at bay. Get freedom for more in life

No more endless hours of sorting out J2EE 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.autopilot2005.com**

AUTOPILOT™
AUTO OPTIMIZER

Creating Web Applications with the Eclipse Web Tools Project

Using open source

by Boris Minkin

The Web Tools Project (WTP) by the Eclipse Foundation is a set of open source tools that substantially reduce the time required for the development of Web applications, EJBs, and Web services. The WTP's current version is 0.7.1 and version 1.0 is coming later this year. The framework provides wizards and tools to create EJBs, Web components such as servlets and JSPs, and Web services using the Axis engine. It also provides source editors for HTML, JavaScript, CSS, JSP, SQL, XML, DTD, XSD, and WSDL; graphical editors for XSD, WSDL, J2EE project builders, models, and a J2EE navigator; a Web service wizard, explorer, and WS-I Test Tools; and database access, query tools, and models.

In this article I'll show you how to develop and deploy a JSP Web application with WTP in less than an hour. I'll also cover the creation and deployment of a basic servlet and editing JSP with WTP. Let's develop the WTP application together but, first, we need to install the following software:

1. J2SE 5.0 JRE: http://java.sun.com/j2se
2. Eclipse 3.1: www.eclipse.org
3. WTP 0.7.1: www.eclipse.org/webtools
4. Tomcat 5.0: http://jakarta.apache.org/tomcat/
5. MySQL 4.0.25: http://www.mysql.com
6. MySQL Connector/J Driver 3.1: http://www.mysql.com/products/connector/j/

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

Application Overview

Our application will be a basic Web application implementing the following use cases:

- Customers need to register on the site to place orders.
- Customers can place orders.
- Customers can view their orders.
- Admin can list all registered customers.

The system will be implemented using a servlet programming model and MySQL database.

Configuring MySQL Database and the Data Source

By default, when MySQL gets installed, a TEST database is available. Be sure to launch C:\Mysql\winmysqladmin.exe to specify the user name and password (the first time you launch it, it lets you do it and also starts the database server). It's necessary to copy MySQL Connector/J JDBC Driver: mysql-connector-java-3.1.10-bin.jar to the Tomcat/common/lib directory, so the Tomcat server can recognize it.

To configure MySQL database access in Tomcat, we have to add a separate file called Listing 1: DBTest.xml; the file follows a convention of "application_name.xml" under $TOMCAT\conf\catalina\localhost directory. The only problem with this file is that it may get deleted when the application is undeployed, so if you undeploy and redeploy the application, you have to place this file into the same folder again (so it's a good idea to save it somewhere else). Looking inside the DBTest.xml file, please note that in our case we are using "ODBC" for the username and don't provide any password.

Building Our Web Application Using Web Tools and a Database

Before we can start working on the Web project, we must configure Tomcat 5.0 in Eclipse to be our default server. When Web Tools are installed in Eclipse, new perspectives and options are added, such as a J2EE perspective where you can create J2EE projects, Web projects, and Web services. New options are available under the Window-Preferences menu for configuring Tomcat servers with Eclipse. Go to Window – Preferences menu, under Server select – Installed Runtimes, click Add, and then specify Tomcat 5.0 server with the installed JRE and a path to the Tomcat installation directory (see Figure 1). Now create a Dynamic Web Project using the Web Tools wizard by selecting File-New-Other, then expanding Web-Dynamic Web Project.

We'll name the project DBTest, which will also become its context root. The Web module will be targeted to our default server: Tomcat. Click Finish and the DBTest Web project gets created. This project will contain all of our Web resources, such as HTML and JSP files, and servlets, and you'll be able to export it into a standard WAR file later, if needed.

Creating Supporting Domain Classes and Tables

Before creating servlets, let's create supporting classes to represent a customer and an order. The class diagram in Figure 2 depicts the Customer and Order relationship.

Note that when creating Customer and Order classes, we define corresponding fields as their public instance variables and then can automatically generate getters and setters from those fields. This can be easily done by going to Outline view (appears after you double-click on an existing class name or create a new class), selecting a class, and selecting "Source – Generate Getters and Setters…" from the right-button menu (see Figure 3).

Along with the classes, we'd have to create corresponding database tables in a MySQL database:

```
CREATE TABLE CUSTOMER (
     ID INT          PRIMARY KEY,
     FIRST_NAME      VARCHAR(50),
     LAST_NAME       VARCHAR(50),
     ADDRESS         VARCHAR(150));

CREATE TABLE ORDERS (
    ID INT      PRIMARY KEY,
    CUST_ID      INT  REFERENCES CUSTOMER,
    DATE_PLACED  DATE,
      AMOUNT     INT);
```

Creating Database Command Classes

We'll create a special package with classes that implement a Command design pattern to perform necessary database updates. The Command pattern allows the classes to implement the common interface executing some particular command. Examples of the Command pattern in Java would be classes that implement the ActionListener interface with the actionPerformed() method. Our Command pattern interface for database integration is presented in Listing 2. (Listings 3–10 and additional source code can be downloaded from http://jdj.sys-con.com.)

Classes implementing this command will be performing the actual database operations for reading and inserting rows into customer and order tables. The following use cases will be addressed:

- Customers want to register on the site in order to place orders (a new row is created in the customer table)
- Customers can place the order (the order gets created in the database for a particular customer)
- Customers can view the orders they have placed
- Admin can list all the customers

Based on these use cases, the following classes have implemented the DatabaseCommand interface (see Listings 3–6):

```
public class CreateCustomer implements
DatabaseCommand
   public class CreateOrder implements
     DatabaseCommand
   public class ListCustomers implements
     DatabaseCommand
   public class ListCustomerOrders imple-
     ments DatabaseCommand
```

Finally, in order to execute our command classes, we will need to create a class that would access the database datasource, obtain a SQL connection, and then execute a particular database command. This class will implement a Singleton design pattern, which we'll call CommandExecutor:

```
Object o = CommandExecutor.getInstance().
executeDatabaseCommand(<<instance of the
particular database command class goes
here>>).
```

The CommandExecutor class will perform the datasource lookup as follows:

```
InitialContext ctx = new InitialContext();
Context envCtx = (Context) ctx.
lookup("java:comp/env");
ds = (DataSource) envCtx.lookup("jdbc/
TestDB");
```

This finds the reference to the datasource we have defined in DBTest.xml by resolving its reference in a Web deployment descriptor (web.xml), which we'll define in a section below. Listing 7 has the complete code of the Command Executor class.

Creating Servlets and JSPs

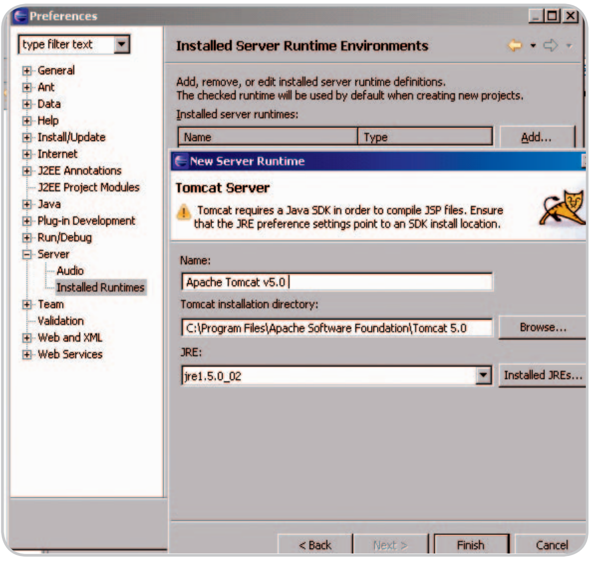
The Model-View-Controller (MVC) paradigm requires the implementation of three loosely coupled tiers in the application:

- ***Model:*** Business logic and domain objects of the application
- ***View:*** Presentation, user interface
- ***Controller:*** Something that sits between the model and view, allowing them to interact in a loosely coupled manner, meaning that the model does not have to be aware of the view, and a different view could possibly be used with the same model.

So far we have implemented only *Model* – domain objects and command classes. *Controller* will be implemented as a servlet and *View* will be implemented as JSPs.

Web Tools provides comprehensive wizards and editors for creating servlets and JSPs. We'll start by creating a package called "servlet" in our Eclipse project where all our servlets will reside. Next, we'll launch the wizard for creating a CreateCustomerServlet by selecting File-New-Other, and then Web-Servlet (see Figure 4).

On the screen in Figure 4, we can specify the Web Project and package where servlet should reside, and the next screen will look like Figure 5.



Add, remove, or edit installed server runtime definitions.
The checked runtime will be used by default when creating new projects.

Tomcat requires a Java SDK in order to compile JSP files. Ensure that the JRE preference settings point to an SDK install location.

Name: Apache Tomcat v5.0

Tomcat installation directory: C:\Program Files\Apache Software Foundation\Tomcat 5.0

JRE: jre1.5.0_02

Figure 1

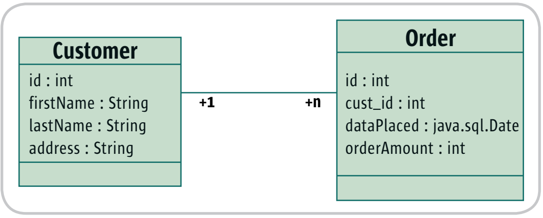
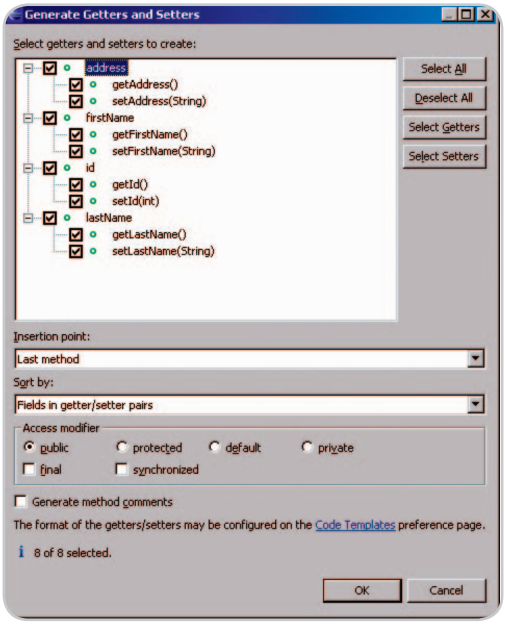


Figure 2



Insertion point: Last method

Sort by: Fields in getter/setter pairs

Access modifier: [✓ public] protected default private

final synchronized

Generate method comments

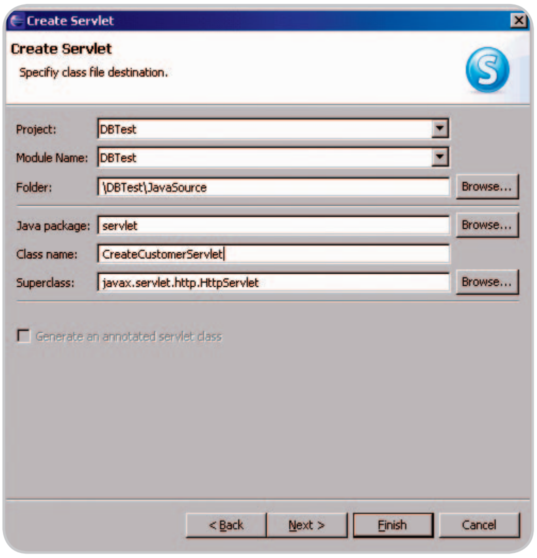
The format of the getters/setters may be configured on the Code Templates preference page.

8 of 8 selected.

Figure 3

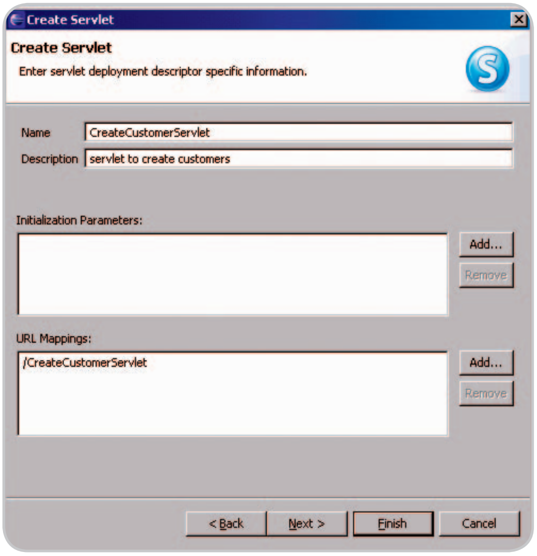
On this one, we can specify the optional description for the servlet, initialization parameters (that can be retrieved by the servlet in the init() method from its ServletConfig object), and URL mapping. Figure 6 shows the final screen for defining servlet.

At this point, we are presented with the capability to specify the interfaces that this servlet will implement (by de-



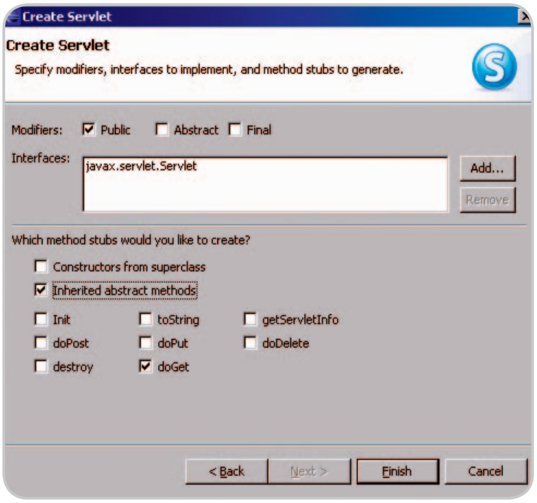
Project: DBTest
Module Name: DBTest
Folder: \DBTest\JavaSource
Java package: servlet
Class name: CreateCustomerServlet
Superclass: javax.servlet.http.HttpServlet
Generate an annotated servlet class

Figure 4



Name: CreateCustomerServlet
Description: servlet to create customers
Initialization Parameters:
URL Mappings: /CreateCustomerServlet

Figure 5


Create Servlet
Create Servlet
Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: [✓ Public] Abstract Final
Interfaces: javax.servlet.Servlet
Which method stubs would you like to create?
Constructors from superclass
[✓ Inherited abstract methods]
Init, toString, getServletInfo
doPost, doPut, doDelete
destroy, [✓ doGet]

Add...
Remove
< Back
Next >
Finish
Cancel

Figure 6

fault, all the servlets implement the javax.servlet.Servlet interface), and the methods that should be automatically generated when the servlet is created.

Just click on Finish and the servlet is generated. This also creates the following entry in the web.xml file:

```
 <servlet>
 <description>Create Customers
 Servlet</description>
 <display-name>ListCustomers</display-
name>
 <servlet-name>ListCustomers</servlet-
name>
 <servlet-class>servlet.ListCus-
tomersServlet</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>ListCustomers</servlet-
name>
 <url-pattern>/ListCustomers</url-pat-
tern>
</servlet-mapping>
```

Another thing that must be added to web.xml is the resource reference to the datasource that has been configured in DBTest.xml:

```
  <resource-ref>
     <description>DB Connection</descrip-
     tion>
     <res-ref-name>jdbc/TestDB</res-ref-
     name>
     <res-type>javax.sql.DataSource</res-
     type>
     <res-auth>Container</res-auth>
  </resource-ref>
```

This XML is necessary for the CommandExecutor datasource lookup code to work properly.

In the init() method of ListCustomersServlet, we will get the instance of the CommandExecutor (see Listing 8), which will retrieve the data source and cache it for future use to supply database connections. It's a good practice to cache a reference to JNDI data source, because retrieving it is a time-consuming operation.

In the doGet() method (see Listing 9), we will execute the database command and forward the list of registered customers to the JSP to display them.

In this doGet method, request attribute "customers" is set to the generic ArrayList and then RequestDispatcher object is used to forward a response to customers.jsp (see Listing 10).

Using some of the new features of J2SE 5.0 such as generics and enhanced "for" loop improves the JSP code by reducing the amount of Java code in it. Creating and editing JSP using WTP is simple. Select the Web content folder of your Web project, right-click, and select New – Other from the menu. From the dialog that appears, select Web – JSP, then specify the name of the JSP you want to create. WTP also allows you to select an existing template for your JSP to be based on. Three built-in templates are provided based on HTML, XHTML, and XML.

In a similar way, we can create other servlets and JSP pages in our application (see the source code), which is easy to import into Eclipse by using its File-Import menu. Figure 7 shows the Web site map of our simple Web application. As you can see, users initially get to index.html page from which they can perform all the other tasks, such as creating customers and orders, and listing existing customers and their orders.

Debugging Application in Eclipse with WPT

The debugging application in Tomcat involves launching the server in the debug mode. To do this, simply right-click on the Tomcat server and select "Debug." To debug some particular line of code, insert a breakpoint at the line shown in Figure 8.

The server currently running in the debug mode will be automatically selected. After a couple of clicks, the embedded browser page should appear. From now on you can debug the Web application the same way as a regular Java program.

Deploying the Application to Tomcat

Once the application has been tested, it may be ready to deploy. To deploy a Web application, first export it as a WAR file. Simply select File – Export – WAR file.

On the popup window you can check "Export source files" and "Overwrite existing file" if you wish to have your source contained within your WAR file (it may not be a good idea in production) and want to suppress a warning if the file with the same name (perhaps a previously exported version) already exists in the directory and you want to simply overwrite it.

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

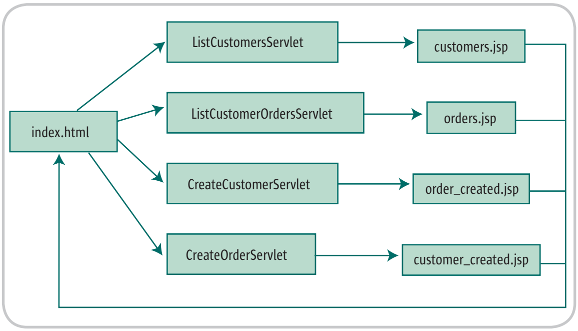
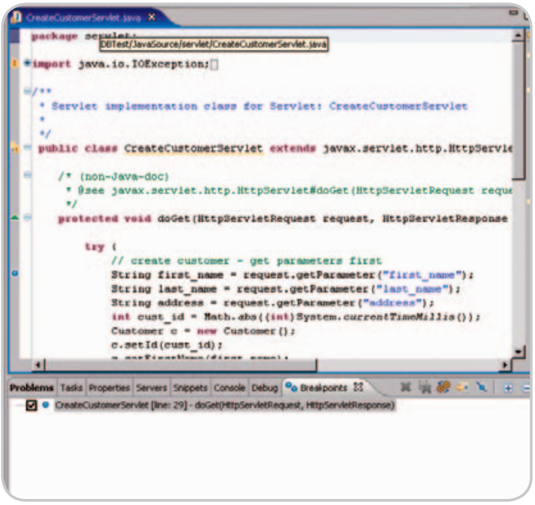


Figure 7


CreateCustomerServlet.java
DBTest/JavaSource/servlet/CreateCustomerServlet.java
Problems Tasks Properties Servers Snippets Console Debug Breakpoints

```
package servlet;

import java.io.IOException;

/**
 * Servlet implementation class for Servlet: CreateCustomerServlet
 *
 */
 public class CreateCustomerServlet extends javax.servlet.http.HttpServle

    /* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest reque
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse

        try {
            // create customer - get parameters first
            String first_name = request.getParameter("first_name");
            String last_name = request.getParameter("last_name");
            String address = request.getParameter("address");
            int cust_id = Math.abs((int)System.currentTimeMillis());
            Customer c = new Customer();
            c.setId(cust_id);
```

CreateCustomerServlet [line: 29] - doGet(HttpServletRequest, HttpServletResponse)

Figure 8

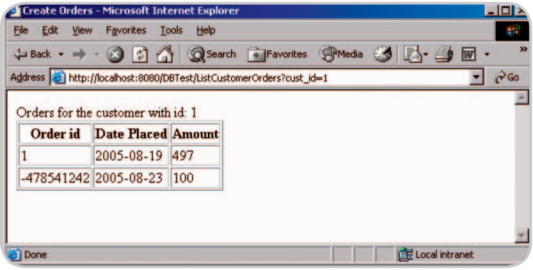
Once the WAR file is exported, deploying it to Tomcat can be done through the Tomcat 5.0 administrative console, which in our case is located at http://localhost:8080/manager/html.

The console displays currently deployed applications and allows you to deploy new, undeploy, and reload existing applications.

Once deployed, we need to make sure that our DBTest.xml file is still there under $TOMCAT\conf\catalina\localhost with all the correct datasource definitions. If it's not, replace it with the correct one that have been saved by you before.

Now, our application is ready to run, so point your browser at http://localhost:8080/DBTest/.

Figure 9 shows what the list of orders for a registered customer will look like.


Create Orders - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://localhost:8080/DBTest/ListCustomerOrders?cust_id=1
Go
Done
Local intranet

Orders for the customer with id: 1

Order id	Date Placed	Amount
1	2005-08-19	497
-478541242	2005-08-23	100

Figure 9

Summary

I've discussed the development, debugging, and deployment of the Web-based database application using J2SE 5.0, Eclipse Web Tools, the Tomcat application server, and the MySQL database. WTP is still in its beta release, so some of its wizards and capabilities may pose problems; however, we were able to successfully debug and deploy the Web application.

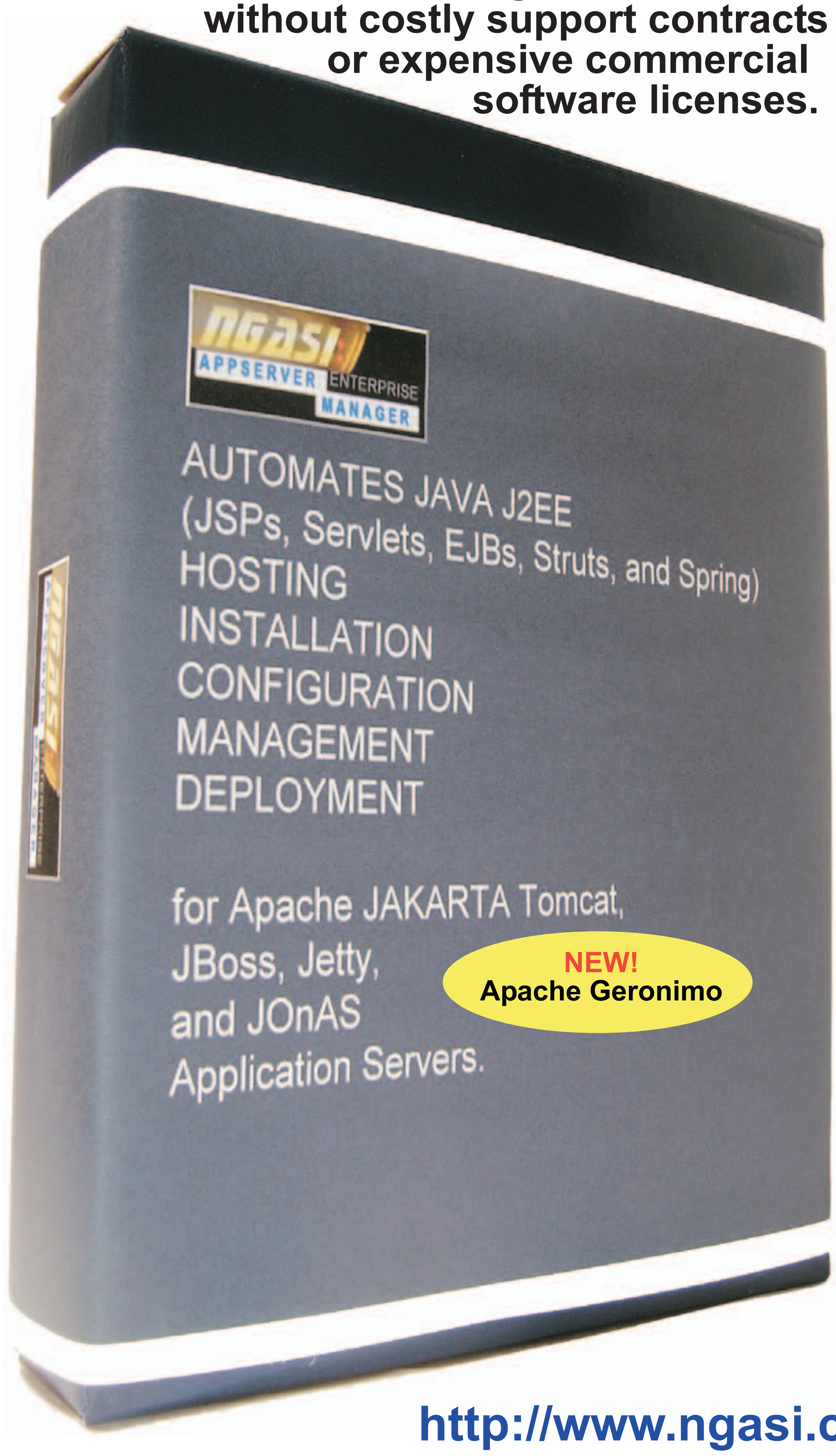
Listing 1

```
<?xml version='1.0' encoding='utf-8'?>
<Context displayName="DBTest" docBase="C:/Program Files/Apache
Software Foundation/Tomcat 5.0/webapps/DBTest" path="/DBTest"
workDir="work\Catalina\localhost\DBTest">

<Resource auth="Container" description="DB Connection" name="jdbc/
TestDB" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/TestDB">
   <parameter>
   <name>factory</name>
   <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
   </parameter>
   <parameter>
   <name>url</name>
   <value>jdbc:mysql://localhost/test?autoReconnect=true</value>
   </parameter>
   <parameter>
   <name>password</name>
   <value></value>
   </parameter>
   <parameter>
   <name>maxActive</name>
   <value>100</value>
   </parameter>
   <parameter>
   <name>maxWait</name>
   <value>5000</value>
   </parameter>
   <parameter>
   <name>driverClassName</name>
   <value>com.mysql.jdbc.Driver</value>
   </parameter>
   <parameter>
   <name>username</name>
   <value>ODBC</value>
   </parameter>
   <parameter>
   <name>maxIdle</name>
   <value>2</value>
   </parameter>
 </ResourceParams>
</Context>
```

Listing 2

```
package command;

import java.sql.Connection;
import java.sql.SQLException;

public interface DatabaseCommand {
 public Object executeDatabaseOperation(Connection conn) throws
SQLException ;
```

JAVA J2EE Hosting Automation Software without costly support contracts or expensive commercial software licenses.



NEW!
Apache Geronimo

FREE DOWNLOAD

NGASI APPSERVER ENTERPRISE MANAGER

http://www.ngasi.com/jdj.jsp

1.866.256.7973

DEVELOPING INTELLIGENT WEB APPLICATIONS WITH AJAX

PART 2

A peek into modern technologies for browser-based applications

by Igor Nys, Victor Rasputnis, and Anatole Tartakovsky

The publicity that AJAX grabbed over the last half a year is based on closing the gap between the Web applications and the desktop applications, combining the "reach" and "rich." At the same time, the gap between the technological level of AJAX and what corporate developers expect in their modern arsenal is really astonishing. After all, AJAX is neither a tool nor a platform. There is no AJAX standards committee or community process in place. While software vendors are crafting proprietary development platforms on top of AJAX (which pretty much means "from scratch"), early adopters of AJAX are left on their own.

In Part 1 (JDJ, Vol. 10, issue 9) we touched on the foundation of AJAX – the ability to establish script-to-server communication. This is what makes HTML pages dynamic and responsive. Does it mean we are ready to kick-off our own version of Yahoo mail? No, we are not. Here is why: AJAX is a mixed blessing. On one hand it enables us to create rich, desktop-class applications on the Web. On the other, if we compare "page-flipping" Web applications with the client/server or Swing ones, the development practices are not quite the same. What about management expectations? We'll need to get used to the fact that it takes time to build a rich UI. The more flexibility with more permutations the user is allowed – the more time it takes.

The answer, of course, is component libraries, frameworks, and industrial-strength tools. Leaving tools aside, this article concentrates on what is available for AJAX enthusiasts today. Addressing a need to build reusable business components, it focuses on the "hidden" object-oriented power of JavaScript. Also, by addressing a need to build custom-rich UI components, it illustrates a convenient way to encapsulate presentation logic in custom client-side HTML tags.

AJAX Language: Object-Oriented JavaScript

By definition, JavaScript is the language of classic AJAX. Unlike Java, JavaScript does not enforce the OO style of coding. That said, it's surprising how often it's overlooked that JavaScript fully supports all the major attributes of an OO language: inheritance, polymorphism, and encapsulation. Douglas Crockford even named Java Script "The World's Most Misunderstood Programming Language." Let's review the object-oriented side of JavaScript.

Data Types

In Java, a class defines a combination of data and its associated behaviors. While JavaScript reserves the class keyword, it does not support the same semantic as in conventional OOP languages.

It may sound strange but in JavaScript, functions are used as object definitions. By defining a function in the example below you, in fact, define a simple empty class - *Calculator*:

```
function Calculator() {}
```

A new instance is created the same way as in Java – by using the *new* operator:

```
var myCalculator = new Calculator();
```

The function not only defines a class, but also acts as a constructor. The operator new does the magic, instantiating an object of class *Calculator* and returning an object reference in contrast to merely calling the function.

Creating an empty class is nice but not really useful in real life. We are going to fill-in the class definition using a JavaScript *prototype* construct. JavaScript uses prototype to serve as a template for object creation. All prototype properties and methods are copied by reference into each object of a class, so they all have the same values. You can change the value of a prototype property in one object, and the new value overrides the default, copied from the prototype, but only in that one instance. The following statement will add a new property to the prototype of the *Calculator* object:

```
Calculator.prototype._prop = 0;
```

Since JavaScript does not provide a way to syntactically denote a class definition, we'll use the *with* statement to mark the class definition boundaries. This will also make the example code smaller as the *with* statement is allowed to perform a series of statements on a specified object without qualifying the attributes.

```
function Calculator()  {};
with (Calculator) {
 prototype._prop = 0;
 prototype.setProp = function(p) {_prop = p};
 prototype.getProp = function() {return  _prop};
}
```

Igor Nys is a director of technology solutions at EPAM Systems. He was closely involved in the software development based on XMLSP technology – one of the AJAX pioneers.

igordnys@gmail.com

Victor Rasputnis is an IT consultant who has been working in Java, PowerBuilder, C, Assembler - whatever language has appeared since 1976. Victor is one of the creators of XMLSP, the product that pioneered AJAX in 1999.

victorrasputnis@teamcti.com

So far we have defined and initialized the *public* _prop variable as well as provided getter and setter methods for it.

Need to define a *static* variable? Just think of the static variable as being a variable owned by the class. Because classes in JavaScript are represented by function objects, we just need to add a new property to the function:

```
Calculator.iCount = 0;
```

Now that the iCount variable is a property of the Calculator object, it will be shared between all instances of the class calculator.

```
function Calculator()  {
 Calculator.iCount++;
};
```

The above code will count all created instances of the class *Calculator*.

Encapsulation

Using "Calculator", as defined above, permits access to all the "class" data, increasing the risk of name collisions in inherited classes. We clearly need encapsulation to view objects as self-contained entities.

A standard language mechanism of data encapsulation is private variables. And a common JavaScript technique for emulating a private variable is to define a local variable in the constructor, so that this local variable is accessible exclusively via getter and setter – inner functions of the very same constructor. In the following example, the *_prop* variable is defined within the *Calculator* function and is not visible outside of the function scope. Two anonymous inner functions, assigned to *setProp* and *getProp* attributes, provide access to our "private" variable. Also, please note the use of *this*, quite similar to how it is used in Java:

```
function Calculator()  {
 var _prop = 0;
 this.setProp = function (p){_prop = p};
 this.getProp = function() {return  _prop};
};
```

What is often overlooked is the cost of such encapsulation in JavaScript. It can be tremendous, because inner function objects get repeatedly created for each instance of the "class".

Accordingly, since constructing objects based on the prototype is faster and consumes less memory, we cast our vote in favor of public variables wherever performance is critical. You can use naming conventions to avoid name collisions, for example, by prefixing public variables with the class name.

Inheritance

At first glance, JavaScript lacks support for the class hierarchy similar to what programmers of conventional object-oriented languages expect from the modern language. However, although JavaScript syntax does not support class inheritance as in Java, inheritance can still be implemented by copying an instance of a previously defined class into the prototype of the derived one.

Before we provide an illustration, we need to introduce a *constructor property*. JavaScript makes sure that every *prototype* contains *constructor*, which holds a reference to the constructor function. In other words, *Calculator.prototype.constructor* contains a reference to *Calculator()*.

Now, the code below shows how to derive the class *ArithmeticCalculator* from the base class *Calculator*. "Line 1" results in borrowing all properties of the *Calculator*, while "Line 2" restores the value of the *prototype*, *constructor* back to *ArithmeticCalculator*:

```
function ArithmeticCalculator() { };
with (ArithmeticCalculator) {
    ArithmeticCalculator .prototype = new Calculator();   //Line 1
    prototype.constructor = ArithmeticCalculator;         //Line 2
}
```

Even if the example above looks like a composition rather than inheritance, the JavaScript engine knows about the prototype chain. In particular, the *instanceof* operator will work correctly with both the base and derived classes. Assuming you create a new instance of a class ArithmeticCalculator:

```
var  c = new ArithmeticCalculator;
```

expressions *c instanceof Calculator* and *c instanceof ArithmeticCalculator* will both evaluate to *true*.

Notice, that the constructor of the base class in the example above is called at the point when the ArithmeticCalculator prototype is initialized and not when an instance of the derived class is created. This could have unwanted side effects and you should consider creating a separate function for initialization purposes. As the constructor is not a member function, it can't be called through *this* reference directly. We will need to create a "Calculator" member function to be able to call *super*:

```
function Calculator(ops) { …};
with (Calculator) {
 prototype.Calculator =  Calculator;
}
```

Now we can write an inherited class that explicitly calls the constructor in the base class:

```
function ArithmeticCalculator(ops) {
 this.Calculator(ops);
};
with (ArithmeticCalculator) {
 ArithmeticCalculator .prototype = new Calculator;
 prototype.constructor = ArithmeticCalculator;

 prototype.ArithmeticCalculator = ArithmeticCalculator;
}
```

Polymorphism

JavaScript is a non-typed language where everything is an object. Accordingly, if there are two classes A and B, both defining method *foo()*, JavaScript will allow polymorphic invocation of *foo()* across instances of A and B even if there is no hierarchical relation (albeit implementational) whatsoever. From that perspective, JavaScript provides a wider polymorphism then Java. The flexibility, as usual, comes at a price. In this case, it is a price of delegating the type checking job to application code. Specifically, if there is a need to check that a reference indeed points to a desired base class, it can be done with the *instanceof* operator.

Anatole Tartakovsky is a New York-based software developer, lecturer, consultant, and author. He is currently working as the CTO at Computer Technology, Inc., focused on developing AJAX/FLEX solutions for the financial and retail industries.

anatolet@teamcti.com

On the other hand, JavaScript doesn't check parameters in the function calls, which prevents from defining polymorphic functions with the same name and different parameters (and let the compiler choose the right signature). Instead, JavaScript provides an *argument* object – Java 5 style – within a function scope that allows you to implement a different behavior depending on the parameter's type and quantity.

Example

Listing 1 implements a calculator that calculates expressions in a reverse Polish notation. It illustrates the main techniques described in the articles and also shows the usage of the unique JavaScript features, such as accessing object properties as an array element for a dynamic function call.

To make Listing 1 work we also need to provide a piece of code that instantiates the calculator objects and calls the *evaluate* method:

```
var e = new ArithmeticCalcuator([2,2,5,"add","mul"]);
alert(e.evaluate());
```

AJAX Component Authoring

All AJAX component authoring solutions known today can be logically divided into two groups. The first group specifically targets the seamless integration with the HTML-based UI definition. The second group drops HTML as a UI definition language in favor of certain XML. In this article we illustrate one approach from the first group, an analog to JSP tags, albeit in the browser. These browser-specific component authoring extensions are called *element behaviors* in the IE case or *extensible bindings* in the case of the latest versions of Firefox, Mozilla, and Netscape 8.

Custom Tag Dialects

Internet Explorer, starting with version 5.5, enables the JavaScript authoring of custom, client-side HTML elements. Unlike JSP tags, these objects are not preprocessed into HTML on the server side. Rather, they're legitimate extensions of a standard HTML object model and everything, including control construction, happens dynamically on the client. Similarly, Gecko-engine based browsers can dynamically decorate any existing HTML element with a reusable functionality.

It's possible, therefore, to build a library of rich UI components with methods, events, and attributes that will have HTML syntax. Such components can be freely mixed with standard HTML. Internally, these components will communicate with application servers, AJAX style. In other words, it's possible (and relatively simple) to build your own AJAX object model.

The IE flavor of this approach is called HTC or HTML components; the Gecko version is called XBL – eXtensible Bindings Language. For the purposes of this article, we'll focus on IE.

Enter the HTML Components – HTC

HTC or HTML components are also called behaviors. In turn they are divided into *attached* behaviors that decorate any existing HTML element with a set of properties, events, and methods, and *element* behaviors that look like an extended set of custom HTML tags to the hosting page. Together, *element* and *attached* behaviors provide a simple solution for authoring both components and applications. Here we'll illustrate the most comprehensive case, *element* behaviors.

Data-Bound Checkbox Control

As an illustration of element behavior, we'll construct a custom data-bound checkbox. The rationale behind building such a control is that a standard HTML checkbox has several noticeable shortcomings:

- It requires application code to map the value of the "checked" attribute into business domain values, such as "Y[es]"/"N[o]", "M[ale]"/"F[emale]", etc. The HTML checkbox uses "checked" contrary to many other HTML controls using "value".
- It requires application code to maintain the state of the control (modified versus not modified). This is actually a common problem with all HTML controls.
- It requires application code to create an associated label that should accept click and change the state of the checkbox accordingly.
- The standard HTML checkbox doesn't support "validation" events to allow the canceling of a GUI action under certain application conditions.

To settle on a syntax, let's say that a sample usage of the control we are building could look the following way:

```
<checkbox id="cbx_1" value="N" labelonleft="true"
label="Show Details:" onValue="Y" offValue="N"/>
```

In addition, our control will support the cancelable event onItemChanging and the notification event onItemChanged.

Custom Tag Definition

Structurally, a custom tag is a file with an HTC extension that describes its properties, methods, and events between <PUBLIC:COMPONENT> and </PUBLIC:COMPONENT>.

To define a custom CHECKBOX tag, we create a file checkbox.htc as in the following snippet where the first line sets the tag name of the component:

```
<PUBLIC:COMPONENT NAME="cbx" tagName="CHECKBOX">
<PROPERTY NAME="value" GET="getValue" PUT="putValue" />
// Here we place all other properties of the component
<METHOD NAME="show" />
// Here we place all other methods of the components
<EVENT NAME="onItemChanging" ID="onItemChanging"/>
// Here we place all other events that component will fire to
    application
<ATTACH EVENT="oncontentready" HANDLER="constructor" />
// Here we place all other events that component handles itself
<SCRIPT >
// Here we place all methods, properties getters and setters and
    event handlers
</SCRIPT>
</PUBLIC:COMPONENT>
```

ALTOVA® umodel™ 2005

Visualize works of software art

Draw on UModel™, the simple, cost-effective UML® 2.0 modeling and code generation tool from Altova®

Exhibited in Version 2005:

- Intuitive UML drawing capabilities with helpful design aids
- Java code generation from UML diagrams
- Reverse engineering of UML models from existing Java applications
- Round trip synchronization of output after amending generated objects

Utilize UModel 2005 to interpret and devise software designs. Decode Java programs into clear, accurate UML 2.0 diagrams, or draw up your application strategy and generate standard Java code from your plans. UModel makes visual software design practical for programmers and project managers. With customizable templates and styles, built-in entry helpers, unlimited undo, and many other artful features, getting started with UML has never been easier. Take the mystery out of UML!

Download UModel™ 2005 today: www.altova.com

ALTOVA®

Custom Tag Use Case

While the contents of the HTC file matter a lot, the name of the file is irrelevant, although, ultimately, the URL to the HTC file needs to be specified using the IMPORT instruction. It has to be done before the corresponding custom tag is mentioned for the first time (on the page). Here is how the simplest possible page utilizing a custom checkbox might look, assuming the page and the HTC file are located in one folder:

```
<HTML xmlns:myns>
<?IMPORT namespace="myns" implementation="checkbox.htc" >
<BODY>
<myns:checkbox id='cbx_1' label='Hello'/>
</BODY>
</HTML>
```

Please notice how custom CHECKBOX has been mapped to a nondefault namespace "myns" in the opening HTML tag. The IMPORT instruction performs a synchronous load of the HTC into the browser's memory and also instructs the browser how to perform name resolution for the appropriate namespace (HTC to namespace association can be many-to-one).

Constructor of the Custom Tag

The best way to initialize HTC, once it's loaded, is to process the "oncontentready" event. Accordingly, we define a handler function, which for sheer clarity is called *constructor*:

```
<ATTACH EVENT="oncontentready" HANDLER="constructor" />
```

The logic of *constructor()* is simple: concatenate a regular HTML checkbox and HTML label in the order dependent on the value of property *labelonleft* (see property definition below):

```
function constructor()          {
  // We will add an HTML checkbox and label to the element body
  // See Listing 2 for details
}
```

Defining Custom Tag Properties

To define property *labelonleft*, we add one more line to the <PUBLIC:COMPONENT> section:

```
<PROPERTY NAME="labelonleft" VALUE="true"/>
```

Please note that this property does not contain getter and/or setter methods. Properties *onValue* and *offValue* that provide the mapping of the checkbox status into a business value domain also don't need getters and setters:

```
<PROPERTY NAME="onValue" VALUE="true"/>
<PROPERTY NAME="offValue" VALUE="false" />
```

However, property *checked* is defined with both getter and setter:

```
<PROPERTY NAME="checked" GET="getChecked" PUT="putChecked" />
```

Accordingly, we have definitions for both methods in the <SCRIPT> section. As you can see, setter *putChecked()*, which is triggered every time checked is modified, sets the value property to one of two variants: *onValue* or *OffValue*. Please note that *putChecked()* will get triggered not only by the script on the checkbox-hosting page, but also by any assignment to *checked* done inside this very checkbox.htc.

```
var  _value;
function putChecked( newValue ) {
 value = (newValue?onValue:offValue);
}
function getChecked(){
 return ( _value == onValue);
}
```

Defining Events for the Custom Tag

Let's look at the definition of *onItemChanging* and *onItemChanged* events and how these events are being fired and processed inside the setter for value property (see Listing 2).

Method *putValue()* has a couple of points of interest. First, it can be called during the parsing on the CHECKBOX tag, as long as the HTML value attribute is specified. That's why we have a separate logic branch for not constructed objects, to differentiate the process of construction from a reaction to a user click. Second, we illustrate here the creation and processing of the custom event *onItemChanging*, which allows the application to cancel (block) the action. Please note that both clicking as well as programmatic assignment to the *value* can get cancelled this way.

Event Canceling

To cancel the event, an application should intercept the event and set *event.returnValue* to *false*. The schematic snippet of code illustrating how the application would cancel the processing is presented below:

```
cbx_1::onItemChanging() {
. . . . .
if (canNotBeAllowed) {
 event.returnValue=false;
.   .   .   .   .
}
```

If the event is not cancelled, *putValue()* sets the internal, plain HTML checkbox's *checked* property as per the current value: if that is equal to *onValue*, the internal checkbox will get checked; if it is equal to *offValue* (there is no third option), unchecked (the full listing is presented in Listing 2).

HTML Internals of the CHECKBOX

The painting of our control is done by the helper functions *addLabel()* and *addCheckBox()* and is called from within a *constructor()*. These functions inject HTML inside the *element*'s innerHTML (*element* is the analog for *this* in HTC parlor). The injected HTML, in the simplified form, looks like the following:

```
<LABEL for=cb_{uniqueID}>Show Details:</LABEL>
<INPUT id=cb_{uniqueID} type=checkbox />
```

where *uniqueID* is a unique (within a page) string literal generated by IE, which identifies the instance of the HTC.

Encapsulation ... Again

There is one shortcoming in our CHECKBOX. The way we built it, the HTML injected during the constructor() contributes to the DOM of the page that hosts the HTC. Also global JavaScript variables like_value fall into the global scope of the document they're included in. This is dangerous since we run into the possibility of name clashes: the most obvious case is more than one instance of the same control. In addition, it presents a possibility that our control may accidentally reference other objects with the same names and vice versa.

To put it simply, a special mechanism is required to enable a truly modular approach for object authoring. Fortunately, HTC technology supports an intelligent answer – *viewLink*.

The easiest way to declare a control as encapsulated is to put one extra declaration between opening and closing PUBLIC:COMPONENT tags:

```
<PUBLIC:DEFAULTS viewLinkContent/>
```

Instantly, the control becomes encapsulated; it has its own HTML document tree, atomic to the master document. Every instance of the object has its own set of instantiated values and only public methods and properties can be accessed by the outside code. In other words, the viewLink mechanism fully enables the design and implementation of sophisticated Web applications using a true OO component–based approach.

In particular, we can simplify our code by removing *uniqueID* suffixes from the definition of internal checkboxes and HTML labels, since we are not afraid of name clashes anymore. Accordingly, we may replace the line:

```
    eval( ‘cb_’+uniqueID).checked = ( _value == onValue );
```

with the

```
    cb.checked = ( _value == onValue );
```

as well as change *addCheckbox()* and *addLabel()* accordingly.

Conclusion

Since the AJAX race has just started, there are no AJAX standards and no commonly accepted RAD tools you can rely on to build your applications. While software vendors have a long way to go to create the robust development platforms, AJAX enthusiasts can prepare by encapsulating reusable blocks of code as business components with a well-defined API.

Navigating in this direction, this article outlined the OO "powers" of the AJAX language – JavaScript. It also illustrated one of the available component-authoring strategies – client-side custom tags technology. While we presented only IE-specific custom tags, we also provide a downloadable example of *extensible bindings* example for the Mozilla browser. All article-related examples can be downloaded from http://www.ajaxmaker.com/JDJ/AJAX/partII.html.

Listing 1

```
// Constructor
function Calculator(ops) {
     this.ops = ops;
}
with (Calculator) {
 // Make constructor available as a member function
 prototype.Calculator = Calculator;
 // Member functions
 prototype.evaluate = function () {
  this.opsStack = new Array();
  for (var i = 0; i < this.ops.length; i++) {
   var op = this.ops[i];
   if (typeof op == “number”)
    this.push(op);
   else if (typeof this[op] == “function”)
    this[op]();
  }
  return this.pop();
 };
 prototype.pop = function () {return this.opsStack.pop();}
 prototype.push = function (val) {this.opsStack.push(val);}
}

//Constructor
function ArithmeticCalcuator(ops) {
 this.Calculator(ops);          // super
};
with (ArithmeticCalcuator) {
 // Establish inheritance
 ArithmeticCalcuator.prototype = new Calculator();
 prototype.constructor = ArithmeticCalcuator;
 // Make constructor available as a member function
 prototype.ArithmeticCalcuator = ArithmeticCalcuator;
 // Member functions
 prototype.add = function () {this.push(this.pop() + this.pop()); }
 prototype.sub = function () {this.push(this.pop() - this.pop());}
 prototype.mul = function () {this.push(this.pop() * this.pop());}
 prototype.div = function () {this.push(this.pop() / this.pop());}

}
```

Listing 2

```
<PUBLIC:COMPONENT NAME=”cbx” tagName=”CHECKBOX”>
<PROPERTY NAME=”label” />
<PROPERTY NAME=”checked” GET=”getChecked” PUT=”putChecked” />
<PROPERTY NAME=”labelonleft” VALUE=”true”/>
<PROPERTY NAME=”value” GET=”getValue” PUT=”putValue” />
<PROPERTY NAME=”onValue” VALUE=”true”/>
<PROPERTY NAME=”offValue” VALUE=”false”/>
<METHOD NAME=”show”/>
<EVENT NAME=”onItemChanging” ID=”onItemChanging”/>
<EVENT NAME=”onItemChanged” ID=”onItemChanged” />
<ATTACH EVENT=”oncontentready” HANDLER=”constructor” />
<SCRIPT >

 var _value;

 function constructor()        {
  var s;
  s = ‘<INPUT name=”cb_’+ uniqueID + ‘” id=”cb_’ + uniqueID + ‘” type=”checkbox”
‘ +
   ‘onclick=”’ + uniqueID + ‘.checked= +cb_’ + uniqueID+ ‘.checked” ‘;
  if( _value == onValue )
   s +=’ checked=”true” ‘;
  s+= “/>”;
  element.insertAdjacentHTML(‘afterBegin’, s);
  element.insertAdjacentHTML((labelonleft == “true”)?’afterBegin’:’beforeEnd’,
   ‘<LABEL for=”cb_’ + uniqueID+ ‘”>’ + label + ‘</LABEL>’);
 }

 function putChecked( newValue ) {
  value = (newValue?onValue:offValue);
 }
 function getChecked(){
  return ( _value == onValue);
 }

 function getValue(){
  return _value ;
 }
 function putValue( newValue ) {
  if (window.event == null)
   _value = newValue; /* initial*/
  else if (_value != newValue) {
   var oEvent = createEventObject();
   oEvent.newValue = newValue;
   oEvent.returnValue = true;
   onItemChanging.fire(oEvent);
   if (oEvent.returnValue)
    _value = newValue;
   eval(‘cb_’+uniqueID).checked= ( _value == onValue );
   if (_value == newValue)
    onItemChanged.fire(oEvent);
  }
 }

 function show(bCmdShow){
  style.visibility= (bCmdShow?’visible’:’hidden’) ;
 }

</SCRIPT>
</PUBLIC:COMPONENT>
```

Jason Bell

'As Long As It Works'

In most cases I'm a patient and tolerant person. Once you get to know me, I'm easy to get along with, occasionally complex, but not very often. My patience and tolerance has pretty much gone out the window in the last week or so. It all stems from two technologies: Ruby On Rails (RoR) and AJAX.

Now let's be fair, no one really gave a garbage-collected object about AJAX until those boffins at Google brought us the "suggest" and maps utilities (this is what happens when you give programmers spare time; they come up with good stuff). What followed were Web sites, APIs, tutorials, and more applications than you can shake a stick at. JavaScript used to be a dirty word among Java programmers a number of years ago. At one point you may as well have called JavaBlogs JavaScript-Blogs.

We used to laugh, have secretly coded words when talking among JavaScript programmers, just to make them feel out of place. JavaScript was the annoying little brother that got in the way when you were trying to impress your new girlfriend. Now VCs are rubbing their hands together waiting to throw money at the next line of AJAX-fueled apps. For about two days I was thinking, "I may actually have to learn some of this stuff," then I came to my senses and thought better of it.

My main problem with all of this goes back to my original problem with all JavaScript things from years ago – browser compatibility. I still can't, to this day, get an AJAX app to work on my Pocket PC version of Internet Explorer (in fact I still can't get a decent JVM for it either). Another issue for me is that everyone has conveniently forgotten about testing any of this stuff with a dial-up connection (or at least it's never mentioned). Is a user seriously going to sit and wait for the data to bounce to and fro every time the mouse moves? Don't get me wrong, there are some real nice applications about, but at the end of the day AJAX is still the name of a cream cleanser in the UK. I usually wash the sink with it.

The arguments, agreements, and articulate announcements about RoR are also currently, in my opinion, spiraling out of control. It sort of reminds me of the same sort of arguments regarding Python and Java a couple of years ago. Ruby, as a language, has been kicking about

for years but never really broke into the mainstream. RoR came about from a project management tool for 37 Signals. Now even the most seasoned Web app programmers can be in a complete giddy spell on how they can knock up a MVC-type Web app in, well, a bit less time than before.

I thought we'd gone beyond this "technology *x* is better than technology *y*" debate when .NET was supposed to be the biggest threat to humanity. We all gave it six months to blow over and then the landscape was quiet once again. Now the soothsayers are out again with their scythes of code reduction telling me that RoR will replace Java in the way that Web apps are done. The Java camp seems to be extremely defensive and a touch prickly toward the arguments, any argument in fact, and in some respects it's how we react to these comments that make us easy prey with the emerging technologies.

As far as the landscape goes, there will be a place for all these technologies. Some will be adopted more than others. Would you write a multi-language banking application in RoR? I wouldn't. What I do keep in my tool belt is a fresh open mind to new things. I looked at RoR, played with it, tried a few things, and thought, "Not for me now" and uninstalled it. More to the point I have more than enough paid work in Java, PHP, and a heap of other things. So I tend to go where the money is calling to make sure there's food on the table.

Where do I stand in all of this? AJAX, well I've seen good uses and some pretty dreadful ones too. These are just the baby steps of what's to come with AJAX-driven apps. I still need some convincing whether we'll all be lapping up AJAX apps to every Web site we ever go to. And where there is JavaScript, there are browser-compatibility issues.

With Ruby On Rails, I'm taking a back seat for six months to see what happens. I don't jump on these things easily, just like I didn't jump on Python. There is a place for it but I still think it's in the realms of the developer fantasy of "this might take over the world one day," when the rest of people really couldn't give a hoot about what language it was written in.

As a client said to me yesterday, "I don't care what you use, as long as it works." I feel the same. Enjoy what you enjoy.

Jason Bell is founder of Aerleasing, a B2B auction site for the airline industry. He has been involved in numerous business intelligence companies and start ups and is based in Northern Ireland.

jasonbell@sys-con.com

Are you using PDF technology in your daily business?

Do your requirements go beyond the standard features offered by your current PDF products?

PDF Tools AG has spent years addressing your needs and is proud to offer the newest line of PDF tools designed for customers relying on PDF technology for their critical business processes:

The 3-Heights™ family of PDF programming components

PDF Tools AG is a world leader in PDF programming technology, delivering reliable PDF products to international customers in virtually all market segments.

PDF Tools AG provides server-based software products designed specifically for developers, integrators, consultants, customizing specialists and IT-departments. Thousands of companies worldwide use PDF Tools AG's products directly and tens of thousands more use the technology indirectly via our growing network of OEM partners. The tools are easily embedded into application programs and are available for a multitude of operating system platforms.

pdf-tools.com
Premium PDF Technology

Contact:
pdfsales@pdf-tools.com

or visit us at:
http://www.pdf-tools.com

Precision Team Management

Interview by Jeremy Geelan

Interview with Nigel Cheshire, CEO, Enerjy Software

Nigel Cheshire's company, Enerjy, has just launched Enerjy CQ^2, a code quality solution for Java development managers. In this exclusive Q&A with *JDJ*, Cheshire contends that software quality from 2002 to 2004 has diminished: "As an industry, it's time to take this problem seriously."

***JDJ:** Nigel, thanks for agreeing to talk with us. Let's get right to it: Could you very quickly just "situate" Enerjy in the i-Technology space, say what the company does and how long it's been doing it?*

Nigel Cheshire: Enerjy is a division of Teamstudio Group, which has been around since the early part of 1996. We provide software integrity solutions to Java development organizations and, before you ask, I can tell you that software integrity, in a nutshell, means finding the problems in your software before they become problems in your business. In other words, it's about early detection and correction, and raising the quality of the software development process.

***JDJ:** What then is the latest release/launch/product from Enerjy?*

NC: We've just launched Enerjy CQ^2, which is a code quality solution for Java development managers. We'd been in the business of providing code quality tools to Java developers for a while and, after much feedback from our customer base, came to realize that we were barking up the wrong tree. The problem with our approach was that it's not developers who truly the shoulder responsibility for the quality of the finished application – it's the development manager. When we looked at what was available by way of tools to help development managers monitor and measure the quality of code as it is being written, we found that almost nothing exists today. Hence Enerjy CQ^2

But Enerjy CQ^2 is not just a tool, it encapsulates a new process for development managers to help them better manage and coach their teams. We call it Precision Team Management, which in essence means the three-step process of:

1. Acquiring data from the project team's activities
2. Analyzing that data to detect key performance indicators
3. Acting on the results in terms of coaching developers to improve their metrics

***JDJ:** Why "CQ^2"?*

NC: We put the finest marketing brains on the planet on to the task of coming up with a name for our new product, as you can tell. I guess you could say that "CQ" stands for "Code Quality"

"Software development is still considered an 'art,' and it's time to change that"

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

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

EXTENXLS4™
JAVA | XLS REPORTING TOOLKIT

extenTECH™
Call Us: 415-759-5292

and " ²" implies it's the second generation. But, at the end of the day, it's just a name!

JDJ: What are the main drivers behind the problem that Enerjy CQ² solves?

NC: Our industry is facing a credibility crisis. You'll remember that back in 2002, the NIST issued a report estimating the annual cost of software errors to the U.S. economy at $59.5bn. You might think that would be a wake-up call to the industry. Well, the interesting thing to me is that the Standish Group reports that in the two years following the NIST report, in other words from 2002 to 2004, software quality actually diminished! So, as an industry, it's time to take this problem seriously.

JDJ: How can it be that Java code quality, and software integrity in general, in these SoX-conscious times, has lagged so far behind as a corporate priority?

NC: We are still an immature industry. Look at virtually any other business process and you will find well-understood and documented standards and procedures for ensuring quality. The problem is no one yet really knows what KPIs to capture from a development team. Software development is still considered an "art," and it's time to change that, but that doesn't mean trying to shoehorn the software development process into a manufacturing discipline. There's a reason software is called software – it's supposed to be soft. In other words, it needs to be agile – to be organic, to the point

using Enerjy CQ² in preference to the other solutions "out there" – so that they in turn can convince their CTOs?

NC: There are two things that make this product truly unique. First, unlike some of our larger competitors, there is no need to retool the development team with Enerjy developer tools. Most developers have a carefully assembled set of tools from a variety of vendors, many of which may be open source. We say leave the developer alone – let them continue to use the same tools they are already using. Second, Enerjy CQ² is the only product that uses patent-pending technology to show development managers their quality metrics per developer. If a code quality tool tells you that you have a problem with a particular method, that may be useful, but there is no real call to action.

"Technology buyers are pickier than ever before – you really need to show a clear return on investment for your solution"

JDJ: Can you unpack "Precision Team Management" for us, why's that the key value proposition here?

NC: As I said earlier, it's a three-step process: Acquire, Analyze, Act. The key to implementation success for this process is that it has zero negative impact on the development process itself – in other words, the members of the development team continue to work exactly as they did before. It allows the manager to define what quality means in their organization – in terms of coding standards, unit test results, testing coverage percentages, and so on, and then to set thresholds that trigger alerts if those thresholds are crossed. The feedback loop occurs when the manager acts on those alerts, coaching the developers on how to improve their development hygiene and thereby raising the quality of the overall code base, as well as the overall performance of the team.

where as business processes change, the business software that helps to automate those processes changes too. As a result, we need processes such as Precision Team Management that will bring control to the environment, without being stifling.

JDJ: Is it ever possible to actually get to the point where code generates zero errors?

NC: Dijkstra famously pointed out that software testing cannot prove that there are no errors in your code; it can only prove that they exist. But that is not a reason not to do everything possible to squeeze bugs out as early in the development process, and monitor and measure those indicators of quality that we already have reasonably easy access to.

JDJ: Even though development managers are the ones who need Enerjy CQ², how does Enerjy go about making the necessary business case for

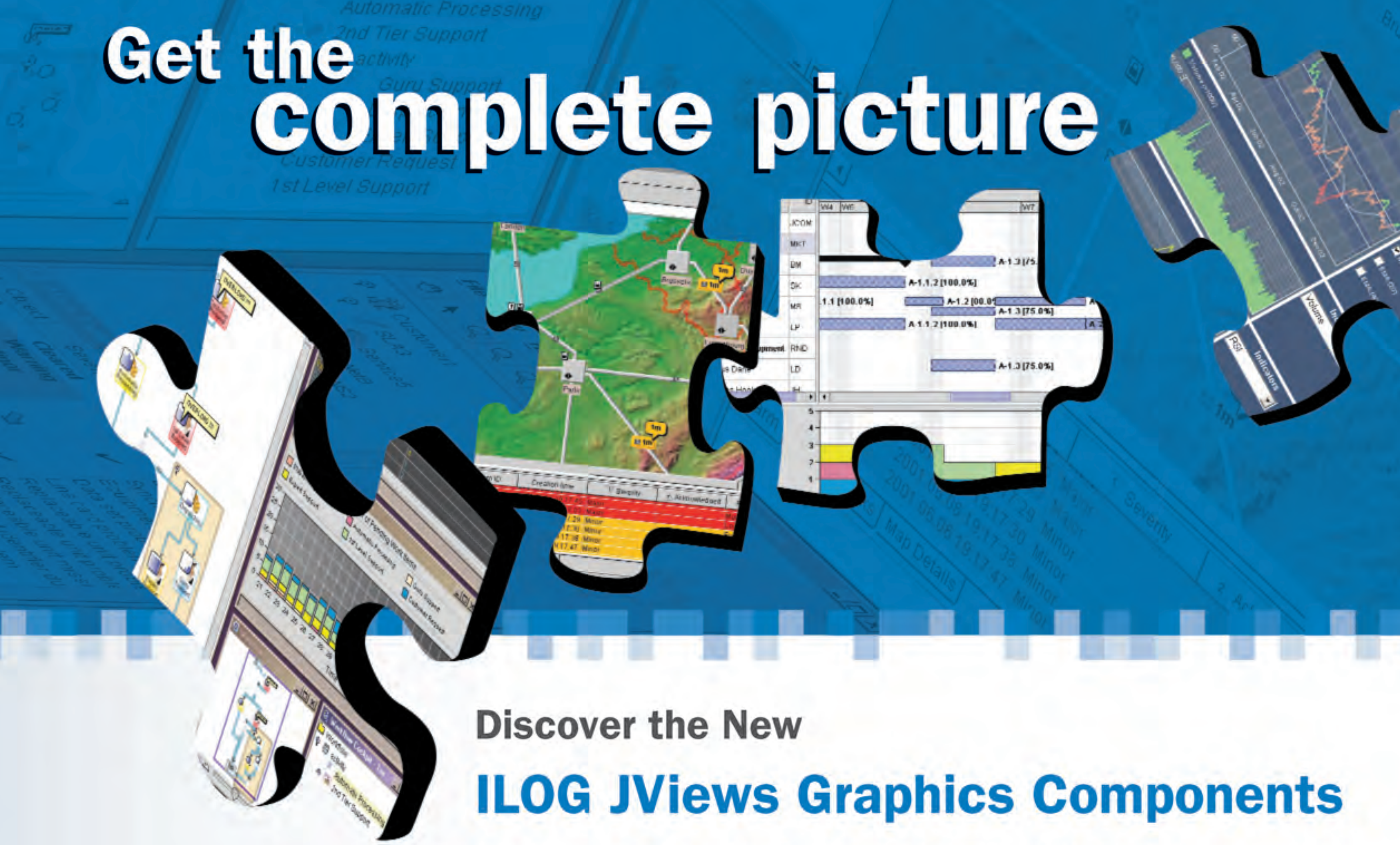
Enerjy CQ² will tell you how your individual team members are doing, and how they are trending over time. Now, if you care about the quality of your code base, that's powerful stuff!

JDJ: Enerjy is, as you say, Massachusetts based; are you getting a sense, from MA, of there being an overall "return" of technology? Is technology back in 2005, do you think?

NC: Yes, no doubt about it. But the other thing we're aware of is that technology buyers are pickier than ever before – you really need to show a clear return on investment for your solution. Any technology company with a mediocre offering will not last long in this market.

JDJ: Best of luck with Enerjy for the remaining quarter of 2005 and, of course, throughout 2006. We'll try and keep up with the speed of you guys! ;-)

NC: Thank you very much.



Discover the New

ILOG JViews Graphics Components

First comprehensive Java graphics toolkit for Eclipse
Full JavaServer Faces (JSF) Support

Build better GUIs in less time. Exceed your application requirements.

Reduce your development time & risk. Improve user experience & value.

Advanced BPM modeling and monitoring displays, data charts, Gantt scheduling & resource displays, network & equipment displays, network diagrams, EMS/NMS telecom displays, custom dashboards and more. Whatever your display needs – for desktop or Web client – JViews has the solution.

Learn more. Test-drive an Eval. Call: 1-800-FOR-ILOG or go to:

■ ILOG JViews Diagrammer	**http://diagrammer.ilog.com**
■ ILOG JViews Charts	**http://charts.ilog.com**
■ ILOG JViews Gantt	**http://gantt.ilog.com**
■ ILOG JViews Maps	**http://maps.ilog.com**
■ ILOG JViews TGO	**http://jtgo.ilog.com**

ILOG®

Changing the rules of business™

Rich Internet Components with JavaServer Faces

Providing the end user with a rich and responsive user interface

by Jonas Jacobi and John Fallows

JavaServer Faces (JSF) standardizes the server-side component model for Web application development but doesn't standardize the presentation layer at the browser. In a series of articles we are going to look at how JSF can fulfill new presentation requirements without sacrificing application developer productivity building Rich Internet Applications (RIA).

Consumer Requirements

It's always the end user who feels the effect of any chosen technology, especially at the presentation layer. The end-user experience should be top priority for any developer building applications, whether they are Web applications or desktop applications. End users demand feature-rich and highly interactive user interfaces.

Introduction to Rich Internet Technologies

Web application developers today are faced with higher demand for richer functionality using technologies such as HTML, CSS, JavaScript, and DOM. These technologies were not developed with enterprise applications in mind and, with increasing pressure from consumers to provide applications with features not fully described or supported by these technologies, developers are looking for alternative solutions or extensions to these standards.

From a combination of both consumer requirements and developer needs, a new breed of Web technologies has evolved. Often referred to as Rich Internet Technologies, these technologies enhance the traditionally static content provided by Web applications.

In the absence of sufficiently powerful standards, new technologies such as Mozilla's XUL, Microsoft's HTC, Java applets, Flex, and OpenLaszlo have emerged to meet the requirements. Such technologies support application-specific extensions to traditional HTML markup while still leveraging the benefits of deploying an application to a central HTTP server. XMLHTTP has also returned under a newly branded name, AJAX (Asynchronous JavaScript and XML). Applications built with these technologies are often referred to as Rich Internet Applications (RIA).

In this article series we are going to focus on AJAX and XUL to illustrate the potential that JSF brings to the JavaEE developer.

Jonas Jacobi is a principal product manager and evangelist for Oracle's Java/J2EE tool offering, JDeveloper, and over the past three years has been responsible for JavaServer Faces, Oracle ADF Faces, and Oracle ADF Faces Rich Client development features within Oracle JDeveloper. Jonas has been in the software business for 15 years. Prior to joining Oracle, he worked at several software companies in Europe, covering many roles including support, consulting, development, and project team leadership.

jonas.jacobi@oracle.com

AJAX

AJAX has gained momentum primarily due to the XMLHttpRequest browser object, which supports asynchronous communication with any business services used by the Web application. Popular sites such as Google Mail and Google Suggest use AJAX to deliver RIA.

With the *XMLHttpRequest* object, developers can now send requests to the Web server to retrieve only the data needed and use JavaScript to process the response. This ability to reduce the amount of data transferred between client and Web server reduces the bandwidth to a minimum and saves processing time on the server since most of the processing is done on the client using JavaScript.

It is important to note that the *XMLHttpRequest* object, although widely used, could at best be called a "de facto" standard, since most browsers, including Firefox, Inter-net Explorer, Opera, and Safari, support it. It is also worth noting that the *XMLHttpRequest* object is not exclusive to AJAX; potentially the *XMLHttpRequest* object can be used by any HTML/XML-based Web technology such as XUL or HTC.

Desktop vs Web

Traditional Web applications are in most cases slower than their counterpart desktop applications. With AJAX we can now send requests to the Web server to retrieve only the data needed and use JavaScript to process the response, which leads to a more responsive Web application.

A typical AJAX application leverages standard HTML/XHTML at the presentation layer and uses JavaScript to dynamically change the DOM. This creates an effect of "richness" in the user interface with no dependency on additional browser plug-ins.

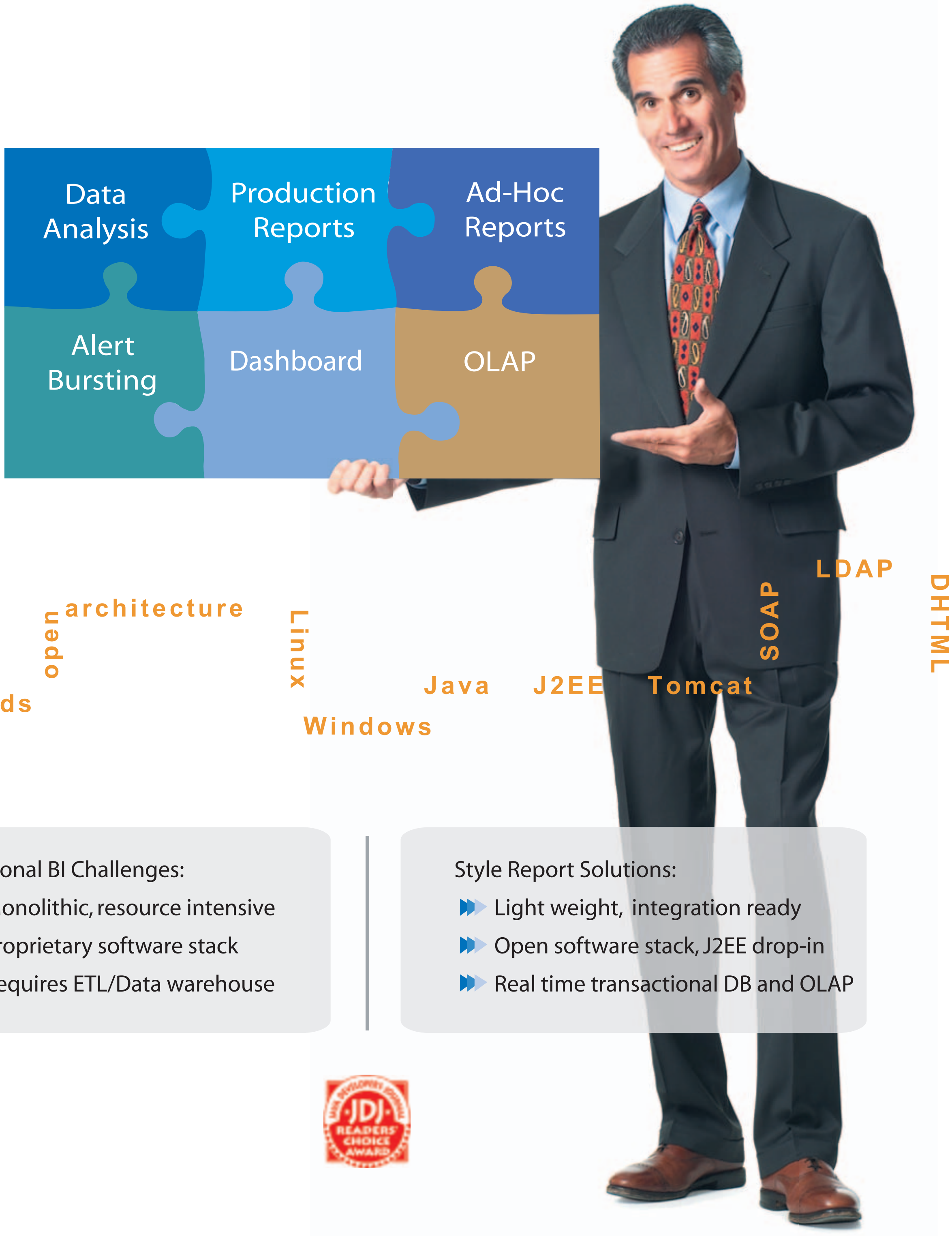
Mozilla XUL

XUL (pronounced zool) was created by the Mozilla organization (Mozilla.org) as an open source project in 1998. With XUL, developers can build rich user interfaces that may be deployed either as "thin client" Web applications, locally on a desktop, or as Internet-enabled "thick client" desktop applications. To fully provide the consumer with a rich user interfaces, XUL needs to execute

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

JDJ READERS' CHOICE AWARD

For more information and to download a free evaluation copy .... www.inetsoft.com/jdj in the Mozilla Gecko Runtime Environment (GRE). There are many examples of applications using XUL, including the Firefox browser, the Thunderbird e-mail client, and numerous plug-ins to each.

XUL Structure

The base idea behind XUL is to provide a markup for building user interfaces much like HTML, and to leverage technologies such as CSS for look-and-feel and JavaScript for events and behavior. There are even APIs available to access files systems and Web services over the network. As an XML language, XUL can be also be used in combination with other XML languages, such as XHTML and SVG.

XUL Components

XUL comes with a base set of components that are available through the Mozilla GRE, and has the added benefit of not needing to download components to display an application in the browser. You can also design your own components with XUL; these will need to be downloaded on request and cached in the browser.

XUL uses a language called XBL (eXtensible Binding Language) to define new components. XBL is also used to bridge the gap between XUL and HTML, making it easy to attach behavior to traditional HTML markup.

Listing 1 is a sample XUL file that embeds XUL elements and standard, name-spaced HTML elements.

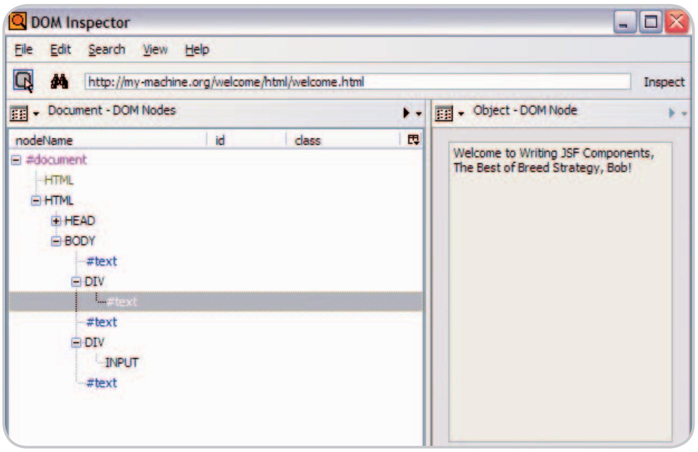
XUL Event Handling

Using XUL event handling is not that different from using HTML event handling. The GRE implementation supports DOM Level 2 (and partially DOM Level 3) and is virtually the same for HTML and XUL, and changes to the state and events are propagated through a range of DOM calls. XUL elements come with predefined event handlers, much like the event handlers provided with the standard HTML elements (see Listing 2).

As in HTML, developers can use JavaScript functions located in external libraries, or embedded in the page.

```
<script type="text/javascript"src="http://
www.bob.org/js/myscript.js">
```

There is a large set of built-in XUL components, each with its own set of event handler attributes. For example, the XUL window element delivers events such as load and paint.

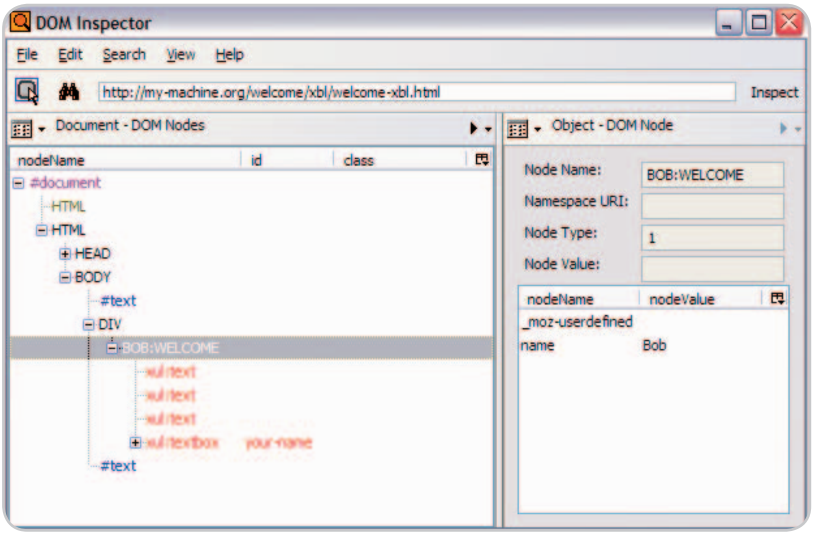


http://my-machine.org/welcome/html/welcome.html

```
#document
  HTML
  HTML
    HEAD
    BODY
      #text
      DIV
        #text
      #text
      DIV
        INPUT
      #text
```

Welcome to Writing JSF Components, The Best of Breed Strategy, Bob!

Figure 1 A Page's DOM tree without using XBL component



http://my-machine.org/welcome/xbl/welcome-xbl.html

```
#document
  HTML
  HTML
    HEAD
    BODY
      #text
      DIV
        BOB:WELCOME
          xul:text
          xul:text
          xul:text
          xul:textbox    your-name
      #text
```

Node Name: BOB:WELCOME
Namespace URI:
Node Type: 1
Node Value:

nodeName	nodeValue
_moz-userdefined	
name	Bob

Figure 2 A Page's DOM tree with an XBL component. The nodes highlighted red are not visible in the source

Creating Custom XUL Components Using XBL

XBL is an XML language that allows developers to extend XUL by adding "custom" components to the extensive set of existing XUL elements. In XUL, developers can define the look and feel using CSS, and they can define the behavior using XBL.

Developers can look at XUL as the "implementation" that comes with a pre-set of components, or tag libraries that can be used to build a user interface, much like the JSF Reference Implementation. XBL is the language developers use to extend XUL components and enable integration with HTML, in much the same way that Java is used to extend JSF components.

XBL Structure

An XBL file contains a set of bindings, each describing the behavior, properties, and methods of an XUL component.

A quick look at Listing 3 shows the root <bindings> element containing two binding elements. A <bindings> element can contain any number of <binding> elements. The namespace in the <bindings> element defines which syntax will be used, and in Listing 3 it is XBL – xmlns=http://www.mozilla.org/xbl. The file also contains a combination of XUL and HTML elements <xul:text/> and <html:input/>. This is extremely useful for simplifying development by encapsulating several components as a single, reusable component. The xbl:inherits attribute allows the text element to inherit values from the bound element by defining a variable called name and assigns it to the value attribute. If no value is defined when the component is used in a page, the text field's value will default to "Guest".

The ID attribute on the binding element (in Listing 3 "welcome" and "bye") identifies the binding. Using CSS, a developer can assign a binding to an element by setting the –moz-binding URI property to reference the binding inside the XBL document.

Note: The way custom behavior can be attached to an HTML element has been submitted as a proposal to the W3C organization – "A Modular Way of Defining Behavior for XML and HTML" (http://www.w3.org/TR/NOTE-AS) by Netscape Communications Corp.

```
bob\:welcome
{
  -moz-binding: url('bob-xbl.xml#welcome');
}
```

In the above sample, the CSS selector for the *bob:welcome* element has the *-moz-binding* set to point to an XBL file named *bob-xbl.xml* and refers to a specific binding with ID *welcome* in the XBL file. This is similar to how anchors are used in HTML files.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="bob.css" >
    <title>Mozilla XBL Example</title>
  </head>
  <body>
    <div><bob:welcome name="Bob" ></bob:welcome></div>
  </body>
</html>
```

In the above HTML sample the bob.css stylesheet has been attached to the HTML document, and there is one element <bob:welcome name="Bob"></bob:welcome> inserted in the page. One cool feature to using XBL's encapsulation of behavior is that it creates a document tree within the scope of the custom component that is separate from the main HTML page. This means that the XBL component markup is not "exploded" into the main document, losing encapsulation (see Figure 1).

The direct benefits of encapsulation are full control over both the behavior and the look-and-feel, but without exposing any internal implementation details (see Figure 2).

Adding Functionality to an XBL Binding

XBL can be used to add new fields, properties, and methods to XUL elements. A field is a simple container that can store a value that can be retrieved and set. A property is slightly more complex and is used to validate values stored in fields or values retrieved from XBL-defined element attributes. Since the property cannot hold a value, there is no way to set a value directly on a property without using the onset or onget handlers. By using these handlers, a developer can perform precalculation or validation of the value being retrieved or modified.

Methods are object functions such as window.open(); and allow developers to add custom functions to custom elements. In the XBL file, fields, properties, and methods are

John Fallows is a consulting member of technical staff for server technologies at Oracle Corporation, and has been working in distributed systems for over a decade. During the past five years, he has focused on designing, developing, and evolving Oracle ADF Faces, and is now lead developer for Oracle ADF Faces Rich Client.

john.fallows@gmail.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

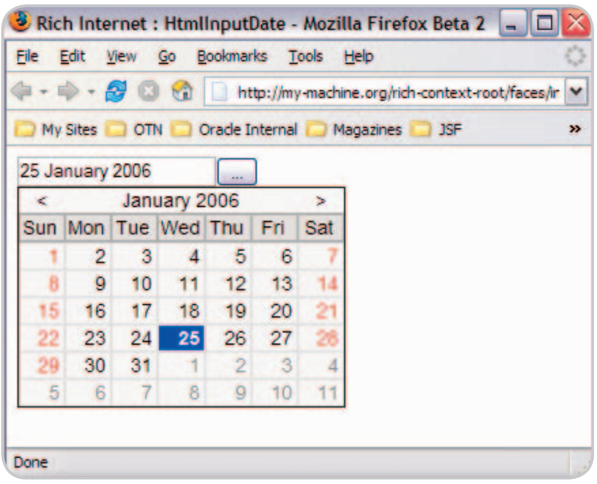
*Dynamic***PDF**™ Generator v3.0
Our popular and highly efficient class library for real time PDF creation is available natively for Java
- Intuitive object model
- Unicode and CJK font support
- Font embedding and subsetting
- PDF encryption ◆ 18 bar code symbologies
- Custom page element API ◆ HTML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


Rich Internet : HtmlInputDate - Mozilla Firefox Beta 2
File Edit View Go Bookmarks Tools Help
http://my-machine.org/rich-context-root/faces/ir
My Sites OTN Oracle Internal Magazines JSF
25 January 2006
January 2006
Done

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

Figure 3 Rich Internet Component – InputDate

defined inside an implementation element that is a child of the binding element (see Listing 4). (Listings 4–7 and additional source code can be downloaded from http://jdj.sys-con.com.)

Developers can use the getElementById function to first locate a custom element before accessing custom properties and methods.

In Listing 5, an HTML button has been added that triggers the onclick event handler. The onclick event handler will execute the assigned script – alert(document.getElementById('welcome').bob) – and display the value of the field called bob.

Handling Events in XBL

In XBL, developers can add event handlers directly to the individual XUL elements inside the content element, e.g., <xul:button label="Press me!" oncommand="alert('welcome')" />. Sometimes it's useful to add an event handler for all the child elements of the content element. In XBL, this can be done by adding a handler element. Each handler defines the action that will be taken for a particular event triggered by any of its content elements.

In XBL sample shown in Listing 6, a button has been added to the welcome binding and one handler has been added to capture all click events in the scope of the welcome binding.

Why Would JSF Be of Any Help?

Of the two technologies used in this article, only XUL is designed to support reusable components in Web applications. XUL allows for the encapsulation of HTML, CSS, and JavaScript into a single component that can be reused by the application developer. AJAX, on the other hand, solves the asynchronous communication to the server that can be used to provide end users with a very responsive UI.

These technologies solve most of the consumer requirements, but are still lacking in support for the application developer. We need a standard way to define Rich Internet Applications that are deployable over the Web without vendor lock-in.

The WHAT (Web Hypertext Application Technology) group is working on a standard tag library for HTML extensions that can work on any browser. Meanwhile, developers are falling back to the lowest common denominator – HTML – and using scripts to build dynamic Web applications, e.g., AJAX. However, this approach of developing Web applications has one severe drawback – there is no good reuse model or easy integration with existing server-side logic.

Listing 1

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window id="myWindow" title="Bob's search"
        xmlns:html="http://www.w3.w3.org/1999/xhtml"
        xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.
is.only.xul">
<html:p>
  Search for:
  <html:input id="find-text"/>
  <button id="bobOK" label="My Button"/>
</html:p>
</window>
```

Listing 2

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window id="myWindow" title="Bob's search"
        xmlns:html="http://www.w3.w3.org/1999/xhtml"
        xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.
is.only.xul">
<html:p>
  Search for:
  <html:input id="find-text"/>
  <button id="bobOK" label="My Button"
          oncommand="alert('This is Bob\'s label: ' +  event.tar-
get.label);
          return false;"/>
</html:p>
</window>
```

Listing 3

```
<?xml version="1.0"?>
<bindings xmlns="http://www.mozilla.org/xbl"
          xmlns:xbl="http://www.mozilla.org/xbl"
          xmlns:html="http://www.w3c.w3.org/xhtml/19991999/xhtml"
          xmlns:xul="http://www.mozilla.org/keymaster/gatekeeper/
there.is.only.xul">

 <binding id="welcome">
  <content>
   <xul:text value="Welcome to Writing JSF Components, " />
   <xul:text value="Guest" xbl:inherits="value=name" />
   <xul:text value="!" />
      <xul:textbox id="your-name"
                   value="Guest"
                   xbl:inherits="value=name" />
  </content>
 </binding>
<binding id="bye">
    <content>
      <xul:text value="Welcome back, "/>
      <xul:text value="Guest" xbl:inherits="value=name" />
      <xul:text value="!" />
      <html:input type="button" value="Press me!" />
    </content>
  </binding>
</bindings>
```

JavaServer Faces defines a standard component model to server-side Web application development. By integrating JavaServer Faces with XUL and AJAX, we can provide a highly interactive Rich Internet Component suite that also delivers a standard reuse model for Web application developers.

Why a Component Model?

Using components instead of markup is an effective model that allows application developers to build Web applications with prefabricated blocks of functionality without concern for implementation details. Over time, the component author can maintain the internal implementation without adversely impacting the application developer.

Cross-Platform Support

Cross-platform support is critically important for today's technologies, never mind what tomorrow might bring. Consumers are already requesting increased support for platforms such as handheld devices, telnet clients, and instant messaging.

In most cases, applications need to be designed to use features of a specific platform, which in turn is time-consuming and costly. As each new platform is added, the time and cost can increase significantly.

For the two technologies previously mentioned in this article, one claims that it has cross platform support – Mozilla's XUL. This is partially true; you can deploy a XUL application to any operating system that the Mozilla platform (GRE) supports.

You could also argue that AJAX provides cross-platform support, but it is the provider of the AJAX solution that needs to ensure that every browser-specific quirk is supported.

So, although you have plenty of environments to deploy to, there is no true solution that has full cross-platform support.

With Imagination as the Only Limit

JSF standardizes the server side for the application developer now, while we wait for the presentation layer at the browser to standardize for component developers.

JSF separates the user interface from the application, making it possible for the component author to change the presentation layer without tampering with the application. This is not solving the cross-platform issue or the extensibility of HTML, but it will help application developers build Rich Internet Applications in an unswervingly standard way.

A JSF component developer could take advantage of XUL's presentation layer and AJAX for communication, creating a highly interactive component. If the client does not support XUL, the application can dynamically fall back to a non-XUL solution. The application developer will still be able to build one application supporting multiple rendering technologies with one common programming model – JavaServer Faces.

A Page in JSF Supporting XUL and AJAX

The JSF sample shown in Listing 7 illustrates what a page developed with reusable JSF components leveraging XUL and AJAX as rendering technologies can look like.

Apart from the obvious namespaces, the sample contains one namespace that maps to a custom component library – xmlns:bobh=" http://www.bob.org/jsf/html" – and a custom component – <bobh:inputDate ..."/>.

The source of the page is not that different from what we have seen so far with XUL or AJAX, but the main difference is that the Web application developer will not need to learn two ways of supporting Rich Internet Applications in today's browsers.

Summary

This article provides some insight into two of the market's leading view technologies for Rich Internet Applications (RIA) – XUL and AJAX. These technologies have proven that they are more than capable of providing the end user with a highly rich and responsive user interface. We have also touched on the issues with these technologies such as platform support, being non-standards based, and maintenance.

Looking ahead, the potential for JavaServer Faces as a UI component technology is without boundaries. Component developers can provide the community with a wide range of components supporting technologies from HTML to XUL, wireless, and even character-based solutions with the imagination as the only limit.

In future articles, we are going to discuss how to build reusable JSF components that leverage AJAX and XUL.

This article is based on, and contains excerpts from, the book *Pro JSF: Building Rich Internet Components* by Jonas Jacobi and John Fallows, to be published by Apress in January 2006.

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™

Report Writing Simplified

Take advantage of MS Word's® powerful layout tools to build complex reports

by Windward Studios, Inc.

Report writing creates an expensive bottleneck for many companies because it demands ongoing, iterative involvement from in-house developers. This article describes a novel approach to report writing which allows business users to layout elegant reports using Microsoft Word® as the layout tool.

Introduction

Report writing is costly and time consuming. The people who need the reports – business users – generally can't create them without the help of a programmer or developer. Creating reports becomes an iterative process. The business user enlists the developer's help. The developer creates a report and shows it to the business user. With feedback from the user, the developer then refines the report. This process happens for reports across the organization.

IT and business resources get tied up in this time-consuming process. Plus, report writing is often the lowest priority for developers because they are under pressure to deliver revenue-generating products.

This endless report design process is very slow, expensive and frustrating for all departments involved. Worse still, the final report created is inadequate -- getting it right becomes too expensive and takes too much time.

What if report design were easy enough that program managers, product managers, accounting managers or any business user could do it alone, without help? What if there were no ad hoc query tools or complex report writers that require programming knowledge? What if there was no learning curve?

Simplified Report Writing

Microsoft Word (or any other word processor) is the key to a new and innovative approach to report writing called Windward Reports™. Microsoft Word is a powerful layout tool that almost everyone knows how to use. Setting up tables and formatting text in Word is almost second nature to most business people. Leveraging the familiar Word environment, business users have an easy-to-use tool to create the sophisticated reports they really want.

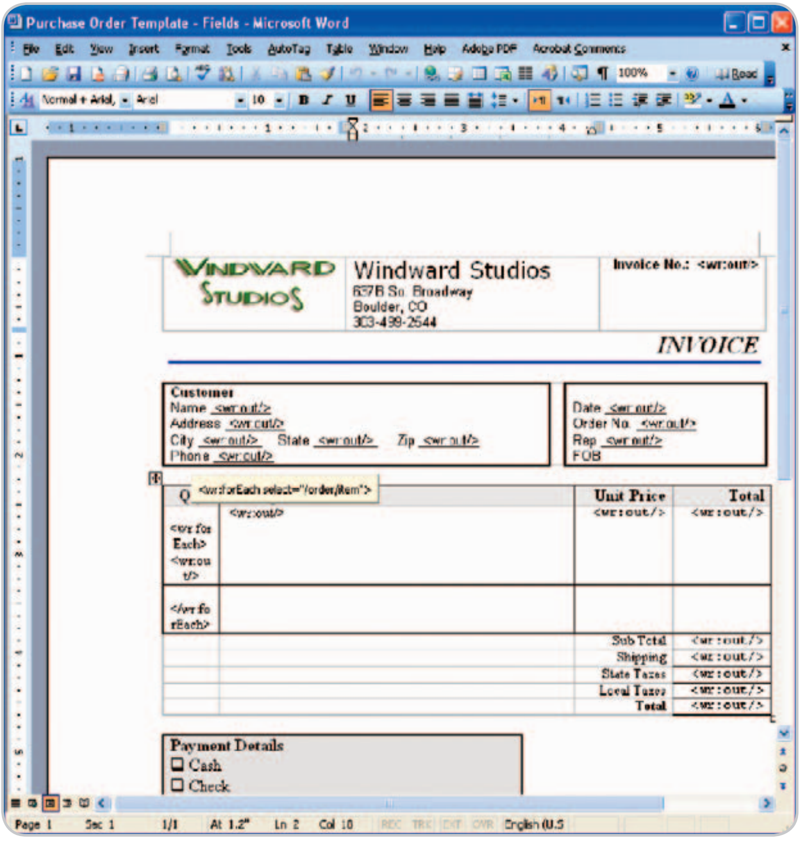
Business users can take full advantage of Word's formatting and What-You-See-Is-What-You-Get environment. There's little or no learning curve.

Everything you would do to create a similar report template you already know how to do.

Windward Reports is an enterprise-level reporting engine that uses Microsoft Word, or any other word processor such as Open Office, to design a report template. Windward Reports takes the template you design and any number of data sources (such as xml or sql databases) and merges them together to create your final report.

Using Windward Reports

Windward Reports can be called by java or .NET programs, either directly or in a client server configuration.


Purchase Order Template - Fields - Microsoft Word
File Edit View Insert Format Tools AutoTag Table Window Help Adobe PDF Acrobat Comments
Normal + Arial, Arial 10
Page 1 Sec 1 1/1 At 1.2" Ln 2 Col 10 REC TRK EXT OVR English (U.S

Windward Studios
637B So. Broadway
Boulder, CO
303-439-2544

Invoice No.: <wr:out/>

INVOICE

Customer
Name <wr:out/>
Address <wr:out/>
City <wr:out/> State <wr:out/> Zip <wr:out/>
Phone <wr:out/>

Date <wr:out/>
Order No. <wr:out/>
Rep <wr:out/>
FOB

<wr:forEach select="/order/item">

Q		Unit Price	Total
<wr:forEach> <wr:out/>	<wr:out/>	<wr:out/>	<wr:out/>
</wr:forEach>			
		Sub Total	<wr:out/>
		Shipping	<wr:out/>
		State Taxes	<wr:out/>
		Local Taxes	<wr:out/>
		Total	<wr:out/>

Payment Details
☐ Cash
☐ Check

Figure 1 Purchase Order Template

Windward Reports includes AutoTag™, an add-in for Microsoft Word which greatly simplifies data layout and data source mapping, eliminating the expense and frustration of the iterative report development process. Because the report is designed in Word, business users can design it correctly and quickly. Figure 1 shows a Purchase Order Template that was created in Word.

Tags are added to the template to indicate where data will be merged into the template to create the final report. The simplest tag is the out tag, whose format is <wr:out select='/root/name'/>. In this case, the tag will be replaced with the name of the customer from the xml data merged with the template. While the entire tag can appear in the template, AutoTag displays just the tag name. Placing the cursor over the tag name pops up a tooltip that displays the entire tag.

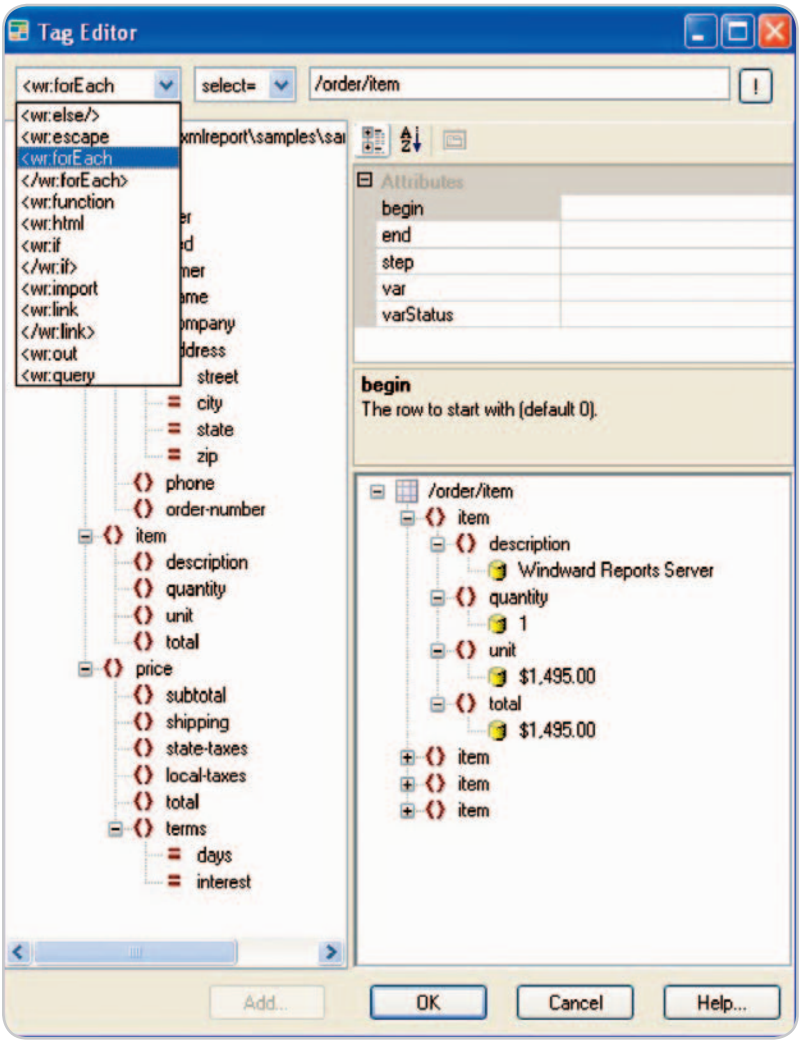
In addition, when using AutoTag, there is no mistyping of tags in the template. A user selects "Insert Tag" from the drop-down menu and a pop-up box displays the available tags. A business user can't get the tag format wrong (see Figure 2).

The Tag Editor makes it easy for a non developer to:

- See the data nodes and select the one you want, easily setting the tag to point to the correct data. There is no need to understand how to access a database.
- Displays the data that the tag will return. This uses a sample dataset and delivers immediate visual feedback of the data that will be inserted.

While there are 9 tags, most work is performed with two tags: the out tag (described above), and the forEach tag. The forEach tag will loop through a set of data, once for each row of data. In the screenshot above, the forEach will loop through 4 times, once for each item in the invoice. In the template displayed in Figure 3, it will create one row in the table for each of the 4 items purchased.

Once the template is complete and the desired report output format is chosen (PDF, RTF, HTML, WordML, XLS, SpreadsheetML, TXT, or multipart MIME email), Windward Reports merges the template with the corresponding report data. The result is a finished report with all the formatting and positioning specified by the business user - available in minutes.



begin
The row to start with (default 0).

Figure 2 Tag Editor

Proven Efficiency

Windward Reports' customers attest to this ease of use and time savings:

Mr. S.S. Mohanty of Nucleus Software Exports, says his favorite feature of Windward Reports is the Word based template because it made his life easier. Prior to using Windward Reports, he said *"I [previously] spent half my time designing and scripting reports to meet the Bank's requirements. With Windward, we can layout a report in Word and get immediate approval from the client."*

Mr. Bjerregaard Pedersen of Backbone Digital Systems said *"compared to systems where both layout and data must be coded, as we did before, the process of creating a new report is at least 5 times faster."*

More Information

This article is brought to you by Windward Reports, a robust J2EE/.NET reporting engine that uses Microsoft Word as a layout tool, putting report-design and generation power in the hands of the business user, technical or non-technical. Windward Reports can run as a standalone application or as an enterprise-wide service running on a corporate server. To see if Windward Reports is right for your organization, download a free demo at http://www.windwardreports.com or call 303-499-2544.

WindwardReports
advanced J2EE/.NET report generation

Java GoF Behavioral Design Patterns

Solutions for object interactions

by Puneet Sangal

While creational patterns decouple a client from the objects it instantiates, behavioral patterns dictate the object interaction. If not carefully planned, coupling and cohesion can pose major design issues. My previous article, "Java GoF Creational Design Patterns" (***JDJ***, Vol. 9, issue 9), discussed creational patterns. This article will focus on behavioral patterns.

Knowledge of the behavioral pattern vocabulary will help designers come up with appropriate solutions for object interactions, minimizing or eliminating these issues. One basic thing to understand is that there are two kinds of patterns:

- ***Class:*** Describes compile-time relationship between classes via inheritance
- ***Object:*** Describes the runtime relationship between classes via composition

The patterns elicited in this article have been classified as either object or class based. Generally, object patterns allow more flexibility in design.

When working with design patterns, it's important to know not just how but also when to use them. A design pattern should occur naturally as a part of a design and shouldn't drive the design, but guide it. Overuse of any pattern is harmful, especially in a situation where you're biased, desperately trying to fit a pattern into the design. If you look at the Golden Hammer anti-pattern (an anti-pattern is the reverse of a design pattern), it states just that: when you are already well versed with a particular pattern, and overuse a technology or pattern, then learn or implement a more befitting technology/pattern to the domain/problem in question. It is prudent to look at the domain and problem context first and then use patterns to help in the design process.

At a bare minimum, you should strive for a loosely coupled design and program to an interface rather than an implementation. This leads to a good design without forcing you to apply design patterns in a problem context. Patterns are meant to aid designers, not compel the designers to use them.

It's also not a good idea to override an implemented method of a base class or derive from a concrete class. If you also adhere to not having a variable holding a reference to a concrete class, together this qualifies the dependency (or inversion) principal that can also be used as a guideline for design.

Puneet Sangal is a Technical Architect for a software services company. He has been working with Java technology for seven years.

pmsangal@gmail.com

In accordance with my earlier article on creational patterns, I'll attempt to keep this one terse as well, simultaneously keeping ease of understanding as the denominator.

Table 1 lists the behavioral patterns with their intent. Our next step will be to validate each one with a laconic, yet usable template. Each pattern will have a standard definition, template, and description attached to it. I will then attach a real-world example in which the respective pattern shall be applicable. Together, these elements will help you understand the patterns from the perspective of a software manager, architect, designer, or developer.

Behavioral Design Patterns	Behavioral Pattern	Class-based or Object- based
	Strategy	Object
	Visitor	Object
	Template	Class
	Iterator	Object
	Command	Object
	Memento	Object
	Interpreter	Class
	Observer	Object
	Chain of Responsibility	Object
	State	Object
	Mediator	Object

Table 1 The GoF Behavioral Design Patterns

Strategy Pattern

This pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. It allows algorithms to vary independently of the clients that use it.

The Strategy pattern is widely used and suggests programming to an interface over implementation. It allows building dynamic software that can be changed with minimal impact on existing code.

Let's look at the template below:

1. The first step is to define an interface Strategy. This defines a method that each ConcreteStrategy shall implement (there can be more than one ConcreteStrategy).
2. The Context class encapsulates the Strategy interface.
3. Now this is ready to be used. A client can create a Context object and dynamically assign the ConcreteStrategy at runtime.

Strategy is an object pattern. If this solution were to be approached via inheritance (as in a class pattern), it would not allow the flexibility to dynamically bind ConcreteStrategy to Context.

Template

```
package strategy;

public interface Strategy {

 /**
  * This method declaration must be implemented by the
ConcreteStrategy implementations.
  */
 public void algorithmInterface();
}
```

```
package strategy;

public class Context {

 /** stores the Strategy instance of the Context */
 private final Strategy fStrategy;

 /**
  * Constructor
  */
 public Context(Strategy strategy) {
   super();
   fStrategy = strategy;
 }

 /**
  * This method invokes the algorithm interface of the current
Strategy
  */
 public void contextInterface() {
   fStrategy.algorithmInterface();
 }
}
```

```
package strategy;

public class ConcreteStrategy implements Strategy {

 public ConcreteStrategy() {
   super();
 }

 /**
  * This method implements the algorithm operation defined by
the Strategy interface.
  */
 public void algorithmInterface() {
   // Write your algorithm code here ...
 }

}
```

Example

Imagine a scenario in which a company is interviewing for different positions. The recruiter wants to schedule the interviews with the flexibility to change the position that a candidate might interview for. To apply the above template:

```
Strategy = Position
Context = Candidate
ConcreteStrategy1 = DotNET
ConcreteStrategy2 = J2EE
ConcreteStrategy3 = CPlusPlus
```

Mediator Pattern

The Mediator pattern defines an object that encapsulates how objects interact. It promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently. This centralizes the logic and decouples the objects supported by the Mediator from the rest of the classes. To reduce the coupling, the Mediator class provides detailed information about the methods of other classes.

Beware while using this pattern because if not used properly, it can make the system quite complicated and, as a result, hard to maintain.

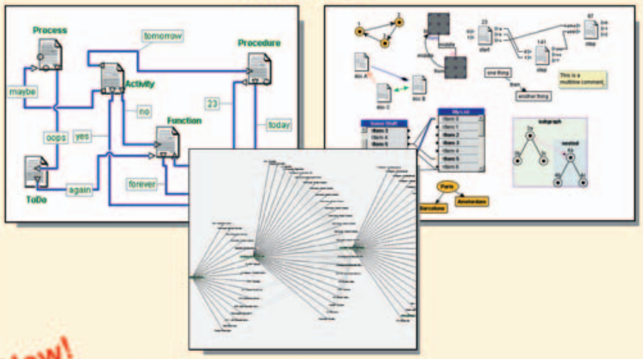
A Mediator acts akin to the Observer pattern. It observes the changes in the Colleague classes.

Template

```
 package mediator;

public interface Mediator {

 /**
  * This method creates all colleagues instances.
  */
 public void createColleagues();

 /**
  * Each associated colleague have to call this method if its
properties of
```

-continued on page 46

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

-continued from page 45

```
* interest are changed.
  */
 public void colleagueChanged(Colleague colleague);

}
```

```
package mediator;

 /** stores the associated Mediator */
 private final Mediator fMediator;

 /**
  * Constructor
  */
 public Colleague(Mediator mediator) {
   super();
   fMediator = mediator;
 }

 /**
  * Call this method from derived classes if a property of interest
has been changed.
  */
 protected void changed() {
   fMediator.colleagueChanged(this);
 }

}
```

```
package mediator;

public class ConcreteColleague extends Colleague {

 /**
  * Constructor
  */
 public ConcreteColleague(Mediator mediator) {
   // call inherited constructor
   super(mediator);
 }

}
```

```
package mediator;

public class ConcreteMediator implements Mediator {

 /**
  * This method creates all colleagues instances.
  */
 public void createColleagues() {
 }

 /**
  * Each associated colleague have to call this method if its prop-
erties of
  * interest are changed.
  */
 public void colleagueChanged(Colleague colleague) {
   //  Inform all other colleagues that one colleague has changed.
 }
}
```

Example

Think of an e-mail system. In order for any person to e-mail anyone else in the e-mail system, every person doesn't need to store everyone else's e-mail IDs. Rather, an LDAP directory can act as the mediator and provide information about the e-mail addresses to any person to e-mail any other person. If any person leaves the company or the e-mail ID changes, the LDAP directory can easily synchronize it.

Another example: the control tower at a controlled airport demonstrates this pattern well. The pilots requesting to land or depart communicate with the tower, rather than explicitly communicating with one another. The tower is the mediator here, imposing any constraints in the airport and not controlling a particular flight or pilot.

Visitor Pattern

This pattern comes in handy when you have a number of classes implementing different interfaces. Visitor lets you define a new operation without changing the classes of the elements on which it operates. In addition, if an operation needs to be performed on these classes, Visitors can be used.

Visitor is one of the complex patterns and needs to be designed carefully. Although Visitor is an object pattern, it should be used when encapsulation is not deemed more important and you want to add capabilities to a composite of objects.

Template

```
public interface Visitee {
      public void acceptVisitor(Visitor v);
   }
```

```
public abstract class Visitor {
      public void VisitObjects(Enumeration visitedItems) {
         while (visitedItems.hasMoreElements()) {
            // The visitor object asks the visitee to accept this
visitor
            Visitee obj = (Visitee) visitedItems.nextElement();
            obj.acceptVisitor(this);
         }
      }

      public abstract void visitTypeA(Visitee obj);
      public abstract void visitTypeB(Visitee obj);
      public abstract void visitTypeC(Visitee obj);
        // ... and so on, for each type of object to visit
   }
```

```
// Now some concrete visitor classes for various operations to
   // perform on the structure, e.g., "Print", "Display",
"Compute",
   // "Save", "Clear" ...

   public class ConcreteVisitorA extends Visitor {
      public void VisitTypeA(Visitee obj) {
         //  ... do what needs to be done for a TypeA object
         //  for this type of visit-operation
         obj.doSomething();
      }
      public void VisitTypeB(Visitee obj) {
         //  ... do what needs to be done for a TypeA object
         //  for this type of visit-operation
         obj.doSomethingElse();
      }
        // ... and so on, for each type of object to visit
   }
```

```
public class ConcreteVisitorB extends Visitor {
      // ... same stuff as above, but for a different operation,
      // e.g., "Print", "Display", "Compute", "Save"
   }
```

```
// Now for the types of objects in the structure that we need to
visit

   public class TypeA implements Visitee {
      public void acceptVisitor(Visitor v) {
          v.VisitTypeA(this);
      }
```

```
        // ...
    }

    public class TypeB implements Visitee {
        public void acceptVisitor(Visitor v) {
            v.VisitTypeB(this);
        }
        // ...
    }
```

Example

This pattern can be observed in the operation of a cab service. When a customer (Visitor) needs a cab, he or she calls for a cab. The cab company adds this customer to its existing list and dispatches the taxi. From then on, the driver is in control of the transportation and not the customer. The customer can only dictate where he or she wants to be transported to.

Template Pattern

In this pattern, the subclasses decide how to implement steps of an algorithm. Once a skeleton of an algorithm is defined in the form of steps, it might be useful to defer some of these steps to the subclasses. The subclasses can provide their own implementation of these steps. For the client, it is still the same method call but, based on the subclass object created, the steps for that subclass are invoked, without changing the algorithm structure.

A template is provided below. When working with a template pattern, sometimes it's better to keep the abstract methods small in number, otherwise it might be a big job to implement them in the subclass.

Template

```
package template;

public abstract class AbstractClass {

 /**
  * Default constuctor
  */
 public AbstractClass() {
  super();
 }

 /**
  * This method defines a skeleton of an algorithm, delegating some
steps
  * to primitive operations to its subclass implementation.
  */
 public void templateMethod() {
  // Customize this method based on your application needs.
  primitiveOperation1();
  primitiveOperation2();
 }

 /**
  * A ConcreteClass must implement this abstract method. It is used
  * by the templateMethod to proceed the algorithm.
  */
 protected abstract void primitiveOperation1();

 /**
  * A ConcreteClass must implement this abstract method. It is used
  * by the templateMethod to proceed the algorithm.
  */
 protected abstract void primitiveOperation2();

}
```

```
package template;

public class ConcreteClass extends AbstractClass {

 /**
  * Default constuctor
  */
 public ConcreteClass() {
  super();
 }

 protected void primitiveOperation1() {
  //  Implement this primitive operation to perform a step in the algo-
rithm defined in the AbstractClass.
 }

 protected void primitiveOperation2() {
  //  Implement this primitive operation to perform a step in the algo-
rithm defined in the AbstractClass.
 }

}
```

Example

Imagine database-driven application where you have to write different select queries. If there is a QuerySelect abstract class, it can define an execute() method. This execute method has some common steps and one step called executeSelect() that needs to be implemented by each subclass of the QuerySelect class. The executeSelect() is also defined as an abstract method inside the QuerySelect class.

-continued on page 48

SeeJava

Have you ever wanted to "see" what's going on inside your Java servers and applications?

Now you can.

Monitor and troubleshoot your Java applications and servers in real-time with SeeJava.

Key Features:

* View metrics for request times, queries, throughput, memory utilization, server uptime, and other critical data
* Attach debugging output to pages
* Log metrics to a database for in-depth analysis
* Stop long-running/hung requests
* Easily monitor multiple servers at once
* Connect to notification systems via XML bridge
* And more!

www.seejava.com

SeeJava is a product of Webapper Services, LLC - Development, Consulting, Products
www.webapper.com

-continued from page 47

To write a select query, extend the QuerySelect class and implement the executeSelect() method. When the client needs to execute this query, all it has to do is:

```
QuerySelect querySelect = new SampleQuerySelect();
querySelect.execute();
```

where SampleQuerySelect extends the QuerySelect class. Refer to the above template to visualize how these classes fit in.

Iterator Pattern

Iterator provides a common interface for traversing an aggregation of objects, without exposing the implementation. This addresses the need to abstract the traversal of different data structures transparently.

The Iterator takes the responsibility of traversal and not the aggregate. This simplifies the aggregate interface and its implementation, as seen below in the template.

Template

```
package iterator;

public interface Iterator {

 /**
  * Moves the item pointer to the first item in the collection
  */
 public void first();

 /**
  * Moves the item pointer to the next item in the collection
  */
 public void next();

 /**
  * This method returns false if are still items in the collection
to
  * iterate through, otherwise true.
  */
 public boolean isDone();

 /**
  * This method returns the item at current position.
  */
 public Object currentItem();
}
```

```
package iterator;

public interface Aggregate {

 /**
  * This method creates and returns a ConcreteAggregate instance.
  */
 public Iterator createIterator();

 /**
  * This method returns the count of items.
  */
 public int count();

 /**
  * This method appends a new item to the collection of items.
  */
 public void append(Object item);

 /**
  * This method removes an item from the collection of items.
  */
 public void remove(Object item);
}
```

```
package iterator;

public class ConcreteIterator implements Iterator {

 /** This field stores the reference to the ConcreteAggregate
instance. */
 private final ConcreteAggregate fListAggregate;

 /** This field stores the current item index. */
 private int fIndex = 0;

 /**
  * Constructor
  */
 public ConcreteIterator(ConcreteAggregate aggregate) {
  super();
  fListAggregate = aggregate;
 }

 /**
  * Moves the item pointer to the first item in the collection
  */
 public void first() {
  fIndex = 0;
 }

 /**
  * Moves the item pointer to the next item in the collection
  */
 public void next() {
  fIndex++;
 }

 /**
  * This method returns false if are still items in the collection
to
  * iterate through, otherwise true.
  */
 public boolean isDone() {
  return fIndex == fListAggregate.count();
 }

 /**
  * This method returns the item at current position.
  */
 public Object currentItem() {
  return fListAggregate.getList().get(fIndex);
 }
}
```

```
package iterator;

import java.util.ArrayList;
import java.util.List;

public class ConcreteAggregate implements Aggregate {

 /** This list stores references to all item instances. */
 private ArrayList fItemList = new ArrayList();

 /**
  * This method creates and returns a ConcreteAggregate instance.
  */
 public Iterator createIterator() {
  return new ConcreteIterator(this);
 }

 /**
  * This method returns the count of items.
  */
 public int count() {
  return fItemList.size();
 }

 /**
  * This method appends a new item to the collection of items.
  */
 public void append(Object item) {
  fItemList.add(item);
 }

 /**
  * This method removes an item from the collection of items.
  */
 public void remove(Object item) {
  fItemList.remove(item);
 }

 /**
  * This method returns the internal collection implementation.
  */
 public List getList() {
  return fItemList;
 }

}
```

Example

This is a very common pattern and is also provided in the standard Java API. The template provided can be used as is or customized per your needs. Some of the standard collection classes may use the Factory Method pattern to determine what kind of Iterator to instantiate.

Command Pattern

When using the Command pattern, an object encapsulates a request to execute a method in another object, giving the client the ability to make requests without knowing anything about the actual action that will be performed. The Command pattern also allows changing that action without requiring any modification in the client.

Basically, a Command object encapsulates a request (set of actions) on a set of receivers. To achieve this, it packages actions and the receiver into an object that exposes just one method, execute(). When called, execute() causes the actions to be invoked on the receiver.

This pattern is commonly used for operations like logging and undoing.

Template

```
package command;

public interface Command {

 /**
  * This abstract method must be implemented by the ConcreteCommand
implementation.
  */
 public void execute();

}
package command;

public class Receiver {

 /**
  * This construtor creates a Receiver instance.
  */
 public Receiver() {
  super();
 }

 /**
  * This method performs an action.
  */
 public void action() {
  //  Write your action code here ...
 }
}
```

```
package command;

public class ConcreteCommand implements Command {

 /** stores the Receiver instance of the ConcreteCommand */
 private final Receiver fReceiver;

 /**
  * Constructor
  */
 public ConcreteCommand(Receiver receiver) {
  super();
  fReceiver = receiver;
 }

 /**
  * This method executes the command by invoking the corresponding
  * method of the Receiver instance.
  */
 public void execute() {
  fReceiver.action();
 }
}
```

```
package command;

public class Invoker {

 /** stores the Command instance of the Invoker */
 private Command fCommand;

 /**
  * Default constructor
  */
 public Invoker() {
  super();
 }

 /**
  * Constructor
  */
 public Invoker(Command cmd) {
  super();
  fCommand = cmd;
 }

 /**
  * This method stores a ConcreteCommand instance.
  */
 public void storeCommand(Command cmd) {
  fCommand = cmd;
 }

 /**
  * This method performs the actions associated with the
ConcreteCommand
  * instance.
  */
 public void execute() {
  fCommand.execute();
 }
}
```

-continued on page 50

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

-continued from page 49

```
package command;

public class Client {

 /** stores the Receiver instance of the Client */
 private final Receiver fReceiver;

 /**
  * This construtor creates a Client instance and stores the given
Receiver.
  */
 public Client(Receiver receiver) {
  super();
  fReceiver = receiver;
 }

 /**
  * This method creates a ConcreteCommand instance and specifies a
Receiver object.
  */
 public void initConcreteCommand() {
  ConcreteCommand cmd = new ConcreteCommand(fReceiver);
  Invoker invoker = new Invoker();
  invoker.storeCommand(cmd);
 }

}
```

Example

A customer (Client) walks into a restaurant, looks at the menu, and places an order (Command) with the waiter (Invoker). Now the cook (Receiver) picks up the order and prepares the dishes. You may use the template provided as a reference to start with this pattern.

Also, continuing with our previous example in the Template pattern, let's define a base Query class. This has three subclasses called SelectQuery, InsertQuery, and UpdateQuery. Each of these can also be treated as commands.

Memento Pattern

Memento allows an object to store the state of another object, restoring the object's internal state at a later point. Video games frequently implement this pattern.

This essentially implies creating a snapshot, as in a relational database. At a later time, when the originator needs to recall a previous state, it asks the caretaker to return the previously stored memento.

A drawback to using this pattern is that saving and restoring state can be time-consuming.

Template

```
package memento;

public interface State {
}
```

```
package memento;

public class Memento {

 /** stores the State instance of the Memento */
 private State fState;

 /**
  * Default constructor
  */
 public Memento() {
  super();
 }

 /**
  * Constructor
  */
 public Memento(State state) {
  super();
  fState = state;
 }

 /**
  * This method stores a State instance.
  */
 public void setState(State state) {
  fState = state;
 }

 /**
  * This method returns the associated State instance.
  */
 public State getState() {
  return fState;
 }
}
```

```
package memento;

public class Originator {

 /** stores the internal state of the Originator */
 private State fState;

 /**
  * Default constructor
  */
 public Originator() {
  super();
 }

 /**
  * Constructor
  */
 public Originator(State state) {
  super();
  fState = state;
 }

 /**
  * This method creates and returns a Memento object containing a
snapshot
  * of the Originator instance's state.
  */
 public Memento createMemento() {
  return new Memento(fState);
 }

 /**
  * This method passes back a Memento object to an Originator
instance. The
  * Memento object must be used to restore an earlier Originator
state.
  */
 public void setMemento(Memento memento) {
  fState = memento.getState();
 }
}
```

```
package memento;

import java.util.Stack;

public class Caretaker {

 /** Stores a list of history states. */
 private final Stack fHistoryStack = new Stack();

 /** Stores the associated Originator instance. */
 private final Originator fOriginator;

 /**
  * Default constructor
  */
 public Caretaker(Originator originator) {
  super();
  fOriginator = originator;
 }

 /**
  * This method backups the current Originator state and stores it
in a
  * history stack. This snapshot can be used to restore the current
state
```

```
   * later.
   */
  public void backupState() {
    fHistoryStack.push(fOriginator.createMemento());
  }

  /**
   * This method restores a previous Originator state.
   */
  public boolean restoreState() {
    // check whether snapshots available
    if (!fHistoryStack.empty()) {
      Memento memento = (Memento) fHistoryStack.pop();
      fOriginator.setMemento(memento);
      return true;
    } else {
      return false;
    } // if - else
  }

}
```

Example

The serialization process in Java exemplifies the idea that is exhibited in this pattern. Serializable interface can store as well as restore the state of an object through the process of serialization and de-serialization.

Interpreter Pattern

Given a language, Interpreter defines a representation for its grammar, along with the facility to interpret sentences in the language.

The Interpreter pattern can be visualized in the following steps:

- Define a domain language (i.e., problem characterization) as simple language grammar
- Represent domain rules as language sentences
- Interpret these sentences to solve the problem

A class is used to represent each grammar rule. Since grammars are usually hierarchical in structure and owing to the class pattern nature of Interpreter, the inheritance hierarchy of rule classes maps fits aptly.

In the template shown below, an abstract base class specifies the method interpret(). Each concrete subclass implements interpret() by accepting the current state of the language stream and adding its contribution to the problem-solving process.

Template

```
package interpreter;

public interface AbstractRule {

  public void interpret(Context context) {
  }

}
package interpreter;

public class Terminal Implements AbstractRule {

  public void interpret(Context context) {
  }

}
package interpreter;

public class CompositeRule Implements AbstractRule {

  public void interpret(Context context) {
  }

}
```

It's outside the scope of this article to come up with a usable template for Interpreter. The above template should only be used as a reference.

Interpreter is a combination of grammatical expressions, sequences, commands, repetitions, and variables. Although the Interpreter provides the ability to extend the grammar, it is only wise to use when the grammar in question is relatively simple, otherwise Interpreter can become complex to maintain.

Example

The Interpreter pattern is particularly useful in scripting languages. Rhino (used in JavaScript) and Jelly (used in Maven) provide for scripting needs outside of Java.

Observer Pattern

Publisher + Subscriber = Observer Pattern. You already know this pattern, if you subscribe to and read newspapers.

Observer notifies objects in the event of a state change. It defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

The subject is the keeper of the business rules. The "view" functionality is delegated to distinct Observer objects. Observers register themselves with the Subject and when a Subject changes, it broadcasts to all registered Observers that it has changed. An Observer can then query the Subject for the change that has occurred.

The Observer pattern provides the flexibility to configure the number and type of "view" objects to be configured dynamically, instead of being statically specified at compile-time.

The difference between Mediator and Observer is that the latter distributes communication by introducing "observer" and "subject" objects, whereas a Mediator object encapsulates the communication between other objects.

Template

```
package observer;

public interface Observer {

 /**
  * This method requests the Observer object to update itself to
  * reconcile its state with that of the Subject object.
  */
 public void update();

}
package observer;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Subject {

 /** This list stores references to all Observer instances. */
 private List fObserverList = new ArrayList();

 /**
  * This method attaches an Observer instance to the Subject
instance.
  */
 public void attach(Observer observer) {
   fObserverList.add(observer);
 }

 /**
  * This method detaches an Observer instance from the Subject
instance.
  */
 public void detach(Observer observer) {
   fObserverList.remove(observer);
 }

 /**
  * This method notifies the Subject instances that a change has
been observed.
  */
 public void toNotify() {

   Iterator it = fObserverList.iterator();

   // for each registered observer call the 'update'-method
   while (it.hasNext()) {
     ((Observer) it.next()).update();
   } // while

 }
}
```

```
package observer;

public interface State {
}
```

```
package observer;

public class ConcreteSubject extends Subject {

 /** stores the state of the ConcreteSubject */
 private State fSubjectState;

 /**
  * This method returns the state of the ConcreteSubject instance.
  */
 public State getState() {
   return fSubjectState;
 }

 /**
  * This method sets the state of the ConcreteSubject instance.
  */
 public void setState(State state) {
   fSubjectState = state;
 }
}
```

```
package observer;

public class ConcreteObserver implements Observer {

 /** stores the state of the ConcreteObserver */
 private State fObserverState;

 /** stores the associated ConcreteSubject */
 private final ConcreteSubject fConcreteSubject;

 /**
  * Constructor
  */
 public ConcreteObserver(ConcreteSubject subject) {
   super();
   fConcreteSubject = subject;
   // copy initial state
   fObserverState = fConcreteSubject.getState();
 }

 /**
  * This method updates the ConcreteObserver's state to be
consistent
  * with the ConcreteSubject's state.
  */
 public void update() {
   // Customize this method based on your application needs.
   fObserverState = fConcreteSubject.getState();
 }

}
```

Example

Auctions demonstrate this pattern. The auctioneer starts the bidding, and "observes" when a bidder accepts the bid. The acceptance of the bid changes the bid price that is broadcast to all of the bidders in the form of a new bid.

State Pattern

The State pattern encapsulates state-based behaviors and uses delegation to switch between behaviors. In this pattern, an object appears to change its class; however, note that this pattern isn't the same as the state of a finite-state machine. This refers to the state of a particular object, which can be thought of as the set of values of all its member variables at a particular time. Thus the intent is not to emulate FSM, but to allow an object to appear to alter its behavior-type at runtime.

State might seem similar to the Strategy pattern. But with the State pattern, we have a set of behaviors encapsulated in state objects; at any time the context is delegating to one of these states. The client knows very little, if anything, about the state objects. With Strategy, the client usually specifies the strategy object that the context is composed of. In general, Strategy is a flexible alternative to subclassing, whereas State is an alternative to putting lots of conditionals in your context. Classes that have several *switch* or *if-else* statements introduce maintenance issues. The State pattern modifies this spaghetti to an elegant pattern. By encapsulating the behaviors within state objects, you can simply change the state object in context to change its behavior.

Template

```
package state;

public class Context {

 /** stores the associated State instance */
 private State fState;

 /**
  * Constructor
  */
 public Context(State state) {
   super();
```

```
    fState = state;
  }

  /**
   * The Context object delegates all state-specific requests to its
   * associated State instance.
   */
  public void request() {
    // Customize this method based on your application needs.
    fState.handle(this);
  }

  /**
   * This method changes the state of the Context instance.
   */
  public void changeState(State state) {
    fState = state;
  }

}
package state;

public abstract class State {

  /**
   * This method changes the state of the given Context parameter.
   */
  protected void changeState(Context context, State state) {
    context.changeState(state);
  }

  /**
   * The ConcreteState must implement this abstract method.
   */
  public abstract void handle(Context context);

}
package state;

public class ConcreteState extends State {

  /**
   * Default constructor
   */
  public ConcreteState() {
    super();
  }

  /**
   * This method handles a request from a Context instance.
   */
  public void handle(Context context) {
    super.changeState(context, this);
  }

}
```

Example

Vending machines have states based on the inventory of items available at any point, the money deposited (in turn the ability to make change), the item selected, selecting a different item, start all over, etc. The machine will deliver an item only when the requisite amount has been deposited. The vending machine will then deliver a product and no change, or deliver a product and change, or deliver no product due to lack of inventory.

Chain of Responsibility Pattern

This pattern allows several objects in a chain to handle a request. The objects that handle the request are called receivers, thus the coupling between the sender and receiver can be avoided. No one receiver needs to know about any other receiver as well. The request is passed along the receivers until one of them can handle it.

This is useful when you want to issue a request to one of several objects without specifying a particular object explicitly, or if you want the ability to modify objects dynamically that can handle requests.

Template

```
package chainofresponsibility;

public abstract class Handler {

  /** each object in the chain acts as a handler and has a successor. */
  private Handler successor;

  /**
   * handle the request
   */
  public abstract void handleRequest() {
  }

}
package chainofresponsibility;

public class ConcreteHandler1 {

  /** inherited */
  private Handler successor;

  /**
   * if the request can be handled, it is; else  it is forwarded to its
successor
   */
  public abstract void handleRequest() {
  }

}
package chainofresponsibility;

public class ConcreteHandler2 {

  /** inherited */
  private Handler successor;

  /**
   * if the request can be handled, it is; else  it is forwarded to its
successor
   */
  public abstract void handleRequest() {
  }

}
```

Example

Filters, a very commonly used servlet, are an example of a chaining pattern. A request passes through a set of filters until a condition is satisfied. Filters are configurable via the Web configuration XML file. It's particularly useful in security, logging, and validation.

The template shown above assumes two handlers. Since there can be any number of handlers, you may customize the template according to your needs.

A classic example of this pattern is the inheritance mechanism. When executing a method in a multiple-level derived class, the method in the first parent class of this inheritance chain is executed.

Summary

I've discussed eleven behavioral design patterns here. If you are interested in learning more about patterns, I would recommend reading one of the many books available about design patterns.

References

- Java GoF Patterns: http://www.fluffycat.com/java/patterns.html
- Cooper, J. (1998). *The Design Patterns Java Companion*. IBM Thomas J. Watson Research Center: http://www.patterndepot.com/put/8/JavaPatterns.htm
- Sangal, P. (2004). "Java GoF Creational Design Patterns." *JDJ*, Vol. 9, issue 9: http://java.sys-con.com/read/46239.htm
- Freeman, E.; Freeman, E.; Bates, B.; and Sierra, K. (2004). *Head First Design Patterns*. O'Reilly.
- Sangal, P. (2005). "An Introduction to Antipatterns in Java Applications": http://www.devx.com/Java/Article/29162

Joe Winchester
Desktop Java Editor

What, Where, or Who Is Java?

Ask most people on the street what Java is and they might tell you it's an Indonesian island. If you happen to bump into some programmers, they'll probably tell you it's a language that reads like C++ but has garbage collection and a virtual machine to make it portable. The connection is reputedly the syllogism where the island gives name to its coffee, people drink coffee while surfing the Web, and Java is the computer language of the Web.

Most folks don't mistake volcanic islands with programming languages and there's enough clear water between Jakarta and San Jose to avoid any confusion. What troubles me is that the adjective Java within IT itself is no longer as clear as it once was.

The words *Desktop Java* when browsed on java.sun.com, for example, talks very clearly about J2SE, with *Desktop* being a subgroup along with *Core Java, Embedded*, and *Realtime*. However, you'll also find hits around the *Java Desktop System* and stories of how it's going to be bundled with Linux and dethrone the Microsoft operating system and MSOffice. The Java Desktop System is a grouping of various apps such as a browser, office, mail, and other commodity programs that all run on Linux and, bundled together, provide a pretty solid operating environment. I take my hat off to everyone who works on it and wish them nothing but success; my fight is with whoever decided to give it the "Java" adjective.

A few weeks ago *The Hindu*, an Asian newspaper, ran a story that showed Scott McNealy unashamedly plugging the Java system for the desktop environment in India (http://www.hindu.com/thehindu/seta/2003/10/16/stories/2003101600070200.htm). It's a fun read except that it horribly confuses the Java programming language with Sun's all-in-one Linux desktop program suite. To make things clearer to its readers in case any of them were confused about what Scott is selling, it has a paragraph authoritatively lifted word for word from the *Computer Desktop Encyclopedia* that extols the virtues of the Java language and its ability to run cross platform in applets.

At *Java Developer's Journal* I have the nice title of "Desktop Java Editor." I review and help to put together content related to J2SE, yet I get press releases from people wanting me to plug the "Java Desktop System"; recently I received an e-mail from an editor of another very reputable technology publication who asked me to comment on the recent news that the "Java Desktop system" now included StarOffice 7. He assumed, not unnaturally, that "Desktop Java" = "Java Desktop System."

There are countless other examples of where the Java desktop is being confused with the Java language, and I can't help but believe that this is something being deliberately done by the lad or lassie at Sun who named it in the first place. Did a desperate marketing person trying to come up with a sexy name for their desktop suite think that no confusion would exist, or was this Machiavellian move an attempt to deliberately confuse people and let the kudos, image, and good name of the Java language be used to market Sun's Linux desktop system?

In the guidelines on usage of their trademarks (http://www.sun.com/policies/trademarks/) it states: "*Unauthorized use of Sun trademarks or of marks that are confusingly similar to Sun trademarks may constitute an infringement of Sun's trademark rights.*" This means that if you came up with, for example, a 100% Pure Java suite of desktop programs and wanted to call it "My super duper Java desktop," you couldn't unless you got a license agreement from Sun. Furthermore the site http://www.sun.com/smi/Press/sunflash/1997-10/sunflash.971015.0001.tm-noindex.html states that to use Java branding, you must certify against the Java Test Compatibility Kit, something that most of the "Desktop Java" suite doesn't even come close to, given that parts of it are written in C++.

While I'm on my soap box, I'd like to mention Sun Java Studio Creator. I love the product, haven't a bad word to say about anyone who works on it, but on www.java.com/en it lists the creation of Sun Java Studio Creator in 2004 as one of the three most important events in Java's history that year – together with the release of Java 5 (aka Tiger) and the Mars Rover buggies being powered by Java (the bit about the rover's software crashing and crippling the unit for a month is left out). My memory may not be as good as the Sun employees who write the content for java.com, but in 2004 I also remember key events such as the incredible growth of JBoss and the release of Eclipse 3.0.

If Sun can't eat their own dog food when it comes to the usage of the Java brand, and repeatedly ignore everything else that goes on in the Java community that they don't control or are part of, then it is their loss. Java technology is big enough that it has, and will, continue to win over C# and Java is the language choice of three-quarters of the people building Web apps. There are millions of programmers who write Java programs, countless individuals who tirelessly create great and truly open communities based around Java technology, and thousands of successful companies that use Java technology effectively to solve their business problems on a daily basis. History books are written by the winning site, while propaganda can be written by anyone.

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

joewinchester@sys-con.com

ENGAGE AND EXPLORE...

The Technologies, Solutions and Applications that are Driving Today's Initiatives and Strategies...

CALL FOR PAPERS NOW OPEN!

SOA WebServices Edge 10th International conference+expo Edge 06

The Sixth Annual SOA Web Services Edge 2006 East - International Web Services Conference & Expo, to be held June 2006, announces that its Call for Papers is now open. Topics include all aspects of Web services and Service-Oriented Architecture

Suggested topics...

- > Transitioning Successfully to SOA
- > Federated Web services
- > ebXML
- > Orchestration
- > Discovery
- > The Business Case for SOA
- > Interop & Standards
- > Web Services Management
- > Messaging Buses and SOA
- > Enterprise Service Buses
- > SOBAs (Service-Oriented Business Apps)
- > Delivering ROI with SOA
- > Java Web Services
- > XML Web Services
- > Security
- > Professional Open Source
- > Systems Integration
- > Sarbanes-Oxley
- > Grid Computing
- > Business Process Management
- > Web Services Choreography

CALL FOR PAPERS NOW OPEN!

2006 ENTERPRISE> OPENSOURCE CONFERENCE+EXPO

The first annual Enterprise Open Source Conference & Expo announces that its Call for Papers is now open. Topics include all aspects of Open Source technology. The Enterprise Open Source Conference & Expo is a software development and management conference addressing the emerging technologies, tools and strategies surrounding the development of open source software. We invite you to submit a proposal to present in the following topics. Case studies, tools, best practices, development, security, deployment, performance, challenges, application management, strategies and integration.

Suggested topics...

- > Open Source Licenses
- > Open Source & E-Mail
- > Databases
- > ROI Case Studies
- > Open Source ERP & CRM
- > Open-Source SIP
- > Testing
- > LAMP Technologies
- > Open Source on the Desktop
- > Open Source & Sarbanes-Oxley
- > IP Management

Submit Your Topic Today! www2.sys-con.com/events

Sponsored by

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

Attention Exhibitors:
An Exhibit-Forum will display leading Web services and OpenSource products, services, and solutions

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

For Exhibit and Sponsorship Information ▸ Call 201 802-3066

Produced by SYS-CON EVENTS

List-Based UI Framework

Ideas for your Swing application development

by Phil Herold

In this month's article I continue my discussion of a list-based UI framework that I started last month ("ArrayListModel," [*JDJ*, Vol. 10, issue 10]). The primary concept behind this idea is a data model that contains elements that describe parts of an application's user interface. Through a single model, various aspects of the user interface can be controlled, manipulated, and visually synchronized. There is a lot of interesting code that accompanies this article, so I encourage you to download it and check it out. Let's get started.

Data Model

Recall from last month's article the UIElement interface that describes our user interface element:

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

The UIElement interface includes methods that allow other Swing component consumers to construct the user interface. The getDescription() method would normally be used to provide tooltip text, while the get*Icon() methods can be used appropriately to provide icons on such things that include but are not limited to menu items, toolbar buttons, internal window icons, and tabbed pane icons (we'll see examples of each of these in our discussion). The getComponent() method is used to associate some other visual component with the UIElement. This is typically used when the UIElement shown in the user interface is selected.

Our data model, UIElementListModel (see Listing 1), is based on the ArrayListModel that I also introduced in last month's article. This is an ArrayList sub-class that implements the ListModel interface, which allows a consumer to listen for changes in the underlying Collection via the ListDataListener interface. UIElementListModel extends ArrayListModel and types the collection to contain only UIElement objects. In addition, the class overrides the remove() method in the Collection, allowing for a VetoableChangeListener to intercept a remove operation. Remember that our data model consists of user interface elements (UIElement objects), which can be removed by some user-initiated operation. The classic example is the closing of an internal frame that contains unsaved data – the user must be prompted to save the data first or allow the close operation to proceed without saving. The remove (close) operation is done through the data model as other aspects of the user interface are listening to changes in the UIElementListModel model. We'll see an example of this later.

Since the UIElement construct is fairly ubiquitous in our framework, it begs for an AbstractUIElement implementation that handles some of the common tasks described by the interface. The code accompanying the article includes an immutable AbstractUIElement implementation. Probably the most interesting thing about it is that it handles the getLargeIcon() method by scaling the small icon associated with the UIElement, or vice versa (i.e., if a large icon is provided, but not a small icon, it will scale down the large one). Listing 2 shows the implementations of these methods. Obviously the scaling isn't perfect, and it certainly isn't meant to replace the work of a good graphics designer. Nevertheless, it's great for prototyping your UI when you might have only crude versions of one icon or the other.

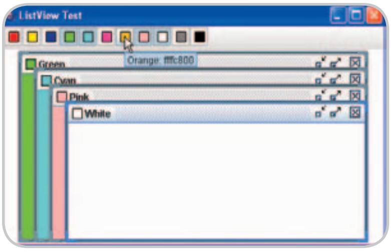


Figure 1 ListViewTest

Views on Our Model

Now that we have a solid foundation for the model side of our framework, we need views that work with this model. Our views are naturally going to implement some interface, and we'll provide an abstract implementation of this interface to simplify the task of implementing various views. Listing 3 defines our interface, UIElementListView.

Our view interface supports our data model (UIElementListModel) with a getter and setter. It also has the concept of a selected item/index and a SingleSelectionModel that is conveniently used to enable selection listeners via the add/remove ChangeListener methods. Finally, our interface is supported visually by some component – the getComponent() method. My choice of a SingleSelectionModel here is intentional, as usually there is only a single "active" item (window, internal frame, tabbed pane, etc.) in an application.

The accompanying code to this article, which can be downloaded from http://jdj.sys-con.com, provides an AbstractListView implementation of the UIElementListView interface. The most interesting method in this class is the setModel() method (see Listing 4). This method first checks to see if it currently hosts a model and, if so, it removes all of the underlying views (there are

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

abstract insert and remove methods in this class that are used to add/remove a view). Remember that a view corresponds to a UIElement (contained in our data model, UIElementListModel), with an associated component. So, for example, a tabbed pane AbstractListView implementation returns a JTabbedPane in its getComponent() method. Each tabbed pane in this view hosts a component that is returned by the getComponent() method of the UIElement object. The insert and remove methods of a tabbed pane AbstractListView would be used to add/remove a tabbed pane. You can see in Listing 4 where a view is inserted for each element in the new data model.

AbstractListView also implements ListDataListener and adds itself as a listener on the UIElementListModel. In this manner, it can respond to changes in the data model that are initiated in other parts of the application. For example, a menu item can be used to "close" a frame, internal frame, or tabbed pane of an application. The menu item action simply has to remove the element from the model and any views using the model will react accordingly.

Finally, AbstractListView provides a VetoableChangeListener on our UIElementListModel. The vetoableChange() method in the listener simply delegates to a canClose() method in AbstractListView, in which the default implementation returns true. A subclass can override canClose() if needed, for example, to prompt the user for unsaved data. There is an example of this below, too.

Putting It All Together

I've introduced quite a few concepts and classes up to this point, so it's time to show a few examples of how all of this is used (you may be wondering about this at this point as well). You may recall the sample program from last month's article. Figure 1 shows a similar application, consisting of a toolbar with toggle buttons and a JDesktopPane.

In the code provided with this article, you can run the ListViewTest Eclipse launch target to execute this sample program (you'll need a J2SE 5.0 JDK with Eclipse to compile and run the code). The toolbar is constructed from all of the elements returned by a UIElementFactory implementation. The JDesktopPane is provided in the getComponent() method of a class called MDIView, which simply extends AbstractListView. The model for the application is a UIElementListModel, where each element of the model corresponds to an internal frame in the JDesktopPane (MDIView). Selecting a toggle on the toolbar either adds or removes (based on the toggle state) the corresponding UIElement from the model. In Figure 1, I've selected the Green, Cyan, Pink, and White toolbar items, respectively.

When you run ListViewTest application, you specify the class name of the UIElementFactory and the class name of the underlying AbstractListView. Figure 2 shows another UIElementFactory implementation. (There are two UIElementFactory implementations provided with the sample code. Each returns instances of UIElement objects that are created by extending AbstractUIElement.)

There are two additional AbstractListView implementations provided as part of the sample code: ListView, which uses a JList as its view on the model (see Figure 5 below); and TabbedPaneView, which uses a JTabbedPane as the view (see Figure 3). Other AbstractListView implementations are possible – use your imagination. (For example, though not provided with the sample code, I have a ShortCutPanel implementation where a custom icon button on the left is used to select a view on the right [for an example, see the Tools->Options dialog in the Firefox browser]. The panel supports small and large icons – and these are gotten from the UIElement objects in the model).

Again, the different looks of each of Figures 1 through 3 are created simply by providing different runtime arguments to the ListViewTest application.

Figure 4 is a result of de-selecting the "Pink" item on the toolbar (refer back to Figure 1). This user-initiated action removed that UIElement from the model, and the AbstractListView (MDIView) reacted accordingly by removing the corresponding internal frame.

You can easily imagine this happening as a result of menu item selection, or even more directly from pressing the window close button (X) on the internal frame. In fact, MDIView does just that. For each JInternalFrame

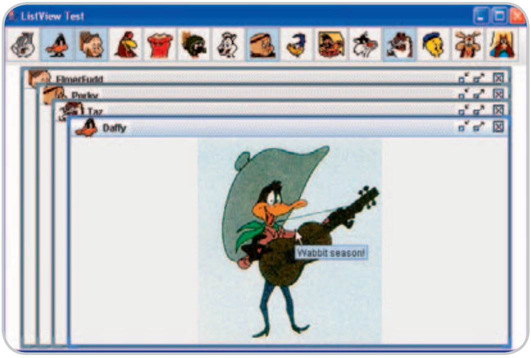


Figure 2 ListViewTest – alternate UIElement factory

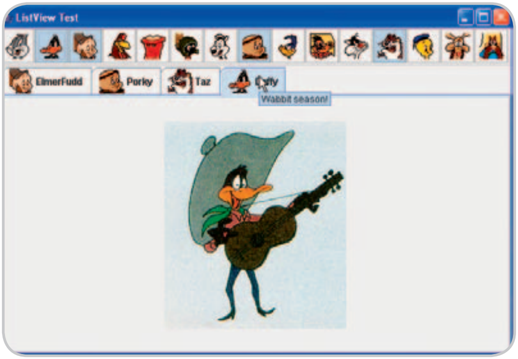


Figure 3 ListViewTest – Tabbed Pane AbstractListView

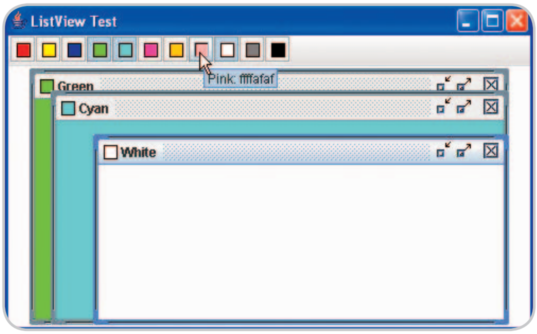


Figure 4 ListViewTest – Removing a UIElement from the model

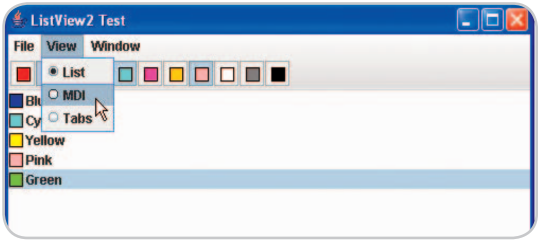


Figure 5 ListViewTest2 – View Menu

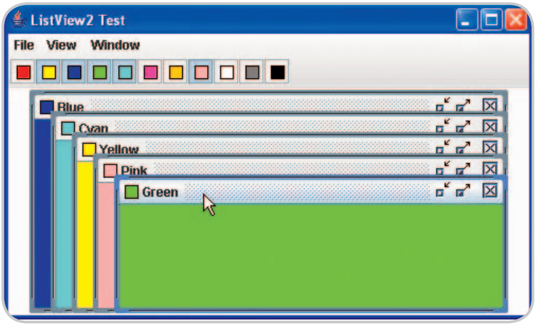


Figure 6 ListViewTest2 –MDI AbstractListView

that's added, it adds an InternalFrameAdapter to the frame (see Listing 5).

Note that the internalFrameClosing() method does the removal from the UIElementListModel (and *not* the AbstractListView nor the JDesktopPane). The AbstractListView (MDIView) will react accordingly, as will other user interface elements that might be listening to changes in the model.

A More Sophisticated Example

With the sample code for this application I've also provided a more comprehensive implementation of the concepts presented earlier. In particular, it allows for the view to be dynamically changed. It also provides an additional listener on the model that tracks the changes to the model, the very familiar "Window" menu concept.

Run the ListViewTest2 Eclipse launch target in the sample project. As in the ListViewTest above, you specify the UIElementFactory implementation. Figure 5 shows this application, with the default ListView as the AbstractListView, and several items already selected on the toolbar. Figure 6 shows the result of switching to the MDIView.

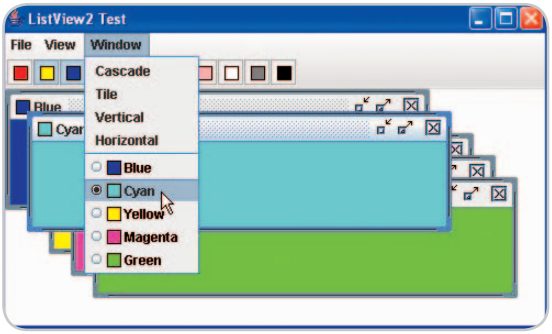
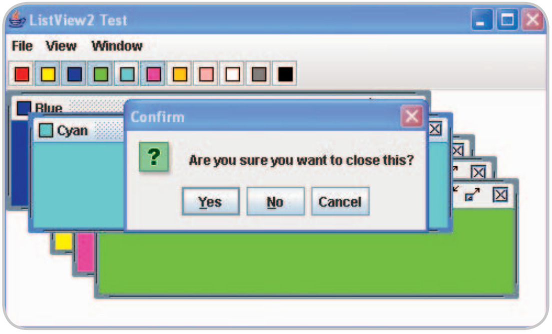


Figure 7 ListViewTest2 – Window Menu



Are you sure you want to close this?

Figure 8 ListViewTest2 – Intercepting model remove()

The ability to change views dynamically while maintaining the underlying UIElement views in the model was accomplished by the the setModel() method of AbstractListView that we previously examined. At this point I'll admit that this capability was inspired by a need I had in an application I was writing: a 3270 terminal emulator application, in which each view (UIElement) was a TN3270 session with a remote mainframe host. Originally implemented as an MDI application only, I wanted the ability to switch to a tabbed view without losing any data. I thought that would be pretty cool, and it was (though I must also confess that I never finished working on this, since it was a skunk work project, and a large one at that). It's interesting to note that the Microsoft Visual Studio 2003 IDE (I haven't seen the 2005 version) supports either an MDI or tabbed view, but you have to shut down the application and restart it to affect the change – now how lame is that?

ListViewTest2.java contains an inner class WindowMenu that is used to track changes to the UIElementListModel (it's a ListDataListener on the model). It also tracks the currently selected view as a ChangeListener on the SingleSelectionModel of the current view. To do this, it has to know when the current view changes, which is accomplished via a PropertyChangeListener on the test application. Remember that selection is an attribute of the view, not the model. This is visually shown in Figure 7, where the "Cyan" internal frame was activated by clicking its frame boundary, and that item is subsequently selected in the Window menu, reflecting the current state of the application.

Similarly, selecting another "window" item on this menu activates the corresponding internal frame. This is accomplished with a single line of code in the actionPerformed() method of the menu item's ActionListener:

```
 currentView.setSelectedIndex(windowItems.
indexOf(e.getSource()));
```

where currentView is the current AbstractListView shown, and windowItems is a cache (ArrayList) of the dynamic JRadioButtonMenu items on the Window menu.

As an added bonus, the MDIView in the sample code has implementations of Cascade and three different Tile methods as shown on the Window menu in Figure 7 – you might find these useful.

Finally, Figure 8 shows the result of selecting the "Cyan" toolbar toggle, with the resulting remove() call on the model being intercepted by MDIView. (Again, this action could also have been initiated by using the "Close" button on the internal frame.)

The MDIView subclass of AbstractListView simply has to override the canClose() method. Listing 6 shows how this is done in ListViewTest2.java.

The call to updateToggleState() shown in Listing 6 is used to synchronize the toggle buttons on the toolbar, to ensure that they are synchronized with the UIElementListModel. This is necessary because a toolbar toggle button may have been used to initiate the "close", but it ultimately was rejected, so the button that was pressed becomes out of sync with the application. (I suppose you could implement an AbstractListView based on a JToolBar, which would listen to changes in the model directly. I don't really consider a JToolBar a good AbstractListView candidate, simply because you often need the ability to disable/enable toolbar buttons, and this framework doesn't provide that, by design. In this particular test application, we're using toggle buttons on the toolbar, which implies multiple selection state, which is also contrary to the design.)

Conclusion

In this article, I've presented a small framework for building a Swing application based on the familiar Model/View/Controller concept. The twist on this implementation is that the data model is actually a model of visual components, but like traditional MVC, there are various views on the model. The framework is complete enough to allow various aspects of the user interface to react to changes on a single model, rather than the customary delegation of change events through a common PropertyChangeSupport object.

I wouldn't expect a large application or framework to be built solely on these concepts, but I hope I have given you some ideas to consider in your Swing application development.

Listing 1: UIElementListModel class

```
public class UIElementListModel extends ArrayListModel<UIElement> {
   public interface ConstrainedProperties {
         final String REMOVE = "remove";
   }
   private VetoableChangeSupport vcs;


    @Override
   public UIElement remove(int index) {
         boolean okToRemove = true;
         UIElement removedElement = get(index);
         if (vcs != null) {
               try {
                     vcs.fireVetoableChange(new VetoableChan
geEvent(this, ConstrainedProperties.REMOVE, new, removedElement));
               } catch (PropertyVetoException pve) {
                     okToRemove = false;
               }
         }
         if (okToRemove) {
               removedElement = super.remove(index);
         }
         return removedElement;
   }


   public void addVetoableChangeListener(VetoableChangeListener lis-
tener) {
         if (vcs == null) {
               vcs = new VetoableChangeSupport(this);
         }
         vcs.addVetoableChangeListener(listener);
   }


   public void removeVetoableChangeListener(VetoableChangeListener
listener) {
         if (vcs != null) {
               vcs.removeVetoableChangeListener(listener);
         }
   }
}
```

Listing 2: Sample of AbstractUIElement methods

```
public Icon getSmallIcon() {
   if (smallIcon == null || largeIcon == smallIcon) {
         if (largeIcon != null) {
               smallIcon = scaleIcon(getComponent(), largeIcon,
0.5);
         }
   }
   return smallIcon;
}

public Icon getLargeIcon() {
   if (largeIcon == null || largeIcon == smallIcon) {
         if (smallIcon != null) {
               largeIcon = scaleIcon(getComponent(), smallIcon,
2.0);
         }
   }
   return largeIcon;
}

private Icon scaleIcon(Component component, Icon icon, double scale) {
   int newWidth = (int)(icon.getIconWidth() * scale);
   int newHeight = (int)(icon.getIconHeight() * scale);
   if (!(icon instanceof ImageIcon)) {
         BufferedImage bi = new BufferedImage(icon.getIconWidth(),
icon.getIconHeight(), BufferedImage.TYPE_INT_ARGB);
         icon.paintIcon(component, bi.getGraphics(), 0, 0);
         icon = new ImageIcon(bi);
   }
   return new ImageIcon(((ImageIcon)icon).getImage()
.getScaledInstance(newWidth,
                     newHeight, Image.SCALE_SMOOTH));
}
```

Listing 3: UIElementListView interface

```
public interface UIElementListView {
   Component getComponent();
   UIElementListModel getModel();
   void setModel(UIElementListModel model);
   UIElement getSelectedItem();
   void setSelectedItem(UIElement element);
   int getSelectedIndex();
   void setSelectedIndex(int index);
   SingleSelectionModel getSingleSelectionModel();
}
```

Listing 4: setModel() Method in AbstractListView

```
public void setModel(UIElementListModel newModel) {
   if (model != newModel) {
         if (model != null) {
               model.removeListDataListener(this);
               model.removeVetoableChangeListener(vcs);
               int size = model.size();
               for (int i = 0; i < size; i++) {
                     remove(0);
               }
         }
         if (newModel == null) {
               newModel = new UIElementListModel();
         }
         model = newModel;
         model.addListDataListener(this);
         model.addVetoableChangeListener(vcs);
         SwingUtilities.invokeLater(new Runnable() {
               public void run() {
                     int index = 0;
                     for (UIElement element : model) {
                           insert(index++, element);
                     }
               }
         });
   }
}
```

Listing 5: InternalFrameAdapter for MDIView

```
private InternalFrameListener frameListener = new InternalFrameAdapt-
   er() {
       @Override
   public void internalFrameActivated(InternalFrameEvent e) {
         int index = frames.indexOf(e.getSource());
         setSelectedIndex(index);
         selectionModel.setSelectedIndex(index);
   }
      @Override
   public void internalFrameClosing(InternalFrameEvent e) {
UIElement element = ((InternalFrame)e.getSource()).element;
         getModel().remove(element);
   }
};
```

Listing 6: Overriding canClose() in an AbstractListView

```
mdiView = new MDIView() {
   @Override
   protected boolean canClose(UIElement element) {
boolean canClose =
   JOptionPane.showConfirmDialog(ListViewTest2.this,
                     "Are you sure you want to close this?", "Confirm",
         JOptionPane.YES_NO_CANCEL_OPTION)
== JOptionPane.YES_OPTION;

   SwingUtilities.invokeLater(new Runnable() {
         public void run() {
               updateToggleState();
         }
   });
return canClose;
}
};
```

2005 JCP EC Elections

Ratification results in, open nominations under way

by Onno Kluyt

Last month I introduced the 2005 ratified ballot nominees for the JCP ECs. Meanwhile ratified voting concluded on October 17 and results confirmed all nominees: BEA Systems, SAP AG, and SAS Institute for the Standard/Enterprise Edition Executive Committee; and Nokia Corporation, IBM, and Philips Electronics UK for the Micro Edition Executive Committee. Congratulations to all and a warm welcome to new EC member SAS. You'll find more details about this round of elections at http://www.jcpelection2005.org/jcp/ratification_results.

What's next? Open nominations are now under way and run until the end of October. This year there are two seats available on the SE/EE EC and two seats available on the ME EC. Several companies and individual developers have already self-nominated to participate in open elections and I'd like to intro duce them to you as candidates for the JCP ECs. First those for the JCP SE/EE Executive Committee.

Intel Corporation has participated in 14 JSRs: JSR 277 – Java Module System, JSR 271 – Mobile Information Device Profile 3, JSR 253 – Mobile Telephony API (MTA), JSR 250 – Common Annotations for the Java Platform, JSR 218 – Connected Device Configuration 1.1, JSR 217 – Personal Basis Profile 1.1, JSR 209 – Advanced Graphics and User Interface Optional Package for the J2ME Platform, JSR 205 – Wireless Messaging API 2.0, JSR 184 – Mobile 3D Graphics API for J2ME, JSR 179 – Location API for J2ME, JSR 174 – Monitoring and Management Specification for the Java Virtual Machine, JSR 163 – Java Platform Profiling Architecture, and JSR 113 – Java Speech API 2.0.

In support of its candidature, Intel also brings its leadership in microprocessor innovation and other cutting-edge technologies in the vast array of devices and computing platforms to where the Java platform is being deployed. These include multi-core and threading, wireless and wired networking, and semiconductor devices for embedded controllers, mobile phones, PDAs, set-top boxes, laptops, PCs, servers, and supercomputers.

IONA Technologies has been an elected member of the JCP SE/EE Executive Committee since 2000. It contributed to over 40 JSRs including JSR 215 – Java Community Process version 2.6, JSR 261 – Java API for XML-based Web Services Addressing (JAX-WSA), and JSR 265 – API for Utilizing Web Services Policy. Its EC representatives have worked closely with Intel and Sun to formulate the procedures

now adopted by the EC for running an EC meeting. The company helped evolve the JCP program to version 2.6, which requires of members more transparency in the development process, encourages more community involvement, and helps move specifications more quickly through the process.

EMC Corporation brings, in support of its self-nomination, its broad expertise in developing products, services, and solutions for information storage, its participation in industry bodies that develop open standards to facilitate interoperability including its contributions to the JCP through EMC Documentum. EMC participates in a number of JCP JSRs, most important of which are JSR 170 – Content Management, JSR 168 – Portlets, JSR 283 – Content Repository, and JSR 127 – Faces. In some of these the company participated as formal members of the expert committees and on others as contributors. EMC plans to leverage the acknowledged technical expertise in systems management within its SMARTS division to broaden its participation in JSRs in this area as well.

The self-nominations for the JCP ME EC are:

Aplix Corporation comes with extensive JCP experience through its Expert Group participation in JSRs including JSR 118 – Mobile Information Device Profile 2.0, JSR 220 – Enterprise JavaBeans 3.0, JSR 135 – Mobile Media API, JSR 139 – Connected Limited Device Configuration 1.1, JSR 172 – J2ME Web Services Specification, JSR 179 – Location API for J2ME, JSR 180 – SIP API for J2ME, JSR 184 – Mobile 3D Graphics API for J2ME, JSR 185 – Java echnology for the Wireless Industry, R 195 – Information Module Profile, R 205 – Wireless Messaging API 2.0, JSR 211 – Content Handler API, JSR 226 – Scalable 2D Vector Graphics API for J2ME, JSR 228 – Information Module Profile - Next Generation (IMP-NG), JSR 229 – Payment API, JSR 230 – Data Sync API, JSR 232 – Mobile Operational Management, JSR 233 – J2EE Mobile Device Management and Monitoring Specification, JSR 234 – Advanced Multimedia Supplements, JSR 239 – Java Bindings for OpenGL ES, JSR 246 – Device Management API, and JSR 271 – Mobile Information Device Profile 3. The company quotes its global leadership in deploying Java technology in mobile phones with over 129 million mobile phones and consumer electronics devices as of the end of June 2005. The Aplix implementation of a Java Runtime Environment for devices has been licensed by over 30 handset manufacturers and deployed in over 250 phone models offered by wireless opera-

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

onno@jcp.org

JAVA DEVELOPER'S JOURNAL

Advertiser Index

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	4, 27
Arcturus Technologies	www.arcturustech.com	703-822-4582	17
ceTe Software	www.dynamicpdf.com	800-631-5006	39
ExtenTech	www.extentech.com/jdj	415-759-5292	33
Google	www.google.com/jdj	650-253-0000	41
IBM	www.ibm.com/middleware/tools		Cover IV
ILOG	http://diagrammer.ilog.com	800-FOR-ILOG	35
InetSoft	www.inetsoft.com/jdj	888-216-2353	37
InterSystems	www.intersystems.com/cache12p	617-621-0600	Cover II
IT Solutions Guide	www.itsolutions.sys-con.com	888-303-5282	61
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	57
Jinfonet Software	www.jinfonet.com/jp10	301-838-5560	21
MapInfo	www.mapinfo.com/sdk	800-268-3282	13
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	45
Parasoft Corporation	www.parasoft.com/jdjmagazine	888-305-0041	7
PDF Tools	http://www.pdf-tools.com	403-932-4220	31
ReportingEngines	www.reportingengines.com	88-884-8665	9
Smart Data Processing, Inc.	www.weekendwithexperts.com	732-598-4027	49
Software FX	www.softwarefx.com	800-392-4278	Cover III
SYS-CON Events	www2.sys-con.com/events	201-802-3066	55
WebAppCabaret	www.ngasi.com/jdj.jsp	866-256-7973	23
Webapper	www.seejava.com		47
Windward Studios, Inc.	www.windwardreports.com	303-499-2544	11, 42-43

tors around the world. Aplix became a Sun Java license in 1996 and has been a JCP member since 2000.

BenQ Corporation – on October 1, 2005, Siemens Mobile Devices, a division of Siemens AG, officially became a new business group of BenQ Corporation. BenQ is one of the world's top producers of digital displays, mobile phones, scanners, and keyboards and a technology leader in digital projectors, storage devices, wireless technologies, and electronic components. The nine Java ME spec leaderships have transferred from Siemens to BenQ Mobile, which makes BenQ Mobile the number three top Java ME spec leads contributor after Sun Microsystems and Nokia. The skills and experience that come with the spec leaderships from Siemens are met by the BenQ traditional Java skills and experience, resulting into a robust Java team.

Cox Communications, Inc., is one of the leading North American cable system operators with over 6.6 million customer households. The company is in the process of a multi-year deployment of JSR 242 and the Open Cable Applications Platform (based on CDC Personal Basis Profile) throughout its entire network. Cox is a member and active participant in many industry standards bodies including JCP, ATSC, CableLabs, CEA, DVB, IETF, SCTE, and SMPTE. Cox is the specification lead for JSR 242, which recently completed its Final Approval Ballot and a long-time observer status member of JME EC and JSRs.

Established in 1958, ***LG Electronics, Inc.***, is one of the world's major forces in developing leading-edge digital convergence technology and products in electronics and mobile communications. As a world-class mobile handset and consumer electronics manufacturer, LG Electronics has entered the open nominations ready and willing to contribute its expertise for technical innovations in the JCP. LG Electronics also takes advantage of Java technologies to develop value-added products across its broad range of product line. For example, more than 30% of LG mobile phones are equipped with the Java platform. LG Electronics currently participates in JSR 271, JSR 272, and JSR 281 actively as well as JSR 256 and JSR 257 as an observer. LG Electronics has a strong will to lead technical innovation and contribute to the JCP by actively participating in and proposing valuable JSRs.

SiRF Technology is a global market and technology leader in location and wireless mobility technology. Since its founding in 1995, SiRF has been in the forefront of GPS technologies and has brought the benefits of wireless mobility to the masses by driving open standards, high-performance products, and lower price-points. SiRF is a strong advocate of Java standards in the location and wireless industry and maintains a large marketing and technical group to promote and implement Java standards globally across its large wireless operators and handset OEM customer base.

Sony Ericsson Mobile Communications AB is among the top five handset manufacturers in the world and deploys Java ME in most of its phones. The company plays an active role in the Java ME industry, working toward advancing Java ME technology to be the best open platform for application development on mobile devices. Sony Ericsson has been a member of Java ME EC since 2002 and has taken an active role in the ME Executive Committee, working together with the other members to ensure an open, well-managed community. The company has also been active in many JSR expert groups and contributed to the development of platform JSRs like JTWI and MSA. Its portfolio of contributions includes JSR 185 – Java Technology for the Wireless Industry, JSR 215 – Java Community Process version 2.6, 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, and JSR 248 – Mobile Service Architecture for CLDC, and JSR 249 – Mobile Service Architecture for CDC. The company has implemented at least five new JSRs in its products this year including JSR 75 – PDA Optional Packages for the J2ME Platform, JSR 82 – Java APIs for Bluetooth, JSR 172 – J2ME Web Services Specification, JSR 184 – Mobile 3D Graphics API for J2ME, and JSR 205 – Wireless Messaging API 2.0.

Symbian Ltd is a software licensing company that develops and supplies the advanced, open standard operating system – Symbian OS – for data-enabled mobile phones. During the first half of 2005, 14.5 million phones using Symbian OS shipped worldwide, bringing the worldwide installed base to more than 39 million phones. Symbian provides an implementation of CLDC, MIDP2, and other JSRs with the OS. Symbian has a long history of active participation in the JCP and JSR specification process aimed at driving forward mobile Java standards and delivering optimized product quality implementations into smartphones.

> "The race is on; don't miss the opportunity to cast your vote in support of the individual developer or company that you consider to be the best candidate for JCP ECs membership"

More JCP members are expected to self-nominate before the close of open elections at the end of October. The elections ballot will open on November 1 and will run for 14 days. Check out details about the open nominations and the upcoming election ballot at https://www.jcpelection2005.org/jcp/nominate .

The race is on; don't miss the opportunity to cast your vote in support of the individual developer or company that you consider to be the best candidate for JCP ECs membership. And stay tuned for final election results in the next *Java Developer's Journal* issue!

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

New! version 6.2
Now Includes Maps!

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

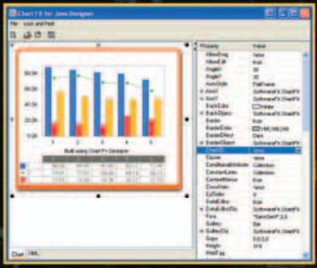


Chart FX for Java Designer

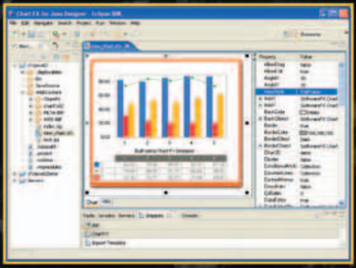


Chart FX for Java - Eclipse

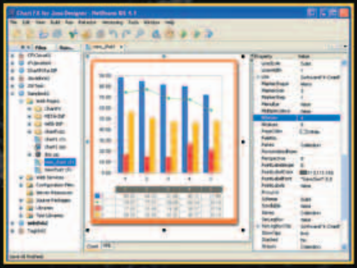


Chart FX for Java - NetBeans

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