

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

JDJ

JDJ.SYS-CON.COM VOL.10 ISSUE:9

IN THIS ISSUE...

Leveraging Open Source Solutions to Improve Productivity While Using EJBs

PAGE 8

Spring Web Flow

PAGE 14

Intelligent Web Applications with AJAX

PAGE 28

Navigating the Global Enterprise

PAGE 56

ARE PORTALS THE
MAGIC BULLET
OF WEB APPLICATION DEVELOPMENT?
The many advantages to utilizing portal software

RETAILERS PLEASE DISPLAY
UNTIL OCTOBER 31, 2005

\$5.99US \$6.99CAN

10>



PLUS...

▶ The Performance
of EJB 3.0

▶ Dealing with
Open Source Software

▶ A Blueprint for Developing
Language Tools

Your potential. Our passion.™

Microsoft

IF IT TAKES EIGHTEEN MONTHS TO WRITE AND INTEGRATE A NEW APPLICATION,

IT'S NOT REALLY NEW ANYMORE, IS IT?

.....
**WINDOWS SERVER SYSTEM WITH .NET HELPS YOU
BUILD AND INTEGRATE FASTER.**
.....

The Microsoft® .NET Framework, an integral component of Windows Server System™ is the development and execution environment that allows different components and applications to work together seamlessly. That means applications are easier to build, manage, deploy, and integrate. The .NET Framework uses industry standards such

as XML and Web Services which allow enterprise applications to be connected to infrastructure of any kind, which removes many of the headaches that can stretch development times endlessly. Find out more about application integration with Windows Server System and .NET: Simply get the Connected Systems Resource Kit at microsoft.com/connectedsystems



A LOT CAN HAPPEN IN 18 MONTHS.



© 2005 Microsoft Corporation. All rights reserved. Microsoft, the Windows logo, Windows Server System, and "Your potential. Our passion." are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Java Developers: 'Just Do It!'



Jeremy Geelan



DESKTOP



CORE



ENTERPRISE



HOME

Editorial Board

Desktop Java Editor: **Joe Winchester**
 Core and Internals Editor: **Calvin Austin**
 Contributing Editor: **Ajit Sagar**
 Contributing Editor: **Yakov Fain**
 Contributing Editor: **Bill Roth**
 Contributing Editor: **Bill Dudney**
 Contributing Editor: **Michael Yuan**
 Founding Editor: **Sean Rhody**

Production

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

Writers in This Issue

Calvin Austin, Jason Bell, Jason Collins, Shyman Kumar Doddavula, Yakov Fain, Jeremy Geelan, Onno Kluyt, Raghu Kodali, Kishore Kumar, Alex MacLinovsky, Paul Mukherjee, Igor Nys, Michael Poulin, Victor Rasputnis, Roy Russo, Anatole Tartakovsky, Julien Viet, Joe Winchester, Peter Zadrozny

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

Subscriptions

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

888 303-5282
 201 802-3012
subscribe@sys-con.com

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

Editorial Offices

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

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

©Copyright

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

Worldwide Newsstand Distribution
 Curtis Circulation Company, New Milford, NJ
 For List Rental Information:

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

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

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

September is here and since the name comes from the Latin septem, for "seven" – September having been until 153 BCE not the ninth but the seventh month of the Roman calendar – I have no hesitation in saying that it's an appropriate month to pluck just seven items from the wealth of information and insight in this issue and say just a little about each of them, to help you to decide what to read first in this issue of *JDJ*.

In Part Two of his new "Gas Station" series, Yakov Fain gives us a crash course on open source software, through a conversation with the best-selling author of *Succeeding with Open Source*, Bernard Golden. They discuss licensing and QA, and Golden notes in passing that one advantage to open source, from the user perspective, is that open source "breaks the linkage between product and company." In other words, if a product is useful but the company behind it is unsuccessful, open source means that the user community will still have access to the product source base and can continue using the product even if the company goes bust – very different from the dot-com days, when companies would shut down and leave their users in the lurch.

Java experts Peter Zadrozny and Raghu Kodali road test EJB 3.0's performance for us – using Oracle's implementation of the specification – by doing a few things that developers typically do with EJB 2.1 and then trying out the equivalent with EJB 3.0. They use a developer's preview of EJB 3.0 and find themselves very attracted to the simplicity and power of EJB 3.0. "The wonderful work done with annotations and the persistence, the ability to use POJOs, and the ability to test outside of the container are very attractive all by themselves," the authors note.

J2EE technology provides a good base to develop and deploy AJAX-based applications, so in this issue we say a major hello to Asynchronous JavaScript + XMLHTTPPre-

quest. Victor Rasputnis and Anatole Tartakovsky – who have been developing AJAX applications for the past five years and attest that "it's sound and very effective" – present a simple example and remind us that the best known AJAX applications already "out there" are probably Google Maps and Google Suggest from Google Labs (<http://labs.google.com>).

Java architect Kishore Kumar looks at part of the Spring Web application development suite, Spring Web Flow (SWF), which he finds to be a very powerful and elegant Web flow solution, suitable – as he shows in a sample scenario – for handling complex page navigations in any Web application.

Two of JBoss's best-known developers, lead developer Julien Viet and his colleague Roy Russo, discuss the overwhelming influence of the Portlet Specification (JSR 168) and wrestle with the question, "Are portals the magic bullet of Web application development?" The case for the use of portal software is not cut and dry, the authors suggest, but there are numerous advantages in adopting it.

Well-known Java writer Paul Mukherjee in "Blueprint for Developing Language Tools" presents an approach to developing language tools that has been used extensively for several products and projects that he has been involved with. The key, he shows, is to separate the triumvirate of core functionality, input format, and parsing to ensure the flexibility required for evolution over time.

Finally, in my (highly subjective) pick of seven pieces from the dozen or so excellent items in this issue: this month's "Back Page" is by *JDJ*'s own Jason Bell, written from his newfound standpoint as founder of a company – a B2B auction site for the airline industry – using all his own technology and launched without funding. Be sure to read it, because it is a heartening entrepreneurial *i*-Technology tale in its own right. As Jason writes, "If you've ever wondered, 'What if...?' don't wonder any more. Do it!"



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



Bring your development plans to light

Sneak a peek at XMLSpy® 2005,
the industry leading XML development
environment from Altova®.

Revealed in Version 2005:

- XSLT 2.0 support and debugger
- XPath 2.0 support and analyzer
- XQuery 1.0 support and debugger
- Eclipse integration

See for yourself why XMLSpy 2005 is the standard
for modeling, editing, debugging and transforming all
XML related technologies. Illuminate your strategy with
advanced standards compliance, innovative development
and analysis tools, and extended platform integration.
Use XMLSpy 2005 to structure XML Schemas and
devise XML documents, then automatically generate
runtime code in multiple programming languages.

Become a markup mastermind!

Download XMLSpy® 2005
today: www.altova.com

TOP SECRET

JDJ contents

JDJ Cover Story

by Roy Russo and Julien Viet

Are Portals the 'Magic Bullet' of Web Application Development?

The many advantages to utilizing portal software

50

FROM THE GROUP PUBLISHER

Java Developers: 'Just Do It!'
by Jeremy Geelan 3

JAVA ENTERPRISE VIEWPOINT
Corporate Java Training
by Yakov Fain 6

EJB
Leveraging Open Source Solutions to Improve Productivity While Using EJBs
Improving enterprise-level business services productivity
by Shyman Kumar Doddavula 8

TESTING
The Performance of EJB 3.0
The simplicity and power
by Peter Zadrozny and Raghu Kodali 20

YAKOV'S GAS STATION

Dealing with Open Source Software *Part Two*
by Yakov Fain 24

CASE STUDY
What Java Developers Can Learn from Boston's Big Dig...
...using SAP NetWeaver
by Jason Collins 34

CORE AND INTERNALS VIEWPOINT
Help I'm Out of Memory!
by Calvin Austin 36

SECURITY
How to Deal with Security When Building Application Architecture
Are you ready to face the challenge?
by Michael Poulin 38

TECHNIQUES

A Blueprint for Developing Language Tools
A proven approach to making them modular, extensible, and maintainable
by Paul Mukherjee 44

DESKTOP JAVA VIEWPOINT
One Size Fits No One
by Joe Winchester 48

NAVIGATION
Navigating the Global Enterprise
Developing a ubiquitous navigation utility
by Alex Maclinovsky 56

JSR WATCH
JCP Launches New Program
First constellation of Star Spec Leads takes shape - Part 2
by Onno Kluyt 60

@ THE BACKPAGE
Do I Really Need That?
by Jason Bell 62

Features

14



Spring Web Flow
by Kishore Kumar

28



Intelligent Web Applications with AJAX

by Victor Rasputnis, Anatole Tartakovsky, and Igor Nys



Yakov Fain
Contributing Editor

Corporate Java Training

Back in the '90s, we became accustomed to receiving half-inch thick glossy brochures from various training companies. Five days of such instructor-led training would cost more than \$2,000. For corporate employees this was "other people's money," and usually employees were entitled to at least one week of such training annually.

In '98, I was finishing my PowerBuilder career working as an independent contractor and decided to switch to Java. I had learned the language by reading dozens of books (yes, we used to buy technical books in the last century). But when you switch from one language to another, the most valuable knowledge is not in the books. I needed to know how real-world Java projects were designed and developed, so I paid \$2,500 for a week of WebLogic training, and it was worth every penny. The instructor was a knowledgeable guy and this course was an eye-opener. I figured out what had to go in servlets and what went into EJBs, what is a Façade pattern, and what to watch for. This training worked out well, because of my motivation: I needed to pay my bills, and when you apply for a Java position, your previous PowerBuilder experience (other than an understanding of OOP) doesn't count.

In 2001, the U.S. economy went into a long recession. When an enterprise goes through difficult times, its management lays off some people and immediately cuts the training budget. The mandatory trainings like Six Sigma or CMM will always survive, but the real stuff gets frozen. In the beginning of this millennium, those training companies that managed to survive reduced tuition costs and their fat brochures turned into flyers. Course enrollment dropped drastically. They would even run classes for just three students. If the course was designed for five days, the corporate clients would ask for it to be delivered in three days.

Less expensive online training came into the picture, but it proved to be boring and less effective than classroom training. However, since the economy remained in recession for three years, many people suspended their computer education and started to whine about outsourcing.

It's now 2005 and instructor-led training is back, and tuition is getting higher again. Guess what the most expensive training is these days? Some companies that make open source (free) software, charge a premium for training: \$3,000 for a four-day course. Well, they need to make money somehow, but I'm sure this won't last long; a new breed of startups that sells support of the open source tools will balance supply and demand by offering more reasonably priced training.

Who Is Teaching

When I was doing contract training, it worked as follows: I was getting an e-mail with the title of the class, airplane tickets, an overnight package with a training manual, and a CD with code samples. Smaller training providers don't develop their own manuals, but purchase the courseware from third parties. Once I had to deliver a one-day, MQ Series training. The manual was poorly written, but since I was right off a messaging project, I had lots of things to say on the subject. The students were happy and the class was saved. But I have to admit that in some cases, I've also taught technologies of which I have only book knowledge. Some instructors just read the manual aloud. Their version of the manual may include additional comments that you don't see, so it looks as if they know more than you.

Finding Quality Java Training

Most of the large corporations have a list of approved training vendors and courses to choose from, and it seems that there is nothing you can do about it. Wrong. Instead of using training vendors, find a Java-related conference or a seminar. Such seminars always have technical sessions on Java technologies with first-class speakers who are practitioners, and many of them are book authors as well. These seminars usually run training over parallel tracks so you can pick the classes that match your objectives. These events are less expensive than comparable vendor training, and the quality is better (just try to avoid marketing presentations, unless you are really interested in a particular product).

—continued on page 13

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

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

Advertising

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

Editorial

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

Production

Production Consultant:
Jim Morgan jim@sys-con.com
Lead Designer:
Tami Lima tami@sys-con.com
Art Director:
Alex Botero alex@sys-con.com
Associate Art Directors:
Abraham Addo abraham@sys-con.com
Louis F. Cuffari louis@sys-con.com
Assistant Art Director:
Andrea Boden andrea@sys-con.com
Video Production:
Frank Moricco frank@sys-con.com

Web Services

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

Accounting

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

SYS-CON Events

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

Customer Relations

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

The Developer Paradox:

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



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

■ **Automated:**

Automatically analyzes your code, applying over 500+ industry standard Java coding best practices that identify code constructs affecting performance, reliability and security.

Automatically generates and executes JUnit test cases, exposing unexpected exceptions, boundary condition errors and memory leaks and evaluating your code's behavior.

Groundbreaking test case "sniffer" automatically generates functional unit test cases by monitoring a running application and creating a full suite of test cases that serve as a "functional snapshot" against which new code changes can be tested.

■ **Extendable:**

Industry standard JUnit test case output make test cases portable, extendable and reusable.

Graphical test case and object editors allow test cases to be easily extended to increase coverage or create custom test cases for verification of specific functionality.

■ **Integrated:**

Integrates seamlessly into development IDE's to support "test as you go" development, and ties into source control and build processes for full team development support.

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



Automated Software Error Prevention™

Leveraging Open Source Solutions to Improve Productivity While Using EJBs

by Shyman Kumar
Doddavala

Improving enterprise-level business services productivity

This article provides a solution for improving productivity in scenarios where EJBs are used to implement business services using Spring, an Open Source POJO container, as a lightweight mock container for testing and using XDoclet attributes to define design-time considerations. The proposed solution has been validated using a POC. The subsequent sections explain the problem context, the different alternatives and their pros and cons, the industry trends and best practices, and a solution based on these trends and best practices.

Problem Context

One of the common debates while developing a business service using J2EE technologies is whether the EJB model or a POJO model is a better option. Let's analyze the good and bad aspects of these two choices. EJB Containers are standards compliant, usually backed by a vendor providing support. They provide several features required for enterprise-level business services:

- Enabling the service to be accessible remotely with inbuilt support for clustering (with load balancing and failover for the remote calls).
- Support for declarative transaction demarcation (that enables better reuse of the business services) with support for JTA (for distributed transactions) and propagation of the transaction context even across remote calls.
- Support for developing business services with an asynchronous invocation model through JMS and MDBs.
- Support for making the business services accessible as Web services through standardized mechanisms. They also provide support for EIS integration through JCA.

- Support for declarative security and propagation of the security context across remote invocation too.
- Provide tools for development like WSAD for WebSphere.

One of the biggest complaints about using the EJB model for developing business services has been the long build-deploy-test cycle involved in this approach, which results in low productivity. POJO Containers like Spring, which is an open source framework, have come up with solutions to fill this gap by providing the framework needed for developing business services using a POJO model, wherein the services are



modeled as POJOs and Spring provides features similar to the EJB container like declarative transaction demarcation, etc. Spring also provides a good configuration mechanism similar to that provided by an EJB container.

Since Spring by itself doesn't provide the equivalent of JMS and MDBs and many of the features required for enterprise application development, including remoting, clustering etc., it can't be used to replace a J2EE container. Developing the business services as POJOs in Spring thus provide higher productivity, but it usually results in the services getting tied to Spring's equivalents of the J2EE container-provided features such as its declarative transaction model, remoting model, Web services stack, etc.

J2EE containers usually provide more robust, standards-compliant features than their equivalents provided by Spring. For example, J2EE containers are more likely to support the latest JSRs like JSR 109, which defines a standard and portable mechanism for packaging and deploying Web services, and similarly JSR 181, which defines a simple-to-use annotations-based model for developing Web services. There are several other JSRs such as 104, 105, 106, 156, and 157 for defining standards for Web services that are more likely to be supported by J2EE containers than Spring. Similarly the remoting capabilities provided by most commercial J2EE containers provide advanced features needed by enterprise services like support for clustering with load balancing and failover for the remote calls as well as transaction propagation across the remote calls, etc., through the remote EJB model and for the other models built on them. Similarly as explained in the article that is listed first in the References section, the EJB end-point model provides a simpler mechanism for developing Web services since the container takes care of serializing the requests to the EJBs. In comparison, the Web Tier end-point model is generally supported by POJO containers that require the POJO developers to take care of the synchronization aspects. So, strategically it's better to rely on standards-compliant and vendor-supported J2EE container features than Spring's equivalent features.

Trends

The EJB3 spec is trying to address some of these problems by providing an annotations/attributes-based mechanism that will allow the business services to be developed as POJOs and allow all design-time consid-

Shyman Kumar Doddavala

works as a technical architect in the J2EE Center of Excellence Group at Infosys Technologies Ltd. He has an MS in computer science from Texas Tech University and over 8 years experience in software development.

erations like transaction demarcation and remoting to be defined through annotations (see the fifth entry in the References section). This enables the EJB interfaces and stubs to be autogenerated and allows the services to be developed and unit tested as POJOs. Some of the application server vendors are offering test harnesses and mock containers (refer to the test harness, Reference #2, provided by Oracle in their EJB3 preview, which supports testing entity beans out of the container) that provide the basic container features like JNDI and CMT needed to test services modeled as POJOs with the annotations. Unfortunately, EJB3 is still not finalized and it will be long before AppServer vendors provide support for it. The next issue is extending the support provided by the test harnesses for testing the business services outside of an EJB container, which may be limited and may not be portable.

Industry Best Practices

The following is a look at the best practices, in terms of use of technologies while developing J2EE applications, and especially modeling the business services:

- Use of Stateless Session Beans for implementing the stateless business services
–A related pattern that recommends the use of EJBs with POJOs behind them (see Reference #3).
- Use of JMS and MDBs for business logic with asynchronous invocation.
- Use of the J2EE container–provided features like the declarative transaction demarcation (CMT) mechanism and its Web services stack–based features for making the business logic accessible as Web services.
- Use of POJO-based persistence model.

Solution

Solution Strategy

Using only Spring to model the business services is flawed because it

wouldn't be based on standards and wouldn't leverage the high-end features provided by AppServer vendors, and using only EJBs has the disadvantage of low productivity. So, an ideal scenario is to use a J2EE application framework that combines the good aspects of these two options. It should provide a mechanism through which it allows POJO-based development to enable faster build-deploy-test cycles, thus overcoming the productivity issues associated with EJBs, but then translates all of the required features to J2EE/EJB container features so that the application can be deployed in an application server leveraging the vendor-supported container features. It should therefore enable using a POJO container such as Spring as a mock container for testing.

Solution Design

It is possible to design a framework based on the strategy defined earlier using attributed-oriented programming techniques that will allow the core business logic to be developed and tested as POJOs and then deployed as EJBs. All of the design-level considerations for the business services like its transactional requirements are specified using attributes. Then a precompile step can generate the descriptors and configurations that allow the business services to be executed in a POJO (Spring) container as well as in EJB container contexts. The Spring container can be leveraged as the POJO container to enable easier unit testing of the business services. The “POJO using Spring”–based implementation can be used for the development and the daily unit testing, while the EJB-based implementation can be used for occasional unit testing and for the integration testing.

JDK1.5 provides support for annotations, which seems like a natural choice for something like this. However, since most application servers currently don't support JDK1.5, XDoclet attributes or Apache Commons Attributes can be used to implement this to address the current needs. A POC has been developed

using XDoclet attributes to validate these concepts as explained below.

The class diagram of a sample OrderService designed using this solution is shown in Figure 1. The OrderService interface is implemented by the POJO OrderServiceImpl and the EJB wrapper for it is generated. The design time considerations like the transaction requirements of the createOrder() method are specified as XDoclet attributes, which are used to generate the POJO Container(Spring) and EJB container–specific deployment descriptors.

The code samples for these are shown in Listing 1.

A few pojo_ejb_fw attributes (very much similar to the EJB attributes) are defined to define the Design-time considerations like transaction requirements and local/remote interface requirements.

The “@pojo_ejb_fw.service” class-level attribute is used to capture the service-level configuration like the service name, which is then translated to a Spring bean name for POJO deployment and a JNDI name for an EJB-based deployment.

Similarly, the “@pojo_ejb_fw.transaction” attributes are used to define the transaction requirements at class level as well as method level. These attributes are defined to be compatible with the EJB attributes so that it allows for both POJO and EJB deployments.

The “@pojo_ejb_fw.interface” attribute is used to determine if a local/remote EJB interface are required

An XDoclet Task is created for pojo_ejb_fw tags with one subtask to generate the spring configuration files for POJO-based development and another to generate the EJB classes for the EJB-based development. The Listing 1 sample class attributes translate to the Spring configuration file shown in Listing 2.

```
</bean>
<bean id="transactionManager"
class="com.infosys.j2ee.radien.spring.
MockPlatformTransactionManager"/>
<!-- End runtime configuration. -->
</beans>
```



An ideal scenario is to use a J2EE application framework that combines the good aspects of these two options”

Perforce.

The *fast* SCM system.

For developers who don't like to wait.



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

[Fast]

[Scalable]

[Distributed]

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

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

PERFORCE
SOFTWARE

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

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

Feature	EJB Model	POJO Model Without Any Container	POJO With Container (Spring)	Proposed Solution
Standards Based	Yes	No	No	Yes
Declarative Transactions	Yes	No	Yes	Yes
Productivity	Low	High	High	High
Easy Migration to EJB3	No	No	No	Yes
Allows Using Mocks Easily	No	No	Yes	Yes

Table 1

develop and test a service especially in a Test Driven Development environment, and thus can improve productivity. But the EJB Model provides a solution that is standards-based and with vendor-supported, enterprise-level features. The proposed solution helps develop a business service as a POJO using the “POJO with Container (like Spring)” model while at the same time autogenerating the EJB stubs to wrap it for the final deployment, thus providing the best of both worlds solution.

Even the transactional behavior can be tested using the Spring declarative transaction Model while using the EJB CMT for the final deployment. The EJB3 spec is also moving towards an annotations-/attributes-based model and so the proposed solution provides an easier migration to EJB3 when it is available without getting locked into proprietary alternatives for EJBs like Spring.

Conclusion

The proposed solution allows a POJO-based development without compromising on the “Stick to Standards” principle. It allows developing and testing business services as POJOs, thus improving the productivity, while at the same time generating an EJB-based wrapper for the final deployment, thus leveraging the enterprise-level features like EJB CMT based on JTA, clustered remot-ing capability (through Remote EJBs) etc., provided by the J2EE Containers.

The POC demonstrates the feasibility of the proposed solution and can be extended further to a full-fledged implementation of the solution. ☛

References

- JSR 109: Web Services Inside of J2EE Apps: <http://www.onjava.com/pub/a/onjava/2002/08/07/j2eewebvs.html?page=1>
- How-To Build out-of-container example using EJB 3.0: <http://www.oracle.com/technology/tech/java/oc4j/ejb3/howtos/howtoejb30outofcontainer/doc/how-to-ejb30-out-of-container.html>
- An Introduction to Attribute-Oriented Programming: <http://dssg.cs.umb.edu/resources/attribute-oriented-programming.html>
- Simplifying EJB Development with EJB 3.0: http://www.oracle.com/technology/tech/java/news-letter/articles/simplifying_ejb3.html

Listing 1

```
package com.infosys.j2ee.radien.demo.order.service;
import com.infosys.j2ee.radien.core.Service;
import com.infosys.j2ee.radien.demo.order.domain.Order;
/**
 * Example of a simple Radien Service
 *
 * @radien.service id="OrderService"
 * name="OrderService"
 * type="Stateless"
 * view-type="both"
 *
 * @radien.transaction type="Required"
 * @radien.transaction-type type="Container"
 *
 * @weblogic.pool initial-beans-in-free-pool="1"
 * max-beans-in-free-pool="3"
 */
public interface OrderService extends Service{
/**
 * creates an order.
 *
 * @radien.interface-method view-type="both"
 *
 * @radien.transaction type="Required"
 */
public void createOrder(Order order);
}
```

Listing 2

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC
"-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans
default-autowire="no"
default-lazy-init="false"
default-dependency-check="none"
>
<!-- Begin design time configuration. -->
<bean id="spring-services" lazy-init="true"
class="org.springframework.context.support.ClassPathXmlApplicationC
ontext">
<constructor-arg>
<list><value>spring.xml</value></list>
</constructor-arg>
<constructor-arg type="boolean">
<value>true</value>
</constructor-arg>
</bean>
<bean
id="OrderServiceTarget"
class="com.infosys.j2ee.radien.demo.order.service.OrderServiceImpl"
>
<property name="serviceConfig"><ref bean="OrderServiceConfig"/></>
```

```

property>
</bean>
<bean
id="OrderService"
class="org.springframework.transaction.interceptor.TransactionProxy
FactoryBean"
>
<property name="target"><ref bean="OrderServiceTarget"/></property>
<property name="transactionManager"><ref bean="transactionManager"/>
</property>
<property name="transactionAttributes">
<props>
<prop key="createOrder">PROPAGATION_REQUIRED</prop>
</props>
</property>
</bean>
<!-- End design time configuration. -->
<!-- Begin runtime configuration. -->
<bean id="OrderServiceConfig"
class="com.infosys.j2ee.radien.demo.order.service.
OrderServiceConfig">
<property name="param1" value="value1"/>

```

Listing 3

```

/*
 * Generated file - Do not edit!
 */
package com.infosys.j2ee.radien.demo.order.service.ejb;
import javax.ejb.*;
import com.infosys.j2ee.radien.core.Service;
import com.infosys.j2ee.radien.core.ServiceLocator;
import com.infosys.j2ee.radien.spring.AbstractStatelessSessionBean;
import com.infosys.j2ee.radien.demo.order.service.OrderService;
/**
 * Service interface.
 * @radien-generated at 27-05-05

```

```

 * @copyright Infosys Technologies Ltd
 * @author Radien
 * @version ${version}
 *
 * @weblogic.pool initial-beans-in-free-pool="1" max-beans-in-free-
pool="3"
 *
 * @ejb.bean name="OrderService"
 * description="A session bean named OrderService"
 * display-name="OrderService"
 * type="Stateless"
 * view-type="both"
 * jndi-name="ejb/OrderService"
 * local-jndi-name="ejb/OrderServiceLocal"
 * local-business-interface="com.infosys.j2ee.radien.demo.order.ser-
vice.OrderService"
 * impl-class-name="com.infosys.j2ee.radien.demo.order.service.ejb.
OrderServiceBean"
 *
 * @ejb.transaction-type type="Container"
 * @ejb.transaction type="Required"
 *
 * @ejb.home extends="javax.ejb.EJBHome" local-extends ="javax.ejb.
EJBLocalHome"
 *
 * @ejb.interface extends="javax.ejb.EJBObject" local-extends
="javax.ejb.EJBLocalObject"
 *
 */
public class OrderServiceBean extends AbstractStatelessSessionBean
{
OrderService _myRadienServiceImpl;
public static final String SERVICE_NAME="POJO-OrderService";
/**
 * Obtain our POJO service object from the Service Registry
 * @see com.infosys.radien.spring.AbstractStatelessSessionBean#onEj
bCreate()
 */

```

Corporate Java Training

—continued from page 6

Here's the list of some of the training events to consider for Java developers and architects:

- JavaOne: <http://java.sun.com/javaone>
- WebServices Edge: <http://webservicesedge.sys-con.com>
- O'Reilly Open Source Convention: <http://conferences.oreillynet.com/os2005>
- Weekend With Experts: <http://www.weekendwithexperts.com>
- No Fluff Just Stuff: <http://www.nofluffjuststuff.com>
- Eclipse World: <http://www.eclipseworld.net>
- OOPSLA: <http://www.oopsla.org>
- JavaPro Live: <http://www.ftponline.com/conferences/javaprolive>

- JavaPolis, Europe: <http://www.javapolis.com>
- Colorado Software Summit: <http://www.softwaresummit.com>

Pick a conference and remind your boss about all those long hours you've spent on the project. You need and deserve quality Java training! In some cases your company can even invite the entire seminar to your town.

Recently I've asked a couple of seasoned Java programmers what the acronyms AJAX, AOP, and JBI mean. They didn't know. How about you? Ask your colleagues the same question and see for yourself. There are plenty of great programmers who just write Java code for their employer day in and day out. Raise your head and look around. The Java community is exciting and vibrant and something is happening all the time. Be a part of it. ☺

Spring Web Flow

The promise of handling complex page navigations in any Web application

by Kishore Kumar

Page navigation requirements become more demanding as Web applications get bigger and more complex. Hard-coded page flow rules make applications less resilient to changes. In this scenario, reusing business logic is one aspect and reusing page flow becomes another aspect. Especially in situations that demand a wizard-kind behavior, it's essential to capture application page flow logic in a declarative fashion.

Spring Web Flow (SWF) is a powerful framework for implementing page flows in a Web application. Even though SWF is part of the Spring Web application development suite that includes Spring MVC, its flexible architecture allows it to be used with any presentation tier Web framework including Apache Struts and JSE. SWF is now available as a development-ready PR 3.0 version and is expected to be part of Spring 1.3.

In this article, we'll explore the features of SWF and use it to build a sample page flow scenario.

Spring Web Flow in Action

In SWF, every flow is a Finite State Machine (FSM). It consists of states and transitions. A state represents the execution of some actions (business logic) or a view that's displayed to the user for user input.

When the FSM starts, the flow enters the start state (a state that's specially marked as start state). A state can be an action state or a view state (SWF has other state types too like decision state, sub-flow state, and end state). Both action states and view states can define one or more transitions from them to a target state. An event triggers a transition from one state and moves the flow to the new state.

When an action state is entered, the flow executes one or more configured action objects for that state. The execution of an action can produce a logical result in the form of an event and it's processed to select a suitable transition. This transition is applied (executed) and as a result the flow enters the next state as defined by this transition.

When a view state is entered, the flow is temporarily paused and a view, as configured for that state, is displayed to the user. When the flow receives an event (user input can be encapsulated as an event) from the user, it resumes execution and processes the event to transition to the next state.

This new state is then processed in the same way and so on. The flow continues to execute until it reaches the end state (specially marked as end state) where the flow is terminated and a view (as configured for the end state) is displayed to the user. Figure 1 depicts how states are processed in FSM.

Flow Builders

SWF can build a flow definition consisting of states and transitions from an XML configuration file. Since SWF implements the classic GoF Builder design pattern, it provides the flexibility to define and use custom flow building logic. Figure 2 shows the flow builder class diagram.

SWF has many *FlowBuilder* implementations. The *XMLFlowBuilder* is one of them that can build a flow from an XML configuration file.

The *FlowFactoryBean* class internally uses a *FlowBuilder* implementation and acts as an assembler for creating a Flow.

```
FlowBuilder builder = ...
Flow flow = new FlowFactoryBean(builder).getFlow();
```

The *XMLFlowFactoryBean* implementation uses the *XMLFlowBuilder* to create a flow. In a Spring container, you can configure an *XMLFlowFactoryBean* as shown below and use it to build a flow:

```
<bean id="aFlow" class="org.springframework.web.flow.config.
XMLFlowFactoryBean">
  <property name="location" value="classpath:myflow.xml" />
</bean>

BeanFactory factory = ...
Flow flow = (Flow)factory.getBean("aFlow");
```

[Spring recognizes the defined bean as a 'factory bean' and invokes its getObject() method, which in turn invokes the getFlow() method to create and return a flow.]

You can also create a flow by sub-classing the *AbstractFlowBuilder* class as shown below. This class defines many helper methods like *addViewState()* and *addActionState()* to define the flow at runtime.

```
public class AFlowBuilder extends AbstractFlowBuilder {
  protected String flowId() {return "aFlow";}
  public void buildStates() {
    addViewState("stateId","viewName",transition);
    addActionState("stateId", targetAction, transition);
    ...
  }
}
```



Kishore Kumar works as a Java architect at U.S. Technology (www.ustri.com). He specializes in J2EE applications.

kishore_kumar@usswi.com

REPORTING MADE EASY!

But don't take our word for it.

“ Windward Reports saved us 90% development time on reporting. We no-longer have to hand code reports.”

- I.T. Project Manager, State of Virginia agency

“ We dynamically produced RTF documents with minimal training. A real lifesaver! ”

- Ciber for Lloyd's of London

“ Windward's documentation is clear and easy to follow.”

- S.S.Mohanty, Nucleus Software Exports LTD

“ The graphic quality of our reports improved tremendously.”

- Chad Boggs, Developer

“ The process of creating a new project is at least 5 times faster. Plus, Windward's technical support team listens to customers' needs, and responds very fast.”

- Mr. Bjerregaard Pedersen, Backbone Digital Systems

To get started goto www.windwardreports.com
or give us a call at 303.499.2544 or email us at sales@windward.net



A flow builder internally uses a *FlowServiceLocator* to create all flow elements including a *Flow*, a *State*, and a *Transition*. The *BeanFactoryFlowServiceLocator* implementation uses Spring's bean factory to look up and create flow elements. By default the *XMLFlowBuilder* uses the *BeanFactoryFlowServiceLocator*.

The *XMLFlowBuilder* allows a flow element (flow/state /transition, etc.) to have the common optional properties bean, classref, class, and autowire. If a 'bean' or 'classref' property is provided the *XMLFlowBuilder* uses this value for resolving the bean reference using Spring. Otherwise, if a "class" property is given, a new element of the given type is created using reflection. The 'autowire' property denotes whether to use Spring's autowire (the ability to automatically apply dependency injection) capability or not.

A Sample Scenario – Shopping Cart Checkout

Let's discuss a simple scenario where a user uses a Web wizard to check out items in his shopping cart that he wants to buy. Figure 3 shows this scenario.

- Step 1:** Create an XML Web flow definition (checkoutFlow.xml) as shown in Listing 1.
- Step 2:** Create an XML flow factory bean entry in Spring's context XML configuration file as shown in Listing 2.
- Step 3:** Execute the flow. See Listing 3.

A *FlowExecutionManager* is a controller façade to the Spring Web Flow system. On receiving an event, it loads the flow defined by the

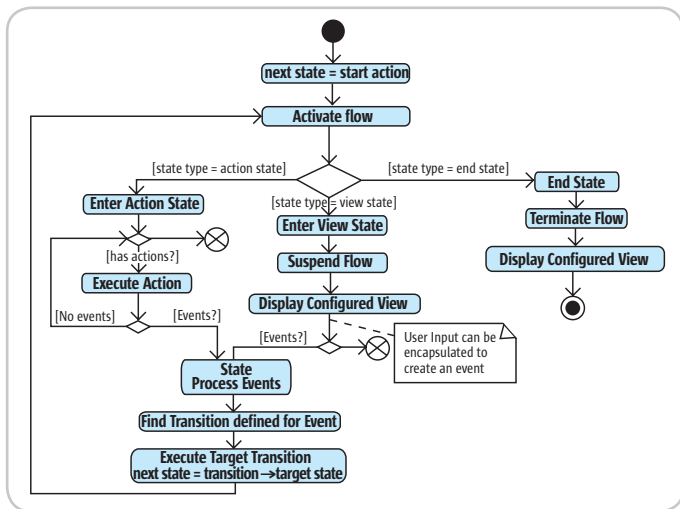


Figure 1 Activity diagram showing how states are processed in SWF

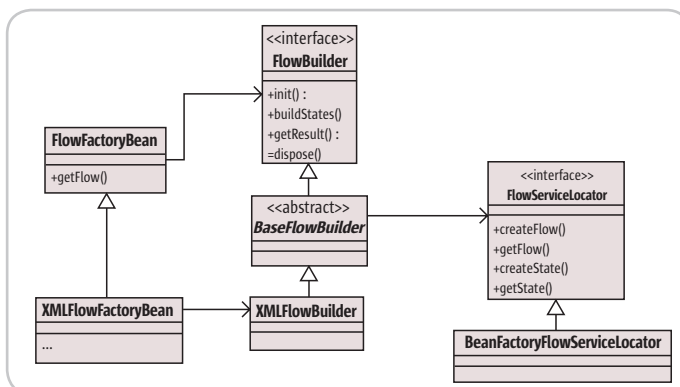


Figure 2 Flow builder class diagram

'_flowId' parameter using Spring's bean factory (using *XMLFlowFactoryBean*). Then, it looks for an event parameter ' _flowExecutionId' and if it isn't present, it will create a new *FlowExecution* instance and uses it to start this new flow. A *FlowExecution* encapsulates the execution state of a flow. Otherwise, the flow execution manager loads the existing flow execution instance from its storage (in our case this is HTTP Session-based storage) and uses it to continue execution. Figure 4 depicts this as a UML sequence diagram.

[Spring provides standard controllers to integrate with Spring MVC and struts framework. The controller servlet shown in Figure 4 and Figure 5 shows the logic that goes into these controllers.]

Let's look at each state in our sample checkout flow.

The 'viewShoppingCart' is a view state (it's also the start state). When the flow enters a view state, it triggers the rendering of a view. The view name defined by the 'view' attribute of any view state is a logical name that needs to be mapped to a physical view. In Spring MVC a *ViewResolver* and in struts an *ActionForward* can do this mapping. When the flow execution enters the 'viewShoppingCart' state, the flow execution suspends the active flow and returns a *ViewDescriptor* that represents a logical view 'sc.view.' A controller servlet can now map this logical view to a JSP (or any other view type) and forward the request to that JSP to render the view.

Once the user selects 'next' on the view shopping cart page, the controller servlet uses the flow execution manager to resume the flow execution. The flow execution manager requires the following parameters to continue flow execution:

- **flowExecutionId** – denotes the current flow execution state. [Required]
- **currentStateId** – denotes the current state. [Optional – required only to support browser back button usage]
- **eventId** – denotes the generated event. [Required]

The flow execution manager can be triggered to resume flow execution as show below:

```
//Lookup flow execution manager from spring's context
FlowExecutionManager mgr = ...
Map params = new HashMap();
// copy all request parameters to params HashMap
Event event= new Event(source, eventId, params);
ViewDescriptor vd = mgr.onEvent(event);
...
```

Figure 5 depicts this as a sequence diagram.

The flow execution manager loads the flow execution from the storage and uses it to process the event. The flow execution instance will apply the event on the known current state (the supplied current state will override this), which will result in a transition to a new state.

In our case, the current state is 'viewShoppingCart' and the '_eventId' value is 'next.' Since there's a transition defined for this criterion, it's executed and the flow execution enters the target state 'viewShippingAddress.' This will display the shipping address page to the user. Had the '_eventId' value been 'cancel' it would have resulted in entering the 'cancelFlow' state.

If the user now inputs the shipping details and selects 'next,' it should take the flow to the 'bindAndValidateSA' state, which is an action state. An action state executes one or more implementations of the 'Action' interface when entered. A typical action can act as a bridge between the presentation tier and the business tier or even execute the business logic directly (not a good practice).

The SWF supplies a ready-made 'action' implementation 'FormAction' for binding form parameters to a POJO bean. The 'FormAction' class extends from 'MultiAction', which gives it the ability to dispatch a given method at runtime. The method to be executed can be specified using a 'method' attribute to an 'action' in the Web flow configuration.

In our example, we haven't specified a 'method' attribute and hence the default behavior is to execute a method with the same name as the state id, which is 'bindAndValidate.' The default implementation of the method 'bindAndValidate' in 'FormAction' dynamically binds the event parameter values to a POJO form bean, validates the bean if a 'validator' is specified, and finally places the form bean in the specified scope (flow scope or request scope).

In the checkout flow definition, the 'bindAndValidateSA' state has an action configured with a bean property of 'checkout.shippingAddress.' This is resolved using Spring's context XML.

```
<bean id="checkout.shippingAddress" class="org.springframework.web.flow.action.FormAction">
  <property name="formObjectName" value="shippingAddress"/>
  <property name="formObjectClass" value="checkout.beans.ShippingAddress" />
  <property name="formObjectScopeAsString" value="flow" />
</bean>
```

This form action uses the POJO 'checkout.beans.shippingAddress' and puts it in flow scope after binding the event parameter values. When this action completes execution, it signals a 'success' event. This 'success' event selects the transition that takes the flow to the 'viewBillingAddress' state. The flow then continues to execute this new state and so on.

When the flow enters an end state, it terminates the flow and displays the configured view.

```
<end-state id="finishCheckout" view="homePage" />
```

Sub Flows

Spring Web Flow also supports sub-flows. A sub-flow is a logically separate but complete flow in itself. It's similar to a subroutine in a procedural language. SWF has a separate state to denote a sub-flow. A sub-flow state can also refer to a flow defined in another flow definition file.

```
<subflow-state id="lookupPreviousShippingDetails" flow="shippingDetailsLookupFlow">
  <attribute-mapper >
    <input name="customerId"/>
    <output name="shippingAddress">
  </attribute-mapper>
  <transition on="success" to="viewShippingAddress">
</subflow-state>
```

At any given time the *FlowExecution* manages a stack of flow sessions. A flow session is a placeholder for flow variables and flow state. Action implementations can access the flow-scoped parameters using the *RequestContext* interface. When the flow execution enters a sub-flow state, the flow execution creates a new flow session and adds it as the top item in the flow session stack. This session remains as the active session until the sub-flow completes execution and the flow execution activates its parent flow session. A parent flow can pass its flow session parameters to the newly spawned sub-flow using an attribute mapping as shown in the above configuration. An input element can map a flow-scoped parameter in the

parent flow and copy it to the sub-flow session. Similarly, an output element copies the parameter value from the sub-flow back to the parent flow when the flow completes.

An attribute mapper configuration can optionally reference a *FlowAttributeMapper* implementation defined in Spring's context using the 'bean' element attribute. In this case the input and output elements aren't necessary. If no flow attribute mapper implementation is referenced, SWF will use a default implementation

ParameterizableFlowAttributeMapper uses the input and output element mappings to do the attribute mapping.

Integration with Struts

SWF provides a *FlowAction* class, which is a struts action class that can act as the front controller entry point into the Web flow system. Typically, a single parameterized (by flow id) *FlowAction* can manage all flow executions. The *FlowAction* delegates every request to a *FlowExecutionManager*. This class is also aware of a *BindingActionForm* adapter that adapts Spring's binding facility to the struts action form model. A flow action can be configured as shown below in the 'struts-config.xml' configuration file.

```
<action path="/checkout" type="org.springframework.web.flow.struts.FlowAction" name="bindingActionForm" scope="request" className="org.springframework.web.flow.struts.FlowActionMapping">
  <set-property property="flowId" value="CheckoutFlow" />
</action>
```

A struts action form with the name 'bindingActionForm' and the class 'org.springframework.web.struts.BindingActionForm' has to be defined in the struts-config.xml file.

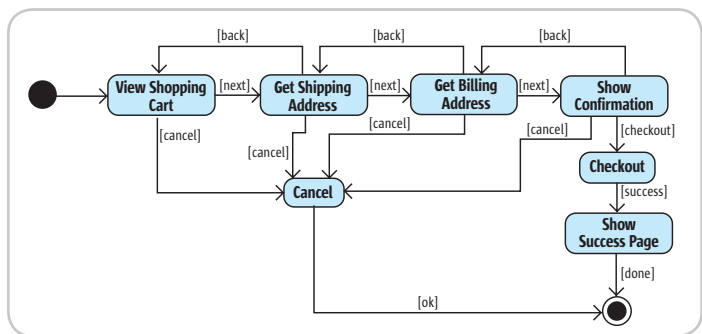


Figure 3 A sample checkout scenario

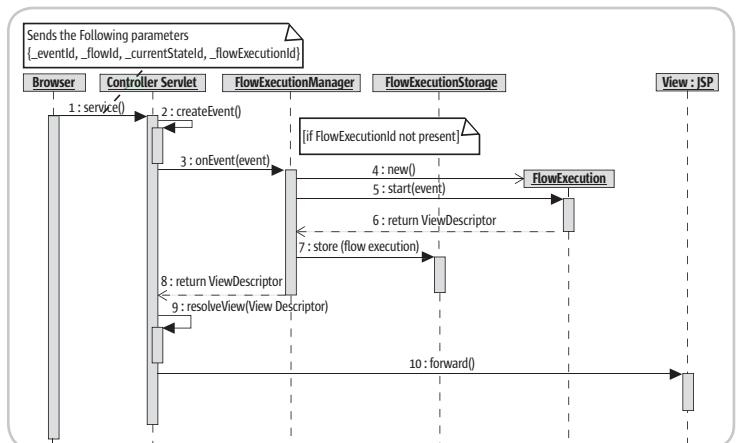


Figure 4 Flow execution sequence diagram with no flow execution id parameter

The *FlowAction* controller maps all logical view names returned from a view state and end state as action forward mappings (typically global forwards) in the struts configuration.

The use of Spring's binding-aware action form requires some additional struts configuration. A custom binding action-aware request processor is required to defer the form population.

```
<controller processorClass="org.springframework.web.struts.BindingRequestProcessor"/>
```

A binding plug-in is also needed to plug in an errors-aware jakarta-commons-beanutils adapter:

```
<plug-in className="org.springframework.web.struts.BindingPlugin"/>
```

Integration with Spring MVC

SWF has a very natural integration with the Spring MVC framework. It provides a Spring *FlowController* class that can act as a front controller for executing Web flows. A single *FlowController* instance can be parameterized (using the flow id parameter) to execute different flows.

Making the Back Button Work

A page flow system like SWF uses server-side state management to direct page navigation. This can easily conflict with browser back button use. The easiest way to tackle this is to not use the browser back button.

However, if required you can support browser back button functionality by sending the current state id (`_currentStateId`) explicitly with every request to the flow controller. This way the *FlowExecution* will know which state to use and result in correct navigation even when the back button is used.

```
<form name="someForm" action="checkout">
    <input type="hidden" name="_flowId" value="<%=request.getAttribute("flowId")%>" />
    <input type="hidden" name="_currentStateId" value="viewShippingAddress" />
    <input type="hidden" name="_eventId" value="next" />
    <input type="hidden" name="_flowExecutionId" value="<%=request.getAttribute("flowExecutionId")%>" />
    ...
</form>
```

However, this approach won't work if there are sub-flows. In this case, there's still confusion as to what flow (sub-flow or its parent flow) should be used. SWF provides a continuation-based approach to solve this problem. In SWF a continuation is a snapshot of the

flow execution state (serialized form of *FlowExecution* instance) at any given point of time. To support this, an *HttpSessionContinuationFlowExecutionStorage* mechanism should be used instead of the default *HttpSessionFlowExecutionStorage*.

```
<action path="/checkout" type="org.springframework.web.flow.struts.FlowAction" name="bindingActionForm" scope="request" className="org.springframework.web.flow.struts.FlowActionMapping">
    <set-property property="flowId" value="CheckoutFlow" />
    <set-property property="storage" value="sessionContinuation" />
</action>
```

This way, for every request, a serialized form of the current *FlowExecution* instance is stored in the HTTP session (under a separate flow execution id attribute). For a new client request (with the flow execution id parameter), the *FlowExecutionManager* will de-serialize the saved *FlowExecution* instance and uses it to process the event. Hence, every request will be processed using the correct context.

Conclusion

Spring Web Flow is a very powerful and elegant Web flow solution. It's definitely a promise for handling complex page navigations in any Web application. ☺

References

- *Spring Framework*: <http://www.springframework.org>
- *Spring Web Flow*: <http://opensource.atlassian.com/confluence/spring/display/WEBFLOW/home>
- *Ervacon*: <http://www.ervacon.com/products/springwebflow>

Listing 1

```
<webflow id="checkout" start-state="viewShoppingCart">
    <view-state id="viewShoppingCart" view="sc.view">
        <transition on="next" to="viewShippingAddress" />
        <transition on="cancel" to="cancelFlow" />
    </view-state>
    <view-state id="viewShippingAddress" view="sa.view">
        <transition on="next" to="bindAndValidateSA" />
        <transition on="back" to="viewShoppingCart" />
        <transition on="cancel" to="cancelFlow" />
    </view-state>
    <action-state id="bindAndValidateSA">
        <action bean="checkout.shippingAddress" />
        <transition on="success" to="viewBillingAddress" />
        <transition on="error" to="viewShippingAddress" />
    </action-state>
    .....
</webflow>
```

Listing 2

```
<bean id="CheckoutFlow" class="org.springframework.web.flow.config.XMLFlowFactoryBean">
    <property name="location" value="classpath:checkoutFlow.xml" />
</bean>
```

Listing 3

```
BeanFactory factory = ...
FlowExecutionManager mgr = (FlowExecutionManager)factory.getBean("mgrID");
//This assumes that a Flow Execution Manager is defined in spring's configuration file
mgr.setStorage(new HttpSessionFlowExecutionStorage());
//Use HTTP session to store flow execution state
Map parameters = new HashMap();
parameters.put("_flowId", "CheckoutFlow");
Event event = new Event(source, eventId, parameters);
ViewDescriptor vd = mgr.onEvent(event);
...
```

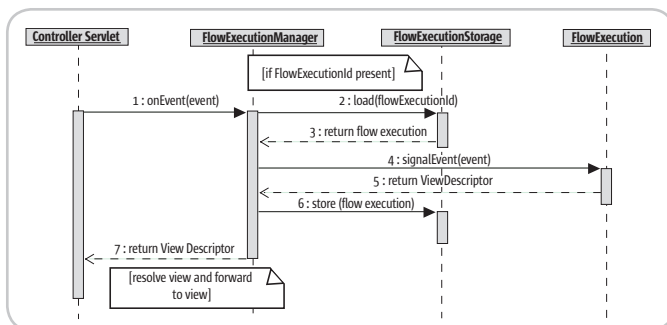
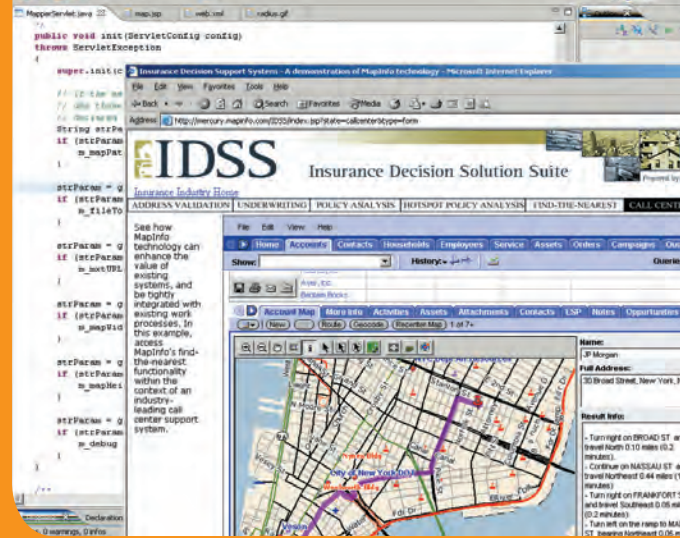


Figure 5 Flow Execution sequence diagram when a flow execution id is provided

SUPERCHARGE YOUR APPS WITH THE POWER OF LOCATION INTELLIGENCE



100% Java SDK enables Location Intelligence through web services

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

Integration with Eclipse, NetBeans, IntelliJ and more

Create applications for:

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

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

Contact us at sales@mapinfo.com

 **MapInfo**
Be Location Intelligent™

The Performance of EJB 3.0

The simplicity and power

by Peter Zadrozny and
Raghu Kodali

We've all heard about the simplicity and power of the EJB 3.0 specification. And because this has proven to be true, we can't help but think that performance must be rather poor. After all, all that simplicity must come at a price.

With this in mind, we set out to test EJB 3.0's performance using Oracle's implementation of the specification. Although the implementation we used is a developer preview, where the focus is typically on product stability instead of performance, our expectations are that the performance will be below or in the best case the same as previous versions of the EJB specification.

We thought that the best way to test was to do a few things that developers typically do with EJB 2.1 and then try out the equivalent with EJB 3.0. This would let us compare performance rather than deal with raw numbers that probably wouldn't mean much. We tested the Data Transfer Object (DTO) design pattern, the Session façade design pattern, and the use of the Container Managed Relationships (CMR) functionality of the entity beans.

The application we used, as well as the test harness and the methodology, is described in *J2EE Performance Testing* by Peter Zadrozny (Expert Press, 2002). The harness is a simple dispatcher servlet that executes each discrete test case based on the JazzCat application, a catalog of jazz recordings. The database schema includes tables for bands, musicians, instruments, tracks, and albums. The application also handles the storage and retrieval sessions recording and takes of a track.

Our test environment was based on Oracle Application Server EJB 3.0 Preview (Developer Preview 3), and Sun Java HotSpot Client VM (build 1.5.0_03-b07, mixed mode, sharing). We used Oracle Database 10g Enterprise Edition Release 10.1.0.2.0. We used the default settings on all of the software.

The users were simulated with The Grinder 3.0 Beta 25 (<http://grinder.sf.net>), with a sample size of 5,000 milliseconds. Each test run lasted eight minutes; we ignored the first three minutes to allow the test to stabilize. Each simulated user ran a test script that called the corresponding test case 10 times. The test scripts were executed continuously in a sequential fashion for the duration of the test run. Two things are worth noting: there was a separate HTTP session for each execution of the test scripts, and there was no sleep time between each call to the test case. The latter was done to create a highly stressful situation so we could see how EJB 3.0 behaved when pushed to its limits.

To get a complete picture of the performance, we used two key indicators: Aggregate Average Response Time (AART), to reflect the end user perspective, and Total Transactional Rate (TTR) or throughput, to reflect the load on the systems involved.

We used three Dell PowerEdge 2850 computers with dual Intel Xeon (HT) at 3.4GHz with 4GB of memory, running Microsoft Windows Server 2003, Standard Edition. We used one of the computers to generate the load with The Grinder, another to run Oracle Containers for J2EE (OC4J), and the last to run the database. All three were connected to a switched network in which the only traffic was generated by the tests themselves.

Data Transfer Object

Using JazzCat application to test the DTO design pattern, we created a servlet that lists all the albums in our test database via an entity bean, where each row in the listing shows the field values of the album entity. We started the EJB 2.1 test by programming this functionality without using DTO. In this case, which we called "DTOff", our test servlet would retrieve a list of all the album entities, and then get the individual field values for each entity via the entity's accessor methods.

Using the DTO design pattern, our test servlet, called "DTOn", makes a single method call (getData). The entity bean constructs a corresponding DTO, loads it with the entity's field values, and returns the object to the servlet. Now the servlet can access the fields on the local object.

In EJB 3, the entity beans have been replaced by Plain Old Java Objects (POJOs) and all the traditional CRUD and query operations are performed with the Entity Manager (EM). Queries can be defined as standalone named queries, which are predefined in the bean class, or dynamic queries that can be constructed using the query method.

In the JazzCat application, the dispatch servlet of the test harness calls the corresponding test servlet, which in turn looks up the entity manager.

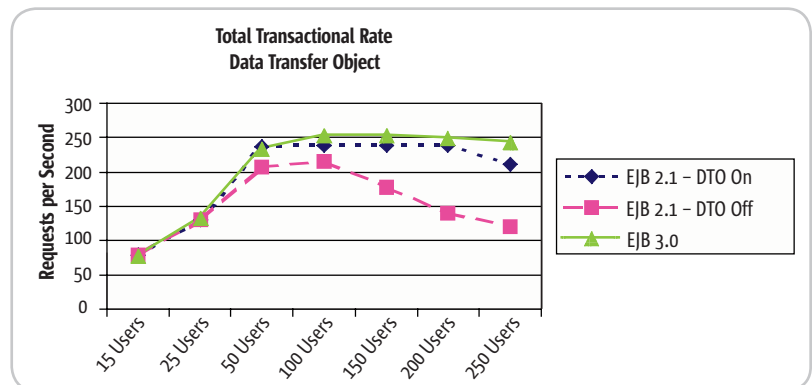


Figure 1

Peter Zadrozny is vice president and chief evangelist for Oracle Application Server. Previous to this, he was the chief technologist of BEA Systems for Europe, Middle East, and Africa. He held this role since starting WebLogic Inc.'s operations in Europe in 1998. Prior to his position at BEA, Peter held various executive and technical roles in many countries around the world for companies such as Macromedia, McKesson, Electronic Data Systems, Petroleos de Venezuela and Sun Microsystems, for whom he started their operations in Mexico.



We thought that the best way to test was to do a few things that developers typically do with EJB 2.1 and then try out the equivalent with EJB 3.0”

The entity manager then executes the named query that was defined for the Albums class. Here’s the definition of this class and the named query:

```
@Entity
@Table(name="ALBUMS")
@NamedQuery(name="findAll",queryString="Select object(a) from Albums a")
public class Albums {
//.....
}
```

Once the named query returns the list of all the albums, it’s then printed out as an HTML table using the getter methods of Albums POJO.

We ran the tests with 15, 25, 50, 100, 150, 200, and 250 simultaneous users and we found that in both cases, response time and throughput, the difference between EJB 2.1 using DTO and EJB 3.0 was within 4%. This is within the margin of error, so for all practical purposes the performance is similar. To give you an idea of the actual performance, with 250 simultaneous users the average aggregate response time was 1,100 milliseconds and the throughput was an average 225 requests per second. This is very impressive, especially since there was no sleep time in the test scripts.

The DTOOff test case produced an AART 28% higher than DTOOn and EJB 3, and a TTR that was 19% lower. This is because each call to an

accessor method on the entity bean is a transaction by itself, whereas using DTO and EJB 3, there is only one transaction. Remember to demarcate your transactions correctly to get better performance.

It’s interesting to note the throughput curves for each case (see Figure 1). Remember that the TTR or throughput is a measure of system capacity as a whole (application, JVM, database, OS, hardware). The curves start by going upward, which shows the system is handling the capacity as the requests increase. Then they reach a point of stability where the system has reached its limit and the response time starts to increase dramatically. Finally, when the curves begin to fall, this is where the system can’t handle any more load and things start going downhill. In this chart you can see that in all cases, the peak has been reached at 50 simultaneous users; however, on the DTOOff case performance goes downhill at 150 users, whereas EJB 3 and DTOOn hold on longer before going down. Interestingly, EJB 3 holds on a little longer by presenting 12% better throughput than DTOOn with 250 users. This is likewise for the response time, where EJB 3 has 13% better number than DTOOn with 250 users. You can say that under heavy loads, EJB 3.0 tends to perform better than EJB 2.1, at least in this test case.

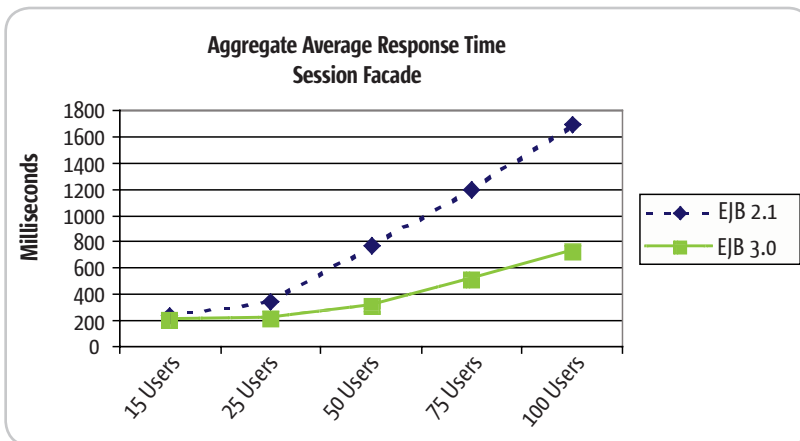


Figure 2

Session Façade

To test this design pattern we searched JazzCat catalog for albums containing a particular song title or a substring of it. This was an interesting case, as we had to call a few entities to provide this listing: Album (the item we are searching for), Take (holds the song title), Track (identifies occurrences of Takes on Albums), and Musician (identifies the artist name for an Album).

This is straightforward with EJB 2.1. The dispatcher servlet of the test harness calls the FacadeOn servlet, which in turn calls the stateless session bean that acts as the façade and calls all the necessary entities to obtain the required information. This is later sent back to the FacadeOn servlet in the form of a Data Transfer Hash Map and presented to the user.

EJB 3.0 works pretty much the same until you get to the point of calling the entity beans. Instead, we call the Entity Manager and, using named queries, we load the information from the POJOs into the Data Transfer Hash Map. The main difference is that with EJB 2.1, we look up the entity beans using ejb-local-refs and then work on these entity beans to get the required information.

We conducted these tests with 15, 25, 50, 75, and 100 simultaneous users. The test script did 10 different searches starting with the letter a to the letter j. To our surprise, we found that EJB 3.0 had roughly double the performance of EJB 2.1. Let’s look at the response time in Figure 2. We can see that as the load increases, the difference in response time increases dramatically, from 18% for 15 users all the way up to 58% for 100 users, giving an overall average of 46%.

Even more interesting is the TTR chart in Figure 3, which clearly shows the throughput for EJB 2.1 Session Façade more or less stable for all user loads, but decreasing very slowly at 25 users. In the case of EJB 3.0, it peaks at 50 users and then begins a slow descent. However, the average throughput of EJB 3.0 is roughly double that of EJB 2.1.

Raghu R. Kodali is a consulting product manager and SOA evangelist for Oracle Application Server. He leads next-generation SOA initiatives and J2EE feature sets for Oracle Application Server, with particular expertise in EJB, J2EE deployment, Web services, and BPEL. He holds a master’s degree in computer science and is a frequent speaker at technology conferences. Raghu maintains an active blog at Loosely Coupled Corner (<http://www.jroller.com/page/raghukodali>).

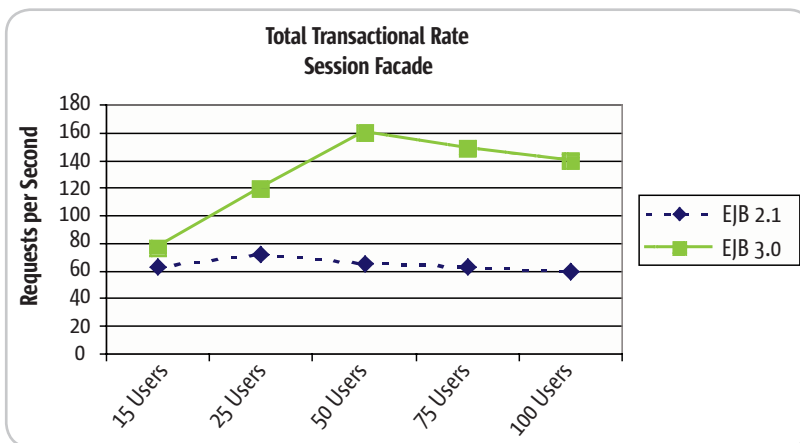


Figure 3

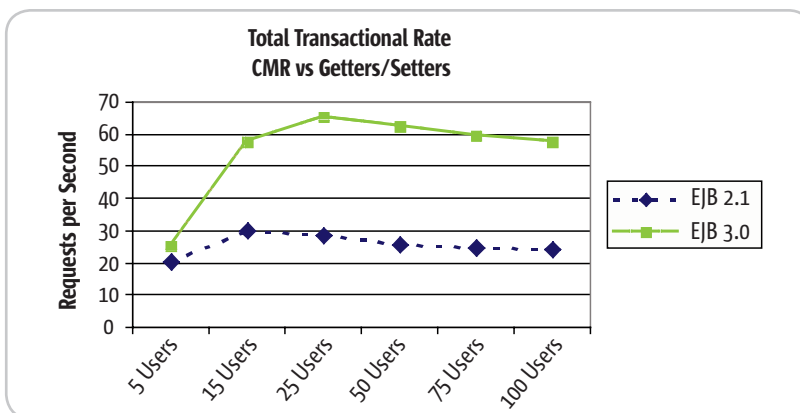


Figure 4

Just to provide some numbers, with 100 users the AART of EJB 3.0 is 716 milliseconds and the throughput is 140 requests per second. The wonderful performance of EJB 3.0 for such a popular design pattern is a welcome surprise, which only makes us want to use EJB 3.0 even more.

Container-Managed Relationships

Although entity beans in general and CMR in particular have a bad reputation, various EJB containers actually have high-performance implementations. Because of this, we expected the performance comparison would be basically the same.

This test case is based on a search that exercises the relationships between the tables of the JazzCat application. Given a song title, or a substring of it, to search for, the test servlet displays a list of takes matching the title, along with the band members playing on each particular take of that title. Among the many relationships and navigations that exist in this application and this search,

a couple are worth highlighting: there's a one-to-many relationship between Takes and Bands (a Take has many Band members), and a many-to-one relationship between Bands and Musicians (several different Bands can have the same Musician as a member).

Compared to programming this search using Bean Managed Persistence Entity Beans, using Container Managed Persistence is really easy. However, we still have to deal with the Deployment Descriptor, where we have to declare all the relationships of each entity.

In contrast, using the annotations of EJB 3.0, we can quickly and easily define the relationships, such as the one between Takes and Bands:

```
@OneToMany(targetEntity="org.migrate.entity.
Bands")
@JoinColumn(name="BANDS.TAKE_ID", referencedColumnName="TAKES.ID")
public List<Bands> getBandsList()
{
    return bandsList;
}
```

The EJB 3.0 test code is very similar to that of EJB 2.1 up to the point where the session façade bean calls the entity manager. Using the entity manager, we navigate through the POJO relationships using getter methods and we assemble the information for the test servlet to present to the user.

The test scripts for this test case were basically the same as those used for the Session Façade test case: 10 searches starting with the letter "a" up to the letter "j". We began with five users and then increased to 15, 25, 50, 75, and 100 users. (We had to add the case of five users because the throughput of EJB 2.1 peaked at 15 users.)

The results again were pleasantly surprising: the performance of EJB 3.0 was roughly double that of EJB 2.1. The chart for the response time looks very similar to that of the Session Façade test case – the difference starts with 20% for five users and pretty much jumps to 55% for the rest of the user load.

The results from the TTR standpoint can be seen in Figure 4. Here, the throughput of EJB 3.0 starts with a 25% difference and quickly grows to 140% to present an overall average of 111% better throughput.

If you're interested in running your own test cases using JazzCat application, the test harness, and the methodology we used, you can download a package that contains everything you need from www.jroller.com/resources/r/raghukodali/jazzcat.jar.

Conclusion

Considering that we used only a developer's preview of EJB 3.0, we're very impressed with this specification – at least Oracle's implementation of it. As mentioned earlier, we're very attracted to the simplicity and power of EJB 3.0. The wonderful work done with annotations and the persistence, the ability to use POJOs, and the ability to test outside of the container are very attractive all by themselves. But now, with an implementation that equals or doubles the performance of EJB 2.1 (at least for our test cases), we can't wait for the final specification and formal release of EJB 3.0.

Acknowledgement

Thanks to Phil Aston for his contributions with The Grinder 3. ☺

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

Formula One e.Spreadsheet Engine

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

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

Excel-enable your Java applications

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

Embed live, Excel-compatible data grids

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

Automate complex calculations and rules

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

Read and write server-based Excel spreadsheets

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

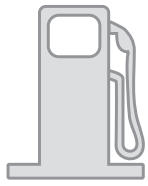
Make spreadsheets a part of your strategies

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

ACTUATE
ReportingEngines™

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

**FREE TRIALS,
DEMOS AND
SAMPLE CODE**



Dealing with Open Source Software

Part Two

by Yakov Fain

In the first article of this series (see <http://java.sys-con.com/read/108260.htm>), I “bought” a gas station with a convenience store and a repair shop and started to think about automating this small business using various Java technologies. This time, I’m getting a crash course on open source software.

Mentality Shift

The most surprising thing is how quickly my programming preferences have changed after I left the corporate world and started working at my gas station. I used to easily recommend expensive software tools, application servers, RAID devices, grid servers, and fiber optic connectors. Need scalability? No problem. We’ll create a cluster of two 8-CPU application servers. Let’s allocate another \$100K for the server licenses for our development, UAT, and contingency environments... But now I’m buying coffee beans in bulk quantities for my convenience store trying to save a couple of dollars. I’m wondering if CIOs of large firms know how much money has been spent just on unused software licenses? Oh well, it’s none of my business, I’ve got a customer: “\$20, regular, credit card”... Oops! The name on the card reads Bernard Golden.



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

“Excuse me, sir! Are you by any chance the author of the best-selling book *Succeeding with Open Source*?”
“Yes I am, but what makes me famous at a gas station?”

“I’m pumping gas during the day, but I read IT books and write Java programs in the evening. At this point I’m trying to figure out if I can use the open source software in my small business. Please help.”

Bernard kindly agreed and here’s our conversation.

What Is Open Source Software?

Yakov: *Professional programmers make a living by developing software. On the other hand, there is large community of people who are willing to donate a couple of hours here and there to add a piece of code to an open source project without being paid for it. Why would they do this?*

Bernard: There are a number of common reasons, and each contributor has his or her own combination of those reasons. First and foremost, contributors are often donating a piece of code that makes the product work better for them. Because the source of the product is available, they can improve it to better run in their environment. Contributing it makes sense because that way their fix becomes part of the mainstream code base and they don’t have to reapply their changes to each future release of the product.

Other contributors do so out of interest in the technology – contributing allows them to explore a personal interest – and all of us benefit from their interest.

Other contributors are developing their skill set to enable them to obtain a better job or set them up for more interesting work in the future.

What’s great about open source is it enables all of us to benefit from those individuals’ creativity and passion.

Yakov: *In 1999, I was hired by a large firm for a WebSphere project even though WebLogic was more popular back then. When I asked the manager why they chose IBM’s application server, he replied, “We want to be sure that the company will be still around in 10 years.” What if the key developers of a particular open source project lose their interest and stop working on it?*

Bernard: The truth is you don’t know if the key developers of a particular project will choose to remain involved with it.

It makes sense to look at the size and quality of the development team to form an opinion about its capability and likely dedication to the project. That said, there is a pretty strong ethos in the open source community to arrange for a successor in the project, and most (if not all) projects with a strong community never face the issue of developer abandonment.

Yakov: *Does the term open source software really mean free software?*

Bernard: Free is an ambiguous term in English, meaning both no cost and complete liberty. Open source is, at base, about liberty: liberty of use, liberty to modify, liberty to redistribute. Nothing in open source licenses forces the software to be distributed at no cost; however, the realities of unlimited distribution typically mean that software is distributed without license fees. Of course, many companies like Red Hat, JBoss, and Zope distribute their open source product with no license fee, but charge for additional services like support and training.

Yakov: *I’m too busy with my customers and won’t have time to contribute to any open source project. But I’d like to have all these liberties at no cost for automation of my small business. To be honest with you, I’m also thinking of selling some of my solutions to my fellow gas station owners. What type of software licenses fits my objectives?*

Bernard: There is no blanket – or should I say mechanical! – answer. Open source licenses, broadly speaking, fall into two camps: so-called Berkeley licenses and GPL-style licenses. Berkeley-style licenses allow recipients of open source software to modify and redistribute without making the modified source code available, while GPL-style licenses require source availability if the modified open source product is itself distributed.

Get the complete picture



Discover the New

ILOG JViews Graphics Components

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

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

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

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

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

- ILOG JViews Diagrammer <http://diagrammer.ilog.com>
- ILOG JViews Charts <http://charts.ilog.com>
- ILOG JViews Gantt <http://gantt.ilog.com>
- ILOG JViews Maps <http://maps.ilog.com>
- ILOG JViews TGO <http://jtgo.ilog.com>



Changing the rules of business™

Depending on whether you're modifying an existing open source product or creating your own product from scratch, and even how the product is architected and distributed, makes a difference about the requirements and implications of the license. Before doing anything, you should understand these issues. Otherwise, you may find that your commercial product is really an open source product, with all that entails. An excellent resource for this subject is Larry Rosen's book *Open Source Licensing: Software Freedom and Intellectual Property Law* (Prentice Hall PTR, 2004).

Yakov: *Should I be aware of any potential legal risks in this area? Say, for example, I've changed 10 lines of code of some free Java component to improve its functionality. Do I have to make these changes available to the entire world to avoid legal issues?*

Bernard: The rule of thumb in open source is that any changes you make and use internally are perfectly fine. The challenge comes if you distribute the modified product, depending upon the license of the original product. Again, Rosen's book can provide a lot of help in this area.

Yakov: *How can I assess which open source tool is the best in a given category, e.g., Java Web frameworks?*

Bernard: It's easy to ask around or attend a local user group meeting and pick attendees' brains. You can get a good sense of the functionality, liveliness, and hospitality of the community by monitoring the mailing lists/forums. These methods can give you a good informal feel for the various products.

If you want a more formal assessment, you can use the Open Source Maturity Model (OSMM) developed by my company. It's a structured assessment methodology for open source products to determine their maturity and usefulness for organizations. The OSMM is open source, and more can be learned at www.navicasoft.com/OSMM.

Yakov: *As everyone else, open source developers should hate writing documentation. Besides, they don't have any motivation to write quality manuals. Am I supposed to be a hacker to learn the open source products?*

Bernard: The quality and availability of documentation for a given open source product tends to improve in lockstep with its overall maturity. Immature products typically have incomplete and/or poorly written documentation, while more mature products benefit from a larger user community, some of whom write and contribute documentation. Of course, the very best established open source products have user communities large enough to attract the attention of commercial publishers. It's often the case that you need to be very technically sophisticated to use immature open source products due to the paucity of quality documentation.

Yakov: *The Internet is also free, but there are plenty of firms that offer paid services such as Internet-related training and support. Since the open source products become widely used, most likely a new breed of startups will appear to offer training and support of popular open source tools.*

Bernard: Just as the quality and availability of documentation improves as the maturity of an open source product improves, so to does the availability of product-oriented services. Both documentation and services are assessed as part of the OSMM; in addition, support, training, and product integration go into the OSMM assessment process. All are vital for mainstream use of an open source product.

Yakov: *I remember that during the dot-com era, small companies were raising money from investors by showing them a business plan and promising a bright future. Don't you feel that we can expect a similar trend in the open source arena? If a company has created a piece of a free software and can show that it's been downloaded several thousand times,*

they can raise capital by promising sales of support and training in the future.

Bernard: I work a lot with open source startups, and it's true that there's a lot of venture activity in open source today. Number of downloads is one metric investors look at as part of their company assessment, but they also evaluate the founder(s), the importance of the software in terms of the typical use scenarios, as well as the potential business model of the company.

One advantage from the user perspective is that open source breaks the linkage between product and company. If the product is useful but the company is unsuccessful, open source means that the user community will still have access to the product source base and can continue using the product even if the company goes bust. This is very different from the dot-com days, when companies would shut down and leave their users in the lurch.

Yakov: *I'm the only person here who will make all decisions, deal with the vendors, and develop the applications for my business. Do you see it as an advantage or disadvantage when it comes to implementation of the open source products?*

Bernard: Well, you'll be in control of all the decisions, but you'll carry a lot of responsibility. Because you're on your own, you should focus on open source products with large and active communities so that you'll have resources to call upon when you face challenges.

Yakov: *Bernard, thank you for this improvised open source training! From now on, if I'll get anything at no cost, I'll call it an open source deal. I wish my suppliers could offer me some open source gasoline...*

Bernard: Good luck!

• • •

I really appreciate all your comments to my previous gas station column posted online. Please share your thoughts on the use of the open source software in a small business at the online version of this article at <http://java.sys-con.com/read/124664.htm>. ☺



Open source is, at base, about liberty:
liberty of use, liberty to modify, liberty to redistribute”

Style Report 7

Light Weight, Integration Ready Enterprise Reporting & Analysis



open standards open architecture Linux Windows Java J2EE Tomcat SOAP LDAP DHTML

- Traditional BI Challenges:**
- ▶▶ Monolithic, resource intensive
 - ▶▶ Proprietary software stack
 - ▶▶ Requires ETL/Data warehouse

- Style Report Solutions:**
- ▶▶ Light weight, integration ready
 - ▶▶ Open software stack, J2EE drop-in
 - ▶▶ Real time transactional DB and OLAP



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

Intelligent Web Applications with

AJAX

A peek into modern technologies for browser-based applications

by Victor Rasputnis,
Anatole Tartakovsky, and Igor Nys

Browser-based applications are widely used and we like the fact that we can access them from anywhere. But from the users' perspective, the productivity level of Web applications still doesn't approximate the productivity of desktop programs. The good news is the gap is closing: the accumulated potential of multiple technologies has boosted a whole new breed of HTML-based apps that are as powerful as the desktop ones. Meet AJAX.

What Is AJAX?

The name stands for Asynchronous JavaScript + XMLHttpRequest and means you can establish socket communication between browser-based JavaScript and the server. AJAX isn't a new technology. It's a successful branding of possibilities implanted in several technologies available in modern browsers. All AJAX applications deliver a rich UI via extensive JavaScript manipulation of the HTML Document Object Model based on the precision-pointed data retrieval via XMLHttpRequest. Typical examples of AJAX applications are Google Maps and Google Suggest from Google Labs (<http://labs.google.com>). These applications actively monitor user input and provide real-time page updates. Most importantly, this happens without a page refresh while the user navigates through the map or types a search string.

In fact, the technologies behind these wonders have been around for a while, although they require sophisticated skills in using browser-specific tricks. Proprietary offerings with similar capabilities – Macromedia Flash plug-in, Java Applets or .NET runtime – have been available for quite some time too. The idea of integrating a scriptable transport component talking to the server into the browser was pioneered by IE 5.0. Then Firefox and other popular browsers joined the club of browsers in supporting XMLHttpRequest as a built-in object. With cross-browser availability, these technologies gained visibility and in March of 2004 a company called Adaptive Path introduced AJAX.

In short, backing from Google and having the right browser technologies available out-of-the-box tipped the scale: these days everyone is adding client-side technologies to Web applications for a “better user experience.”

AJAX vs. Classical Applications

A classic Web application model is literally a triumph of form over substance: users are forced to submit forms in exchange

for pages. That said, the form submission and page delivery aren't guaranteed: worse case the user clicks again though some pages specifically warn against that. It's quite different with AJAX, where the data travels across the wire instead of entire HTML pages. This data exchange is scripted via a specific browser object – XMLHttpRequest; the appropriate logic handles the outcome of each data request, the specific region of the page is updated instead of the entire page. The results are more speed, less traffic, and better control of information delivery.

Traditional “click-refresh” Web applications force users to interrupt the work process while waiting for the page to reload. With AJAX, a client-side script can asynchronously talk to the server while the user keeps entering data. Besides being transparent to the user, such asynchrony means more time for the server to process the request.

Classic Web applications delegate all processing to the server and force the server to manage the state. AJAX allows flexible partitioning of the application logic and state management between the client and the server. This eliminates a “click-refresh” dependency and provides better server scalability. When the state is stored on the client-side you don't have to maintain sessions across the servers or save/expire state: the lifespan is defined by client.

AJAX: Distributed MVC

Although AJAX applications rely on JavaScript for the presentation layer, the processing power and knowledge base remain on the server. For that matter, AJAX applications talk heavily to J2EE servers, feeding data to and from Web Services and servlets. The difference between J2EE applications with an AJAX-based presentation tier and standard J2EE application is that in the first case MVC is distributed over the wire. With AJAX, View is local, while Model and Controller are distributed giving the developer the flexibility to decide which components will be client-based. Specifically, a local View renders graphics by manipulating with HTML DOM; the controller handles user input locally and at the developer's discretion extends the processing to the server via HTTP requests (Web Services, XML/RPC or others); the remote part of the Model is downloaded as needed to the client achieving in-place real-time updates of the client page; and state is collected on the client.

In future AJAX articles we'll talk about each of these components in depth and provide examples of how they came to play



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

victorasputnis@teamcti.com

together. Now, without further ado, let's dive into a simple AJAX example.

Zip Codes Validation and Lookup

We'll create an HTML page containing three INPUT fields: Zip, City, and State. We'll make sure that as soon as the user enters the first three digits of the zip code, the state will get populated with the first matching state value. Once the user types in all five zip digits, we'll instantly determine and populate the appropriate city. If the zip code isn't valid (not found in the server's database), we'll turn the zip's border color to red. Such visual clues are helpful to users and have become standard in modern browsers (as an example, Firefox finds matching words in an HTML page by highlighting them in the browser search field while you type).

Let's start with a simple HTML containing three input fields: zip, city, and state. Please note that the method `zipChanged()` is called as soon as a character is entered in the zip field. In turn, the JavaScript function `zipChanged()` (see below) calls the function `updateState()` when the zip length is three and `updateCity()` when the length of the zip is five. Both `updateCity()` and `updateState()` delegate most of the work to another function – `ask()`.

```
Zip:<input id="zipcode" type="text" maxlength="5"
onKeyUp="zipChanged()" style="width:60"/>
City: <input id="city" disabled maxlength="32" style="width:160"/>
State:<input id="state" disabled maxlength="2" style="width:30"/>

<script src="xmlhttp.js"></script>
<script>
var zipField = null;
function zipChanged(){
    zipField = document.getElementById("zipcode")
    var zip = zipField.value;
    zip.length == 3?updateState(zip):zip.length == 5?updateCity(zip):"";
}
function updateState(zip) {
    var stateField = document.getElementById("state");
    ask("resolveZip.jsp?lookupType=state&zip="+zip, stateField, zipField);
}
function updateCity(zip) {
    var cityField = document.getElementById("city");
    ask("resolveZip.jsp? lookupType=city&zip="+zip, cityField, zipField);
}
</script>
```

The function `ask()` communicates with the server and assigns a callback to process the server's response (see the following code). Later, we'll look at the content of the dual-natured `resolveZip.jsp` that looks up the city or state information depending on the number of characters in the zip field. Importantly, `ask()` uses the asynchronous flavor of the `XmlHttpRequest` so that populating the state and city fields or coloring the zip border is done without slowing data entry down. First, we call `request.open()`, which opens the socket channel with the server using one of the HTTP verbs (GET or POST) as the first argument and the URL of the data provider as a second one. The last argument of the `request.open()` is set to true, which indicates the asynchronous nature of the request. Note that the request hasn't been submitted yet. That happens with the `request.send()`

call, which can provide any necessary payload for POST. With asynchronous requests we have to assign the request's callback using the `request.onreadystatechange` attribute. (If the request had been synchronous, we could have processed the results immediately after `request.send`, but we would have blocked the user until the request was completed.)

```
HTTPRequest = function () {
    var xmlhttp=null;
    try {
        xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (_e) {
        try {
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (_E) { }
    }
    if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
        try {
            xmlhttp = new XMLHttpRequest();
        } catch (e) {
            xmlhttp = false;
        }
    }
    return xmlhttp;
}

function ask(url, fieldToFill, lookupField) {
    var http = new XMLHttpRequest();
    http.open("GET", url, true);
    http.onreadystatechange = function () { handleHttpResponse(http,
fieldToFill, lookupField)};
    http.send(null);
}

function handleHttpResponse(http, fieldToFill, lookupField) {
    if (http.readyState == 4) {
        result = http.responseText;
        if ( -1 != result.search("null") ) {
            lookupField.style.borderColor = "red";
            fieldToFill.value = "";
        } else {
            lookupField.style.borderColor = "";
            fieldToFill.value = result;
        }
    }
}
} } }
```

The `HttpRequest()` function (see above) used by `ask()` is a cross-browser constructor of an instance of the `XMLHttpRequest`; we'll look at it a bit later. For now, note how the invocation of `handleResponse()` is wrapped by an anonymous function (a so-called *closure*) `function () { handleHttpResponse(http, fieldToFill, lookupField)}`.

The code for that function is dynamically created and compiled every time we do an assignment to the `http.onreadystatechange` property. As a result, JavaScript creates a pointer to the context with all variables that the enclosing method – `ask()` – has access to. It's done so the anonymous function and `handleResponse()` are guaranteed full access to all context-hosted variables until the reference to the anonymous function is garbage-collected. In other words, whenever our anonymous function gets invoked, it can refer to the `request`, `fieldToFill`, and `lookupField` variables as seamlessly as if they were global. It's also true that every invocation of `ask()` will create a separate copy of the environment with the variables holding the values of the moment the closure was formed.



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

anatolet@teamcti.com



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

igordnys@gmail.com

Let's look at the function `handleResponse()`. Since it can be invoked at different states of the request processing, the function ignores all cases except the one when the request processing is complete. This corresponds to the `request.readyState` property equal to 4 ("Completed"). At this point the function reads the server's response text. Contrary to what its name may suggest, neither the input nor the output of XMLHttpRequest has to be restrained to XML. In particular, our `resolveZip.jsp` (see Listing 1) returns plain text. If the return value is "unknown" the function assumes that the zip code was invalid and changes the border color of the lookup field (zip) to red. Otherwise, the return value is used to populate the fill field (state or city), and zip's border is assigned a default color.

XMLHttpRequest – the Transport Object

Let's return to our cross-browser implementation of XMLHttpRequest. The last listing contains an `HttpRequest()` function that's upward-compatible with IE5.0 and Mozilla 1.8/FireFox. For simplicity's sake, we just try to create a Microsoft XMLHttpRequest object – and if that fails we assume it's Firefox/Mozilla.

At the heart of this function is the XMLHttpRequest – a native browser object, which facilitates anything that involves HTTP protocol in communicating with the server. It allows specifying any HTTP verbs, headers, and payload and works in either asynchronous or synchronous mode. No downloads or plugins are required, although in the case of IE, XMLHttpRequest is an ActiveX integrated inside the browser. Accordingly, the "Run ActiveX Control and Plugins" default IE permission should be in place to use it.

Most important, XMLHttpRequest allows an RPC-style programmatic query to the server without any page refresh. It does it in a predictable, controlled way, offering complete access to all details of the HTTP protocol, including the headers and any custom formatting of the data. In future articles, we'll show you industrial protocols that you can run on top of this transport including Web Services and XML-RPC that greatly simplify developing and maintaining large-scale applications.

The Server-Side Logic

Finally, the server-side `resolveZip.jsp` is invoked from the function `ask()` as shown in Listing 1. The `resolveZip.jsp` is called in two separate scenarios differentiated by the current length of the zip code (see the `zipChanged()` function.) The value of the request parameter `lookupType` is either state or city. For simplicity's sake, we'll assume that two files, `state.properties` and `city.properties`, are located in the root directory of the c: drive of the server. The `resolveZip.jsp` logic is confined to returning the lookup value with the appropriate pre-loaded file – once in each case of course.

Our AJAX-enabled page is ready. The complete working example is available at: <http://www.ajaxmaker.com:8080/blog/zipsearch.htm>.

Remote Scripting – An Alternative Approach

Some older AJAX implementations are based on so-called remote scripting. The idea is that the user's actions result in querying the server via IFRAME, and the server responds with the JavaScript, which is immediately executed as soon as it reaches the client. This is a big difference compared to XMLHttpRequest

request approach, where the server responds with the data and the client interprets the data. The advantage is that this solution supports older browsers.

The HTML portion of the IFRAME-based example (see Listing 2) is similar to the one we've used in the XMLHttpRequest scenario, but this time we'll introduce an extra IFRAME element – *controller*:

```
Zip:<input id="zipcode" type="text" maxlength="5"
onKeyUp="zipChanged()" style="width:60" size="20"/>
City: <input id="city" disabled maxlength="32" style="width:160"
size="20"/>
State:<input id="state" disabled maxlength="2" style="width:30"
size="20"/>
<iframe id="controller" style="visibility:hidden;width:0;height:0"></
iframe>
```

We keep calling `zipChanged()` per every key stroke, but this time the function `ask()`, called from `zipChanged()` (see Listing 3), sets the IFRAME's `src` property, instead of invoking an XMLHttpRequest:

```
function ask(url, fieldToFill, lookupField)
{
  var controller = document.getElementById("controller");
  controller.src = url+"&field="+fieldToFill.id+"&zip="+lookupField.id;
}
```

The server-side logic is presented by a sketchy `resolveZip.jsp` (see Listing 4). It's different from its XMLHttpRequest counterpart in that it returns JavaScript statements, which set the global values of the variables `field lookup` and `city` and the call function `response()` from the global window's execution context as soon as it gets to the browser. (Listings 5–7 can be downloaded from www.jdj.sys-con.com.)

The function `response()` is a modified version of the `handleResponse()` which is absolved from dealing with uncompleted requests (see Listing 2.)

The Fine Print

For simplicity's sake, we've "overlooked" some important issues in our sample code:

1. The fact that the instances of the XMLHttpRequest object and callback invocations haven't been destroyed after being used, which causes memory leaks after every call. Properly written code should destroy or reuse such instances in the object pool. Object management techniques common to the server software have to be used for the client
2. In quite a few places the errors weren't handled properly. For example, the call to `request.open()` in the method `ask()` can throw an exception that has to be caught and processed even though JavaScript exceptions don't have to be checked. The `handleResponse()` function is another example. It has to check `headers` and `responseText` for possible server-side and communication errors. In case of an error, it has to try to recover and/or report an error. Properly developed AJAX applications eliminate losing data on "submissions" due to disconnects and other low-level communication problems via a robust, self-recovering framework.

IT CAN THINK
AND DECIDE..
On its own.



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



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

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

- Automatically optimizes WebLogic server, saving you time and effort.
- Detects problems automatically.
- Gives advice from a dynamic knowledge base.
- Provides WebLogic system state for analysis in the event of a catastrophic failure.
- Optimizes existing J2EE resources.
- Increases the life, effectiveness, and

quality of existing resources by reducing or eliminating problem areas.

- Improves resource communication and response time.
- Enhances server capacity to handle more loads and transactions.
- Provides advanced system monitoring.
- Furnishes customizable reports.
- Generates email alerts.

- Cost Effective at only \$999 per IP address, regardless the number of CPUs

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



AUTOPILOT™
AUTO OPTIMIZER

3. Current server-side frameworks provide quite a few functions that have to be reconciled with a refresh-free approach. For example, let's consider a custom server-side authentication with a timeout. In that case we'd have to intercept security system response to the XMLHttpRequest calls, bring up the login screen, and then re-issue the request after the user was authenticated.

All these problems are typical of any application code working with low-level APIs and all of them can be resolved. The good news is that the technologies needed to resolve these issues are quite familiar to most Java developers like Web Services, custom tags, and XML/XSLT. The only difference is that nowadays these technologies come to the rescue on the client in the form of:

- Web Services using SOAP/REST/RPC for a simple communication standard
- Client-side custom tags for packaging rich client-side con-

- trols with integrated AJAX functionality
- XML- and XSLT-based data manipulation

JDJ will publish more articles on AJAX where you'll learn how to use these technologies to make AJAX solutions both simple and robust.

Summary

The AJAX approach offers a rich Internet experience on a par with that of desktop applications. AJAX features have to be applied selectively: you definitely don't want your credit card charged by the background process while you're still shopping. Is AJAX momentum sustainable? We certainly hope so. We've been developing AJAX applications for the last five years and can attest that it's sound and very effective. However, it requires that a developer be exposed to a much wider set of technologies than the ones used in the traditional "click-refresh" Web applications. ☺

Listing 1: resolveZip.jsp

```
<%@ page info="resolveZip" %>
<%@page import="java.util.Properties"%>

<%
String lookupType = request.getParameter("lookupType");

Properties properties = (Properties)getContext().getAttribute(lookupType);

if (properties == null)
{
properties = new Properties();
try {
properties.load(new java.io.FileInputStream("c:\\"
+ lookupType + ".property"));
getContext().setAttribute(key, properties);

} catch (java.io.IOException e) {
out.println("Property File not found");
}
}
out.println(properties.getProperty(request.getParameter("zip")));
%>
```

Listing 2: zipsearch.html (IFRAME)

```
Zip:<input id="zipcode" type="text" maxlength="5"
onKeyUp="zipchanged()" style="width:60" size="20"/>
City: <input id="city" disabled maxlength="32" style="width:160"
size="20"/>
State:<input id="state" disabled maxlength="2" style="width:30"
size="20"/>
<iframe id="controller" style="visibility:hidden;width:0;height:0"></
iframe>

<script src="iframe.js"></script>
<script>
var zipField = null;
function zipchanged(){
zipField = document.getElementById("zipcode")
var zip = zipField.value;
zip.length == 3?updateState(zip):zip.length == 5?updateCity(zip):"";
}
function updateState(zip) {
var stateField = document.getElementById("state");
ask("resolveZip.jsp?lookupType=state&zip="+zip, stateField, zipField);
}
function updateCity(zip) {
var cityField = document.getElementById("city");
ask("resolveZip.jsp?lookupType=city&zip="+zip, cityField, zipField);
}
</script>
```

Listing 3: iframe.js

```
function response(result, fieldToFill, lookupField)
{
var lookup = document.getElementById(lookupField);
var fill = document.getElementById(fieldToFill);

if ( result == "null" ) {
lookup.style.borderColor = "red";
fill.value = "";
} else {
lookup.style.borderColor = "";
fill.value = result;
}
}

function ask(url, fieldToFill, lookupField)
{
var controller = document.getElementById("controller");
controller.src = url+"&field="+fieldToFill.id+"&lookup="+lookupField.id;
}
}
```

Listing 4: resolveZip.jsp (IFRAME)

```
<%@ page info="resolveZip" %>
<%@page import="java.util.Properties"%>

<script>
<%
String lookupType = request.getParameter("lookupType");

Properties properties = (Properties)getContext().getAttribute(lookupType);

if (properties == null)
{
properties = new Properties();
try {
properties.load(new java.io.FileInputStream("c:\\"
+ lookupType + ".property"));
getContext().setAttribute(key, properties);

} catch (java.io.IOException e) {
out.println("Property File not found");
}
}

out.println("var field= " + request.getParameter("field") + "'");
out.println("var lookup= " + request.getParameter("zip") + "'");
out.println("var result= " + properties.getProperty(request.
getParameter("zip")) + "'");
%>
window.top.window.response(result, field, lookup);
</script>
```




Powering Eclipse to the next level...

NitroX™

PROFESSIONAL TOOLS FOR ECLIPSE™

JSP | Struts | JSF

Professional development for JSP, Struts & JSF

Simultaneous visual/source Struts & JSF configuration

Code completion for all levels

Automatic validation & error checking for all levels

Debugging support for all levels

AppXRay, AppXaminer, AppXnavigator

download at: www.m7.com/power



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

What Java Developers Can Learn from Boston's Big Dig...

by Jason Collins

Highway and Java enterprise application projects have much in common. Both can suffer from design flaws, stalled flow, and unforeseen performance glitches. But in the case of Java enterprise applications, performance management tools can help developers highlight potential problems before they derail these critical projects.

Java's infancy in the late '90s began as a free-for-all with simple applications that exposed information to an exploding number of users. As the Java language and Java developers matured, J2EE emerged and enterprises began a more cautious approach to developing Java applications. Applications moved away from simple presentation layers into a torrent of complex Java deployments.

Today we see massive, composite applications built on platforms like WebLogic, WebSphere, and SAP NetWeaver, changing the way the world conducts business. But they, like huge engineering projects, sometimes become so complex that it is difficult to gain visibility into all elements of the application, and to proactively manage it so that small problems are identified and solved before they impact performance and availability.

Are there lessons that Java application developers can learn from engineers

to ensure applications move smoothly from development through QA and into production?

Consider Boston's Big Dig, one of the largest American public engineering projects since the Hoover Dam was built in the 1930s. Consider its magnitude in terms of money (over \$14 billion thus far) and scale (diverting miles of highway underneath the city).

This project has become one of the biggest engineering missteps in recent history; the latest problem to beset the project is a flood of millions of gallons of water gushing from over 400 leaks breaching the tunnel walls. Moreover, the mistake wasn't fully realized until thousands were using the tunnel daily. The city of Boston is now knee-deep in problem analysis, fingerpointing among vendors, and questions such as: Could this have been predicted and prevented? Why didn't the project team isolate the problem during testing?

A business enterprise may be considering not so dissimilar engineering projects in mass and scale, investing millions of dollars in information technologies. And project goals are likely the same: optimize throughput and increase user satisfaction while minimizing time and cost.

Today, many projects that depend on integration and collaboration will rely on enterprise portals and J2EE technologies. Multiple silos within an enterprise can integrate seamlessly and share information. Reusable components lead to faster time to develop-

ment and more robust applications. The success of these projects is critical to the success of business processes – in many cases they are the business.

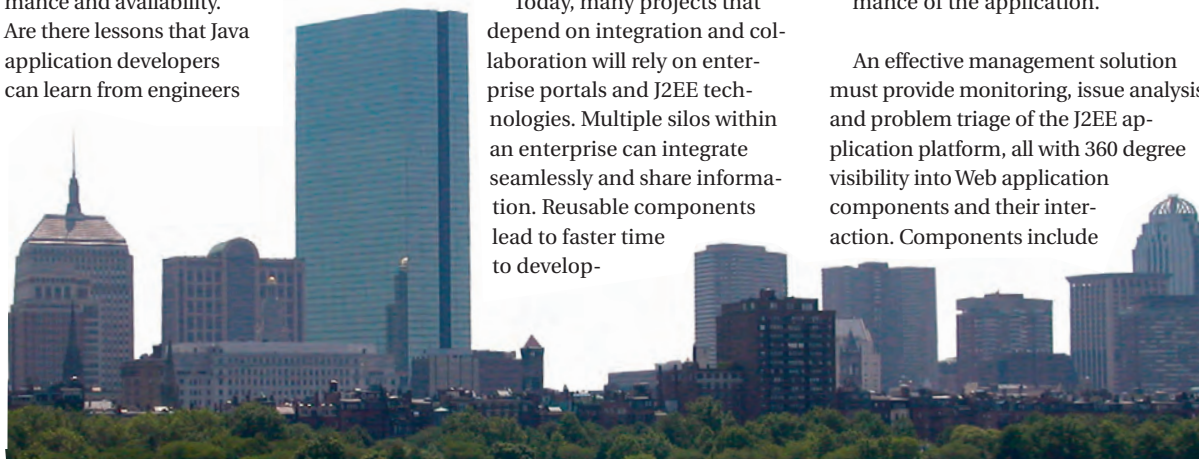
With so much at stake, how can companies isolate and eliminate the blunders that eluded Boston's engineers? How can they maintain quality and ensure that minor issues do not escalate into massive outages? How can they predict and eliminate problems before they affect their customers?

Proven Performance Management Solution

Based on Wily Technology's extensive experience helping its customers manage the performance of their J2EE environments, we have identified the following criteria as essential for effective application performance management:

- Monitor application health and availability 24x7 in real time.
- Isolate the cause of performance issues throughout the application environment, at each stage of the application life cycle, from the J2EE application itself to back-end systems.
- Provide customized, detailed data to everyone with a stake in the performance of the application.

An effective management solution must provide monitoring, issue analysis, and problem triage of the J2EE application platform, all with 360 degree visibility into Web application components and their interaction. Components include



Jason Collins is a systems engineer with Wily Technology in the San Francisco Bay Area.

Prior to joining Wily in 2002, Jason consulted with Computer Sciences Corporation and developed business technology implementations for Bay Area startup companies. He received his BS from Duke University in biomedical engineering.



Today, many projects that depend on integration and collaboration will rely on enterprise portals and J2EE technologies”

user response, network latency, the virtual machine, the J2EE application server database, messaging, Web services, and back-end applications.

Further, this information must be shared among development and operations within IT, and in some cases with the business units themselves. This requires the use of a single tool that draws on the same consistent data to produce highly customized dashboards, highlighting the specific areas of concern for each stakeholder.

Consider the Big Dig again. The 400 gushing leaks were once minor cracks. Cracks escalated into holes and holes into inch-wide fissures. Applications behave the same way; a management solution must detect minor issues in the application before they become critical outages.

Leaks can become as severe as system crashes every few hours in a production application. Java developers are not burdened by memory management, since the JVM does garbage collection. In many cases, however, data structures are misused in memory. Monitoring the most aggressively growing data structures is necessary to eliminate misuse in the source code.

J2EE applications have resources – called threads – to perform a unit of work. The number of threads in an application is limited, like the number of lanes on the underground highway. To move beyond the limit requires new hardware – a new tunnel or new CPU.

Construction or accidents can block highways. Application threads become stalled for a variety of reasons - back-end systems are down, resources are over-utilized, or the network is busy.

It is necessary to detect these stalled threads. Because performance response time is a lagging indicator, monitoring leading indicators such as stalls can predict and isolate problems before they escalate.

Measuring and Monitoring Flow

Imagine “concurrency” on a highway as the number of lanes holding a car within a highway cross-section. Concurrency for a highway can be predicted for the time of day but go unpredicted for events such as heavy rains. Similarly, applications are quantified by throughput – requests processed per second. Concurrency monitoring provides a deeper analysis of where application threads are busy.

Concurrency in an application can be base lined for a particular time of the day. When levels exceed normal, concurrency will show which application components are problematic. Historical performance data helps determine which components need to scale as user load increases. After all, the goal is to ensure and increase user acceptance.

Consider the Big Dig one last time. Critics blame the vendors for discovering the problems only after commuters started using the tunnel. Let’s assume that the vendors performed due diligence. Leaks could indicate the stress of both user load and time. Applications may also pass user-acceptance testing and system-integration testing, but usage-testing scenarios may not expose deficiencies.

Most problems occur only when real customers start to use the application. Retesting the application after implementation will not work either. That J2EE is the middleware between different systems increases the risks dramatically. A production-monitoring tool is needed to expose these problems – one that can provide full visibility with negligible overhead. An early warning system should alert operators and support personnel of potential problems before traffic is slowed or halted altogether.

J2EE projects face issues similar to those of large-scale engineering projects. Problems are inevitable, but nevertheless resolvable and often times predictable. Wily research shows that problems are seldom caused by the vendor platform, but often lie in back-end connections or custom development. The earlier those problems are detected, the less likely they will escalate into major setbacks. Problem mitigation becomes more difficult further along the project timeline. Projects can minimize these issues effectively by using a management solution that meets the requirements for successful application deployment.

Ultimately, projects that include performance management throughout the application life cycle are more likely to succeed and provide faster time to value.

Can you dig it? ☞

We’ve got problems with your name on them.

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

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

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

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

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





Calvin Austin

Core and Internals Editor

Help I'm Out of Memory!

Many years ago I saved up for a 16K RAM pack for my tiny Sinclair ZX81 computer. I thought, rather naively, that this was going to be the answer to all my memory issues. I would be able to use increasingly complex programs, okay games, and I could program without the restriction of literally making every byte count. I quickly found out, as you have already discovered if you have been writing Java applications for a while, that adding more memory to your machine is not always the answer to the running out-of-memory problem, the infamous "OutOfMemoryError". Dr. Phil, a TV psychologist, likes to use the quote, "You can't solve money problems with money." I believe the same thing applies to the JVM, "You can't always solve memory problems, with more memory." Treating just the symptom doesn't have a long-lasting effect.

So why, when Java was going to save us all from thinking about memory allocation, do we now have to think about memory allocation? If you look around, it's obviously a common issue, and there is no shortage of profiling and diagnostic tools. Carlos E. Perez covers a long list of tools in his manageability blog: www.manageability.org/blog/stuff/open-source-profilers-for-java/view and that's not even the full list Java developers have at hand.

Go to any Java conference and you are bound to hear at least one presentation about out-of-memory errors. There were some good sessions at this year's JavaOne. One was a packed BOF led by some of my old colleagues from the Sun serviceability team discussing the six ways to meet an out-of-memory error and another from Steffan and the JRockit JVM team at BEA.

Memory Leaks in Production

The challenge has always been to provide a memory leak tool that your IT staff will let you try in production. Realistically, many slow memory leaks only occur at deployment time, partly due to

the changed environment and partly due to the longer uptime of the application. In many cases an IDE-based tool may not be convenient in those scenarios; the other catch is that you also want the diagnostics to have a minimal affect on the application itself. So recompiling to add profiling hooks is often really only the last resort, if at all. To help ease this barrier to adoption, JDK 5.0 included byte code insertion, this technology has been used by profiling tools to do this 're-compilation' at runtime to make this more palatable. One thing that you may not be aware of is that the Sun JVM automatically generates some very lightweight counters, available by default in 1.4.2. Sun JVM tools such as jmap and visualgc can provide a view into the garbage collectors operation. The jconsole demo in JDK 5.0 also provides some insight into the garbage collector providing you started the JVM with the option `-Dcom.sun.management.jmxremote`.

There is also an improved hprof profile agent in JDK 5.0 that can be used to dump out profiling information in a format that can be used by the hat analysis tool from java.net. I find it more useful for snapshotting a JVM, than letting it run to completion. To generate a hprof file I can use QUIT signal, kill -3 on Unix after starting the JVM with the hprof agent `-agentlib:hprof=heap=all,format=b`. The hat tool looks for the java.hprof output file by default and starts a mini Web server at port 7000. This allows you to browse the object instances used by the JVM. It's by no means as powerful as the commercial tools, but can give you some clues as to what is going on.

Who Has My Memory?

Everyone generally has the same advice about memory leaks in Java. Providing you have already sized your maximum heap correctly (if heap is the detected out-of-memory condition), e.g.,

`-Xmx512m`, then some number of objects is holding onto references to other objects that are no longer required. As these objects are still reachable, and hence live, they will not be marked by the garbage collector as objects that can be deleted.

One of the classic examples given is that you wrote some JNI code and forgot to free the memory. Now, admittedly, that's fairly easy to do if you are using JNI, but what if your application has no JNI code at all?

First, rule out your code or dependent libraries or even the JVM. It's less common than it used to be, but JVMs do still have bugs. One area to check is if there are exceptions at deployment time. Sometimes the unsuccessful code paths are not as diligent at cleaning up non-local instances as the successful code path. Even a simple operation, such as creating a substring from a string, and some StringBuffer operations have caused leaks in a few of the 1.4 releases.

Dependent libraries could be an XML parser or JDBC driver. Both need to frequently build, manipulate, and copy chunks of data. Try using an alternative release number or version of both. Again it may not be your code at fault. If you still strongly suspect you've introduced the leak, what should you look at? Caches are prime suspects, or any place where you store references, whether it's a vector, array or Collection can easily always hang on to reachable objects. Make sure you null the references or use weak references. Other areas to check are custom classloaders if you use them. By their very nature they hold on to many references, and if you're using finalizers, remember they may never get called.

Now that desktop machines have 1Gb of memory installed instead of 1K, it's still necessary to watch your bytes. Tracking down a memory leak is never easy, however, there are certainly more tools and techniques at your disposal than ever before. ☺



A section editor of JDJ since June 2004, Calvin Austin is an engineer at SpikeSource.com. He previously led the J2SE 5.0 release at Sun Microsystems and also led Sun's Java on Linux port. calvin.austin@sys-con.com

If your users can freeze
their reporting requirements
for the next 18 months...

Don't read this ad

*Intelli***VIEW**

is designed for users' constantly
changing reporting requirements.
It cuts out repetitive tasks and gives you
time for more exciting challenges.

- Key Features** • Familiar spread-sheet like interface
• Ad-hoc reporting & easy analytics
• Powerful charting • Zero learning curve

Deploy IntelliVIEW and free yourself from report creation.
call toll-free **1-866-99IVIEW** or visit <http://www.intelliview.com/jdj>



Deploy *Intelli***VIEW**...
and you can be the
IT Super Hero

product of
Synaptris

3031 Tisch Way, Suite 1002, San Jose, CA 95128. USA
Email: sales@intelliview.com

How to Deal with Security When Building Application Architecture

by Michael Poulin

Are you ready to face the challenge?

Application architects have heard about the increased importance of security, but in many cases they really don't know how to approach this issue. In this article, I'll share my experience and define a few basic steps and checkpoints for building application architecture with security in mind.

This year, architects have started to face several domestic (SOX and HIPPA) and even international (Basel II) regulations that require a certain level of protection of the personal and financial data that's processed and owned by the companies. Though network and operating system security solutions have done a great job in their domains, there is still one weakly protected path to corporate data – it's a spectrum of commercial and homegrown applications. I won't discuss why security is important and what is required by the regulations because you can find a lot of related materials in *JDJ* and other on- and off-line resources. My goal is to identify and explain the most important steps to be taken toward security when building application architecture.



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

mpoulin@usa.com

Step 1: Requirements

Review your business and technical requirements to see if security is addressed there. If you find security requirements are absent or denied, e.g., "encryption of communication between servers is not required," collect and verify the requirements. In both cases, check requirements with the legal and compliance department – the statement "we usually do not do encryption for internal data exchange" may not be valid anymore in the light of new regulations.

During requirements gathering and

analysis, identify corporate security resources and policies. In particular:

- Consider integration with identity management and access control systems (for example, Liberty Alliance Identity Management or BEA's WebLogic Enterprise Security solutions).
- Discuss user activities that should be controlled and later audited in order to meet the policies.
- Consider the use of application state/status monitoring (e.g., via JMX) to easier recognize abnormal behavior potentially caused by security violations.
- Consider the operational procedures for obtaining access permissions for application users and periodic user access recertification.



You can find some examples of security policies in the sidebar: *Examples of Security Policies for Web Applications*.

Step 2: Sensitive Resources

Examine business models, data sources, the data, and the user community for your future application. This should help you to recognize the points that might be the most lucrative to a potential intruder. In the process, you have to answer, at least, some questions, such as:

- How does the business model implemented by the application affect the financial reporting and status of your organization?

- Does the data include personal and/or financial information and how attractive it may be to an intruder?
- Are the data sources reliable and secured, and can/may you verify it if necessary?
- Are the application customers internal or external with regard to the served organization?
- If they are internal, do you really know who they are, e.g., does your application use information from the user provisioning and identity management system? If the users are external, are they controlled by your partner/provider or consumer organization or is it an open public audience?
- Due to the nature of the application and data, should you expect targeted intruder attacks or is it likely the application will be under less sophisticated "curiosity" attacks?

Step 3: Creation of the Architecture

Armed with the knowledge of corporate and industry policies, on the one hand, and with the picture of potential spots of security violation in your application on the other, create the architecture and the high level-design. Since we know there is no such thing as 100% security for the functioning application, the architect has to prioritize the potential risks of deliberate and accidental security violations and intrusions.

It's very important to estimate the consequences of a security violation or an impact of security breaches on the application, other applications, and your entire organization. Keep in mind that some intrusions are highly possible but may have no or very little impact. The others, on the contrary, are much less probable but cause consequences that may be terrible.

Here are two examples:

1. Let's assume you've designed a Web application that would be deployed in a DMZ (demilitarized zone, i.e., the network zone between two firewalls). In our day, you have to expect continuous hacker attacks; if you constructed and deployed the application smartly (see Web Application Security Consortium, <http://www.webappsec.org/>), the majority of attacks would "die" in there and not penetrate into the middle and back-end layers.
2. If you design an "internal" application that is accessible to the operation team and supporting developers, there are a lot of risks as well. For instance, you have to watch whether a database user name/ password pair is stored "in clear" in the application configuration file. The probability of password misuse is low, but, as you know, in crisis situations we use all available developers, some of whom may be foreign contractors or even offshore programmers who have had little or no background check. So, a "clear" password is a "piece of cake" for

really bad guys; with these passwords he or she can get access to a company's strategic data. Then, just use your imagination...

When security risks are prioritized, it's easier to concentrate on the most dangerous ones and address them sequentially, in an iterative manner, spreading the cost of security controls over multiple phases of application implementation.

Step 4: Secured Design Solutions

After you have identified and prioritized security risks, try to come up with more secure solutions, to the best of your knowledge. For this, don't ignore the operational aspects of the application – a lot of security concerns may be covered via operational activities, not by the code only. For example, the application might not need to manage login credentials for the users if there is a strong user authentication operational procedure in place. In another example, Application A maintains a password for Application B in the configuration file in the encrypted form; if both

applications are deployed on the BEA WebLogic platform in a trusted domain, Application A would not need neither the password encryption nor the password itself for Application B.

In most of the cases, just following two architectural principles can provide a much higher level of security for the application. The principles are:

- *Layered application architecture.* The J2EE platform perfectly supports layered architecture. It contributes to the scalability, stability, and security of the entire application. If you accept an idea that each layer of the application responds to its special requirements (e.g., the Presentation layer responds to user experience requirements; the Business layer, business requirements; the Persistence layer, data management requirements), it will be much easier for you to design the application and preserve security integrity.
- *Separation of responsibilities.* For example, the delegation of data access from the Presentation layer to the Business layer can protect your database from easy exposure to the intruder of the Web application.



DynamicPDF™ Merger v3.0

Our flexible and highly efficient class library for manipulating and adding new content to existing PDFs is available natively for Java

- ♦ Intuitive object model
- ♦ PDF Manipulation (Merging & Splitting)
- ♦ Document Stamping
- ♦ Page placing, rotating, scaling and clipping
- ♦ Form-filling, merging and flattening
- ♦ Personalizing Content
- ♦ Use existing PDF pages as templates
- ♦ Seamless integration with the Generator API

DynamicPDF™ Generator v3.0

Our popular and highly efficient class library for real time PDF creation is available natively for Java

- ♦ Intuitive object model
- ♦ Unicode and CJK font support
- ♦ Font embedding and subsetting
- ♦ PDF encryption ♦ 18 bar code symbologies
- ♦ Custom page element API ♦ HTML text formatting
- ♦ Flexible document templating

Try our free Community Edition!



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



I can easily anticipate that many architects will say, “Look, you recommend that we apply a lot of extra work while we have to implement the business tasks in the application,” or “If we start to implement all these security controls and protections, we’ll get certain performance degradation in the application,” and so on. Yes, you’re absolutely right! If you think about security as an additional feature instead of an organic business requirement, which makes the application trusted, the application will be sacrificed. That’s why it’s better to embed security into the structure of the application at the earliest possible step of the architecture design process.

Security requires resources and processing time. To support a certain level of performances and address security, the architecture has to represent “compensating” solutions. That is, you have to find solutions that

save resources and execution time to spend them on application security needs at runtime. For example, if security controls take much time, use caching more intensively; if gathering audit information about user activities slows down application response, invoke asynchronous acquisition of audit data. For example, when implementing an MVC pattern as Struts and deal with audit, the `ActionServlet` or the code pointed by `ActionForward` class can send audit data via JMS to the audit storage instead of holding up the response thread while writing audit data into the database by itself.

Step 5: Architecture Security Review

When the architecture and high-level design are complete, invite the legal and compliance department to review them. Your goal is to get a sign-off on your design and security risk mitigation solutions.

Step 6: Post-Design Security Testing

The architect can contribute a lot to security even in post-design phases:

- If you are the architect who accompanies the project through the implementation, I would recommend invoking the legal and compliance department once again to review the code from a security perspective.
- You shouldn’t concentrate on a static analysis of the code against security policies and delegate such tests to QA. However, since you are probably the one who knows the weaknesses of the whole application (short-cuts and design compromises taken during implementation), you are the right person for the task of designing penetration tests.
- Finally, since you know what to expect from the integral execution of different elements of the application, you can identify “unusual” activities in the application log (open source commons-logging with log4j or standard java.util.logging logging package may be recommended for logging). Recognition of these activities may help to automate a security log reviews in the application maintenance phase.

Now, you are ready to face the security challenge! By guarding your application, you guard your organization, your customers, and, finally, yourself. ☺

Resources

- Kolawa, A., and Fain, Y. “Java Application Security in the Corporate World”: <http://java.sys-con.com/read/99704.htm>
- Pasley, K. “Sarbanes-Oxley (SOX)—Impact on Security In Software”: <http://www.developer.com/java/ent/article.php/3320861>
- Newman, H. “HIPAA and SOX: What You Need To Know”: <http://www.enterprisestorageforum.com/continuity/features/article.php/3506751>
- Owen, M. “Preparing for the Pain of Basel II”: <http://www.developer.com/java/ent/article.php/3403901>

Examples of Security Policies for Web Applications

- UserID and password must be sent via the POST method only.
- Password should be encrypted when stored in the database.
- In production, remove all “backdoor” login code.
- Assigned accounts should be locked if a number of wrong attempts to login exceeds N times.
- Validate all data coming from external resources, especially from the user’s browser.
- Input data should be validated using the strongest validation level (Exact Match, Known Good Validation, Exclude Known Bad).
- In case of fatal errors, the application has to fail securely (error handling).
- Confidential information should not be written in regular log files.
- Log file access should be protected.
- No messages sent back to the browser may contain any debug information.
- No messages sent back to the browser may contain any server-side information.
- Logging error messages should not differentiate between wrong UserID and wrong password.
- No database-related information should be returned back to the user in error messages.
- If the cookies are not protected by an encryption (hashing, obfuscation), the data in them should be validated before use.
- Session time-to-live must be agile to corporate security policies, not to the business model (if needed, the session has to be gracefully refreshed).
- Avoid hidden fields in the Web page.
- Where possible, for all requests choose the API that can provide a UserID and password, and use authorization control with the least privileges rule.



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

>>> **Think JReport®**

Deliver Intuitive, Easy-to-Understand Reports

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

Seamlessly Integrate with Your Java Infrastructure

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

Robust Security for Enterprise Deployment

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

Support Enterprise Service Level Requirements

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

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

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

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

Download a FREE, fully functional copy of JReport at www.jinfony.com/jp9 or call (301) 838-5560.

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

xPhoneApp™

Do

Phoneomena products allow the enterprise to easily create its own mobile applications or mobilize existing applications without having to learn new technologies or make upfront investments. With Phoneomena products you will not be locked into any mobile platform and will not waste time and resources learning J2ME, Brew, Windows Mobile, Palm OS or Symbian. You will be freed to focus on your mobilization strategy to achieve your business goals. You will literally be able to mobilize overnight, using only the standard web knowledge your IT team has today.



Mobilize rapidly with minimum investment and guaranteed

ROI xPhoneApp™ is an eXtensible Phone Application middleware innovated by Phoneomena, Inc. It allows developers to create mobile business applications without having to learn complicated or new technologies such as Java 2 Micro Edition, BREW, or Microsoft .NET Compact Framework. The technology enables developers with web programming experience to use their skill set and learn nothing more to develop sophisticated mobile applications. Furthermore, the technology provides significant portability across different platforms.

Main Features

- » Rapid development and prototyping: develop, test, and load phone applications within hours or days instead of weeks or months
- » Short learning curve: nothing more to learn than standard XML
- » Portability: develop mobile platform-independent applications
- » Auto-provisioning of applications (no over-the-air application downloads required)
- » Easy maintenance
- » Application size not limited by phone capacity
- » Scalable to large enterprise
- » Built in database for local storage
- » Secure on SSL-enabled handsets

Try xPhoneApp Today

Want to test xPhoneApp out for yourself? We have developed an interactive demo system on our website. The xPhoneApp online demo will allow the creation of up to five applications and allow you to see and actually use them on your cell phone for immediate testing and evaluation.

www.phoneomena.biz/xPhoneAppDemo/



it yourself.

consumer

Use the xPhoneApp developer toolkit to create your own custom mobile applications. The consumer mobile market is growing daily; don't miss the opportunity to provide value-added services to your current and/or future business!



healthcare

Build medical applications such as patient lists, approvals, lab reports, charge capture, and image scans quickly and efficiently using pre-built templates available for xPhoneApp.



education

Universities, colleges, and public school systems can quickly and easily build applications to access class schedules, campus events, and even look up grades using a secure internet connection. xPhoneApp enables all of these functions and more.



field force automation

Enable your field force team with xPhoneApp and ready applications that you can take, use, and/or modify to your own needs. Barcode scanning, RFID readers, and GPS tracking are also available for certain phones.



PHONEOMENA

Phoneomena, Inc.
104 N. Main St. Suite 300
Gainesville, FL 32601, USA

T: +1 (352)-373-3966

F: +1 (352)-373-3651

www.phoneomena.com

Toll-free: +1(888) 891-7316

Email: info@phoneomena.com

A Blueprint for Developing Language Tools

by Paul Mukherjee

A proven approach to making them modular, extensible, and maintainable

Language tools such as compilers, interpreters, and code generators are a critical part of the software development landscape. Any software project will include several procured tools and very likely several in-house tools. Experience shows that the only guarantee with such tools is change: the underlying language may change due to improvements or extensions and the functionality provided by the tool expands, driven by user-requested features and the need to stay in front of the competition. The specific changes that will be made are rarely known at the outset, but change is coming.

This implies that when designing such a tool, extensibility is paramount. So it's important that the design is modular. A clean division of responsibilities is needed to support maintainability, which in turn is needed to support the rapid pace of change typically associated with language tools.

This article describes a proven approach to developing language-based tools in a way that is modular, extensible, and maintainable. The approach is based on two principles: establishing the core modules at the outset and using the visitor pattern to interact with language sentences.

To illustrate the approach a simple calculator example is given. This calculator supports the addition, subtraction, multiplication, and division of integers. The language supported by this calculator is informally described below:

```
expression = constant |
            expression op expression
constant = 0 | 1 | 2 | ...
op = + | - | * | /
```

This language is very simple, but it's sufficient to illustrate the main

concepts related to the development of language tools. The example is developed in Java based on the JavaCC parser generator. However the concepts presented are language-independent and apply equally to C++ and C#. All of the code shown in this article is available for download.

A key objective when developing a language tool is to ensure that the tool isn't dependent on the actual representation of the language. This allows simple support for multiple language formats, imports from other tools, and so on. To meet this objective, a distinction is made between concrete and abstract syntax. This is described below. After this the notion of context is introduced followed by a description of the actual mechanism for parsing. Then the use of the generated parse tree is explained.

Concrete Syntax

Concrete syntax represents the way in which a specific file format represents the input for the tool. This is often described using BNF or a similar structured representation. If a parser

generator such as Lex/Yacc, Antlr, or JavaCC is used, the concrete syntax will be described in a generator-specific manner. A simple concrete syntax for the calculator using JavaCC shown is Listing 1.

Note that normally semantic actions would be included in such a JavaCC description. These are presented later in the article.

In general there can be several concrete syntaxes for a language (plain text, XML, RTF, etc.). The tool design should be sufficiently flexible to support multiple concrete syntaxes (as well as the ability to add further concrete syntaxes) without having a major impact on the rest of the tool.

Abstract Syntax

Abstract syntax is a representation of the language that's independent of the concrete syntax. The abstract syntax representation contains only information that's necessary for the tool to perform its task; any other information is discarded. In principle this necessary information should be included in all concrete syntaxes. So



Paul Mukherjee works as a technical architect for Systematic Software Engineering Limited in Britain, and is a Sun Certified Enterprise Architect. He has worked on a number of small and large projects ranging from small J2EE solutions to large mission-critical enterprise application integration programs.

pmu@systematic.dk

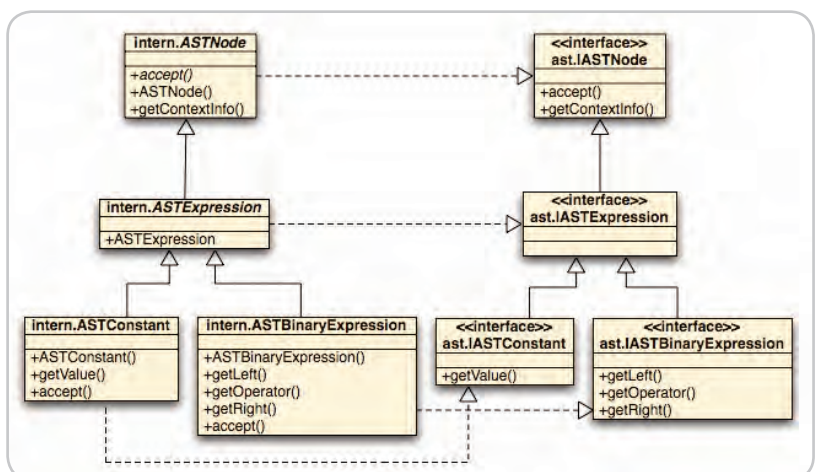


Figure 1 Abstract syntax classes

there should only be one abstract syntax regardless of the number of concrete syntaxes. In an OO setting, an abstract syntax is typically a tree-structure that reflects the way in which language sentences can be constructed. The abstract syntax fulfils a number of functions. It:

- Represents the program
- Stores any necessary information related to the concrete representation (e.g., for pretty printing, relating error messages to specific file locations, etc.). This is explored further in the next section.
- Exposes the above information to tools without revealing implementation details.

The tree-like structure of the abstract syntax is represented using inheritance; we use context information objects to store the necessary information from the concrete representation; exposing information to tools without revealing implementation is achieved with interfaces. And since the abstract syntax has a tree-like structure, tools will access it using a visitor pattern. This basic structure is shown in Figure 1.

Notice that the package arrangement in Figure 1 follows the convention used in Eclipse that dictates that classes in a package named `intern` are not to be exposed to other tools.

Context Information

The idea of creating abstract syntax is that the tool will use these objects to achieve its goals. This means that the tool is not dependent on the specific concrete format being used. However, this separation can be problematic since in some situations information from the concrete representation is actually needed. For instance, a type checker might generate an error message that has to be displayed to the user. For this message to be of value, the specific location of the error has to be provided.

The solution to this problem is to associate each abstract syntax node with an object defining the context of that node. This might,

for example, be the start and end line and column for the node; the information required here may vary according to the nature of the language, the tool, and the concrete format in question. As with the abstract syntax, client tools should be shielded from the implementation details of context information, so an interface is used and classes implementing this interface are internal. It's possible that there may be multiple context information classes according to the concrete representation. This is suggested in Figure 2 where a plain text context information class is used.

Parsing

To create a parser, the concrete syntax presented earlier needs to be married with the abstract syntax classes. This is shown in Listing 2. Note that there will typically be one parser for each concrete syntax supported.

This example is based on JavaCC, but the principle applies to other parser generators: the semantic actions in the matching rules in the parser definition are used to create and instantiate abstract syntax objects, resulting in the creation of an abstract syntax tree corresponding to the input text. This abstract syntax tree will be the input to other components in the tool that require the input text.

If the input format is XML, then the approach would be somewhat different; typically a JAXB compiler would be used to convert an XML document into a parse tree. However, the details, though interesting, are beyond the scope of this article.

Use of the Parse Tree

Components interact with the parse tree using the visitor pattern. This allows for a clean separation between the abstract syntax and the components using the abstract syntax. In this case only concrete abstract syntax classes should be visited. For example, consider the calculating engine used to compute the result of an expression entered in the calculator (see Figure 3).

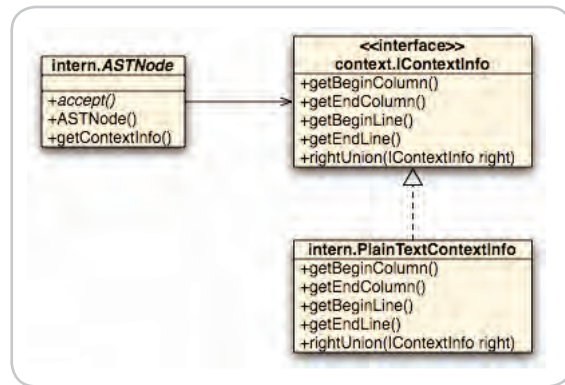


Figure 2 Context information classes

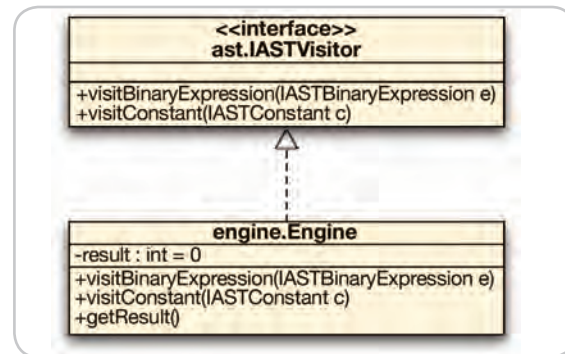


Figure 3 Engine Class



COMMON CONTROLS www.common-controls.com

The Java™ Presentation framework for J2EE™ Web applications

Based on:

- Java™
- Servlets™
- Java Serverpages™
- and Struts



For Free! Get your free trial version – www.common-controls.com

See the common controls in action – go for the [Online Demo!](#)

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

Lists

Trees

Tabfolders

Menu

Forms

BreadCrumbs

Calendar

Colorpicker

www.common-controls.com

The implementation for the engine is relatively straightforward recursing down the tree. This is shown in Listing 3.

Similarly a visitor can be created to generate an XML representation of the text input by the user as shown in Listing 4. This visitor would, for example, generate the following XML given the input "1+2-3" (white space added for legibility):

```
<binaryexpr>
  <binaryexpr>
    <constant value="1"/>
    <op value="PLUS"/>
    <constant value="2"/>
  </binaryexpr>
```

```
<op value="MINUS"/>
<constant value="3"/>
</binaryexpr>
```

Note that the usual provisos concerning the use of the visitor pattern apply. In particular if the language is such that the syntax will change frequently, there's a significant overhead associated with updating all of the visitors. However, most stable languages change their syntax infrequently.

Conclusion

This article has presented an approach to developing language tools that has been used extensively for

several products and projects that I have been involved with. Experience shows that separating the triumvirate of core functionality, input format, and parsing ensures the flexibility required for evolution over time. By explicitly treating concrete syntax, abstract syntax, and context separately, it's possible to develop language tools that are modular, extensible, and maintainable. ☺

References

- Source code <http://jdk.sys-con.com>
- <https://javacc.dev.java.net/>
- <http://www.eclipse.org/>
- <http://java.sun.com/xml/jaxb/>

Listing 1: Calculator concrete syntax

```
TOKEN:
{
  <DECIMAL_LITERAL: ["1"- "9"] (["0"- "9"])* >
}

void Input() :
{}
{
  Expression() ("\n"|"r")
}

void Expression() :
{}
{
  c1 = Constant()
  ( op = BinaryOperator() c2 = Constant()
  )*
}

void BinaryOperator() :
{}
{
  (
    "+" | "-" | "*" | "/"
  )
}

void Constant() :
{}
{
  <DECIMAL_LITERAL>
}
```

Listing 2: Parser Definition

```
TOKEN:
{
  <DECIMAL_LITERAL: ["1"- "9"] (["0"- "9"])* >
}

ASTExpression Input() :
{
  ASTExpression astExpression;
}
{
  astExpression = Expression() ("\n"|"r")

  {
    return astExpression;
  }
}

ASTExpression Expression() :
{
  ASTExpression c1 = null;
```

```
ASTConstant c2 = null;
ASTBinaryOperator op = null;
}
{
  c1 = Constant()
  ( op = BinaryOperator() c2 = Constant()
  {
    c1 = new ASTBinaryExpression(c1, op, c2);
  }
  )*
  {
    return c1;
  }
}

ASTBinaryOperator BinaryOperator() :
{
  ASTBinaryOperator op;
}
{
  (
    "+" { op = ASTBinaryOperator.PLUS; } |
    "-" { op = ASTBinaryOperator.MINUS; } |
    "*" { op = ASTBinaryOperator.MULTIPLY; } |
    "/" { op = ASTBinaryOperator.DIVIDE; }
  )
  {
    return op;
  }
}

ASTConstant Constant() :
{
  ASTConstant c;
  Token t;
}
{
  t = <DECIMAL_LITERAL>
  { c = new ASTConstant(Integer.parseInt(t.toString()),
    new PlainTextContextInfo(
      t.beginLine,
      t.endLine,
      t.beginColumn,
      t.endColumn));
  }
  {
    return c;
  }
}
```

Listing 3: Engine Implementation

```
public class Engine implements IASTVisitor {
    private int result = 0;
    public void visitBinaryExpression(
        ASTBinaryExpression binaryExpr) {
        binaryExpr.getLeft().accept(this);
        int leftValue = result;
        binaryExpr.getRight().accept(this);
        int rightValue = result;
        switch (binaryExpr.getOperator()){
        case PLUS:
            result = leftValue + rightValue;
            break;
        case MINUS:
            result = leftValue - rightValue;
            break;
        case MULTIPLY:
            result = leftValue * rightValue;
            break;
        case DIVIDE:
            result = leftValue / rightValue;
            break;
        }
    }
    public void visitConstant(ASTConstant c) {
        result = c.getValue();
    }
    public int getResult(){
```

```
        return result;
    }
}
```

Listing 4: XML Generator

```
public class XMLGenerator implements IASTVisitor {
    private StringBuffer buffer = new StringBuffer();
    public void visitBinaryExpression(
        IASTBinaryExpression binaryExpr) {
        buffer.append("<binaryexpr>");
        binaryExpr.getLeft().accept(this);
        buffer.append("<op value=\"");
        buffer.append(binaryExpr.getOperator());
        buffer.append("\"/>");
        binaryExpr.getRight().accept(this);
        buffer.append("</binaryexpr>");
    }
    public void visitConstant(IASTConstant c) {
        buffer.append("<constant value=\"");
        buffer.append(c.getValue());
        buffer.append("\"/>");
    }
    public String getXML(){
        return buffer.toString();
    }
}
```

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

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

Only **\$69⁹⁹** ONE YEAR
12 ISSUES

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

www.JDJ.SYS-CON.com
or **1-888-303-5282**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



Joe Winchester
Desktop Java Editor



One Size Fits No One

At a presentation a number of years ago given by Josh Bloch he made a comment that Java as a language hit the “sweet spot” of programming. His metaphor was based around the fact that the language was straightforward to learn and that rather than containing many esoteric coding constructs, writing and understanding a Java program was a relatively easy task.

I think Java is at a very critical point at the moment where it is slipping away from its sweet spot and this worries me. Two things are to blame: annotations and aspects.

An annotation allows a programmer to flag a part of a program with @ statements that at first glance are a glorified comment. A developer can define his or her own annotation that has typed properties and validation rules about where it can be used in code, both of which the compiler enforces. What you do with annotations is up to you, but a good example could be to formally flag which methods were fixed in a particular release, by whom, and why. For example, with an annotation called Mod you could write code like:

```
@Mod(bugNumber=5477, fixedBy="Fred");
public void foo(){
    ...
}
```

This is better than putting the details in a comment *// Fixed by Fred for 5477* because programs can use the `java.lang.reflect` API to query classes and methods for the presence of annotations, so a report of fixes done by Fred could be written.

The problem occurs when an annotation is more than a glorified comment and contains information that is an instruction to the program itself. EJB 3.0 has fallen desperately foul of this and I saw some horrid sample code recently:

```
@Stateless
@Remote(Example.class)
public class ExampleBean implements Example
```

```
{
    @PermitAll
    @TransactionAttribute(TransactionAttribut
eType.REQUIRES_NEW)
    public String getName(int id)
    {
        // Method body here
    }
    @RolesAllowed({"administrator", "power_
user"});
    public void deleteName(int id)
    {
        // Method body here
    }
}
```



What has occurred is, basically, a ton of semantic program details about how the EJB should be deployed that used to be provided in the deployment descriptor has been slammed into the class as annotations. It is way, way wide of the sweet spot and looks more like a cross between a set of assembler op-codes mixed with some kind of fourth generation language syntax. Whatever it is, it isn't Java.

Kent Beck once said that having lots of classes and lots of methods is basically what good OO is about. Design Patterns, the bible of good OO practice, espouses patterns such as the strategy and mediator that encourage and teach the strength and power of separation. By contrast, the EJB specification actually boasts the fact that having everything defined in a single file is a good thing. Separate XML wasn't great but what would have been wrong with just moving from XML to something akin to BeanInfo where the logic and rules were held in Java code elsewhere?

This segues nicely into aspects. At the first presentation I saw on the subject I was told that the *raison d'être* for their existence is so that only a single source file has to be touched to implement a piece of functionality. While I understand this as a goal, it just isn't really practical to make this your overriding goal and then jump through hoops to ensure that this is the focus of all your development. In my experience, with all but the most trivial change, to fix a bug or implement a feature you need to change several classes, perhaps an interface or two, and hopefully do some refactoring along the way to improve the overall system entropy. There is nothing wrong with good old-fashioned programming like this and, when I encounter aspects, I see a group of people driven with a zeal to do differently. Instead they code files that contain sets of instructions to a preprocessor that will go and modify the existing multiple files that you should have fixed by hand in the first place. As with annotations, there are good uses of aspects, namely introducing logging behavior, performing code metrics, or coding rule enforcement. Too far beyond this and they just become clever magic that is both confusing and silly. Aspect fever seems to be riding the hit parade at the moment as the silver-bullet answer to everyone's problem.

What both annotations and aspects bring to Java is some kind of powerful dark magic where source is now littered with semantic fluff disguising itself as something more trendy but wielding terrible power. What you write is no longer what gets run – someone else's preprocessor mangles it, clever code obscures this fact from you while you debug it, and rather than the JVM just executing the bytecodes from the source you wrote, there is now some kind of execution inference engine analyzing formalized comments to determine the code path instead.

I fear the worst for the language. Pandora's box has been opened, Java coding no longer has any rules to govern it, and muggle programmers are no longer safe. ☹

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

joewinchester@sys-con.com

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



Your Solution: Spreadsheet Automation with ExtenXLS

Astronaut Training Greatly Reduced!

With the familiar look and feel of Excel, ExtenXLS does not demand any learning curve from your end-users. ExtenXLS unlocks the business logic stored in static spreadsheets throughout your organization, saving time & money.

Put a Rocket Under the Hood!

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

Escape the Gravitational Pull of Obsolete Reporting!

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

Beam ExtenXLS Down Now!

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

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

- Project Leader, AT&T

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

- Project Leader, John J. McMullen Company

EXTENXLS4TM
JAVA | XLS REPORTING TOOLKIT

 **extentech**TM
Call Us: 415-759-5292

Are Portals the 'Magic Bullet' of Web Application Development?

The many advantages to utilizing portal software

by Roy Russo and Julien Viet

When speaking of Web application development today, it's difficult to ignore the overwhelming influence of the Portlet Specification (JSR-168). Even before the specification was formally finalized by the expert group, the Java world saw older CMS application implementing it and new portal software arrivals in the market. The proverbial "gold rush" to develop new applications as portlets, refactor existing applications to comply with the specification, and deploy new Web sites on portal software is not without good reason. The Java community was lacking a unifying specification in the Web tier, where all previous work could be brought together and leveraged, removing the tedious tasks developers once had to endure when creating most common Web applications.

Portals, as defined by the specification, are a new arrival in the market and much of the fanfare is due to just that. They have been touted as the "magic bullet" of Web application development and a new standard in developing scalable, flexible, and pluggable software components. Having lived through the dot-bomb era, we are not alone in knowing that the "newness factor" and the endless search for the "killer-app" can often cloud the judgment of decision makers regardless of functionality.

Although portals, as they exist today, promise to provide improved functionality by building on and consolidating previous work in this area, features and functionality should be the main determiners of whether to deploy a portal, a CMS, or develop a Web application using JSPs and servlets. However, before considering deploying a portal, you must have a solid understanding of what a portal actually is, what technologies are commonly found in them, and even appreciate how portlets interact with the portal.

Portal Overview

Reading section 2.1 of the Portlet Specification, a portal is defined as:

... a web based application that – commonly – provides personalization, single sign on, content aggregation from different sources and hosts the presentation layer of

Information Systems. Aggregation is the action of integrating content from different sources within a web page. A portal may have sophisticated personalization features to provide customized content to users. Portal pages may have different set of portlets creating content for different users.

As the specification states, portals commonly allow for personalization, SSO, and content aggregation. Figure 1 shows elements commonly found in open source and proprietary portal software.

Before we continue, it's important that you understand the items in Figure 1. It's likely that these functions alone will dictate whether you decide on implementing portal software. Frankly, these are among the most important and labor-intensive features to develop for most Web applications, so allowing a portal to perform the heavy-lifting exercises may be in your best interest.

- **Content aggregation:** Portals have the ability to present the user with information from different sources, displayed within portlets on a portal page (see Figure 2).
- **Caching, clustering:** Like most enterprise Web applications, portals tend to leverage existing caching and clustering technologies for increased performance and reliability.
- **Security and SSO:** The ability of a portal to integrate with an existing security schema used for authentication and/or authorization.
- **JSR 168 compliance:** Java portals, open source or not, all share this common bond as a unifying theme, allowing for portability across all vendor platforms.
- **Content management:** The ability of a portal to serve and allow administrators to manage content.
- **Personalization:** The capability of a portal user or administrator to personalize the portal and/or the individual portlets.

Of course, the diverse group of portal vendors presently in the market will offer differing sets of components to leverage within their portal, even addressing points where the Portlet Specification in its current state falls short, such as Inter-Portlet communication, portlet filters, extending the CSS support, and integration of existing frameworks (e.g., Struts or JSF).



Julien Viet is lead developer at JBoss Inc.

julien@jboss.com



Roy Russo is a developer at JBoss Inc.

roy.russo@jboss.com

In addition to the above cases of technology commonly found in portal software, the specification also defines the concept of a portal page. A screenshot of JBoss Portal using a custom layout and theme is used as an example here (see Figure 3).

The process of generating a portal page works like this (see Figure 4):

1. Portlet generates markup and dispatches it to the portlet container.
2. The portlet container sends the portlet content to the portal.
3. The portal adds decorations to these fragments, e.g., titles and window controls
4. The portal places a new decorated fragment on a page.

JSR 168 seeks to define the contract between the portal and the components running inside it. If you elect to deploy a portal, it is this specification you should adhere to, ideally. Some vendors may have their own proprietary APIs that, we believe, should be avoided for standard portlet development. These proprietary APIs result in vendor lock-in scenarios affecting the portability of the portlets created and increased maintenance and training costs in the long run. We would invite anyone comfortable developing servlets to read JSR 168, as it's very easy to read.

Portlet Overview

This section provides a brief overview of some of the items covered within the specification document. We made an effort to not describe in deep detail all the technical facts in the portlet API. Frankly, that was not the goal of this article, as countless other articles and books have covered these points in the past. This section will cover items at a high level that we see as important differentiators for those evaluating the use of portal software.

Portlet, Defined

A portlet is a Java application, packaged in a WAR file, and managed by the portlet container. They are pluggable components responsible for presenting fragments of data from information systems. Portlets can be as small as a content portlet that simply displays a fragment of HTML, or as large and complex as a CRM or e-commerce application.

The Portlet Life Cycle

The life of a portlet can be summarized by listing the specific methods that are called during a transaction:

1. **init(PortletConfig):** Called by the portlet container, this method initializes the portlet using a configuration object. A sample configuration is shown in Listing 1. Configuration information for an individual portlet can be accessed at any time after initialization.
2. **processAction(ActionRequest, ActionResponse):** This method is called if the client initiated a call request from an action URL. If the client's request is a render URL, this method is not called.
3. **render(RenderRequest, RenderResponse):** This method generates the content upon being called by the portlet container.
4. **destroy():** Called by the portlet container when it determines the portlet should be removed from service.

Portlet Modes

Portlet mode functionality allows a portlet to display different information depending on which mode the user is interested in (the same can be said for window states, covered later). The portlet mode names given are rather obvious to anyone reading them as to what their function should be. Using a generic example of a WeatherPortlet, we can speculate on the definitions for each of the default modes covered in the specification:

- **VIEW – doView():** Generates a content fragment displaying weather from NOAA.
- **EDIT – doEdit():** Allows a user to modify his or her preferences, say, forcing the portlet to only show the weather in Miami, FL, USA.
- **HELP – doHelp():** Displays a help HTML fragment with instructions on how to use the portlet.

Some vendors define custom portlet modes that your portlets can implement. Support for custom portlet modes is briefly defined in the specification, and is as simple as adding a *custom-portlet-mode* node to the portlet descriptor and any other vendor-specific modifications to the descriptor or the portlet class.

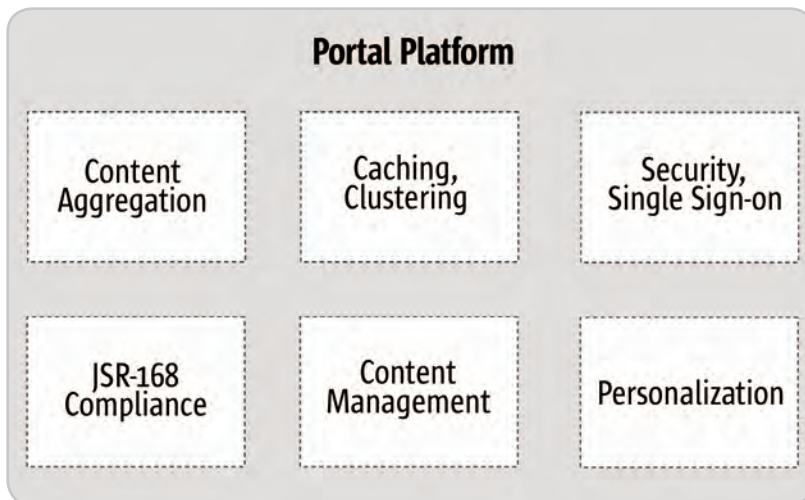


Figure 1 Elements in a typical portal platform

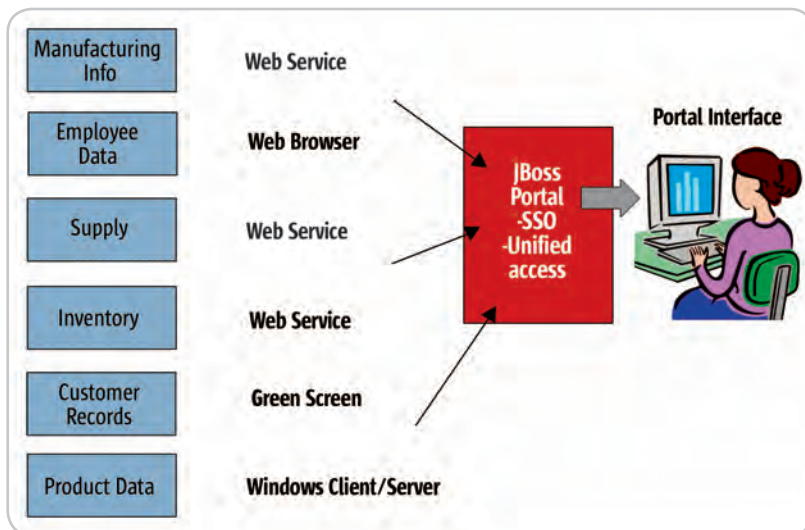


Figure 2 Aggregate contents in a portal

Window States

Window states control how much space any portlet takes up on any given page. They function when the portlet container passes the user's desired window state to the portlet. The portlet can then modify the fragment of data to display or even change the window state while processing the action request. The use of custom window states, much like custom portlet modes, is briefly mentioned in the specification and implemented by some vendors. Much like custom portlet modes, the portlet descriptor must be modified to use the *custom-window-state* element. Window states are briefly explained below. Note that not all portal vendors translate the specification in the same way, leaving much of the window mode behavior up to the portal implementation:

- **NORMAL:** This portlet has a limited space and is probably sharing a page with other portlets.
- **MINIMIZED:** In this state, a portlet renders no output, or very little content.
- **MAXIMIZED:** Normally, this state implies that the port-

let will take up all the viewable area on the page. Often, it's the only portlet displayed on that page.

The Future of Portal Software

A subsequent version of the portlet specification should begin in the near future. Planned for future releases are topics such as:

- Interportlet-communication (IPC), allowing communication between portlets on events
- Portlet filters, which are similar to servlet filters
- Portlet markup extensions, where a portlet will be able to modify the markup outside of the markup fragment

Aside from additions to the portlet specification, there has been a trend by the portal industry to integrate outside technologies, augmenting their products' feature sets. You could expect this trend to continue and expand, given the growing popularity and success of portal software. Some of the technologies being integrated that are worth noting are listed here:

- **WSRP (Web Services for Remote Portlets) specification:** WSRP (www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf) allows for the creation of a distributed portal infrastructure. This facilitates WSRP-compliant portals being able to display portlets from another WSRP-compliant portal.
- **JCR (Java Content Repository, JSR 170) specification:** Adoption of this specification seems to be universal in the portal space. It seeks to provide a common API to content repositories; allows for architecture-agnosticism; and features support for versioning, locking, transactions, and searching, among other things.
- **Framework support:** Facilitates the use of existing Web application frameworks to be leveraged in portlet development, such as JSF/My Faces, Struts, and Spring MVC. Figure 5 shows Sun's JSF CarDemo application running as a MyFaces portlet inside of JBoss Portal.

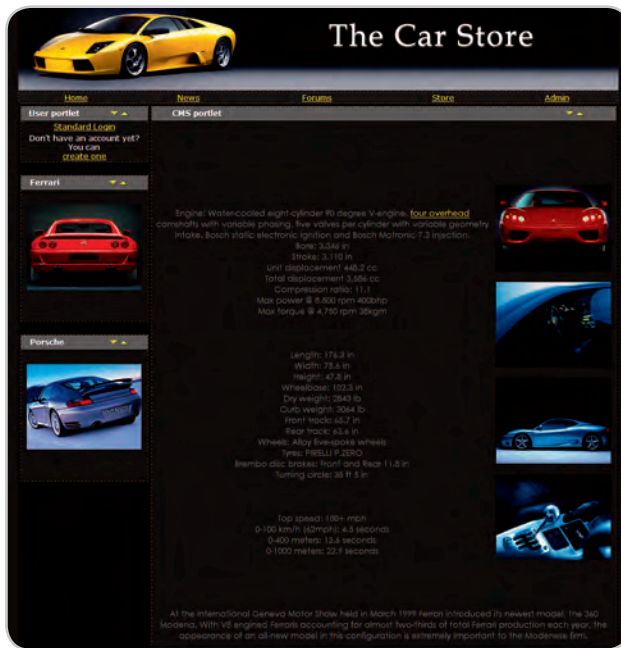


Figure 3 Customizable look-and-feel in a JBoss Portal

The Portal Option

Now we must answer the question: "Are portals the magic bullet of Web application development?" To answer this question, you must look at the functional specification for the particular application being developed. Simply

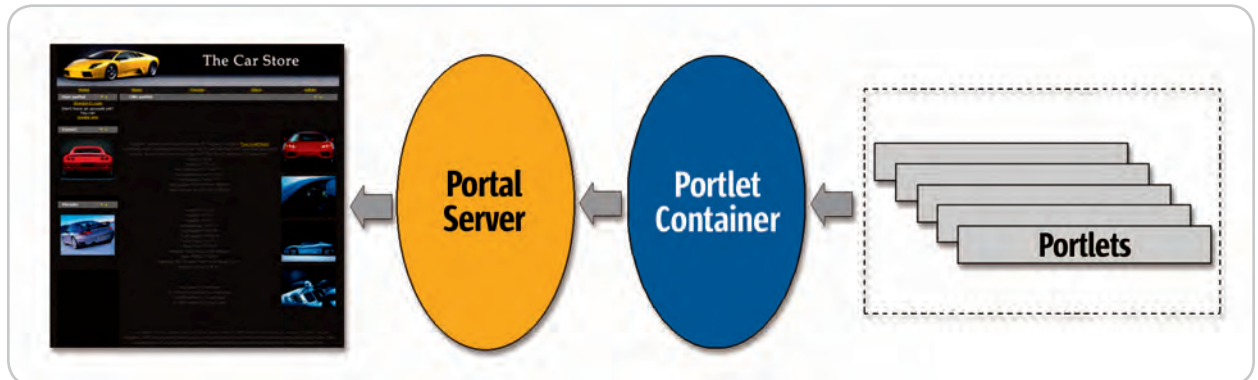


Figure 4 Generate a portal Web page

because you are developing a Web application doesn't mean you need to build or implement an existing portal. However, if your application specifications call for implementing some of the following, it would be wise to heavily consider using a portal:

- Single sign-on
- Personalization
- Content management
- User and role management
- Security and permissions management

Consider also some of the questions that we encounter from those evaluating portal software in their development projects:

My proposed Web application requires some of the features commonly found in portals; should I just develop my own proprietary architecture in-house then?

What we have found in the past when development teams create systems such as these, they tend to cobble in random odds and ends to fill the requirements listed with little consideration given to how all these disjointed parts will work together – if the component will be maintained in the future, will there be someone responsible for keeping up with the particular component, or if support outside of a mailing list is even offered for that particular component. During the development of JBoss Portal, we stayed away from creating what we refer to as a “Frankenstein Project” and leveraged technologies that were proven in production environments (see Figure 6) and supported in-house by the lead developers. This philosophy aided us in identifying problems when they popped up, and having the knowledge on-staff to deal with them. How comfortable are you allowing your developers to pick and choose random components from the Internet and glue them together? Leveraging portal software to handle the heavy lifting and intricacies of a Web application's development allows developers to concentrate on portlet development, which is probably the integral part of the business anyway.

Do I need a CMS or a Portal?

The line between each is a blurry one. Portals tend to offer CMS capabilities, in general, on par with all the features traditional content management systems do. The main differentiator between a “pure CMS” and a portal is adherence to the portlet specification, or allowing JSR 168-compliant portlets to be deployed within the container. So you see, a portal can have CMS capabilities, but not all CMSs are portals. Aside from this one dissimilarity, most CMS offer the same feature set as portals.

We will also say that we are cautious of traditional CMS that later cobble in support for the portlet specification. Having our own JBoss Nukes CMS, we can attest to the difficulties inherent in performing such a refactoring of the core architecture. Sure, it cost us in terms of time-to-market, but starting from scratch with a new portal architecture bought us endless flexibility and scalability. Cutting corners is not an option and we would be wary of any CMS-vendor-turned-portal-vendor for this reason.



Figure 5 A JSF application running as a portlet in the JBoss Portal server

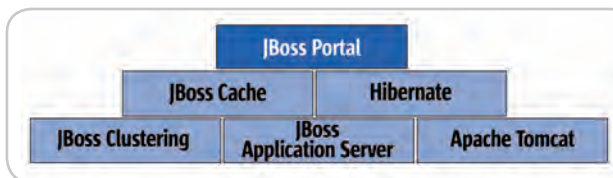
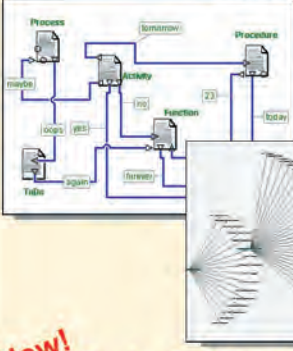
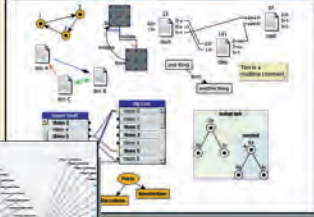


Figure 6 The technology stack in JBoss Portal

Build Incredible Interactive Diagrams with JGO™





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

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

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

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



“Portals, as defined by the specification, are a new arrival in the market and much of the fanfare is due to just that”

Wouldn't retraining my staff to develop portlets nullify all economic gains compared to creating our own Web application with existing competencies?

If your developers are savvy enough to tackle SSO, personalization, internationalization, caching, clustering, etc.... and make it all accessible via servlets and JSPs, they are more than able to absorb all the technical knowledge found in the portlet specification in a short amount of time. After all, the portlet specification is extremely comparable to the servlet specification. The main differences between the specifications are:

- Portlets are not tied to specific URLs, making use of the API to create URLs that are parsed through the portal server and eventually execute within a specific portlet.
- Many instances of portlets can co-exist on any assigned page.
- Portlets leverage Portlet Modes and Window States, allowing for different content to be rendered depending on a user's desire.
- Portlets generate content fragments.
- Portlets are written to the javax.portlet package.

There are close to a dozen major portal players in the market. Which one should I choose?

Being JBoss Portal developers, it would be seen as biased for us to evaluate the different players and furnish a result on which you should choose to go with, but there are a few items to investigate when choosing which is best for your business.

Is the portal JSR168 compliant? (Does it pass Sun's TCK?)

For reasons of portability and future maintenance, it would not be wise to develop to a proprietary API. The portlet specification continues to advance and knowledge-sharing by the community is extremely active. For these reasons, vendor lock-in should be avoided.

Are you considering open source or proprietary?

Open source and proprietary portals share essentially the same features currently, with one exception: some of the proprietary vendors supply smooth integration with other applications and systems they offer. For instance, a proprietary portal can be bundled with an existing CRM or e-commerce tools offered by the same company. If the need and budget exists to have a portal tightly integrated to other proprietary information systems, you should consider the proprietary offerings. On the other hand, we have seen a trend recently where businesses deploy open source portals and then custom develop the portlets that aggregate data from the proprietary systems in the back end. In many cases, the cost savings were immense, even when custom portlet development was outsourced to a third party as opposed to having paid the licensing and consulting costs from the proprietary vendors. In addition, a business may find the ROI acceptable for many more projects using an

open source portal instead of an expensive proprietary portal. The open source portal will save a lot of developer time and reduce project risk because the presentation architecture is standards based and community tested where the in-house custom presentation infrastructure is not.

Does the portal vendor offer reliable support services for its products?

As with any piece of software, reliable support is extremely important. This is especially true with portals, as there is often a diverse set of components sitting under the hood managing a myriad of operations. Although some open source portals are backed by proven and professional support services, some of them will leave your development team at the mercy of a mailing list or a message board. Proprietary or not, you should ask who/what/when will be answering your support questions and only contract support from an organization with demonstrated high-quality support and customer satisfaction.

Conclusion

As you can see, there are many advantages to utilizing portal software. The essential decision that will take place will require evaluating whether a portal, CMS, or custom Web application is the path to proceed on. The case for the use of portal software is not cut and dry, but there are numerous advantages in adopting it like leveraging existing technology to perform the heavy lifting associated with Web application development, numerous open source and proprietary vendors backing their products with support services, and a shallow learning curve for JEE developers with regards to the portlet specification and existing MVC frameworks. ☺

Listing 1: A sample portlet configuration file

```
<portlet>
  <portlet-name>ContentPortlet_1</portlet-name>
  <portlet-class>org.jboss.portal.core.portlet.cms.
    ContentPortlet</portlet-class>
  <supported-locale>en</supported-locale>
  <resource-bundle>Resource</resource-bundle>
  <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>VIEW</portlet-mode>
  </supports>
  <portlet-info>
    <title>Ferrari</title>
  </portlet-info>
  <portlet-preferences>
    <preference>
      <name>uri</name>
      <value>/default/side.html</value>
    </preference>
  </portlet-preferences>
</portlet>
```



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*

Navigating the Global Enterprise

by Alex Maclinovsky

Developing a ubiquitous navigation utility

In the five years that I have worked in Web solutions practices, a typical business problem has changed from “we need a new Web site” to “we need to regain control over our existing sites.” It’s not uncommon for large corporations to have hundreds or even thousands of different Web sites spread over various service lines, geographies, and organizational boundaries. This presents challenges ranging from logistical and technical, to creative, business, and legal. This article focuses on solving the problem of ubiquitous navigation across diverse *Webscapes*.

In the Beginning There Was the RFP

Let’s start with a real-life problem. A global financial services company whose products you probably carry in your wallet identified a need for a ubiquitous navigation utility. They have to manage myriads of intra- and inter-site references spanning nearly 1,000 of their Internet, intranet, extranet, and partner Web sites across the globe. It should maintain the infrastructure of navigation assets, such as familiar top, left, and bottom navigation bars, site maps, navigable taxonomies, features and highlights, inline ads, and cross references, while ensuring that all content is reachable and there are no broken links.

Being a large company, it issued an RFP and received numerous responses, mostly from product vendors, including a Web testing tool and several portal and content management platforms. Unfortunately, none of them would satisfy *all* the requirements. CMS is able to publish the assets and can do some initial validation, but cannot maintain the validity of links at runtime. Portals don’t have that problem, but can only maintain links to their own pages. Site crawlers can detect broken links only *after* they appear on the sites.

My team was the only services organization that responded to the RFP. To differentiate against stronger competition, we needed to come up with a perfect solution that would satisfy all the requirements, while delivering comparable or better scalability, maintainability, and, ultimately, value. Thus we came up with the idea of a target-centric navigation solution.

Target-Centric Navigation Solution

The only way to ensure that the site never displays a broken link is to validate all the links at the page rendering time, suppressing the ones that point to unavailable resources. The simplest way to do it is to use classes from package `java.net` as shown in the following code:

```
public static boolean isBrokenLink(String
linkURL)
{
    try
    {
        // Throws MalformedURLException if
        //invalid syntax
        URL toTest = new URL(linkURL);
        // Throws IOException for some URL
        //types
        URLConnection c = toTest.open-
        Connection();
        // Throws IOException when site not
        //found
        c.connect();
        // Throws IOException when page not
        //found
        c.getContent();
        return false;
    }
    catch (MalformedURLException ex) {}
    catch (IOException ex) {}
    return true;
}
```

However, a single page can contain dozens even hundreds of links, many of them pointing to dynamic pages, so validating each one using the above method would result in very

poor performance. Furthermore, this method will not work at all if the target site intercepts its own 404 errors and redirects them to a **not found** page. A better solution would be to create a Target Registry containing records about every *navigable target*, including pages, documents, navigation nodes, and media streams. Every navigation link on the page can then be validated just before rendering with a single query to the registry, ensuring optimum combination of performance and reliability.

Sometimes it is not sufficient to remove a broken link by itself – when it is accompanied by the contextual information, such as text or images, the entire page element should be removed. A good way to implement navigation assets is to assemble them from *contentlets*, or elementary units of dynamic content presentable on a Web page, as introduced in the article I co-wrote with Alexey Yakubovich about Vitrage framework, “*Development of Component-Oriented Web Interfaces*” (*DDJ*, Vol. 10, issue 3). A contentlet has a number of core attributes: Name, Description, Image, and Display Order. Additional attributes, such as Date, Type, and Source, may be represented as needed for a particular problem domain or implementation. All of these attributes are optional. *Content Reference* is a type of contentlet that points to a specific URL. As shown in Figure 1, there are several concrete types of content references pointing to different types of targets, such as dynamic pages, documents published through CMS, external links, and media streams.

Due to the diversity of possible targets, the target registry can be realized as a single entity or as a logical collection of target-specific registries. PageRegistry would contain the information about all dynamic pages in the system. It usually has to be imple-



Alex Maclinovsky is a principal application architect with SeeBeyond Technology’s Center of Excellence specializing in composite applications based on J2EE technologies. For over 15 years, he has focused on architecting and developing large distributed object systems on the enterprise on a national and global scale. Alex’s professional interests include solution-oriented architectures, adaptive frameworks, and OO methodologies, and his professional profile can be found at www.geocities.com/maclinovsky/pro amaclinovsky@seebeyond.com

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	4
Arcturus Technologies	www.arcturustech.com	703-822-4582	31
ceTe Software	www.dynamicpdf.com	800-631-5006	39
Common Controls	www.common-controls.com	+49 (0) 6151/13 6 31-0	45
ExtenTech	www.extentech.com/jdj	415-759-5292	49
Google	www.google.com/jdj	650-253-0000	35
ILOG	http://diagrammer.ilog.com	800-FOR-ILOG	25
InetSoft	www.inetsoft.com/jdj	888-216-2353	27
Information Storage & Security Journal	www.issjournal.com	888-303-5282	57
InterSystems	www.intersystems.com/cache14p	617-621-0600	Cover IV
IT Solutions Guide	www.sys-con.com/it	888-303-5282	59
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	47
Jinfony Software	www.jinfony.com/jp9	301-838-5560	41
M7	www.m7.com/power	866-770-9770	33
MapInfo	www.mapinfo.com/sdk	800-268-3282	19
Microsoft	microsoft.com/connectedsystems		Cover II
Northwoods Software Corp.	www.nwoods.com/go	800-434-9820	53
Parasoft Corporation	www.parasoft.com/jdjmagazine	888-305-0041	7
Perforce Software	www.perforce.com	510-864-7400	11
Phoneomena	www.phoneomena.com	352-373-3966	42-43
ReportingEngines	www.reportingengines.com	888-884-8665	23
Smart Data Processing, Inc.	www.weekendwithexperts.com	732-598-4027	61
Software FX	www.softwarefx.com	800-392-4278	Cover III
Synaptris	www.intelliview.com/jdj	866-99VIEW	37
SYS-CON Website	www.sys-con.com	888-303-5282	55
Windward Studios, Inc.	www.windwardreports.com	303-499-2544	15

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.

Subscribe Today!

— INCLUDES —
FREE DIGITAL EDITION!
(WITH PAID SUBSCRIPTION)
GET YOUR ACCESS CODE INSTANTLY!



The major infosecurity issues of the day... identity theft, cyber-terrorism, encryption, perimeter defense, and more come to the forefront in ISSJ the storage and security magazine targeted at IT professionals, managers, and decision makers

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only \$39⁹⁹ ONE YEAR 12 ISSUES

www.ISSJournal.com
or 1-888-303-5282



The World's Leading i-Technology Publisher

mented from scratch; however, it is well worth the effort because it can be used not only to support navigation, but also for search, security, automatic site maps and indices, supporting page life cycles, etc. Document Registry or CMS_Cache usually comes as a part of the Content Management System and does not require any custom implementation. Approved External Links store can be implemented as a part of PageRegistry or on its own, and similarly has uses beyond navigation. Regardless of the implementation details, use of target registries would allow bulk validation of all navigation assets on a page in a single though possibly complex query.

Implementation Considerations

The Links

Content References are flexible enough to implement any navigation assets found on today's Web sites. They have a built-in validation mechanism to prevent rendering of broken links at runtime while maintaining the correct structure of the remaining page. When used in conjunction with a compartment-oriented presentation framework such as Vitrage, which allows bulk loading of all contentlets used on a page in a single database hit, content references have the same or better performance than any other dynamic navigation mechanism.

Navigation compartments can be easily extended to non-Java technologies, such as static, PHP, and even ASP pages. You can define a special type of compartment, which, instead of gen-

erating HTML into a ServletResponse, would rewrite a predefined region of an external file.

If you opt to not use compartment-oriented presentation framework, you can still implement a deferred contentlet-based validation through the use of a Response Filter from package javax.servlet. To use this mechanism described in detail in "Adding Internationalization to Business Objects" (WLDJ, June 2005), each contentlet should be rendered surrounded with meta-information tags containing references to the target registries. The response filter then parses ServletResponse, finds all the tags, validates them against the registries, and removes the tags along with invalid content references.

The Registries

So far we assumed a single target registry at least per target type. However, for an organization that maintains hundreds of Web sites hosted in dozens of locations, this approach would not work due to large amounts of WAN traffic required to validate every link on every site. Such cases call for a distributed solution based on federated registries. There are multiple ways to implement such a federation, ranging from a centralized replication tree to a true federation of independent registries. We found that a *domain-based federation* offers an optimal combination of control and performance. In this approach, a master copy of each target description is stored in a single registry or *domain*, usually collocated with the site that hosts this target. A copy of this record is kept in each registry collocated with sites that contain links to that target. These secondary registries are responsible for keeping it up to date with the master.

The External Links Store differs from other target registries, because the actual targets are usually outside of the control of the organization. It can be kept up to date via a crawler that would continuously go over the store, validating links through a mechanism similar to one described in the beginning of the article. This would happen in a separate process and would not affect the performance of the site.

The Control

Both the target description and contentlet-based navigational assets can be centrally controlled and published via a Business Object Publishing Service, as described in the article "Publishing Business Objects in Portals" (WLDJ, July/August 2004). This mechanism allows authoring and controlling *highly structured content* or application data in a content management system. Using a CMS has multiple advantages and allows navigation administrators to use a familiar environment and to leverage all versioning, security, and workflow mechanisms available in it. The Business Object Publishing Service pushes this data into target databases while maintaining its structure, semantics, and referential integrity.

Conclusion

Maintaining uniform, up-to-date, and reliable Web navigation has become an accepted challenge in today's enterprise infrastructure. Although many tools on the market can help to address parts of this problem, there is not a single one that offers a complete end-to-end solution by itself. The proposed target-centric navigation approach described in this article, in combination with compartment-oriented presentation frameworks and business object publishing services, can offer such a solution that would meet most functional and non-functional requirements. ♦

References

- Maclinovsky, A., and Yakubovich, A. "Development of Component-Oriented Web Interfaces." *JDJ*, Vol. 10, issue 3: <http://java.sys-con.com/read/48536.htm>
- Maclinovsky, A. "Adding Internationalization to Business Objects." *WLDJ* (Vol. 4, issue 3): <http://java.sys-con.com/read/102698.htm>
- "Publishing Business Objects In Portals." *WLDJ* (Vol. 3, issue 6): <http://www.sys-con.com/story/?storyid=45559&DE=1>
- Web Developer's Virtual Library contains many useful resources on navigation: <http://wdvl.internet.com/Location/Navigation/>

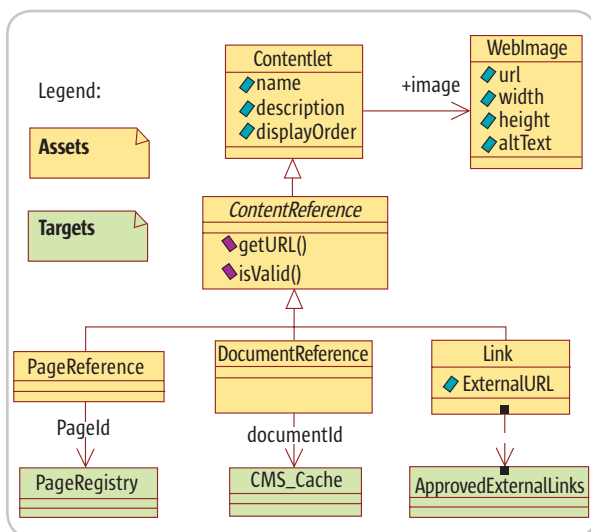


Figure 1 Conceptual model of target-centric navigation

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...



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

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

Get Listed as a Top 20* Solutions Provider

For Advertising Details Call 201 802-3021 Today!

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



The World's Leading i-Technology Publisher

JCP Launches New Program

First constellation of Star Spec Leads takes shape – Part 2



Onno Kluyt

In the August issue of *JDJ* (Vol. 10, issue 8) I introduced to you some of the JSR Spec Leads who won the distinction of *Star Spec Lead* at JavaOne. What they all share, I was noting, is their passion for Java and their belief in the benefits of evolving the platform based on binary standards that ensure compatibility, which can make developer life a lot easier and save costs of all kinds down the road. It's now time you met the other stars of the constellation.

I'll start with *Mark Hornick*. He admits from the start that his keen interest in object-oriented design and development dates back to grad school. He confesses that Java technology had been of particular interest to him, "Ease of programming, especially with threads, was particularly exciting," he says. Mark is currently a senior manager in the Data Mining Technologies Group at Oracle Corporation and by the time he joined the company, he was ready to undertake two Java technology development efforts: Oracle Data Mining and Oracle Personalization. Oracle Data Mining was the seed that grew into the Java Data Mining (JDM) standard, versions 1.0 and 2.0 (JSR 73, JSR 247), which Mark initiated as Spec Lead and began participating in the JCP program mid-2000.

Mark had a head start on the Star Spec Lead program, having been nominated for "Most Outstanding Spec Lead" in 2004 and again in 2005. Mark says, "I think the Star Spec Lead program is useful in that others can learn how Expert Groups are run and who is running them. It creates more of a community rather than abstract names on JSR numbers."

Jere Käpyaho is also a Star Spec Lead. He works for Nokia Corporation as a specialist for Java Platform Standardization, in the technology platforms unit. He thinks "there is so much potential in Java technology to

do globalized applications properly" that he is set on tapping it for the JCP standards initiatives he's working on. He became a Spec Lead in 2004, previously served as an Expert Group member, and also helped with other JSRs that Nokia, a member of the Executive Committee, is involved in. He has contributed to six JSRs in all: JSR 238 Mobile Internationalization API and JSR 258 Mobile User Interface Customization API as a spec lead; JSR 75 PDA Optional Packages for the J2ME Platform and JSR 204 Unicode Supplementary Character Support as an expert; and JSR 118 Mobile Information Device Profile 2.0 and JSR 139 Connected Limited Device Configuration 1.1

Another spec lead from Nokia, *Kimmo Loytana*, wants "to make Java technology in embedded devices a rich and robust platform for applications." A Star Spec Lead himself, Kimmo has been heavily involved in the standardization of Java technology and Java technology-based software platforms and serves as a consultant for the creation of Java technology implementations for Nokia's products. Kimmo had gotten involved in Java API specification activities in cooperation with Sun even before the JCP program existed (JavaTV API, partly also following JavaPhone API). From the very beginning in 1999 Nokia formally joined the JCP program as a member, and since then Kimmo has participated in more than 15 JME-related JSRs as an Expert Group member.

The next Star Spec Lead takes us to Day Software in Switzerland. *David Nuescheler*, the company's CTO, began working with Java technology when the decision was first made to adopt the Java Platform for Day's entire suite of products. Since then, he has worked primarily on server-sided Web projects as a solution and product architect. By 2001, he had joined the JCP program,

and he is currently Spec Lead for JSR 170 Content Repository for Java Technology API.

About the Star Spec Lead initiative, David says, "The Star Spec Lead program is a great platform for exchanging experiences between Spec Leads, probably the best way to learn how to run a JSR..."

When it comes to involvement with Java technology, *Eric Overtom* may be one of the most recent converts, but he has certainly made up for it by jumping straight into Spec Leader stardom.

As a Distinguished Member of the technical staff at Motorola, Eric is involved in handset software architecture and internal API design, and, to some degree, system architecture. He became involved with the JCP program in September, 2004, when he became co-Spec Lead of JSR 253, along with Ekaterina Chtcherbina of Siemens AG. "Having both developers/manufacturers and users has helped us to identify requirements that didn't come from the initial proposed API, and cases where the API as proposed would have led to extra work for application developers trying to use it (although less implementation work for the JSR developers). We've been able to balance the complexity between API implementation and application to share the pain," says Eric.

Vincent Perrot of Sun Microsystems' Telecom Management Network team is also the recipient of the Star Spec Lead distinction. He got involved with the JCP early on first as an observer of one of the earliest Java Specification Requests, the JSR 3, Java Management Extension (JXM) Specification. He helped develop and later became the technical lead of JDMK (Java Dynamic Management Kit), Sun's implementation of JMX. He stays involved in the creation of this network management

Onno Kluyt is the director of the JCP Program Management Office, Sun Microsystems.
onno@jcp.org

technology by participating as an observer of the JMX technology JSRs. (JSRs 70, 71, 146, 160, 255, and 262).

Vince, today, is the driver of the OSS/J specifications. "Working on OSS/J is by far the most enriching experience in my professional life," he says. "It's a privilege to work everyday with some of the best telecommunications experts in the world. And what many people would consider as a human challenge, quickly turned for me into a fantastic journey along which I made friends that I admire, and who I hope do care about me. But OSS/J gets all of us so busy and focused that there is no time to dream about awards. When I was told that I was chosen by my Java developer peers as a JCP Star Spec Lead, I was really taken by surprise and deeply moved. I was only hoping to get the respect of my OSS/J fellows. So being distinguished by the Java community at large was like reaching the summit of the mountains that surround my house, after a 10-hour hike, to see the sunset over the Alps – a warm feeling of accomplishment. I sincerely hope my personal experience will help other Spec Leads move the Java platform to even greater success."

When it comes to Java technology, *Jim Van Peurse* does nothing by half measures. This has not passed unnoticed and his peers in the community voted him Star Spec Lead. He is drawn to all aspects of Java technology that relate to mobile and wireless communication. He's delved into Java technology in practically every way imaginable – as user, project lead, developer, vendor, and so on. Even his e-mail signature, *jvp*, looks like a Java technology acronym. He started working with Java technology in the very early days, say 1995-ish, before it was even integrated with the Netscape browser. He jumped into the JCP Program the minute the doors opened and has been involved ever since as a member, participant, Expert, Spec Lead, and Executive Committee member.

Jim found the Java technology world irresistible. At Motorola, Jim is a Fellow of Technical Staff, holding a PhD in computer engineering. Jim was part of the Motorola team that worked with Sun labs on the Spotless VM that became the KVM. From within Motorola, Jim has been responsible for many aspects of Java technology deployment, from an independent Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP) implementations, to handset development, to working with the industry in defining many related standards. MIDP is an essential technology. Based on the number of units shipped that incorporate it, MIDP is arguably the most successful computing platform in the world. In addition, MIDP contains a lot of functional areas that span a broad range of required expertise. Jim says, "As you can imagine, the combination of these two factors means that a lot of companies and people want to join the Expert Group to shape the solution, and companies want to have several people with different expertise to participate in the different areas. Contrary to some JSRs, we prefer to adopt a more inclusive model and enable broad representation. This leads to a much larger Expert Group than is typical." For example, 122 people participated in the MIDP2 Expert Group.

One of the Spec Lead's most critical tasks is to stay in constant communication with the Expert Group members, who should all feel they have an equal voice in the direc-

tion of the JSR solution. This is especially tough with such a large Expert Group, where, for example, face-to-face meetings are handled in a unique way. "It's not practical to have a productive working meeting with 100+ people in a room. So what we did was create two segments of members of the Expert Group. Those who had a direct shaping influence in the market, versus those who didn't." For example, in MIDP2 the first category consisted of device manufacturers and network operators, while everyone else was placed in the second category. For MIDP3, VM vendors were moved into the first category since the spec touches on some issues that dramatically impact VM vendors.

For each face-to-face meeting, every company within the first category is allowed to send one representative. A few extra seats are reserved for people in the second category to attend. People are selected from the second category using a kind of round-robin lottery system, giving everyone a chance to attend at least one face-to-face meeting if they want to. What gives all Experts an equal voice, however, is that everyone has full access to the e-mail discussions.

Would you like to participate? To become part of the community that shapes Java standards for compatible solutions? Don't hesitate to get in touch with these folks; you'll find it all out directly from the champions: what it means to be a JCP member, to contribute, have Spec Lead responsibilities, and take a specification to the finish line. For more information about their accomplishments and contact information visit <http://jcp.org> and the pages of the JSRs they're leading. ☺



WEEKEND WITH EXPERTS

Get a Java injection on the go!

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

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

The next Roadshow is in New York City on October 15 and 16.

www.weekendwithexperts.com

Do I Really Need That?



by Jason Bell

In February I took on the daunting task of starting a new venture. It was based on an idea I had while reading a book on the low cost airline,

Ryanair. I never knew you could lease an aircraft; I thought an airline with billowing amounts of cash just bought the machines and got on with it. Wrong, wrong, and wrong with a capital W. My rationale was simple: there are a lot of aircraft on the ground, let's help get them back up in the air. How can I provide a system that makes sure that both parties benefit. B2B auctions!

Aerleasing is an enterprise auction engine for the airline industry. More to the point, it's 100% Java. The requirements were simple. Use as many open source libraries as possible so there is no major outlay. Borrow no money; use only what you have on hand.

The most expensive thing was hiring an excellent graphic designer. Brand is still everything, no matter how good the programming is. With no brand identity, you're dead in the water before you even start. So February was spent phoning, talking, and listening to as many people as I could get my hands on. I don't think I have ever learned so much in a short space of time.

I settled on Sitemesh for my template framework, mainly because I had used it before and could get up and running easily. I wanted users to be able to upload assets to their auctions (such as images and documents), so the Jakarta Commons file upload was an obvious choice and easy to implement.

The first prototype got a lot of reworking after a few of my contacts commented on the system. "Can you get a PDF copy?" they asked. No problem! I had a couple of choices, FOP or iText. I found iText was excellent in providing

PDF documents from a servlet. Easy to set up and I put images in the document as well. If you don't mind crafting XSLT stylesheets, then FOP is wonderful too.

In the original version of Aerleasing I provided RSS feeds (using Rome to generate them), thinking I could convert an entire industry into using this wonderful data. Not so. I spent more time explaining it and it still caused confusion. What these folks live on are spreadsheets, so why try and fix something that isn't broken. Jakarta POI was downloaded and worked on; in fact it's a work in progress but I've used it a handful of times before so I know what's going on.

I spent a lot of time thinking about how auctions could be updated in terms of their start and stop times. After a bit of Googling around I found Quartz as one thing I did require was that the time worked outside of the app server just in case the server crashed for whatever reason. I didn't want the timer stopped because the server stopped.

The site launched in early July and was creating a bit of stir with some of the industry press. Nothing like the concept of Aerleasing has been seen before, but that still didn't stop me from having to pick the phone up and cold call some companies. As a technical architect it was a bit nerving but now I actually enjoy it. They don't ask me about SOA, RSS,

J2EE, or SOAP but they do ask about the system and the benefits of it.

If you have the drive, you can work in any industry you want. More to the point, you can take your skills of Java, programming, and analysis and start crafting systems that will possibly change the way people work. Some of you are doing it already. I take my hat off to you.

One thing that came out of all of this is how much users are not really bothered about technical requirements. As I said before, RSS feeds didn't figure much in anyone's thought process, quite the opposite from where I sat. This train of thought was backed up by a piece on the Forbes Web site stating that 91% of Internet users still don't know what RSS actually means. A sobering thought considering the number of RSS libraries that are out in the Javascap.

Another surprise was trying to find an auction engine. You'd think that SourceForge or one of the other open source repositories would have something. With the exception of a demo engine for Hibernate, there was nothing that I could find. Searching on SourceForge brought up nothing; plenty of utilities to put the last bid on eBay, but not an engine. At the end of the day it's no real worry. I had the ability to put my own system together. In terms of getting something out in the open though, writing my own cost me in time. I missed the Paris Air Show by two weeks.

So now I stand here as a founder of a company, all my own technology, no funding (that's another long story for another time) and I am loving every minute. If you've ever wondered, "what if...?" don't wonder any more. Do it!



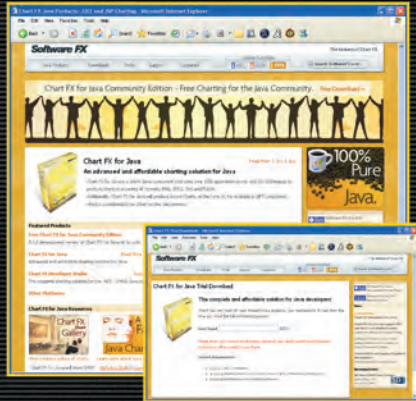
“ You can take your skills in Java, programming, and analysis and start crafting systems that will possibly change the way people work”

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

jasonbell@sys-con.com

Zero To Chart

In Less Than An Hour!



9:04 am

To learn more about the Chart FX products visit www.softwarefx.com and decide which one is right for your platform and target. Then download the 30-day trial version or the Chart FX for Java Community Edition which is a FREE, non-expiring, full development version.

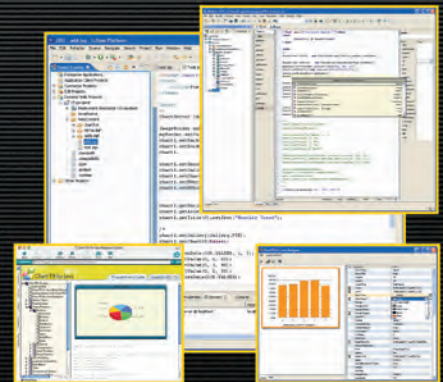
9:07 am

After download, simply install the product appropriate for your needs, platform and target environment.



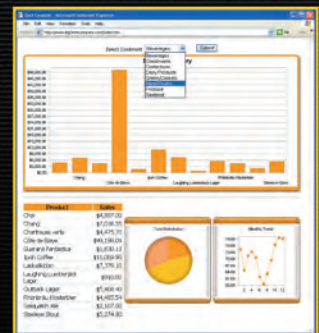
9:13 am

Open the Chart FX for Java Designer to get started with the look and feel of your chart. Use your favorite IDE to add functionality and to populate the chart by connecting to your data source. The Chart FX Resource Center is also available to help with any questions you may have. Then deploy to your specific application server. ➤



9:58 am

Your application is then displayed with an active chart or image that makes any data look great!



Ready... Set... Download!



Innovations by InterSystems



Rapid development with robust objects



Lightning speed with a multidimensional engine



Easy database administration



Massive scalability on minimal hardware

No More Object-Relational Mapping.

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

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



Try an innovative database for free: Download a fully functional, non-expiring copy of Caché, or request it on CD, at www.InterSystems.com/Cache14P

* Read about our money-back guarantee at the web page shown above.
© 2005 InterSystems Corporation. All rights reserved. InterSystems Caché is a registered trademark of InterSystems Corporation. 8-05 CacheInno14PJaDeJo