SYS-CON

BEST PRACTICES FOR SECURING YOUR SOA: A HOLISTIC APPROACH PAGE 36

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

JDJ

JDJ.SYS-CON.COM VOL.11 ISSUE:6

www.AjaxWorldExpo.com
AJAXWORLD
CONFERENCE & EXPO
SEE PAGES 50-51 FOR DETAILS
SANTA CLARA SILICON VALLEY
October 3-4, 2006 • Santa Clara Convention Center

AJAX, Java, Fla$h, and .NET

Rich internet applications market land$cape

RETAILERS PLEASE DISPLAY
UNTIL AUGUST 31, 2006

$5.99US $6.99CAN

PLUS...

- Building an Instant Messaging Application Using Jabber/XMPP
- Deep Diving with ADF Faces
- Swing Baby, Yeah!!!

THE WORLD'S LEADING *i*-TECHNOLOGY MAGAZINE JDJ.SYS-CON.COM

Fireproof Your Code

Prevent Java code fires with JProbe®

Constantly fighting Java code fires? Prevent infernos — before code moves into production — with award-winning JProbe from Quest Software.

Quickly pinpoint Java code hot spots with line-level analysis. Discover and debug memory leaks to dramatically improve performance. Automate the task of performing code analysis during off-peak hours. And release applications with confidence, knowing they have been fully tested. JProbe is the proactive solution that gives you higher levels of productivity and end user satisfaction.

Stop Java code performance flare ups — before they start. Improve code quality and increase application efficiency with JProbe.

Watch JProbe in action. View the new product demo at:
www.quest.com/JavaCode

©2006 Quest Software, Inc. All rights reserved. Quest and Quest Software are trademarks or registered trademarks of Quest Software.
All other brand or product names are trademarks or registered trademarks of their respective holders. 05/2006/JavaCode-JDJ.

Application Management | Database Management | Windows Management

QUEST SOFTWARE®

JAVA DEVELOPER'S JOURNAL

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
Contributing Editor:	Tilak Mitra
Founding Editor:	Sean Rhody

Production

Associate Art Director:	Tami Lima
Executive Editor:	Nancy Valentine
Research Editor:	Bahadir Karuv, PhD

To submit a proposal for an article, go to http://jdj.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 © 2006 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 Megan Mussa, megan@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.

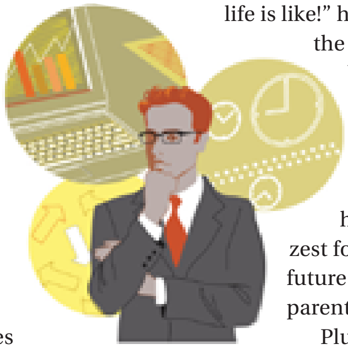
BPA SYS-CON MEDIA MPA

Random Thoughts

Jeremy Geelan

Ever since Google realized that 12% of the population would consult Google prior to seeing a doctor, which was followed by a *British Medical Journal* editorial suggesting that one of the natural next steps for Google would be some kind of medical database for personal use, rumors have been circulating that "Google Health" would be the next addition to the Google stable. Last week the rumors were proven to be true.

Unveiled as just one part of the Google Co-Op, the del.icio.us-like new tagging system that was one of the top items on the agenda at the annual Google Press Day, Google Health is the product of volunteer Web-aware activist MDs like Dr. Enoch Choi, who was asked by Google to help compile a list of the URLs to improve the result sets of health-related searches on Google. These labels will appear at the top of Google search results for search queries regarding any health-related term.



The whole Google Health initiative shows the invisible hand of Google's Adam Bosworth, who recently wowed the audience not only live at the Real-World AJAX Seminar in San Jose but also asynchronously worldwide via SYS-CON.TV (http://sys-con.tv/read/category/1260.htm).

Bosworth explained, in the course of the recent AJAX Power Panel (http://ajax.sys-con.com/read/211578.htm), which also looked at numerous "Web 2.0" issues, the Google approach to things:

Our approach to technology at Google is very simple. We launch a new feature or application and then watch it very, very closely. If users like it, then we add more of that type of feature or functionality, and if they don't like it then we add less of it.

From which one can readily infer that, even if it doesn't take MDs out of the loop completely, which obviously it won't, the traction for Google Health-type initiatives is almost certain to bring it to the top of Google's do-list in terms of fine-tuning, widening, and deepening.

On to JavaOne 2006 and Scott McNealy's Top 10 List

"All the big announcements have been made. I'm the warm-up act for James Gosling," quipped former Sun CEO Scott McNealy on the final day of JavaOne 2006 in San Francisco. "This is what post-CEO life is like!" he added, as he announced the winner of "Bike to Work Week." But he was still able, wholly justifiably, to bask in the reflected glory that is Java. And so he did, with complete humility and with his characteristic zeal and zest for helping the technology future arrive more quickly and apparently undiminished.

Plus he was still able to make the audience laugh.

"There are some changes going on," he said, understatedly, before adding, "All of us in the leadership group are desperately trying to grow a pony tail."

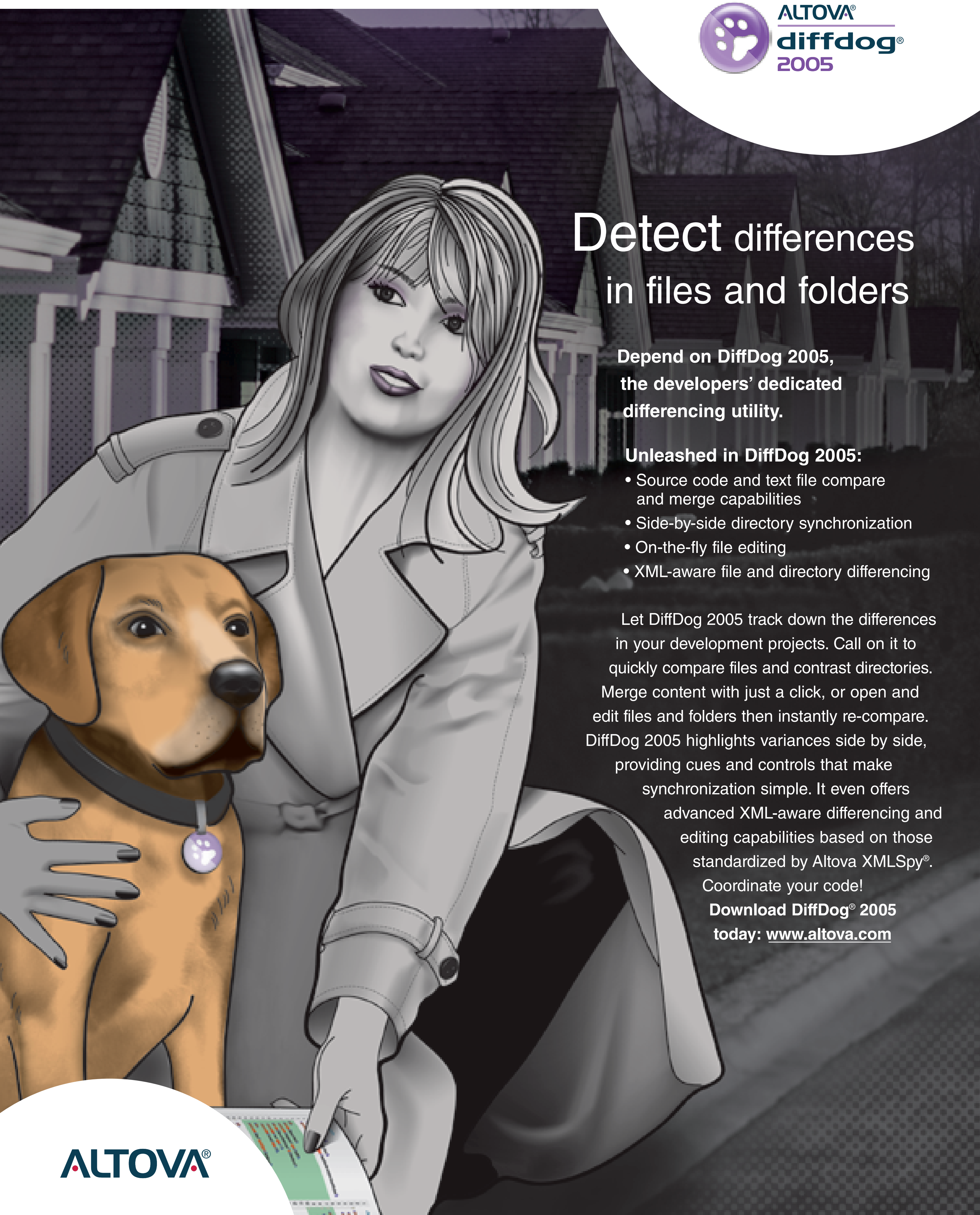
McNealy then presented one of his now-ritual Top 10 Lists. "Here are the Top 10 things about not being CEO," he said:

10. I don't have to apologize for things I say to Wall St. Jonathan does.
9. I'm no longer on the "Most Overpaid CEO" List.
8. I get to just say, "See Jonathan on that."
7. I get to read hockey news without guilt.
6. I need to shave even less often.
5. I don't have to sign the SoX report any more.
4. I have someone to blame now.
3. I can sell my last business suit.
2. Jonathan doesn't play golf, but there are still many Sun customers who play golf, so...
1. My new office is very close to the men's room.

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

jeremy@sys-con.com

ALTOVA®
diffdog®
2005



Detect differences in files and folders

Depend on DiffDog 2005, the developers' dedicated differencing utility.

Unleashed in DiffDog 2005:

- Source code and text file compare and merge capabilities
- Side-by-side directory synchronization
- On-the-fly file editing
- XML-aware file and directory differencing

Let DiffDog 2005 track down the differences in your development projects. Call on it to quickly compare files and contrast directories. Merge content with just a click, or open and edit files and folders then instantly re-compare. DiffDog 2005 highlights variances side by side, providing cues and controls that make synchronization simple. It even offers advanced XML-aware differencing and editing capabilities based on those standardized by Altova XMLSpy®. Coordinate your code!

Download DiffDog® 2005 today: www.altova.com

ALTOVA®

JUNE 2006 VOLUME:11 ISSUE:6

JDJ contents

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

Yakov Fain
Enterprise Editor

JavaOne 2006 Notes

This major Java event was one of the largest conferences ever. Sun Microsystems deserves a lot of credit for accommodating the needs of thousands and thousands of people so efficiently and smoothly. The electronic registration for the show and receiving a special badge with an embedded chip took less than a minute. The auditoriums in which the technical sessions were held were huge – each holding between 700 to 1,000 people. How long did it take to check enrollment and let all these people into the room? Less than 10 minutes. This is clearly registration 2.0. For the most popular sessions, meeting planners arranged so-called overflow rooms in which people could watch a live video broadcast of the session on two huge screens. Serving lunch was another wonder. Fast food chains can only dream of being this efficient. Imagine hundreds of people moving into a huge food court non-stop. The entire lunch process took 10 minutes, unless you wanted to network with other people.

It seems that the slogan of show was "Innovation Happens Elsewhere." At least several executives kept repeating this mantra, meaning that Sun its eyes open, watch what their competitors are up to, and try to learn from them and do better.

Brazil was another highlight of the show. Java is on the rise there, and JavaOne attendees from this soccer-or-die country are slowly considering playing with Java for a change. They were wrapped up in green and yellow flags, and, I would not be surprised if soccer is played at next year's conference. Go Brazil, go!

On the technology side, the most attention was given to NetBeans/GUI and Java ME. I didn't feel the same energy from the Java EE camp.

NetBeans 5.0 offers a nice GUI designer, and I was also impressed with the ease of internationalizing Swing applications. It automatically extracts all hard-coded text from your program into a properties file; just type in the translation of these words/phrases next to each other in a text editor. But Swing programming, with its custom look and feel, listeners, and layouts, still has a lot of room for improvement. As Sheryl Crow sings, "No one said it would be easy, but no one said it'd be this hard." Finally, a JSR was created for data binding and its pre-alpha version was demo'd at the show. "Here's the code before with property change listeners, and here's the code after." Not good enough. It's still difficult. Please take another look at the component-based programming. Look at MXML from Adobe. Look at XAML from Microsoft. The GUI innovation elsewhere is right there. I want to drive your car, but I don't want to know how the engine operates. Give me a minimum of public interfaces like an accelerator and brake pedals, a steering wheel, and tell me how to turn the music on. The argument that Swing offers great flexibility but complexity comes with it would have been fine 10 years ago. I want a simple and flexible GUI tool/framework, and I want it now.

AJAX-related topics were also popular, but mostly because everyone has heard the buzzword, and they were trying to figure out if it's good for anything else other than Google maps. No, JavaScript is not my programming language of choice. Some vendors were pushing their AJAX frameworks, but none of them looked serious to me.

Of course, the question of whether Java will be open sourced was raised. Rich Green from Sun gave a politically correct answer, "The question is not when, but how." Where do people learn to give such open ended answers? This phrase doesn't mean yes or no. Java is a lot bigger than Sun Microsystems, but Sun controls this JButton. I don't really want Java to be open sourced, as long as Sun starts moving a bit faster when it comes to responding to requests from the Java community. For example, they maintain an online list – Top 25 Java Bugs (http://bugs.sun.com/bugdatabase/top25_bugs.do). People can vote for a particular bug to get the attention of Sun engineers. Guess what? Some of these bugs are 6–7 years old. If Sun can offer a faster way of fixing bugs and implementing the most requested language features, there is no need to open source the language. Just give an HTTPResponse for each of our HTTPRequests. Unfortunately, we are still getting the 404 error quite often. The Java Champions program should help in relaying the feedback from the masses back to Sun's engineers.

Having said all this, I'm taking my hat off to Sun Microsystems for putting on such an amazing event as JavaOne 2006. This was not just another technical conference. This was the best event that Java developers could attend. Try to be there next year.

Yakov Fain is a principal consultant of Farata Systems. He's responsible for the enterprise architecture and emerging technologies. Yakov has authored several Java books, dozens of technical articles, and his blog is very popular. Sun Microsystems has nominated and awarded Yakov with the title Java Champion. He leads the Princeton Java Users Group. Yakov holds a BS and MS in applied math.

yfain@faratasystems.com

SYS-CON MEDIA

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*
Advertising Sales Manager:
Megan Mussa *megan@sys-con.com*
Associate Sales Manager:
Kerry Mealia *kerry@sys-con.com*
Lauren Orsi *lauren@sys-con.com*

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

Production
Lead Designer:
Tami Lima *tami@sys-con.com*
Art Director:
Alex Botero *alex@sys-con.com*
Associate Art Directors:
Abraham Addo *abraham@sys-con.com*
Louis F. Cuffari *louis@sys-con.com*

Web Services
Information Systems Consultant:
Robert Diamond *robert@sys-con.com*
Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Wayne Uffleman *wayne@sys-con.com*

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

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

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

RCP Developer™ is created by the experts who brought you the popular book *Eclipse: Building Commercial Quality Plugins* — Eric Clayberg & Dan Rubel

Java on the desktop is back!

RCP Developer™

Instantiations RCP Developer harnesses the power of the Eclipse Rich Client Platform. Quickly create native desktop applications that are rich, cross-platform, and high-performance. The vast expertise of the Eclipse developer community and the experience of Instantiations lets you...

STAND ON THE SHOULDERS OF GIANTS

Create blazing fast rich Java clients.

Tap into the ultra high performance Eclipse graphical user interface libraries and ensure a native look and feel.

Build rich Java client applications faster.

Focus on creating business specific functionality rather than reinventing the low-level logic required for applications to run.

Deliver a more consistent user experience.

Tame the most demanding user interface requirements and reliably deliver a rich and consistent user experience across all applications.

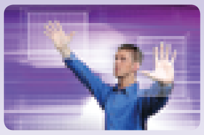
Fully exploit the Eclipse Rich Client Platform.

"The Eclipse Rich Client Platform is the leading framework for creating Java client applications. RCP Developer is the first product to bring comprehensive application construction, GUI testing and packaging of rich-client applications to Eclipse RCP."

—Mike Milinkovich, executive director of the Eclipse foundation

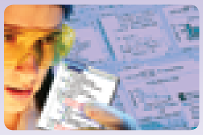
RCP Developer™

SWT Designer™



Design. Quickly create views, editors, perspectives, and preference pages for your applications. Intuitive visual design and access to the high performance SWT GUI framework of Eclipse RCP.

WindowTester™



Test. Automatically test the GUIs that comprise Eclipse RCP applications. Minimize the need to hand-code GUI test cases. Automated recording, test generation, assertion coding and playback facilities.

RCP Packager™

Deploy. Streamline build and deployment for Eclipse RCP applications. Assemble application elements into a single package. Quickly create and maintain high-quality Eclipse RCP installations.

Download a risk-free trial copy:
www.instantiations.com/rcpdeveloper

instantiations
the development tools experts

www.instantiations.com 1-800-808-3737

The leader in Eclipse RCP development tools

eclipse FOUNDATION MEMBER

© Copyright 2006 Instantiations, Inc. RCP Developer, SWT Designer, WindowTester and RCP Packager are trademarks of Instantiations. All other trademarks mentioned are the property of their respective owners.

IBM, the IBM logo and Rational are registered trademarks of International Business Machines Corporation in the United States and/or other countries. ©2006 IBM Corporation. Eclipse is a trademark of Eclipse Foundation, Inc. Java is a trademark of Sun Microsystems, Inc. All rights reserved.

_INFRASTRUCTURE LOG

_DAY 15: This project's out of control. The development team's trying to write apps supporting a service oriented architecture, but it's taking forever. Gil's resorted to giving them all coffee IVs. Now they're on java while using JAVA. Oh, the irony.

_DAY 16: Big crisis—we've just run out of half-and-half!!

_DAY 18: I've found a better way: IBM Rational. It's a modular software development platform based on Eclipse that helps the team model, assemble, deploy and manage service oriented architecture projects. The whole process is simpler and faster, and all our apps are flexible and reusable. The software we write today will be the software we use tomorrow. :)

_The team says it's nice to taste coffee again, but actually drinking it is sooo inefficient!

Download the IBM Software Architect Kit at:

IBM.COM/**TAKEBACKCONTROL**/FLEXIBLE

Exploring the Architecture of Javelin with a Dependency Structure Matrix

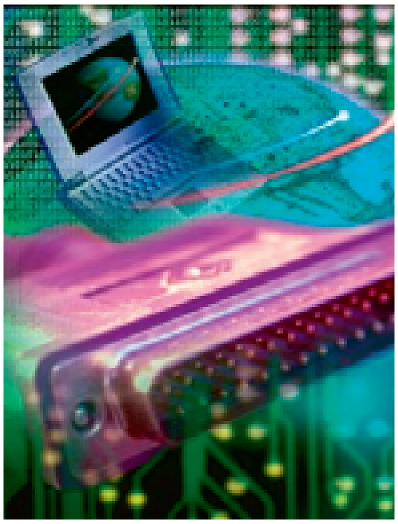
by Tim Hanson and Neeraj Sangal

How a DSM-based analysis would fare when applied to Javelin

Javelin is a compiler framework written entirely in Java. It lets developers support new languages or extend current languages. It also provides interfaces to the parsing and compilation process. For instance, an IDE's editor can take advantage of Javelin to support syntax highlighting, code completions, and refactoring. Javelin could be extended to compile languages like XSLT, XQuery, and PHP. It already contains a Java compiler and an XML Schema compiler, which is fully integrated into BEA Workshop and provides support for various editing and compiling tasks. The JSP compiler also uses the Javelin framework.

However, this article isn't about Javelin. It's about the *architecture* of Javelin. We decided to explore the architecture of Javelin using its inter-module dependencies. For purposes of this analysis, all we needed was the Javelin jar file. Given that Javelin has more than 800 classes and thousands of dependencies between these classes, we needed a representation that could scale up while allowing us to dig into any specific dependencies.

Our choice for this analysis was to use a Dependency Structure Matrix (DSM) to represent the architecture. This powerful new approach (see http://sdg.lcs.mit.edu/pubs/2005/oopsla05-dsm.pdf) has recently been introduced for specifying software architectures but it's been built on ideas that have actually been around for a long time. Historically, DSMs have been used to represent complex product development processes. The new approach has two key elements: (1) A precisely ordered hierarchical decomposition and (2) explicit control over allowed and disallowed dependencies between the subsystems. DSM provides a compact representation that can easily scale up to tens of thousands of classes, whereas conventional box-and-arrow diagrams become unusable in systems composed of even a few hundred classes.



Tim Hanson is the Javelin compiler architect at BEA Systems. Tim developed much of BEA's Java compiler - one of the earliest 1.5-compliant implementations. He has written numerous other compilers, including a CORBA/IDL compiler while at IBM, and an XQuery compiler.

tim.hanson@bea.com

We used Lattix LDM for this analysis. Lattix LDM recently won the Jolt award for design and modeling tools, a category that has traditionally been dominated by UML tools. We wanted to see how a DSM-based analysis would fare when applied to Javelin.

Our goal for this project was to discover the architecture and identify the undesirable couplings that might have crept into the architecture, since it's undergone development over the last few years. Ultimately, we want to improve the modularity of the architecture so that it's easier to understand and maintain.

An Initial DSM for Javelin

We loaded the file, javelinx.jar, into Lattix LDM. The initial picture represents the decomposition of the jar organized as a tree of packages and classes. You can expand each level of the tree to display the next level of decomposition. At any given level of expansion the matrix shows you the dependencies between the subsystems. You can select any cell of the matrix to see the exact dependencies between any two of the subsystems.

The initial DSM, in Figure 1, shows a decomposition based on a package structure. We expanded the *javelin* and the *com.bea* packages to reveal the packages inside them. Note that when a package contains classes and packages, Lattix LDM creates a partition called "*" that contains all the classes at the root level of that package.

To see the dependence of one subsystem on another just read down the column. By definition the DSM is a square matrix where the rows and columns are numbered to correspond to a subsystem. For instance, the selected cell in Figure 1 shows the dependence of the source subsystem, *javelin.java,* on the target subsystem, *com.bea.languages.java*. In Lattix LDM, as you select a cell, a usage pane on the right shows you the exact dependence of each class in the source on classes in the target.

Reorganizing the DSM To Represent the Javelin Architecture

Trying to make sense of all these dependencies at first appears daunting. However, the use of partitioning algorithms can quickly bring order to it. Partitioning can be applied to any subsystem of the tree and it reorders the subsystems, which are directly within the selected subsystem. Partitioning is a precise mathematical algorithm, which returns a block triangular matrix. This means that those subsystems, which are used more by others, are moved closer to the bottom and those subsystems, which tend to use other subsystems more are at the top. For instance, a utility package would tend to move to the bottom because it's more likely to be used by other packages. Conversely, a package called client is more likely to be at the top because it's going to make use of the services offered by other packages.

The interesting cases are the ones where the abstraction is blurred. What if the interface depends on the implementation? What if the compiler depends on the client? Those cases require untangling of the code. This can sometimes be as easy as moving a few files around but more generally requires code refactoring to achieve the separation of abstractions. The purpose of architecture discovery is to decompose the problem into clean abstractions. The initial step is to start with a decomposition that uses packages; we partition them to see if reordering them based on their dependencies will reveal the structure.

Note that to come up with this structure we partitioned many of the subsystems. Partitioning the *$root* node (the top level) moved the *javelin* package above the *com* package. This conforms to our expectations because *com.bea* and *com.sun* packages contain the interface while *javelin* is the implementation of that interface. We expanded *com.bea* to reveal *com.bea.languages* and *com.bea.compiler*. These were, in turn, expanded to reveal their subsystems. Paritioning *com.bea* moved *com.bea.compiler* underneath *com.bea.languages*, once again confirming our expections that *com.bea.compiler* contains the general API extended by *com.bea.languages*. Finally, we partitioned *com.bea.compiler*, *com.bea.languages,* and *javelin*. This reordering based on partitioning gave us a sense of which subsystems underlie which other subsystems. It also showed us that many of the subsystems were coupled. Figure 2 shows the results of partitioning these selected subsystems.

Recognizing that Javelin had an infrastructure built around the Java compiler we created an appropriate abstraction to reflect it. We also recognized that there were a few subsystems that were either obsolete or simply reflected abstractions built on top of other systems. We moved them out to the top level. Having made these manipulations, we came up with the conceptual picture in Figure 3.

The conceptual diagram is derived from the matrix view to reflect the decomposition of Javelin. We used the layout further to show both “horizontal” and “vertical” splitting. The horizontal splitting is the splitting up into layers. For instance, the *API* is layered to be underneath *javelin* to indicate that we expect *javelin* to depend on the *API* but not vice versa. Similarly, we use vertical splitting to indicate independent components. For instance, *com.sun.mirror.apt* and *com.bea* in the *API* are split up to indicate that these two have no dependencies on each other.

Identifying the Improvements

Examining the DSM corresponding to the conceptual architecture then let us determine the dependencies that were the targets for elimination through refactoring.

Area 1 In Figure 4 shows the dependence of the *API* on *javelin* while Area 3 shows the dependence of *javelin* on the *API*. Normally, we expect the implementation to depend on the interface, not the other way around. As a result, we would target the dependencies in Area 1 for removal through refactoring.

Most of these references come from the fact that the original interfaces to the Java typesystem were removed from the public API and refactored into *javelin.java.typesystem*. References to these newly private classes were never fully removed from the public API.

Area 2 shows the dependence of *com.bea.compiler* on *com.bea.languages*. Once again we expect the *com.bea.languages* to depend on *com.bea.compiler*,

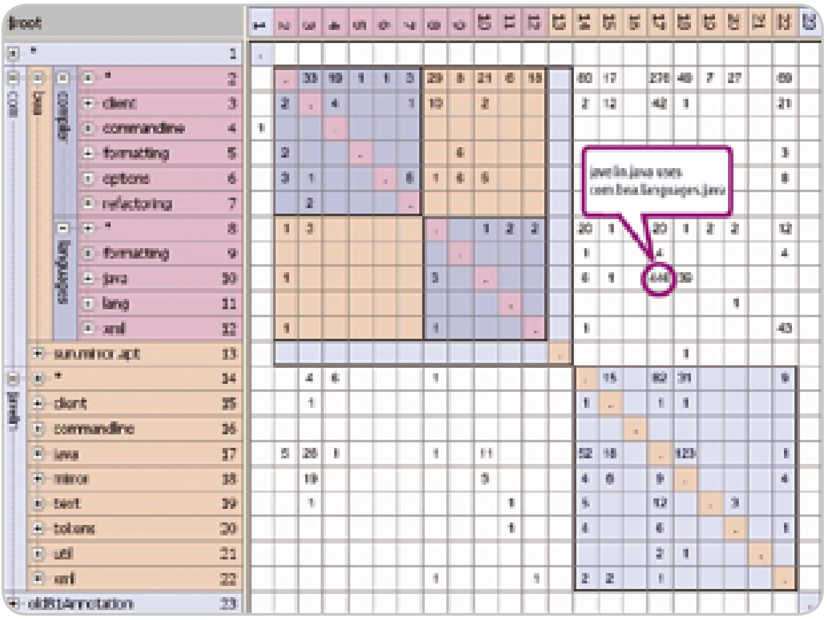


Figure 1 The initial DSM

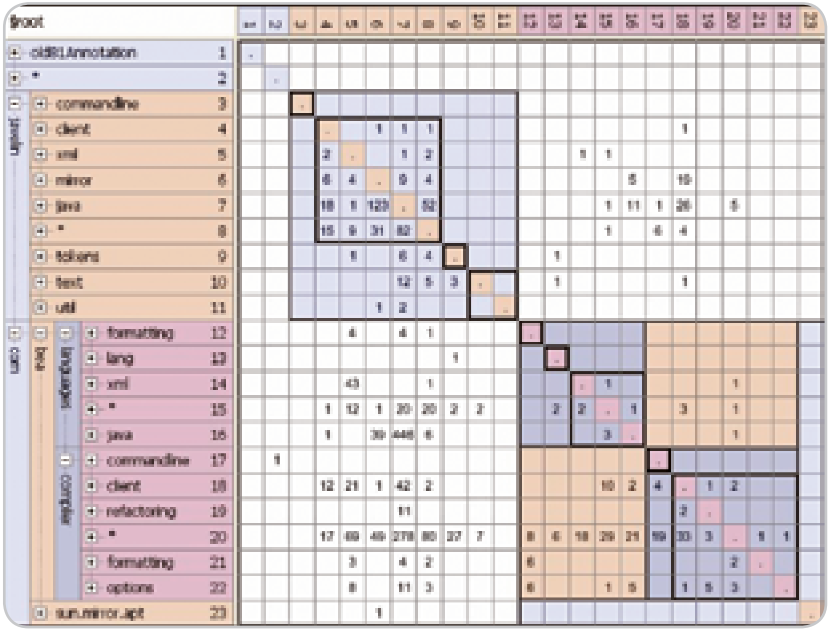


Figure 2 DSM after partitioning certain subsystems

Neeraj Sangal is president of Lattix Inc., which specializes in software architecture management solutions and services. He has analyzed many large proprietary and open source systems. Previously, Sangal was president of Tendril Software that pioneered model-driven Enterprise Java Beans development and synchronized UML models for Java. Prior to Tendril, he managed a distributed development organization at Hewlett Packard.

neeraj.sangal@lattix.com

not the other way around. As a result, we'd like to get rid of the dependences in this area.

Notice also that *javelin.** and *javelin.java* are tightly coupled together. This is an implementation artifact that arose because the team that wrote the Java compiler wrote the compiler framework. We never tried to decouple these, but that seems like a mistake. If the Java language can't be implemented using only the bare compiler framework, how can we expect another language to be implemented? In particular, we never distinguished a generic project from a Java project. The base project class has all the methods to get and set the classpath, as well as find a Java type by its qualified name. This should be refactored to separate the notion of a generic project from a Java project.

The dependency of *javelin.xml* on *javelin.java* tells us that the implementation of our XML schema compiler depends on the **implementation** of our Java compiler because it needs to generate Java bindings to access XML instances. This tells us that our API isn't sufficient to meet our client's needs. This is an obvious area for refactoring. Any features needed by the XML Schema compiler should be put in the interface layer. Intuitively, we know that that's possible because the JSP compiler, which also generates Java code, uses only the public API. This sort of interface creeping occurred primarily because the same team developed both parts of the code.

Finally, we haven't yet looked inside *javelin.java*. This is the heart of the infrastructure and needs a more in-depth examination. It also shows the power of the approach. The DSM lets us systematically analyze our system in a top-down fashion. For larger projects, this means that different groups can focus on different parts of their system while maintaining the overall architecture.

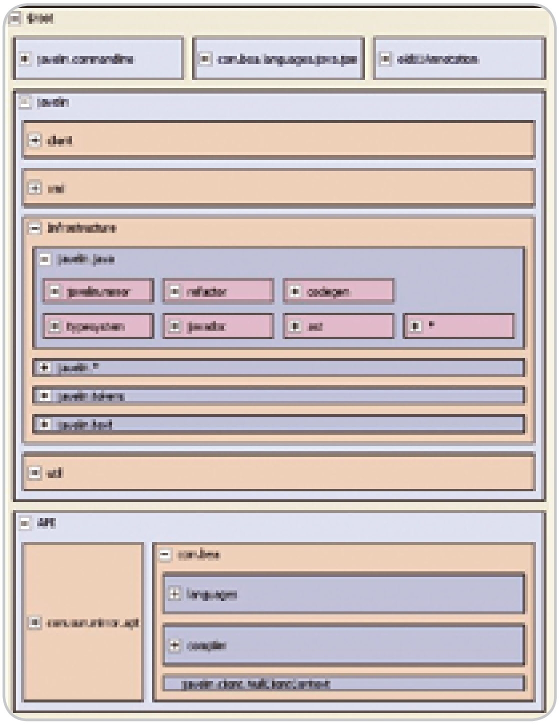
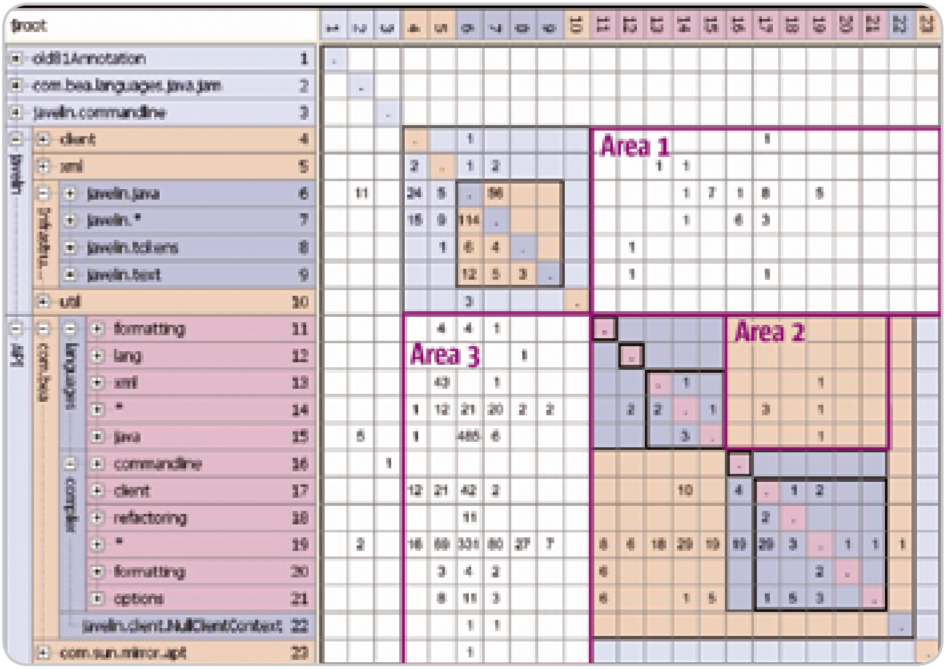


Figure 3 Javelin's conceptual architecture



$root		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
old81Annotation	1	.																						
com.bea.languages.java.jam	2		.																					
javelin.commandline	3			.																				
client	4				.		1											1						
xml	5				2	.	1	2						1	1									
javelin.java	6		11		24	5	.	56							1	7	1	8		5				
javelin.*	7				15	9	114	.							1		6	3						
javelin.tokens	8					1	6	4	.				1											
javelin.text	9						12	5	3	.			1					1						
util	10						3				.													
formatting	11					4	4	1				.												
lang	12								1				.											
xml	13					43		1						.	1					1				
*	14				1	12	21	20	2	2			2	2	.	1		3		1				
java	15		5		1		485	6							3	.				1				
commandline	16			1													.							
client	17				12	21	42	2							10		4	.	1	2				
refactoring	18						11											2	.					
*	19		2		16	69	331	80	27	7		6	6	16	29	19	19	29	3	.	1	1	1	
formatting	20					3	4	2				6								2	.			
options	21					8	11	3				6			1	5		1	5	3		.		
javelin.client.NullClientContext	22						1	1															.	
com.sun.mirror.apt	23						1																	.

Figure 4 DSM corresponding to the conceptual architecture

Conclusion

Once we finish our analysis we'll create design rules that enforce the architecture so future developers don't end up creating unnecessary coupling. On a DSM this is represented by annotating the cells to indicate those cells where dependencies are allowed and those where they aren't. Once these rules are in place, they can be verified as part of the regular builds.

We note that we were able to identify undesirable couplings very quickly. There are more than 800 classes in the Javelin jar file, which have nearly 80,000 dependencies among them. The power of the DSM is that it could incorporate our architecture knowledge and we could winnow down the dependencies that we needed to worry about to a tiny number.

We were also pleased with the matrix representation. The conventional visualizations for representing architectures have typically used box-and-arrow diagrams. These diagrams get cluttered quickly and simply don't scale when systems get large. The matrix representation scaled easily, letting us create new abstractions and try out what-if scenarios by moving classes and packages. All of the changes that we made to the architecture were maintained in a WorkList that could then become a task list for refactoring.

We urge you to try this new approach on your own software to see how easy and effective it is.

Useful Links

You can learn about BEA's Javelin compiler from http://dev2dev.bea.com/wlworkshop/javelin/index.html

You can download and try Lattix LDM from http://www.lattix.com

You can learn more about DSMs from the following URLs:

- http://www.dsmweb.org
- http://sdg.lcs.mit.edu/pubs/2005/oopsla05-dsm.pdf
- http://www.lattix.com/technology/whatisdsm.htm

Bridge the Gap Between Windows and Java

Give your users the freedom to use Microsoft Windows based reporting, analysis, and development applications against your Java data source

OpenAccess™ SDK lets you quickly implement a custom ODBC driver with full SQL support for any data source or application.

Key OpenAccess™ SDK Features & Benefits

- Write the ODBC driver with SQL support in just weeks - 99% of the required code is contained in the SDK.
- Use Java to implement the data source specific code.
- Guaranteed to work with ODBC compliant applications – hundreds of organizations have used OpenAccess™ to set their data free.
- Fully Unicode enabled for globalization.
- When required easily add JDBC, OLE DB, or .NET drivers.
- Designed from ground up for reliability, scalability, and portability.

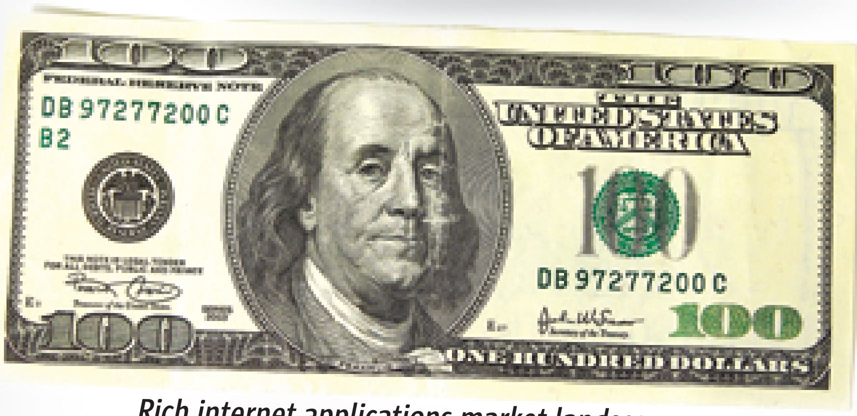
OpenAccess can also be used to add ODBC support on top of your JDBC driver and to add a security layer in between a client application that uses ODBC or JDBC and the data source. Please call us or visit our web site to learn how different problems have been solved using OpenAccess.

To learn more about OpenAccess™ SDK for Java or to try it out, go to **www.OpenAccessSoftware.com/JDJ or call 1.888.805.ODBC**

OpenAccess SOFTWARE

OpenAccess Software, Inc. 2001 Gateway Place, Suite 100 San Jose, CA 95110 www.OpenAccessSoftware.com
Copyright © 2006 OpenAccess Software, Inc. All rights reserved. OpenAccess is a trademark of OpenAccess Software, Inc. All other brand names are trademarks of their respective holders.

AJAX, Java, Flash, and .NET



Rich internet applications market landscape

by Coach K. Wei

Enterprise Rich Internet Applications (RIAs) are the next evolution of business application development. There are four different approaches to RIA development — AJAX, Java, Flash, and .NET — and many different RIA solutions available today. This article answers the following questions: What are enterprise RIAs? Which approach should you use? Which solutions are appropriate for you? And how are RIAs being adopted today?

Welcome to a New Paradigm

The Web began as an environment for content sharing and small-scale data transfer via e-mail, newsgroups, and so forth. These initial uses quickly led to more sophisticated applications particularly in the e-commerce arena. However, the Web wasn't architected with rich application services in mind. Its document-centric model has by and large thwarted developers looking to leverage the Web as a platform for enterprise-class applications.

Beginning in early 2005, popular new Web applications like Gmail, Google Maps, and Flickr awakened the entire Internet community to the possibility of a far richer Web experience. Web developers were quick to discover and leverage the technical approach that these applications used, which was first termed AJAX (for Asynchronous JavaScript and XML). The excitement around AJAX focused more attention on the wide spectrum of Rich Internet Application (RIA) development tools and the various approaches available.

Gartner calls RIAs "the next evolution of the Web." They represent the next big evolutionary step for enterprise application development. They deliver the high performance and robust functionality of desktop or client/server software combined with the universal reach, no-install deployment, and centralized management of browser-based apps. RIAs represent the next paradigm for building, deploying, and maintaining enterprise applications. The impact of RIAs on business will match that of PC desktop computing — bringing operational efficiency and productivity to a whole new level, while decreasing costs.

Enterprise RIAs versus Consumer RIAs

In general, Rich Internet Applications can be classified into two categories: enterprise RIAs and consumer RIAs. Enterprise RIAs refer to RIAs whose users are primarily business users. This includes internal enterprise IT applications as well as B2B applications. Consumer RIAs refer to RIAs whose users are primarily individual consumers, such as consumer Web sites, as well as B2C applications.

Enterprise RIA Opportunities

A growing number of *Fortune* 1000 companies have already adopted Enterprise RIAs or will do so in the near future. According to Gartner, "By 2010, at least 60% of new application development projects will include RIA technology, and at least 25% of those will rely primarily of RIA (0.7 probability)."

Organizations that seek competitive advantage or greater operational efficiency are increasingly exploiting RIA technology to re-architect traditional client/server applications, such as those written in Visual Basic or Java Swing. RIAs can offer all the rich features and performance benefits of these "thick client" alternatives, while eliminating the need to install and maintain a custom client on user desktops.

Enterprise RIA technology is also of great value to companies that wish to improve the performance and user experience of traditional HTML-based Web applications. RIAs can radically improve the responsiveness of browser-based applications because they enable processing to take place on the client, thus reducing network demands in comparison to HTML's inefficient "click-wait-refresh" model.

Moreover, Enterprise RIAs mesh perfectly with Service Oriented Architecture (SOA) and Web Services initiatives (see references to Dion Hinchcliffe and Dana Gardner). Their role in this model is to deliver SOA-based services to users via a wide range of devices, while at the same time reducing the cost and complexity associated with managing networks and client-side deployments. In particular, RIAs can reduce the need for development teams to create multiple interfaces to applications using disparate technologies, as is the case with client/server and HTML-based architectures today. As SOAs become the method of choice to deploy both new and existing business services, enterprises will increasingly employ RIAs to bring those services to their end users.

Figure 1 UI created by Laszlo code

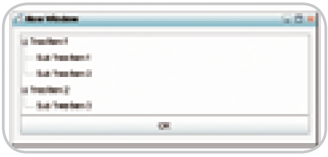


Figure 2 Example 2 UI Display created by Nexaweb code

Coach Wei currently serves as CTO for Nexaweb (www.nexaweb.com), developers of the leading software platform for building and deploying Enterprise Rich Internet Applications. Previously, he played a key role at EMC Corporation in the development of a new generation of storage network management software. Coach has his master's degree from MIT, holds several patents, is the author of several technology publications, and is an industry advocate for the proliferation of open standards.

cwei@nexaweb.com

Register for an Online Webinar
www.opnet.com/pinpoint

STILL SEARCHING ?

MAKE ANSWERS TO PERFORMANCE PROBLEMS COME TO YOU.

OPNET Panorama offers powerful analytics for rapid troubleshooting of complex J2EE/.NET applications. Panorama quickly identifies how application, web, and database servers are impacting end-to-end performance. With Panorama, you can pinpoint the source of a problem, so time and money aren't spent in the wrong places.

The world's most successful organizations rely on OPNET's advanced analytics for networks, servers, and applications.

PANORAMA

OPNET Panorama
Real-Time Application Analytics

www.opnet.com/pinpoint

OPNET®
Making Networks and Applications Perform™

OPNET Technologies, Inc. 7255 Woodmont Avenue, Bethesda, Maryland 20814 phone: (240) 497-3000 • e-mail: info@opnet.com • NASDAQ: OPNT

© 2006 OPNET Technologies, Inc. All rights reserved. OPNET is a registered trademark of OPNET Technologies, Inc.

Approaches to RIA Development

Though it's still evolving, today's RIA marketplace is already rich in choice, and IT teams are challenged to match technology options with business goals. But while there are a variety of approaches and products available for building and deploying RIAs, they nearly all fall into one of only two basic categories:

	Strengths	Weaknesses
Java	• Broad industry support • Large developer community • Widely adopted in the enterprise • Robust performance, scalability, and reliability • Robust OOP model • Designed for team development • Maintainable and manageable code	• Requires a higher programming skill set than scripting • Requires a Java Virtual Machine to run the application
.NET	• Supported by Microsoft • Robust performance, scalability, and reliability • Robust OOP model • Designed for team development • Maintainable and manageable code	• Supported only by Microsoft • Requires a .NET Virtual Machine to to run applications • Requires a higher programming skill set than scripting
AJAX	• Highly compatible with existing HTML infrastructure and content • Built-in support in most browsers — therefore easy to try without needing additional software	• DHTML/JavaScript code is difficult to develop and maintain • Not designed for team development • Performance and functionality limitations
Flash	• Supports rich UI features like animation and video • Flash engine is small and widely available • Large Flash designer community	• Performance and functionality limitations • Flash designers are not developers (lack of mindshare ammong enterprise developers)

Table 1 Strengths and weaknesses of RIA approaches

	Runtime Solutions	Tooling
Java	• Nexaweb Platform • jRex • Thinlet	• Nexaweb Studio
.NET	• XAML (Microsoft)	• Visual Studio
AJAX	• Open Source: –Dojo, Apache, Kabuki, Rico, DWR... • Closed Source: –Bindows, Backbase, JackBe, Isomorphic...	
Flash	• Adobe Flex • Laszlo	• Adobe Flex Builder

Table 2 RIA solutions

- Object-oriented programming (OOP) based approaches, such as Java and .NET and
- Scripting-based approaches, including AJAX and Flash

The comparative strengths and weaknesses of the different RIA approaches center largely on the programming model and application execution environment they employ. The programming model impacts development and maintenance efforts, the availability and cost of developer skills, the availability of industry and development community support, and such. The execution environment significantly impacts not only application performance, functionality, and reliability but the deployment model as well.

Comparing RIA Approaches

In general, OOP approaches confer the advantages of strongly typed object-oriented programming such as improved code maintainability and reuse, and are better suited for enterprise-class applications. Scripting-based approaches offer the advantages of scripting and are best suited to quickly finishing simple tasks done.

Among the OOP-based approaches:

- ***Java-based RIAs*** generally leverage a client-side Java engine. Client-side application logic (if any) is written in Java, while the UI is defined using XML. The client-side components execute inside a Java Virtual Machine (JVM) that is typically embedded in a browser.
- ***.NET-based RIAs*** rely on a .NET virtual machine. The UI can be programmed using .NET controls or Microsoft's XAML. Client-side logic is generally programmed in C# or a similar language.

Among scripting-based approaches:

- ***AJAX-based RIAs*** typically employ a relatively simple browser-based JavaScript library for greater interactivity. The UI is most often defined using DHTML/JavaScript; client-side logic is also written in JavaScript. The client-side execution environment is the browser itself.
- ***Flash-based RIAs*** run in the Flash animation engine. The UI is defined using SWF (a proprietary binary format for defining Flash-based movies) or with XML markup compiled into SWF. Client-side logic is programmed in ActionScript, a scripting language developed by Macromedia (now Adobe).

Table 1 summarizes the advantages and disadvantages of these four approaches.

RIA Solutions Today

There are many RIA solutions available today. Each of them fits into one of the approaches mentioned above. Some of the solutions come with tooling that can simplify development and maintenance. Table 2 shows a list of solutions available today.

General RIA Programming Model

Although there are many different RIA solutions based on different underlying technology platforms, the general RIA programming model is actually converging into a single common model.

Fiorano SOA™ 2006

The Quickest path to an SOA

- **FioranoMQ™ 2006** – world's fastest, most scalable JMS
- **Fiorano ESB™ 2006** – CAD/CAM for distributed applications
- **Fiorano BPEL Server** – simplifying business process orchestration
- **Fiorano Tools** – BPEL Studio, Mapper, FEPO, etc
- **Fiorano Components** – 60+ pre-built adapters

Benefits

- **Adherence to popular industry standards** - JMS, COM, .NET, JCA, JMX, BPEL, SOAP, etc.
- **Multi-language, Multi-platform, Multi-protocol**
- **Unmatched Scalability and High Performance**
- **Quick, Measurable ROI**

Download your copy of Fiorano today!

www.fiorano.com/downloadsoa

Fiorano®

Enabling Change at the speed of thought

© Copyright 2006, Fiorano Software Inc. All rights reserved. Fiorano, Fiorano SOA Platform, Fiorano Business Integration Suite, Fiorano ESB, FioranoMQ and the Fiorano logo are trademarks or registered trademarks of Fiorano. Other trademarks are property of their respective owners.

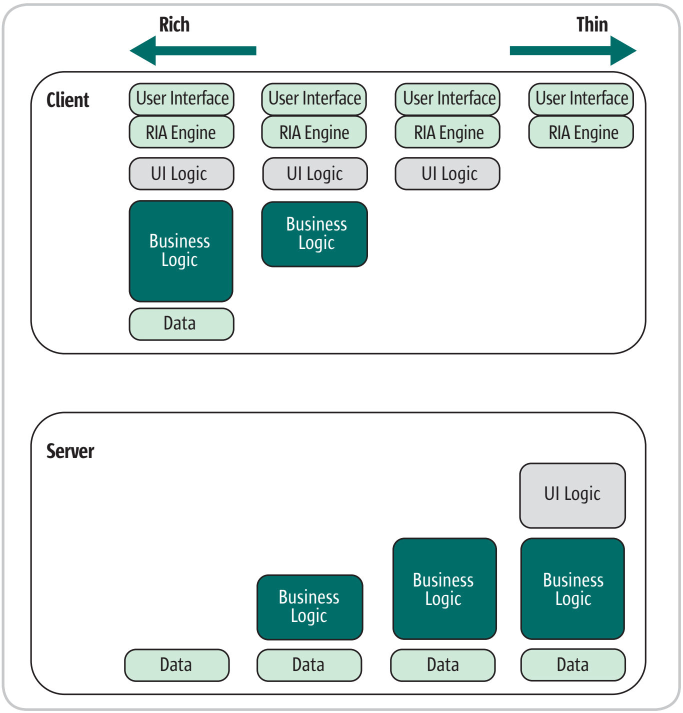


Figure 3 RIA enables flexibility in application logic partitioning

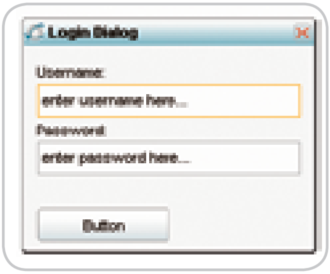


Figure 4 Login Dialog

Declarative UI Development

The general RIA programming model is centered on using an XML-based UI markup language to create a rich user interface. The XML-based UI markup provides a much higher level of abstraction than HTML for building rich user interfaces. XML UI frees programmers to focus on the application's core logic and explicitly complements the roles of a typical enterprise development team (see reference "XML for Client-side Computing").

Below are examples from a scripting-based approach (Laszlo Systems) as well as an OOP-based approach (Nexaweb). Both are zero-install and can run inside any popular Web browser today without any software download. On the client side, Laszlo requires a Flash engine (Flash 6 and above) while Nexaweb requires a JVM (JDK 1.1 and above).

Laszlo is a Flash-based RIA solution. It uses an XML UI markup language called "lzx" to describe the UI and uses ActionScript to code application logic. The Laszlo server will automatically compile lzx files into the Flash binary format (SWF), deliver the SWF files for rendering inside a Flash engine, and execute the application. Following is an example of the Laszlo code:

```
<canvas height="450">
  <window x="10" y="10" width="300" height="200"
          title="my window"
          resizable="true" closeable="true">
        <button x="10" y="100">Hello, World</button>
  </window>
</canvas>
```

By comparison, Nexaweb is a Java-based RIA product. Developers would use an XML-based UI markup to create a rich user interface and build client-side business logic by writing client-side Java objects called Managed Client Objects that are standard Java program objects. The Nexaweb client runtime dynamically renders the XML UI markup to present a rich user interface, and dynamically downloads client-side Java objects to the client side for execution in a "on-demand" fashion. Here is a simple Nexaweb UI that defines a tree and a button managed by a layout manager:

```
<xal xmlns="http://www.openxal.org/xal">
   <window title="New Window">
   <boxLayout orientation="vertical" pack="start" align="stretch
"/>
       <tree>
         <column/>
           <row expanded="true">
              <cell text="Tree Item 1"/>
              <row>
                   <cell text="Sub Tree Item 1"/>
              </row>
              <row>
                   <cell text="Sub Tree Item 2"/>
               </row>
           </row>
           <row expanded="true">
               <cell text="Tree Item 2"/>
               <row>
                   <cell text="Sub Tree Item 3"/>
               </row>
           </row>
        </tree>
        <button text="OK"/>
   </window>
</xal>
```

As shown in these two code examples, though Nexaweb uses Java and Laszlo uses Flash, RIA UI development is conceptually identical between the two different RIA solutions.

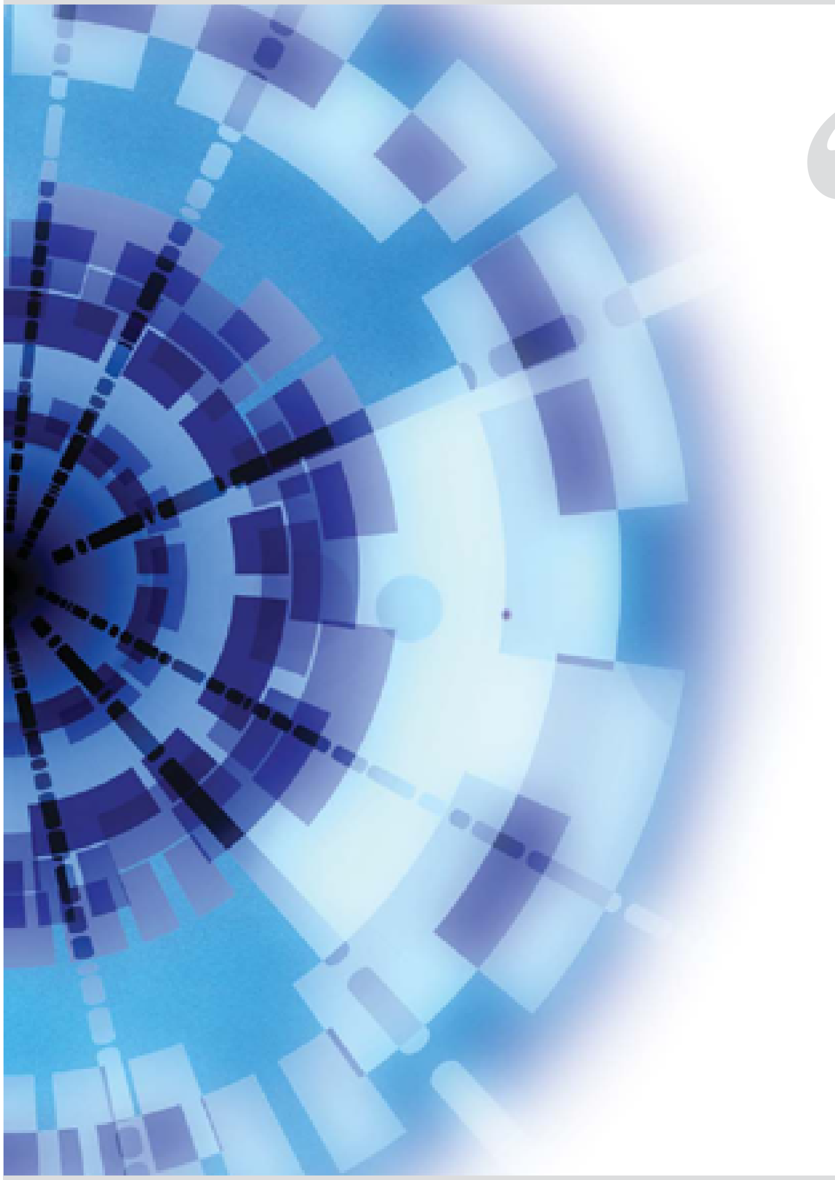
Application Logic Development

RIA solutions enable a wide range of options for application logic development. In typical HTML applications, most of the logic has to be on the server side. In typical desktop applications, most of the logic resides on the desktop. RIAs offer significant flexibility so developers put the logic either on the client side or server side. They can also adjust the location — ranging from very limited logic on the client side all the way to almost 100% logic on the client side, as shown in Figure 3.

The flexibility in partitioning application logic brings significant benefits. Some applications are best suited to having all their logic centralized on the server side while other applications require that the logic run on a local desktop. Traditionally, developers have to make tradeoffs depending on whether they choose to build the application as Web application or a desktop application, and bear with the problems associated with that particular choice.

#1 Rated AJAX and Rich Internet Application toolkit* TIBCO General Interface

Build 100% Pure Browser Rich Internet Apps with TIBCO General Interface™



“TIBCO General Interface is a friendly, capable toolkit for building sophisticated JavaScript Web applications that run in a browser.”

Download today
at **http://www.tibco.com/mk/gi**

TIBCO®
The Power of Now®

©Copyright 2006, TIBCO Software Inc. All rights reserved. *InforWorld 2006 Technology of the year award

RIAs combine the best of both worlds, enabling developers to meet different application requirements without making costly tradeoffs.

In contrast to UI development, the choice of a particular RIA approach — Java, AJAX, .NET, or Flash — has a direct impact on and creates significant differences in application logic development. Choosing a scripting-based solution requires that the logic be written as scripts, which limits the amount and scope of logic that can be developed and maintained cost-effectively. An OOP-based RIA solution gives maximum flexibility to logic development and maintenance, but requires a higher-level skill set than scripting.

There's a code sample of logic development using Laszlo in Listing 1.

By contrast, OOP-based approaches use an object-oriented true programming language for the application logic development and typically enforce separation between the logic and the UI markup. For example, Nexaweb uses standard Java for the application logic called mco. Nexaweb also enforces a clear separation between the UI and logic, preventing the mixing of the UI with the logic in the same document for better application maintenance. Nexaweb separates the UI from the application logic:

```
<xal>
  <mco:declarations xmlns:mco="http://nexaweb.com/mco">
    <mco:mco id="myMco" src="com.nexaweb.test.MyTestMco"/>
</mco:declarations>

  <dialog title="Login Dialog">
    <boxLayout orientation="vertical" pack="start" align="start"/>
    <label text="Username:"/>
    <textField height="25" text="enter username here..."
width="200"/>
    <label text="Password:" />
    <textField height="25" text="enter password here..."
width="200"/>
    <panel height="25" width="100"/>
    <button height="25" text="Button" width="100" onCommand="mco://
myMco.handleOnCommand()"/>
  </dialog>
</xal>
```

Nexaweb enables application logic to be written using standard Java:

```
/**
 *
 */
package com.nexaweb.test;

import com.nexaweb.client.ClientEvent;
import com.nexaweb.client.ClientSession;
import com.nexaweb.client.mco.AbstractMco;
import com.nexaweb.client.mco.McoContainer;


/**
 * @author cwei
 *
 */
public class MyTestMco extends AbstractMco {

 public void handleOnCommand() {
  ClientSession clientSession = McoContainer
     .getClientSessionFromMco(this);
  ClientEvent clientEvent = clientSession.getEventHandler()
     .getClientEvent();

  //additional business logic here…

  System.out.println("Hello, you clicked the button!");
 }

}
```

Choosing the Right RIA Solution

Given the various RIA approaches and solutions available, selecting an RIA solution can be confusing. There's no universal "right" RIA solution. It depends on the application's requirements.

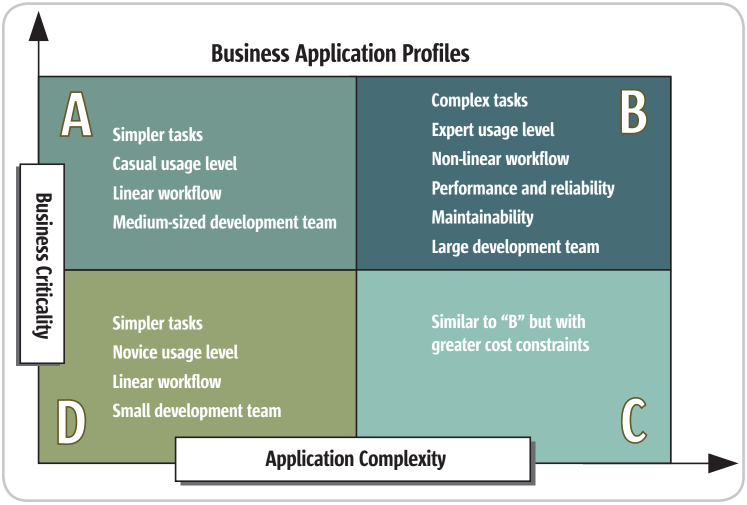


Figure 5 Enterprise application profiles

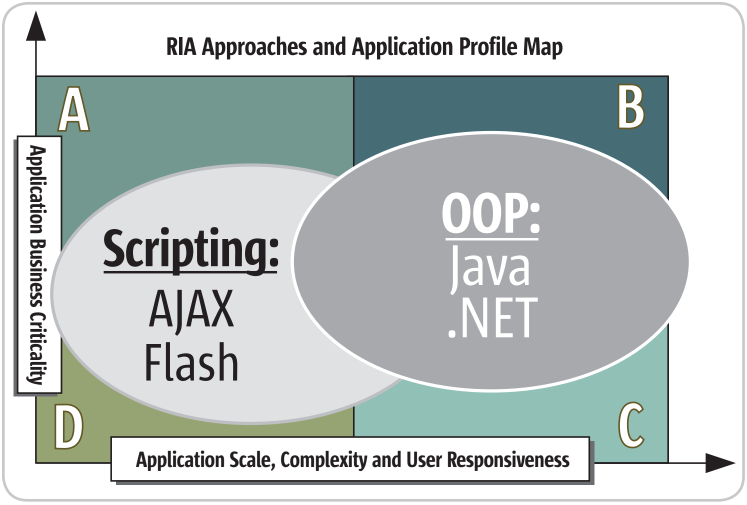
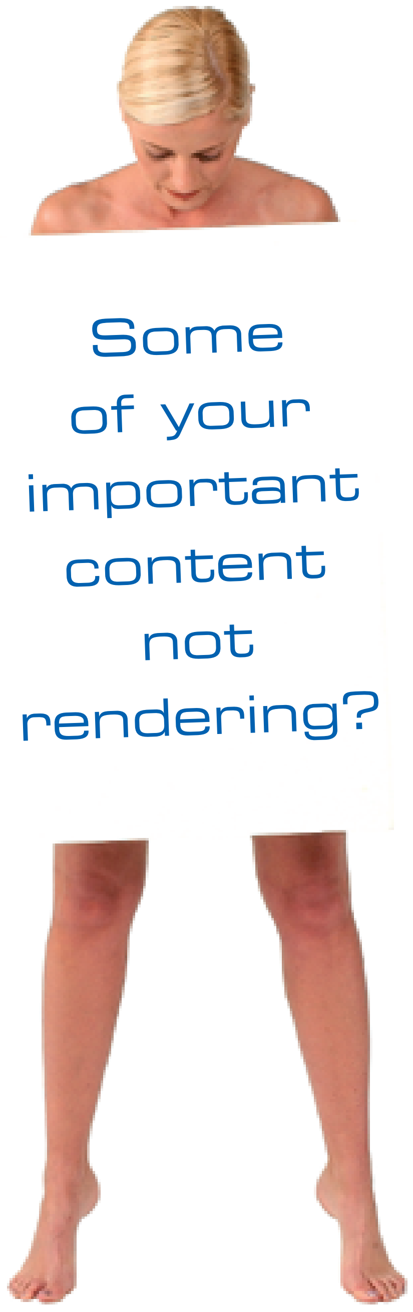

RIA Approaches and Application Profile Map
A
B
C
D
Scripting:
AJAX
Flash
OOP:
Java
.NET
Application Business Criticality
Application Scale, Complexity and User Responsiveness

Figure 6 Mapping RIA approaches to enterprise application types



get WebRenderer™

Java Browser SDK

Desktop Edition

Embed WebRenderer into your Java client side application interface to display rich web content.

- ✓ Swing and AWT support
- ✓ Extensive DOM
- ✓ Plugin support
- ✓ Powerful API
- ✓ HTTP, SSL, TLS
- ✓ JavaScript
- ✓ Complete Event model

Server Edition

Automate Server Side web content reporting, printing and processing with WebRenderer Server Edition.

- ✓ Headless server optimized
- ✓ J2EE support
- ✓ No AWT or Swing dependencies
- ✓ Complete DOM support
- ✓ HTML to thumbnail image
- ✓ Print support
- ✓ Standards compliant rendering

To download a free 30 day trial visit www.webrenderer.com

© JadeLiquid Software 2006. JadeLiquid and WebRenderer are trademarks of JadeLiquid Software in the United States and other countries.
Java and Java related marks are trademarks of Sun Microsystems, Inc. in the United States and other countries.

JadeLiquid™ Software

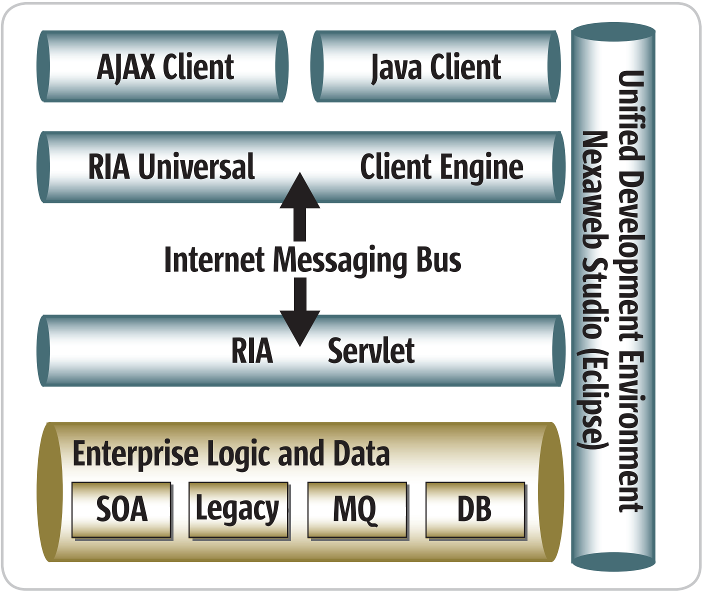


Figure 7 Cross-technology runtime environment for RIAs (Source: Nexaweb)

RIA Approach	Suitable Application Profile	Developer Fit
Java	• Transaction-oriented applications • Responsive user interaction and runtime performance are important • Expert use pattern applications (frequent usage, long duration usage) • Performance, scalability, and reliability can't be sacrificed • Applications that must be maintained for many years	• Java Developers
.NET	• Suitable application profiles are similar to Java	• .NET Developers
AJAX	• HTML-centric or Web content-oriented applications • Casual use pattern • Fast application loading and startup are important • Limited client-side logic (lower maintenance requirement)	• JavaScript developers (CSS, DHTML, JavaScript, cross-browser skills)
Flash	• Casual use pattern • Limited client side logic (lower maintenance requirement) • Rich media-oriented applications	• Flash/Flex developers

Table 3 Mapping RIA approaches to enterprise requirements

Enterprise Application Requirements

For the purposes of this discussion, it's useful to categorize the full spectrum of software applications that enterprise IT departments build, deploy, and maintain across two related dimensions: business criticality and application complexity.

- ***Business criticality*** concerns the degree to which an application is critical to running the business or meeting business objectives. Disrupting access to a business-critical application, or even unacceptable performance, has an immediate and significantly negative impact on the business. Other applications are less critical to operations; if there's a problem, the user can wait a few minutes to perform a task without major consequences.
- ***Application complexity*** refers to its feature richness and sophistication from a user's perspective. Some enterprise applications have thousands of screens, with usage metaphors characterized by multi-path, non-linear state transitions. (In other words, you might rarely use them exactly the same way twice.) Other applications have rather linear state transitions and fixed usage paths — using them is comparatively routine.

Classified as either "high" or "low" across both these dimensions, an application falls into one of four categories as Figure 5 illustrates.

The applications in quadrant A are business-critical and less complex. Users rely on these "helper" applications to do simple but highly important business operations (e.g., an employee portal, partner extranet, or e-commerce Web site). These applications are used less frequently and/or for shorter durations ("casual usage level") than more complex applications. The workflow is typically linear; users do the same tasks in roughly the same order each time they interact with the application. From a development perspective, the client-side development team typically comprises fewer developers than a more complex application would require.

A classic example of a high-criticality/low-complexity application is an airline's online ticketing application. Most users interact with it only occasionally, for a short duration, and in a step-by-step fashion.

Applications in quadrant B are both business-critical and complex. These applications are used for many hours each day to do complex non-linear tasks that are central to business operations. The performance, availability, and scalability of these applications are extremely important. From a development perspective, maintenance is important and may cost more than the initial development. The development team comprises many developers who require close collaboration.

Examples of high-criticality/high-complexity applications include the trading applications used by portfolio managers, call center applications and banking applications accessed by tellers.

The applications in quadrant C are complex but less business-critical. As a result, they are managed much more cost-consciously. High-complexity/low-criticality applications include some legacy applications in which companies wish to minimize further investments, as well as some corporate R&D projects.

The applications in quadrant D are less complex and less business-critical. They are typically written by a small development team of one or two people. Developers' individual experimentation would fall into this category.

Different RIA Technology for Different Applications

Seen against the backdrop of business criticality and UI complexity, different RIA technologies are appropriate for implementing or re-architecting the various classes of enterprise applications.

As Figure 6 illustrates, the applications in quadrants B and C are much better suited to OOP-based RIA development approaches like Java and .NET, because these technologies offer better maintainability and support for team development. Scripting-based approaches are more suited for applications that fall into quadrants A and D where programming tasks are simpler, development teams are smaller, and maintainability is a less mission-critical concern.

Table 3 provides details on how different RIA approaches fit with different enterprise requirements for application profiles and developer skill sets.

The diverse nature of enterprise application requirements, combined with the clear strengths and weaknesses of different RIA technologies, lead to the inevitable conclusion that "one size does not fit all."

No single RIA development approach is ideal for all enterprise environments. Some requirements are better met by scripting-based RIA approaches, while others require OOP. And in these two categories, a particular application need will be better served by AJAX versus Flash, or by Java versus .NET. In short, all four of these RIA technologies are likely to co-exist in many enterprise environments for the near future.

Interesting New Developments

All RIA solutions are fundamentally constrained by their underlying technology — AJAX, Flash, Java, or .NET. If a developer picks Flex to develop his RIA, he has to live with the pros as well as cons of Flash. Likewise, if a developer picks an AJAX toolkit to develop his RIA, she must live with the various challenges associated with DHTML and JavaScript. As we mentioned earlier, among the four technologies, each has its strengths and weaknesses. One of the major goals of enterprise IT departments is "common flexibility" — providing standardization and simplification across different business applications and initiatives, while enabling flexibility for innovation within business units. Different business units have different programmer skills and therefore need different types of applications. As a result, dictating the use of one RIA technology across a large organization is unlikely to work well.

There's been a very interesting development in the RIA marketplace recently: cross-technology RIA solutions. Both Laszlo Systems and Nexaweb recently announced that their products are supporting more than one technology so that the same application can be delivered and rendered on different technology platforms. Laszlo supports both Flash and AJAX (DHTML). Nexaweb supports Java and AJAX. With this development, developers don't have to fight the "religious war" of JavaScript versus Java, Java versus .NET, or .NET versus Flash. Such development accommodates not only different developer skill sets, but also opens the door to combining the benefits of scripting-based approaches with those of OOP-based approaches, delivering optimal results.

Figure 7 shows cross-technology RIA solution architecture.

Listing 2 is a sample application written using a cross-technology RIA solution. It is an RSS reader that would read RSS feeds from Yahoo and display all the feeds in a table. The code is Listing 2 and the UI screen display is shown in Figure 8.

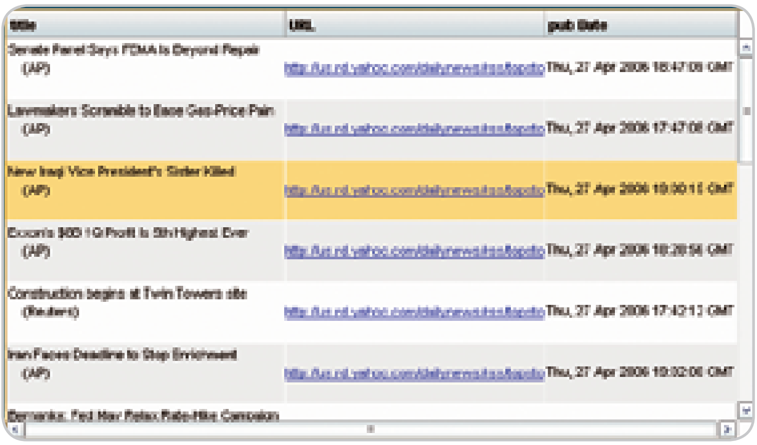
Enterprise RIA Adoption Today

Though still in an evolutionary stage, RIAs have been adopted and proven at many leading organizations over the world. Many companies have adopted RIAs as the foundation for their business applications and achieved great success.

How broadly have RIAs been adopted? Though there are no industry-recognized statistics available, numbers from RIA solution vendors provide some insight. For example, Adobe claims that Flex has about 300 customers. Nexaweb claims that its platform has been deployed to over 4,000 enterprises.

It is also meaningful to look at which industries are adopting RIA. According to a market study done by Nexaweb in October 2005, RIA adoption spans a wide range of industries, with no single one dominating. Financial services leads with a 17% share, followed closely by healthcare, hospitability, and consumer products.

From an application profile perspective, companies adopt RIA solutions for many different kinds of applications, including internal IT applications, B2B applications, B2C applications, and B2C Web sites. According to the same Nexaweb research, 48% of the RIAs deployed today are enterprise business applications, either B2B or internal, while 45% of them are deployed as consumer applications.



title	URL	pub Date
Senate Panel Says FEMA Is Beyond Repair (AP)	http://us.rd.yahoo.com/dailynews/rss/topsto	Thu, 27 Apr 2006 16:47:09 GMT
Lawmakers Scramble to Ease Gas-Price Pain (AP)	http://us.rd.yahoo.com/dailynews/rss/topsto	Thu, 27 Apr 2006 17:47:08 GMT
New Iraqi Vice President's Sister Killed (AP)	http://us.rd.yahoo.com/dailynews/rss/topsto	Thu, 27 Apr 2006 10:30:15 GMT
Exxon's $8B 1Q Profit Is 5th Highest Ever (AP)	http://us.rd.yahoo.com/dailynews/rss/topsto	Thu, 27 Apr 2006 10:20:58 GMT
Construction begins at Twin Towers site (Reuters)	http://us.rd.yahoo.com/dailynews/rss/topsto	Thu, 27 Apr 2006 17:42:13 GMT
Iran Faces Deadline to Stop Enrichment (AP)	http://us.rd.yahoo.com/dailynews/rss/topsto	Thu, 27 Apr 2006 10:32:06 GMT
Bernanke: Fed May Relax Rate-Hike Campaign		

Figure 8 Screen display of Listing 2 code

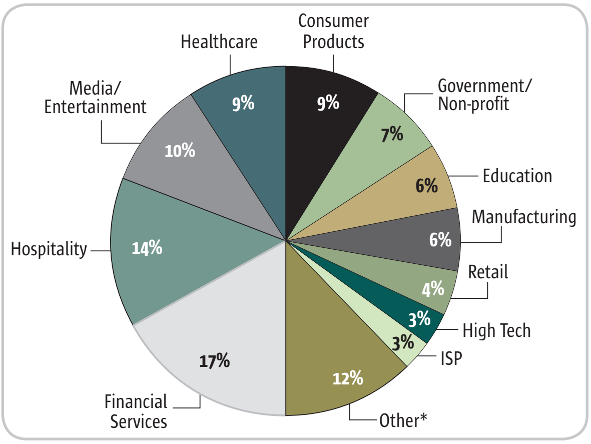


Figure 9 RIA adoption by industries (Source: Nexaweb)

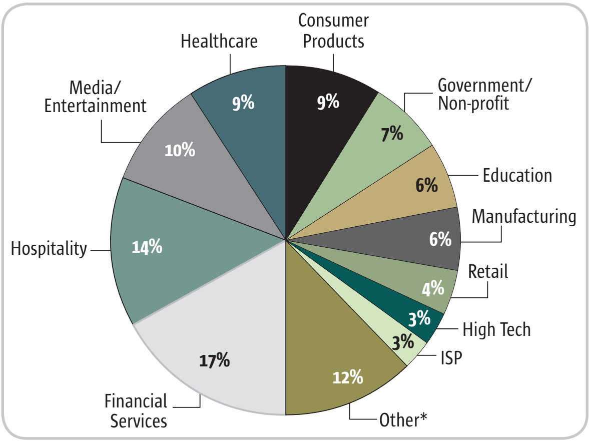


Figure 10 RIA adoption by application types (Source: Nexaweb)

Conclusion

To leverage the Internet for competitive advantage and lower operating costs, businesses need RIA solutions to overcome the inherent limitations of the Web as a platform for developing, deploying, and maintaining business applications.

There are different approaches based on Java, .NET, AJAX, and Flash for RIA solutions, and each approach has its strengths and weaknesses. Given the diverse application requirements in enterprise environments, no single approach will be able to span all enterprise environments. In the end, all four approaches will co-exist serving different application requirements.

Though different RIA solutions may be based on different technology approaches, the programming model centered on a declarative UI is common. The real differentiator is application logic development, which is determined by the RIA approach used by the chosen RIA solution. In the end, the application logic development determines application maintenance and scalability.

Cross-technology RIA solutions are exciting new developments. Such solutions should enable enterprises to adopt a common model and framework to meet different application requirements, while still enabling innovation and accommodating different developer skill sets.

Though relatively young, enterprise RIA solutions have already been adopted by many companies in many different industries led by the financial services. As RIA solutions are further developed, RIA adoption in enterprise environments will continue to grow.

Resources

- M. Driver, R. Valdes, and G. Phifer. "Rich Internet Applications Are the Next Evolution of the Web." Gartner Research Note. G00126924. May 4, 2005.
- Coach Wei. "XML for Client-side Computing." XML Journal. (http://www.sys-con.com/story/?storyid=44013&DE=1). March 10, 2004.
- Dion Hinchcliffe "When the worlds of SOA and Web 2.0 collide."ZDNet. http://blogs.zdnet.com/Hinchcliffe/?p=35). April 2006.
- Dana Gardner. "Nexaweb draws a brilliant bead between SOA and AJAX values." ZDNet, http://blogs.zdnet.com/Gardner/index.php?p=2269. March 2006.
- Laszlo Systems: http://www.lazslosystems.com
- Adobe Flex: http://www.macromedia.com/software/flex/
- Laszlo Systems: http://www.openlaszlo.org/lps-latest/docs/reference/script.html
- Nexaweb Platform: http://www.nexaweb.com
- Nexaweb jRex: http://www.nexaweb.com
- Dojo toolkit: http://www.dojotoolkit.org
- Rico Ajax toolkit: http://openrico.org/
- DWR: http://getahead.ltd.uk/dwr/
- Thinlet: http://www.thinlet.com
- Apache Kabuki: http://kabuki.apache.org
- Nexaweb aRex: http://www.nexaweb.com/products.aspx?id=326
- Microsoft, XAML, and Windows Vista: http://msdn.microsoft.com/windowsvista/about/#wpfx
- Backbase: http://www.backbase.com
- JackBe: http://www.jackbe.com
- Isomorphic: http://www.isomorphic.com
- Bindows: http://www.bindows.net
- Laszlo Systems. "Laszlo Systems Announces Plans To Extend OpenLaszlo Platform to Support Delivery of Web 2.0 Applications in Browsers Without Flash." http://www.laszlosystems.com/company/press/press_releases/pr_mar_06.php. March 2006.
- Nexaweb. "Nexaweb To Introduce Ajax Developer Edition." http://www.nexaweb.com/news.aspx?id=329. March 2006.

Listing 1

```
<canvas debug="true" height="200" width="400">
  <script>
  <![CDATA[
    // Add a find method to Array
    Array.prototype.find = function (what ) {
      for (i in this ) {
        if (this[i] === what) {
          return i;
        }
      }
    }

    sneaky = {example: 'sneaky'};
    tryit = ['foo', 42, sneaky, Math.PI, false];

    Debug.write("42 is at: " + tryit.find(42));
    Debug.write("false is at: " + tryit.find(false));
    Debug.write("'bar' is at: " + tryit.find('bar'));
    Debug.write("{example: 'sneaky'} is at: " + tryit.
find({example: 'sneaky'}));
    Debug.write("sneaky is at: " + tryit.find(sneaky));
  ]]>
  </script>
</canvas>
```

Source: Laszlo Systems

Listing 2 A simple rss feed application

```
<xal xmlns="http://www.openxal.org/xal">
    <data:documentDataSource id="yahoo"
        source="http://rss.news.yahoo.com/rss/topstories"
        xmlns:data="http://www.openxal.org/data" />
    <rootPane>
            <borderLayout/>
            <label img="yahoo_news.gif" borderPosition="north"/>
            <table borderPosition="center">
                <column>
                    <header text="Title"/>
                </column>
                <column>
                    <header text="URL"/>
                </column>
                <column>
                    <header text="pub Date"/>
                </column>
                <data:iterator xmlns:data="http://www.openxal.
org/data"
dataSource="yahoo" type="ONE_WAY" name="newsIterator" select="//
item">
                    <row>
                       <textView>{*('title')}</textView>
                       <link text="{*('link')}"  alignHorizonta
l="left"/>
                       <cell text="{*('pubDate')}"/>
                    </row>
                </data:iterator>
            </table>
    </rootPane>
</xal>
```

Source: Nexaweb

IT'S THE END OF THE LINE FOR SPAM.

PUT IT IN THE CAN.

Ads, fraud schemes, porn, viruses, phishing attempts…
if it's offending e-mail, Roaring Penguin Software's CanIt™
solutions will stop it before it reaches your mail server.

CanIt
CONTENTS: JUNK E-MAIL

- Simple administration
- Automatic spam and virus updates
- Compatible with all e-mail servers
- Available as software or hardware appliance
- Flexible end-user controls

**Throw the book at spam today!
Request a free 20-day CanIt evaluation
and stop spam on YOUR terms!**

ROARING PENGUIN
SOFTWARE INC.

www.roaringpenguin.com +1 (613) 231-6599 sales@roaringpenguin.com

Deep Diving with ADF Faces

The benefits of the AJAX RenderKit

by John Fallows and Jonas Jacobi

In an effort to provide developers with a productive environment, Oracle has been working on a very rich UI component framework for several years. This framework – ADF Faces – has now been donated to the open source community. More precisely, it has been donated to the Apache Software Foundation and is currently hosted in the Apache Incubator – http://incubator.apache.org/projects/adffaces.html. Craig McClanahan is mentoring the project during the Apache incubation. The Apache MyFaces community is also involved in the project to assist with graduation from the incubator, into the Apache MyFaces "ADF Faces" subproject.

What is ADF Faces? ADF Faces is a comprehensive, free JSF component library. This library contains 12 helper objects such as converters and validators, and 93 components ranging from simple input components to complete page components with built-in menu model support. In addition, ADF Faces provides a set of extended services such as a dialog framework and a skinning framework.

In a series of articles we will be deep diving into various aspects of this donation and give developers insight into its internal functionality and how to use and extend ADF Faces. In this first article we discuss the HTML AJAX RenderKit provided by this JSF component library.

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

john.fallows@oracle.com

The ADF Faces HTML AJAX RenderKit

As most of you probably know by now, AJAX is a new phenomenon, but the technologies that make up AJAX are not. AJAX, for those who don't know, is a technique that uses native browser technologies to provide highly interactive user interfaces and to asynchronously communicate with server-side logic.

Since the technologies backing AJAX have been around for quite some time, there have been AJAX solutions available long before the term was coined. One of those is the ADF Faces "AJAX" RenderKit. This solution has been called partial page rendering (PPR) and has been used by Oracle and its customers for several years. Instead of re-rendering the entire page on a postback, PPR will only update a portion of the page.

With AJAX, or PPR, application developers can then provide a rich and interactive, desktop-like user interface on the Web.

Practical Benefits of ADF Faces AJAX RenderKit

To fully appreciate the benefits that the ADF Faces AJAX RenderKit provides, we first need to look at some of the common problems end users are facing with a traditional Web application.

Problems with Full Page Refresh

End users normally spend time waiting for a server response without being able to do anything else, and, if they do, it might actually cause problems, like double-submit. The scroll position and focus in the page are usually lost even when the same page is redisplayed after a user interaction such as expanding a tree node. Validation is usually delayed until the user explicitly presses submit, and then he or she has to wait for the response and scroll back to the input field with the error message to fix the problem.

With an AJAX JSF solution, all of this can be addressed with little or no effort from the application developer and a great outcome for the end user. The ADF Faces AJAX RenderKit addresses these issues by asynchronously updating only portions of the page, and by keeping track of the form post (basically controlling whether a form post has occurred). The partial update of the page keeps the scrollbar position and the focus, and provides instant validation of user input.

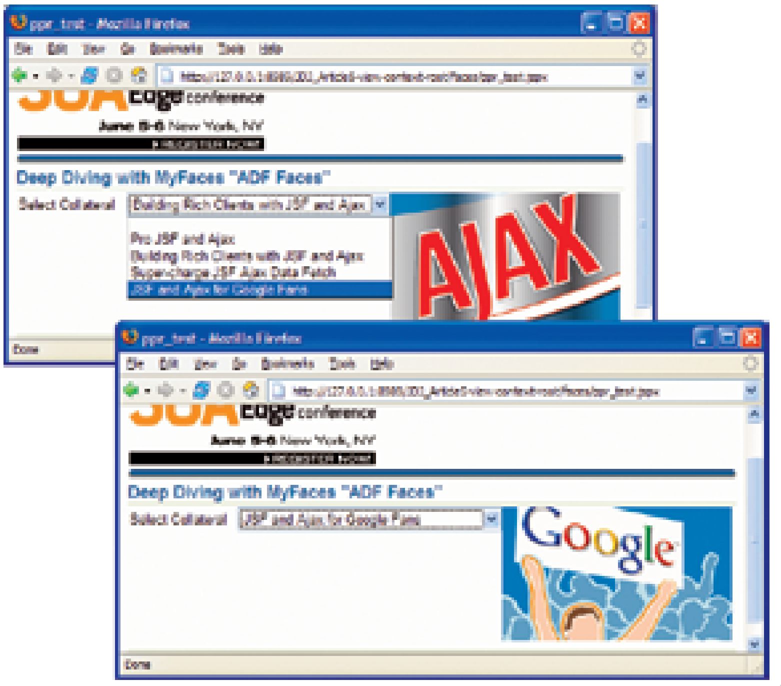

ppr_test - Mozilla Firefox
Deep Diving with MyFaces "ADF Faces"
Select Collateral
Building Rich Clients with JSF and Ajax
Pro JSF and Ajax
Building Rich Clients with JSF and Ajax
Supercharge JSF Ajax Data Fetch
JSF and Ajax for Google Fans
AJAX
Google
Done

Figure 1 Showing partial update and scrollbar position

Lacking visibility in your development process?

Enerjy CQ2 delivers complete visibility for a quality-driven development process

Enerjy CQ2 integrates with your version control system and existing tool set. It runs parallel with existing project build systems to collect, store and graphically display code quality metrics.

Enerjy CQ2 helps to produce the highest quality applications by measuring

- Adherence to coding standards
- Unit test results
- Code coverage metrics
- Developer trends and activity

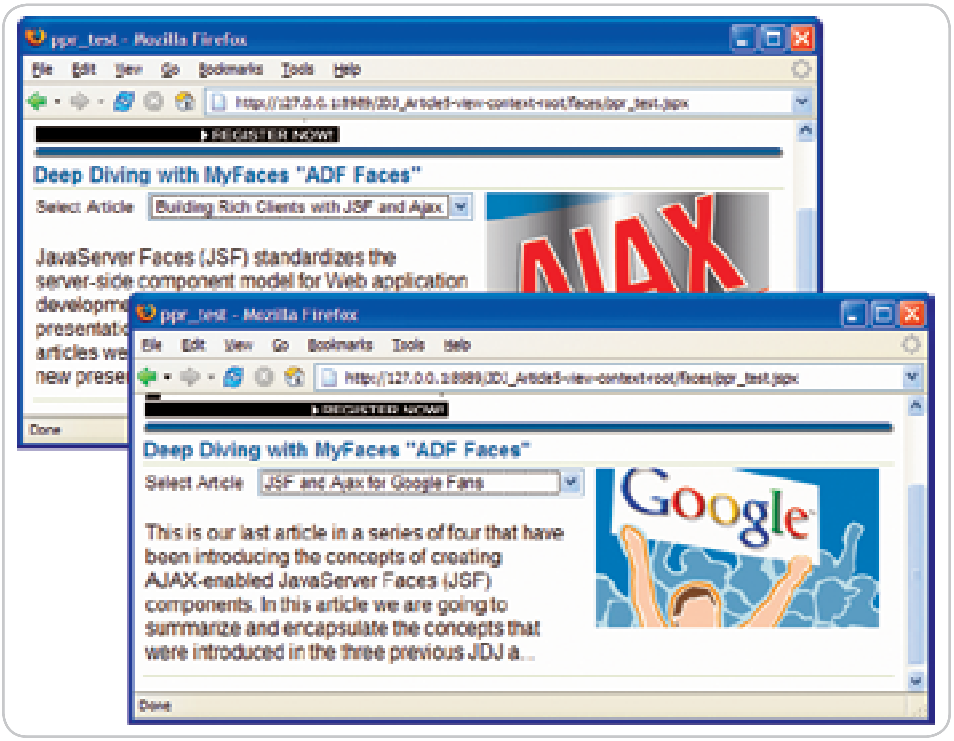
Enerjy CQ2 presents metrics in a centralized Web application

- Provides instant access to data reports and analysis
- Graphically displays metrics and trends on a per project, per team and per developer basis
- Leverages version control system to identify code ownership

ENERJY®

Software Integrity

To learn more about Enerjy CQ2 visit www.enerjy.com/visible


ppr_test - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://127.0.0.1:8989/JDJ_Article5-view-context-root/faces/ppr_test.jspx
REGISTER NOW!
Deep Diving with MyFaces "ADF Faces"
Select Article
Building Rich Clients with JSF and Ajax
AJAX
JSF and Ajax for Google Fans
Google
Done

JavaServer Faces (JSF) standardizes the server-side component model for Web application developme… presentati… articles we… new prese…

This is our last article in a series of four that have been introducing the concepts of creating AJAX-enabled JavaServer Faces (JSF) components. In this article we are going to summarize and encapsulate the concepts that were introduced in the three previous JDJ a...

Figure 2 Update page with description field.

Using the ADF Faces AJAX RenderKit

Creating an application using the HTML AJAX RenderKit provided by the Apache "ADF Faces" Incubator project is easy. There are two ways of achieving a partial update – declaratively and programmatically. Let's first look at the declarative approach.

Figure 1 illustrates the application we are going to build in this article. It contains a dropdown component that lists a set of *JDJ* articles and a graphic component that shows the featured images for each article. When an end user accesses this application and selects an article to read, only the image is updated, not the entire page. As shown in Figure 1 this adds the benefit of a kept scrollbar position when the end user changes from one article to another. Furthermore, focus is still set on the dropdown list, allowing the end user to continue to use the dropdown list without the need to reselect it.

The JSP source for the page displayed in Figure 1 in shown in Listing 1. Using the declarative approach, the application developer needs only to ensure that there is a unique identifier for the "triggering' component – <af:selectOneChoice> – in this example selectArticle, and that the component targeted for partial update includes this identifier in the partialTriggers attribute. That's it!

Programmatically Adding a Partial Target

In response to a partial event, only components registered as partial targets are re-rendered. With ADF Faces, application developers can also dynamically add components to the list of partial targets by calling the addPartialTarget method from the AdfFacesContext as shown in the following code.

```
AdfFacesContext adfContext =
AdfFacesContext.getCurrentInstance();
adfContext.addPartialTarget(target);
```

Let's expand our page with an <af:outputText> component. This component will show an excerpt of each selected article. This time we won't use the declarative approach, instead we'll dynamically add the component to the list of partial targets for the <af:selectOneChoice> component.

In Listing 2, a valueChangeListener has been added to the <af:selectOneChoice>, and a <af:outputText> component has been added underneath <af:selectOneChoice> to provide the excerpt of the selected article. In this case the <af:outputText> component does not include the partialTrigger attribute.

The valueChangeListener in Listing 3 ties it all together by adding the outputText component as the partial target to the selectOneChoice component.

The updated page will now contain a description field that will be part of the partial update when the end user selects an article from the dropdown list.

In Figure 2, note that the scrollbar maintains the same position after the partial postback and that the description has been updated.

The Internal Implementation Design

When a page containing ADF Faces components is accessed for the first time, the initial page render includes a hidden iframe element. This hidden iframe is used to manage the partial update of the page. When a commandButton is clicked, the form is still submitted but targets the hidden iframe instead, and additional information is included in the request to let the AJAX RenderKit detect that a partial response is expected.

During the ApplyRequestValues phase in the Faces Lifecycle, the AJAX RenderKit determines that this is a partial request and decorates the ResponseWriter with a PPRResponseWriter implementation (see Figure 3). The purpose of this is to "clip" the rendered markup to produce only the markup fragments for those components that should be included in the partial response back to the iframe.

Each markup fragment is identified by HTML ID, which is used to find the original section of the HTML DOM that needs to be replaced. Figure 4 shows a DOM inspector and the iframe structure of the partial response for the page created in Listing 2. Note the ID match of each fragment e.g., _id5:selDesc and _id5:selImg.

Challenges Addressed by Choosing <iframe> over XmlHttpRequest

Much of the success of AJAX has come from the widespread availability of XMLHttpRequest (XHR), and many people believe that XHR is a required strategy for AJAX. However, AJAX is simply defined as using native browser technologies (no browser plug-ins)

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

jonas.jacobi@oracle.com

and asynchronous communication with the server. This is also possible to achieve using a hidden iframe, which provides some additional capabilities but with some additional issues as well.

The main functionality that is much more easily supported by the hidden iframe communication channel is file upload. For security reasons, JavaScript executing at the browser is not permitted to read the contents of the local file system. So, when using a plug-in free browser environment, uploading a file from the local file system must leverage the <input type="file" ... > native browser support, which requires the form to be submitted so that the file contents will be sent to the server. Using a hidden iframe allows ADF Faces to target the form submission at the iframe instead of the main document, so that the response from the server is received by the hidden iframe. Of course, the same approach works for any type of form submission, not just file upload input elements.

One of the difficulties introduced by using the hidden iframe communication channel is the problem of knowing when the response is complete. Fortunately, since the PPR response is managed by the AJAX RenderKit, it is straightforward to include an additional script block at the end of the response received by the hidden iframe that will inform the main HTML document when the response is complete. In the rendered response a callback function is added (see Listing 4).

Compatibility with Non-ADF Faces Components

How generic is this functionality? Does it only work with ADF Faces components? What happens when you try to use other components, such as the Apache MyFaces Tomahawk or Tobago component libraries?

Fortunately, the implementation of PPR makes no extra assumptions about the child components of any partial target. If an ADF Faces component is added as a partial target, all of its child components are also re-rendered in the partial response, no matter what component library is used.

Conveniently, any JavaServer Faces component can be added as a partial target using the programmatic API. The PPRResponseWriter will detect when this component is being rendered and ensure that the markup is included in the partial response.

Finally, adding declarative support for any JavaServer Faces component, by adding a partialTriggers attribute, requires only minor modifications to the component API and JSP Tag to expose the attribute and a call to AdfFacesContext.addPartialTriggerListeners in the Renderer.

Conclusion

This is the first successful implementation of AJAX in JSF and, as discussed in this article, is currently used by the ADF Faces component library. This type of architecture relies on a regular form submit. The response is in fragments that contain the information about what is needed for the change. The PPR handler will then figure out where to slot in these changes. This approach is straightforward to use, allowing the application developer to control what components are involved in each partial update. In this architecture, the unit of update is a UIComponent subtree, so the markup for each UIComponent subtree is replaced for each partial target. PPR is also relying on iframes, not XMLHttpRequest, to provide asynchronous communication, which has the benefit of supporting older versions of browsers.

What is interesting is that Oracle announced intentions to donate a new generation of ADF Faces components to the open source community at JavaOne 2006. In contrast to the components discussed in this article, these new components leverage the XMLHttpRequest object to provide AJAX JSF components.

Finally, we believe that MyFaces "ADF Faces" will become the most dominant open source JavaServer Faces component library on the planet. The question is: How will you contribute?

–Listings on page 30

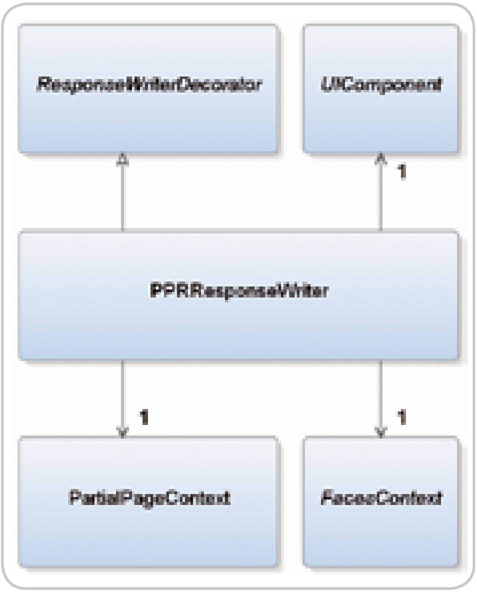
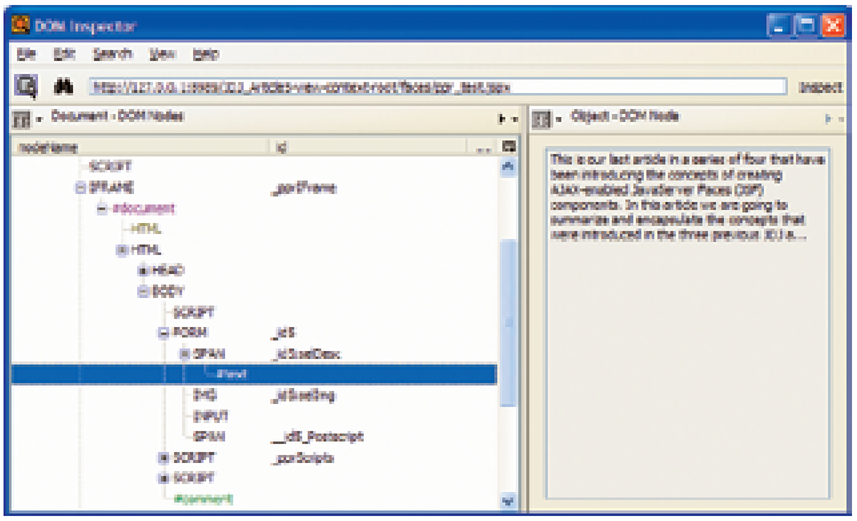


Figure 3 Class diagram over the PPRResponseWriter implementation



```
SCRIPT
IFRAME                 _pprIFrame
  #document
    HTML
    HTML
      HEAD
      BODY
        SCRIPT
        FORM           _id5
          SPAN         _id5::Desc
            #text
          IMG          _id5::Img
          INPUT
          SPAN         __id5_Postscript
        SCRIPT         _pprScripts
        SCRIPT
        #comment
```

This is our last article in a series of four that have been introducing the concepts of creating AJAX-enabled JavaServer Faces (JSF) components. In this article we are going to summarize and encapsulate the concepts that were introduced in the three previous JDJ a...

Figure 4 Partial Response shown in a DOM inspector

Listing 1: Page showing partial page update

```
<?xml version="1.0" encoding="windows-1252"?>
<jsp:root xmlns:jsp="...">
  ...
  <f:view>
    ...
    <h:form>
      <af:panelPage
           title="Deep Diving with MyFaces &quot;ADF Faces&quot;">
        <f:facet name="branding">
          <af:objectImage source="#{magazineBrand.image}"/>
        </f:facet>
        <af:panelBorder>
          <f:facet name="start">
            <af:selectOneChoice id="selectArticle"
                                label="Select Article"
                                value="#{selectMagazine.title}"
                                autoSubmit="true">
              <af:forEach items="#{magazines}" var="row">
                <af:selectItem label="#{row.title}"
                               value="#{row}"/>
              </af:forEach>
            </af:selectOneChoice>
          </f:facet>
          <af:objectImage partialTriggers="selectArticle"
                          source="#{selectMagazine.name}"/>
        </af:panelBorder>
      </af:panelPage>
    </h:form>
    ...
  </f:view>
</jsp:root>
```

Listing 2: The updated page with a description field

```
<?xml version='1.0' encoding='windows-1252'?>
<jsp:root xmlns:jsp="...">
  ...
  <f:view>
    ...
    <h:form>
      <af:panelPage
          title="Deep Diving with MyFaces &quot;ADF Faces&quot;">
        <f:facet name="branding">
          <af:objectImage source="#{magazineBrand.image}"/>
        </f:facet>
        <af:panelBorder>
          <f:facet name="start">
            <h:panelGroup>
              <af:selectOneChoice
                id="selectMagazine"
                label="Select Article"
                value="#{selectMagazine.magazine}"
                autoSubmit="true"
                valueChangeListener="#{selectMagazine.valueChanged}">
                <af:forEach items="#{magazines}" var="row">
                  <af:selectItem label="#{row.title}"
                                 value="#{row}"/>
                </af:forEach>
              </af:selectOneChoice>
              <af:objectSpacer height="20"/>
              <af:outputText
                         id="selDesc"
                         value="#{selectMagazine.desc}"
                         binding="#{selectMagazine.descUpdate}"/>
            </h:panelGroup>
          </f:facet>
          <af:objectImage id="selImg"
                          partialTriggers="selectArticle"
                          source="#{selectMagazine.name}"/>
        </af:panelBorder>
      </af:panelPage>
    </h:form>
    ...
  </f:view>
</jsp:root>
```

Listing 3: The selectMagazine managed bean

```
package com.jdj.sample;

import ...

public class SelectMagazine {

...

    public void setDescUpdate(UIComponent descUpdate) {
        this._descUpdate = descUpdate;
    }

    public UIComponent getDescUpdate() {
        return _descUpdate;
    }

    public void valueChanged(ValueChangeEvent vce)
    {
      UIComponent component = vce.getComponent();
      String rendererType = component.getRendererType();

      if (rendererType.equals("oracle.adf.Choice"))
      {
        _addTarget(_descUpdate);
      }
    }

    private void _addTarget(UIComponent target)
    {
      AdfFacesContext adfContext =
                      AdfFacesContext.getCurrentInstance();
      adfContext.addPartialTarget(target);
    }

    ...
    private UIComponent _descUpdate;

}
```

Listing 4: The PPR callback function

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
SYSTEM "http://www.w3.org/TR/html4/loose.dtd" >
<html dir="ltr" lang="en-US">
<head>
  ...
  <meta http-equiv="Content-Type"
        content="text/html; charset=windows-1252"/>
</head>
  <body onunload="_partialUnload()">
  ...
  <script>
    var _pprLibraries=['/JDJ_Article5-view-context-
                        root/adf/jsLibs/ScriptEval10_1_3_0_4.js'];
    var _pprTargets=['_id5:selImg',
                     '__id5_Postscript',
                     '_id5:selDesc'];
    _partialChange()
  </script>
</html>
```

EMPOWERING

DEVELOPERS WORLDWIDE

NetBeans

NetBeans Platform:
Your Business Solution

☑ Open Standards, 100% Java

☑ Cross-Platform, Modular, Scalable, Extensible

☑ Application Prototyping Utilizing NetBeans GUI Builder Matisse

☑ Free & Open Source Application Development Framework

Find out more at
www.NetBeans.org

Comparing Tomcat Performance Across Platforms

by Frank C. Ziglar

Part 2: Performance and distinct error handling under computational load

We measured performance information to see the differences between Windows and Linux platforms. The test done against the Windows server gave users marginally shorter wait-times by electing to turn some traffic away. The Linux server, however, scaled to serving a greater number of users with reasonable responsiveness before its maximum capacity was reached.

In Part 1 of this article, we examined the relative performance differences, noting the behavioral differences as the servers approached capacity. This limitation was easily remedied by increasing the memory available to Tomcat. Here we'll present the behavior of the servers after this change was made to their configuration.

Our guidelines and testing procedures remained the same, except where noted.

Testing Procedure

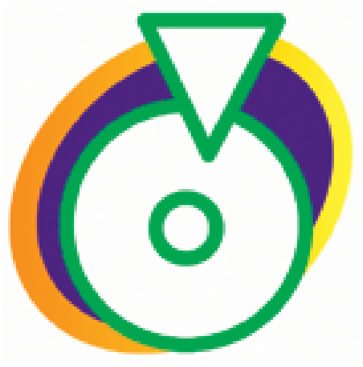
The testing tool managed the test case recording, virtual user simulation, and data gathering. The testing tool used was Web Performance Trainer 2.8, Build 629.

Servlet Container Preparation

Tomcat was installed following the procedure described in the first part of this article. The only difference was to increase the JVM memory limits. Since each server was equipped with 512MB of RAM, the new memory configuration setting was arbitrarily 200MB initially, with a maximum of up to 384MB. In the Windows server, the settings were entered directly into the configuration manager applet installed with Tomcat. In the Linux server, the same effect was achieved by specifying the JVM parameters in the CATALINA_OPTS variable.

Frank C. Ziglar works at Web Performance, Inc.

fziglar@Webperformance.com



Results

Throughput

Our first measurement of interest was roughly how much throughput the server could sustain under an increasing load. This measurement was taken from the total megabytes of HTTP traffic and didn't consider lower-level implementation overhead.

The test was terminated after 100 minutes after a plateau made it evident that the server couldn't improve its responsiveness to further users. Since our load was distributed from multiple engines connected to a 1GbE switch, the throughput of both servers was well below the total Ethernet bandwidth of our test lab and can be ruled out as a significant limiting factor in these results.

The leveling of the throughput indicates that the server had reached its absolute capacity to serve data and couldn't scale any further. The maximum throughput measured was approximately 8.8MB/s on the Windows server compared to 12MB/s on the Linux server. It was evident that the Linux server could scale to meet the needs of many more users than the Windows server (see Figure 1).

CPU Utilization

CPU Utilization gives some insight on how well the server could cope with the increasing load. Each server in the test was equipped with a single physical CPU with the HT Technology enabled. This graph is then an aver-

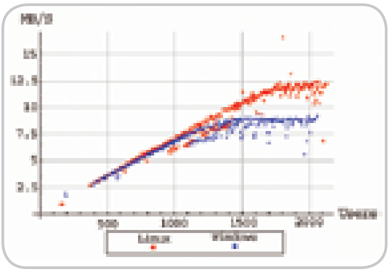


Figure 1 Throughput

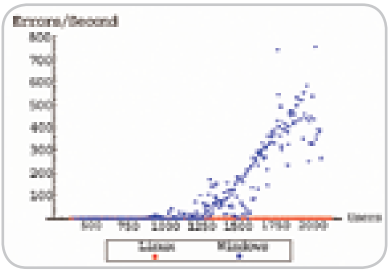


Figure 3 Errors – hits per second

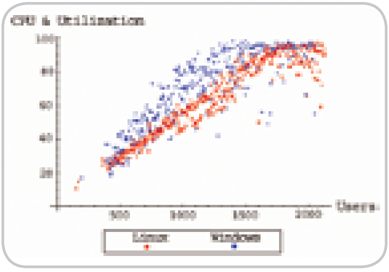


Figure 2 CPU

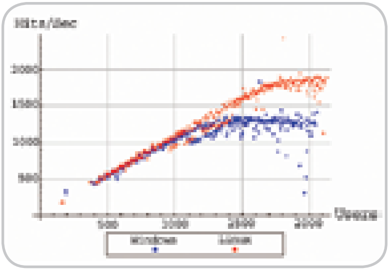


Figure 4 Http – hits per second

age of the processor load of the two virtual CPUs (see Figure 2).

We see here that the Windows server started out consuming more CPU time earlier in the test. As more users are added, performance eventually becomes processor-bound and the server was too computationally overwhelmed to continue to accept users. The Linux server too was fated to produce the results, but lasted longer before the CPU time was completely exhausted.

Errors per Second

During the tests the only errors that propagated their way to the virtual user were transmission errors that occurred when a user would try to open a connection to the server. To the end user this meant that his web browser would display an error message usually something similar to "Connection refused by server." Every page returned during the test was a complete page, not an error page from the server.

We noticed that the Windows server tended to refuse connections instead of trying to service them. The Linux server wasn't completely without error, but rarely exceeded a consistent rate of over one error a second, enough to account for over 180 errors logged after our 100-minute test. However, this number is still negligible considering the Windows server was turning away more traffic per second than the Linux server turned away during the entire test (see Figure 3).

Hits per Second

The next step was to examine how responsive the servers were under an increasing load. We measured the number of responses per second of HTTP traffic. We anticipated that under an increasing amount of traffic, we would see an increasing number of completed responses from the server.

We produced this graph by examining the servers' throughput (see Figure 4).

One can see that once again the two servers maintained a relatively even and competitive pace with each other until the Windows server began to slow down under an increasing load at approximately 1,200 users and leveled out at a capacity of around 1,300 users. The Linux server also leveled out as the user count continued to rise but only after it scaled to nearly 1,800 users. The increase in user capacity let the Linux server serve at slightly over 1,800 hits per second compared to the 1,400 hits per second reached by the Windows server.

Duration

In the last phase of this category, we concerned ourselves with how long a user has to set aside to complete his task. We measured the full time until we got a response from the server, waiting until it arrived before moving onto the next page of the test.

For the duration times of a business case, the baseline for each graph is defined as the amount of time the user spends "interacting" with the page before moving on to the next page. The simulated time the user spent on the last page was omitted (see Figure 5).

The duration graphs show the full range of measured durations during a given interval (dark background), with the averages on top (highlighted points).

Complete Scenario Durations

We started by examining the amount of time the user spent working his way through the entire test case up to the response of the last transaction from the server. Note that each column shows a series of graphs occurring roughly concurrently. Hence, the number of users for each graph is measured for the number of users in a specific case and not the total hitting the entire server (refer to User Distribution in Part 1 of this article if necessary)(see Figure 6).

In each figure, we trimmed off the extreme outlying maximum durations. These outliers were measured as high as 10 minutes in the long scenario, seven minutes in the medium scenario, and 100 seconds in the short scenario.

However, a general trend seemed to emerge: the longer scenarios with more pages are more likely to get stuck on a resource that didn't get a prompt response and will see a greater increase in average duration. Overall, the Windows server maintained a somewhat even average duration under load, though this number appears to jitter more under increasing load. The Linux server showed a gradually increasing average duration under load once the server's capacity was reached. Both servers showed that the potential maximum duration grows with the amount of load on the server.

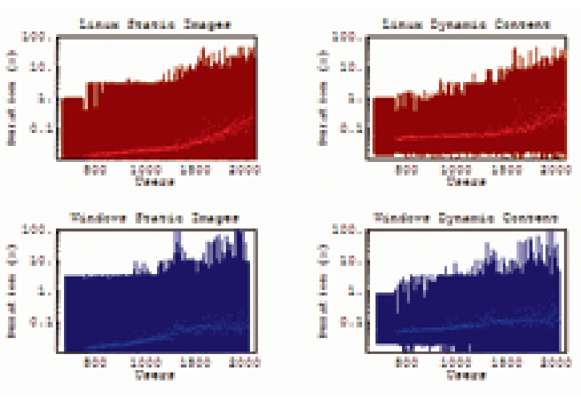


Figure 5 Duration URLs

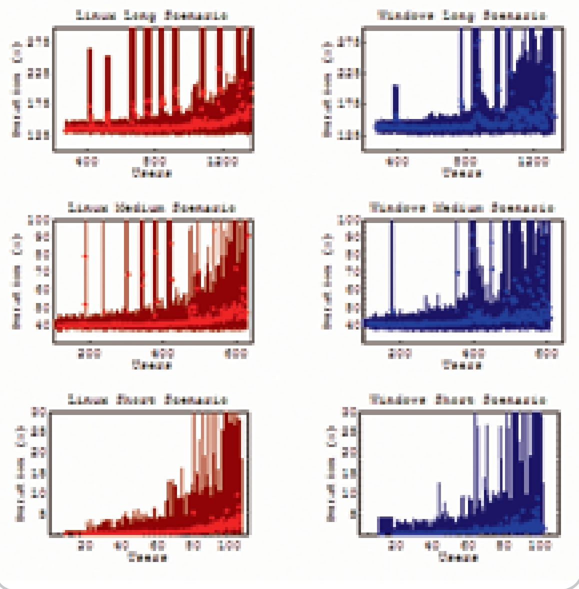


Figure 6 Scenario Duration

"The Linux server served more users **than its Windows counterpart**"

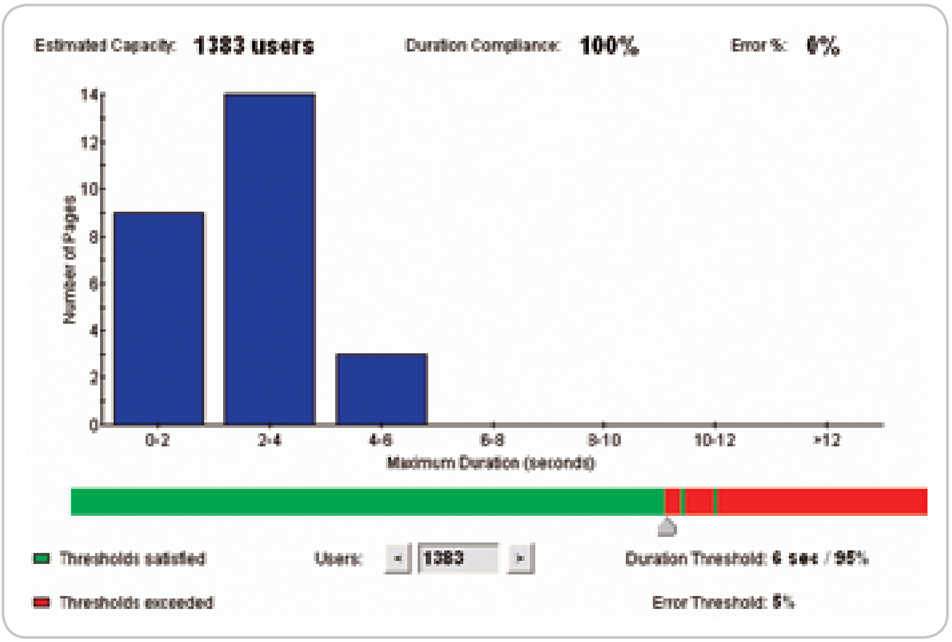


Figure 7 Linux capacity

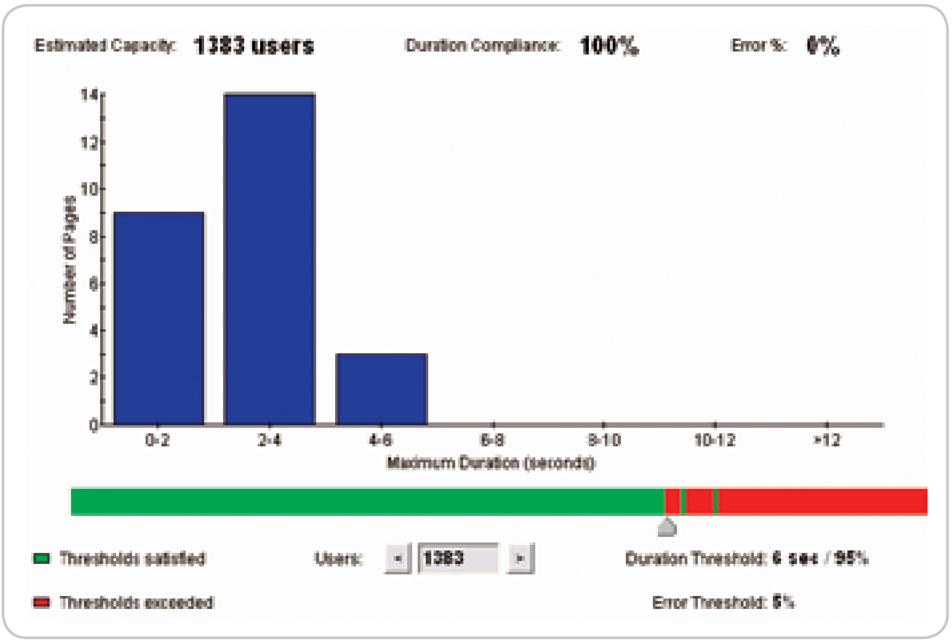


Figure 8 Windows capacity

Static & Dynamic Content

Evidently for the entire duration of the test the average request for any given URL used was followed by a practically instantaneous response from the server. Our maximum durations were more pronounced than those for our static URLs during lower user counts, but this was only because there were significantly more static URLs in each test to potentially return with a maximum duration. However, we noticed that the maximum duration for URLs had a greater tendency to increase on the Linux server under high load. Presumably, since our average durations were returned at a quick rate, the perceived long durations for each test was the consequence of users getting "stuck" on a resource that was delayed to the maximum duration.

Capacity

When we discussed error counts, we mentioned that the only errors were purely transmission errors with the server. However, with the delays we saw, we needed to impose some restrictions so the server would respond in a reasonable amount of time. With this in mind, we used the Web Performance Trainer to see the server's total capacity and focus on how much load could be handled when at least 95% of the users had a total page duration of six seconds or less, with no more than 5% of the responses indicating errors. This value appeared somewhat smaller than we might expect from looking at the scalability of the server. Despite an increasing throughput from the server under higher load, an increasing percentage of users were likely to hit significant delays or connection failures as the load increased. Hence, we can use this analysis to evaluate the realistic capacity of the servers, given the parameters we have provided above (see Figures 7 and 8).

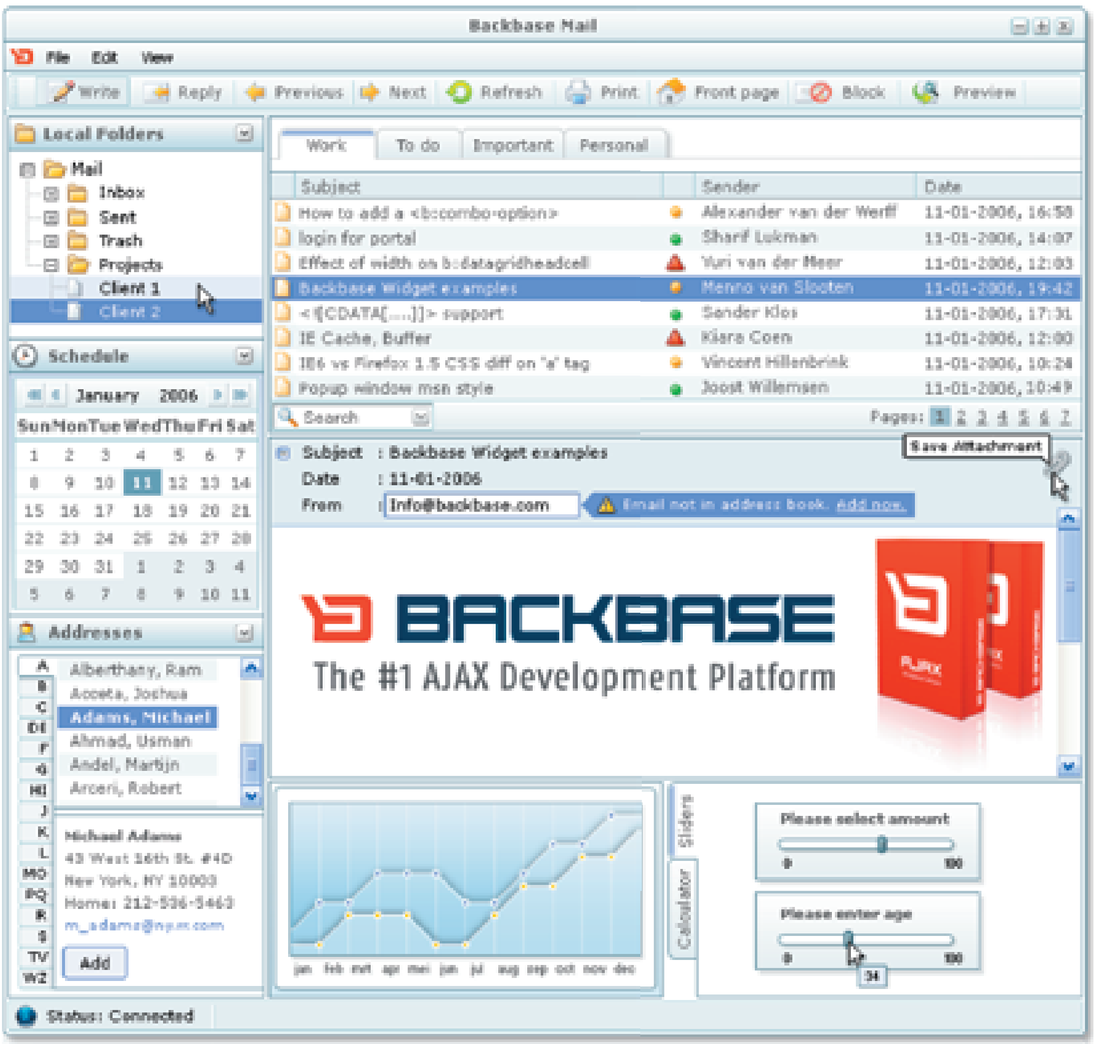
Analysis

We gave Tomcat the opportunity to compete when installed on two different platforms. We saw that the average duration of each URL, even under extensive load, remained relatively instantaneous on both servers. However, the Linux server slowly had an increasing number of URLs that didn't get serviced quickly and delays service for longer periods of time than the Windows server when forced under load. In contrast, the Windows server showed somewhat shorter durations per URL by turning away traffic that it couldn't handle. Despite this quick but incomplete response on some resources, many users found themselves waiting for another resource that wasn't serviced as promptly.

From an administrator's standpoint, the Linux server didn't tie up its CPU resources as early and was able to service many more connections overall, but some of those connections were consistently delayed, leaving users waiting on the other end. However, even given some normal delays, the Linux server was still able to serve more users than its Windows counterpart.

BACKBASE

Create Rich AJAX User Interfaces, Quickly



Subject	Sender	Date
How to add a <b:combo-option>	Alexander van der Werff	11-01-2006, 16:58
login for portal	Sharif Lukman	11-01-2006, 14:07
Effect of width on b:datagridheadcell	Yuri van der Meer	11-01-2006, 12:03
Backbase Widget examples	Menno van Slooten	11-01-2006, 19:42
<![CDATA[....]]> support	Sander Klos	11-01-2006, 17:31
IE Cache, Buffer	Kiara Coen	11-01-2006, 12:00
IE6 vs Firefox 1.5 CSS diff on 'a' tag	Vincent Hillenbrink	11-01-2006, 10:24
Popup window msn style	Joost Willemsen	11-01-2006, 10:49

Pages: 1 2 3 4 5 6 7

Subject : Backbase Widget examples
Date : 11-01-2006
From : Info@backbase.com
Email not in address book. Add now.

BACKBASE
The #1 AJAX Development Platform

January 2006

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

Alberthany, Ram
Acosta, Joshua
Adams, Michael
Ahmad, Usman
Andel, Martijn
Arceri, Robert

Michael Adams
43 West 16th St. #4D
New York, NY 10003
Home: 212-536-5463
m_adams@nym.com
Add

eclipse

Eclipse Plug-in
for Fast Development

JSF

Server-side Extensions
for Java Server Faces

AJAX

Powerful and Scalable
AJAX Components

Download free trial today at www.backbase.com/jsf

i: www.backbase.com | **t:** (866) 800-8996 | **e:** sales-us@backbase.com

© Backbase BV - all rights reserved. BACKBASE is a trademark of Backbase BV.

Best Practices for Securing Your SOA: A Holistic Approach

Testing server-side components

by Mohamad Afshar, Roger Goudarzi, Nickolaos Kavantzas, Barmak Meftah, Ramana Turlapati, and Prakash Yamuna

Service-Oriented Architectures offer a number of potential benefits: They can provide new opportunities to connect enterprises with customers, partners, and suppliers; improve efficiency through greater reuse of services across the enterprise; and offer greater flexibility by breaking down IT silos. But these benefits make security more critical than ever. Why? Services are highly distributed, multi-owner, deployed to heterogeneous platforms, and often accessible across departments and enterprises — and this creates major security issues for developers, architects, and security and operations professionals. Fortunately, there are ways to make your SOA more secure. If you're building applications to SOA using J2EE, BPEL, or XML, you can build security into an SOA by addressing security throughout the entire application lifecycle — not just at deployment time.

We'll examine the types of attacks applications are vulnerable to, and then outline a holistic approach to protecting applications and services that encourages secure coding practices and the use of Web Services security infrastructure. You'll learn how to protect services from the outside (with infrastructure) and inside (through static analysis of your code). Finally, you'll see how all the pieces come together as we work through an example of an auto loan application.

Typical Attacks

What types of attacks do you need to be aware of when you're building applications to SOA using Web Services? As it turns out, the key aspects that make your SOA more successful — the ability for developers across your departments and trading partners to be able to find applications exposed as services, and the way that WSDL documents define operations that can be invoked — may also make it appealing to hackers.

Here are some potential vulnerabilities:

- *Authentication/authorization.* Without the appropriate mechanisms in place, anyone can access a service, invoke it, and perform sensitive operations even if he's not authorized. More often what typically happens is that it takes a while to de-authorize entities from accessing systems. In the meantime, they have free rein. Identity Management and Web Services Management (WSM) solutions, and XML firewalls can help.
- *Spoofing.* Gaining access to a system by using a stolen identity. If you log and audit service interactions, operations employees can identify who did what and when in the event of a spoof.
- *Tampering.* Unauthorized modification of data, such as header information including WS-Addressing, as it flows over the network. Digital signatures, which are based on the message, ensure message integrity. Another form of tampering takes the form of JavaScript that alters the contents of the XML document and can redirect the communication from the Web browser.
- *Repudiation.* Users (legitimate or otherwise) may deny that they made specific transactions. Logging and auditing capabilities address this.
- *Information disclosure.* The unwarranted exposure of private data when on the wire. Encryption helps ensure confidentiality but doesn't ensure message integrity.
- *Denial of service.* Making an application unavailable by sending a huge number of requests or very large XML payloads. (This is a problem only if it breaks your application!) XML firewalls and appliances let you protect your services from such attacks.
- *Invalid messages.* Certain XML messages can contain malicious code such as SQL statements, viruses, or worms that can compromise or corrupt data, applications, and systems, and potentially leave them exposed. This code can appear in the body of the XML document or in CDATA tags. Hackers tamper with the SOAP messages themselves to disable the service through illegitimate or malformed requests.
- *Replay attacks.* If tokens/cookies are used for secure conversation, the attackers can capture the tokens/cookies and re-use them later by replaying the same.

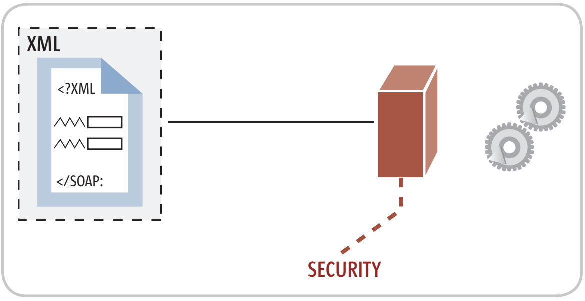


Figure 1 Security policies implemented by the service developer

Mohamad Afshar is director of product management for Oracle Fusion Middleware and has 16 years experience in the industry. His main focus is on middleware vision, strategy, and architecture, with an emphasis on Web Services, SOA, and event-driven architectures. He has a PhD in parallel database systems from Cambridge University.

mohamad.afshar@oracle.com

Roger Goudarzi is a senior software architect at Arkasoft, LLC, currently consulting with Fortify Software. He has more than 15 years of experience building multi-tiered software applications and a BS from Imperial College, London.

rogerg@arkasoft.com

- *XML routing detours/external referencing.* XML documents can contain references to external structures.

You'll recognize many of these items, as they equally apply to both Web Services and Web applications.

A Holistic Approach To Protecting Services

To protect against these potential vulnerabilities, you should take a holistic approach to security that includes infrastructures, tools, and software development practices:

- *Protect services from the outside at deployment time* by using WSM solutions. This addresses issues related to identifying the message sender, authentication, authorization, ensuring message security, and verifying the structure of SOAP headers and XML documents. Security policies can be applied to both inbound and outbound service interactions. Don't worry about these: Your WSM infrastructure sits outside your service and takes care of them for you.
- *Protect services from the inside by building security into your software development process.* You must validate the input in your code to protect yourself against attacks that result from invalid messages. By validating the content of XML documents, you address attacks that result from invalid messages, such as buffer overflow, SQL injection, XML injection, and XPath exploits. Use static analysis tools to help identify such vulnerabilities.
- *Simulate known attack patterns and fix vulnerabilities before going live* by using dynamic analysis tools in a running deployment during the QA process. By putting the attackers' hat on and stress-testing your SOA applications, you can help uncover and resolve vulnerabilities before deployment. Although you may have to work a little harder during the QA process, you'll get fewer post-deployment headaches, which more than makes up for it.
- *Monitor post go-live* using monitoring dashboards (part of WSM solutions) that present data that's collected as security policies are executed on services. These dashboards let administrators monitor compliance with IT operational best practices in real-time, such as audits on security violations on a per-Web Service, per-operation, and per-client basis. To address vulnerabilities, administrators can configure operational rules and propagate them to the appropriate enforcement components across an application deployment of any scale and complexity in real-time.

Let's look at all of these except monitoring (since the latter is more operationally focused).

Protecting Services from the Outside

Most Web Services are based on the same technology as the Web, namely HTTP. As a result, all common technologies used to secure the Web, such as Web authentication and SSL, work equally well with Web Services – for point-to-point security. With SSL alone, you can do authentication (the communication is established between two trusted parties); confidentiality (the data exchanged is encrypted); and, message integrity (the data is checked for possible corruption). However, solutions such as SSL are a little heavy-handed since they secure the entire channel. Furthermore, for many message-based interactions, intermediary steps are required before the messages arrive at their target endpoint. This leaves XML messages unsecured at each intermediary checkpoint — exposed to tampering, information disclosure, and message altering.

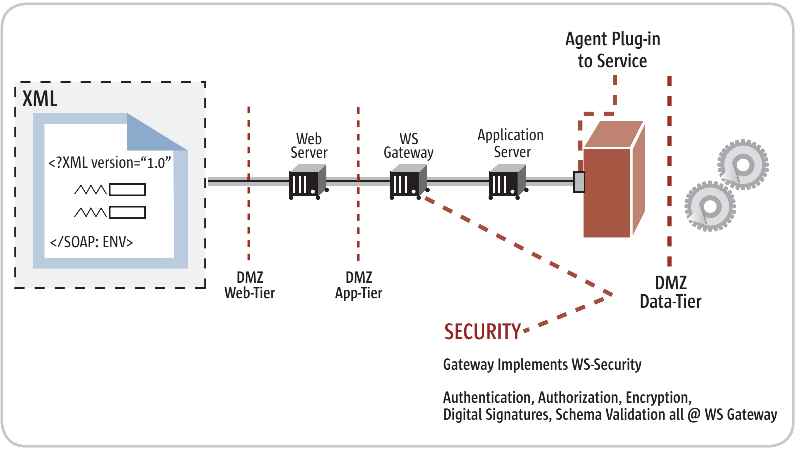


Figure 2 Security policies implemented by a Web Services management solution

To get a finer level of control and avoid intermediary security issues, it's best to secure the message rather than the complete transport. The idea is to replace SSL capabilities with message-level security, where the security information is carried in the message itself. This way, unless the intermediary or endpoints have the correct security infrastructure in place and are trusted, the message will remain secure and unreadable and can be forwarded to the next endpoint.

So how do you secure the message rather than the transport? WS-Security defines a mechanism for adding three levels of security to SOAP messages:

1. *Authentication tokens.* WS-Security authentication tokens let the client provide a user name and password or X509 certificate for the purpose of authentication headers.
2. *XML encryption.* WS-Security's use of W3C's XML encryption standard enables the XML body or portion of it to be encrypted to ensure message confidentiality.
3. *XML digital signatures.* WS-Security's use of W3C's XML digital signatures lets the message be digitally signed to ensure message integrity. The signature is based on the content of the message itself (by applying the hash function and public key), so if the message is altered en route, the signature becomes invalid.

A final thought on this – don't forget that you can use transport and message-level security together, e.g., use a WS-Security Username token without encryption and use SSL to encrypt the transaction.

There are two ways to implement this: You can embed the logic for processing tokens, handling encryption, and applying hash functions and digital signatures in the service implementation itself, or you can use a WSM solution. The first option is shown in Figure 1.

WSM solutions intercept incoming and outgoing service communications and apply a set of policies in a pipeline fashion, including authentication, authorization, decryption, signature validation, and XML schema validation. They move the active enforcement of the policies and agreements to the boundaries of an application, letting the application developer concentrate on the business logic. These solutions typically provide a policy management tool for building new security and operations policies, storing policies, and managing distribution and updates to runtime policy enforcement points. In this way, policies are defined and changed centrally but enforced locally.

Nickolaos Kavantzas is an architect for Oracle's security and identity management products. His focus is on middleware vision, strategy, and architecture, with an emphasis on policy, contract management, and security within SOAs and event-driven architectures. He was the architect of the W3C Web Services Choreography Language and lead editor of the Web Services Choreography Language working group.

nickolas.kavantzas@oracle.com

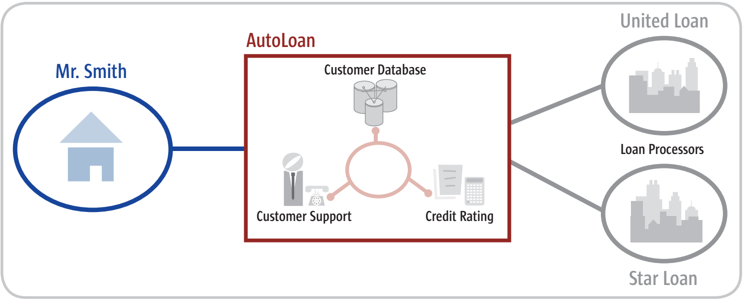


Figure 3 Loan application process

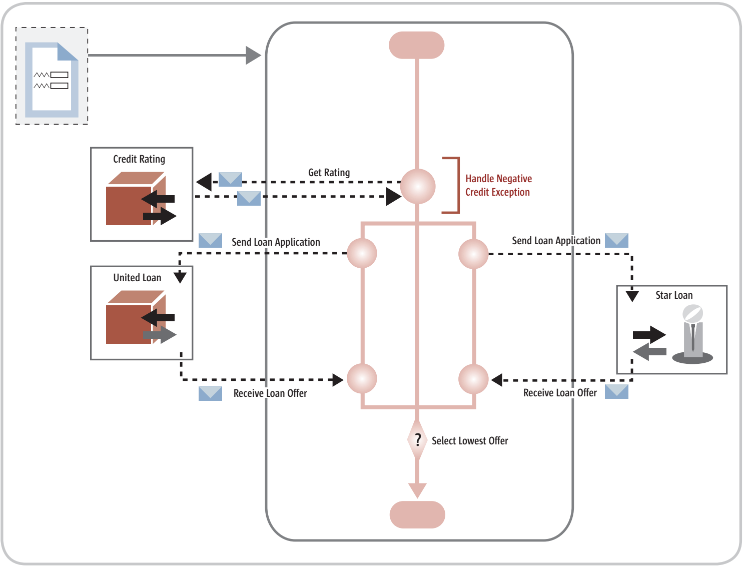


Figure 4 End-to-end message flow for loan application process in BPEL

If you have to authenticate to an Identity Management System that's not supported by the WSM solution out-of-the box, such as a JAAS login module, Oracle Web Services Manager, as well as many WSM products, provides an SDK for developing policy steps. They provide operational dashboards for monitoring policies as they execute to ensure service levels, incorporating alerts so corrective actions can be taken in a timely fashion.

WSM solutions typically provide two types of enforcement components and policy enforcement points: Gateway and Agents. Gateways are deployed in front of a group of applications or services. They can intercept inbound requests to these applications to enforce policy steps, adding application security and other operation rules to applications that are already deployed. They also allow (or deny) inside users access to predetermined outside services. Agents provide an additional fine-grained level of security by plugging directly into an application or service.

Service virtualization is also a key capability. Typically an Internet user makes a service request using a username and password combination, but the service may be expecting a SAML assertion. With a WSM solution, you can have a gateway on the requester side that intercepts the request, authenticates the user with the username/ password combination, and inserts a SAML assertion that can be validated at the service by a WSM agent. In effect, the user calls a service in a virtual way through the credentials that he knows, not the credentials that the Web Service is expecting.

Barmak Meftah, vice-president of engineering at Fortify Software, has more than 15 years of experience in enterprise software development and product management with acknowledged industry leaders. He joined Fortify from Sychron, where he was vice-president of engineering and product management. Meftah previously spent seven years at Oracle overseeing the delivery of the Oracle Database for the Windows Server family.

barmak@fortifysoftware.com

Figure 2 shows the context in which a WSM solution can be deployed. Note that you could deploy an XML firewall or appliance before the gateway. These appliances are typically good at applying macro policies and protecting against attacks such as buffer overflow attacks, which don't require application context.

Securing Services from the Inside

Securing applications from the outside isn't enough to protect your application. Web Services gateway solutions, whether implemented in an appliance or software, can't confidently check the content of XML messages since they don't have the application context. Hackers use this knowledge to embed malicious content in the XML documents that pass straight through the WSM software to the service interface of the application.

Common Hacking Techniques

Hackers use common techniques such as SQL injection, stack and heap buffer overflows, command injection, and cross-site scripting to find pathways into applications. Their intention is to have the software return information that it wasn't designed to give, letting them access sensitive data they're not authorized to see or cause a denial of service by crashing applications. These techniques work on any type of software — it's a Web-based application or a Web Service with any kind of attack surface exposed.

XML Injection

Two of the more common hacking techniques with Web Services are XML and XPath injections, where a hacker manipulates or injects malicious input into an XML document or an XPath query.

In SQL injection attacks, a hacker injects malicious input (SQL statements) disguised as data into an application via an XML document or Web form, hoping that the input will end up in a WHERE clause of a SQL query that's executed against a backend database. The intention is to gain access to data that the hacker isn't authorized to see. You can easily address this by checking the data in the XML document for SQL statements.

XPath Exploits

XPath exploits are much like SQL injection attacks. In this case a hacker injects malicious input into an XML document, hoping that it will go through a dynamically created XPath query against an XML document in a native XML database and retrieve data that the developer didn't intend for it to retrieve. A typical malicious input for XPath exploits is *OR 1=1 OR "=*. This expression, when executed in the content of an XML document, will always return true and end up returning some data along the way!

Let's use a computer element in an XML document as an example:

```
<computer>
 <manufacturer>Toshiba</manufacturer>
 <name>T200</name>
 <ram>1GB</ram>
 <cpu>1.83MHz</cpu>
 <monitor>14.2</monitor>
</computer>
```

Here are some Xpath expression examples that could be issued against this element:

```
"/computer" returns the root computer element
"//computer" returns all computer elements in the document
"computer//ram" returns all RAMs under computer element
```

Now, here's how an XPath injection attack would work:

```
//computer[name = 'T200' and ram = '1GB']  will return the com-
puter with this name and ram.
```

Here's the same expression with an XPath exploit:

```
//computer[name= 'T200' or 1=1 or '' = '' and ram = '1GB']  will
return all of the computers.
```

Imagine replacing this scenario with one in which we have HR records that embed Social Security numbers!

Command Injection and Cross-Site Scripting

Command injection exploits occur when a hacker embeds an XML document with a script, such as a shell script, in hopes that the script will be executed.

With cross-site scripting, a hacker also embeds a script into an XML document — but this time, he hopes the script will be stored (for example, in a database) and then served to someone's Web browser as part of a subsequent Web interaction. The script is executed in the browser of the unknowing user and can do things such as steal usernames and passwords. In both of these examples, the hacker ends up "smuggling" some code through the firewall, Web Service Gateway, and XML appliance (if deployed), and relies on it being run at a later date — on either the server or client side.

Solution

The way to secure applications from inside and protect against the type of attacks we've discussed is to do input cleansing in the application code. You can use your development skills to do this, but there's help at hand.

Static analysis tools are evolving to provide an automated way to check large amounts of code and data flows to find common software security vulnerabilities related to malicious input during the software development phase. They can help you easily catch and fix common mistakes such as those mentioned above. Using proper threat analysis and abuse-case modeling during the software requirements and design phases to identify security flaws architecturally can also mitigate potential security risks during deployment — as well as in the future.

Simulating Known Attack Patterns

Another good technique to deploy in combination with those mentioned above is dynamic software analysis. This step could easily fit into the quality assurance and testing phase of the software development lifecycle.

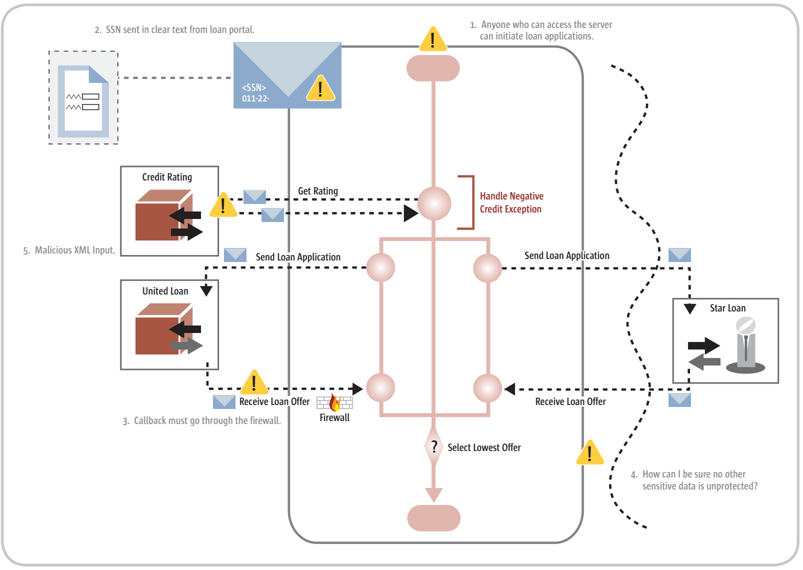

2. SSN sent in clear text from loan portal.
<SSN> 011-22-
1. Anyone who can access the server can initiate loan applications.
Credit Rating
Get Rating
Handle Negative Credit Exception
5. Malicious XML Input.
Send Loan Application
Send Loan Application
United Loan
Star Loan
Receive Loan Offer
Firewall
Receive Loan Offer
3. Callback must go through the firewall.
Select Lowest Offer
4. How can I be sure no other sensitive data is unprotected?

Figure 5 Security vulnerabilities in BPEL loan application process

Much like static analysis, which analyzes source code statically, dynamic or runtime analysis analyzes software at runtime. The dynamic analyzer simulates attacks against the application and tries to exploit it. When it succeeds, it gives the result and identifies which areas of the application are vulnerable to the simulated attacks.

Dynamic analysis tools target a different class of exploits that static analysis tools can't catch since they are runtime attacks. Examples of these attacks include probing, forceful browsing, and time-based attacks such as click fraud. In the latter case, the number of times a page is hit in a given amount of time is of interest. This metric can't be extracted during static analysis. In fact, you must add checks in your code to detect them or use a solution that discovers such behavior at runtime.

Ideally you should use both dynamic and static analysis tools. Aside from increased security, this lets you pinpoint the root cause of the vulnerabilities caught during dynamic analysis and correlate them back to the exact sector in the code that's vulnerable.

Case Study: Auto Loan Application Processing

Let's take the example of a loan procurement application implemented by a loan broker named *AutoLoan* that offers its application as a Web Service invoked from a portal or directly by any of its trading partners. The loan application process (service) is implemented as an orchestration of services provided by internal systems and services provided by two trading partners — Star Loan and United Loan — that provide the financing. These companies provide Web Services for accepting loan applications, returning loan offers, and issuing loan policies when an offer is accepted. The setup is shown in Figure 3.

AutoLoan implemented this process using a service orchestration platform based on Business Process Execution Language (BPEL). The message flow for this scenario is shown in Figure 4.

As part of the loan application, the customer, Mr. Smith, must provide certain sensitive data, such as his Social Security number. Without an appropriate security infrastructure in

Ramana Turlapati is a consulting member of the technical staff at Oracle with 12 years of industry experience. In his current role as security architect for Oracle Web Services Manager, he contributes to Oracle's overall Web Services security solutions and strategies.

ramana.turlapati@oracle.com

Prakash Yamuna is a principal member of Oracle's technical staff, working on Oracle Web Services Manager. His current focus is on policy management and security within SOAs.

prakash.yamuna@oracle.com

"As it turns out, the key aspects that make your SOA more successful appeal to hackers"

place, as the loan application request (message) flows through the various services, the SSN is sent in clear text — exposing sensitive information and potentially leading to unwarranted *information disclosures*. The message may be subject to attacks that include *tampering*. Besides tampering, there may be *replay attacks* where the message is hijacked and re-submitted multiple times. Since Mr. Smith logs onto a Web site and submits the application online, the identity of Mr. Smith must be part of the message sent from the loan portal to the Web Services. This requires the ability to propagate the identity of Mr. Smith as part of the service request and the ability to enable *single-sign on* and *identity propagation* in the backend systems.

Certain checks may also have to be done to make sure the customer is *authorized* to request a loan (for example, loans may be available only to existing customers). Since Mr. Smith may claim that he never submitted the loan application, the system must support *non-repudiation* through message auditing. In the new regulatory environment, companies may be required to adhere to *audit compliance standards* that would require the system to be able to *audit* events such as authentication and authorization.

Since AutoLoan exposes its application to trading partners, messages flowing back and forth could potentially contain SQL injection, XML injection, or other attacks aimed at AutoLoan's systems.

These security vulnerabilities are shown in Figure 5.

Securing the BPEL Loan Application Process

To develop the loan application process, we advocate using a WSM solution so you avoid embedding security policies in the service code. You would then do static and dynamic analysis on the application to make sure it's protected from the inside and simulate some attack patterns.

Applying Security Policies to the Service

The developer of the loan application service or the operational manager must select the appropriate security scenario policy from the policy library provide by the WSM software. In general, security scenario policies capture security patterns that have emerged from common best practices when providing end-to-end message security to services. For example, a security scenario policy includes authentication information such as whether the service accepts username/password or X509 certificates, whether the message should be signed and encrypted, and the information needed for authorization and auditing. Since the body of the message must be integrity-protected and encrypted for the message exchange, the developer specifies this by editing the message protection attributes of the X509 security scenario.

Once the scenario policies are associated with the service, the WSM software will automatically enforce them when a message is received at this service.

Applying Matching Policies on the Client Side

Now let's look at the developer who creates the client/service consumer that calls the loan application service.

The developer discovers the Loan Flow Service from a UDDI registry and retrieves the WSDL containing the service provider policies. The server hosting the loan application service will return the WSDL, which contains the policies, and policy references that are attached at various levels in the WSDL based on the WS-Policy Attachment specification. These policies, based on the WS-Policy specification, describe in an interoperable way the capabilities and constraints that the consumer must meet to communicate successfully with the loan application process. The client developer finds a policy from the policy library governing the interaction for the client side that adheres to constraints specified by the service provider policy. In some environments, the integrated development environment (IDE) may be integrated with a centralized metadata repository; if so, the developer can browse for policies through a resource catalog. (If one isn't available, the policy information has to be provided to the developer of the service consumer by some other means, such as text.)

The client developer uses the policy designer (part of the WSM solution) to specify the exact parameters — for example, the algorithm to be used for encryption/decryption and the order in which the message should be signed and encrypted — and attaches it to the service consumer through the IDE. The policies will be automatically enforced by the WSM software on both the client and server side when the client sends the message to the service. If no WSM software is available on the client side, the application server software on which the client runs will implement the policies, as long as it conforms to the WS-* specifications noted above.

Runtime View of WSM in Action

Let's explore the interactions between the messages and infrastructure components shown in Figure 6.

The service consumer's call to the service is intercepted by the WSM policy enforcement agent on the client side.

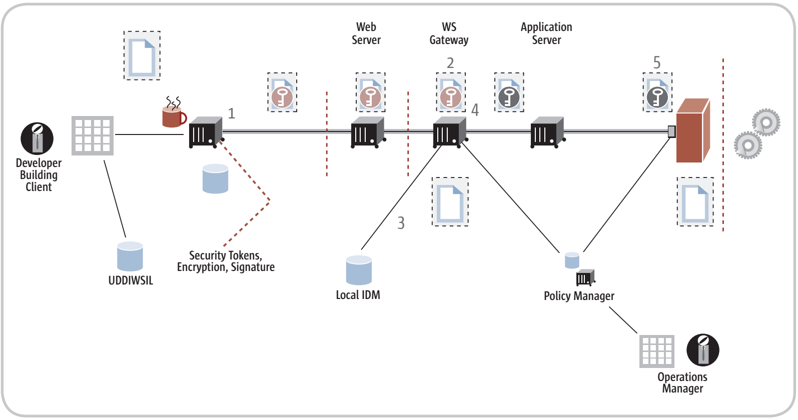


Figure 6 Service invocation walkthrough

The agent enforces the security policy scenario that was attached to the service consumer. For instance, the agent inserts a security token, such as username and password, for authentication purposes and signs and encrypts the message body to satisfy the service's integrity and confidentiality requirements. The security information is attached to the SOAP request using standard WS-Security mechanisms. The request traverses firewalls and Web servers to reach the virtual endpoint of the service exposed typically through a Web Services gateway. The gateway contacts the policy manager to query the security policy requirements associated with the service endpoint and enforces these requirements on the incoming service request in a pipeline fashion.

In this case, the SOAP request is first decrypted and its signature is validated. Subsequently, the claims presented in the authentication token are validated against a locally configured Identity Management system. Optionally, the agent can do an authorization check to determine if the authenticated user has access to the service. Once the user's identity and access is established, it's asserted to the service implementation to facilitate additional service-specific authorization checks. The WSM gateway can then propagate the user's identity to the actual service by inserting a SAML assertion into the message. Once the message is received by the actual service, the identity in the message is retrieved by the WSM agent and is asserted to the container as a JAAS subject.

Conclusion

We hope we've convinced you that it's best to take security policies out of service implementations and capture them in a WSM solution that implements authentication, authorization, and encryption, applies digital signatures;,and performs schema validation.

As developers, we can't always purport to be security (or identity management) experts. As a result, the more security policies are kept out of the application code, the better – externalizing security allows better assert management and makes the developers' lives easier! Static and dynamic analysis tools help you cover the bases by making sure your code does the right checks on the input XML data. Simulating attacks before deploying, and monitoring your setup afterward, helps you obviate risk. We didn't talk much about monitoring. Suffice to say that it's an integral part of the feedback loop for continuously improving security. A holistic approach to security in SOAs that combines infrastructures, tools, and secure coding practices, and builds security early — from design through deployment — will make SOA security less of a headache, and may even simplify it.

Acknowledgements

The authors would like to thank Marc Chanliau for providing many helpful comments on this paper.

DEVELOPERS,
LOOK NO FURTHER

ODTUG 2006
KALEIDOSCOPE
CLARIFYING THE DEVELOPER'S CHANGING UNIVERSE

Highlights:

- Web Architecture Symposium Saturday, June 17
- Hands-on Training
- BI Summit
- More Than 100 Technical Presentations
- Built-in Time to Meet and Talk with the Experts

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

To obtain the special conference rate of $175/night at the Wardman Park Marriott Hotel, call 1-800-Marriott by May 12, 2006. Attention government employees: Government rates are available, but conference room rates are lower.

TOPICS:

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

www.ODTUG.com

Joe Winchester
Desktop Java Editor

Swing Baby, **Yeah!!!**

Back in 1996, Java was originally hailed as a way of making the Web more appealing through applets, and, with its "write one, run anywhere" philosophy, as the holy grail for desktop apps that would be truly cross platform. The truth is that both were oversold at the time. With the combination of low bandwidth Internet connections and early Swing releases not living up to user expectations occurring in the middle of the Microsoft vs. Sun "pure Java" fight that resulted in JVMs being pulled from Internet Explorer, Java's attention moved off the desktop and onto the server.

It's now 2006 and the world is a very different place. Java Virtual Machines are now present on about 85% of desktop PCs, about two-thirds of which are at 1.3 or higher. The top 10 PC OEMs now redistribute a JRE with their product and, in January 2006 alone, there were over 30 million downloads of the Windows Java SE.

More intelligent and demanding users are become increasing disillusioned with the poor page-based, latency bound user interface model that the browser-centric world delivers. To a large extent all of the AJAX hype being whipped up at the moment is a recognition of this fact, although what it's actually doing is elevating the discussion to be one of "how do we deliver rich content" rather than bun fights over "HTML rocks, Swing sucks, baa."

At JavaOne's opening keynote this year, the absolute highlight for me that showed how far Swing has come was a demo of an application that used Web services APIs from flickr.com and maps.google.com, popular and high-content Web sites in their own rights, and trumped both sites with a beautiful Swing program. Romain Guy, one of the Swing engineers who'd written the program, used it to navigate his photo album pictures from flickr, using 3D smooth animation, reflection techniques, and beautiful user interface effects that fully demonstrated the power of desktop applications. The program then picked up Google maps data to show the route Romain had taken during a recent road trip that merged with photos he'd taken on the way. The power of this message was simple: the desktop can be used to integrate back-end data that is available on the Web (through Web services) and create a user interface experience that is richer than anything the Web could offer and truly puts the user back in control. It was beautiful to watch, and ironically reminded me of how Flash developers used to show off how they could make the Web experience better, except the Swing demo was an order of magnitude higher than anything Web 2.0 could possibly hope to offer. It also struck a chord with the idea that while all the server guys are focusing on SOA and ways for back ends to publish their data and re-usable services, it's now open season again as to what the user interface is going to be. Swing is totally there now: mature, fixed, better, and 100% ready to step up to the challenge.

Leaving bombast aside, technology never wins because it's better. The battleground in traction is largely based on perception, ideas, and adoption by the community. What's encouraging about Swing is that it is doing well on all three fronts.

In a recent Evans Data survey of Java development trends, developers were asked where they spent most of their time building apps. Forty-one percent said they wrote desktop apps, 37% J2EE, 4% Java on mobile devices, while the remainder apparently didn't know what they do at work each day As for the future, when questioned about which Mustang features people were most looking forward to, over 60% answered that they're turned on by the Desktop Java enhancements.

The community around Desktop Java has really grown over the past few years and Mustang is now reaping some of this crop. SwingLabs (http://swinglabs.org/index.jsp) is an umbrella for an open source laboratory of projects whose common goal is to make Swing easier to write, faster, and better and to rally around Desktop Java. It's crop includes the timing framework being showcased at JavaOne, a very crisp and elegant API to do animation effects. SwingX has reaped the cool visual effects like drop shadow borders and custom highlighting that powered the JavaOne keynote demo. In addition to these and other fabulous improvements to Swing on the glass, there is also recognition that serious Desktop Java projects have to wrestle many other types of animal to the ground as well. The Data Binding project is looking at taking away the pain involved when binding Swing GUIs to back-end data sources. The Application Framework project is looking at the life cycle surrounding desktop. This is all good stuff.

One of the features Swing used to always get beaten up about was that it's an emulated widget toolkit and, as such, has to fake out in low-level Java 2D drawing code what the actual native widgets look like. Even though 1.4.1 improved this over previous releases, the arrow didn't quite hit the bulls-eye. Thankfully though this looks like it's now got closure as Mustang is going to use native APIs on Windows and GTK to find out how exactly, pixel perfect, each OS's widgets are being rendered. No more Swing versus SWT fights on newsgroups, phew.

Swing is really coming back hard with some great new features and for me was definitely the star of JavaOne. I talked to many other people who were also impressed with what they saw and are now seriously considering revisiting it. Web 2.0 hysteria is turning the focus of IT away from server-implementation technologies and back onto the user interface experience. When the guy next to you steps up to home plate clutching his JavaScript for Dummies book, let's hear it for the desktop: "Swing AJAX developer swing, swing AJAX developer swing."

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

joewinchester@sys-con.com

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.

SYS-CON MEDIA

ONLY $69.99

ONE YEAR
12 ISSUES

Subscription Price Includes FREE JDJ Digital Edition!

www.JDJ.SYS-CON.com

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

OUT OF MEMORY!
IN THIS ISSUE...
Leveraging Open Source Solutions to Improve Productivity While Using EJBs
Spring Web Flow
Intelligent Web Applications with AJAX
Navigating the Global Enterprise
JDJ
ARE PORTALS THE MAGIC BULLET OF WEB APPLICATION DEVELOPMENT?
PLUS...
The Performance of EJB 3.0
Dealing with Open Source Software
A Essential for Developing Language Tools
THE WORLD'S LEADING i-TECHNOLOGY MAGAZINE JDJ.SYS-CON.COM

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Building an Instant Messaging Application Using Jabber/XMPP

An adventure with Smack and Wildfire

by Pramod Jain and Mahaveer Jain

This article will describe our experiences with developing a Java-based instant messenger application using Jabber/XMPP (Extensible Messaging and Presence Protocol) — a free, open and public protocol and technology for instant messaging. According to the Jabber Software Foundation, "Under the hood, Jabber is a set of streaming XML protocols and technologies that enable any two entities on the Internet to exchange messages, presence, and other structured information in close to real-time."

Google Talk uses the standard Jabber/XMPP protocol for authenticating, presence, and messaging. But using Jabber goes beyond instant messaging to almost real-time server-to-server communication.

This article will describe the Jabber/XMPP protocol for messaging, the Jabber/XMPP client program based on JFace and Eclipse, and the Jabber/XMPP Java server. These will be illustrated through Open Source Smack and the Wildfire Server 2.4.4. It will offer examples of a login-class plug-in for custom authentication, a plug-in and server extension for custom messages, and a server-side extension for database interactions. Each example will contain the XML messages, client-side code, and server-side code.

What's Different About Jabber?

Compared to traditional IM programs, Jabber:

1. Is an XML-based messaging protocol that is open and extensible
2. It lets you build custom client IM applications. We'll talk about Smack, a project based on JFace and Eclipse that provides tools for building custom client applications
3. It lets you extend the functionality of the server to process custom messages by writing server-side plug-ins. We'll describe the plug-in architecture of the Wildfire Server

What Are the Basic Features of Instant Messaging?

Any instant messaging system will have these basic features: It connects to server, registers new users, logs in, gets presence information, exchanges messages, and does custom interactions (VoIP, Web conferencing, etc.). Below each of these IM features are described using XMPP:

- ***Connect to Server*** – This is done through a XML stream header exchange.

The client sends:

```
<stream to='192.168.0.12:5222' xmlns='jabber:client'/>
```

The client receives:

```
<stream id='xxxx' from:'192.168.0.12:5222' xmlns='jabber:
client'/>
```

- ***Register*** –The client sends a message to discover the server's requirements for registration:

```
<iq type='get' id='reg1' to='192.168.0.12:5222'><query
xmlns='jabber:iq:register'/></iq>
```

The server responds with a message that lists the fields required for registration:

```
<iq type='result' id='reg1'> <query xmlns='jabber:iq:register'>
  <instructions>

Choose a username and password for use with this service.
Please also provide your email address.

  </instructions>
   <username/>
   <password/>
   <email/>
 </query></iq>
```

The client responds by sending the requested information:

```
<iq type='set' id='reg2'>
  <query xmlns='jabber:iq:register'>
    <username>a3</username>
    <password>password</password>
    <email>a3@sow.com</email>
  </query>
</iq>
```

On successful registration, the server responds:

```
<iq type='result' id='reg2'/>
```

Pramod Jain is president of Innovative Decision Technologies, Inc. (INDENT, www.indent.org), in Jacksonville, FL. Their clients include Recruitmax, NASA, and NIH. Pramod has a PhD from the University of California, Berkeley.

pramod@indent.org

Mahaveer Jain is a lead programmer at INDENT. His expertise is in developing collaboration applications with Java technologies.

mahaveer@indent.org

Declare Presence and Get Presence Information

To declare presence information, the client sends:

```
<presence xmlns=”” id=”d6vNV-3” from=”mahaveerj@sow/1139352545625”/>
```

Note the “to” attribute isn’t required in the above message because the client knows which server it’s talking to.

To get presence information, client sends:

```
<iq xmlns=”” id=”d6vNV-2” type=”get” from=”mahaveerj@sow/113935254
5625”>
<query xmlns=”jabber:iq:roster”/>
</iq>
```

The server responds with presence data about two people in the roster:

```
<iq xmlns=”” type=”result” id=”d6vNV-2” to=”mahaveerj@sow/11393525456
25”>
 <query xmlns=”jabber:iq:roster”>
  <item jid=”jorge@sow” name=”Jorge” subscription=”both”></item>
  <item jid=”bruce@sow” name=”Dr. Bruce” subscription=”both”></item>
 </query></iq>
```

Exchange Messages

User a1 sends message to user a2:

```
<message xmlns=”” id=”H49LH-12” to=”a1@sow” type=”chat” from=”a2
@sow/1139385809707”>
  <body>Good Morning, I am testing chat in Jabber Client.</body>
</message>
```

The elements above form the core of XMPP (Extensible Messaging and Presence Protocol) XML syntax. It also gives you a sense of the flow of XML messages between the client and server to accomplish a task.

Jabber/XMPP Client and Server

Since Jabber/XMPP is an open protocol scores of implementations have been created for the client and server, some Open Source and others commercial. There are literally hundreds of Jabber clients available. (See References.)

In this article we’ll use the Smack library (http://www.jivesoftware.org/smack/) to illustrate client-side functionality. Smack provides an API for implementing the GUI and managing XML data. Managing XML data involves creating XML messages, sending the messages, receiving messages from the server, and processing the incoming messages.

We’ll use the Wildfire Server (http://www.jivesoftware.org/wildfire/) for illustrating creating server-side plug-ins. For basic IM needs, no server-side programming is required. For other needs Wildfire provides a plug-in architecture for extending the server’s functionality. We’ll look next at a use case to illustrate how custom messages can be created and processed at both the client and server.

Use Case 1: Custom Authentication and Custom Queries

Consider a case in which an IM client and a Web portal are part of a suite of collaboration applications. The Web portal has database-driven MD5-based authentication and the requirement is that IM client use the portal’s authentication service. This will allow users to log in to IM and then launch applications on the portal from the IM client without logging into the portal.

Part A: Custom Authentication

The Wildfire Server uses the DefaultAuthProvider class to authenticate users and DefaultUserProvider to get user information. These use the database tables Wildfire provides.

If you want to use a custom authentication method you’ll have to change the user and authentication provider in wildfire.xml.

```
<provider>
<user> <className>org.indent.wildfire.user.SOWUserProvider</class-
Name> </user>
<auth><className>org.indent.wildfire.auth.SOWAuthProvider</className>
</auth>
</provider>
```

The simple extension defined in Listing 1 uses custom tables and custom encryption methods. The SOWUserProvider class is for user info and SOWAuthProvider is for authentication.

The two static variables shown define the database authentication strings. And if custom encryption is used then the encryptPassword() method provides the encryption logic.

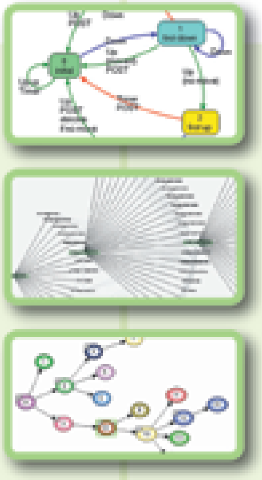
Part B: Retrieving a Token from the Database

In this use case, after authenticating, the client is going to ask the database for an encrypted token that uniquely identifies the client. This token is later used to launch applications in the Web portal without explicitly logging into the portal. The steps are:

a) The client generates a custom message to request a token

Build interactive diagrams easier than you ever imagined

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



FREE Download With Full Support!

GoDiagram

Interactive diagram components

www.nwoods.com

b) The server gets the message and processes the request by getting a token from the database. The server generates a reply message that contains the token and sends it to the requesting client

c) The client gets the message and extracts the token

This process is illustrated in Figure 1.

Client to Server

The client generates a <iq> packet that looks like:

```
<iq to="serverName" type="get"><query xmlns="jabber:iq:token"/></iq>
```

jabber:iq:token is a namespace for our custom query. This message is generated at the client using the following code:

```
   XMPPConnection connection = session.getConnection();
   ClientID packet = new ClientID();
   connection.sendPacket(packet);
```

The Class ClientID is defined in the section "Client-side Processing of Reply Packets" below.

The Server-Side Plug-In

The first step is to create a plug-in that will process the custom message with xmlns="jabber:iq:token." The plug-in is called ClientTokenPlugin. It first creates an IQTokenHandler; the handler is then added to the iq router that routes all the incoming iq messages:

```
public class ClientTokenPlugin implements Plugin {
 public ClientTokenPlugin() {
   IQHandler iqRDHandler = new IQTokenHandler();
   IQRouter iqRouter = XMPPServer.getInstance().getIQRouter();
   iqRouter.addHandler(iqRDHandler);
 }
 public void initializePlugin(PluginManager manager, File pluginDirec-
tory) {               }
 public void destroyPlugin() {   }
}
```

For more on how to create a plug-in, see http://www.jivesoftware.org/builds/wildfire/docs/latest/documentation/plugin-dev-guide.html.

IQTokenHandler has two key methods:

a) getInfo() which returns the type of queries in the iq messages that the token handler is going to process

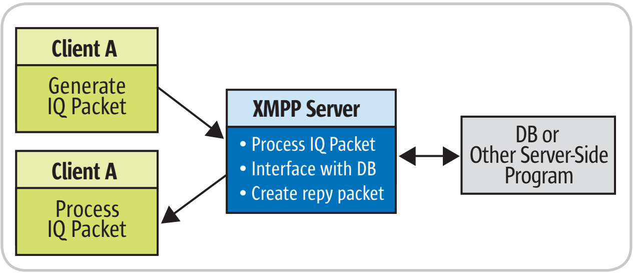


Figure 1 Data flow for interaction between client, server, and DB

b) handleIQ() which processes the incoming message/packet and creates a reply message/packet

```
public class IQTokenHandler extends IQHandler {
 public IQTokenHandler() {      super("Client Token Handler");    }
 public IQHandlerInfo getInfo() {
   return new IQHandlerInfo("query","jabber:iq:token"); }

  public IQ handleIQ(IQ packet) {
   IQ replyPacket = IQ.createResultIQ(packet);
   Element m = replyPacket.setChildElement("query", "jabber:iq:token");
// 2 lines below are specific to our needs; change them to get data
according to your needs
   ClientSession session = sessionManager.getSession(packet.getFrom());
    String token = (String)UserTokenList.get(JID.unescapeNode(session.
getAddress().getNode().toLowerCase()));
    m.addElement("tokenNum").addText(token);
    return replyPacket;
 }
}
```

The reply message will look like:

```
<iq to="clientName">
 <query xmlns="jabber:iq:token">
  <tokenNum>abcdefgh</tokenNum>
 </query></iq>
```

Client-Side Processing of Reply Packets

At the client end, you'll have to define a class to handles the IQ message. This is defined in a configuration file called smack.provider:

```
    <iqProvider>
        <elementName>query</elementName>
        <namespace>jabber:iq:token</namespace>
        <className>org.jivesoftware.smack.packet.ClientID$Provider</
          className>
    </iqProvider>
```

The iqProvider defines the class and methods for both generating custom iq messages with the namespace jabber:iq:token for sending to server or other clients and for processing incoming iq messages with the namespace jabber:iq:token.

The first section of the code is for generating an iq packet, as described in the section "Client to Server" above. The provider contains a parseIQ() method to parse the incoming message and extract useful data and the tokenNum in the reply message above and stores them in object c in Listing 2. This ends the lifecycle of the iq packet.

Use Case 2: The Custom Message

During collaboration, userA wants to launch a browser with a URL in userB's machine. This will be implemented as a custom <message>. Usually <message> is used to carry chat messages, but in this use case it will be used to carry a custom message (<xsow>) like:

```
 <message xmlns="" id="WHF12-7" to="userB@sow" from="userA@sow/11393225
62328">
```

```
  <xsow xmlns="http://www.indent.org">
   <conf Url="http://my.shareonWeb.com/jetspeed/indent/sowWCLogin.
jsp?p=2079810"/>
  </xsow>
 </message>
```

The steps are:

1. At the client SOWExtension, a custom PacketExtension, is implemented to create the message and the provider to parse the incoming custom message
2. At the receiving client, a packet filter and listener are implemented to trap the custom message and process it.

This is illustrated in Figure 2.

Creating a Custom Message at the Client

In the use case, the first step for userA to send a custom message to userB is to do the following in the client GUI: userA chooses "Open Document" from a menu or right-clicks on userB in the roster list and chooses "Open Document." The client GUI in our case uses Smack and JFace; "Open Document" triggers a call to the run() method associated with the selection. The run() method creates a message and sends to server, which it then routes to userB.

```
  public void run() {
   Message iceMessage = new Message();
   SOWExtension icePacket = new SOWExtension();

   String urlToSend = this.appData.aspireRequest + "&p=" + ClientID.token;
   icePacket.setUrl(urlToSend);
   iceMessage.addExtension(icePacket);
   iceMessage.setType(Message.Type.NORMAL);
   for (Iterator i=selection.iterator(); i.hasNext();) {
    Object user = i.next();
    RosterEntry entry = (RosterEntry) user;
    iceMessage.setTo(entry.getUser());
    Session.getInstance().getConnection().sendPacket(iceMessage);
   }
  }
```

A standard message/packet is created first; an extension is created on the message using the new SOWExtension() — this is going to carry the custom packet; the custom packet is added to the message using addExtension(icePacket). The for loop then sends the message to each selected user (userB, userC, …). The sendPacket() method calls toXML() in Listing 3 to create an XML string.

The SOWExtension class implements PacketExtension; one of the methods of interest is the class is toXML() and the class of interest is the Provider class. The Provider class provides the parse method for incoming custom messages with the extension <xsow>. The provider class will be discussed in the next section.

Processing the Custom Message at the Client

Note that the server doesn't do any operation on the message other than route it to userB. userB gets the message first by parsing it using the parseExtension() in Listing 3, and then passing the message through a filter described below. *filterExt* below is a filter that looks for packets with the extension "xsow." When the filter recognizes such a message, a *listener* then processes the message. The processPacket() method below first extracts the extension in the custom message by calling message.getExtension(). getExtension() returns an SOWExtension object defined in Listing 3. Since the custom message above has already been parsed, this object contains all the relevant information contained in the packet extension. sowExt.getURL() returns the desired value. The last two lines in listener below correspond to the desired action, in this case, opening a browse with a URL that was contained in the extension of the custom message above.

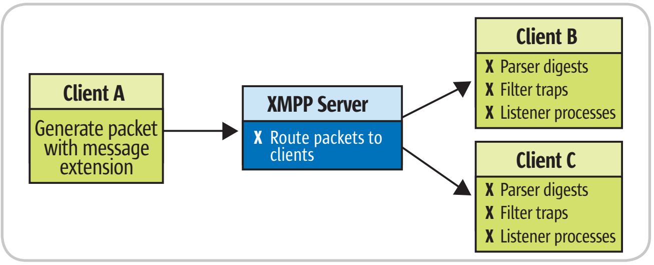


Figure 2 Custom message data flow and steps

```
PacketListener listener = new PacketListener() {
 public void processPacket(Packet packet) {
  final Message message = (Message) packet;
  SOWExtension sowExt = (SOWExtension) message.getExtension(
SOWExtension.elementname, SOWExtension.namespace);
  String url = sowExt.getUrl() + "&uname=" +
   Session.getInstance().getConnectionDetails().getUserId();
  Program.launch(url);
 }
};
PacketExtensionFilter filterExt = new PacketExtensionFilter("xsow",
"http://www.indent.org");
connection.addPacketListener(listener, filterExt);
```

The message extension and the SOWExtension's Provider are defined in smack.provider.

```
    <extensionProvider>
        <elementName>xsow</elementName>
        <namespace>http://www.indent.org</namespace>
        <className>org.jivesoftware.smackx.packet.
SOWExtension$Provider</className>
    </extensionProvider>
```

Note: iqProvider to handle custom IQ messages was defined in reply packets code above. extensionProvider to handle custom message extension is defined in the code immediately above.

Summary

There are several Open Source and commercial implementations of XMPP server and client to choose from that provide basic chat, presence, and roster functionality. Here we discussed how to extend the functionality of your XMPP-based IM applications through two mechanisms: defining custom queries in the <iq> element and defining extensions in the <message> element. Client-side extensions and server-side plug-ins are described to accomplish the custom functionality.

Some of the specific uses of the <iq> extension will be to create rich IM applications that interact with a backend application server and/or database; examples of uses of the <message> extension include doing things like opening browsers and sending files from one user's client IM application to another.

References

- Jabber/XMPP: For functionality see http://www.jabber.org/software/clients.shtml.
- For Jabber server implementations see http://www.jabber.org/software/servers.shtml.
- For components that can be plugged into a Jabber server see http://www.jabber.org/software/components.shtml.
- For libraries to develop client, server, and components see http://www.jabber.org/software/libraries.shtml.
- D.J. Adams. *Programming Jabber*. O'Reilly. 2002.
- William Wright and Dana Moore. *Jabber Developer's Handbook*. SAMS. 2003.
- http://www.jivesoftware.org/
- http://www.jivesoftware.org/builds/wildfire/docs/latest/documentation/plugin-dev-guide.html

Acknowledgements

We would like to acknowledge partial funding from NIMH of NIH, and the help of Gaurav Mantro for coding some pieces of the client application and thank Ryan Graham, Gaston Dombiak, and a few others for answering questions we posted on www.jivesoftware.org's discussion forum for developers.

Listing 1

```
package org.indent.wildfire.auth;

public class SOWAuthProvider implements AuthProvider {
 // Change these select to point to your table rather than using jiveUs-
er of wildfire.
   private static final String AUTHORIZE =
         "SELECT username FROM jiveUser WHERE username=? AND password=?";
   private static final String SELECT_PASSWORD =
         "SELECT password FROM jiveUser WHERE username=?";

   public void authenticate(String username, String password) throws Un-
authorizedException {
   // Encrypt the password (optional)
                        String ePassword = encryptPassword(password);
  }

  public void authenticate(String username, String token, String digest)
throws UnauthorizedException { ....      }
   public boolean isPlainSupported() {       return true;    }

   public boolean isDigestSupported() {       return true;    }

// This is optional, used if you have encryption algorithm for authen-
tication
   public String encryptPassword(String password) {        .....  return
ePassword;    }
}
```

Listing 2

```
public class ClientID extends IQ {
public static String token;
public static String namespace = "jabber:iq:token";
public static String elementname = "query";
public ClientID() {
   setType(IQ.Type.GET);
}
   public String getChildElementXML() {
         StringBuffer buf = new StringBuffer();
         buf.append("<query xmlns=\"jabber:iq:random\">");
         buf.append("</query>");
         return buf.toString();
  }
public void setToken(String token)
   {
            this.token = token;
   }
public String getToken()
{
            return this.token;
}
public static class Provider implements IQProvider {
  /*Creates a new Provider. ProviderManager requires that every Pack-
etExtensionProvider has a public,no-argument constructor    */
            public Provider() {
   }
   /** * Parses an incoming packet (extension sub-packet).        */
   public IQ parseIQ(XmlPullParser parser) throws Exception {
                        ClientID c = new ClientID();
     boolean done = false;
     while (!done) {
            int eventType = parser.next();
       if (eventType == XmlPullParser.START_TAG) {
         if (parser.getName().equals("token"))
         {
                                    c.setToken(parser.nextText());
             done = true;
         }
       }
     }
     return c;
   ]
```

Listing 3

```
package org.jivesoftware.smackx.packet;

import org.jivesoftware.smack.packet.PacketExtension;
import org.jivesoftware.smack.provider.PacketExtensionProvider;
import org.xmlpull.v1.XmlPullParser;

public class SOWExtension implements PacketExtension {
   private String Url;
   public static String namespace = "http://www.indent.org";
   public static String elementname = "xsow";

   public SOWExtension() {
            super();
            // TODO Auto-generated constructor stub
   }

   public void setUrl(String Url)           { this.Url = Url;     }
   public String getUrl() {      return Url; }
   public String getNamespace() {           return "http://www.indent.
org";      }
   public String getElementName() { return "xsow";     }
   public String toXML()
   {
StringBuffer buf = new StringBuffer();
buf.append("<").append(getElementName()).append("xmlns=\"").
append(getNamespace()).
   append("\">");
    if (getUrl() != null)
      buf.append("<conf Url=\"").append(getUrl()).append("\"/>");
    buf.append("</").append(getElementName()).append(">");
    return buf.toString();
   }

  public static class Provider implements PacketExtensionProvider {
   public Provider() {        }
            public PacketExtension parseExtension(XmlPullParser parser)
throws Exception {
                        SOWExtension sowExtn = new SOWExtension();
     boolean done = false;
     while (!done) {
            int eventType = parser.next();
       if (eventType == XmlPullParser.START_TAG) {
         if (parser.getName().equals("conf")) {
                        sowExtn.setUrl(parser.getAttributeValue("",
"Url"));
                        done = true;
         }
       }
     }
     return sowExtn;
   ]
```

Welcome to the Future of Video on the Web!

REGISTER NOW!
www.iTVcon.com

CALL FOR PAPERS NOW OPEN!

LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

iTVCON.COM
INTERNET TV CONFERENCE & EXPO 2006

Coming in 2006 to New York City!

"Internet TV is wide open, it is global, and in true 'Web 2.0' spirit it is a direct-to-consumer opportunity!"

For More Information, Call 201-802-3023
or Email itvcon@sys-con.com

Welcome to the Future!

Did you already purchase your ".tv" domain name?
You can't afford not to add Internet TV to your Website in 2006!

2005 was the year of streaming video and the birth of **Internet TV**, the long-awaited convergence of television and the Internet. Now that broadband is available to more than 100 million households worldwide, every corporate Website and every media company must now provide video content to remain competitive, not to mention live and interactive video Webinars and on-demand Webcasts.

20 years ago the advent of desktop publishing tools opened the doors for the creation of some of today's well-known traditional print media companies as well as revolutionized corporate print communications. Today, with maturing digital video production, the advent of fully featured PVRs, and significant advances in streaming video technologies, **Internet TV** is here to stay and grow and will be a critical part of every Website and every business in the years to come.

It will also very rapidly become a huge challenge to network and cable television stations: **Internet TV** is about to change forever the $300BN television industry, too.

The Internet killed most of print media (even though many publishers don't realize it yet), Google killed traditional advertising models, and **Internet TV** will revolutionize television the way we watch it today. You need to be part of this change!

Jeremy Geelan

Jeremy Geelan
Conference Chair, iTVCon.com
jeremy@sys-con.com

PRODUCED BY SYS-CON EVENTS

List of Topics:

- > Advertising Models for Video-on-demand (VOD)
- > Internet TV Commercials
- > Mastering Adobe Flash Video
- > How to Harness Open Media Formats (DVB, etc)
- > Multicasting
- > Extending Internet TV to Windows CE-based Devices
- > Live Polling During Webcasts
- > Video Press Releases
- > Pay-Per-View
- > Screencasting
- > Video Search & Search Optimization
- > Syndication of Video Assets
- > V-Blogs & Videoblogging
- > Choosing Your PVR
- > Product Placement in Video Content
- > UK Perspective: BBC's "Dirac Project"
- > Case Study: SuperSun, Hong Kong

Track 1	Corporate marketing, advertising, product and brand managers
Track 2	Software programmers, developers, Website owners and operators
Track 3	Advertising agencies, advertisers and video content producers
Track 4	Print and online content providers, representatives from traditional media companies, print and online magazine and newspaper publishers, network and cable television business managers

COPYRIGHT ©2006 SYS-CON MEDIA ALL RIGHTS RESERVED

Rich Internet Applications: AJAX,

www.AjaxWorldExpo.com

AJAXWORLD™ CONFERENCE & EXPO

SANTA CLARA SILICON VALLEY

SYS-CON Events is proud to announce the first-ever AjaxWorld Conference & Expo 2006!

The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this uniquely timely conference – especially the web designers and developers building those experiences, and those who manage them.

CALL FOR PAPERS NOW OPEN!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested learn about a wide range of RIA topics that can help them achieve business value.

©COPYRIGHT 2006 SYS-CON MEDIA. ALL RIGHTS RESERVED

Flash, Web 2.0 and Beyond...

REGISTER TODAY AND SAVE!

October 3-4, 2006

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

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

May 7-8, 2007
First International AjaxWorld Europe
Amsterdam, Netherlands

"Over the two information-packed days, delegates will receive four days' worth of education!"

Early Bird*
(Register Before August 31, 2006)
.... $1,495**
See website or call for group discounts

Special Discounts*
(Register a Second Person)
.... $1,395**
See website or call for group discounts

(5 Delegates from same Company)
.... $1,295/ea.**
See website or call for group discounts

On-Demand Online Access
(Any Event)
.... $695

*Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

**OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

"It Was The Best AJAX Education Opportunity Anywhere in the World!" —John Hamilton

Topics Include...

Themes:
> Improving Web-based Customer
> Interaction
> AJAX for the Enterprise
> RIA Best Practices
> Web 2.0 – Why Does It Matter?
> Emerging Standards
> Open Source RIA Libraries
> Leveraging Streaming Video

Technologies:
> AJAX
> The Flash Platform
> The Flex 2 Framework & Flex Builder 2
> Microsoft's approaches: ASP.NET, Atlas, XAML with Avalon
> JackBe, openLaszlo
> JavaServer Faces and AJAX
> Nexaweb
> TIBCO General Interface

Verticals:
> Education
> Transport
> Retail
> Entertainment
> Financial Sector
> Homeland Security

GROUP DISCOUNTS AVAILABLE:
— 5 Delegates from Same Company — for only $995 (each)
— Register a Second Person — for only $1195

LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

Receive FREE WebCast Archives of Entire Conference!

The best news for this year's conference delegates is that your "Golden Pass" registration now gives you full access to all conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.

This on-demand archives set is sold separately for $995

SYS-CON EVENTS
For more great events visit www.EVENTS.SYS-CON.com
VISIT WWW.AJAXWORLDEXPO.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

AccuRev 4.0 Facilitates Effective Team Software Development

Isolating and versioning your code changes until you are ready to integrate them

Reviewed by
Michael Sayko

Sharp tools make software development quicker and more productive. They automate manual tasks to speed development. They provide useful information intuitively to enhance productivity. Eclipse, the powerful and well-designed IDE, is such a tool. Few Java developers would want to return to a text editor, command-line compiler, and stand-alone debugger after experiencing the power of this integrated development environment. In the same way that an intuitive and elegant IDE aids modern software development, a cutting-edge yet robust software configuration management (SCM) tool enables successful team software development.

AccuRev 4.0 is the latest release of the innovative SCM tool that fosters effective team software development. AccuRev's strength lies in its straightforward client/server architecture, transaction-based and append-only data repository, and stream-based model for managing software configurations. Of these strengths, the powerful stream-based model is what distinguishes it from other SCM tools. It's also the feature that can have the greatest impact on developers and on the productivity of the development team.

Installing AccuRev 4.0

Installing AccuRev 4.0 is quick and simple. In fact, it's as easy to install as Perforce, the popular SCM tool known for its dependable performance and minimal administration. For this evaluation, I installed both the AccuRev 4.0.1 server and client on a 1.2GHz Wintel notebook running Windows XP Professional, Service Pack 2. Installing the server and client only took a few minutes. No separate or involved setup was needed for the AccuRev data repository that maintains the files under version control.

Preparing To Use AccuRev 4.0

After installing AccuRev 4.0, the first step in using the tool successfully is to read the documentation. Surprisingly, this is not painful. The Concepts Manual, the Day-to-Day Usage of AccuRev section in the User's Guide, and the Administrator's Guide contain the information essential to start using AccuRev.

The Concepts Manual describes the approach that AccuRev uses to support the effective version management of files in a team environment. The Day-to-Day Usage of AccuRev section in the User's Guide connects the theory in the Concepts Manual with the commands that make AccuRev work. The Administrator's Guide explains the steps that an administrator has to take to ensure the integrity of the data repository and the reliability of the AccuRev server.

What impressed me most about the documentation is the understanding it gave me about how and why AccuRev (the company) designed and developed AccuRev (the tool). While reading the documents, I felt like I was listening to a comprehensive technical overview by AccuRev's designers and developers.

Understanding AccuRev's Approach for Saving Changes and Releasing Changes

Team software development is characterized by several developers making frequent changes to source code files at the same time. Disciplined developers want to save their changes frequently to prevent the loss of their work and to support the roll back of any file to a known state. However, they don't always want to share their changes because the modified code may not be stable. This scenario can pose a problem for a configuration management tool. How do you let developers version files frequently without requiring them to make each version available to other developers?

Michael Sayko is a software configuration management consultant based in Austin, Texas. He is experienced with the practice of software configuration management from having served as a configuration manager on large, fast-paced software projects. Michael helps software development organizations practice software configuration management by applying SCM patterns to solve real world problems.

mss@acm.org

JDJ Product Snapshot

Target Audience: All members of the software development team including developers, build and release engineers, and project managers.

Level: All levels from beginner to expert.

Pros:
- AccuRev's elegant yet straightforward stream-based model is ideally suited for team software development, especially development that's highly parallel.
- AccuRev is well designed. Its software architecture is clean and modern.
- AccuRev is easy to use and easy to administer. A team of highly specialized tool administrators isn't needed to keep the tool running.

Cons:
- Since AccuRev follows a best-of-breed strategy, the product isn't a standalone application lifecycle management (ALM) solution. However, AccuRev provides an ALM solution when it's linked to third-party products using the AccuBridge SDK.

Platforms: Intel and AMD x86-based systems:
- Windows XP, 2003, 2000, NT 4.0
- Windows 98/ME (client-only)
- Linux (kernel versions 2.4.9+, Red Hat AS/ES 2.1+)
- FreeBSD (version 3.3+)
- Sun Solaris and Solaris x86 (version 2.5.1+, including Solaris 10)
- Apple OS/X+ (client-only)
- HP-UX (version 11.0 +)
- IBM AIX (RS/6000) (version 4.3.2+)
- SGI Irix (version 6.2+)
- Alpha Compaq Tru64 Unix (version 4.0+)

Pricing:
- *AccuRev Enterprise* - $1,495 per seat
- *AccuRev Professional* - $795 per seat

Using rigid check-in policies with a traditional SCM tool can lead to infrequent versioning because developers delay each check-in until the changed files are stable enough to release to their colleagues. In contrast, following lax check-in policies with a traditional SCM tool can lead to the propagation of unstable code because each check-in may not be reliable enough to share with everybody. Ideally, developers should be able to save (i.e., to version) their changes as frequently as needed, but only release stable changes to other developers.

Java developers familiar with IBM's VisualAge for Java (VAJ), the precursor to its WebSphere Studio Application Developer (WSAD), may recall a built-in configuration management tool called ENVY. By separating the versioning mechanism from the release mechanism, ENVY let developers version frequently but only share the version that was stable enough to be used (and modified) by others.

Like ENVY, AccuRev has a simple yet elegant model that distinguishes saving changes from making these changes public. The model used by AccuRev is based on first-class objects called streams. AccuRev (the company) defines a stream as "a configuration of a collection of version-controlled files." This means a stream is like a list of the version numbers of the files that form a collection. The collection can be large enough to contain the version numbers of all the files in a software release or small enough to contain only the version numbers of the files modified by a developer to fix a bug.

Streams act as public data areas. Typically, they're organized into a hierarchy to mimic the way teams develop software. Streams at the top of the hierarchy contain the versions of files in a software release. Workspaces at the bottom of the hierarchy contain the versions of files changed in each developer's local development environment. Developers promote their changes up the stream hierarchy when they're ready to integrate them with others. Developers update their workspaces from changes that propagate down the stream hierarchy. Figure 1 shows a typical stream hierarchy in AccuRev's StreamBrowser.

Using AccuRev 4.0 To Manage Changes to Files

Developers use four AccuRev commands, Keep, Promote, Update, and Merge, to version file changes and integrate their file changes with changes made by other team members. For this evaluation, I executed these commands from the AccuRev GUI client. These commands are also available in the AccuRev command-line client.

Using an IDE or text editor, a developer modifies and saves one or more files while working on a development task. Since these files reside in a private workspace, the developer uses the Keep command to copy the modified files to an AccuRev data repository called a *depot*. The Keep command creates a new version of each modified file, but it doesn't release these versions to an integration environment used by other developers.

Eventually the changes made to files in the developer's workspace are stable enough to be shared with others. The developer then uses the Promote command to make the saved versions public. The Promote command makes the changes public by propagating them from the developer's workspace to the *backing stream*, the public data area connected to the developer's workspace.

The developer uses the Update command to stay current with the changes released by other developers. The Update command refreshes the developer's private workspace with changes from its backing stream.

There are times when the Promote and Update commands won't make changes because the file to promote or update was modified in the backing stream. When this happens, the developer uses the Merge command to combine the change in the developer's

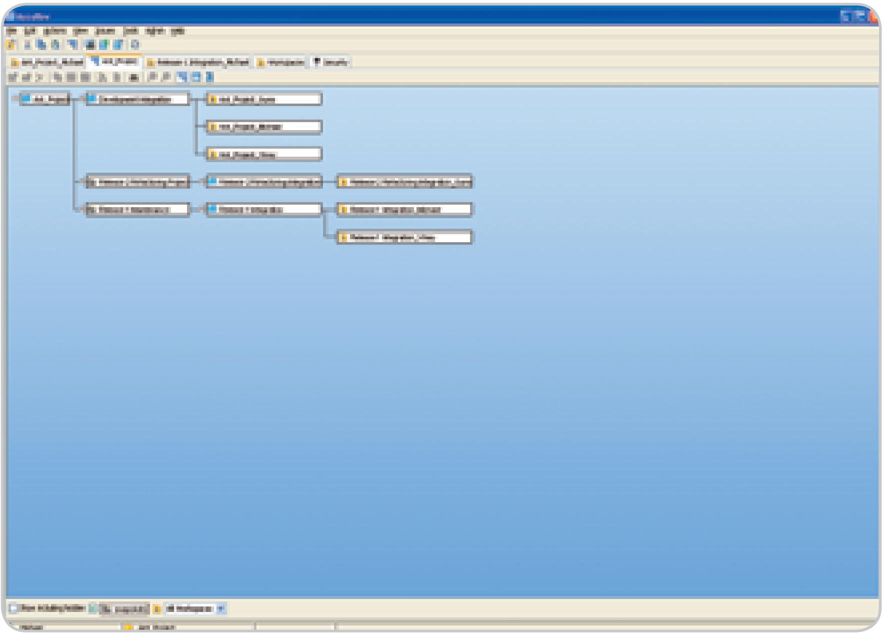


Figure 1 The StreamBrowser shows the stream hierarchy for a project in the depot

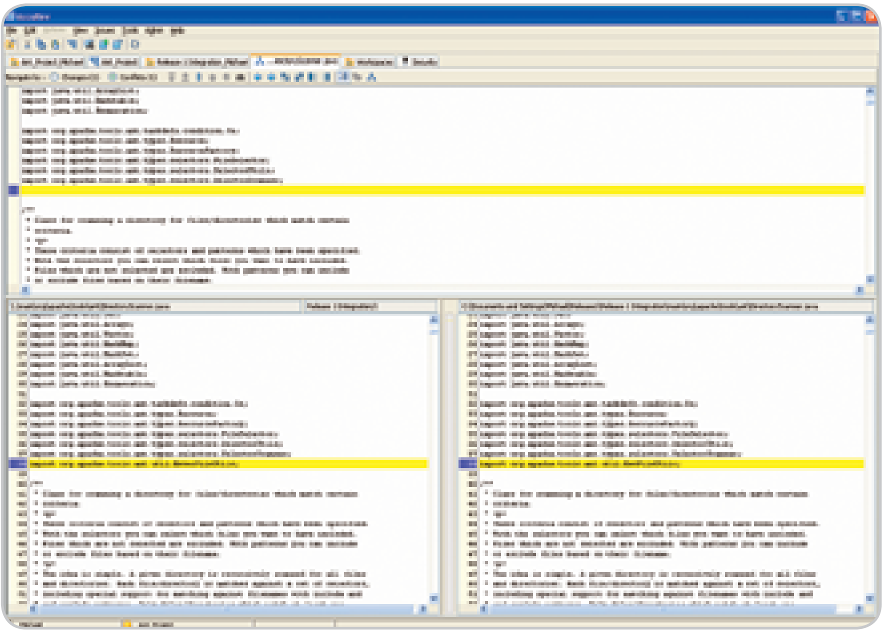


Figure 2 The Merge command opens a three-way merge tool to resolve a conflict between two developers

private workspace with the change in the backing stream. The Merge command opens a three-way merge tool (shown in Figure 2) that lets the developer combine the changes made to the two variants of the file. AccuRev automatically issues a Keep command when the developer exits from the merge tool.

Together, the Keep, Promote, Update, and Merge commands nicely implement the promotion model used by AccuRev. Alternatively, AccuRev supports file locking for serial development. However, using AccuRev for serial development circumvents the tool's conceptually clean and easy-to-use promotion model.

Using Streams to Support Concurrent and Parallel Software Development

Concurrent changes to the same files are likely when a team develops a software system. Concurrent changes are almost inevitable when one or more teams develop or maintain parallel releases of a software system.

I used branches, created from labeled configurations in a traditional SCM tool, to support concurrent and parallel development. Branching from a labeled configuration at the project-level is most meaningful because it lets changes be viewed from the perspective of the entire software system. Although many traditional SCM tools support project-oriented branching, it's still implemented using metadata at the file-level. Using file-level metadata makes branching resource intensive, particularly for branches that contain several thousand files. For this reason, adept development teams use branches sparingly.

Streams in AccuRev are conceptually equivalent to project-oriented branches. However AccuRev implements streams more efficiently than traditional SCM tools implement project-oriented branches. Traditional SCM tools store file version and branch information in the same data structure. In contrast AccuRev uses one data structure to model file versions and another data structure to model streams.

Streams in AccuRev can be nested in a hierarchy to model the promotion of code from developers' private workspaces to integration environments to test environments. The hierarchy can also be changed as needed. For example, an administrator can use the AccuRev GUI client to re-parent a developer's private workspace or an integration stream from one backing stream to another.

Using Issues to Support Task-Based Development

Modern software development entails more than following a code-compile-test-debug cycle. No matter what kind of software development methodology you use, from the rigid waterfall model to an adaptive, iterative model, you make code changes to satisfy a development task. The task is typically either a new enhancement or a bug fix.

A software configuration management tool supports your development approach by letting you identify development tasks and tie each new version of a file to a development task. *Task-based development* is the term used to describe developing a software system by completing a series of tasks.

AccuRev enables task-based development through change packages in its issue management facility, AccuWork. AccuRev users create issues to identify development tasks to be completed, such as new enhancements and bug fixes. As developers work, they associate file changes with an issue. The file versions are recorded in a change package that's displayed on the Changes tab of an issue record. Change packages are available only in the AccuRev Enterprise product. Figure 3 shows the change package for an issue.

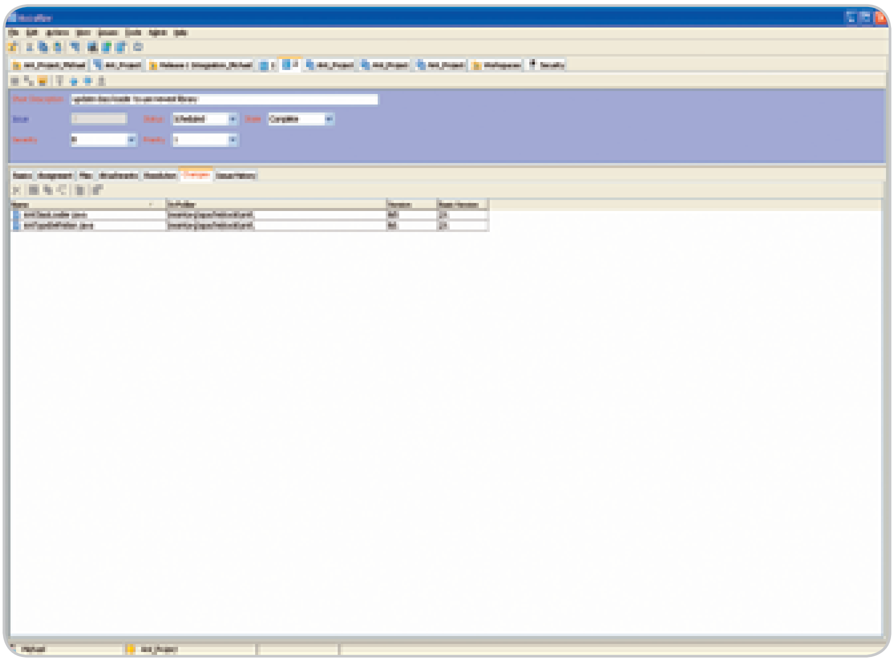


Figure 3 An issue record in AccuRev Enterprise contains a change package that associates file versions with an issue

Limitations

AccuRev 4.0's main limitation is that it's not a standalone application lifecycle management (ALM) solution. Competing SCM tools integrate version control with sophisticated change management capabilities in a single product. For example, SCM tools marketed as an ALM solution use process items to automate software development (or business process) workflow. While AccuRev lacks a built-in workflow modeling tool it does provide an API called AccuBridge that integrates with other issue tracking and workflow automation products, including Serena TeamTrack, MKS Integrity Manager, Rational ClearQuest Atlassian JIRA, and Bugzilla. AccuRev (the company) is also a member of the Eclipse Application Lifecycle Framework (ALF) project, which is working on developing an interoperability framework that supports the logical exchange of information between development tools.

Conclusion

AccuRev is a powerful software configuration management tool that employs an elegant model to support team software development, particularly team development that is highly parallel and geographically distributed. AccuRev's conceptual model is extremely well suited for facilitating team development because it lets developers save changes as needed, but only release changes when they're ready. Its stream hierarchy supports the efficient integration of changes across a development team.

introductory
subscription offer!

.NET

A TRULY INDEPENDENT VOICE IN THE WORLD OF .NET

.NET Developer's Journal is the leading independent monthly publication targeted at .NET developers, particularly advanced developers. It brings .NET developers everything they need to know in order to create great software.

Published monthly, ***.NET Developer's Journal*** covers everything of interest to developers working with Microsoft .NET technologies – all from a completely independent and nonbiased perspective. Articles are carefully selected for their prime technical content – technical details aren't watered down with lots of needless opinion and commentary. Apart from the technical content, expert analysts and software industry commentators keep developers and their managers abreast of the business forces influencing .NET's rapid development.

Wholly independent of both Microsoft Corporation and the other main players now shaping the course of .NET and Web services, ***.NET Developer's Journal*** represents a **constant, neutral, expert voice** on the state of .NET today – the good, the bad, and the ugly…no exceptions.

SYS-CON MEDIA

SUBSCRIBE ONLINE!
www.sys-con.com/dotnet/
or Call
1 888 303-5282

Here's what you'll find in every issue of .netdj:

- Security Watch
- Mobile .NET
- .NET Trends
- Tech Tips
- Standards Watch
- Business Alerts
- .NET News
- Book and Software Announcements

.NET Developer's Journal is for .NET developers of all levels, especially those "in the trenches" creating .NET code on a daily basis:

- For beginners:
  Each issue contains step-by-step tutorials.
- For intermediate developers:
  There are more advanced articles.
- For advanced .NET developers:
  In-depth technical articles and columns written by acknowledged .NET experts.

Regardless of their experience level, *.NET Developer's Journal* assumes that everyone reading it shares a common desire to understand as much about .NET – and the business forces shaping it – as possible. Our aim is to help bring our reader-developers closer and closer to that goal with each and every new issue!

SAVE 16% OFF

THE ANNUAL COVER PRICE

Get 12 issues of .NETDJ for only $69.99!

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

ANNUAL COVER PRICE:
~~$83.88~~
YOU PAY
$69.99
YOU SAVE
$13.89
OFF THE ANNUAL COVER PRICE

Fiorano SOA 2006 Platform

A honey of a product

Reviewed by
Warren Hampton

SOA, EDA, BCM, ESB and BPEL...More than IT Catch Phrases?

I recently had the chance to evaluate the next-generation Fiorano SOA Platform 2006 suite from Fiorano Software, Inc. As an architect and developer who's worked with previous versions of the kit over the last three years in addition to several competitor offerings, I looked forward to sitting down with Fiorano's latest release.

For those unfamiliar with the product, it's a feature-rich SOA/BPM development, deployment, and administration suite built on the company's J2EE ESB technology. It relies on standards-based services, shared component modeling, peer-to-peer communications, and high-performance event-driven messaging (pub/sub and queuing) across a distributed network. This latest release brings a standalone BPEL Editor, Composite Components, pre-built JCA-compliant adapters, a Business Development Component Kit and Shared Resource Pools among other enhancements (see Figure 1).

With intuitive interfaces, pre-built components, and data adapters, business analysts can build business process workflow applications with little or no programming. These components aren't graphical representations for diagram purposes; they are fully functioning feature-rich services that can be added to the application with drag-and-drop ease and are configured using simple parameters.

This is not to say that developers can't build complex coarse-grained custom services to leverage the full power of the suite. The extensive toolset has allowed our developers to concentrate on extending and refining richer applications to meet business requirements in a fraction of the time of competitors' offerings while encouraging code re-use, SOA design, and solidifying patterns and practices. Another key point is the platform's low overhead, full scalability, unassisted fail-over, and flexibility to run on a single server or across a highly distributed WAN environment or load-balancing cluster with ease.

Getting Started

First note the decision to rename the suite from Fiorano ESB to Fiorano SOA. This makes perfect sense. I've always maintained that previous versions were more than a toolset

Fiorano Software, Inc.

718 University Avenue, Suite 212
Los Gatos, CA 95032
Web: www.fiorano.com
Phone: 800 663-3621

for building ESB-based solutions. Although Fiorano originated in high-performance message queuing, the platform has matured into a powerful, scalable service-oriented development platform on top of the proven messaging abilities. Earlier versions allowed for rapidly building robust workflows and quick integrations across disparate systems and relied on intuitive administration interfaces for views into daily development and production processes. I was eager to see what the latest iteration would add.

The installation is simple, a few point-and-clicks is all you need to get the full Enterprise version installed and ready to roll. Although the suite is intuitive I'd recommend browsing the Startup and the Platform Concepts guides before beginning the install to familiarize yourself with the primary components and the depth of the product. You can find extensive documentation and code samples at http://devzone.fiorano.com/dev-zone/dev_zone.jsp.

An improvement over previous versions is the built-in ability to launch both the servers and individual processes as Windows Services. This

Warren Hampton is a senior e-commerce architect with Quicken Loans/Title Source Inc.

"Note the decision to rename the suite from Fiorano ESB to Fiorano SOA; this makes perfect sense; I've always maintained that older versions were more than a toolset for building ESB-based solutions" allows for auto-restarting services and applications as well as monitoring the services using commonly available network monitoring tools. I would, however, have liked to see this as part of the initial installation. The rules-based security is easily understood, set up, and maintained but also robust and flexible enough to pass stringent security standards. Integration with LDAP services is also an option.

Once installed you'll find the following:

- Fiorano ESB Server 2006 – A web Services-capable middleware platform
- FioranoMQ Server 2006 – Peer-to-peer JMS messaging platform
- Fiorano BPEL Server 2006 – A distributed BPEL processing orchestration engine.
- Fiorano Business Components and Adapters 2006 – Ready-to-use JCA-compliant components
- Fiorano Process Orchestration Tools 2006 – Integrated development and admin toolset
- Fiorano BPEL Editor 2006 – A standalone BPEL design, development, test, and deployment tool set

SOA, EDA, and ESB in the Real World

Although real-world sample applications are included I wanted to see how I could improve existing workflows created in previous versions as well as solutions built with other tools to see if there were some viable gains in this "major" release to further simplify the development process and make it easier to build, track, debug, extend, and monitor these solutions. I choose more complex workflows that get XML messages from external sources via HTTPS posts or Web Services, rely on content-based XPATH queries for routing and XSLT transforms from external to internal formats, create physical archive files, update an SQL DB, and again transform messages to one of many positional or delimited file formats for delivery to legacy systems. Also included along the way are extensive error-handling alerts and simple business rules.

What I found was very encouraging. As previously mentioned the suite includes an Event Process Orchestrator to build flows. I was glad to see that this had retained its look-and-feel while the pre-built services have improved in many key areas in regards to continuity and configurable parameters. Also apparent was an improvement in speed when opening the configuration dialog screens. I was quickly able to build new loosely coupled versions of the existing applications (including non-Fiorano solutions) leveraging many of the enhancements such as improved services, data adapters, composite components, and shared resource pools to create highly efficient applications.

Several new or enhanced pre-built services and data adapters are offered, some of which include BeanShell script components, improved DB, HTTP, Web Service, and transform components as well as support for adding iWay JCA-compliant services to the flows (http://www.iwaysoftware.com/). These, added on top of the existing extensive palate of services, protocols, and adapters, (60+) went a long way to letting me build richer enterprise-level solutions with little or no custom programming. Not that you're limited in this area; improvements have been made making it easier than ever to build custom components that can be added to the existing palette using Java, C, C++, and C# as required.

One of the main improvements is the enhanced ability to build composite components that are repeated many times in day-to-day workflows and integrations. Easily and effectively I re-created large pieces of a complex workflow that can be called on from multiple applications. This is a great time-saver and helps you adhere to code re-use principles and enforce standard handling of repeated processes. Another welcomed enhancement is the improvements to what was arguably the best mapping tool available. FioranoSOA 2006 now retains existing mappings when adding to the underlying schema. This is a real time-saver when you consider the implications of re-mapping an extensive transformation from scratch to add an additional tag to a complex schema.

Business Process Meets Application Workflow...

One of the most striking things I always found with the platform was its ability to bridge the divide between business process and application workflow modeling. This is even truer today using a combination of the BPEL Studio, the Event Process Orchestrator, and an extensive palette of standards-based, pre-built functional services; everyone from business analysts to developers and administrators can conceptualize workflows, build applications, test, and monitor the full development process from a common interface without extensive knowledge of the underlying services.

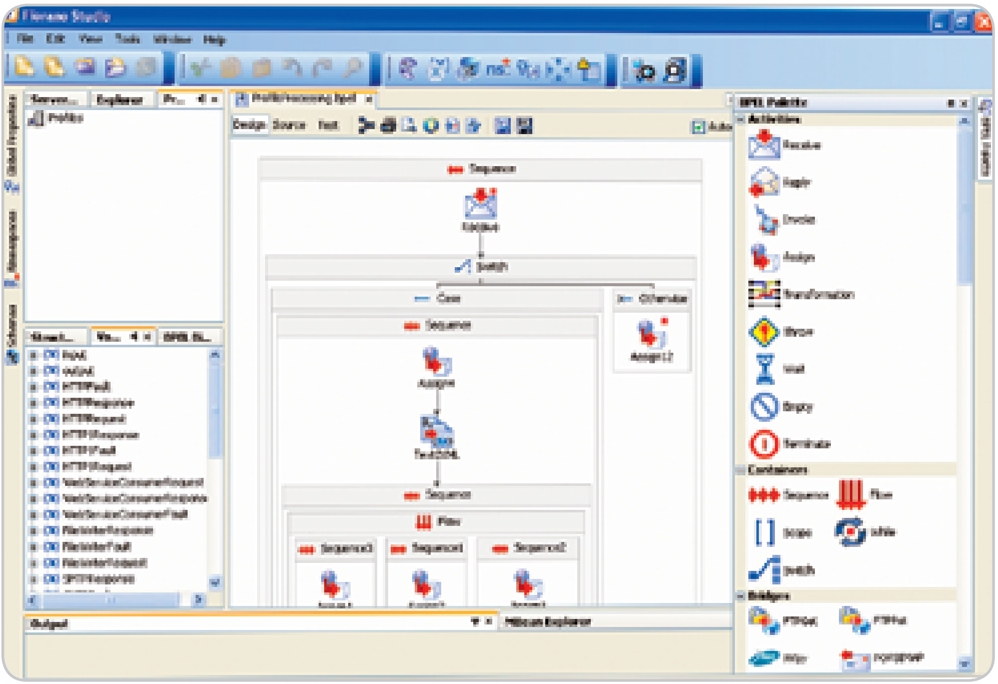


Figure 1

The ease with which you can create and deploy working solutions must be seen to be appreciated. Many times I have built a working solution before someone's eyes and deployed it only to be asked, "Okay, so now what are the steps required to build, code, and deploy the application?". By immediately running the application and showing the messages flowing is it clear a fully functioning flow can be created almost as quickly as you visualize it. Compared to competitor offerings FioranoSOA excels in this area.

Production applications can even be extended, debugged, or corrected without downtime to the application, something unique to the platform and eye-opening to those accustomed to late-night builds, pushes, and testing. Application profiles help control versioning between development phases and push-to-production environments. The platform challenges the traditional development process methodology, saving significant time and money when a production issue has to be corrected. But it still fits perfectly into the standard process and eases many of these tasks compared to alternative methods.

The speed with which you can do change requests to a business process is where the application and toolsets really shine. I have faced major production issues after a deployment in which the ESB platform had no direct affect, acting largely as a transport/transformation layer. However, due to the power and flexibility of the platform I was able to correct the issue by applying business rules that altered the messages in the production application in less than one hour, whereas the legacy system sending the data in question would have taken four to six hours of development, testing, and a late-night build to resolve the issue by the next business day. Starting and stopping services at runtime, components like the Feeder Service that can send messages to the orchestration or the display service that shows you the output from any service port help to speed debugging and re-queuing from any point in the process. The ability to capture messages via event interceptors on the routes between services empowers you to queue messages, edit services, and analyze log files without disrupting other services in the flow.

The cost savings in these scenarios is hard to quantify since it's very far-reaching in a high-volume transaction-based business that directly affects revenues based on these messages; needlesstosay it's significant. The savings starts at the development and testing stage where the cost savings created by the speed with which adjustments can be made carries all the way through ongoing maintenance of the platform and solutions in production. At each stage significant gains are realized when compared to competitive offerings or traditional coding methods.

This flexibility also allows for simple scaling of the platform. You can easily add more peer servers, fail-over solutions, load-balancing flows, and error handling without significant impact on your applications. You can go from test to beta to QA to production in a very short period of time and deploy these updates with confidence and minimal risk. The support included for resource sharing across multiple instances of components in composite applications reduces the memory load of intensive services such as DB connections across multiple components further extending scalability.

An often-overlooked strength is the power of centralized modeling, development, deployment, and administration. Through rules-based security, roles can be restricted as required allowing all disciplines to view, alter, or administer the applications, servers, real-time messages, services, and related components from a common set of tools as required. Being able to have the BA, developer, QA analyst, and administrator all look at the same visual interface with no risk to the application and work through testing, deployment, monitoring, and debugging is invaluable. The ROI continues well past the development stage of your applications. The flexibility of the platform also excels at extending the lifecycle of legacy applications by adapting to the older platforms, then allowing you to extend the legacy applications' abilities by integrating newer protocols, standards, and abilities such as HTTPS, XML Messaging, and Web Services.

Conclusion

With this release Fiorano has firmly established a mature development platform further increasing the ability of a business analyst to create application flows and transformations and freeing highly skilled developers to build richer solutions, custom services, and loosely coupled composite services. With minimal training the ROI can begin immediately and principles like RAD (Rapid Application Development), SOA, and code re-use become reality. The cost savings begins with the first stage of Business Process Modeling and continues all the way through day-to-day administration, extending the lifecycle of existing platforms where applicable.

Also noted are the improvements in performance and throughput in several key areas, enhanced simple and distributed transaction handling, improved ANT script support, and AXIS and Eclipse integration. On the horizon a native C# version of platform and enhanced Web Service support will further extend the scope of the platform. Consuming Web Services is a breeze but there's still work to do regarding incoming Web Services. Overall I was very impressed with the new release and have already realized immediate gains in time-to-production, quicker ROI, and richer applications.

"An improvement over previous versions is the built-in ability to launch both the servers and individual processes as Windows Services"

JAVA DEVELOPER'S JOURNAL Advertiser **Index**

Advertiser	URL	Phone	Page
.NET Developer's Journal	www.sys-con.com/dotnet	888-303-5282	55
AJAX Seminar	www.ajaxseminar.com	201-802-3022	50-51
Altova	www.altova.com	978-816-1600	4
Backbase	www.backbase.com	866-800-8996	35
Enerjy	www.enerjy.com/visible	866-598-9876	27
Fiorano	www.fiorano.com/downloadsoa	800-663-3621	17
IBM	ibm.com/takebackcontrol/flexible		8-9
Instantiations	www.instantiations.com/rcpdeveloper	800-808-3737	7
iTVcon.com Conference & Expo	www.itvcon.com	201-802-3023	49
IT Solutions Guide	www.itsolutions.sys-con.com	888-303-5282	59
JadeLiquid Software	www.webrenderer.com	+61-3-6226-6274	21
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	43
Northwoods Software Corp.	www.nwoods.com	800-434-9820	45
ODTUG	www.odtug.com		41
OpenAccess Software, Inc.	www.openaccesssoftware.com/jdj	888-805-ODBC	13
OPNET Technologies, Inc.	www.opnet.com/pinpoint	240-497-3000	15
Parasoft Corporation	www.parasoft.com/jdjmagazine	888-305-0041	Cover IV
Quest Software	www.quest.com/javacode	949-754-8000	Cover II
Roaring Penguin	www.roaringpenguin.com	613-231-6599	25
Software FX	www.softwarefx.com	800-392-4278	Cover III
SUN Microsystems	www.netbeans.org		31
SYS-CON Website	www.sys-con.com	888-303-5282	61
TIBCO Software Inc.	www.tibco.com/mk/gi	800-420-8450	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...

Linux Java Web Services .NET XML Wireless Storage Security
IT solutions
GUIDE
www.sys-con.com/IT
Fresh IT Perspectives
THE RIGHT TECHNOLOGIES AND RESOURCES FOR YOUR VARIED BUSINESS NEEDS
Top IT Solution Providers!
ArchMind
Azul
DocSoft
ElasticPath
Extentech
Hosting.com
MapInfo
Parasoft
Windward
Xenos
Xythos

*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!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE.

SYS-CON MEDIA

The World's Leading i-Technology Publisher

GWT: The Most Important Announcement at JavaOne?

by Rick Hightower

Time is a brutal enemy of youth and exuberance. Time makes cynics of us all. Time is the universal truth serum that reveals all authenticity. Time will tell, but the announcement at JavaOne 2006 by Google may change the face of AJAX development; strike that, Google's announcement may change Web development forevermore.

This cynic heard an announcement at JavaOne that changed his viewpoint and beliefs on the future of Web development.

Certainly, in the recent past, the chances of doing an entire application in AJAX seemed remote for the vast sea of developers. The thought of writing a rich application in JavaScript, for most developers, is total anathema – akin to having one's body shaved and thrust into a pool of warm alcohol.

Please don't write and plead for a change in my aversion to writing metric tons of JavaScript, for my heart is not in it, as most developers' hearts are not in it. It is not so much the writing of JavaScript as it is the lack of tools and the horror of debugging it. Sure tools exist, but they are a far cry from what you get in the Java world.

Writing JavaScript is like changing a baby's diaper. You don't like to do it, but you love your child and do it anyway. You may not like to write JavaScript, but you want to deliver richer applications to your end users, so you do it anyway. Dare I say the best developers are the ones who love their end users?

Thus the missing ingredient is the ability of Java developers to develop AJAX applications in Java instead of JavaScript, i.e., to take the smell and rank out of it. This would allow a vast community of

developers to develop rich Web applications where before only a select few script-heads would dare to go.

Enter stage left, the contender for changing, once and for all, the way the world uses the Web: Google!

Google introduced the Google Web Toolkit (GWT), a free, publicly available Java development framework. This framework allows developers to develop and debug applications in Java and deploy them in AJAX. The Google approach to AJAX development is to avoid JavaScript (for most developers anyway).

You can write all of your AJAX code in plain old Java. You can debug it. Use breakpoints. They have a plug-in where they allow your code to run in the browser and hook back to your Java code. Then, when your code is ready to deploy, you run the translator that converts your Java code into JavaScript code that can run on any browser, or so the vision states. They also have an RPC mechanism to call back to Java objects on the server for data and business rule validation. The Java code looks like AWT, Swing, or perhaps SWT code. In other words, it is what most rich GUI app developers are familiar with.

The framework is also extensible, so if your favorite Dojo JavaScript widgets don't exist, you can extend the framework to support them. Most developers won't have to do this, but you can. One of the Google examples is an Outlook clone. It doesn't look like a Web application. It looks like a rich application.

If it all seems to good to be true, you're right. However, if half of it is true, this changes everything. Cynicism is good. Without cynicism, you would be driven hither and thither, to and fro, back and forth with each new buzzword and vendor marketing claims. However, cynicism must be balanced with potentially the most disruptive technology.

The mantra at JavaOne seemed to be JSF, JSF, JSF; NetBeans, NetBeans, NetBeans; AJAX, AJAX, AJAX. Never mind that Eclipse is the dominant developer platform by far, and most vendors that have a plug-in seem to target Eclipse first. Nevermind Google released the GWT, which may prove disruptive to further JSF adoption for those who care about AJAX anyway.

Among the noise, there is this announcement from a company that started the AJAX phenomena and has the most popular AJAX applications. Is this announcement from Google the most important announcement for AJAX and the most important announcement at JavaOne? It's too soon to tell if this is the most important message at JavaOne 2006 due to how much reality and robustness is in the GWT, but the potential is there. Time will tell.

"Is this announcement from Google the most important announcement for AJAX and the most important announcement at JavaOne?"

Rick Hightower serves as chief technology officer for ArcMind Inc. He is coauthor of the popular book *Java Tools for Extreme Programming*, which covers applying XP to J2EE development, and also recently co-authored *Professional Struts*. He has been working with J2EE since the very early days and lately has been working mostly with Maven, Spring, JSF and Hibernate. Rick is a big JSF and Spring fan. Rick has taught several workshops and training courses involving the Spring framework as well as worked on several projects consulting, mentoring and developing with the Spring framework. His blog can be found at http://jroller.com/page/RickHigh

rick_m_hightower@hotmail.com

Visit the *New*

www.SYS-CON.com

Website Today!

The World's Leading *i*-Technology News and Information Source

24/7

FREE NEWSLETTERS
Stay ahead of the i-Technology curve with E-mail updates on what's happening in your industry

SYS-CON.TV
Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!
Read web logs from the movers and shakers or create your own blog to be read by millions

WEBCAST
Streaming video on today's i-Technology news, events, and webinars

EDUCATION
The world's leading online i-Technology university

RESEARCH
i-Technology data "and" analysis for business decision-makers

MAGAZINES
View the current issue and past archives of your favorite i-Technology journal

INTERNATIONAL SITES
Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

IT Solutions Guide
Information Storage+Security Journal
JDJ
Web Services Journal
.NET Developer's Journal
LinuxWorld Magazine
Linux Business News
Eclipse Developer's Journal
MX Developer's Journal
ColdFusion Developer's Journal
XML Journal
Wireless Business & Technology
Symbian Developer's Journal
WebSphere Journal
WLDJ
PowerBuilder Developer's Journal

SYS-CON MEDIA *The World's Leading i-Technology Publisher*

Meet the Winners of the 4th JCP Program Annual Awards

Onno Kluyt

Last month at the 2006 JavaOne Conference, the Java Community Process (JCP) Program was brought into the spotlight repeatedly when Sun Microsystems CEO Jonathan Schwartz and other speakers urged attendees to join the community. The JCP made center stage again on Wednesday night at the JCP Program Community Event when the winners of the 4th JCP Annual Awards were announced. If you're not familiar with the selection process for the JCP Annual Awards, you should know that the JCP Executive Committees' (EC) representatives first select nominees and then cast votes to choose the winners from among them. There are five categories in which contenders vie each year to make the top four or five: Member of the Year, Most Outstanding Spec Lead for Java Standard Edition/Enterprise Edition, Most Outstanding Spec Lead for Java Micro Edition, Most Innovative JSR for Java Standard Edition/Enterprise Edition, and Most Innovative JSR for Java Micro Edition. This year, after the nominations round, there were about three to four candidates for the winner title in each category – all very strong contenders. This made the task of the EC representatives quite difficult and to better understand how tough it was to decide, go to http://jcp.org/en/press/news/awards/2006award_nominees for a complete list of the nominees and descriptions of the awards categories.

If you missed the opportunity to meet the winners at "the JCP party," as the JavaOne Conference goers have come to dub the JCP Community Event, here's your second chance, minus the excellent food and the fine California wine and a hand shake with Duke. For the whole package, we invite you to next year's at the JavaOne Conference, May 8–11, 2007.

And the Winners Are!

JCP Member of the Year

Sony Ericsson made it to the top this year in the "Member of the Year" category. The JCP Executive Committees (EC) voted the company as one of this year's best examples of how standards development and innovation work together in advancing Java technology. Sony Ericsson has been a JCP member since the company formed in October 2001. Shortly after, it was elected to the EC and reelected in 2005 for a three-year term. Hanz Hager, who accepted the award for Sony Ericsson, represents the company on the Java Micro Edition (ME) EC. He has also served on the Expert Groups for Java Specification Request (JSR) 185 Java Technology for the Wireless Industry (JTWI), and JSR 248/249 Mobile Service Architecture (MSA). As Java product manager, Hanz is responsible for the Java technology strategy at Sony Ericsson.

Onno Kluyt is the director of the JCP Program Management Office, Sun Microsystems.

onno@jcp.org

Most Outstanding Spec Lead for Java ME

Spec Leads merit special recognition from the community. Key to the JCP Program in that they are the engines behind the development of the Java standards, Spec Leads take on challenging work that often goes beyond what they are paid to do in their day jobs. Asko Komsi of Nokia and Mark Duesner of Vodafone were nominated for their role in co-leading JSR 248 Mobile Service Architecture (MSA). Highly experienced in research, standardization, and strategy development for mobile technologies, Asko joined the JCP program in 2004. Currently director of industry relations at the Nokia Research Center, Asko is responsible for next generation, mobile Java technology standardization and related industry cooperation. Sharing the honors with him at the awards ceremony was Kay Glahn, from Vodafone Group Services Limited, who will tag team with Asko as co-spec lead to take the API to its next development level.

Most Outstanding Spec Lead Java SE/EE

The top honors in the "Most Outstanding Spec Lead Java SE/EE" category went to Linda DeMichiel. She accepted her award with gratitude toward the Enterprise JavaBeans (EJB) Expert Group, saying "I couldn't have done it without you."

Linda's experience with the JCP program goes back to 1999. She is a veteran Spec Lead, having guided the development of JSR 19 EJB 2.0, JSR 153 EJB 2.1, and JSR 220 EJB 3.0. She is a senior architect in the Java Enterprise Edition (EE) Platform group at Sun Microsystems and Sun's chief architect for Enterprise JavaBeans and the Java Persistence API. If you are about to become a JSR Spec Lead and want to talk to someone who's seen it all, done it all as a Spec Lead, Linda is your go-to expert.

Most Innovative JSR for Java ME

Innovation is the lifeblood of the JCP, and it occurs only because people make it happen. One such instance of innovation is JSR 272 Mobile Broadcast Service API for Handheld Terminals which the ECs' votes sent to the very top of the "Most Innovative JSR for JavaME" category this year. In accepting their award, the folks who made it happen, co-Spec Leads Antti Rantalahti of Nokia and Ivan Wong of Motorola, gratefully acknowledged their Expert Group of 22 members: "It's a group effort. Thanks to all who participated in it." Earlier, Antti had noted, "Because of the complexity of the JSR, the Expert Group members really bring value to the effort. No single company is the best Expert on all areas we have to cover."

The result of their innovative work on JSR 272 Mobile Broadcast Service API for Handheld Terminals is that someday soon, everybody will be able to catch a CNN report, a rerun of "Friends," or the local weather on their mobile devices.

Most Innovative JSR for Java SE/EE

When it's about innovation, the ECs are not shy to take a risk and pick as winner one of the newest submissions. JSR 292 Supporting Dynamically Typed Languages on the Java Platform has got JCP members excited. Spec Lead Gilad Bracha accepted the award, saying, "It's not my idea. It's 30 years old, and its time has come." Even so, it takes innovative effort to turn an idea into technological innovation and Gilad is at it now.

A "computational theologist" according to himself, Gilad is a distinguished engineer at Sun Microsystems who has been a Spec Lead for a large number of JSRs: 14, 65, 175, 201, 202, 292, and 294. He says that JSR 292 plans to extend sufficient support in the Java Virtual Machine (JVM) for implementers of dynamically typed languages such as Lisp, Python, Ruby, and Smalltalk to directly target the JVM's object model. This support will allow these languages to leverage the high-end performance capabilities of JVMs.

Congratulations to the winners and the runners-up for their great contributions to the JCP! For details regarding the standards they are developing and input you may want to provide, go to the respective JSR pages on http://jcp.org

Eclipse Data Visualization

(No Silly Glasses Required)

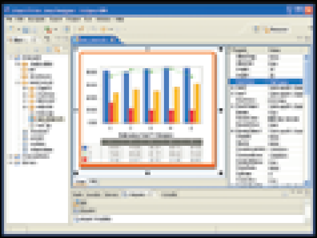


Chart FX for Java Eclipse plug-in.

The Leading Charting Solution Now Provides Powerful Data Visualization for Eclipse

The **Chart FX for Java 6.2 Eclipse plug-in** brings enterprise-level data visualization features to the Eclipse IDE. The Designer is integrated into the IDE allowing quick customization of the charts and the required code generation. In addition to a myriad of traditional chart types, the **Chart FX Maps** extension is included to create dynamic, data-driven image maps, such as geographic maps, seating charts or network diagrams, among others. Chart FX for Java 6.2 is available as a Server-side Bean that runs on most popular Java Application Servers. The 100% Java component produces charts in PNG, JPEG, SVG and FLASH formats. The **Chart FX Resource Center** integrates into the Eclipse Help and includes a Programmer's Guide, the Javadoc API and hundreds of samples. This makes Chart FX for Java the most feature-rich, easy-to-use charting tool available for Java development. *Learn more about the seamless integration and powerful features at www.softwarefx.com.*

Chart FX

www.softwarefx.com

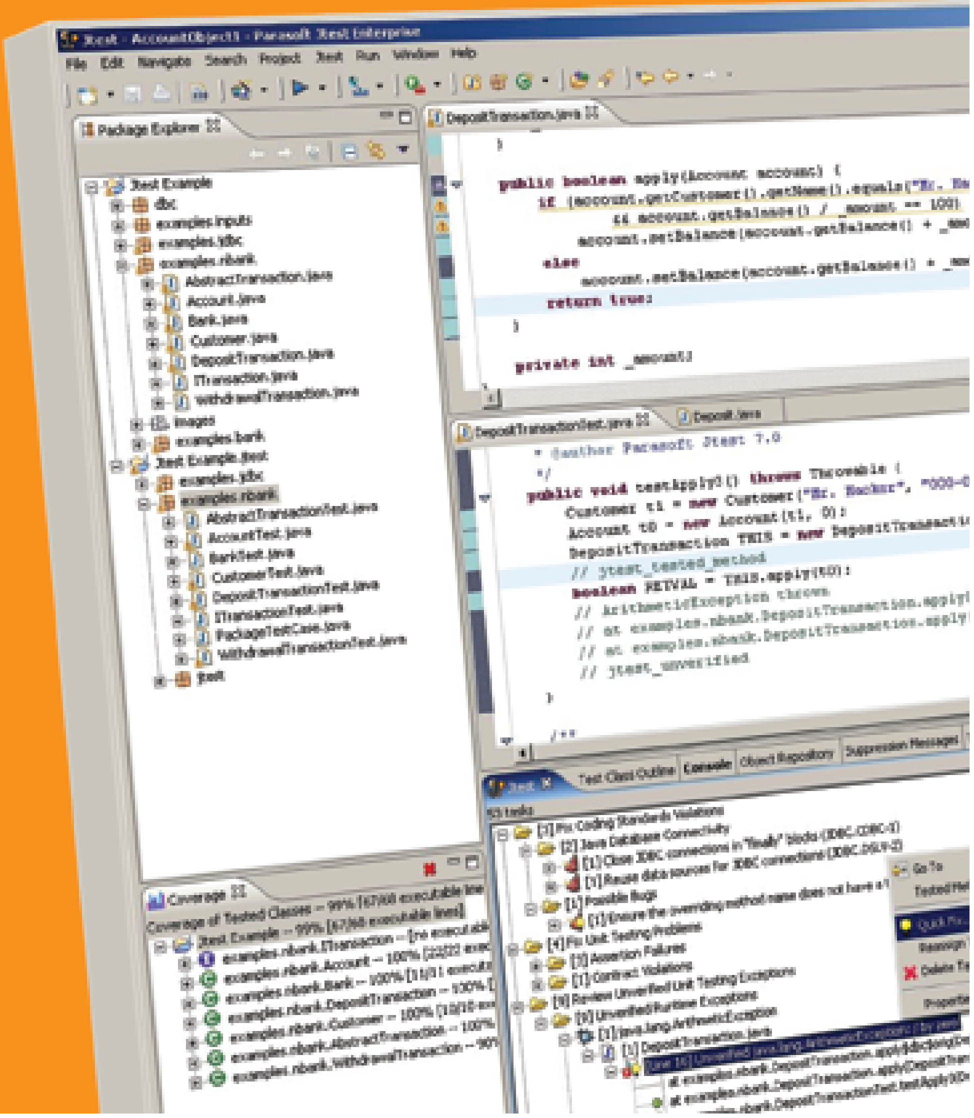
New! version 6.2

Now Includes Maps!

©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.

Code in quality. Test out bugs.

Take control of your code through Automated Unit Testing and Code Analysis...


Jtest - AccountObject1 - Parasoft Jtest Enterprise
File Edit Navigate Search Project Jtest Run Window Help
Package Explorer
DepositTransaction.java
DepositTransactionTest.java
Coverage
Test Class Outline
Console
Object Repository
Suppression Messages

```
}

public boolean apply(Account account) {
    if (account.getCustomer().getName().equals("Mr. Ba
        && account.getBalance() / _amount == 100)
        account.setBalance(account.getBalance() + _amo
    else
        account.setBalance(account.getBalance() + _am
    return true;
}

private int _amount;
```

```
 * @author Parasoft Jtest 7.0
 */
public void testApply0() throws Throwable {
    Customer t1 = new Customer("Mr. Bachor", "000-0
    Account t0 = new Account(t1, 0);
    DepositTransaction THIS = new DepositTransacti
    // jtest_tested_method
    boolean RETVAL = THIS.apply(t0);
    // ArithmeticException thrown
    // at examples.nbank.DepositTransaction.apply(
    // at examples.nbank.DepositTransaction.apply
    // jtest_unverified
}
```

PARASOFT®

Jtest®

- Automatically analyzes your code, applying 500+ Java coding best practices that surface poor code constructs affecting reliability, security and performance.
- Automatically unit tests your Java code evaluating code behavior and exposing unexpected exceptions, boundary condition errors and memory leaks.
- Generates extendable, high coverage test cases in JUnit format that can be quickly turned into libraries of reusable test assets.
- 100's of Quick Fix options for fast, accurate resolution of errors.
- Integrated code coverage analysis ensures that your code is thoroughly tested.
- Team collaboration support allows you to define, distribute and enforce code quality standards across entire development teams.

Stop bugs before they start:

Jtest automates the valuable, yet often tedious tasks of code reviews and unit testing allowing development teams to adopt a "test as you go" strategy that promotes testing code as it is developed so that quality is built into the code from its inception and bugs eliminated before they infect the rest of the code base.

The result? Cleaner, safer, more reliable and consistent code. Reduced code rework. Faster, more predictable release cycles. Reliable applications and happier, more productive end-users. **To try Jtest today, call 888-305-0041 (x-3501) or go to www.parasoft.com/JDJmagazine.**

Parasoft Jtest clients include: Lehman Bros., Cisco, Disney, Raytheon, ADP, Sabre Holdings, GE and more.

PARASOFT®
We make software work.™

Automated Error Prevention™

Parasoft Corporation, 101 E. Huntington Dr., Monrovia, CA 91016. For information, call 888-305-0041 (x-3501). Copyright ©2006 Parasoft Corporation. All rights reserved.
All Parasoft product names are trademarks or registered trademarks of Parasoft Corporation in the United States and other countries. All other marks are the property of their respective owners.