

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

JDJ

JDJ.SYS-CON.COM

VOL.10 ISSUE:12

IN THIS ISSUE...

Integrating AJAX with JMX

PAGE 20

Build Management Allows
Developers to Develop

PAGE 32

Do Strongly Typed APIs Help Ensure
Java's Survival or Extinction?

PAGE 48

Table Layout

Replace your
GridBagLayout code

RETAILERS PLEASE DISPLAY
UNTIL FEBRUARY 28, 2006

\$5.99US \$6.99CAN

01>



0 09281 01751 6

PLUS...

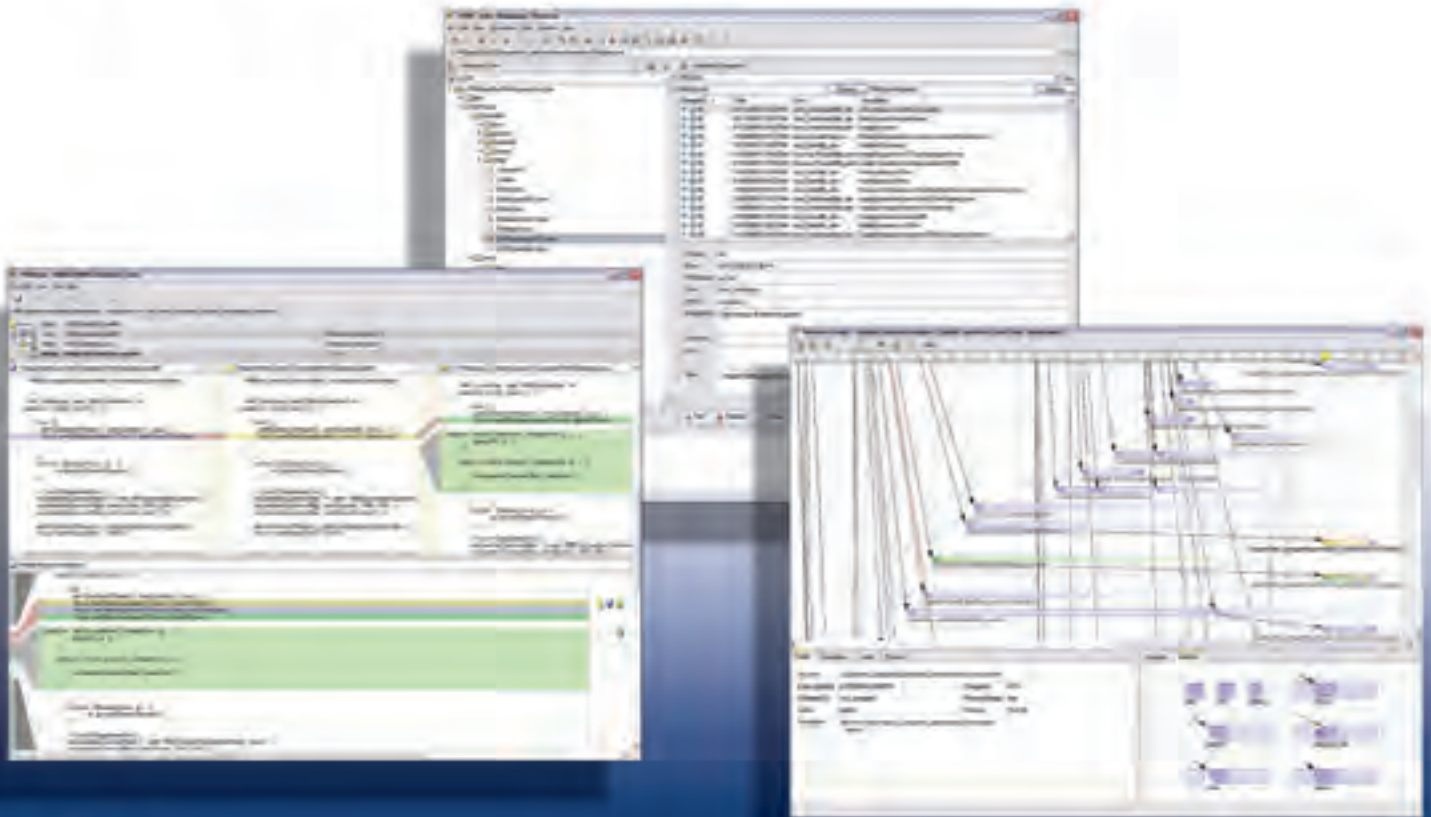
► Managing Java Performance Across
the Application Life Cycle

► Developing
in Java 5

► Extending Rich GUI Clients
with Jython

Perforce Software Configuration Management

Powerful and scalable.



[Fast]

[Scalable]

[Distributed]

Perforce tracks and manages source code and digital asset development, giving you maximum control with minimum interference.

Perforce scales to **thousands of concurrent users**. It manages code bases of **more than a million files**. It processes **terabytes of data** - from source code to documents, web content, and image files.

Perforce is fast, industrial-strength SCM. Any team size, any amount of data - Perforce handles it.



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

All trademarks used herein are either the trademarks or registered trademarks of their respective owners.

Accelerating Uphill



Jeremy Geelan



Editorial Board

Java EE Editor: **Yakov Fain**
 Desktop Java Editor: **Joe Winchester**
 Eclipse Editor: **Bill Dudney**
 Enterprise Editor: **Ajit Sagar**
 Java ME Editor: **Michael Yuan**
 Back Page Editor: **Jason Bell**
 Contributing Editor: **Calvin Austin**
 Contributing Editor: **Rick Hightower**
 Founding Editor: **Sean Rhody**

Production

Production Consultant: **Jim Morgan**
 Associate Art Director: **Tami Lima**
 Executive Editor: **Nancy Valentine**
 Associate Editor: **Seta Papazian**
 Research Editor: **Bahadır Karuv, PhD**

Writers in This Issue

Yakov Fain, Matt Gabor, Jeremy Geelan, Graham Paul Harrison, Phil Herold, Onno Kluyt, Hayden Marchant, David Pauls, Michael Poulin, John Reichard, Ajit Sagar, Roberto Scaramuzzi, Joe Winchester, Michael Yuan

To submit a proposal for an article, go to
<http://jdi.sys-con.com/main/proposal.htm>

Subscriptions

For subscriptions and requests for bulk orders, please send your letters to Subscription Department:

888 303-5282

201 802-3012

subscribe@sys-con.com

Cover Price: \$5.99/issue. Domestic: \$69.99/yr. (12 Issues)
 Canada/Mexico: \$99.99/yr. Overseas: \$99.99/yr. (U.S. Banks or Money Orders) Back Issues: \$10/ea. International \$15/ea.

Editorial Offices

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

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

©Copyright

Copyright © 2005 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission. For promotional reprints, contact reprint coordinator Dorothy Gil, dorothy@sys-con.com. SYS-CON Media and SYS-CON Publications, Inc., reserve the right to revise, republish and authorize its readers to use the articles submitted for publication.

Worldwide Newsstand Distribution
 Curtis Circulation Company, New Milford, NJ

For List Rental Information:

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

Newsstand Distribution Consultant
 Brian J. Gregory/Gregory Associates/W.R.D.S.
 732 607-9941, BJGAssociates@cs.com

Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. SYS-CON Publications, Inc., is independent of Sun Microsystems, Inc. All brand and product names used on these pages are trade names, service marks or trademarks of their respective companies.



Sometimes people ask me what it takes to run a successful business and I, who know only the media business, am always hesitant to reply. What could someone who has “merely” spent the past 25 years exclusively in publishing and broadcasting via radio, TV, print, and, most recently, online possibly tell anyone about the wider world of business – the hurly-burly of globalization, the brouhaha surrounding offshoring, the cut and thrust of M&As, hostile takeovers, poison pills, and platinum parachutes?

And so, instead I talk to them about running. It can't be sheer linguistic happenstance, I tell them, that we speak of “running” a business. Many aspects of running seem to yield insights so plainly useful to anyone engaged in business that it's a wonder (to me) that the Wharton School of Business or Harvard Business School haven't both long ago insisted that their world-class business majors also minor in competitive, long-distance running.

What business could fail to understand the significance of stamina? Or the peril, revealed only later in a race when it's too late, of having neglected to pace yourself at the beginning? How could any biz-whiz fail to appreciate the rich lesson that can be learned in a single afternoon by anyone willing to pull on a pair of running shoes and compare, first hand, accelerating downhill with accelerating uphill?

That last comparison is something that has always preoccupied me. Whether during a morning jog or a full-fledged marathon, it is an easy one to try for yourself. If you accelerate downhill, sure enough you overtake other runners who aren't doing the same; but considering all the energy you are expending, you don't exactly put a million miles between yourself and the rest of the field. Gravity, after all, is in everyone's favor, not only yours.

Now compare this with accelerating uphill. It may require a greater expenditure of energy, and most likely of willpower, but the effect it has on your position in a race is almost always transformational. You don't open up just a gap, you open up a chasm. So the extra cost seems more than comfortably offset by the benefit.

During the current economic cycle in the U.S. – tough, and persistently an uphill struggle – I have kept an eye out for businesses that nonetheless have been accelerating. *These* are the businesses in which, as a runner myself, I

feel fully confident investing. A management team that understands how disproportionately great the impact of accelerating in the bad (“uphill”) times can be, is also highly likely to have mastered some of the other tricks of the competitive running trade, such as capacity building, developing stamina, and ensuring equipment is in tip-top shape.

I was recently in New York City and found myself wanting to retrace the last hour of a N.Y. Marathon I ran a couple of years ago in which I failed so miserably to practice what I preach that I was pipped to the Central Park post by no fewer than 15,654 runners – including (the ultimate indignity, causing my four children to disown me) the rapper P. Diddy...by 17 minutes! As I headed uptown, to pick up the trail of the brutal last 7 miles, it all came back to me – how foolishly I'd pushed so hard in the first 13 miles that the second 13 were nothing short of hell on earth.

Second time around, of course, it wasn't hell on earth at all. I was running in the dead of night, in the chill air of a November evening that was perfect for keeping up a brisk pace – in comparison to the freak N.Y. weather of 2003 that saw temperatures on the day of the marathon soar to the high 70s, reducing heat tolerance by several orders of magnitude. More to the point, I'd learned my lesson. In running, as in business, as in life, endurance isn't enough on its own. Nor is perseverance. To perform *competitively*, and to succeed, what's needed is an acute sense of when it is suitable to push and when not. In a marathon that “when” may be a function of the temperature of the ambient air; in business it may be a function of the temperature of the overall economy or of a particular vertical.

The businesses that are pushing themselves hard now, as I say, are the ones that will win the Race to Greatness in the next economic cycle, which for readers of *JDI* will most likely be characterized by “Web 2.0”-like characteristics. Think Google, not Microsoft. Think Salesforce.com, not Siebel/Oracle. Think Jeremy Allaire's Brightcove, not NBC.

These companies have all, remarkably, been accelerating uphill for the last few quarters. Just imagine what kind of speed they're going to be capable of achieving when they hit the flatter road that seems now to be rushing toward us all. ☛

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



DESKTOP



CORE



ENTERPRISE



HOME



ALTOVA®

semanticworks™
2006

Give more meaning to the Web

Discover SemanticWorks™ 2006, and
put the Semantic Web to work for you.

All new SemanticWorks 2006:

- Graphical composition of RDF, RDF Schema, OWL Lite, OWL DL, and OWL Full documents
- Intelligent entry helpers
 - Automatic syntax and semantics checking
 - Conversion of visual designs to RDF/XML or N-triples

Altova® SemanticWorks™, the brand new graphical Semantic Web development tool from the creators of XMLSpy®, lets you unearth the full potential of your Web-based and knowledge-management applications.

Visually design RDF instance documents, RDF Schema vocabularies, and OWL ontologies then output them in either RDF/XML or N-triples formats. SemanticWorks makes the job easy with tabs for instances, properties, classes, etc., context-sensitive entry helpers, and automatic format checking.

Wise up to the Semantic Web!

**Download SemanticWorks™ 2006
today: www.altova.com**

JDJ contents

JDJ Cover Story



Table Layout

54

Replace your
GridBagLayout code

by Phil Herold

Features

20



Integrating AJAX with JMX

by Graham Paul Harrison

FROM THE GROUP PUBLISHER

Accelerating Uphill

by Jeremy Geelan

3

VIEWPOINT

Sun's "Welcome to Java" Doormat Needs Some Zing

by Dave Paules

6

ENTERPRISE VIEWPOINT

Managing the Stack

by Ajit Sagar

10

PERFORMANCE

Managing Java Performance Across the Application Life Cycle

Adhere to best practices

by John Reichard

12

YAKOV'S GAS STATION

Hangover Thoughts About the Web and AJAX

Who needs Web applications?

by Yakov Fain

18

CORE AND INTERNALS VIEWPOINT

Annotations, Friend or Foe?

by Michael Yuan

30

SECURITY

Entitlement to Data

Based on user profile information and
an O/R mapping tool

by Michael Poulin

40

GENERAL JAVA

Developing in Java 5

Use the new set of tools wisely

by Roberto Scaramuzzi

44

DESKTOP JAVA VIEWPOINT

Do Strongly Typed APIs Help Ensure Java's Survival or Extinction?

by Joe Winchester

48

JAVA DESKTOP

Extending Rich GUI Clients with Jython

Implementing a solution

by Hayden Marchant

50

FEEDBACK

Letters to the Editors

60

JSR WATCH

2005 JCP EC Election Final Results Are In

by Onno Kluyt

62

32



Build Management Allows Developers to Develop

by Matt Gabor

David Paules
Guest Editor

Sun's "Welcome to Java" Doormat Needs Some Zing

Ever hear the phrase "the interface IS the system?" It implies that what people perceive a software system to be is largely determined by how the system looks and how they rate their experiences interacting with it. Is the system aesthetically pleasing? Were simple operations simple and were complex commands easy? Was information organized logically? Was the system well behaved and helpful with tips, prompts, and feedback?

Users today expect a lot of functionality out of graphical desktop applications and they expect a level of "good practices" compliance as well. The Java platform is a wonderful developer environment, providing many basic necessities that let you build the application of your dreams. Java includes support for a logging framework, a print framework, an undo framework, a menu framework, a preferences framework, and a graphical interface framework.

Using all these frameworks, you can build very creative pieces of software. But words are very important here: "can build" and "frameworks." While this approach lets a developer toss together the systems he wants and the way he wants them used, it doesn't solve the problem developers face producing robust Java apps with all the

trimmings quickly and consistently. Imagine that Java apps were recognized by their simplicity, power, and sexiness rather than their scalable, modular design.

So what's my beef you ask? Simply this: Every line of code, no matter how simple it is to write is still a line of code that has to be thought about, written, tested, and maintained. Java developers today still struggle with features that most users consider basic to a rich application. A great example is layouts, borders, and resizing. Does anyone take for granted that their layout will look great the first time and every time under every situation?

So what kind of advanced features do I think should be included – and this is key – without developer intervention – to leverage them? Below is my list. Some are component enhancements, while others are complete subsystems.

- Default application hierarchy like Studio, SDI Application, MDI Application, all with a well-defined "plug-in" architecture for future extensions.
- Default command architecture to simplify display and control of menu, toolbar, and context menu options and their associated actions.



—continued on page 8

“While we wait for Mustang (in mid-2006) to improve the baseline, we can ease the burden of building feature-rich applications”

Dave Paules works as a sales engineer in the Technology Transfer Center of Excellence Group at Quantum Leap Innovations, Inc. He has a BS in computer science from the University of Delaware and roughly 7 years experience in software development with Java desktop and Java Web technologies.

dnp@quantumleap.us

President and CEO:

Fuat Kircaali fuat@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

Accounting

Financial Analyst:

Joan LaRose joan@sys-con.com

Accounts Payable:

Betty White betty@sys-con.com

Accounts Receivable:

Gail Naples gail@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

IDJ Store Manager:

Brunilda Staropoli bruni@sys-con.com

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.



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.
© 2005 InterSystems Corporation. All rights reserved. InterSystems Caché is a registered trademark of InterSystems Corporation. 10-05 CacheInno12JaDeJo

—continued from page 6

It allows these commands (and their associated metadata like images, accelerator keys, etc.) to be read from disk so it's easier for developers to add new commands without recompiling.

- Default menus and layout (file open, close, save, save as; recently opened files; edit undo, redo, cut, copy, paste, paste special, find, find again; dynamic window menu with the current window indicated and an ordered list of the windows opened by the time they were opened; preferences; about...). Configuring these menus should still be possible for different platforms (Mac OS X versus Windows/Linux, multiple icon sets, multiple color depths, multiple button sizes).

ing support on JTables is scheduled for Mustang.

- Auto-computed initial JSplitPane sizes based on the left- and right-panels contents. Toss a scrollpane into one of these and the story is even more complicated. These complications should simply not exist for developers.
- Automatic storing of any JTable dimensions, SplitPane dimensions, window/dialog locations in application specific preferences (registry for windows, .plist for Mac OS X).
- Exact Windows look-and-feel, complete with support for themes! Java's emulation of Windows is consistently incorrect for every version of Java ever released. Do it right the first time!

to remain consistent and the application robust. This is tedious and requires more coding to insulate data models adding to application memory and processing bloat.

- Spell checker and text highlighting on JTextAreas, JTextComponents, and JEditorPanes with user-modifiable dictionaries (so the application becomes smarter as the user uses it)
- Automatic “document has been modified” detection and visual feedback. Automatic “do you want to save changes” if the user closes the document window when the modified indicator is true.

While we wait for Mustang (in mid-2006) to improve the baseline, we can ease the burden of building feature-rich applications. There are several Open

“ Every line of code, no matter how simple it is to write, is still a line of code that must be written, thought about, tested, and maintained ”

- Built-in ability for the user to record macros, play them back, add a toolbar button, attach an accelerator key, etc.
- Component sizes... How many times do I have to wrestle with “appropriate” size issues versus a more complete feature set in my application?
 - Fast, live resizing of the contents of JFrames, JInternal Frames, JToolbars, JSplitPanes, JDialogs, JTable columns/rows, etc.
 - Auto-computed initial sizes on JTable columns and rows based on their data and policies on when to recompute them (when new data added, but not if the user has modified the column or row, etc.). Sorting and filter-

- User-driven PDF creation from any component.
- Default dialog for toolbar customization via drag-and-drop commands.
- Auto-logging of events to assist developers in deployment debugging.
- Automatic undo-redo where possible, e.g., text components, preference panels, canvases
- Smart bindings layers to help tie multiple UI elements together with data models. For example, if I want to show what's selected in a table in an associated JLabel as well I don't want to write the error-handling code around “nothing is selected” for the JLabel

Source solutions tackling various sex-appeal aspects of Java desktop applications on the *java.net* web site.

- **JGoodies Looks** includes ClearLook technology to help developers remove redundant borders at design time.
- **WinLAF** fixes many errors in Sun's default Windows XP look-and-feel implementation.
- The **SwingX project** includes Open Source advanced functionality widgets like a standard find dialog or an improved JScrollbar.
- **SwingLabs**, a community whose mission is to “significantly reduce the effort and expertise required to build rich data-centric Java desktop applications.” ☛

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



Automated Software Error Prevention™

Ajit Sagar
Enterprise Editor

Managing the Stack

As the complexity of enterprise applications grows with the increased offerings in the Java platform, the management of the different building blocks that constitute the application also becomes very complex. The challenge in managing applications in the enterprise is posed from many fronts. Organizationally, the corporation has to decide whether they manage all the IT services for the applications in-house, or leverage the benefits of outsourcing to meet the needs of their environment. Typically the hosting of applications and the management of the lowest tier of the stack – the hardware – is outsourced. Next comes the management of the OS itself – administration of releases, patches, configuration, etc. Above this layer is the application server layer. While the administration of the app server is closely tied to the OS, it has its own nuances and warrants a separate treatise. Above this layer is the management of the application. And finally, as the Java EE platform converges around SOA, the management of services is at the top of the stack. While there are many other layers that actually exist in a typical enterprise, for the purposes of this discussion let's concentrate on the ones mentioned so far.

When it comes to managing hardware, typically the hardware vendor (or supporting vendors) provides the tools to do so. IBM is the most prevalent example of a company that still operates the IT departments of a large percentage of companies that run Java EE applications. In the case of IBM, the Tivoli suite is commonly used to manage the lowest layer of the stack. OS management tools are typically provided from the OS vendor. However, offerings from other vendors allow a more "adaptive approach" to managing the OS releases, patches, and the network

of deployments. An example of such a company is Aduva (www.aduva.com) with their OnStage product. Aduva's product automates the system deployment and management of Linux- and Solaris-based systems and applications, with one of the main features being the capability to learn, compile, and utilize customer-specific system environment dependencies.



As we move up to the app server layer, there is an obvious paradox. While all the Java EE app servers are united in their support of the ever-evolving Java standards, the management of each server is, of course, unique to the vendor. In the app server marketplace, there is a large variance in the maturity of the offerings provided by the vendor to support the product. For example, while IBM has lagged significantly in their support of the Java APIs until they caught up with WebSphere 5, they have always had ample support for managing the administration of their product. This is not surprising, as the support and professional ser-

vices arm of IBM is quite formidable. On the other hand, BEA is one of the companies that have always been at the forefront as far as the Java platform is concerned. However, the administration tools have been quite primitive, and there is no real professional service arm to brag about. This has created an opportunity for other players to grab a chance to provide support for their substantial user base.

ARCTURUS Technologies, Inc. (www.arcturustech.com), is a vendor that saw the opportunity and created a solution to adaptively monitor and tune WebLogic environments through their AutoPilot product. Similar to what Aduva does in the Linux OS space, ARCTURUS' product uses a knowledge engineering base to proactively configure and reconfigure WebLogic deployments.

Let us take a look at the top layer of our stack in the remaining space on this page. Management of Web services is an obvious challenge in the brave new service-oriented world. While the standards around this space in the Java and .NET platforms are maturing, the ability to adapt to change in the service configurations is something that cannot be addressed by basic standards. In this space too, vendors are coming up with offerings to ease the pain. An example is RedRabbit Software (www.redrabbitsoftware.com) with its Corona Enterprise Suite, which provides features for management and "self-healing" of service-oriented applications.

The challenge for most enterprises is to choose the optimal mix of vendor products to address the needs of their particular stack, and to be able to adapt organizationally to make the best use of the offerings that they buy from these vendors. ●

Ajit Sagar is a principal

architect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has been working with Java since 1997, and has more than 15 years experience in the IT industry. During this tenure, he has been a programmer, lead architect, director of engineering, and product manager for companies from 15 to 25,000 people in size. Ajit has served as JDJ's J2EE editor, was the founding editor of XML Journal, and has been a frequent speaker at SYS-CON's Web Services Edge series of conferences. He has published more than 100 articles.

ajitsagar@sys-con.com



100 PROCESSOR CORES • 32GB HEAP • PAUSELESS GARBAGE COLLECTION • OPTIMISTIC THREAD CONCURRENCY

EXTEND JAVA™



50% LOWER SYSTEM MANAGEMENT COSTS • 0% NEED TO CAPACITY PLAN AT APPLICATION LEVEL • 300% HARDWARE UTILIZATION IMPROVEMENT

UNLEASH APPLICATION INNOVATION WITH THE POWER OF MULTICORE AND OPEN SOURCE

Introducing the \$59K Azul CentiCore Solution

- » Enable massively scalable applications
- » Simplify application development & testing and reduce time to market
- » Eliminate garbage collection pauses
- » Unshackle application development innovation
- » Leverage large memory heaps

For more information on how you can develop flexible applications
and run Java the way it was intended, visit

www.azulsystems.com/developerfreedom



Managing Java Performance Across the Application Life Cycle

by John Reichard

Adhere to best practices

A lot depends on J2EE applications – your company's future, for instance. Ensuring the performance of complex enterprise applications built on the J2EE architecture can be difficult. To meet that challenge, managing these strategic assets needs to be looked at as a process that spans the J2EE application life cycle.

End-user experience is everything today. Poor application performance degrades end-user experience; lack of availability eliminates it altogether. External users can click away to the competition. The business suffers lost transactions and revenue, customer frustration, and a poor reputation for availability and online presence. Internal users are also affected, as employee productivity plummets.

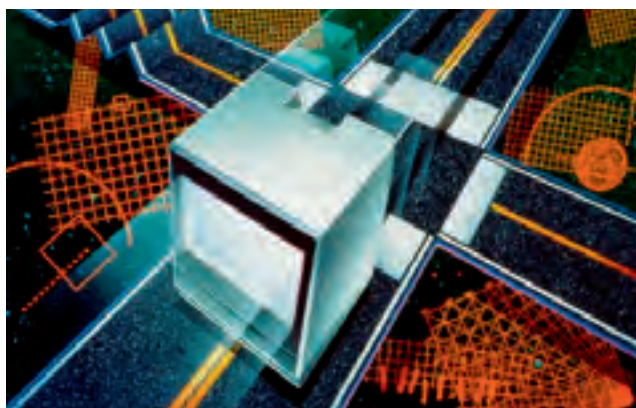
Business reliance on live applications makes it imperative that software performance and availability problems are resolved more quickly than ever before. But why do performance problems happen in the first place? Why do applications that are tuned satisfactorily in development develop performance problems later in the application life cycle? Why is it so difficult for IT operations to solve J2EE application performance problems in production?

Developers, QA testers, and IT operations analysts have accommodated newer technologies like Java, albeit within the context of their traditional IT silos. Development now creates distributed applications with Java technology; QA tests these applications with Java testing tools; yet IT operations monitor these applications with traditional device-centric monitoring tools.

The communication barrier between IT silos becomes painfully obvious when a J2EE application performance problem arises in production. IT operations may be alerted to a performance problem through the help desk or an alert from a monitoring agent. Those responsible diligently check network and server status, CPU, and memory

utilization as well as other components of the infrastructure to ensure that everything is normal. If the IT infrastructure is operating within acceptable service levels, the application itself or its runtime environment are blamed. This is often due to a lack of understanding of the J2EE application or how to diagnose a performance problem that has no infrastructure symptoms.

Eventually a triage meeting convenes in which stakeholders attempt to isolate the problem to a particular device or infrastructure component. Since no single IT staff member has the tools or ability to troubleshoot a multi-tiered J2EE application performance problem, the natural tendency is for each member to demonstrate that their own part of the infrastructure is performing properly. The servers are healthy, the network is fine, the database is okay,



This puts IT operations at a disadvantage when dealing with J2EE applications, mainly due to the lack of visibility into the Java runtime environment. The Java virtual machine appears as a “black box” to traditional IT monitoring tools, which makes diagnosing J2EE application problems problematic. As a recent Gartner report aptly stated, there is a “continuing J2EE skill deficit among IT staff, which has posed a major challenge to the effectiveness of IT operations groups.”



John Reichard is a senior technical specialist for Compuware Corporation.

john.reichard@compuware.com

“Business reliance on live applications makes it imperative that software performance and availability problems are resolved more quickly than ever before”

Get the complete picture



ILOG JViews Family of Products Advanced Graphic Components for Java

Quickly build high-performance displays and interfaces.

Telecom / Datacom / EMS / NMS

NEW Mapping for Defense: C4ISR, REP, COP

Asset Management Map Displays

BPM Modeling and Monitoring

Network Diagramming

Data Charting

Gantt Scheduling

Custom Dashboards and Consoles

Reduce development cost, time and risk.

Eclipse Integration

JSF Thin-Client Support

Open APIs

Easy Designers

Real-time Data Connectivity

Highly Scalable

Standards Support

Comprehensive Services & Support

Learn more. View on-demand webinars. Test-drive an Eval.

ILOG JViews Family of Products

NEW ILOG JViews Maps for Defense

ILOG JViews Telecom Graphics Objects

ILOG JViews Diagrammer

ILOG JViews Charts

ILOG JViews Gantt

ILOG JViews Maps

jviews.ilog.com

mapsfordefense.ilog.com

jtgo.ilog.com

diagrammer.ilog.com

charts.ilog.com

gantt.ilog.com

maps.ilog.com

www.ilog.com



Changing the rules of business™

and therefore the problem must be with the application. Yet, developers and testers counter with evidence that the application has passed functional and load testing, and has been performing just fine in production until now.

Triage sessions like this occur in corporate IT departments around the globe on a regular basis. They are unproductive because each stakeholder views the problem through the lens of their own domain-specific tools and knowledge. Creating and sustaining application performance and availability in today's complex distributed computing environments calls for a life-cycle process to manage application performance, and the proper tools to solve problems quickly through collaboration across IT disciplines.

The Application Performance Life Cycle

In order to develop, deploy and maintain high-performing applications, organizations must integrate performance into their application

In practice, the activities are more numerous and detailed, but the preceding list should provide a reasonable high-level understanding of the performance life-cycle approach.

During the planning phase, line-of-business stakeholders normally define application needs and objectives in terms of business processes and functions. Such objectives may sound something like this: "maximum response time of two seconds for a customer lookup transaction"; "must be able to support 1,200 internal users and up to 30,000 external users simultaneously"; or "must be able to process 85,000 point-of-sale transactions per minute from 2,100 retail locations."

Well-defined requirements are critical because they drive all subsequent phases of a project. Whether you are building a house, a road, or a J2EE application, requirements dictate the desired end result. With a house you might specify the number of floors, rooms, and windows while

design specifications. The knowledge and expertise of the software development team is arguably one of the most critical determinants of an effective implementation. At this stage in the life cycle, bad things can happen to good applications.

Poor coding techniques can easily and transparently introduce performance-robbing side effects into the application code base. A poorly coded application can deliver 100 percent of required features along with hidden performance bottlenecks, memory utilization problems, and potentially fatal thread synchronization issues. Such problems may even slip through QA testing unnoticed, until load testing or live production use exposes their presence. By that time, the cost and business impact of finding and fixing problems in the application code is significantly higher than if they were corrected in development.

Performance tuning an application is a good development practice for ensuring that code executes quickly with no significant bottlenecks.

“In order to develop, deploy and maintain high-performing applications, organizations must integrate performance into their application life cycle”

life cycles. This requires enhancing software engineering and IT practices to include specific performance-related activities at each appropriate stage of the life cycle, and using the right tools to facilitate those activities.

Performance-related life-cycle activities should include:

- Performance objectives and requirements definition
- Architecture and design reviews for performance
- Performance testing, tuning, and optimization in development
- Test case and test suite timing analysis in QA
- Preproduction load testing and performance baselining
- Application service-level specification and monitoring
- Production-level application performance management

with a software application, well-defined performance requirements should specify throughput, response times, scalability, and so on.

During the architecture and design phase, technical performance requirements are further decomposed into elements of a design proposal. For example, if a three-tier architecture is proposed, the requirement for a two-second customer lookup transaction must be broken down into design criteria for the presentation layer, business logic, and database access.

Development

While good architecture and design create a foundation for good application code, it is the work of developers to actually implement application features and functionality as specified in the product requirements and

Similarly, memory profiling an application in development is effective for ensuring correct and efficient use of memory resources. Thread synchronization analysis is a third development-specific task that can help optimize runtime performance and avoid potentially fatal thread deadlocks and race conditions.

Platform and configuration differences between development, QA, preproduction, and deployment environments dictate the type of tuning effort that is appropriate at each stage. Without a reasonable set of guidelines, it is possible to spend too much time tuning and optimizing during the development cycle. In a three-tier J2EE application, all of the presentation-layer components can and should be tuned with precision by the time the code is feature-complete. At this

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

Formula One e.Spreadsheet Engine

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

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

- Excel-enable your Java applications
- 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 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.



Automated load testing tools not only require far fewer humans than manual testing, but also provide the opportunity to stress-test an application by scaling the number of virtual users to a level beyond anticipated use. The result is a very powerful testing solution that no enterprise software organization should be without.

Production Deployment

Performance problems that arise in production have an immediate impact on the business, and need to be dealt with quickly and effectively. Unfortunately, once an application goes live, it is in the domain of IT operations, out of the reach of developers and their application-centric tools. Network analysts, systems analysts, and other support staff monitor

Application Performance Management

Once an application or portfolio of applications has been deployed with acceptable performance results, ongoing monitoring and management of applications and infrastructure is an important factor for documenting service-level agreement compliance and ensuring sustained application performance. Although this topic is out of scope for this article, it is certainly worth mentioning as an epilogue.

Many organizations measure service only at the device level, leaving performance issues undetected until users begin to complain. When service delivery is approached as an integrated whole, IT organizations must not only manage infrastructure performance, but must also manage application-service levels in order

To make the transition from reactive to proactive, IT organizations require a level of end-to-end application insight and analysis that is attentive to business requirements and accurate in predicting performance levels. Performance monitoring tools with predictive analysis features enable IT organizations to assure predictable service levels through a “what if” analysis feature that calculates and displays the impact of various performance adjustments on a transaction’s response time. This technique allows IT planners and analysts to tune parameters such as network bandwidth and latency, server load, and application turns in order to make necessary performance adjustments before application deployment, or to validate corrective actions made in production.

“ Performance problems that arise in production have an immediate impact on the business, and need to be dealt with quickly and effectively”

the infrastructure in which the application runs, and possibly even the application server container in which it runs, but have little knowledge about the health of the J2EE application.

When a production application falls below required service levels, an alert may be triggered or calls may begin to flood the help desk. In either case, IT analysts take notice. They look at server, database, and network utilization, but rarely come up with a root cause in short order. Once they figure out that everything seems to be “performing normally” in the infrastructure, the development organization is consulted. While operations can confirm that an application has slowed to a crawl, there is usually very little useful information they can offer the developers.

Application performance monitoring tools are ideally suited to bridging the gap between IT operations and development in application triage scenarios like the one described above. They provide continuous monitoring of J2EE application performance and resource utilization without impacting production-level performance.

to meet business demands and priorities. To accomplish this, IT organizations need proper tools to:

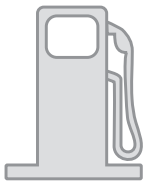
- Measure end-user experience of application availability and response times
- Prioritize performance issues based on their business impact
- Systematically resolve problems via accurate analysis of performance issues
- Monitor and analyze application performance automatically and continuously across the client, network, server, and database tiers
- Capture performance analysis information at the time of service-level exceptions
- Integrate all monitoring and measurements into useful, management-level reports that expedite service and support, and reduce confusion and delay.

The hallmark of an exceptional IT organization is its evolution from reactive application management to proactive service assurance.

Conclusion

Creating and sustaining application performance and availability in today’s complex distributed computing environments calls for a life-cycle process to ensure application performance, and the proper tools to solve problems quickly through collaboration across IT disciplines. Enterprise application performance planning must begin at the earliest possible stage, preferably at the business-unit level, even prior to technical requirements and analysis. Clearly defined performance requirements are the best insurance that application performance will be considered from architecture and design, through development and QA, and on into the production environment.

Software defects, both functional and performance-related, are extremely expensive to fix once an application has reached production. By adhering to best practices at each stage of the application life cycle, application performance can be optimized properly at each stage, avoiding costly downtime and application brownouts. ☺



Hangover Thoughts About the Web and AJAX

by Yakov Fain

Who needs Web applications?

Yesterday, we celebrated the birthday of my employee Alex in a fancy Russian restaurant. If you haven't tried it, go there – once. The party started late, and I've never seen such a variety of food on the table at the same time (they call this setup “bratskaya mogila,” which means “mass grave”). After five shots of straight vodka, we enjoyed a Broadway-type show, and then more drinks and food. Anyway, this morning the last thing I wanted to do was drive to my gas station. Last time I selected a Java Web application framework (<http://java.sys-con.com/read/136518.htm>) and for a second, I regretted that I hadn't implemented a Web application. If I had, I could have opened a Web browser and checked on the business without leaving home. At the moment, I was pretty sure there were only two types of users that could appreciate Web applications:

- Sober people who want to buy goods or use services offered by companies like Amazon, Ebay, Google, CNN, or Playboy
- Drunk owners of small businesses



Yakov Fain is a senior technical architect at BusinessEdge Solutions, a large consulting and integration firm. He authored the best selling book “The Java Tutorial for the Real World,” an e-book “Java Programming for Kids, Parents and Grandparents,” (smartdataprocessing.com) and several chapters for “Java 2 Enterprise Edition 1.4 Bible.”

He leads Princeton Java Users Group.

yakovfain@sys-con.com

But after a while I thought to myself, “Not just a Web application would let me connect to my business from home.” I can create a Java Swing client that will connect, say, to an RMI server. And it won't cost me a penny because RMI comes packaged with Java! Even Open Source application servers may be overkill for a small business. Besides, deploying Java clients on all of my computers can be automatic using Java Web Start (JWS) technology (this will be my answer to the zero-deployment arguments of those who like thin Web clients).

Remotely Delivering Applications with Java Web Start

Let's say I've created a front-end Java application and want to deploy it on

all three of my business PCs and two of my home computers. With JWS, I can automatically deploy Java applications (not applets!) that permanently reside on these computers. Every time a user starts the application, it automatically connects to a central server, compares the local and remote versions of the application, and downloads the latest one if needed. It can also automatically download JRE. Java comes with a so-called Java Network Launching Protocol (JNLP) API, and deploying a JWS application consists of:

- ✓ Configuring the Web server so it can support JNLP files. Free Web servers are readily available (see <http://httpd.apache.org/>)
- ✓ Creating a JNLP file (it's a simple XML file describing your application)
- ✓ Uploading the application to the Web server to a location specified in the JNLP file
- ✓ Creating an HTML Web page that has a link to your application's JNLP file, for example `Start Gas Order Program`



You can read more about the Java Web Start technology at <http://java.sun.com/products/javawebstart/>

I can have a full-featured multi-megabyte Java client that's located and launched locally on each of my computers. If once in a while I need to deploy a new release of this application, I'll just upload the new JAR to the computer that runs my Web server and all client computers will update the local version of the application automatically as soon as they see the newer JAR on the server.

Thin Clients, AJAX, and a Goat

Let me tell you an old Jewish joke. A poor man comes to the rabbi

complaining that his family has only one small room, many kids, and almost no money. The rabbi says, “Take all your money, buy a goat, and keep the goat in your room. Come back in a month.”

“But, rabbi, we don't have enough space even for us,” the man said

“Just do what I say,” the rabbi replied.

A month later the man comes back complaining that the goat smells and breaks everything.

“Sell the goat and come back in a month,” the rabbi tells him.

A month later the man comes back to the rabbi with flowers.

“Thank you, rabbi! We're so happy the goat is out, now we have more room and some money!”

So what has that story to do with thin Web clients and AJAX? Everything! Since the early nineties Visual Basic and PowerBuilder programmers have routinely created rich client applications, and if, for example, they need to repopulate a part of the screen by executing some DB query when a user types a character in a text field, they just put this query in some flavor of the `ItemChangedEvent` of the GUI object.

In Java it's not as simple, but still not too bad. Just register an event listener with a window control, put the db query in one of the methods of this listener, and repopulate the screen using an event-dispatching thread.

Then the Internet rush brought in plain-looking thin HTML clients (aka the goat), which had to refresh the entire page after each request. Several years later, a complex technology called AJAX came about and now people are overwhelmed with joy when they see a portion of the Web page refreshed after typing in a single character. Wow! Isn't it time to get the goat out the room and return to good old fat Java clients? I wonder why sober application architects don't see it this way. ☺

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.**



Integrating AJAX with JMX

*Opposite ends of the
Systems Management stack*

by Graham Paul Harrison

AJAX and JMX are at opposite ends of the Systems Management stack. However, the emerging ubiquity of the AJAX model for rich browser clients has obscured the benefits the model provides in the architectural space for enhancing support patterns within the problem resolution pipeline.

This article elaborates on an architectural benefit of AJAX that lets the management state be 'broadcast' to a browser-enabled user base without waiting for a page refresh.

This architecture is an extension of a general pattern for logging JMX events and properties to a server-side log file; this variation logs or 'broadcasts' management information to the (AJAX-enabled) user base.

Emphasis is placed on the AJAX request/response model and the painting of management data to the page, together with the beautiful JMX notification framework, all neatly integrated in the middle with an instrumented servlet.

Also included is a cursory view of issues not usually covered in the standard AJAX discussion; namely, security and capacity models.

BEA WebLogic 8.1 was used as the deployment platform for the software, although the architecture and method applies to other J2EE application servers. (The source code for this article can be downloaded from <http://jdj.sys-con.com>.)



Graham Paul Harrison is the author of *Dynamic Web Programming* and a former senior consultant with BEA Systems in Europe. He now works as a freelance J2EE architect, specializing in performance.

gpharrison@btinternet.com

Key Requirements

The Systems Management stack for Enterprise Java and J2EE applications forms part of the *problem resolution pipeline* in which the Java/J2EE application interacts with a management tier to monitor potential problems such as application server thread starvation, heap overflow or stale connections to a database.

The management tier usually consists of JMX MBeans, which the application is instrumented to use, together with products such as Wily Introscope to read these JMX properties, and HP OpenView, which can be alerted from Wily if a configured threshold is violated.

One problem with this model is that server-side patterns for JMX management don't help the client at the

browser if the system is going down behind him; it's all server-centric. For example, if a new J2EE Web application is to be deployed, or the application is to shut down in a few minutes because Wily has detected a problem, the user at the browser is unaware of management events that are imminent.

By gracefully allowing the user into the problem resolution pipeline, the systems administrator can manage the end-user experience to his advantage by broadcasting management information to the user base and to some extent control the user's behavior.

To communicate management information to clients over HTTP, a problem exists: how can the management aspect send management information to an HTTP client if the user at the client doesn't overtly refresh the page using GET or POST, and if the client doesn't covertly refresh a hidden frame?

Solution Description

Implementing a simple AJAX script that will take management information in the form of an XML message from the MBean server via a servlet solves the problem. This management information must be administered and fed to all participating application servers that can receive AJAX requests.

On the server side:

- A cluster of J2EE application servers is used to service requests from browser-based users, the on-line transaction processing (OLTP) user-base (two or four servers, for example). User requests are load-balanced across this cluster (OLTP cluster) using a third-party Web server.
- A standard MBean (UserWeb) is used to hold management information, e.g., administration message plus metadata properties. The MBean is hosted on both the J2EE 'administration' server and the remaining J2EE servers in the OLTP cluster.
- On the administration server, a system administrator uses the HTMLAdaptor for JMX to set the alert status, retry interval (the interval between XMLHttpRequests), and alert message; for example, 'System Down in 10 Minutes.' The administrator then broadcasts this state to the MBeans on the managed servers, which reset their state to the master administration state.

On the client side:

- AJAX-enabled clients retrieve the status, retry interval, and message using an XMLHttpRequest, which invokes a servlet to return the relevant MBean values as an XML message.
- JavaScript on the client then parses this XML message, resets the retry interval, and repaints a part of the screen with the management message.
- After retry-interval seconds, the client does another XMLHttpRequest and the client cycle begins again.

Essential Architecture

Figure 1 shows the overall solution architecture. The essential architecture elements are described in Table 1- Architecture Elements.

JMX Notification Model

This model involves two components:

- MBean to raise events for registered listeners, both local and remote
- Listeners, which register themselves with the MBean to listen for events generated by that MBean

The first is implemented by the UserWeb MBean, the second by the ManagementListener.

JMX MBean for Managing User Information

The UserWeb *standard* MBean is a simple class that contains key properties and methods as shown in Table 2 - UserWeb MBean Properties and Methods.

Event Listener

The Singleton ManagementListener class implements WebLogic.management.RemoteNotificationListener, which extends javax.management.NotificationListener and java.rmi.Remote to allow events in a remote WebLogic JVM to be notified of remote listeners using RMI.

At application server start-up, a listener on each JVM registers itself with the UserWeb MBean on the administration server.

MBean Helper

It's best practice to use a helper class as a façade for Mbeans. This helper is invoked from instrumented code to call MBean methods.

The UserWebMBeanHelper class is used as the façade for the UserWeb MBean. The ancestor of all helpers is ApplicationMBeanHelper, which:

- Looks up the local and remote MBean servers
- Invokes those servers to get/set MBean properties and invoke MBean methods

To ensure parity, both the MBean and MBean helper implement the interface UserWebMBean.

Instrumented Servlet

An application is *instrumented* to use JMX. In AOP terms, the management *aspect* is woven into the application code. The first JMX instrumentation point for this article is an HttpServlet. This servlet is the target of the AJAX request and implements a controller pattern that can be elaborated down to handle other AJAX requests using simple request parameters.

Element	Description
Standard MBean (UserWeb)	Properties for alert status and message, with getter/setters and a method to broadcast (notify) the MBean state
MBean Helper (UserWebMBeanHelper)	Facade for instrumental code to use the UserWeb MBean
MBean Server	MBean server inside the J2EE container
Servlet (Admin.java)	Instrumented servlet. Formats the XML response based on the contents of UserWebMBean
Event Listener (ManagementListener.java)	Singleton that registers itself as a listener for events of type "alert.broadcast" with the UserWeb MBean on the administration server
Client AJAX Engine (admin.js)	JavaScript for managing the XMLHttpRequest/-repaint cycle
Client Presentation (main.jsp)	Instrumented JSP. AJAX-enables the page based on MBean properties
HTML Adaptor Wrapper (StartHTMLAdaptor.java)	Starts an HTMLAdaptorServer at listen port+100, for HTTP access to MBeans

Table 1 Architecture Elements

From an MVC perspective, the model is the UserWeb Mbean, the view is the AJAX-enabled (JSP) page, and the controller is the instrumented servlet.

Client AJAX Engine

This is a set of JavaScript functions that:

- Manage the XMLHttpRequest and response processing iterations
- Parse the XML message returned by the XMLHttpRequest
- Repaint the screen with the XML message contents

Client Presentation

This is the main.jsp page that contains the client AJAX engine and repaintable section.

Sequence in Action

Essentially, the server sequence is concerned with managing the setting of the management properties and broadcasting these properties to all interested (listening) JVMs. The client sequence is concerned with retrieving these properties and repainting the HTML page with important management information at management-specified intervals.

JMX Notification (Server Sequence)

- UserWeb MBeans and MBean event listeners are created and registered at application server start-up using start-up classes
- The administrator sets the 'master' UserWeb MBean properties (alert message and retry interval), then broadcasts, or *notifies*, this state to the listeners hosted on the remote managed servers
- The remote listeners handle the notification by copying the master (notification) data to the local UserWeb MBean

XMLHttpRequest Poll (Client Sequence)

- AJAX-enabled clients invoke a servlet at intervals to query management state
- The servlet reads the local UserWeb MBean properties, inserts them into an XML message, and returns the XML message as an XML response to the browser client (alternative message formats are discussed later)
- The AJAX client then parses the XML document, extracts the alert message and retry interval, repaints the screen, and then uses the retry interval to set the delay for the next XMLHttpRequest

Each step is described in detail below.

Register MBeans and MBean Listeners

On each J2EE server instance, two *start-up classes* are run at server start-up:

- **ManagementStartup:** Registers the UserWeb MBean in the *local* MBean server. Startup class parameters include default settings of the alert status, as well as the MBean name and MBean class. For example:

```
<StartupClass
    Arguments="ServerName=admin,
              MBeanName=ExampleApp:Name=UserWeb,
              MBeanClass=com.grahamh.management.userWeb.UserWeb"
    ClassName="com.grahamh.management.startup.
ManagementStartup"
    FailureIsFatal="true" Name="UserWEB" Notes=""
    Targets="admin,OLTPCluster"/>
```

- **MbeanRegistrations:** Used to register a Singleton POJO, ManagementListener, with the UserWeb MBean on the administration server.

A `javax.management.NotificationFilterSupport` object is used to list the notification types that the UserWeb MBean will generate and the listener will receive:

```
// MbeanRegistrations.java
MBeanHelperFactory.getWebHelper().registerListener();

// UserWebMBeanListener.java
public void registerListener() throws UserWebException{

    try {
        // get listener and filter
        ManagementListener listener = MBeanHelperFactory.getListen-
er();
        NotificationFilterSupport filter = listener.getSupportedE-
vents();

        // get admin mbean server;
        //register the listener and filter with the UserWeb MBean
        RemoteMBeanServer rmbs = getAdminMBeanServer();
        rmbs.addNotificationListener("ExampleApp:Name=UserWeb",
                                    listener, filter, null);
    }
    catch (Exception e) {
        throw new UserWebException("Unable to registerListener: "+
                                    e.getMessage(), e);
    }
}
```

The `listener.getSupportedEvents()` method returns the following filter:

```
NotificationFilterSupport filter = new NotificationFilterSupport();
filter.enableType("alert.broadcast");
```

When `ManagementListener` is run at server start-up, a connection is made to the MBean server on the (remote) administration server and the (local) `ManagementListener` is registered as a listener on events generated by the `UserWeb MBean`, with a filter set to "alert.broadcast" event types.

Because the `ManagementListener` implements `Weblogic.management.RemoteNotificationListener`, it can get JMX notifications that are generated in either the local JVM or a remote JVM; in this case, generated in the remote administration server JVM.

Broadcast Admin MBean properties

The administration and managed `UserWeb MBeans` can be set independently, giving any one J2EE server a localized AJAX response. However, a general Operations, Administration & Support (OA&M) support pattern would set the admin MBean properties and then *broadcast* these properties to the MBeans on the remote application servers, using the Notification model, for subsequent AJAX retrieval.

Because the `UserWeb MBean` is based on `ApplicationMBean`, which extends `javax.management.NotificationBroadcasterSupport`, the infrastructure is in place for the `UserWeb MBean` to notify all listeners. Hence, the administrator sets the relevant MBean properties (using the `HTMLAdaptor`) and clicks `BroadcastState` (see Figure 2).

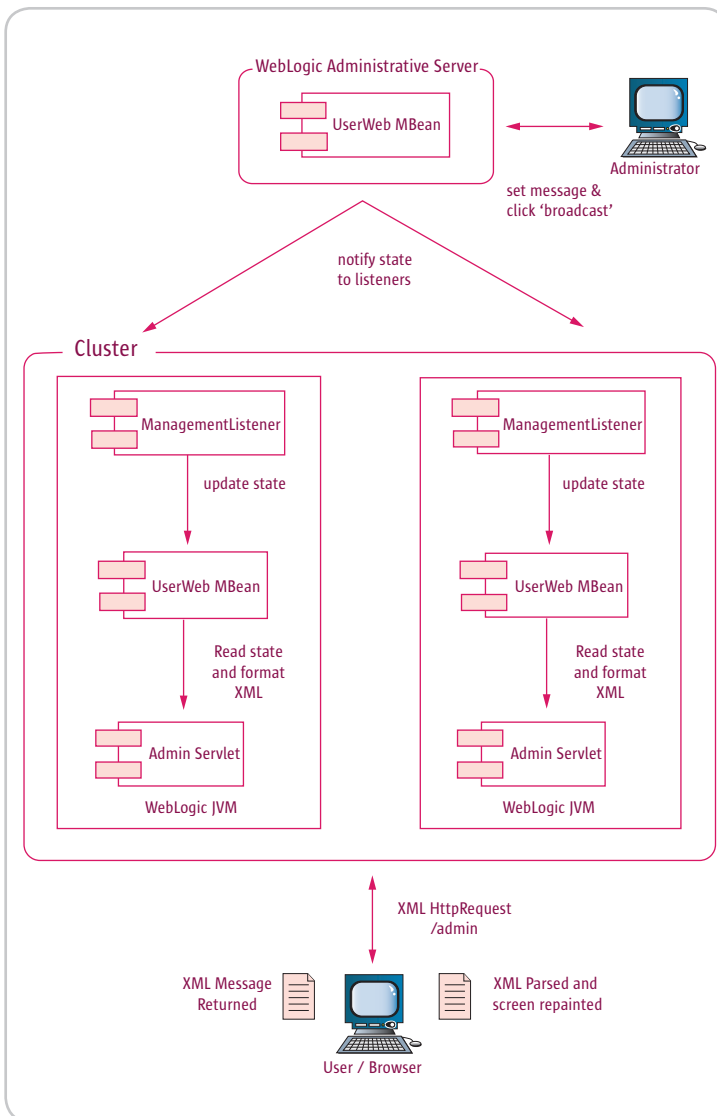


Figure 1 Solution Architecture

Visualize works of software art

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

Now supports
Java 5.0!

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

UML is a trademark or registered trademark of the Object Management Group, Inc. in the United States and other countries.

Consequently, the `UserWeb.broadcastState()` method is executed, which notifies all listeners *synchronously* with the state of the admin MBean:

```
public void broadcastState() throws Exception {
    try {
        Notification n = new Notification("alert.broadcast",
                                           "ExampleApp:Name=UserWeb", 0);
        n.setUserData(new UserWeb(this));
        this.sendNotification(n);
    }
    catch (Exception e) {
        throw e;
    }
}
```

Because the data is serialized over the network, the non-transient object graph must be serializable.

Receive Notification of MBean Props via Listener

The event listener is the `ManagementListener` Singleton. The JMX Notification framework on the administration server makes a remote call to the `ManagementListener` `handleNotification()` method in each listener on each of the OLTP cluster JVMs, which registered on server start-up:

```
public void handleNotification(Notification notification, Object
    handback) {
    System.out.println("Received alert: " + notification.get-
        Type());

    // get event userdata from notification
    Object userData = notification.getUserData();

    if (userData instanceof UserWeb) {
        // comes from destin8 Web
        UserWeb WebVo = (UserWeb)userData;
        UserWebMBeanHelper helper = MBeanHelperFactory.getWeb-
            Helper();

        // get from value object and set into local MBean using
        MBeanHelper
        helper.setAlertMessage(WebVo.getAlertMessage());
        helper.setAlertStatus(WebVo.getAlertStatus());
        helper.setCallBack(WebVo.getCallBack());
        helper.setRefreshAlertStatus(WebVo.getRefreshAlertStatus());
    }
}
```

The 'master' `UserWeb` data is set into the local `UserWeb` MBean via its MBean helper. Consequently each managed server is updated with the master `UserWeb` state.

That's as far as the JMX elements need to go.

AJAX Request of Management State

The browser client is AJAX-enabled as follows:

- `main.jsp` – instrumented JSP page that checks the (JMX) alert status and polls the server for alerts. It includes `admin.js`
- `admin.js` – JavaScript utilities that use `XMLHttpRequest` to poll the server for the management state, parse the XML response, and repaint the 'status' area of the screen

JavaScript (described below) is included in `main.jsp` as follows:

```
<script type="text/javascript" src="./js/admin.js" ></script>
```

Rather than poll continuously, we will only poll if alerting is enabled. We use the `UserWebMBeanHelper` to check this status. If enabled, the JavaScript function `initAdmin()` is invoked when the page loads:

```
<%
if (MBeanHelperFactory.getWebHelper().isAlertEnabled()) {
%>
    <body bgcolor="#F4FFE4" onload="initAdmin();">
<%
} else {
%>
    <body bgcolor="#F4FFE4">
<%
}
%>
```

Element	Description
AlertEnabled	True if AlertStatus > -1
AlertMessage	The message the user will see repainted on the screen
AlertReady	True if AlertStatus > 0
CallBack	Number of milliseconds between each XMLHttpRequest
BroadcastState	Method that notifies all registered listeners on local/remote JVM's with an event (alert.broadcast) and passes the MBean state as event data

Table 2 UserWeb MBean Properties & Methods



Figure 2 MBean view using HTMLAdaptor



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/jp12.htm or call (301) 838-5560.

 **JReport®**
J I N F O N E T S O F T W A R E

The repaintable 'status' area of the screen is defined as follows:

```
<span id="adminBanner" class="style1"></span>
```

'adminBanner' will be used to identify the repaintable area when the XML response is parsed and the message extracted. The `initAdmin()` method schedules a JavaScript method, `trapAlert()`, to be executed after `callbackTimeout` milliseconds:

```
function setCallback() {
    callBack = setTimeout('trapAlert()',callbackTimeout);
}

function initAdmin() {
    setCallback();
}
```

It's the `trapAlert()` method that initiates the `XMLHttpRequest` request with the now familiar ring:

```
function trapAlert() {
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }

    req.onreadystatechange = processRequest;
    req.open("GET", './admin?reqid=0', true);
    req.send(null);
}
```

HTTP GET is used to read data (only a small request parameter is used), and the admin servlet is targeted. The request is *asynchronous*, and when the request state changes, the `processRequest` JavaScript function is invoked:

```
req.onreadystatechange = processRequest;
```

It might seem reasonable to wait for a response before continuing processing. However, you run the risk of having your script hang if a network or server problem prevents completion of the transaction. An asynchronous call with the `onreadystatechange` event is more resilient.

As the request cycles through to completion, the `processRequest` event handler is invoked:

```
function processRequest() {
    if (req.readyState == 4) {
        if (req.status == 200) {
            parseMessages();
        }
    }
    ...
    setCallback(); // only do this when complete
}
```

Listing 1 shows the available status codes. When the request is complete and HTTP status code 200 (OK) is returned, the `parseMessages()` method is called to extract the data from the XML message. Then the `trapAlert()` method is rescheduled again. If the XML response has a different retry interval, this will have been set by the `parseMessages()` function.

Parse XML Response and Repaint Screen

The `parseMessages()` function first extracts the XML response

```
response = req.responseXML;
```

and extracts the elements for alert status, alert text, and retry interval:

```
itemStatus = response.getElementsByTagName('status')[0].firstChild.nodeValue;
itemText = response.getElementsByTagName('textBody')[0].firstChild.nodeValue;
callbackTimeout = parseInt(response.getElementsByTagName('callBack')[0].firstChild.nodeValue);
```

The alert text is then repainted on to the `adminBanner` document element (see above):

```
document.getElementById("adminBanner").innerHTML = itemText;
```

The alert message appears on the screen as shown in Figure 3.

Servlet Formats the XML Response

For the browser to display management alerts to the user, `XMLHttpRequest` is used to request the management state. When the browser sends the request, the servlet uses the MBean helper to check the alert state and, if an alert is available, constructs an XML document as a response.

If there's no state to return, the response status is set as follows:

```
response.setStatus(HttpServletResponse.SC_NO_CONTENT);
```



Figure 3 Repainted screen

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.

Contact us at sales@mapinfo.com

 **MapInfo.**
Be Location Intelligent™

Otherwise the text/XML response type is set:

```
response.setContentType("text/xml");
```

Listing 2 shows the servlet method in full.

When the servlet is invoked and the XML content returned, the console should print:

```
Received alert: alert.broadcast
<message><status>1</status><textBody><![CDATA[System Down in 10
Minutes]]></textBody><callBack>10000</callBack></message>
```

Capacity Modelling and Security

Because AJAX opens up the architecture in interesting ways, two key areas require elaboration:

- Capacity Modelling
- Security

Caching and response message type (XML or text) can also be important.

Capacity Modelling

AJAX-enabled rich clients won't necessarily submit requests any more frequently than before. But with XMLHttpRequest executed asynchronously in the browser, the number of HTTP requests to the server increases in line with the retry interval.

- Retry interval (Think Time) = 20 seconds
- Number of connected users = 5000
- Transactions per second (TPS) = $5000/20 = 250$

We expect an extra 250 requests (transactions) per second generated from the HTTP user base.

Of course, it depends what these requests do at the server to increase latency in response time. In our example, each request must look up MBean properties and format an XML response, but the response is very small and the MBean is in local memory. With each Web server thread able to handle approximately 200 GET requests a second, and the user requests load balanced across a J2EE server cluster of perhaps 200 threads, the increased loading isn't significant.

When modelling AJAX architectures, increased load injection can be offset by reduced bandwidth, since the response contains data only in contrast to data plus mark-up.

Security

Suppose you only wanted users in the WebUser group to access the Admin servlet?

If only authenticated users can access the admin servlet, then the XMLHttpRequest will run as that user *if that user has authenticated*.

For example, once user Joe logs into the application, and Joe is a member of group WebUser, the XMLHttpRequest will be able to invoke the Admin servlet.

Adding the following code to the admin servlet would confirm the authenticated subject, returning true and Joe respectively:

```
request.isUserInRole("WebUser");
request.getRemoteUser();
```

Caching

Some users have found that IE will cache the response from the AJAX request. This could be due to browser/page settings, but a forced workaround is to timestamp the URL:

```
var urlstr = "/admin?reqId=0&ts=" + new Date().getTimeStamp();
```

To XML or Not to XML

Not to XML. Some AJAX designers happily dispense with XML and send the response as plain text:

```
response.setContentType("text/plain");
```

This clearly depends on your client-side requirement and how loosely coupled the client is from the data required. A simple text response would suffice for a text alert. However, the advantage of the XML model for this article is that the response data can be elaborated further to refine both the status and status-related data. This article demonstrates how you'd parse a more complicated response that the client may have to code to accept.

Conclusion

AJAX presents architectural opportunities that shouldn't be overshadowed by the rich-client frenzy. This article has leveraged the architectural advantage AJAX can provide while addressing the technical requirements for both capacity and security that more exotic uses of this technology will demand. ☺

References

- <http://java.sun.com/j2ee/1.4/docs/api/javax/management/NotificationFilterSupport.html> - Details the alert types functionality
- <http://e-docs.bea.com/wls/docs81/jmx/notifications.html> - Details the use of WebLogic MBean Notifications for remote listeners
- http://e-docs.bea.com/wls/docs81/ConsoleHelp/start-up_shutdown.html - Configuring start-up and shutdown classes in WebLogic Server 8.1

Listing 1: admin.js

```
var url = './admin?reqId=0';
var baseCtx;
var callBack=null;
var callbackTimeout=null;

function trapAlert() {
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }

    req.onreadystatechange = processRequest;
    req.open("GET", url, true);
    req.send(null);
}

//
```

```

// readyState Status Codes:
// 0 = uninitialized
// 1 = loading
// 2 = loaded
// 3 = interactive
// 4 = complete

function processRequest() {
    if (req.readyState == 4) {
        if (req.status == 200) {
            parseMessages();
        } else {
            updateStatus(null)
        }
        setCallback(); // only do this when complete
    }
}

function parseMessages() {
    response = req.responseXML;
    itemStatus = response.getElementsByTagName('status')[0].firstChild.
nodeValue;
    itemText = response.getElementsByTagName('textBody')[0].
firstChild.nodeValue;
    paintAlertText(itemText);
    callbackTimeout = parseInt(response.getElementsByTagName('ca
llBack')[0].firstChild.nodeValue);
    updateStatus(itemText);
}

function paintAlertText(itemText) {
    document.getElementById("adminBanner").innerHTML = itemText;
}

function setCallback() {
    callBack = setTimeout('trapAlert()',callbackTimeout);
}

function initAdmin(uri) {
    setCallback();
}

function updateStatus(msg) {
    if (msg == null ) {
        window.status = "No Alerts";
        document.getElementById("adminBanner").innerHTML = " ";
    } else {
        window.status = "Example Alert: " + msg
    }
}

```

Listing 2: admin.java (servlet)

```

package com.grahamh.management.servlet;
/**
 * Admin
 * Processes a request to get various admin states for the OLTP user
 * @author GH
 */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

```

```

import java.util.*;

import com.grahamh.management.UserWeb.UserWebMBeanHelper;
import com.grahamh.management.utils.MBeanHelperFactory;

public class Admin extends HttpServlet
{
    static final private String CONTENT_TYPE = "text/xml";

    public void init() throws ServletException
    {
        //Process the HTTP Get request
        public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
        {
            /**
             * use requestId in future to switch on admin events
             */
            String requestId = request.getParameter("reqid");
            if ((requestId == null) || (!MBeanHelperFactory.getWebHelper().
isAlertReady()) )
            {
                //nothing to show
                // there re two models - if nothing to show :
                // 1. dont send anything, or
                // 2. send a message and let the client interpret the
                alertStatus.
                // (1) used, so client just displays an alert if one
                exists.
                response.setStatus(HttpServletResponse.SC_NO_CONTENT);
            }
            else {
                // only send a message if there is an alert notification to
                send
                StringBuffer sb = new StringBuffer();
                sb.append("<message><status>");
                sb.append(MBeanHelperFactory.getWebHelper().getAlertStatus());
                sb.append("</status><textBody><![CDATA["");
                sb.append(MBeanHelperFactory.getWebHelper().getAlertMessage());
                sb.append("]]></textBody>");
                sb.append("<callBack>");
                sb.append(MBeanHelperFactory.getWebHelper().getCallBack());
                sb.append("</callBack></message>");

                response.setContentType( "text/xml" );
                response.setHeader( "Cache-Control", "no-cache" );
                PrintWriter out = response.getWriter();
                out.write(sb.toString());
                System.out.println(sb.toString());
            }
        }
        //Process the HTTP Post request
        public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
        {
            doGet(request, response);
        }
        //Clean up resources
        public void destroy()
        {
        }
    }
}

```




Michael Yuan



Annotations, Friend or Foe?

Annotation is a new Java language feature introduced in JDK 5.0. It has quickly become one of the most popular, and yet most controversial, language feature in core Java. New Java frameworks, such as EJB 3.0 and Hibernate 3.0, make extensive use of annotations to eliminate the excessive XML configuration files (a.k.a. the “XML hell” in Java EE). Those annotations significantly reduce the amount of code and configuration data, and simplify the overall architecture, making application development easier. At the same time, enterprise architecture purists are complaining that annotations corrupt the separation between code and configuration, reduce the overall level of abstraction, and create more coupling between code and external frameworks. They argue that annotations should not be used to specify relationships between application components. Who is right? My view is that both arguments have merits. But as a developer, you really should make decisions based on the project at hand.

etc.) to Java objects. You have to somehow specify the relationships and dependencies between Java objects and container services. One approach is to use API calls directly in your Java components to consume container services. Java servlets and JNDI clients are such examples. However, too many framework APIs inside a Java class make the code very difficult to read and maintain (i.e., lots of boilerplate code). So, XML configuration files are widely used to specify the relationship between objects and services. This declarative approach allows you to configure the objects from the outside instead of cluttering up the business logic code with framework API calls. In fact, a large part of the Java EE specification is to standardize those XML files. Newer frameworks such as Spring even adopt an approach to put all configuration information in XML files, so that the Java business objects have zero dependency on the external frameworks (Depen-

dency Injection via XML files). That is, you can apply services from any container to the same Java objects by just changing the XML files. This level of abstraction is great from an architecture point of view. However, it also creates huge complexity for developers and results in something known as the “XML hell.”

Using XML to configure Java code has several distinct disadvantages: since XML files are processed separately from the code, they have to duplicate information already contained in the code (e.g., class names and method names, the inheritance structure, etc.) in order to configure the code. The duplication makes XML verbose even for the simplest configuration. Such XML files are also error-prone since spelling errors are not checked by the compiler or even the deployer. At the end of the day, XML configuration files simply shift the maintenance complexity and boilerplate code from Java classes to XML files. Developers are searching for better solutions.

A popular solution that emerged from the open source community is XDoclet. It uses embedded tags in Javadoc comments to specify which container services should be applied to this class or method. Then, a Javadoc-like preprocessor takes in the source code and generates all the necessary XML files based on the relationship. Since the XML configuration files are automatically generated, it's guaranteed to be correct and consistent. The developer only needs to maintain the XDoclet tags that are inside the source code



Michael Juntao Yuan

is a member of JDJ's editorial board. He is the author of three books. His latest book, “Nokia Smartphone Hacks” from O'Reilly, teaches you how to make the most out of your mobile phone. He is also the author of “Enterprise J2ME” - a best-selling book on mobile enterprise application development.

Michael has a PhD from the University of Texas at Austin. He currently works for JBoss Inc. You can visit his Web site and blogs at www.MichaelYuan.com/.

juntao@mail.utexas.edu

“To understand which projects would benefit from annotation frameworks, we have to know a little bit of history on how annotations were introduced and evolved in the Java language”

“The standardization of annotations in specifications like EJB 3.0 will eventually make many annotations part of the Java core API”

file but don't clutter up the code like boilerplate code used to. The popularity of XDoclet is telling. It indicates that a large number of developers would prefer ease-of-development over added abstraction.

To capitalize on the success of XDoclet, Sun decided to introduce annotations as a formal language feature in Java. Instead of being a second class citizen embedded in code comments, annotations are now fully supported by the compiler, IDE tools, and are standardized into APIs. Using annotations, you can specify how container services are delivered to Java objects. For instance, the following example is an EJB 3.0 entity bean. The `@Entity` annotation tells the container that this class should be mapped to a database table. The `@Id(generate=AUTO)` annotation tells the container to automatically generate the primary ID value when the object is saved into the database.

```
@Entity
public class Reply implements Serializable
{

    private long id;
    private String name;

    @Id(generate=AUTO)
    public long getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

To use the entity bean, the `@PersistenceContext` annotation tells the container to inject a valid `EntityManager` object to the `ManagerAction` object. The `@TransactionAttribute` annotation declares that the `reserve()` method is managed by the container's transaction manager, which commits changes to the

database at the end of the current thread.

```
@Stateless
public class ManagerAction implements
Manager {

    private Reply reply;

    @PersistenceContext
    private EntityManager em;

    @TransactionAttribute(TransactionAttribut
eType.REQUIRED)
    public void reserve () {
        em.persist (reply);
    }
}
```

In addition, you can also use annotations to manage relationships between your own objects. The following example uses annotations in the JBoss SEAM Web framework. The `@In` annotation below injects the reply object from the Web context, which is probably mapped to fields in a JSF form requesting a user reply. This way, when the user changes the reply object from a JSF form, it is automatically validated and propagated to the `ManagerAction` object. Likewise, the `@Out` annotation outjects the `allReplies` variable updated in the `reserve()` method to the context so that all the replies can be displayed in a table on a JSF Web page.

```
public class ManagerAction implements
Manager {

    @In
    private Reply reply;

    @PersistenceContext
    private EntityManager em;

    @Out
    private List <Reply> allReplies;

    public String reserve () {
        em.persist (reply);
        allReplies = em.createQuery("from Reply
r").getResultList();
        return null;
    }
}
```

Those annotations eliminate boilerplate code in both Java and XML code. The annotated code is very easy to understand. However, annotations also have a few downsides. Annotation is not as expressive as XML and it's only weakly validated at compile time. For instance, you could mistakenly place the `@PersistenceContext` annotation on the reply variable and the compiler would not complain. The application only fails when it deploys. In addition, perhaps the biggest argument against annotations is that they mix configuration with the Java code, and hence tie your Java objects to the particular service framework that defines the annotations (EJB 3.0 or JBoss SEAM in our examples). If you want to switch to another framework later, you'd have to go into the Java class code and change the annotations. In contrast, XML-configured Java objects (e.g., Spring POJOs) can work with any service framework without changes to the Java class – you only need to change the XML files – due to the clean separation between code and configuration. Of course, the XML approach only has an advantage when you have far more Java code than XML code, which is *not* the case for most business applications today.

What is the verdict? If your objects have very complex relationships or if “framework independent” business logic is essential to you, you could be better off choosing an XML-based configuration framework such as Spring or JBoss Micro-container. However, if you deal with medium-complex systems and are concerned about productivity over abstraction, annotations are perfect. The standardization of annotations in specifications like EJB 3.0 will eventually make many annotations part of the Java core API. Framework independence is less of an issue in a world with standardized frameworks. We will use annotations just as we make API calls to JDK libraries today. That is, of course, the intention Sun had when it introduced annotations into the Java language. ☺

Build Management Allows Developers to Develop

Enable more flexible and manageable development environments

by Matt Gabor

Whether writing code for a small, single-platform environment or contributing to a sophisticated cross-platform, multi-language application, one truth remains consistent: a disorganized or poorly implemented build management strategy will adversely affect a developer's workflow.

Because developers possess an intimate working knowledge of the structure of their code, they are often called upon to write and maintain build scripts. When integration build problems occur during QA, their phones ring. When build problems prevent applications from rolling into production, their inboxes are flooded. These kinds of intrusions on a developer's daily coding practices may start small, but over time, as applications grow in complexity, they can bring a company's entire application development process to a grinding halt. This eventual result can be avoided by implementing a well-thought-out build-management system that empowers the developer while meeting enterprise-level requirements.

In small- and medium-sized environments, using a properly implemented in-house build system can mitigate many of the risks and challenges associated with builds. In addition, with a few simple steps, the burdensome task of developing and maintaining build scripts can be significantly reduced, allowing developers to focus on writing code rather than debugging builds. As development efforts grow in size and complexity, however, there comes a point where significant benefits can be realized by moving to an automated system. Whether using a manually scripted or automated build solution, every enterprise should strive for build consistency, portability, and repeatability.



Matt Gabor is a senior developer, Catalyst Systems Corporation. He has an extensive background in the use and development of Eclipse Plug-ins and is expert in the Openmake Build Plug-in. His technical skills includes J2EE, J2SE, and .NET development.

matt.gabor@openmake.com

Build Management Evolution

Application builds are traditionally managed using a rules-based program derived from Make, one of the oldest, best-known build tool. Make controls the generation of executables and other non-source files from a program's source files. There are many Make versions, each with a unique file syntax that precludes portability between development tools, operating systems, or even compilers within the same operating environment.

Java development required a new, platform-independent build tool, leading to the creation of Apache Software Foundation's Java-based Ant, so-named by developer James Duncan Davidson because, "...it is a little thing that can build

big things." Ant eliminates Make's platform-dependent problems and is extended not with shell-based commands but Java classes. Configuration files are XML-based, calling out a target tree where various tasks get executed. Each task is run by a Java class that implements a particular task interface.

Ant is powerful, but the XML configuration scripts can create limitations. XML does not handle conditional logic effectively and therefore it's difficult to use Ant to write "intelligent" build scripts that support robust batch processing. In addition, many development projects include Java and non-Java components that require both Ant and Make, as neither handles both languages. Scripting for the two is very different. Make scripts are the input to Make programs and dictate how to build each software component. A Make file tells the Make program which binaries are to be created from which source modules. Make rules are then "fired" based on out-of-date conditions between source and object. In contrast, Ant/XML scripting uses serial batch processing. Rules for creating Java binaries such as .jar, .war, and .ear are handled statically for each step or "task" in the XML script. Listing 1 shows the differences between Make and Ant scripts for a similar type of build task.

For either approach, the programmer must understand not only how the application is constructed, but also the specific syntax requirements of the build scripting language they are using. In addition, Make and Ant/XML scripts are not re-usable because static application information is coded into the script.

Make and Ant/XML Challenges

As client/server and Java development has evolved, so has build complexity, especially with Make. When Make is used recursively (one Make file per final target executable, or binary – the most common method of managing large build processes), an application with 50 binaries would require 50 Make files plus a "driver" Make file. The system build is completed by calling the Make program repeatedly and passing a different Make file each time. Dependency checking between the individual Make files is impossible, which means large application Make files can't be managed by a single Make file. Developers get around Make challenges through clever file-ordering to track dependencies, along with object-borrowing and multisystem parallel building techniques to reduce the associated long system build times. The process is still dif-

Complex JAVA J2EE Hosting made easy.

WebAppCabaretsm

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



JAVA J2EE-Ready Managed Dedicated Hosting Plans:

Xeon I

**SAMEDAY
SETUP**

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

\$279
monthly

Pentium 4 I

**SAMEDAY
SETUP**

**FREE
SETUP**

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

\$199
monthly
2nd month
FREE

4Balance I

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

\$1724
monthly

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

PRELOADED WITH:

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

Details:

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

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



WebAppCabaretsm

JAVA J2EE Hosting

Prices, plans, and terms subject to change without notice. Please log on to our website for the latest price and terms. Copyright © 1999-2005 WebAppShowcase • All rights reserved • Various trademarks held by their respective owners.

ficult and cumbersome – especially when there are multiple circular dependencies between executables and libraries, which require multiple builds to achieve the desired results.

Although most Ant build systems do not appear to be as complex as many Make-based build systems, it is only a matter of time. Because Ant is a relatively young technology, Ant-based build systems have not been subjected to the years of additions and changes that can corrupt and obscure what originally appears to be an elegant build scripting solution. Over the years, as Ant scripts suffer from being passed through different developers' hands, as new technology emerges that effects the way Ant scripts are coded or used, and as applications grow more complex, Ant will encounter many of the problems associated with a Make-based system.

Scripted build processes also introduce other challenges that devour developers' time. In fact, build scripting can be the most time-consuming and resource-intensive component faced by development teams. Problems scale with an environment's size and complexity. The key is to implement best practices for manual scripting starting with an in-house build system, while monitoring factors that would signal the need to move to an automated, non-scripting approach.

Solving Typical Scripting Problems

Scripting challenges are easier to solve in small, homogeneous development environments confined to a single language and target operating system. However, most companies strive to expand product reach and capabilities; as scope grows, development complexity follows. A company that lays the groundwork for an organized build process early on will better overcome future growth challenges.

The first step is to shift from a developer-centric view of builds to a team-centric view, and from the notion of scripting "my build solution" to scripting "our build solution."

Build inconsistency is the toughest problem. If developers use their own build scripts in the language or tool of choice, it can be difficult to know whether problems result from bad code or a bad build. Build administrators must standardize on a single scripting approach that best suits the language being used.

The first step to reducing build inconsistency between individual developer's build scripts is to develop build script templates that can be used as a basis for all build scripts. All builds require the same basic information: source code, compiler, and final target. Individual developers can populate the build script templates with their own build specifics, i.e., the source code location, compiler location, and a final target description. Build script templates should be well commented and clearly organized to ease the process of populating the template with build-specific information.

A major contributing factor to build inconsistency is a lack of compiler and third-party library standardization. In a disparate build environment, developers frequently build against different versions of compilers and third-party libraries. This makes it difficult to re-create builds and diagnose problems. Instead, to promote standardization, all compilers should be centralized on a network drive accessible by all developers, on a clean and "locked down" machine. The build script templates should specifically reference the standard compiler versions on the mapped network drives, ensuring that all builds occur against consistent compiler versions. All third-party libraries should

similarly be consolidated on a shared network drive so that the latest, approved versions are used.

Another commonly faced problem is a lack of build portability. Builds often work only on an individual developer's machine, which by default becomes the "production" build machine. The philosophy is, "build it where it builds and get it out the door." Obviously, this approach to builds can cause severe problems when trying to track down bugs that are discovered once an application has been released to production. To solve this problem, development teams should standardize their directory structure. All developers should work on code in the same directory structure. If a versioning or CM tool is used, pull the directory structure from it; if not, enforce strong directory conventions for all developers.

Portability problems can also be mitigated by using global variables in the build script templates that identify the root location for all source code, compilers, and common libraries. By setting environment variables, such as SRC_HOME, COMPILER_HOME and COMMON_HOME, the same build scripts should work on all machines. Using global variables in the build script templates also reduces the amount of template editing that is required by developers.

The Make code (see Listing 2) is a generic example that demonstrates the use of a standardized compiler to ensure build consistency and the use of global variables to guarantee portability. Note the global variables defined at the top of the script. Each developer can edit the values to suit his or her own workstation. In addition, the cc compiler is pulled in through a shared network drive referenced in the COMPILERDIR variable.

The Ant script in Listing 3 demonstrates a similar use of global properties to ensure consistency and standardization.

Finally, isolate the build scripts to just that: builds. Too often, "build" scripts include substantial pre- and post-build logic unrelated to the build. Pre- and post-build logic can be extremely complex, especially as an application matures and development is being performed on multiple versions simultaneously. The Ant script in Listing 4 demonstrates a build script with a very basic and generic deployment portion.

Rather than writing pre- and post-build logic within a build script (where the functionality is often limited by the scripting language or tool), place the non-build logic in external scripts. The external scripts should be written in a scalable, lightweight, and cross-platform language such as PERL or PYTHON. Tightly focused build scripts can then have built-in hooks to the external build utility. Listing 5 takes the overly complex build script of the prior example and replaces it with a call to an external script. (Listing 5 and additional source code can be downloaded from <http://jdj.sys-con.com>.)

By partitioning the build scripts in this way, developers (or build masters) who encounter build problems can drill down to the root cause very rapidly. In addition, as development grows in complexity and new languages or target operating systems are added, the in-house build utility can scale more effectively. For example, consider a C and C++ development shop that uses an entirely Make-based build system with all pre- and post-build logic written in the Make scripts. When the development shop decides to add a Java component to their application, they are faced with writing an Ant component (equivalent to their existing Make scripts) that manages all Java-related pre-build, build, and post-build logic. However, if the development shop

IT CAN THINK
AND DECIDE..
On its own.



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

has a build utility, written in PERL, that executes Make scripts limited to build execution, they only have to write Ant scripts that handle the Java builds, and can use the existing PERL framework as a basis for all of the non-build functionality.

Dealing with Growth

As development environments grow, new distributed-management requirements are introduced and problems grow more complex. Good build management approaches will, for the most part, scale effectively, but there are several new problems that are often added to the mix.

For instance, the more complex the environment, the more likely it is that there will be longer delays between when builds break and when those breaks are identified. Developers might execute builds on their own machine and perform successful unit testing, only for the integration build to break when the developer's code is introduced. The solution is to perform a regular "nightly" build once build scripts are consistent, predictable, and portable. This ensures that each developer's latest revisions will be tested in an integration build so problems are immediately identified. Another solution is to place all enterprise-wide common code on a shared network drive (often called a reference directory). This ensures that all development teams are building against the latest approved versions of enterprise-wide common code. If a change in common code corrupts a developer's work, it will be discovered immediately.

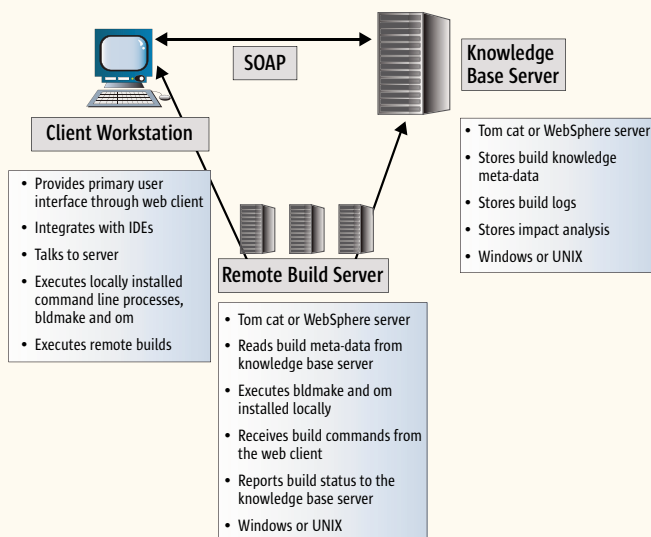
Another common problem in complex environments is build scripting inconsistency resulting from development in multiple languages. Build administrators can either force a single scripting language or maintain different build scripts for different teams (Make, for example, works for C and C++, but is not particularly well suited for Java). The best approach is to maintain different scripts with isolated build functionality using a consistent, cross-platform, lightweight scripting language for all non-build functionality (e.g., retrieving code from a CM tool, moving files around, deploying binaries, etc.). Separating build functionality from all non-build functionality limits variances. There is no reason to be using Make or Ant scripts to copy files around or make a logical decision during batch processing.

The third most common problem in complex environments is the lack of an effective audit trail. Log all build script templates and "non-build" script components and make sure audit trails track source code to the executable. For each action that touches source code (check-out, move, compile, etc.), embed a logging message into the script templates. This is facilitated by adding a basic bill of materials report to the in-house build solution, including:

- Name of the final target being built
- Build machine environment information
- Compiler version information
- Version information derived from the CM tool for every dependency included in the build

Migrating to an Automated, Non-Scripting Build-Management Environment

To solve the problems described in the article, teams within medium- to large-sized development environments are now turning to tools based on a true client/server architecture with a central build knowledge base. Introduced over the past five years, this new class of build tool provides a standardized method for creating and managing Build Control files that replace Make and Ant/XML manual scripting. This approach eliminates the portability issues of rule-based programs derived from Make, while resolving the standardization challenges associated with scripted build processes based on Ant/XML.



One example of this approach is Openmake, a build management tool from Catalyst Systems Corporation that weaves together human and machine intelligence to automate and standardize the enterprise build process. It incorporates a browser-based user

interface and a Tomcat, WebSphere, or OC4J Application Server to provide access to an Openmake Knowledge Base Server (see Figure 1). Enterprise-based features allow for the connection to multiple remote build servers. Simple Object Access Protocol (SOAP) is used as the communication layer between the browser and the application servers.

Developers interface with Openmake through a Web client, a command-line interface, or indirectly through IDE plug-ins. Build metadata is stored and managed via the central Openmake Knowledge Base Server and reused by multiple developers to generate Ant/XML scripts for Java support, or to generate "Make"-like scripts for traditional build requirements. Build Control files can be generated to build a single object (supporting developer daily compile activities) or a complete application (containing hundreds of interdependent modules). When a complete application Build Control file is generated, it eliminates the problem of recursive Make and ensures the accuracy of incremental builds. Builds can be managed from an empty build directory pulling source code from a predefined search path, or by retrieving source code from a version management tool. Openmake also allows control over environment variable settings such as LIB, INCLUDE, and CLASSPATH so that, regardless of the build machine, the build results are the same.

Developers begin by choosing from among an extensive list of predefined build types, with the option of modifying these templates, as needed. The build type templates are comprised of one or more preconfigured build tasks, enabling developers to quickly and easily craft a completely customized build framework without having to write custom scripts or build classes. Because the build types are stored centrally, all development teams work in a consistent, standardized manner throughout the distributed build process.

Openmake does not replace Ant for completing Java builds, but rather extends the use of Jakarta Ant without the need for manually coding XML scripts. In the place of hard-coded Make and Ant/XML scripts, for instance, Openmake's powerful rules engine takes advantage of its knowledge base of build metadata, such as target name and dependency information, to dynamically generate portable, PERL-based build processes at build time that can be referenced by multiple development teams.

Are you using PDF technology in your daily business?

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

World leader in PDF programming technology



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 world-wide 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.



Contact:
pdfsales@pdf-tools.com

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

Identifying Breaking Points

There are a number of critical “breaking points” that cause in-house build systems to become cost- and/or resource-prohibitive. When they occur, development teams generally begin to consider an automated, non-scripting environment (see the “Migrating to an Automated, Non-Scripting Build-Management Environment” sidebar).

One of the first breaking points occurs when the amount of time it takes for an application to build begins to limit unit- and integration-testing effectiveness. Only the items that need to be built should be built, in a true incremental approach. Another breaking point is excessive problem-resolution turnaround time, because the development environment scales beyond the capabilities of the in-house scripted manual build system. Developers find themselves spending most of their time tracking down what source code and common libraries went into a built object rather than resolving coding problems.

A sure sign that developers are reaching the limits of manual scripting efficiency is when they find themselves consistently spending as much as an hour a day working on build problems (either

their own, or debugging build problems of a centralized CM team). Many companies actually assign a dedicated CM team whose sole responsibility is to execute builds. Developers find themselves waiting for the CM team to build their applications before they can move on to the next development effort. It can reach the point where the centralized CM team simply cannot keep up with the demand, especially when builds are cross-language, cross-platform and incredibly complex.

Conclusion

Through careful build-management planning and implementation, developers are freed up to focus on the development they were hired to perform. By optimizing a developer's workflow, companies are more capable of meeting market pressures with increasingly sophisticated and robust applications. The practices outlined above can be used to develop a build system that handles current build requirements, while paving the way for an orderly transition to automated build strategies that will enable significantly more flexible and manageable development environments. ☛

Listing 1: Generic Make Build Script

```
# =====
# Builds the application executable
# =====
application: application.c lib_a.o lib_b.o lib_c.o
    @cc g -qcpluscmt -gidirfirst -I. -I/sys_apps/ref_dir/release/
include -I/usr/include -o ./exe/application $? -bE:/sys_apps/ref_dir/
release/include/application.imp

lib_a.o: lib_a.c
    @cc g -qcpluscmt -gidirfirst -I. -I/sys_apps/ref_dir/release/
include -I/usr/include -o lib_a.o -c $?

lib_b.o: lib_b.c
    @cc g -qcpluscmt -gidirfirst -I. -I/sys_apps/ref_dir/release/
include -I/usr/include -o lib_b.o -c $?

lib_c.o: lib_c.c
    @cc g -qcpluscmt -gidirfirst -I. -I/sys_apps/ref_dir/release/
include -I/usr/include -o lib_c.o -c $?

GENERIC ANT SCRIPT
<!-- ===== -->
<!-- Compiles source code and packages application.jar -->
<!-- ===== -->
<!-- ===== -->
<target name="compile" depends="prepare">
    <javac srcdir="./src"
        includes="**/*.java"
        destdir="./build"
        debug="off"
        deprecation="off"
        optimize="on"
    />
</target>

<target name="application_jar" depends="compile">
    <jar jarfile="./build/application.jar"
        basedir="./build/classes"
        compress="false"
        includes="com/**/*"
    />
</target>
```

Listing 2

```
# ===== #
# Setup global variables. These variables will be edited by developers
#
# in order to make the scripts run on their workstations.
# ===== #

EXEDIR      = ./exe
REFDIR      = /sys_apps/ref_dir
SHAREDDIR   = /shared

COMPILERDIR= $(SHAREDDIR)/compilers
VPATH      = .:$(REFDIR)/release/src:$(REFDIR)/test/src
IPATH      = $(REFDIR)/release/include

CC          = $(COMPILERDIR)/cc
INCLUDES    = -I. -I$(IPATH) -Iusr/include
CFLAGS      = -g -qcpluscmt -gidirfirst $(INCLUDES)
```

```
application: application.c lib_a.o lib_b.o lib_c.o
    $(CC) $(CFLAGS) -o$(EXEDIR)/application $? -bE:/sys_apps/ref_dir/
application.imp

lib_a.o: lib_a.c
    $(CC) $(CFLAGS) -o lib_a.o -c $?

lib_b.o: lib_b.c
    $(CC) $(CFLAGS) -o lib_b.o -c $?

lib_c.o: lib_c.c
    $(CC) $(CFLAGS) -o lib_c.o -c $?
```

Listing 3

```
<!-- ===== -->
<!-- Setup the properties. These values will be edited by developers -->
<!-- in order to make the scripts run on their on workstations. -->
<!-- ===== -->

<property name="src.dir" value="./src"/>
<property name="build.dir" value="./build"/>
<property name="classes.dir" value="{build.dir}/classes"/>
<property name="debug" value="off"/>
<property name="optimize" value="on"/>
<property name="deprecation" value="off"/>
<property name="jar.name" value="application.jar"/>

<!-- ===== -->
<!-- Compiles source code and packages the jar specified in the jar.name -->
<!-- ===== -->

<target name="compile" depends="prepare">
    <javac srcdir="{src.dir}"
        includes="**/*.java"
        destdir="{build.dir}"
        debug="{debug}"
        deprecation="{deprecation}"
        optimize="{optimize}"
    />
</target>

<target name="package_jar" depends="compile">
    <jar jarfile="{build.dir}/{jar.name}"
        basedir="{classes.dir}"
        compress="false"
        includes="com/**/*"
    />
</target>
```

Listing 4

```
<!-- ===== -->
<!-- Setup the properties. These values will be edited by -->
<!-- developers -->
<!-- in order to make the scripts run on their on worksta- -->
<!-- tions. -->
<!-- ===== -->

<property name="src.dir" value="./src"/>
<property name="images.dir" value="./images"/>
<property name="doc.dir" value="./doc"/>
<property name="build.dir" value="./build"/>
<property name="classes.dir" value="{build.dir}/classes"/>
<property name="deploy.dir" value="./deploy"/>
```



```

<property name="deploy.doc.dir" value="${deploy.dir}/doc"/>
<property name="deploy.images.dir" value="${deploy.dir}/
images"/>
<property name="deploy.jars.dir" value="${deploy.dir}/
jars"/>
<property name="debug" value="off"/>
<property name="optimize" value="on"/>
<property name="deprecation" value="off"/>
<property name="appjar.name" value="application.jar"/>
<property name="commjar.name" value="common.jar"/>

<!-- ===== -->
<!-- Compiles source code -->
<!-- ===== -->

<target name="compile" depends="prepare">
  <javac srcdir="${src.dir}"
    includes="**/*.java"
    destdir="${classes.dir}"
    debug="${debug}"
    deprecation="${deprecation}"
    optimize="${optimize}"
  />
</target>

<!-- ===== -->
<!-- Packages application.jar -->
<!-- ===== -->

<target name="common_jar" depends="compile">
  <jar jarfile="${build.dir}/${appjar.name}"
    basedir="${classes.dir}"
    compress="false"
    includes="com/company/application/**/*"
  />
</target>

<!-- ===== -->
<!-- Packages common.jar -->
<!-- ===== -->

<target name="application_jar" depends="common_jar">
  <jar jarfile="${build.dir}/${commjar.name}"
    basedir="${classes.dir}"
    compress="false"
  />
</target>

```

```

includes="com/company/common/**/*"
</target>

<!-- ===== -->
<!-- Prepares the objects for distribution. -->
<!-- ===== -->

<target name="prep-deploy" depends="application_jar">
  <mkdir dir="${deploy.dir}" />
  <mkdir dir="${deploy.doc.dir}" />
  <mkdir dir="${deploy.images.dir}" />
  <mkdir dir="${deploy.jars.dir}" />

  <copy todir="${deploy.doc.dir}" >
    <fileset
      dir="${doc.dir}"
      includes="**/*.html"
      excludes="**/dev/*,**/qa/*"
    />
  </copy>

  <copy todir="${deploy.images.dir}" >
    <fileset
      dir="${images.dir}"
      includes="**/*.jpg,**/*.gif"
      excludes="**/dev/*,**/qa/*"
    />
  </copy>

  <copy todir="${deploy.jars.dir}" >
    <fileset
      dir="${build.dir}"
      includes="*.jar"
    />
  </copy>

  <zip zipfile="deployable.zip"
    basedir="${deploy.dir}" />

  <tar tarfile="deployable.tar"
    basedir="${deploy.dir}"
  </target>

```

total output control



DynamicPDF™ 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

DynamicPDF™ 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!



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



Entitlement to Data

by Michael Poulin

Based on user profile information and an O/R mapping tool

The requirements for different user-facing applications frequently say something like: "User has to see/read/be shown only funds/records/itineraries/policies he or she is entitled to." Permissions in these cases usually depend on multiple factors related to the user profile (job role, locale, etc.), to the protected data (data origin, storage, approval status, etc.), or to both. This represents the fine granular entitlement requirements that are rarely supported by commercial systems.

In this article I will discuss different methods of entitlement to persistent data. The described example allows Java developers to construct controlled access to data using known Java tools. The example may be viewed as an extension to existing entitlement systems or as a lightweight solution for entitlement to data.

Business Task

When referring to a business task, business processes assume that people may read, modify, publish, and massage data when they need to. It sounds very simple; however, it's not a simple technical task at all. Data access requests usually come from the business without any considerations for implemented data models in data storage, data normalization, and distribution. For instance, as we know, database schemas at the enterprise level reflect common needs while data access control must support specific requirements of every business unit.

Sometimes a lot of factors and rules have to be applied to find what data the user is entitled to. Plus, different user activities may require different entitlement rules. The complexity of the rules varies a lot. An example from the financial industry combines a job assignment rule for an accountant with the so-

Example of the "Asia Data Access" Rule

The Law prohibits disclosure of the local client's data outside of the country, except for the following circumstances:

- Customer-written consent obtained
- Internal audit
- Performance of risk management estimation:
 - Users performing risk management functions with a "need-to-know"
- Operational functions outsourced
- Proposed restructure, transfer or sale of credit facility

called "Asia Data Access" rule, which may have several exceptions. The combined rule set has to be applied to every financial transaction to find out which transaction the accountant may work with.

If we use a Rule Engine for this task, we simplify the rule set management dramatically. However, every transaction out of millions per day has to be challenged in the Rule Engine. The efficiency of this approach is doubtful; processing may require a lot of computational resources; it can take a lot of time while it may not be scalable. I didn't even mention resources that have to be spent on controlling the rule consistency and compliance.

If the Rule Engine is not used and we code rules directly, we still have to deal with each individual transaction and with the individual user's profile to calculate entitlement. The number of possible combinations of the financial and user's data becomes enormous and not really manageable.

Design Considerations

An entitlement system may be built based on different models such as user- or resource-centric, and role- or rule-based. In spite of the mode, every entitlement system operates with users and resources/assets. The purpose of an entitlement system is to allow users to access



Michael Poulin is a SW professional with 25 years of experience working as a technical architect for a leading Wall Street firm. He's a Sun Certified Architect for Java Technology and IT Project+ Certified Professional. For the past several years Michael has specialized in distributed computing, application security, and SOA.

mpoulin@usa.com

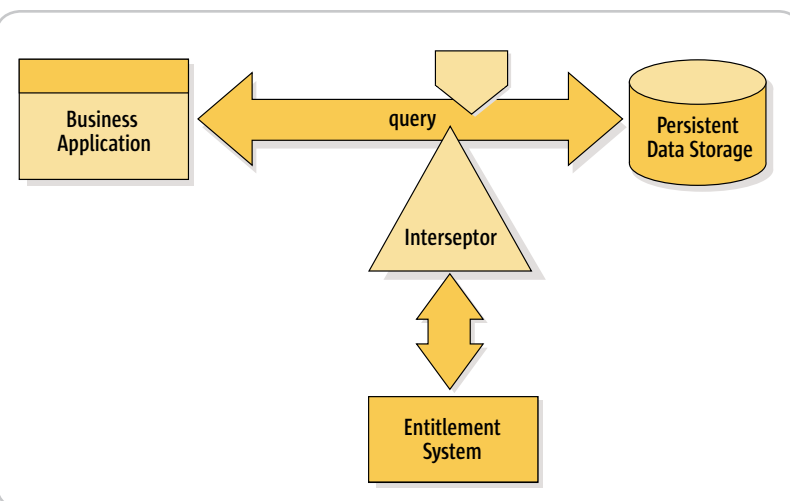


Figure 1 Model of query interception for entitlement control

only the assets they are permitted to access or to protect assets from unauthorized access by the users.

In the case where the asset is a persisted data record, we face a challenge – the amount of such assets may easily exceed billions and grow every day by millions as we mentioned before for the financial transactions. It's obvious that treating each of the transactions as an asset and defining rules to access it is unrealistic. The case gets even worse when we cannot a priori identify every user of the data and grant him or her access rights based on identity. It usually happens when the user's access rights are defined by the rules based on user characteristics such business role and locale.

From a security perspective, proper data access control requires the data to be selected before it is obtained by the application that's available to the user. Several methods of selection may be created but all of them fall into two categories: inside the data storage/database or outside of it. For instance, if the selection is made at the object level, it appears as quite insufficient, potentially poor in performance, and insecure because the full data set, which the user is not supposed to have access to, should be retrieved from the database into the application, e.g., as Value Objects, and, only then, selected. The Value Objects become vulnerable and easier for intruders to access. If we choose to perform the selection inside the database via predefined data filters, we significantly improve performance but anticipate tremendous risk because the decision of “who may access what” happens after the request “gets into” the database. If the request/query is altered on its way, we open the database either to malicious operations or to malicious data access.

To minimize the security risk and provide acceptable performances, an entitlement system has to control data queries, that is, only authorized queries are executed inside the data storage. The job of the entitlement system in this case is to maintain the association between users and the queries they are entitled to use. Moreover, if the user's and asset's characteristics can be used as the query parameters, it allows for the individualized selection of a large amount of data while controlling a relatively small number of selectors/filters.

The ideally secured solution would be to intercept a request for data on its way to the database and dynamically compose or modify the query based on the user's entitlement to data (see Figure 1). This solution is equally effective if data is in single storage or if data is spread over multiple heterogeneous sources. For single storage, we can use a regular object-relation mapping tool like Hibernate. For a distributed data source, we can use something like IBM WebSphere DataStage with a custom adapter and a defined data filter or BEA LiquidData with XQuery. However, for simplicity, we will discuss the single data storage case in the following section.

Example of Data Entitlement Implementation

Let's consider a Web application that is supposed to display a portfolio of personal investments. The application gets data from multiple database schemas of a single Oracle database via a Data Access Layer (DAL). A DAL is based on an object-relational product such as Hibernate 3.0. The content of the individual portfolio is protected by an entitlement system (ES). That is, every user of the

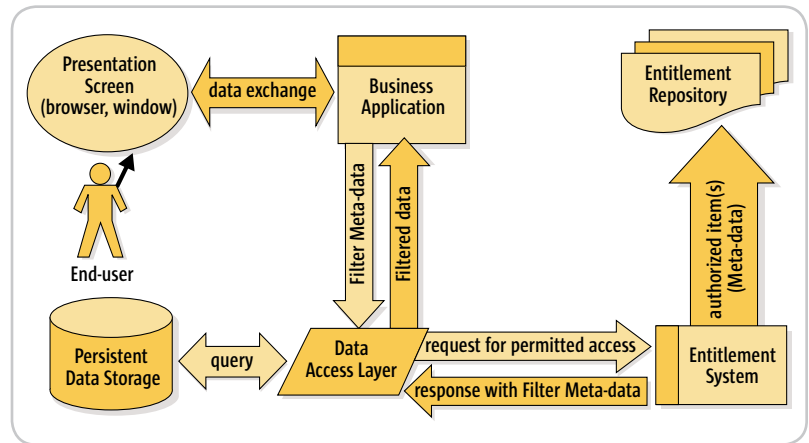


Figure 2 Relationship and interactions between application, entitlement system and DAL

application is registered with the ES, has his or her user profile, and may be entitled to a particular set of funds. Figure 2 shows the relationship and interactions between the application, ES and DAL.

In particular, an end user invokes a Web application (or a business application) via a remote client like a Web browser and, upon authentication, requests the fund information from the portfolio. The application processes the request and prepares the response. The fund data needed for the response is obtained from the database that's accessible via the DAL. The latter invokes the ES to authorize the request.

**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/vjdj> for information. EOE

Google

“From a security perspective, proper data access control requires the data to be selected before it is obtained by the application that’s available to the user”

In the ES, user entitlement to the funds is interpreted as an association between the user and a set of data filters. The data filter is a meta-definition of a real SQL filter, i.e., part of the “WHERE” clause. The data filter comprises the filter name and the ordered list of the filter parameters. To fill in filter parameters, the ES obtains the user profile and matches the user profile’s attributes with the data filter parameters as shown in Figure 3. An example of the Filter class is shown in Listing 1.

In Listing 1, while the `filterName` variable is self-explanatory, the `java.util.HashMap` is constructed with filter parameter names as the keys and the parameter values organized in `java.util.ArrayList`. This allows the use of multi-valued filter parameters.

The ES returns a list of data filters the user is entitled to. At this moment, the DAL does not know which data filter to apply. The DAL iterates through the received data filter collection and, using the Hibernate API, enables matching filters defined in the DAL. For every enabled filter, the DAL sets filter parameters using values obtained by the ES from the user profile. The fragment of related code is shown in Listing 2. If no exceptions are thrown, the DAL runs the HQL query using the `session.createQuery(...).set[ObjectKeyType].list()` or `session.get(...)` API that, in turn, generates a SQL query, executes it against the database, and returns filtered results.

As we see, the DAL is not just a Hibernate implementation. The

ability to control data access and even to load-balance and engage parallel processing for complex requests distinguishes the DAL from an O/R mapping tool.

It is a fair concern about the performance of the DAL. However, if you really need to secure data access, the performance degradation caused by the entitlement controls should be compensated by the application architecture. It might include additional data caching and buffering, optimization of the database schema, and the usage of denormalization in views. This is why it’s very expensive to add security later on in the application life cycle instead of considering it at the architecture and design phases.

Thus, data is selected inside the database with the top performances and the selection is performed only by the filters the user is entitled to. Of course, you can construct a different implementation of the data filter. For instance, a JDBC call may be intercepted and the SQL query may be enriched with entitlement information on the fly, or the solution may be based on stored procedures instead of dynamic queries generated by Hibernate. Nevertheless, it should not change the security model, e.g., the stored procedure name and input parameters to be provided by the security system based on user entitlements.

Conclusion

I have demonstrated how to control access to millions of data records in persistent data storage via entitlement to a few data filters. The data access layer can combine parameterized data filters with user profile information and construct individualized and secured queries executed with maximum perfor-

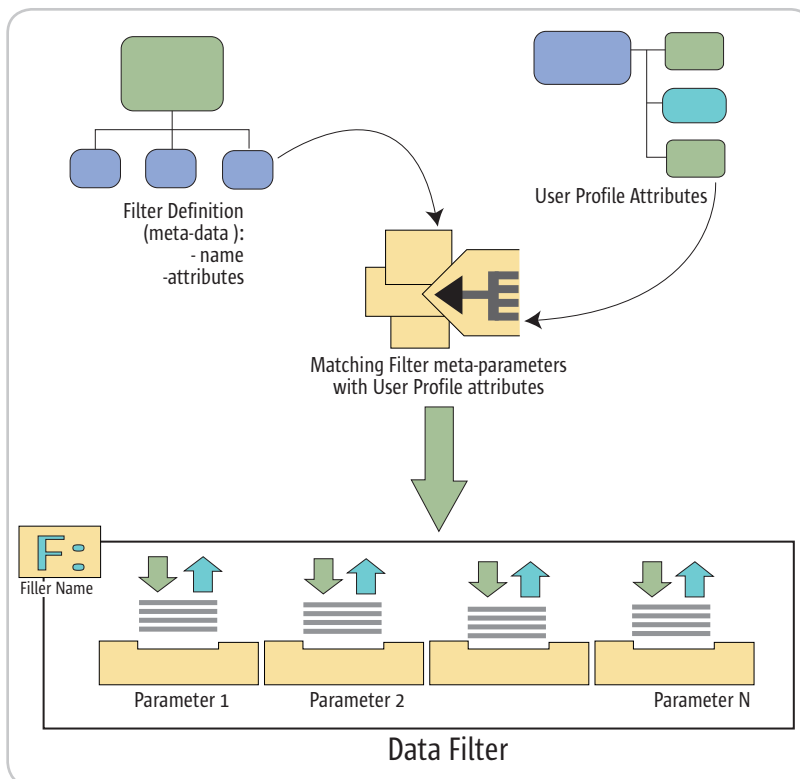


Figure 3 Filling Data Filter parameters with User Profile data

mances inside the data storage. This represents an example of how entitlement to a data solution may be implemented based on user profile information and an O/R mapping tool. ☞

References

- Hibernate: Relational Persistence For Idiomatic Java: <http://www.hibernate.org/>
- IBM WebSphere DataStage: <http://www-306.ibm.com/software/data/integration/datastage/>
- Poulin, M. "How to Deal with Security When Building Application Architecture." JDJ, Vol.10, issue 9
- BEA AquaLogic Enterprise Security: <http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/aqualogic/security/>
- BEA Liquid Data for WebLogic 8.1 Documentation: <http://e-docs.bea.com/liquiddata/docs81/index.html>

Listing 1

```
public class DataFilter implements java.io.Serializable {
    private String filterName;
    private java.util.HashMap filterParameters = new java.util.
HashMap();

    public String getFilterName() {
        return filterName;
    }

    public void setFilterName( String name ) {
        filterName = name;    }

    public java.util.HashMap getFiltrerParameters() {
        return filterParameters;    }

    public void setFiltrerParameters ( java.util.HashMap parameters
) {
        filterParameters = parameters;    }
}
```

Listing 2

```
org.hibernate.Session session = ...;
//...

dataFilterList = ...;
//...

if ( dataFilterList != null && dataFilterList.size() > 0 ) {
    java.util.Iterator filterIter = dataFilterList.iterator();
    while ( filterIter.hasNext() ) {
        DataFilter filter = (DataFilter)filterIter.next();
        String name = filter.getFilterName();
        java.util.HashMap parameters = filter.getFilterParam-
eters();

        java.util.Set keys = parameters.keySet();
        org.hibernate.Filter hibFilter = null;
        try {
            hibFilter = session.enableFilter(name);
```

```
    }
    catch(org.hibernate.IllegalArgumentException iae) {
        continue; // the Filter is not applicable
    }
    java.util.Iterator parameterIter = keys.iterator();
    while( parameterIter.hasNext() ) {
        String paramName = (String)parameterIter.
next();

        try {
            org.hibernate.Filter setFilter = hibFilter.setParameterList(
paramName, ((java.util.ArrayList)parameters.get(paramName)).
toArray() );

        }
        catch( org.hibernate.IllegalArgumentException piae) {


            // log exception ...
            break; // error: match between

User Profile - Filter

        }
    }
}
//...
```

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**

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



Developing in Java 5

by Roberto Scaramuzzi

Use the new set of tools wisely

“Ease of Development” is one of the main focuses in J2SE 5. Accordingly, J2SE 5 introduces several new features designed to simplify the developer's life. If you use these new constructs, your code will become more compact and expressive, hence easier to understand and debug. This article explains how you can use the new features to prevent some silly mistakes, as well as some that are not so silly.

Generics

The Collections framework was introduced in Java 1.2. It provides a set of interfaces and implementations that encapsulate many common data structures, the most common being the List and Map interfaces.

The main drawback of using this well-thought-out framework is that a significant amount of casting is required when extracting elements from a pre-J2SE 5 collection (because the collection necessarily has no knowledge of what kind of elements are stored in it). Listing 1 provides a simple example. In the listing, the line with the cast is clunky. Yet it seems that nothing could go wrong, since the only way to add elements to fileList is through the addFile method, which takes only a string.

However, problems will inevitably surface sooner or later. For instance, assume that two years later, you or another developer decide that it would be a good idea to store actual File objects (instead of their names) in the FileList class.

The addFile method will be changed to read:

```
public void addFile (File file) {
    fileList.add(file);
}
```

The code will compile just fine. Yet, when the customer runs the application, it will crash with a ClassCastException as soon as the listExtensions () method is called. One way to guard against this problem is to wrap each casting operation into some defensive code:

```
while (it.hasNext()) {
    Object name = it.next();
    if (name instanceof String) {
        System.out.println
            (getExtension((String)name));
    }
}
```

This will solve the immediate problem, but it has several drawbacks:

- The code is now even clunkier (and, consequently, more difficult to maintain).
- There are (potentially) many unnecessary instances of operations.
- The application will fail silently instead of crashing when a disruptive change in code (such as the one above) occurs. Whether this is a better or worse situation for the customer and the support staff is debatable.

I'm pretty sure that most developers wouldn't make such an obvious mistake, but the problem can manifest itself in much more subtle ways, and it grows into a major issue when libraries are shared across team and company boundaries.

J2SE 5 provides a solution for all of this: collections can now use generics. For instance:

```
private List <String> fileList = new ArrayList <String> ();
```

Now, only strings can be added to fileList, and method calls, such as fileList.get(0) and fileList.iterator().next(), will return strings with no casting necessary:

```
String name = it.next();
```

If you use generics and try to modify the addFile method, the compiler will flag your attempt to add a File to a List<String>. If you change the declaration of fileList to a List<File>, the compiler will now notice that it.next() returns File and not a string; this will force you to fix the listExtensions method.

Note that this is just one use of generics. One of my other favorites is that since the Comparable interface is now generic, you can write the following code and avoid the boilerplate instanceof/casting code that is common for compareTo method implementations:

```
public class Comp implements Comparable <Comp> {
    public int compareTo(Comp o) {
        ....
    }
}
```



Roberto Scaramuzzi is a software engineer at Parasoftware in San Diego. He grew up professionally as a Perl developer, but is now also adept at Java and PHP. Born in Italy, he later moved to the United States to obtain his doctorate in mathematics from Yale University.

rsjdj@parasoftware.com



“‘Ease of Development’ is one of the main focuses in J2SE 5”

Enhanced For Loop

Here is a snippet of buggy code (slightly modified from the J2SE 5 release notes):

```
List<Suit> suits = ...;
List<Rank> ranks = ...;
List<Card> sortedDeck = new ArrayList<Card> ();

for (Iterator<Suit> i = suits.iterator(); i.hasNext(); )
    for (Iterator<Rank> j = ranks.iterator(); j.hasNext(); )
        sortedDeck.add(new Card(i.next(), j.next()));
```

If you try to run it, it will throw a `NoSuchElementException`. Can you spot the bug? The problem is that `i.next()` is being called every time a new card is created, rather than once per suit. As a result, we run out of suits much faster than expected. This is how to fix the loop:

```
for (Iterator<Suit> i = suits.iterator(); i.hasNext(); ) {
    Suit suit = i.next();
    for (Iterator<Rank> j = ranks.iterator(); j.hasNext(); )
        Rank rank = j.next();
        sortedDeck.add(new Card(suit, rank));
}
```

J2SE 5's new enhanced for loop construct provides a neat solution for this. The nested loop in the above code can be rewritten as:

```
for (Suit suit : suits)
    for (Rank rank : ranks)
        sortedDeck.add(new Card(suit, rank));
```

What appears to be happening is that the variables `suit` and `rank` point (in turn) to the elements of the `Lists` `suits` and `ranks`. Under the hood, the same code as in the fixed loop is being run.

Enhanced for loops can also be used while iterating through arrays. This time, instead of not having to explicitly iterate in your code, you can skip any references to the array indices:

```
String [] vowels = new String [] { "a", "e", "i", "o", "u" };

for (vowel : vowels) {
    System.out.println(vowel.toUpperCase());
}
```

As well as being more compact and easy to understand, the enhanced for construct for arrays prevents common bugs, such as referring to the wrong index in nested loops:

```
boolean isVowel (char c) {
}

int vowel_count = 0;
for (int i = 0; i < words.length; i++) {
    String word = words[i];
    for (int j = 0; j < word.length(); j++) {
        if (isVowel(word.charAt(i)) {
            vowel_count++;
        }
    }
}
```

A poster for the ODTUG 2006 Kaleidoscope conference. The background is a dark blue field with a complex, multi-colored geometric pattern of triangles and polygons in shades of blue, green, and yellow. The title "CLARIFYING THE DEVELOPER'S CHANGING UNIVERSE" is written in large, white, sans-serif capital letters across the center. In the top right corner, white text reads "Be a Presenter—Abstracts Due December 12". Below this, in smaller white text, is "June 17-21, 2006 Wardman Park Marriott Hotel Washington, D.C.". The website "www.odtug.com" is displayed in yellow. The ODTUG logo, featuring a stylized sunburst or kaleidoscope graphic, is positioned to the right of the text "ODTUG 2006". Below the logo, the word "KALEIDOSCOPE" is written in large, white, sans-serif capital letters. At the bottom, a light blue horizontal band contains the word "TOPICS:" in white, followed by three columns of topics in blue text: "Application Development", "Traditional Tools", "Business Intelligence"; "Methodology", "Development DBA", "Professional Development"; and "Non-Oracle Development Tools", "PeopleSoft Development", "Siebel Development".

CLARIFYING
THE DEVELOPER'S
CHANGING
UNIVERSE

Be a Presenter—
Abstracts Due
December 12

June 17-21, 2006
Wardman Park Marriott Hotel
Washington, D.C.

www.odtug.com

ODTUG 2006
KALEIDOSCOPE

TOPICS:

Application Development	Methodology	Non-Oracle Development Tools
Traditional Tools	Development DBA	PeopleSoft Development
Business Intelligence	Professional Development	Siebel Development

Enums

C has an enum construct, which in theory allows the developer to define an enumerated type, but in practice does little more than define a bunch of integer constants. In the pre-J2SE 5 world, the most common way to accomplish the same task was to mimic the way C works and write code like this:

```
public class Card {
    // suits
    public static final int CLUBS = 0;
    public static final int DIAMONDS = 1;
    public static final int HEARTS = 2;
    public static final int SPADES = 3;
    // ranks
    public static final int ACE = 1;
    public static final int DEUCE = 2;
    ...
    public static final int KING = 13; }
```

You could also use an interface for the same purpose. This is common, but it's debatable whether it's a good idea to do so.

This implementation has many problems. Since the constants are just ints, it's easy to pass out-of-range values to methods. There is nothing to prevent you from passing these constants to methods they weren't designed for.

For example:

```
public class Card {
    public Card (int suit, int rank) {
        ...
    }
}
```

You can write something like `new Card(KING,HEARTS)` and you won't find out that something is wrong until runtime.

It's possible to use the "typesafe enum" pattern (for example, see the book *Effective Java* by Joshua Bloch) in JDK 1.4 to overcome these problems, but it requires a lot of boilerplate code, and the resulting classes have a fatal flaw, from the developer's point of view: you cannot switch on this kind of enum, so any switch-like code becomes very clunky.

J2SE introduces the enum keyword to specify first-class enumerated types. Here is simple example code:

```
public enum RegularPolygon {
    TRIANGLE,
    SQUARE,
    PENTAGON;
}
```

A significant advantage of the enum construct (compared to handcrafted enumerations) is that enums are full-fledged Java classes, so they can have constructors, fields, and methods. For example:

```
public enum RegularPolygon {
```

```
    TRIANGLE(3),
    SQUARE(4),
    PENTAGON(5);
    final int numSides;
```

```
    RegularPolygon (int sides) {
        numSides = sides;
    }
    public double perimeter (double sideLength) {
        return sideLength * numSides;
    }
}
```

Client code can switch on the enum:

```
public computeArea(RegularPolygon p, double side) {
    switch (p) {
        case Polygon.TRIANGLE:
            return side * side * Math.sqrt(3) / 4;
        case Polygon.SQUARE:
            return side * side;
        case Polygon.PENTAGON:
            return side * side * 1.72;
        default:
            System.err.println("Unknown polygon");
            return 0;
    }
}
```

The default case in the switch is used in case someone adds a new polygon to the enum.

Class Libraries

J2SE 5 introduces several utility classes that can make your life easier, but you obviously won't gain any benefits from them unless you actually use them. It is almost always a mistake to write your own implementation of a data structure or a method that is present in the Java class library. The base Java implementation is very likely to be better thought-out and less buggy than an implementation that you would come up with. Here are a few new additions to the Java class libraries that may prove useful.

Queue and Implementations

`java.util.Queue` is a new interface that has been added to the Collections framework. `LinkedList` now implements the Queue interface and models a simple FIFO queue. Another implementation you may find useful is `PriorityQueue`.

ProcessBuilder

The `ProcessBuilder` class provides a significantly easier way to launch and control applications than `Runtime.exec()` does. The following code will launch Eclipse on a Linux machine from a specified directory and display its window on the main display of the machine called `italy`.

```
String home = System.getProperty("user.home");
```

```

ProcessBuilder builder = new ProcessBuilder (
    ".\\eclipse", "-data", home + "\\workspace");
builder.directory(new File(home + "\\eclipse"));
builder.environment().put("DISPLAY", "italy:0.0");
builder.start();

```

Formatter

The Formatter class and its cousin methods `String.format()` and `PrintStream.printf()` provide a straightforward and powerful way to finely format strings and output. If you are familiar with the C `printf` and `sprintf` functions, it should be easy to use the new class and methods. For example, the following code will print out an amount of money using the correct padding if the cents are a one-digit number:

```

public static void dollarFormat (int dollars, int cents) {
    System.out.printf("$%d.%02d", dollars, cents);
}

```

Scanner

The Scanner class provides an easy way to extract strings and primitives from a formatted stream. By default, it uses white space for field delimiters, but it's easy to configure it to use any regular expression as a delimiter. J2SE 1.4 classes and methods such as `StringTokenizer`, `String.split()`, and `Integer.parseInt()` could cover some of the functionality that the Scanner class provides, but the new class is more flexible and powerful. The following code will echo lines from `System.in` to `System.out`, as long as they can be parsed as integers:

```

Scanner scanner = new Scanner(System.in);
scanner.useDelimiter(System.getProperty("line.separator"));
while (scanner.hasNextInt()) {
    System.out.println(scanner.nextInt());
}

```

Conclusion

Often, being a programmer is much like being a carpenter. To do a job well, you need to know which tools are available and which one is best suited for the job at hand. J2SE 5 provides a set of new tools that you can add to your Java development tool belt. Use them wisely, and you will spend more time thinking about how your code hangs together and less writing boilerplate and tracking down bugs. ☺

References

- Bloch, J. (2001). *Effective Java*. Addison-Wesley.
- J2SE New Features: <http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html>
- Austin, C. (2004). "J2SE 5.0 in a Nutshell": <http://java.sun.com/developer/technicalArticles/releases/j2se15/>
- JSR 201 (Enumerations, Enhanced for loops, etc.): <http://www.jcp.org/aboutJava/communityprocess/review/jsr201/>
- JSR 14 (Generics): <http://www.jcp.org/aboutJava/communityprocess/review/jsr14/>

Listing 1

```

import java.util.*;


public class FileList {
    private List fileList = new ArrayList();

    public void addFile (String name) {
        fileList.add(name);
    }

    public void listExtensions () {
        Iterator it = fileList.iterator();
        while (it.hasNext()) {
            String name = (String) it.next();
            System.out.println(getExtension(name));
        }
    }

    // more methods including
    // static private String getExtension(String fileName);
}

```



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 all price
- * Connect to notification systems via XML bridge
- * And more!

www.seejava.com

SeeJava is a product of Weblogic Services, LLC. Developing. Consulting. Products. www.seejava.com



Joe Winchester
Desktop Java Editor



Do Strongly Typed APIs Help Ensure Java's Survival or Extinction?

A programming API represents a documented contract between a function that provides some kind of computing service and those who wish to use it. In Java, once an API is used there is a physical contract between the two that the compiler and JVM enforce. If at some point in the future the author of the API wishes to make changes, they are limited in scope; if the author renames methods or removes arguments, programs that are bound to the previous signature will no longer run. The change can be published with the new version of the class library or framework so that users can upgrade their code; however, in many cases this isn't a viable option. It may be that the system that used the old API is no longer being worked on by a development team and is considered stable, or that the release cycle of the now broken system is dependent on other circumstances that prevent it from being upgraded.

No API can be got right the first time, irrespective of how many specifications, architecture meetings, and examples take place prior to its release. The whole idea of launching a set of class libraries and Java-docs onto the world as being the eternal way of doing something goes against the whole philosophy of agile programming. One of the principles of extreme programming is that a good program is built flexibly and in a fluid state; at each stage the feedback loop of usage and reconstruction means that the right solution is arrived at by accident rather than by design. This is akin to evolution, whereby variations in a species arise by random mutation and are mixed in offspring to create new individuals. The fittest varieties survive whatever the environment provides and the cycle continues, allowing the species to adapt and evolve.

The Java language has maintained its API over many releases so a program that is compiled on 1.2 will run unchanged on a Java 5 runtime environment. The benefit of this is that older Java programs remain functional on modern JREs; however, it comes at a price. The language now contains a plethora of deprecated classes

and methods, new features are sometimes sacrificed to ensure backward compatibility, and the evolution of Java is hindered and held back. The class *java.util.Date*, for example, has more public deprecated methods than non-deprecated ones. Not a lot has changed in the domain model of Dates since the Gregorian calendar was introduced in 1582, and this ratio of legacy to current API is a sign that code change is a natural flux that occurs during its usage.

In nature the extinction of a species is not a gradual thing but comes in waves. Punctuated Equilibrium describes an effect whereby the environment remains fairly static for a while and then a huge



amount of change occurs in a small amount of time. These boundaries are mass extinctions that not only cause many species to die out, but are also the event that allows others to flourish and fill the void left by the departing ones. To a certain extent Java benefited from this back in 1995, when it was born out of a combination of the best ideas of C++ and Smalltalk; the catastrophic meteorite striking at the time being the Internet and demand for heterogeneous networked computing.

What worries me 10 years later is that I don't believe Java is evolving fast enough, and that strong typing to antiquated APIs are holding it back. XML won over a slew of competing technologies for program-to-program communication transport because it is text based, allowing it to be read by heterogeneous end points (as opposed to alternatives like CORBA IDL or other binary RPCs). XML is also flexible in its typing, because tags can be declared as optional and new tags can be introduced in a message without breaking callers

using the older format. While Java has done well as an enterprise language in the application server space, this is largely due to the fact that the big names of IT backed it rather than any inherent feature of the language. Languages such as Ruby and Perl have grown tremendously in the past few years and now command a sizeable portion of application development.

If Java is to remain at the forefront of technology for the next 10 years, it must find a better way to evolve. It needs to find a way of decoupling API calls between internal code and external blocks, perhaps even introducing soft typing calls across program boundaries or having flexible message transport across modules. While XML parsing functionality is now included in a Java runtime by default, it feels very shoehorned into place when compared with how .NET has done the same integration. Older languages like COBOL or RPG, as well as Visual Basic, include the concept of structures into a program definition that can be externalized, and having Java include a slew of JARs and runtime bolt ons to do simple XML parsing seems a missed opportunity to embrace the technology.

Becoming strong enough to do battle with Microsoft, the T-Rex of computing, may have fared well for Java in the past, but at the expense of what it has now become and how quickly it can keep up with change and the legacy it has created. I fear that unless a fundamental change to the language is made that allows more flexibility in binding between functional units, between and across JVM boundaries, Java will be a victim of the same fate as the behemoths it fights in the IT tar pit. Survival against change – whether in nature, technology, or business – is determined by one's ability to adapt. Java needs to move forward by being more agile. As the language grows with more and more APIs and divisions, it runs the risk of losing sight of the finish line, while smaller and nimbler technologies become more ubiquitous. When the next IT meteor strikes, Java must be the fastest to adapt if it is to survive and grow over the next 10 years. ☛

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 10th International WebServices Edge 06 conference+expo



June 2006 | New York, NY

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 OPEN SOURCE CONFERENCE+EXPO



June 2006 | New York, NY

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 JOURNAL

eclipse
developer's journal

WebSphere
JOURNAL

Information
STORAGE+SECURITY
JOURNAL

wldj
THE JOURNAL
OF WEB
SERVICES
DESIGN

JDJ

Linux
WORLD

MX
developer's journal

asp.net
MAGAZINE

SD Times

CoDe

Software Test
& Performance

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



Attention Exhibitors:

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

For Exhibit and Sponsorship Information - Call 201 802-3066

Produced by **sys-con**
EVENTS

© 2005 WEB SERVICES EDGE. ALL RIGHTS RESERVED

Extending Rich GUI Clients with Jython

by Hayden Marchant

Implementing a solution

Allowing extensibility of a rich Java GUI is a daunting task. Each user may require slightly different functionality – this one wants to be able to import data from an Excel spreadsheet, and another wants to generate custom XML reports of particular artifacts in the application. You want to make every user happy, but you very rapidly see yourself with multiple code branches trying to satisfy those custom requirements – every Java developer's configuration management nightmare.

Not only is it undesirable to dirty the code base with customer-specific functionality, but you don't want to derail the core development team with continual disruptions for customer requests for simple enhancements to the GUI; it would be much more effective for the customer to be able to add all the functionality that is needed.

At Unicorn Solutions (www.unicorn.com), we developed a Scripting Extension Framework, developed in Jython and XML, that allows non-Java developers to easily enhance our out-of-the-box Swing-based application with custom, rich GUI features. This has greatly increased the extensibility of the Unicorn System, our platform for enterprise metadata management and semantic enterprise information management.

This article is aimed at both developers who are struggling with the task of multiple requests to customize their rich Java client for different customers, and developers who have an interest in Jython and examples of how it integrates with Java applications. It covers the requirements for the framework, and details the design and implementation of the Scripting Extension Framework with sample code. It covers both the usage of XML descriptors for extensions as well as how to call the

Jython Interpreter from Java. It will give the target audience in-depth knowledge on how to implement a similar solution in any similar application.

Introduction – Flexing the Inflexible Application

Release day has finally arrived – the full-featured Swing application that you've been developing for the past two years is finally going out to the customers on time. The problems start when

each customer wants slightly different enhancements, most of which you'd rather not incorporate into the main product. The type of features that a customer might want could range from a simple "Find and Replace" feature, to any of the following customer-specific features (see Figure 1):

- Batch loading of data from external legacy sources (flat file, Excel)
- Verification procedures on custom data

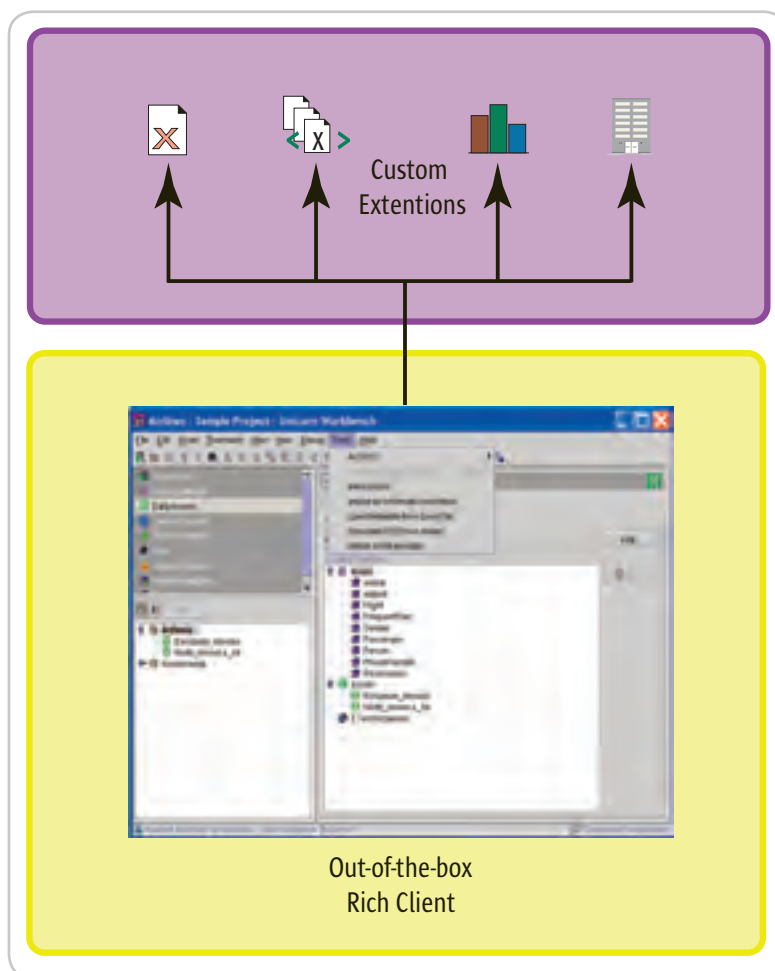


Figure 1 Application with custom extensions



Hayden Marchant is a software architect at Unicorn Solutions Inc., a provider of enterprise metadata management. He has over 10 years experience in designing and implementing solutions in both the enterprise and telecommunications industry, and is a Sun Certified Programmer.

hayden.marchant@unicorn.com

- Wizards for automating custom tasks
- Generation of reports in different file formats (CSV, XML, or Excel)
- Automated processes to update data

If this isn't bad enough as it is, it has become apparent that the customer is not going to stop with those requests but will continue to bombard you with enhancement and utility-feature requests for the long-term future.

The traditional ways of addressing such requirements range from releasing a different version of the product for each customer to providing code-hooks that the customer can code against to add the required functionality.

Releasing a different version of the product to each customer is a Configuration Manager's biggest nightmare – the task of maintaining multiple branches of the same release, together with tracking which features are in which version, is an error-prone process that is very expensive on resources. Not only that, but the amount of R&D development time to work on non-generic customer enhancements can be a big drain on resources, both in development and testing.

Providing code-hooks are generally very limiting in the breadth of functionality that can be exposed, and, generally, most custom enhancements can't be fully implemented in this framework.

Making the Application Flexible

Before getting down to making your rich GUI application flexible, we need to understand what it means to be flexible and what we'd like to see from the framework that provides this flexibility.

1. Low levels of programming skills to develop extensions. Already at Unicorn, there are several customers writing their own extensions with minimal support required from R&D.
2. The extensions should be developed in an easy-to-learn scripting language that integrates well with the main product.
3. Customer should be able to create simple GUI dialogs for their enhancements that control user-input, display reports visually, etc.
4. Seamless integration with core product features. It should not be obvious to users of the customized application which features are part of the core product and which have been developed as custom enhancements.
5. Ease of deployment – it should be

easy for developers of the custom features to deploy these features to the core product. Complicated build processes should be totally avoided.

6. Customer extensions should be guaranteed to be compatible with future versions of the core product.

The Scripting Framework

In order to answer the above list of requirements to create a flexible rich GUI, at Unicorn we developed a full-fledged scripting framework, allowing in-house developers at customer sites to create all the extensions that were required. The Scripting Framework consists of a controller that connects all deployed extensions to the application at runtime. Each extension consists of a descriptor, written in XML, that contains relevant information about the extension; for example, display name, input parameters, and a pointer to the script that contains the execution logic of the script.

The Scripting Framework is needed to adopt a programming language. It was decided to use a language that could tightly interact with Java programs, and there were obvious advantages to an interpreted language for ease of development of scripts. This naturally led to the choice of Jython as the language for the Scripting Framework.

Jython

Jython is an implementation of the high-level, dynamic, object-oriented language Python written in 100% Pure Java, and seamlessly integrated with the Java platform. It thus allows you to run Python on any Java platform. In addition to allowing the use of regular Python syntax, it is also possible to create Java objects and call methods on them within a regular Python expression.

Jython is freely available for both commercial and non-commercial use and is distributed with source code. Jython is complementary to Java and is especially suited for the following tasks:

- **Embedded scripting:** Java programmers can add the Jython libraries to their system to allow end users to write simple or complicated scripts that add functionality to the application.
- **Rapid application development:** Jython programs are typically 2–10X shorter than the equivalent Java program. This translates directly to increased programmer productivity. The seamless interaction between

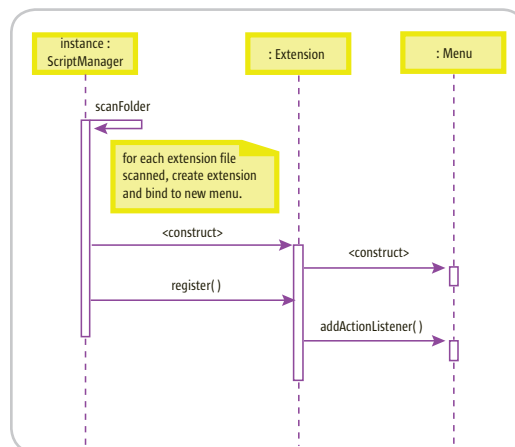


Figure 2 Script Scanning and binding to an application sequence diagram

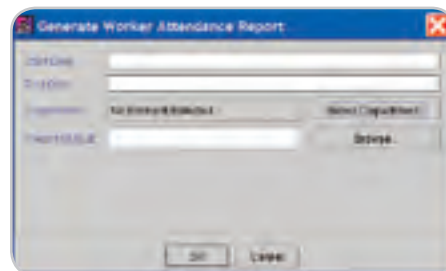


Figure 3 Screenshot of a sample extension dialog

Parameter	GUI Widget
Boolean	Checkbox
String	Text field
Password	Password text field
Choice	Radio button
Number	Number text field
Date	Data selector
Employee	Application-specific Employee selector
Company	Application-specific Company selector

Table 1 Widget definition

Python and Java allows developers to freely mix the two languages both during development and in shipping products.

- **Dynamic compilation to Java byte codes:** Leads to the highest possible performance without sacrificing interactivity.
- **Ability to extend existing Java classes in Jython:** Allows the effective use of abstract classes.

The Python language combines remarkable power with very clear syntax. It also supports a full object-oriented programming model, making it a natural fit for Java's OO design.

Let's look at the steps needed to develop an extension, and how extensions are loaded and executed.

Extension Descriptor

The extension descriptor contains instructions on how the executable

Script Action	Behind the scenes
Initialize Jython Environment	Create instance of PythonInterpreter
All input parameters are placed onto the Jython memory heap, allowing extension developers direct access from the Jython script to the parameters	<ul style="list-style-type: none"> • Converting the Java input objects into PyObjects • For each parameter call PythonInterpreter.set(<param_key>, PyObject) • For GUI purposes, the JFrame of the Swing app is also set so that script can provide custom GUI.
Execute Jython Script	Call PythonInterpreter.execfile(<script_file>)

Table 2 Jython execution

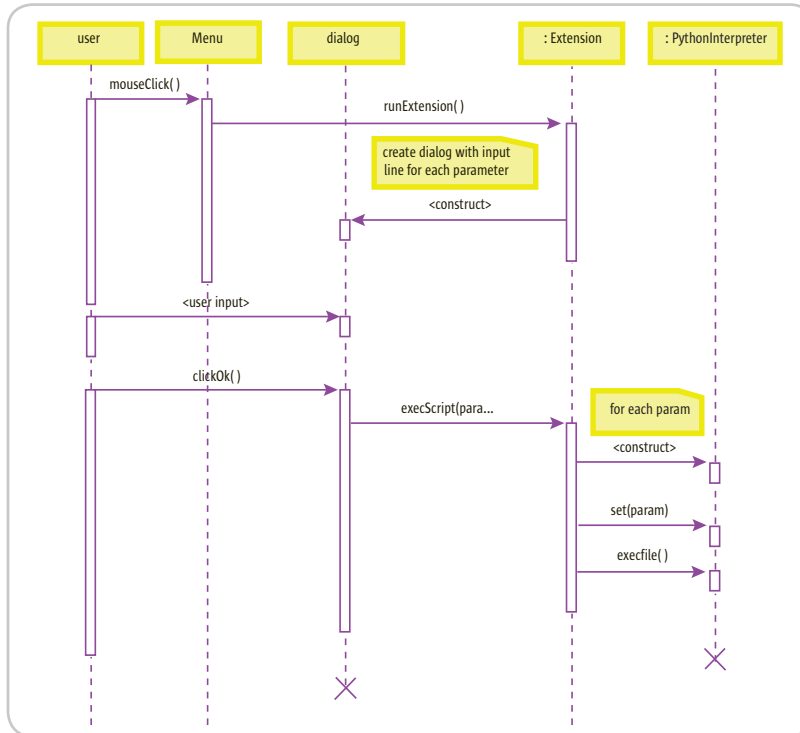


Figure 4 Jython script execution sequence diagram

script should be bound to the application at runtime. The descriptor contains the following information:

- Display name of the extension.
- Informal description of the extension.
- Optional hotkey for the convenient quick launch of the extension.
- Menu/Toolbar location information – the extension developer can decide where the extension will appear in menus/toolbars.
- Instructions for displaying a rich GUI input dialog – this will typically hold a list of parameters, each describing a single piece of input that the user of the extension can enter at runtime, which will subsequently be passed to the script. Each parameter holds the following information:
 - Internal identifier for input parameter
 - Display name of parameter
 - Type of parameter. This can

be simple types such as string, password, number, boolean, string list, or this could also be application-specific objects like Employee, Address, or Company.

- A pointer to the location of the Jython script to be executed.

It's clear that each extension descriptor contains all of the instructions required for extending the application with the extension, including providing the user with a high-usability user interface for accessing the extension.

We chose XML as the file format for the extension descriptor for several reasons. XML is a popular format for expressing information in a structured manner – parsers and validation tools are readily available. Also, by choosing XML, we are able to validate all descriptors against the XML Schema, which allows for catching badly structured de-

scriptors at an early stage. By publishing the XML Schemas to extension developers, we are also providing the developers with a very handy tool to help create syntactically correct descriptors.

How It All Works

On start up of the application, the Extension Manager scans a predefined directory for extension descriptors. Upon scanning, the discovered extensions are registered and bound to the application. Application users will now see menu items and toolbars that expose custom extensions side-by-side with core features of the application. The menu items and toolbars will get their name and location from the extension descriptor (see Figure 2 for an illustration of how the extensions are scanned and bound to the application).

When clicking on a menu item or toolbar for an extension, an input dialog is generated based on the parameters in the descriptor (see Figure 3). A different type of GUI input widget will be used for each type of parameter. For example, a parameter of type boolean will be displayed as a checkbox (see Table 1).

After completing the input of the parameters and the user clicks OK, the following actions are taken (see Table 2):

1. The Jython environment is initialized by using the PythonInterpreter class.
2. All input parameters are placed onto the Jython memory heap, allowing extension developers direct access from the Jython script to the parameters. This is done by converting the Java input objects into PyObjects and passing them to the PythonInterpreter.set() method.
3. The Jython script is now executed using PythonInterpreter.execfile().

See Figure 4 for an illustration of how the extension is executed on the selection of a menu item.

Write Jython Script

Before writing the script, the developer should know which parameters to expect in the script and what the functionality of the script should be. In general, the script will need to access the date that is in the core application.

The suggested way to do this is through a Java API exposed by the core product. This API should allow read and/or write access to information in the application. For example, if it is expected that a script will access Employee records, the API should provide methods to query and then access the information of a particular Employee record. The API should be well documented and with logical naming of classes and methods that reflect the parallel constructs in the application. It's very advisable that the Javadocs of the API are freely available. Developing scripts against a well-documented API is so much smoother than struggling with a badly written, undocumented API.

If the script needs to get further user input or display any messages/warnings, it can use the *frame* (of type JFrame) variable that is in the interpreter's scope as a starting point for creating new dialogs or windows.

Working Through an Example

Imagine a Human Resources management tool – HR Tracker. The tool manages a wide variety of HR details, in particular worker details and shifts that they work on. Doug's mega store would love to buy HR Tracker, but really needs the capability to generate worker attendance reports in Excel in the same format that the shift managers used to prepare by hand for old Doug for the past five years. We will go through the steps to develop the required extension.

First, we need to gather requirements that involve understanding what input is required for the report and what information is required in the report. Let's assume for Doug's report that we need to provide a start date, end date, department, and file name.

Next, we can write the descriptor XML file. It's advisable to use a commercial XML Editing tool so that the descriptor can be verified against the XML Schema. Listing 1 shows the descriptor file for our report. Once the script controller has scanned in this script, a menu item called "Generate Worker Attendance Report" will appear under the Reports menu in the application. When the menu item for this extension is selected, the dialog in Figure 4 would appear. Note that it can also be launched by pressing the F9 hotkey.

The generateWorkerAttendanceReport.py (see Listing 2) creates a report based on the parameters passed in. Note the usages of startDate, endDate, dep, and reportFile in the script. This script generates a simplified version of the report that Doug wants.

Conclusion

You should now understand the design of the Jython Extension Framework and how such a framework allows easy development of custom extensions to your rich GUI Java application. As you can see, once the framework is in place, the customer can quite freely enhance the product to include whatever features may be needed.

The reason that it is now easy for the customer to develop extensions is clear – we have relieved the customer from a lot of the work entailed in binding rich functionality to a GUI application. We have provided the customer with:

1. An easy-to-author descriptor to define input dialog and an entry point into the application with a built-in validation engine (i.e., the XSD).
2. A widely adopted object-oriented scripting language for developing scripts.
3. A well-documented Java API for accessing information within the core product.
4. A neat framework that brings it all together with minimal effort.

Transferring the task of custom extension development from the R&D group to the customer relieves the burden from R&D and allows customers to develop extensions to fulfill their requirements exactly.

Remember, though, this does not come for free – a stable extension framework and a well-documented API together with good support is a must to make this work. ☺

References

- *Jython*: <http://www.jython.org>
- *XML Schema*: <http://www.w3.org/XML/Schema>
- Eckel, B. *Thinking in Patterns* (chapter on Interpreters): <http://mindview.net/Books/TIPatterns/>

Listing 1

```
<scriptExecution displayName="Generate Worker Attendance Report"
accelerator="F9" menu=iReportsi>
  <jythonFile>generateWorkerAttendanceReport.py</jythonFile>
  <parameters>
    <param key="startDate" type="date" required="true" >
      <displayName>Start Date</displayName>
    </param>
    <param key="endDate" type="date" required="true">
      <displayName>End Date</displayName>
    </param>
    <param key="dep" type="package" required="true">
      <displayName>Department</displayName>
    </param>
    <param key="reportFile" type="file" required="true">
      <displayName>Report Output</displayName>
    </param>
  </parameters>
</scriptExecution>
```

Listing 2

```
from com.hrtracker.loader import HRLoader

def writeShiftInfo(emp, shifts):
    outputFile.write("\n")
    outputFile.write("Employee #:" + emp.getNo())
    outputFile.write("-----")
    for shift in shifts:
        outputFile.write(shift.getLocation().getName() + ": " +
            shift.getSummary())

hrManager = HRManager()
emps = hrManager.getEmployeesInDepartment(dep)
outputFile = open(reportFile, "w")
outputFile.write("Shift Report for " + dep.getName() + " department from "
    + "startDate" + " to " + endDate)
outputFile.write("-----")
outputFile.write("\n")
outputFile.write("\n")
for emp in emps:
    shifts = hrManager.getEmployeeShifts(startDate, endDate, emp)

    writeShiftInfo(emp, shifts)

outputFile.close()
```


Table Layout

Replace your GridBagLayout code

by Phil Herold

In this month's article I introduce TableLayout, a robust but easy-to-use LayoutManager for use in any Java Swing application. It's based very loosely on the HTML TABLE paradigm, where components are placed in table cells in row-major order. Vertical and horizontal alignment for the component in a cell can be specified, and a component (cell) may span rows and columns. I also present FormsPanel, a JPanel sub-class that abstracts the underlying TableLayout.

This article assumes you have some familiarity with the design and use of LayoutManagers in Swing.

Background

I have been doing Java Swing applications for many years, and I've never been comfortable using a GUI designer tool, both for the verbose and often horrible Java code they produce and the lack of support for anything other than absolute positioning and sizing of components. I suspect that the quality of these tools has improved tremendously in recent years, but I'm stuck in my ways and have come to be fairly productive in laying out a user interface without the help of a visual design tool. (I have done some desktop application development in C# using Visual Studio .NET, which forces you to use the visual designer of this IDE. I was actually surprised at how productive I could be in this environment, and C# does not have the LayoutManager concept, so perhaps there is hope for me.)

I have found the LayoutManager concept in Swing to be truly inspired, but powerful and easy-to-use implementations a little wanting. My idea of a good LayoutManager is one that is fairly robust yet doesn't require a big learning curve to use or a lot of code to implement. I believe that the code used to lay out a user interface should be fairly self-documenting. In this regard, I have never been a fan of GridBagLayout and its necessary but evil companion, GridBagConstraints. I certainly recognize the power and capability of this LayoutManager, but I have found it difficult to learn and use, and even harder to maintain code that uses it.

I once led a team of UI developers in which I decreed (because I could) that the use of GridBagLayout would

not pass code inspection. Fortunately, we had my TableLayout. I have used one form or another of it on several different projects over the years, and it has slowly evolved during this time as well. I would often maintain that any user interface could be done using a combination of BorderLayout and my TableLayout, given the willingness to subdivide the layout into separate and distinct grids (JPanels).

TableLayout Concepts

TableLayout uses a grid to lay out UI components, which is desirable, since good UI design involves the use of the grid concept. In contrast to Swing's GridLayout, TableLayout does not force every cell in the grid to be the same width and height. (I have never found GridLayout to be useful for much of anything other than a calendar or calculator application.) Rather it lets the preferred height of all the components in a row control the row height, and the preferred width of each component in a column control the width of the column. This by itself is not a new or original concept, but TableLayout brings some other things to the "table" as well. First, it supports the idea of extending the last row or column in the layout to the containing parent's boundaries. Second, it allows you to control the vertical and horizontal alignment of a component in a cell and to have cells span rows and columns if necessary. These are both optional behaviors, and TableLayout takes reasonable defaults so that it can be used in a very straightforward manner to easily lay out a sensible user interface.

CellConstraint

Cell alignment and row/column spanning are controlled using the CellConstraint object, which is only needed to specify something other than the default



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

phil.herold@sas.com

Constraint	Values	Default
align	left, center, right, fill	fill
vAlign	top, middle, bottom, fill	middle
rowSpan	0, 1, 2, 3, 4, etc.	1
colSpan	0, 1, 2, 3, 4, etc.	1

Table 1 CellConstraint name/values

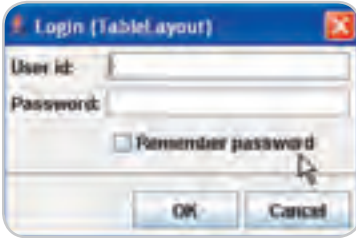


Figure 1 Simple Login dialog



Figure 2 Login dialog with right-aligned labels



Figure 3 Login Dialog using GridBagLayout

alignment or spanning. The `CellConstraint` class has several different constructors, but the easiest one to use is:

```
public CellConstraint(String encoding);
```

In fact, `TableLayout` will accept a `String` as a constraint argument (the `Container`'s `add()` method) and invoke this `CellConstraint` constructor for you (you'll see this in all example code). The encoding is a comma-separated list of name-value pairs that specify alignment and spanning, very similar to HTML `TABLE` attributes. Supported attributes are *align*, *vAlign*, *rowSpan*, and *colSpan* (case-insensitive). *align* is used for horizontal cell alignment and accepts case-insensitive values of *left*, *center*, *right*, or *fill*. *vAlign* is used to specify vertical alignment for a cell, with values of *top*, *middle*, *bottom*, or *fill*. The default value for *align* is *fill*, while the default value for *vAlign* is *middle*. *rowSpan* and *colSpan* each take a numeric argument indicating the number of rows or columns to span. As with the HTML table, a value of 0 for *rowSpan* or *colSpan* means to span to the end of the row or column. Table 1 summarizes the `CellConstraint` values.

You can specify these values directly in other `CellConstraint` constructors if desired. For example:

```
CellConstraint c = new CellConstraint(2, 1, TagValue.LEFT,
    TagValue.MIDDLE);
```

is identical to:

```
CellConstraint c = new CellConstraint("colspan=2, align=left");
```

Both create a `CellConstraint` for spanning two columns, with a left horizontal alignment.

Using TableLayout

`TableLayout` has several constructors that are similar to the constructors for `GridLayout` and accomplish the same. As with `GridLayout`, you can specify either the number of rows *or* the number of columns to be zero, but not both. `TableLayout` has constructors that let you control whether or not the last column in the container is filled, or the last row in the container is filled. The default for the former is **true**, the latter, **false**. `TableLayout` provides getter and setter methods for these attributes as well as the horizontal and vertical spacing between table cells.

How does application code that uses `TableLayout` compare to one that uses `GridBagLayout`? Consider Figure 1, an example of a simple login dialog box.

The dialog consists of three separate components: the top "form" where the login parameters are entered, a horizontal line (normally the horizontal line separator would be accomplished with a custom border around one of the other two panels, but having a component rendering this more clearly demonstrates the capabilities of `TableLayout`; we'll see this `FormSeparator` component again later in this article), and a button panel with two buttons. The dialog uses a `TableLayout` consisting of one-column (and zero rows) to position these three components. The "form" panel uses a two-column (zero rows) `TableLayout`.

You can run the `LoginDialog` test application that comes with the code for this article to create Figure 1. Note that the "User id:" and "Password" labels are left-aligned. Actually, they just use the default horizontal alignment, which is *fill*. Some UI designers believe that labels in a form should be right-aligned. Invoke the `LoginDialog` test application with the command-line argument `align=right`, and you'll see the dialog in Figure 2, which has right-aligned labels.



Figure 4 TableLayout Demo



Figure 5 TableLayout Demo



Figure 6 TableLayout Demo

Figure 3 shows the same dialog produced using a `GridBagLayout` for the login form panel, while Listing 1 shows the code snippet used to produce the two different versions.

From Listing 1, you can clearly see that with `TableLayout`, less code is involved, the code is self-documenting (components are laid out in row-major order), and the code is more maintainable. (Imagine having to insert an additional component in the login form of the dialog. With `TableLayout`, that is very straightforward; not so with `GridBagLayout`, as the additional layout code will likely affect the layout code for components below the new one.) Note the use of the static `TableLayout` `getFillerComponent()` method to create the empty cell below the “Password” label and force the `JCheckBox` into the next column.

The `TableLayoutDemo` class provided with this article's code further demonstrates the capabilities of this `LayoutManager`. Figure 4 shows the resulting window. A `TableLayout` is used to lay out this `JFrame`, and a sub-panel us-

ing a `TableLayout` contains the nine `JButton` components. This `TableLayout` is manipulated using the sliders and checkboxes, which dynamically modify the corresponding `TableLayout` properties.

Listing 2 shows the code used to lay out the nine buttons. Note the alignment values for four of the buttons, and the corresponding behavior as the `TableLayout` attributes are modified.

Figures 5–8 show the effects of dynamically changing the `TableLayout` properties given the alignment attributes in Listing 2.

To demonstrate row and column spanning, run the `ComplexTableLayout` class provided with this article's code. (Code can be downloaded from <http://jdj.sys-con.com>.) Figure 9 shows the resulting window, and Listing 3 shows the code used to produce this.

Note the order in which the components are added to the layout. This would be done very similarly for an HTML table.

How It Works

The code for `TableLayout` is straightforward enough. As components are added to the parent container using a `CellConstraint` (`addLayoutComponent()` method), `TableLayout` ensures that every cell is mapped to a row and column position. Components that span a row or column leave “reserved” cells around them. An inner class, `CellData`, is used to hold the component and the constraint for each cell.

The actual child component layout and preferred size calculation for the parent `Container` is done in the `layoutContainer()` method. First, all of the preferred sizes for each child component are obtained. Based on this information, `TableLayout` is able to calculate the maximum height for each row and the maximum width for each column, i.e., the bounds for each cell. Given the cell bounds, each cell component is assigned its location and size, adjusting for any alignment and spanning. The size of each component in a last row or column is also adjusted if the layout is filling the last row or column. Finally, the preferred size of the parent `Container` is calculated given the cell boundaries and the spacing between each row and column.

Final Thoughts on TableLayout

Here are a couple of final thoughts on this `TableLayout` discussion. First, although buttons are used in the demo applications above, a `TableLayout` cell can obviously host any component, even a `JPanel` with an embedded `TableLayout`. And, although cell alignment and row and column spanning are useful in certain contexts, the real power of `TableLayout` is demonstrated in the `LoginDialog` example, where the user interface layout is constructed in a straightforward and maintainable manner, using nested panels each with a `TableLayout`, if necessary.

FormsPanel – Abstracting TableLayout

Another interesting thing that can be done with `TableLayout` is to abstract its usage by consuming it in some other `Container` implementation. To this end, I created



Figure 7 TableLayout Demo



Figure 8 TableLayout Demo

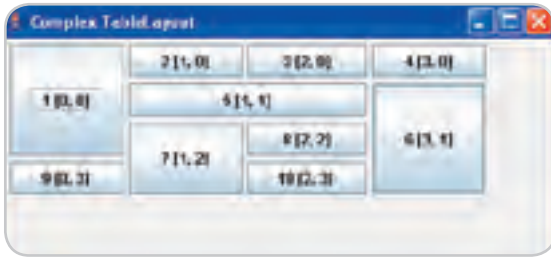


Figure 9 Complex TableLayout

FormsPanel, inspired by the FormFactory in the JGoodies Forms package. (I think very highly of the work that Karsten Lentzsch has done with the various toolkits at JGoodies.)

FormsPanel extends CompositePanel, which extends JPanel. A CompositePanel serves as the base class for a Container in which you want to have control over how (and where) the consuming class places its components. One example is an ExpandablePanel that encapsulates an arbitrary panel of components, hiding them under a header until the header is clicked, then the entire panel is subsequently shown. The code consuming such a panel would not be allowed to set the layout manager for the panel or to add components directly to it, but would instead be doing these operations to a “content pane.” So CompositePanel overrides the add() and setLayout()

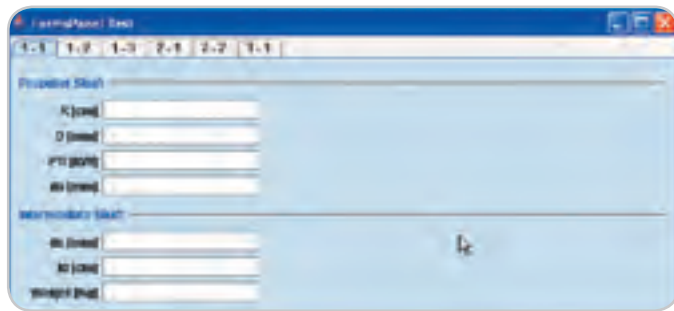


Figure 10 FormsPanel Test, one column

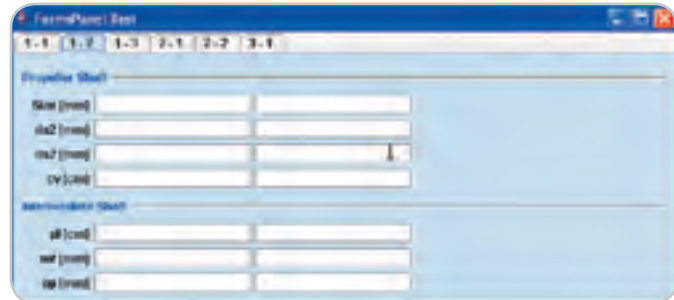


Figure 11 FormsPanel Test, two columns

methods of JPanel. A sub-class must provide other means to add components, such as a getContentPane() method.

The World's Leading Java Resource Is Just a >Click< Away!

JDJ is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of Internet technology professionals who use Java.



ONLY
\$69⁹⁹

ONE YEAR
12 ISSUES

**Subscription Price Includes
FREE JDJ Digital Edition!**

www.JDJ.SYS-CON.com

or **1-888-303-5282**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



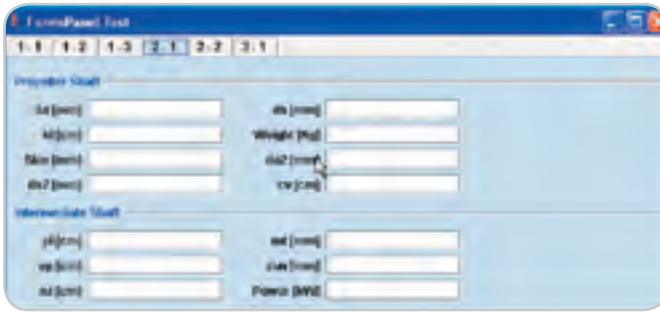


Figure 12 FormsPanel Test, two primary columns, one secondary

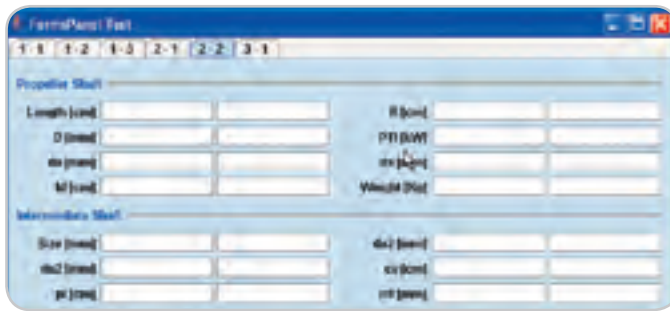


Figure 13 FormsPanel Test, two primary columns, two secondary

FormsPanel provides three `addRow()` methods, each of which adds a row of components to the panel. A row can consist of a label (String) followed by one or more components, or a series of labels followed by a set of components. By invoking the appropriate constructor of FormsPanel, you can specify the number of primary and secondary columns that the panel is to hold. You can also add one or more separator lines to the form with an optional title by calling the `addSeparator(String)` method. When adding components, you can leave a row “short” if needed.

FormsPanel uses a `TableLayout` of course, but does not expose it.

Run the FormsPanelTest provided with the code for this article. Figures 10–13 illustrate several use cases of the FormsPanel contained in this test application. Listing 4 is the code used to produce the panel in Figure 10.

Summary

In this article I have presented a powerful but easy-to-use `LayoutManager` called `TableLayout`. I hope you’ll find it useful in your Swing applications – it has held up very well on many different desktop application development projects. I also demonstrated a technique to build on the power of `TableLayout`.

Listing 1: TableLayout versus GridBagLayout Code

```
if (useTableLayout) {
    loginForm.setLayout(new TableLayout(0, 2, false, false));

    loginForm.add(useridLabel, alignStr);
    loginForm.add(userField);
    loginForm.add(passwordLabel, alignStr);
    loginForm.add(passwordField);
    loginForm.add(TableLayout.getFillerComponent());
    loginForm.add(rememberPasswordCB);
} else {
    GridBagLayout layout = new GridBagLayout();
    loginForm.setLayout(layout);
    GridBagConstraints c = new GridBagConstraints();
    c.anchor = GridBagConstraints.WEST;
    c.insets = new Insets(VGAP / 2, HGAP, (VGAP / 2) + 1, 0);
    c.fill = GridBagConstraints.NONE;
    loginForm.add(useridLabel, c);
    c.weightx = 1;
    c.gridwidth = GridBagConstraints.REMAINDER;
    loginForm.add(userField, c);
    c.weightx = 0;
    c.gridwidth = 1;
    loginForm.add(passwordLabel, c);
    c.gridwidth = GridBagConstraints.REMAINDER;
    c.weightx = 1;
    loginForm.add(passwordField, c);
    c.gridx = 1;
    loginForm.add(rememberPasswordCB, c);
}
```

Listing 2: TableLayout Demo button layout code

```
testPanel = new JPanel(testPanelLayout = new TableLayout(0, 3,
    false, false));
testPanel.add(new JButton("1, 1"));
```

```
testPanel.add(new JButton("1, 2"));
testPanel.add(new JButton("1, 3", "align=center"));
testPanel.add(new JButton("2, 1"));
testPanel.add(new JButton("2, 2"));
testPanel.add(new JButton("2, 3", "align=right"));
testPanel.add(new JButton("3, 1"));
testPanel.add(new JButton("3, 2", "vAlign=bottom"));
testPanel.add(new JButton("3, 3", "vAlign=fill"));
```

Listing 3: Complex TableLayout code

```
setLayout(new TableLayout(4, 4));
add(new JButton(" 1 [0, 0] ", "rowspan=3,vAlign=fill"));
add(new JButton(" 2 [1, 0] "));
add(new JButton(" 3 [2, 0] "));
add(new JButton(" 4 [3, 0] ", "align=left"));
add(new JButton(" 5 [1, 1] ", "colspan=2"));
add(new JButton(" 6 [3, 1] ", "rowspan=3, align=left, vAlign=fill"));
add(new JButton(" 7 [1, 2] ", "rowspan=2, valign=fill"));
add(new JButton(" 8 [2, 2] "));
add(new JButton(" 9 [0, 3] ", "vAlign=top"));
add(new JButton(" 10 [2, 3] ", "vAlign=top"));
```

Listing 4: Single-column FormsPanel

```
FormsPanel panel = new FormsPanel(1);
panel.addSeparator("Propeller Shaft");
panel.addRow("R [cm]", new JTextField(14));
panel.addRow("D [mm]", new JTextField(14));
panel.addRow("PTI [kW]", new JTextField(14));
panel.addRow("da [mm]", new JTextField(14));
panel.addSeparator("Intermediate Shaft");
panel.addRow("ds [mm]", new JTextField(14));
panel.addRow("kl [cm]", new JTextField(14));
panel.addRow("Weight [Kg]", new JTextField(14));
```

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	4, 23
Arcturus Technologies	www.arcturustech.com	703-822-4582	35
Azul Systems	www.azulsystems.com/developerfreedom	650-230-6691	11
ceTe Software	www.dynamicpdf.com	800-631-5006	39
Google	www.google.com/jdj	650-253-0000	41
IBM	www.ibm.com/middleware/tools		Cover IV
ILOG	www.ilog.com	800-FOR-ILOG	13
InterSystems	www.intersystems.com/cache12p	617-621-0600	7
IT Solutions Guide	www.itsolutions.sys-con.com	888-303-5282	59
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	57
Jinfony Software	www.jinfony.com/jp12	301-838-5560	25
MapInfo	www.mapinfo.com/sdk	800-268-3282	27
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	43
ODTUG 2006	www.odtug.com	910-452-7444	45
Parasoft Corporation	www.parasoft.com/jdjmagazine	888-305-0041	9
PDF Tools	http://www.pdf-tools.com	403-932-4220	37
Perforce	www.perforce.com	510-864-7400	Cover II
ReportingEngines	www.reportingengines.com	88-884-8665	5
Smart Data Processing, Inc.	www.weekendwithexperts.com	732-598-4027	61
Software FX	www.softwarefx.com	800-392-4278	Cover III
SYS-CON Events	www2.sys-con.com/events	201-802-3066	49
WebAppCabaret	www.webappcabaret.com/jdj.jsp	866-256-7973	33
Webapper	www.seejava.com	970-223-2278	47
Windward Studios, Inc.	www.windwardreports.com	303-499-2544	19

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *Java Developer's Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in *Java Developer's Journal*. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.

This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

Reach Over 100,000 Enterprise Development Managers & Decision Makers with...



Offering leading software, services, and hardware vendors an opportunity to speak to over 100,000 purchasing decision makers about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management

Don't Miss Your Opportunity
to Be a Part of the Next Issue!

Get Listed as a Top 20* Solutions Provider

For Advertising Details
Call 201 802-3021 Today!

*BASED ON ADVERTISING REVENUE WILL BE DISPLAYED FROM JUNE 1-2007 FORWARD

Letters to the Editors

Looking at Alternatives

["Deriving the Visitor Pattern"
by Nishanth Sastry, Vol. 10, issue 10]

I read the article "Deriving the Visitor Pattern" with interest and found the "why do we have to go through this" style quite engaging. Here are some of my thoughts on the subject and issues brought up in the article.

First off, the question of whether we need the accept() method. To refresh our memory, the way the article presents the visitor pattern is to include the visiting method as a static method of the visitor itself:

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

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

Now the purpose of the accept() method is to dispatch to visitRed and visitBlack methods appropriately for corresponding kinds of nodes. Let's see what alternatives we've got here. The simplest thing that comes to mind is simply add an if/else block right there in the doVisit method to dispatch to the right version of the method:

```
class NodeVisitor {
    ...
    public static void doVisit(List<Node> list)
    {
        NodeVisitor nv = new
NodeVisitor();
```

```
        for (Node n:list) {
            if (n.getClass() ==
RedNode.class) {
                nv.visitRed(n);
            } else if (n.getClass()
== BlackNode.class) {
                nv.visitBlack(n);
            } else {
                throw new NonVis
itableNodeException("I don't know how to
visit: " + n.getClass().getName() );
            }
        }
    }
}
```

You'd shout immediately, "if" statements are considered evil –bad coding! Okay, but let's look a little deeper. The if statements like that are bad because every time you add a new class that your visitor needs

will call back the right method on the visitor. The "if" block solution at least avoids the latter and makes the visiting related code that has to be changed localized to the visitor class itself. Now your Nodes don't need to be aware of the fact that they're being visited, no out-of-place accept() method is needed – how's that for improved cohesion?

If in spite of all those goodies you still can't shake off that righteous indignation at using if-else a pinch of naming convention, a sprinkle of reflection will help you easily eliminate them from the doVisit() method.

Given the above why would anyone ever come up with that crazy idea of using accept() and double dispatch for the Visitor pattern in the first place?? Well, that has to do with the detail mentioned at the beginning that was not really addressed in the original article. And that is the question of where the traversal logic lives. If we assume that the visitor itself has the intelligence to traverse the data structure, then the accept() method mechanism is indeed just unnecessary complexity. If, on the other hand, the traversal logic were to reside elsewhere, e.g., in the data structure, or be defined by each node, possibly a little differently from node type to node type, then the benefits of using the accept() and double dispatch strategy become clear. In that case the code would probably look like:

```
class NodeVisitor {
    public void visitHorizontal(Horizont
alNode n) {
        // do red node specific things
    }

    public void visitVertical(VerticalN
ode n) {
        //do black node specific things
    }
}
```



to handle, you have to go in and update the "if" block. But guess what, even without the if block, with the accept() method approach, every time a new class is to be visited you have to go into the NodeVisitor and modify the code. On top of that you have to add this funny-looking accept method to your new class that


```

interface Node {
    void accept(NodeVisitor visitor);
}

class HorizontalNode implements Node {
    void accept(NodeVisitor visitor) {
        visitor.visitHorizontal(this);
        getLeftNode().accept(visitor);
        getRightNode().accept(visitor);
    }
}

class VerticalNode implements Node {
    void accept(NodeVisitor visitor) {
        visitor.visitVertical(this);
        getLeftNode().accept(visitor);
        getRightNode().accept(visitor);
    }
}

```

You could provide `acceptPreFix`, `acceptPostFix`, and `acceptInFix` methods if different kinds of traversals were needed.

— Pawel Pietrusinski

Training Is Important

[“Corporate Java Training” by Yakov Fain, Vol. 10, issue 9]

Yes, training is important, and the quality of the instructor is a big part of making the class a strong complement to book training. Book training is good, but I agree with your point that most books tend to use oversimplified examples and need to be augmented with experience.

Unfortunately sometimes book training is the only affordable training you can get (fortunately, there are a lot of books available and some of them are good). I can’t speak for everyone, but in the companies I’ve worked for, training was hard to come by. Management in most IT shops is reluctant to pay for it. In some cases, I’ve had to bite the bullet and pay for it myself. That didn’t improve my morale with that company :)

But even taking training isn’t enough...one of the other problems I’ve noticed is that if you don’t use it, you begin to forget it. More IT shops need to be open to letting people apply what they’ve learned in real projects. This would maximize the value of their training costs and potentially keep their systems evolving and improving. Everything doesn’t need to be in the latest and greatest language; it all depends, but if a company is going to invest in training, it ought to put that knowledge to work.

Thanks for the article (and for the training you have offered on this site).

— Bernard Dy

Keep Everything Simple

[“One Size Fits No One” by Joe Winchester, Vol. 10, issue 9]

When I was writing real-time complex software, the complexity was in the application, not in the code. Simple C or C++ would do. Debugging was easy, though there were lots and lots of threads but we knew where the code was and what it was doing.

Then came the tech boom, when every spreadsheet was to be converted into a Web application. Humans want to be praised, so to write a simple application, simplicity does not bring any praise. Then came hoard of XML configurations and frameworks and so on. This caused a small application to look like a bloody complex piece of software.

That trend is still going on. Imagine a complex application written using complex frameworks and XML configuration files. It will be a nightmare to maintain it.

I would like to keep everything simple and nothing more.

— Gurvijay Singh Bhatti

I’ve been doing quite a bit of .NET programming against some “legacy” Web services on Java-based AXIS. Almost all of the linkage between the .NET classes and the Web services XML are done with attributes in the class definition. Like magic, it hides a lot of stuff and makes some debugging a lot harder, and you have to know the correct magic words.

But... this is what Microsoft does: it lowers the bar for entry-level and intermediate programmers, and makes the code much more opaque to those who have to get into the guts to debug or tweak behavior.

Java, on the other hand, should stick to power and elegance through simplicity. If attributes are to be used, then keep the effects simple and clear, and don’t hide the generated code.

— Peter K



WEEKEND WITH EXPERTS

Get a Java injection on the go!

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

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

The next Roadshow is in Edison, NJ on March 11 and 12.

www.weekendwithexperts.com

2005 JCP EC Election Final Results Are In



Onno Kluyt

The election marathon that kicked off in September concluded last month with the open elections for the two JCP ECs. Congratulations and welcome aboard to the elected members: Intel Corporation and Hani Suleiman for the SE/EE EC and Sony Ericsson Mobile Communications AB and Symbian Ltd for the ME EC.

It's now time to meet the individuals designated by these companies to represent them on the ECs.

I'll start with the members elected in the open elections. Intel will be represented on the SE/EE EC by **Wayne Carr** who brings a wealth of technical experience to the committee. Since June 2002, when Intel joined the JCP program, Wayne took an active role in the EC and got involved in the development of various JSRs. He closely followed JSR 185, Java Technology for the Wireless Industry, as an observer, and currently participates as an Expert Group member in the development of several JSRs including JSR 250, Common Annotations for the Java Platform; JSR 270, Java SE 6 ("Mustang") Release Contents; and JSR 277, Java Module System. Wayne also coordinates cross-company Intel participation in the JCP program. He previously represented Intel on the Java ME Executive Committee

Hani Suleiman is an individual developer newly elected to the SE/EE EC. He has been involved with Java technology since 1998, is a regular contributor to a number of open source projects, and also serves as the lead developer/maintainer for some of them. Hani is a member of several JCP Expert Groups including JSR 220, EJB 3, JSR 244, Java EE 5, JSR 245, JSP 2.1, JSR 250, Common Annotations, and JSR 277, Module System for Java. He also serves on the steering committee for the OpenSymphony group, an umbrella organization providing a number of quality open source solutions such as WebWork, OSCache, Sitemesh, and OSWorkflow.

Sony Ericsson Mobile Communications AB will be represented on the ME EC by **Hanz Häger**. Early in his career, Hanz worked primarily with open systems. He

first encountered Java technology around 1995 when porting a Java Virtual Machine to an open operating system for Ericsson. At Ericsson, he developed Internet applications. One of his projects was a community-based group communication solution for mobile phones. He also managed a Unix-based product for the Ericsson AXE-10 Switch used worldwide.

Symbian Ltd will be represented on the ME EC by **Stuart John** who has been Symbian's primary representative on the Micro Edition Executive Committee since the beginning of 2005.

The member companies elected through the ratified ballot in this year's elections are BEA Systems, SAP AG, and SAS Institute Inc. for the SE/EE EC, and Nokia Corporation, IBM, and Philips Electronics UK Ltd for the ME EC. They are represented by IT industry veterans with extensive expertise in Java technology and community work.

Edward Cobb is a vice president with BEA Systems and in this role he oversees BEA's participation in standards and open source organizations, including the Java Community Process (JCP) Program, World Wide Web Consortium (W3C), Organization for the Advancement of Structured Information Standards (OASIS), Object Management Group (OMG), Web Services Interoperability Organization (WS-I), and the Eclipse Foundation. He is also responsible for BEA's involvement in technical collaborations with other software companies including the Web services set of specifications (with IBM and Microsoft) and Service Oriented Architecture (SOA) Programming Model specifications (with IBM).

As vice president of SAP NetWeaver Standards, **Michael Bechtauf** is responsible for defining the company's industry standards strategy. He coordinates all standards-related activities of the SAP Platform Ecosystem including SAP's standards contributions through the JCP Program. In 2002 Michael became SAP's primary representative on the SE/EE Executive Committee and has coordinated the activity of SAP engineers involved in the development of JSRs.

The newest corporate member on the JCP SE/EE EC is SAS Institute and will be represented by its director of Java Development Environments, **Rich Main**. Rich brings to the committee an extensive software development experience and a track record of active participation in industry organizations that promote open standards for the development and deployment of robust Java technology-based enterprise applications.

Moving on to the ME EC, IBM will be represented by **David Girle**, a senior software developer, who previously worked in the Embedded Systems Group at Object Technology International (OTI) in Phoenix, Arizona. David first became involved in the JCP program in 2003 when he participated as an Expert Group member for JSR 226 Scalable 2D Vector Graphics API for J2ME. Two years later, he joined the JSR 246, Device Management API Expert Group as a member.

Nokia Corporation is represented by its director of Java Platform Standardization, **Pentti Savolainen**, a veteran of the mobile communications industry, now re-elected to the ME EC for another three-year term.

Jon Piesing is joining the ME EC for Philips Electronics UK. A senior technical consultant, Jon has worked for the past several years with Java technology in consumer devices, especially the Multimedia Home Platform (MHP) initiative of the Digital Video Broadcasting (DVB) project.

More details about the EC members and their Java technology and community expertise are posted under "Community Focus Series: EC Members Profiles" on JCP.org (http://jcp.org/en/press/ec/commFocus_ec_landing).

For a comprehensive overview of the JCP EC elections check out the PricewaterhouseCoopers site at http://www.jcpelection2005.org/jcp/election_results for open election results and http://www.jcpelection2005.org/jcp/ratification_results for ratification results. Join me in congratulating the newly elected and returning EC members and wishing them a successful term ahead. ☺

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

onno@jcp.org

SoftwareFX

Bringing Your Data, Business & Strategy Into Focus.

New!
version 6.2
Now Includes Maps!

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

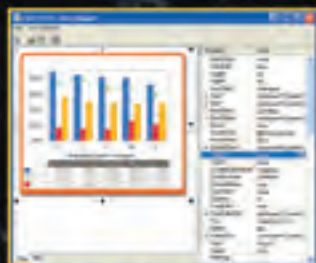


Chart FX for Java Designer

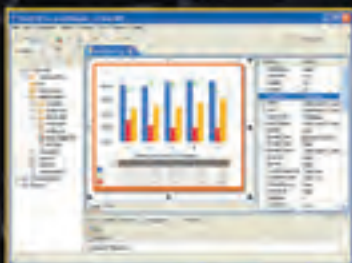


Chart FX for Java - Eclipse

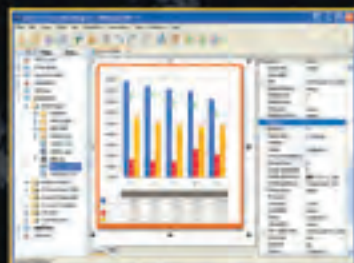


Chart FX for Java - NetBeans

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

©2006 Software FX. All rights reserved. Chart FX is a registered trademark of Software FX, Inc. All other brands are owned by their respective owners.

Rational.

IBM

IBM RATIONAL PRESENTS

YOU ★ VS ★ THE INCREDIBLE SHRINKING DEADLINE

**KNOCKOUT
INNOVATION**

MAIN ATTRACTIONS

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

IBM, the IBM logo and Rational are registered trademarks or trademarks of International Business Machines Corporation in the United States and/or other countries. Eclipse is a trademark of Eclipse Foundation, Inc. Linux is a registered trademark of Linus Torvalds. ©2005 IBM Corporation. All rights reserved.