IS JAVA A "BALL AND CHAIN"?

PAGE 3

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

JDJ

JDJ.SYS-CON.COM VOL.12 ISSUE:2

SVT: SHIP HAPPENS

Insights from the SWT community

RETAILERS PLEASE DISPLAY
UNTIL APRIL 30, 2007

$5.99US $6.99CAN 03>

0 09281 01751 6

PLUS...

- The Benefits of Virtualized WebLogic Clustering
- Code Development in Distributed Environments

Altova® Missionkit™ 2007 – Intelligent tools for XML developers and software architects.

ALTOVA® missionkit™ 2007

Gear up for development excellence

Take off with the Altova MissionKit, and discover the secret to savings on top software tools.

Spied in the Altova MissionKit 2007:

- The world's leading XML development tools
- Plus application design, data management, and modeling options for software architects

The Altova MissionKit 2007 bundles Altova's intelligent application development, data management, and modeling tools at 50% off their regular prices. Available in a variety of configurations tailored to the needs of software architects and XML developers, the Altova MissionKit delivers the highest functionality and best product value. It's your first-class ticket to the power, speed, and simplicity of Altova's award-winning product line. Save a bundle!

Download the Altova MissionKit 2007 today: www.altova.com

ALTOVA®

JAVA DEVELOPER'S JOURNAL

Editorial Board

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

Production

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

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

Subscriptions

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

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

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

Editorial Offices

SYS-CON Media, 577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677
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., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. Periodicals postage rates are paid at Woodcliff Lake, NJ 07677 and additional mailing offices. Postmaster: Send address changes to: Java Developer's Journal, SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

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

Is Java a "Ball and Chain"?

Jeremy Geelan

These are curious times just now for Java. In one and the same month, Steve Jobs stands up, and declares – referring to language support on the new Apple iPhone – "Java's not worth building in. Nobody uses Java anymore. It's this big heavyweight ball and chain." And in the same month a company like Backbase, whose AJAX JSF Edition is aimed at "Java developers who want to leverage the JSF standard by creating a next generation rich component-based AJAX presentation tier," wins a 'Technology of the Year Award 2007' in the category 'AJAX Toolkits.'

So, is Java toast, history, finished, a sucked orange…or does it have plenty of "legs" yet, and Job's remark was just a temporary techno-backlash such as all languages must resist from time to time?

Bruce Eckel, who has since 1986 has published six books and over 150 computer articles, views this backlash as inevitable, foreseeable almost:

"This backlash has only been necessary because of Sun's death grip on the idea of ubiquitous, omniscient Java. It was admirable once, but a language only evolves if its designers and advocates can acknowledge problems. Pretending that a language is successful in places where it's not is just denial."

But the Jobs declaration strikes as some as being a little incongruous.

"Am I the only one that finds this interesting since the format Apple is supporting for HD content is BluRay, which uses Java for all the interactive menus or *BD-J discs*," notes Danny Mavromatis. In other words, Jobs "is supporting a next-gen format which supports a technology that he claims nobody uses anymore."

Jobs's remark was made in an interview with *New York Times* technology correspondent John Markoff, but there must be more than a suspicion that it was calculated to help generate exactly the kind of massive publicity that will be necessary if Apple is to come anywhere near selling the 10 million iPhones that Jobs was predicting for 2008.

Richard Sprague offers a cautionary tale:

"I remember the lessons I learned working with the Newton team many years ago. I was in Apple's marketing department at the time and we did this big fancy user study which basically proved that nobody would buy the thing at the price and functionality we were building. So what did we do? We shoved it into the market anyway because it was "cool". Cool is great, but you still need to make phone calls."

Back to Eckel, though. Here is his take on a major flaw in Java versus AJAX:

"So Java has been around for 10 years and applets are not the primary way that we interact with the web. I think the main reason for this is the installation problem, another area of Java that wasn't well thought-out. In fact, why do we like AJAX?

It's clearly not because JavaScript is so easy to work with — JavaScript cross-platform problems are the reason people have avoided it in the past. AJAX is popular because we know that the necessary software for the client side is *already installed*.

Someone had to figure out how to deal with the cross-platform issues for JavaScript first, but if JRE installation was trivial, everyone might have just created Java applets. But they didn't, applets are not ubiquitous, and everyone got excited about AJAX instead. So AJAX became the favored technology for RIAs."

According to Eckel, the obvious contender, instead of Java, for building RIAs is Flash, and Flex in particular.

"It's clear that we can't wait for Sun to fix all of Java's problems," he writes. "Open-sourcing Java might, eventually, have a huge impact on repairing Java's deficiencies. For example, work on the JMF might get resurrected. Maybe installation issues will even be fixed someday. The possibilities might be limitless, but if you need to solve problems *now*, then the solution is to hybridize parts of the language."

By way of explaining this concept of "Hybridizing Java," Eckels explains that in fact we do this already:

"You don't insist on using a Java database for an application; you use a specialized system like MySQL or Oracle. Sun is directly supporting the development of JRuby for hybrid Java/JRuby programming. We are seeing other special-purpose languages arise to solve specialized problems. Why insist on using a Java library for UI if a specialized system solves the problem better?"

Let's give the last word to Steve Benfield, veteran technologist, who summarizes what he calls his "technology lineage" as PowerBuilder -> Silverstream -> J2EE -> AJAX -> Flex.

"If you are a Java technologist who thinks anything Flash isn't enterprise ready," Benfield states, "then you need to reshift your thinking." He adds:

"We started using Flex 3 months ago and are rocking and rolling – life is good. We can quickly build the GUI we want, integration to our J2EE/Spring/hibernate back end is seamless, and we anxiously await Apollo so we have a full desktop app."

Like I said, these are curious – and challenging – times just now for Java.

Jeremy Geelan is Sr. Vice-President, Editorial & Events of SYS-CON Media. He is Conference Chair of the AJAXjWorld Conference & Expo series and of the "Real-World Flex" One-Day Seminar series. From 2000-6, as first editorial director and then group publisher of SYS-CON Media, he was responsible for the development of all new titles and i-Technology portals for the firm, and regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas. He is executive producer and presenter of "Power Panels with Jeremy Geelan" on SYS-CON.TV, and is actively helping build out the AJAXWorld brand as well as developing entirely new Conferences and One-Day Seminars for SYS-CON Media & Events.

jeremy@sys-con.com

FEBRUARY 2007 VOLUME:12 ISSUE:2

JDJ contents

JDJ Cover Story

Interview by Joe Winchester **26**

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. Periodicals postage rates are paid at Woodcliff Lake, NJ 07677 and additional mailing offices. Postmaster: Send address changes to: JDJ, SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

Bridging the Gap Between Open Source and Commercial Applications

by Charles Lee

In late 2002, Javier Soltero, Doug MacEachern, Ryan Morgan, Jon Travis, and I (the eventual co-founders of Hyperic) began designing and architecting an application management system that was to become Hyperic HQ. We wanted it to be the management system that bridged the gap between open source and commercial applications and, furthermore, we wanted it to use, be built on, and deployed to the applications and operating systems that we managed.

To achieve that goal, we set out to implement on standards using a cross-platform language. Thus the decision to use Java was pretty clear, as the abstraction of the system-specific runtime freed us from having to figure out native APIs and implement different paths to achieve the desired functionality. We chose JBoss (then mildly popular) as our J2EE application server. When it came to our data persistence strategy, we knew we wanted to stay away from developing our own (Javier and I had already done that before for an Apache configuration management project). Database portability was important, as we had experimented with and implemented PointBase, Cloudscape, and InstantDB for that Apache configuration management project, and knew we would be moving on from databases like bad relationships. Much like bad relationships, you only knew they were over after you had put a lot of effort into them. By 2002, EJB2 had become a standard, and Hibernate was very young and unproven. Compared to EJB 1.0, EJB2 was, in our minds, ridiculously easy to implement. We would use XDoclet to annotate the code, and the whole thing would practically write itself!

We set out to implement to EJB2. True, we were able to quickly map out our object model, and the amount of database-specific SQL was kept to a minimum. We wrote our own tools that functioned as Ant tasks and allowed us to create and populate database schemas for different databases that we supported. However, the runtime was handled by container-managed persistence (CMP); we had no need for bean-managed persistence (BMP). Pretty quickly we ran into some issues such as needing to specify an "order by" clause. While EJBQL did not support "order by," JBossQL did. We thought, "We are only deploying in JBoss at the moment and we will continue to maintain compatibility by defining both EJBQL and JBossQL." Everywhere an "order by" was needed, we had defined both "@ejb:finder" and "@jboss:query" XDoclet tags. We were diligent about it for a while, but then pretty soon we'd forget to fix queries in "@ejb:finder" when we fixed them in "@jboss:query". Then, after more time had passed, we would just plain leave the "@ejb:finder" empty sometimes, especially when we needed outer joins, grouping, or other SQL queries. Since EJBQL did not support the full set of standard SQL syntax, we began to add "@jboss:declared-sql" tags, again a JBoss-specific declaration.

At least we had the CMPs working as we needed them. We were getting caching of CMPs, so we were gaining some optimization. However, it was clear that the performance of CMPs would not be acceptable in some areas. In our case, we collected a voluminous amount of metric data. No matter how much we tried to configure JBoss to not lock pessimistically, we could not get HQ to work with metric data as CMPs without locking up everywhere. We were resigned to using SQL statements and JDBC to access the metric data directly. Once we went down that route, it was easy to decide to follow the same pattern elsewhere in the product. However, since we eventually settled on supporting PostgreSQL and Oracle, these code paths would sometimes require database-specific SQL to get the right behavior. Speaking of pessimistic locks, half of our issues became embroiled in transaction demarcation to avoid the dreaded TransactionRolledbackLocalException (the telltale sign of transaction deadlocks). We had to mark every API as transaction REQUIRED, REQUIRESNEW, SUPPORTS, and NOTSUPPORTED through trial and error. We had been pulling our hair out for the past few years over transaction issues. We implemented our own caching to get around some of these problems, even forking our own version of XDoclet to inject caching behavior into EJB2. There were other issues such as paging of the result set, for which there's absolutely no support in EJB2.

Don't get me wrong, we were able to get this far with EJB2. We are pretty proud of what we have put together in Hyperic HQ, and we accomplished what we set out to do. However, it's 2006 and the landscape is very different. Hibernate has emerged as the runaway preferred object relationship mapping (ORM) tool, and for good reasons, too. With the problems that we had faced with EJB2 and the difficulties with new feature implementation, we knew that the right thing to do was to move forward to Hibernate. However, the task seemed daunting, especially since we had over 80 entities and many tweaks in our application to make it work just right. Frankly, I just didn't think we could ever do it in a reasonable amount of time. However, in the fall of 2006 we bit the bullet and just did it.

It was not the gradual migration that we thought it would be. No, we converted wholesale to Hibernate, and we did it fairly efficiently as well. The change occurred between versions 2.7.6 and 3.0 of Hyperic HQ, with the bulk of the work occurring in just about three weeks. The whole team definitely pulled together for this effort. In my forthcoming articles, I will detail how we went about the conversion and the patterns that we adopted. Hopefully it will benefit others who are in the same boat as we were. We knew that we were in a bad relationship with EJB2, and that it was time to move on.

Charles Lee is vice president of engineering at Hyperic.

SYS-CON MEDIA

President and CEO:
Fuat Kircaali *fuat@sys-con.com*

President and COO:
Carmen Gonzalez *carmen@sys-con.com*

Senior Vice President, Editorial and Events:
Jeremy Geelan *jeremy@sys-con.com*

Advertising

Vice President, Sales and Marketing:
Miles Silverman *miles@sys-con.com*

Advertising Sales Director:
Megan Mussa *megan@sys-con.com*

Advertising Sales Manager:
Andrew Peralta *andrew@sys-con.com*

Associate Sales Manager:
Corinna Melcon *corinna@sys-con.com*

Events

Events Manager:
Lauren Orsi *lauren@sys-con.com*

Editorial

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

Production

Lead Designer:
Tami Lima *tami@sys-con.com*

Art Director:
Alex Botero *alex@sys-con.com*

Associate Art Directors:
Abraham Addo *abraham@sys-con.com*
Louis F. Cuffari *louis@sys-con.com*

Web Services

Information Systems Consultant:
Robert Diamond *robert@sys-con.com*

Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Richard Walter *richard@sys-con.com*

Accounting

Financial Analyst:
Joan LaRose *joan@sys-con.com*

Accounts Payable:
Betty White *betty@sys-con.com*

Customer Relations

Circulation Service Coordinator:
Edna Earle Russell *edna@sys-con.com*

Introducing JReport Live™

JReport® 8

Bring Your Reports to Life

JReport Live is the only solution to empower your Java application with operational business intelligence. Now your application can support both operational reporting and analytics – making users and developers happy.

Powerful Analytics Based on Dynamic Cube Technology

For the first time your users can enjoy ad hoc reporting right from the application, working with the most current operational data. Users can now slice-and-dice data, expand/collapse groups, pivot and drill up/down/across any report. The power of JReport Live is based on our Dynamic Cube Technology. JReport Dynamic Cubes are easy to define and are instantly created with no overhead. Information is always current and presented in the context of the application.

An Enterprise-class Foundation for Operational BI

JReport Live is built on our powerful operational reporting solution, delivering all of the benefits of scalability, performance and security of JReport 8. Plus, you can combine reports into report sets for better performance, easy scheduling and fast deployment. The pipeline mode allows users to start viewing reports before generating a full report. JReport 8 can integrate with any Java EE application and any security scheme. In addition, JReport runs reports on demand, on a schedule or by event triggers.

Build Complex Precise Reports Quickly and Easily

Only JReport 8 is not constrained by a rigid report layout, it lets you mix and match report components and control precisely how they are presented on the page. With multimedia objects, Web controls and Web forms to further enhance reports, your users will be happy to work with the highly functional, interactive JReport Live reports.

Download your version of JReport 8 with JReport Live or call 240-477-1000 today.

Bring your Reports to Life with JReport Live Server

For more information, visit www.jinfonet.com/live

JReport®
JINFONET SOFTWARE

The Benefits of Virtualized WebLogic Clustering

An opportunity to overcome cost inefficiencies

by Ajay Vohra

Across the BEA product family, the principle mechanism for meeting the twin requirements of *scalability* and *availability* for business-critical applications is clustering WebLogic application servers. So clustering WebLogic application servers plays a foundational role across all BEA products and provides the underpinnings for the AquaLogic Service Bus that provides a services infrastructure for Service Oriented Architecture (SOA).

WebLogic clustering does a competent job of meeting the principal twin goals of scalability and availability, but injects some new problems into the mix, which have negative implications for the business objectives that demanded clustering in the first place. In other words, WebLogic clustering keeps the lights on, but introduces cost inefficiencies in managing the services infrastructure, especially in the context of SOA. So, how does basic clustering work? What are its benefits? How does it introduce inefficiencies, specifically in the context of SOA? How are these WebLogic clustering inefficiencies overcome through virtualization? These questions are addressed in this article.

Let's start with a quick overview of basic WebLogic clustering – touching on the key features and benefits – followed by a discussion about how

basic WebLogic clustering introduces cost inefficiencies into the mix, and how these inefficiencies are exacerbated by SOA. Finally, we'll introduce the key features and benefits of virtualized WebLogic clustering, and offer some conclusions related to virtualized WebLogic clustering.

Basic WebLogic Clustering

Basic WebLogic clustering is comprised of identically configured managed WebLogic server instances that can be managed as a unit from an administration server. All the managed servers, in a cluster, belong to the same WebLogic *domain*, whereby each domain defines a set of interrelated resources. Each domain can, of course, define multiple clusters. A cluster, however, can only be associated with a single domain and a single administration server.

Key Features

Basic WebLogic clustering offers following key features:

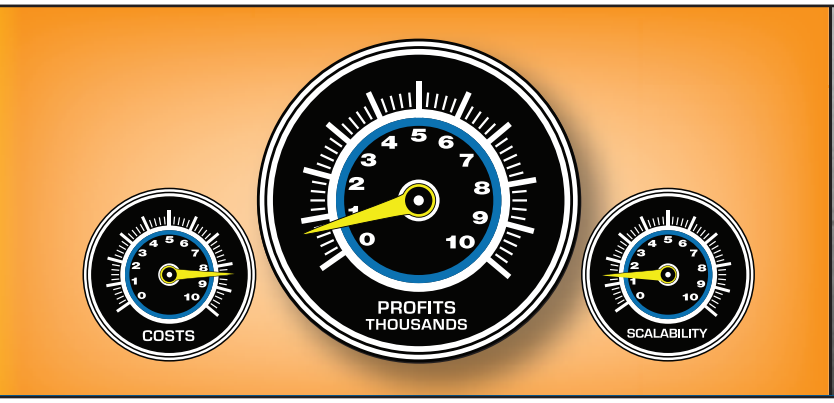
- The core capability of clustering is the *replication* of critical *application components* and their *state* across clustered managed servers, thereby creating the possibility for auto-matic *application failover* from one managed server instance to another.
- This replication capability is bolstered through two related capabilities:
  - Replication of the Java Naming Directory Interface (JNDI) tree that contains references to these objects. Remote clients use the JNDI tree to discover these objects.
  - Clients that need to interact with these replicated objects use replica-aware stubs that are aware of all the replicas in the cluster.
- The ability to *migrate* a complete managed server automatically to another machine selected from a predefined set of available machines is important for migrating *singleton* services that can't be replicated.
- WebLogic clustering provides plug-ins for Web servers, which add load-balancing capability for

> "Basic clustering requires provisioning the cluster for peak demand, which leads to a number of cost inefficiencies that have a multiplicative effect across the application set and can quickly add up, consuming significant chunks of any IT budget"

Ajay Vohra is a Senior Solutions Architect at DataSynapse, Inc., (www.datasynapse.com). His current focus is J2EE application server virtualization. Ajay has an MS in computer science from Southern Illinois University in Carbondale, IL, and an MBA from the University of Michigan Ross School of Business at Ann Arbor, MI.

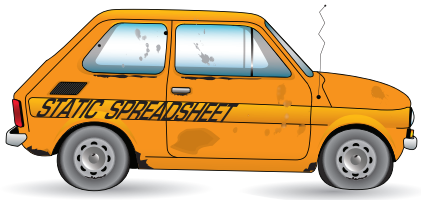
avohra@datasynapse.com

A slapdash approach won't get you there...



ExtenXLS will.

NEW! Round Trip Reporting™



"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 & 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

It's What's behind the Dashboard that Counts!

Slapping a dashboard onto a static spreadsheet file may **look** good, but it's like installing a flashy dashboard on an outdated clunker. Install that same front-end on ExtenXLS — **a scalable, server-based spreadsheet engine** — and watch your flashy dashboard come alive.

Now with Round Trip Reporting™

Reuse your existing spreadsheets as a data-entry tool. Modified data is extracted from your spreadsheets and turned into Java Data Objects for reuse in all your programs.

Reach New Heights without the Learning Curve

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

Escape the Gravitational Pull of Obsolete Reporting!

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

Don't rely on a slapdash approach for your mission-critical reporting needs.
Put a rocket under the hood and achieve escape velocity.

Visit the mother ship at: www.extentech.com/jdj and take your free evaluation copy into orbit for a test flight.

EXTENXLS4™
JAVA | XLS REPORTING TOOLKIT

extenTECH™
Call Us: 415-759-5292

Speed. Simplicity. Style.

download the FREE trial now!

WINDOWS® FORMS ASP.NET WPF JSF

grids scheduling charting toolbars navigation menus listbars trees tabs explorer bars editors

balancing HTTP requests across managed server instances. These Web server plug-ins support HTTP session replication and can automatically detect changes in clusters' managed server set.
- Basic WebLogic clustering uses multicast and socket-to-socket network communications to implement the features discussed above.

Key Benefits

As we alluded to at the outset, basic WebLogic clustering offers two key benefits:
- The first benefit is *scalability*. The overall capacity of a cluster to meet demand can be increased by adding more managed server instances to it. However, in basic WebLogic clustering, the process of adding new managed server instances to a cluster in response to increased demand is largely *manual* in nature.
- The second benefit is increased *availability*. The application failover capability based on the replication of state and critical application components across clusters' managed server set and the ability to migrate complete managed server instances automatically to a different machine offer higher availability for applications deployed on a WebLogic cluster.

These capabilities and benefits are impressive and work fairly well in practice. But, didn't we say something about cost inefficiencies?

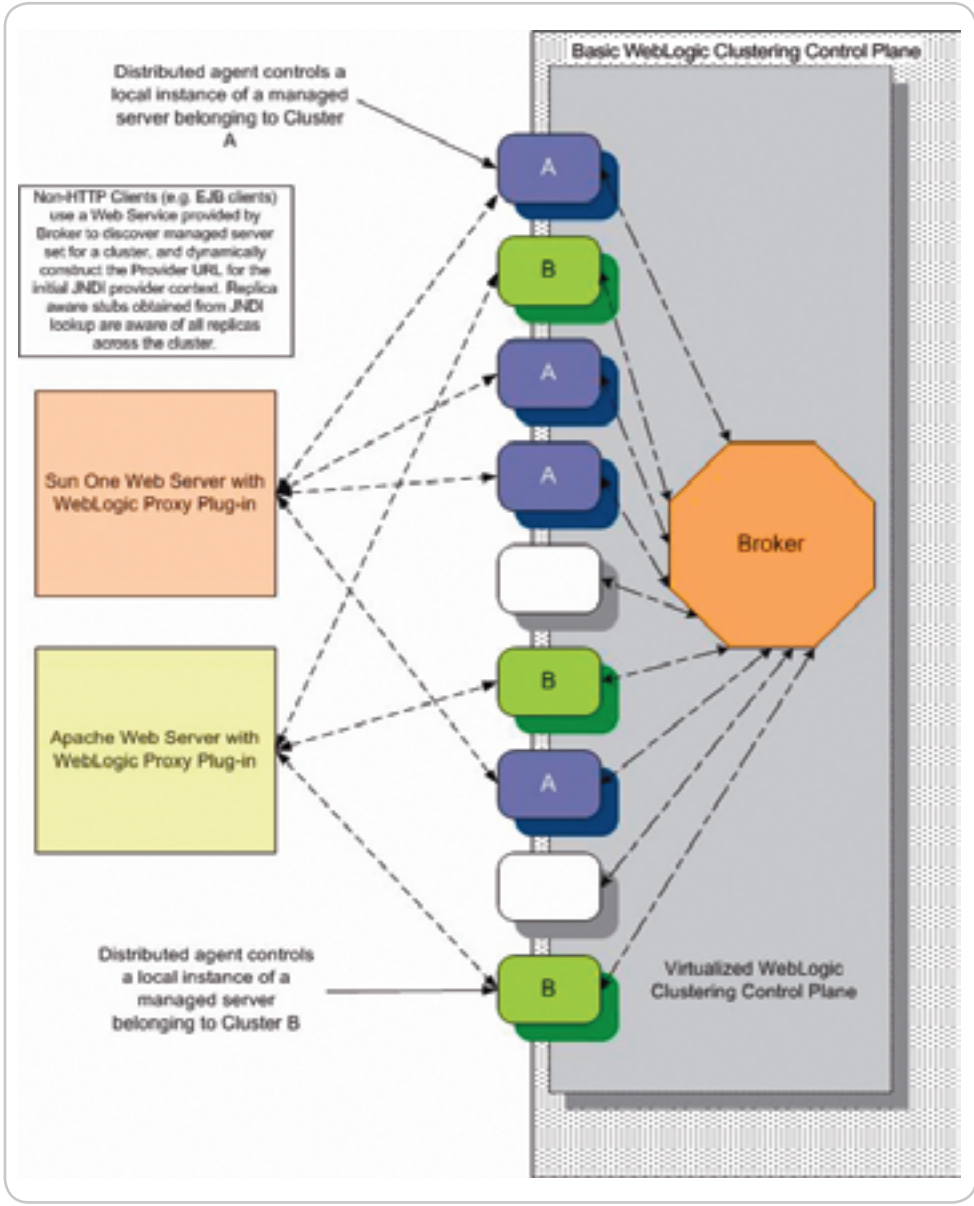


Figure 1 Architecture for virtualized WebLogic clustering

Cost Inefficiencies

Basic WebLogic clustering offers some solid benefits, but introduces the following cost inefficiencies:
- Each application deployed to a cluster needs to be sized for the cluster. This involves estimating peak demand for the application and provisioning enough managed server instances in the cluster so peak demand can be met. This process is repeated for each application. Provisioning for peak demand creates large pockets of underutilized hardware infrastructure capacity. This underutilized capacity creates cost inefficiencies, because each hardware resource has capital, operational, and administrative costs.
- The second issue is that each managed server instance needs to be provisioned with WebLogic software and administered to be brought online as part of a WebLogic cluster. This basically creates inflexible cluster silos that consume significant administrative costs and are largely inflexible in terms of their configurability into different application clusters.

This brings us to the issue of how cost inefficiencies get exacerbated in the context of SOA.

Exacerbation of Cost Inefficiencies Under SOA

SOA is about creating standards-based, interoperable, reusable services that can be loosely coupled to orchestrate complex business processes. In the BEA product family, the AquaLogic Service Bus provides a reliable hub-and-spoke integration system so that arbitrary services can be loosely coupled though two key mechanisms – proxy services and configurable message flows.

> Basic WebLogic clustering is comprised of identically configured managed WebLogic server instances that can be managed as a unit from an administration server"

Naturally, AquaLogic promotes service reuse, but its execution environment is based on basic WebLogic clustering. This means that the cost inefficiencies introduced earlier are exacerbated in the context of AquaLogic, because the more you adopt SOA in your organization, the harder it is to predict the demand profile on a specific service. This is because a given service can be configured into arbitrary business processes. What all this means is that the problem of overprovisioning is exacerbated in the context of AquaLogic. In fact, perhaps anticipating this, AquaLogic provides a dashboard that provides alerts related to the service levels configured in AquaLogic. However somebody has to react to these alerts manually, which brings us to the topic of virtualized Weblogic clustering and how it helps overcome cost inefficiencies.

Virtualized WebLogic Clustering

Virtualized WebLogic clustering, as the name implies, virtualizes the cluster's managed server set, both in terms of size and membership. The members of this set are the machines on which the managed server instances are run.

Key Features

The key features of virtualized WebLogic clustering are as follows:

- Virtualized WebLogic clustering adds a control plane on top of basic WebLogic clustering. This control plane is managed by a centralized *broker* and consists of distributed *agents* installed on a predefined set of machines.
- The predefined set of machines, with distributed agents, becomes the universe from which the managed server set for a given cluster is mapped at any given time. Each machine with a distributed agent is capable of running one or more instances of a managed server.
- A virtualized WebLogic cluster's managed server set is automatically expanded or contracted based on configurable policies, which dictate how the cluster should adapt and respond to its managed server set in response to varying demand.
- The broker, through its distributed agents, effects changes in the managed server set.
- Varying demand is characterized in terms of configurable JMX-based statistics, such as WebLogic Throughput, WebLogic Queue Size, and Active Thread User Count, among others, that are supported by WebLogic application servers and are constantly probed and measured at each managed server instance by the distributed agent and aggregated at the centralized broker.
- HTTP clients access the cluster in the same way they would in a basic clustering scenario. The virtualized WebLogic clustering doesn't introduce any new issues in terms of the load balancing of HTTP requests.
- For non-HTTP clients, such as EJB clients, it's imperative that the clients can create an initial JNDI context based on a JNDI Provider URL. The broker provides a Web Service that helps in the discovery of the virtualized managed server set and thus the construction of the JNDI Provider URL for creating an initial JNDI context.
- The broker is responsible for provisioning all required software to the distributed agents, including WebLogic application server software components and Java software development toolkit software.

The basic architecture for virtualized WebLogic clustering is summarized in Figure 1.

Key Benefits

The key benefits of virtualized WebLogic clustering are as follows:

- Virtualized WebLogic clustering can dynamically change the managed server set in response to configurable policies – clusters don't need to be provisioned for peak demand, which addresses the concerns of cost inefficiencies.
- The general benefits of virtualization are now well understood. It improves infrastructure utilization at all levels of the IT stack. From the adoption of virtualization at the operating system level to the application server level, virtualization can provide direct benefits holistically or independently.
- Through a centralized dashboard, software is automatically monitored, controlled, and provisioned to agents making cluster management and administration simpler and more cost-effective.

Conclusion

Basic WebLogic clustering is critical for scalability and availability and required for business critical applications. However, basic clustering requires provisioning the cluster for peak demand, which leads to a number of cost inefficiencies that have a multiplicative effect across the application set and can quickly add up, consuming significant chunks of any IT budget. The virtualization of WebLogic clusters presents enterprises with an opportunity to help overcome cost inefficiencies by developing an environment that can dynamically adapt a cluster to its demand profile per predefined configured policies.

Code Development in Distributed Environments

Requires effective build management tools

by Matt Gabor
and Steve Taylor

The key to successful build management in distributed environments is a foundational commitment to consistency, repeatability and portability. This is just as true for small, homogeneous development environments using in-house, scripted build systems, as it is for large, complex environments, where a new class of non-scripted, distributed build-management tools are now available.

In small- and medium-sized environments, using a properly implemented in-house build system can mitigate many of the risks and challenges associated with builds. Additionally, with a few simple steps, the burdensome task of developing and maintaining build scripts can be significantly reduced, using build tools such as Make and Ant.

Build Management Evolution

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

But Java development requires a new, platform-independent build tool, which led to the creation of Apache Software Foundation's Java-based Ant, so-named by developer James Duncan Davidson because, "…it is a little thing that can build big things." Ant eliminates Make's platform-dependent wrinkles, and is extended not with shell-based commands but Java classes. Configuration files are XML-based, calling out a target tree where various tasks get executed. Each task is run by a Java class that implements a particular task interface.

Ant is powerful, but the XML configuration scripts can create limitations. XML does not handle conditional logic effectively and it is, therefore, difficult to use Ant to write "intelligent" build scripts that support robust batch processing.

Additionally, many development projects include Java and non-Java components that require both Ant and Make, as neither handles both languages. Scripting for the two is very different. Make scripts are the input to Make programs and dictate how to build each software component.

A Make file tells the Make program which binaries are to be created from which source modules. Make rules are then "fired" based on out-of-date conditions between source and object. In contrast, Ant/XML scripting uses serial batch processing. Rules for creating Java binaries such as .jar, .war, and .ear are handled statically for each step or "task" in the XML script. Listing 1 shows the differences between Make and Ant scripts for a similar type of build task.

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

Make And Ant/XML Challenges

As Client/Server and Java development has evolved, so has build complexity, especially with Make. When Make is used recursively (one Make file per final target executable, or binary – the most common method of managing large build processes), an application with 50 binaries would require 50 Make files plus a "driver" Make file.

The system build is completed by calling the Make program repeatedly and passing a different Make file each time. Dependency checking between the individual Make files is impossible, which means large application Make files can't be managed by a single Make file. Developers get around Make challenges through clever file-ordering to track dependencies, along with object-borrowing and multi-system parallel building techniques to reduce the associated long system build times.

Although most Ant build systems do not appear to be as complex as many Make-based build systems, it is only a matter of time. As Ant scripts suffer from being passed through different developer's hands, as new technology emerges that effects the way Ant scripts are coded or used, and as applications grow more complex, Ant will encounter many of the problems associated with a Make-based system.

The key *to avoiding this* is to implement best-practices for manual scripting starting with an in-house build system, while monitoring factors that would signal the need to move to an automated, non-scripting approach.

Solving Typical Scripting Problems

Scripting challenges are easier to solve in small, homogeneous development environments confined to a single language and target operating system. The first step is to shift from a developer-centric view of builds to a team-centric view, and from the notion of scripting "my build solution" to scripting "our build solution."

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

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

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

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

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

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

Finally, isolate the build scripts to just that: builds. Too often, "build" scripts include substantial pre- and post-build logic unrelated to the build. Pre- and post-build logic can be extremely complex, especially as an application matures and development is being performed on multiple versions simultaneously. The Ant script in Listing 2 demonstrates a build script with a very basic and generic deployment portion. (Listings 2–3 can be downloaded from the online version of this article at http://java.sys-con.com)

Rather than writing pre- and post-build logic within a build script (where the functionality is often limited by the scripting language or tool), place the non-build logic in external scripts. The external scripts should be written in a scalable, lightweight, and cross-platform language such as PERL or PYTHON. Tightly focused build scripts can then have built-in hooks to the external build utility. Listing 3 takes the overly complex build script of the prior example and replaces it with a call to an external script.

By partitioning the build scripts in this way, developers (or build masters) who encounter build problems can drill down to the root cause very rapidly. Additionally, as development grows in complexity and new languages or target Operating Systems are added, the in-house build utility can scale more effectively.

For example, consider a C and C++ development shop that uses an entirely Make based build system with all pre- and post-build logic written in the Make scripts. When the development shop decides to add a Java component to their application, they are faced with writing an Ant component (equivalent to their existing Make scripts) that manages all of Java-related pre-build, build, and post-build logic.

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

Dealing with multiple languages

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

Steve Taylor is an experienced senior developer, bringing 17 years of expertise with Client/Server and mainframe application development and system integration. Prior to founding OpenMake Software Corporation, Mr. Taylor served as a Lead Technical Consultant responsible for the successful implementation of applications into the production environment. In this capacity, he became expert in the use of various configuration management tools and recognized the need to build large applications using a nightly process. At this time he began developing the standardized versioning and build procedures which have since become Openmake. Mr. Taylor received his Bachelors of Science Degree in Computer Science/Mathematics from the University of Illinois-CU.

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

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

Identifying Breaking Points

There are a number of critical "breaking points" that cause in-house build systems to become cost- and/or resource-prohibitive. When they occur, development teams generally begin to consider an automated, non-scripting environment.

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

A sure sign that developers are reaching the limits of manual scripting efficiency is when they find themselves consistently spending as much as an hour a day working on build problems (either their own, or debugging build problems of a centralized CM team). Some companies actually assign a dedicated CM team whose sole responsibility it is to execute builds. Developers find themselves waiting for the CM team to build their applications before they can move on to the next development effort. It can reach the point where the centralized CM team simply cannot keep up with the demand, especially when builds are cross-language, cross-platform and incredibly complex.

Migrating to Automated Build-Management

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

One example of this approach is a build management tool that weaves together human and machine intelligence to automate and standardize the enterprise build process. It is possible to incorporate a browser-based user interface and a Tomcat or WebSphere Application Server to provide access to a Knowledge Base Server. Enterprise-based features allow for the connection to multiple remote build servers. Simple Object Access Protocol (SOAP) is used as the communication layer between the browser and the application servers.

Developers interface through a web client, a command line interface, or indirectly through IDE plug-ins. Build meta-data is stored and managed via the central Knowledge Base Server and reused by multiple developers to generate Ant/XML scripts for Java support, or to generate "Make"-like scripts for traditional build requirements. Build Control files can be generated to build a single object (supporting developer daily compile activities) or a complete application (containing hundreds of inter-dependent modules).

When a complete application Build Control file is generated, it eliminates the problem of recursive Make and ensures the accuracy of incremental builds. Builds can be managed from an empty build directory pulling source code from a pre-defined search path, or by retrieving source code from a version management tool. Build management also allows control over environment variable settings such as LIB, INCLUDE and CLASSPATH so that, regardless of the build machine, the build results are the same.

Build management does not replace Ant for completing Java builds, but rather extends the use of Jakarta Ant without the need for manually coding XML scripts. In the place of hard-coded Make and Ant/XML scripts, for instance, its rules engine takes advantage of a knowledge base of build meta-data, such as Target Name and Dependency information, to dynamically generate portable, PERL-based build processes at build time that can be referenced by multiple development teams.

Listing 1:

```
GENERIC MAKE BUILD SCRIPT
# ================================
# Builds the application executable
# ================================
application: application.c lib_a.o lib_b.o lib_c.o
   @cc g -qcpluscmt -qidirfirst -I. -I/sys_apps/ref_dir/release/include -I/usr/in-
clude -o./exe/application $? -bE:/sys_apps/ref_dir/release/include/application.imp

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

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

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


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

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

Why paint with your fingers?

Java visualization components for desktop and Ajax

ILOG JViews 8.0, the newest release of the ILOG Java-based visualization products, addresses all aspects of advanced visualization.

ILOG JViews 8.0 provides:

- Key visual components: diagrams, dashboards, maps, charts, Gantt charts
- Advanced services: automatic graph layout, high-performance display for large data sets
- Several deployment techniques, including desktop clients, Ajax-enhanced Web applications, Eclipse's Rich Client Platform (RCP) and portals
- A proven track record in the most demanding industries: IT, telecommunications, transportation, utilities, energy and defense

Paint your future with ILOG JViews 8.0.

Get your free 'Ajax for Graphics-Intensive Web Applications' white paper at ilog.com/jdj/ajax

ILOG®

Changing the rules of business™

Joe Winchester
Desktop Java Editor

Those Who Can, Code; Those Who Can't, Architect

At the moment there seems to be an extremely unhealthy obsession in software with the concept of architecture. A colleague of mine, a recent graduate, told me he wished to become a software architect. He was drawn to the glamour of being able to come up with grandiose ideas – sweeping generalized designs, creating presentations to audiences of acronym addicts, writing esoteric academic papers, speaking at conferences attended by headless engineers on company expense accounts hungrily seeking out this year's grail, and creating e-mails with huge cc lists from people whose signature footer is more interesting than the content. I tried to re-orient him into actually doing some coding, to join a team that has a good product and keen users both of whom are pushing requirements forward, to no avail. Somehow the lure of being an architecture astronaut was too strong and I lost him to the dark side.

He'll be in good company though. I was recently called to a customer who expressed interest in a software tool I'm working on. I came armed with the latest build of the product, looking forward to the opportunity to test some ideas and concepts in front of potential users. Instead I found myself in front of the customer who had also invited a competitor in order to create a conference room product shoot out. While I had my PC with running code to show, my opponent had brought along a briefcase full of PowerPoint presentations. Their slides were impressive: good use of color, animation, and a generous splattering of buzzwords and acronyms. Despite the fact I had working code to showcase, the discussion quickly degenerated into a discussion about the fact that mine was a so-called "fat" client, in fact a pretty lean Eclipse RCP-based product, while the opposition had a "thin" client.

The truth was the opposition didn't have a thin Web-based offering; their current product was built six years ago as a desktop application that could be downloaded as an 87M applet. However, they were in the process of rewriting it all to run in a lightweight Java EE container as portlets. In other words, they had nothing. They were peddling vaporwear. Worse than that, despite the fact their company had a perfectly good product offering that I was prepared to go head-to-head with, they seemed to have given up on making it more usable and instead opted for the deep thought option: a total rewrite just to suit the whims of today's architectural fashion.

I kept wanting to take the customer's IT manager and shake him back to reality; however, he somehow got drawn into their trap and was asking me architectural questions rather than focusing on whether the product I had brought to show and tell was going to make his users more productive.

Remember the kid in the playground who knew the name of a band you didn't, or who had a new album? They were cool; they had knowledge we didn't; and whether or not it was any better didn't matter, it was new and shiny and we had to have it too. If we did, then we would also be in possession of knowledge that others didn't own, and we in turn could be the cool kid to someone else.

This kind of atavistic worshipping of the obscure and unknown piece of knowledge is the personality disorder that plagues the software industry and is somehow encouraged and admired by architects who are never satisfied with what they have available to them to build software. They're not innovators or research pioneers pushing knowledge forward though - such people are hugely important as they invent the future and redefine technology boundaries. Instead these silver-bullet junkies just latch onto ideas and fads for the sake of it, because if nothing else it makes them appear ahead of the curve and in possession of secret facts and information. As soon as a project gets into trouble, they can launch these facts at programmers and proclaim, "Aha, it's because you're not using BOJOX and NADA 2.0 combined with YML that you have a bug" in front of the nervous manager who wants nothing more than to buy more time by telling his reporting chain that he needs a year to do a total rewrite. During this time, because nothing ships, nothing can go wrong and, hopefully, the stock price will have grown to the point the manager can cash in his options in time to go be a coward somewhere else.

Meanwhile, the architects seem invincible to failure and rise within the ranks of their organizations, ordering fresh business cards each year with the words "architect," "senior" or, for the power blowhards, "distinguished" in the title. They are drawn to the tar pit of attending and creating presentations, or joining conference calls with fellow architects who showboat their knowledge of obscure standards specifications or bleeding-edge research projects. They'll have copies of Christopher Alexander books in their office and spend hours googling for obtuse and arcane quotations to lace their presentations with and gain kudos from fellow fools.

When confronted by such people, recant the following mantra:

Code ships,
code runs,
code helps users,
get their job done.

Remind any architects in your path that presentation charts, e-mails, project plans, line-items spreadsheets and so forth, are all there to help the code ship on time and to spec. The goal of everyone on a project should be to spend as little time as possible on tasks that distract from the job of creating quality, tested, and shippable code. Please architects, please understand this, respect this, and quietly stay out of the way of those good folk who prefer to spend their day working with an IDE writing code rather than composing e-mails.

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

joewinchester@sys-con.com

> *Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises*
>
> — Yafim Natis, Gartner Analyst

2-DAY EVENT!

SOAWorld 2007

Plus

Enterprise OpenSource Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services

- AJAX and SOA
- Web 2.0
- Universal SOA
- Protecting Web Services
- Troubleshooting SOA
- Governance
- Open-Source SOA
- XBRL
- Service Virtualization

Open Source

- Open Source Business Models
- Open Source ESB
- OpenAjax Alliance
- SaaS and Open Source
- Spring, Hibernate and Eclipse
- Seam
- Open Source Penetration
- Monetizing Open Source
- Open Source Databases
- AMQP
- Open Source Middleware

June 25–26, 2007
Roosevelt Hotel / New York City

Register Online! www.SOAWorld2007.com

SOAEOSCONFERENCE.SYS-CON.COM
REGISTER ONLINE TODAY
SAVE $200!
(HURRY FOR EARLY-BIRD DISCOUNT)

11th International SOAWORLD 2007 CONFERENCE & EXPO

2nd Annual ENTERPRISE OPENSOURCE 2007 CONFERENCE+EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-26, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

BROUGHT TO YOU BY:

SOA Web Services Journal
focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.

Enterprise Open Source Magazine
EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Exhibit and Sponsorship Info:
Call 201-802-3020 or email events@sys-con.com

Enterprise Mashup Services

Part 2: Combining JSF and mashup services to make mashup components

by Ric Smith

In my previous article, "Enterprise Mashup Services: Real-World SOA or Web 2.0 Novelties?" (*JDJ* Vol. 11, Issue 12), I discussed how a Java-to-AJAX library such as Direct Web Remoting (DWR) can bridge the gap between mashup services implemented with JavaScript and business services written in Java, allowing developers to blend corporate services with external services such as Google Maps. The problem with this approach is that it relies on AJAX as an integration point, which entails a fragile development platform as well as the need to maintain browser-specific code due to idiosyncrasies in browser support for JavaScript — the primary technology behind AJAX. In addition, JavaScript lacks a standardized approach for componentizing code, making applications written in it difficult to consolidate and reuse. The solution to these shortcomings is to pair AJAX with a component framework. JavaServer Faces (JSF) provides this foundation and eliminates the complexities of JavaScript — besides providing rich integration with the Java EE platform.

A mashup component is a custom JSF component that encapsulates the code that operates on a mashup API. Once created, the mashup component eliminates the need to work with JavaScript. Thus the code is both simplified and easily reused, making the API accessible to both JavaScript experts and less-experienced developers. The intent of this article is to build on the concepts introduced in the previous article and present the tools to create enterprise-ready components that encapsulate mashup services.

Ric Smith is a principal product manager for Oracle's Java/JEE/SOA tool offering, Oracle JDeveloper. Prior to joining the Oracle JDeveloper team, Ric worked for Oracle's consulting business as a principal consultant, where he specialized in Java EE architecture and development. Before his work in consulting, Ric was a lead software developer at Lockheed Martin. Ric holds a Bachelor's of Science in Computer Science with honors from the University of Arizona.

Rich Components Versus Mashup Components

On the surface, JSF components that abstract mashup APIs appear no different than those that encapsulate AJAX functionality. In fact, the concepts that define both components are the same; it's the philosophy behind the component definitions that differs. AJAX-enriched components abstract the complexities of JavaScript and provide interactive visual effects. JSF components that encapsulate mashup services are created with the same intent; however, mashup components package both visual effects and interactions with services. So mashup components represent Service Oriented Architectures (SOAs) at a micro level within a larger composite application. The example used in this article is a component that blends Google Maps with the Yahoo! Geocoding API. The crux of this solution is the tying of JavaScript events to event handlers implemented with Java code. This marriage lets mashup services work in conjunction with those implemented on the Java EE platform, which in the example are isolated to services provided by the JSF-managed bean facility.

Shale Remoting

The Shale Framework, which can be found on the Apache Web site (http://shale.apache.org/), provides a rich Web development framework that extends JSF. Instead of going into the numerous features that Shale provides, I'll focus on a single aspect, Shale Remoting, which maps a server-side resource such as a static JavaScript file or a method associated with a managed bean to a URL. For example, Shale maps the URL faces/remote/hellobean/welcomeUser to helloBean.welcomeUser(), which calls the welcomeUser method associated with the helloBean managed bean. The URL faces/static/com/thepeninsulasedge/scripts/maps.js identifies a script file located in a Java archive (JAR) under the package structure /com/thepeninsulasedge/scripts/.

In short, Shale Remoting provides a simple mechanism to implement AJAX functionality in custom JSF components that would otherwise require the implementation of a custom PhaseListener or ViewHandler to handle XMLHttpRequests and serve static resources. In this article, Shale is used to map JavaScript event listeners to methods defined in managed beans. More specifically, it's used to tie Glisteners — JavaScript functions that respond to events triggered by Google Map's GMap2 object — to methods implemented in Java. Note that DWR could be used to accomplish this task; however, Shale provides tighter integration with the JSF managed bean facility as well as the ability to serve resource files from archives.

Using Shale Remoting

In the example that follows we construct a simple managed bean (HelloBean) that defines a method (welcomeUser) that's invoked by a JavaScript function in Figure 1.

The function is called when a user enters his or her name into a text field, and a response for each event is displayed in a div below the text field. Listing 1 shows the complete JSP page. The HelloBean managed bean defines three methods: welcomeUser, getParam, and writeResponse.

The latter two methods are simply utility functions. The getParam method extracts a value from the request string via the RequestParameterMap for a given parameter. The writeResponse method writes a response to the FacesContext, which is rendered to the client and processed by an XMLHttpRequest handler (more on this later).

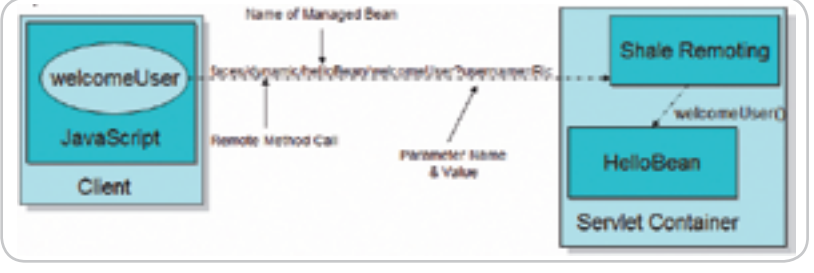


Figure 1 Shale Arch

The welcomeUser method is invoked with the URL faces/dynamic/hellobean/welcomeUser. This method extracts the value associated with the username parameter from the request, manipulates the extracted value, and then writes a response to the client using the writeResponse method. A parameter can be passed to the welcomeUser method by using the URL faces/dynamic/hellobean/welcomeUser?username=Ric, which calls the method and provides a name/value pair as a parameter. Accessing the URL via a browser provides a simple way to test this, and should display a response similar to that shown in Figure 2.

Once mapped to a URL, the welcomeUser method is easily tied to an event listener that uses the JavaScript XMLHttpRequest object to post to the URL mapped to the method innovation. The code to perform this operation is in Listing 2 and is contained in a file, scripts.js. (Listings 2–11 can be downloaded from the online version of this article at http://java.sys-con.com).

For simplicity's sake, the Prototype framework (http://prototype.conio.net/) is used to manipulate XMLHttpRequest objects — the AJAX.Request function handles each request. The function requires a URL, which is used to call the welcomeUser method defined in the HelloBean managed bean. The AJAX.Request function also accepts a JavaScript object as an argument, which defines the request method (for example, POST or GET) as well as the parameters to append to the request string. The final argument passed to the AJAX.Request function identifies a handler used to process the response initiated asynchronously by the request. In the example, the response is handled by the displayResponse function, which writes the text of the response to a div identified by the ID response. The $(…) notation is shorthand for the document.getElementById() JavaScript function and is a feature of the Prototype framework.

The welcomeUser function is fired on the onkeyup event of the text field. With each keystroke the user enters in the text field, the welcomeUser function is called, which invokes the method defined in the managed bean. In essence, the managed bean contains the logic to respond to each JavaScript onkeyup event.

JavaServer Faces Components

Now that we have identified a mechanism to link Java code to JavaScript, let's quickly review the key facets of a JSF component before putting the pieces together to build our first mashup component.

There are essentially three elements to a JSF component: behavior, presentation, and tag definition. (Figure 3 shows the class definitions for each element of the component used in this article.)

Behavior

Component classes characterize behavior and extend the javax.faces.component.UIComponentBase class or one of its subclasses (Listing 2). Each component class is also defined in the faces-config.xml file shown as follows:

```
<component>
<component-type>
com.thepeninsulasedge.components.MapPanel
</component-type>
<component-class>
com.thepeninsulasedge.components.UIMap
</component-class>
</component>
```

Presentation

A component's presentation is delegated to a separate class that extends the javax.faces.render.Renderer class (Listing 3). A renderer produces a graphical representation that need not be implemented with HTML. The presentation could be represented by XUL, ASK, Telnet, or any number of protocols. For our purposes the renderer is used to generate HTML and JavaScript. The JavaScript produced by the renderer is used to post to URLs defined by the Shale Framework as well as consume the Google Maps API. Note that this compromises the clean separation between presentation and behavior implemented by the JSF component architecture because behavior is now also defined in the renderer. Unfortunately this is a necessary evil when mixing JSF with AJAX. Each renderer class must also be defined in the faces-config.xml file shown as follows:

```
<render-kit>
 <renderer>
  <component-family>
com.thepeninsulasedge.components.MapPanel
  </component-family>
  <renderer-type>com.thepeninsulasedge.components.Map</renderer-type>
  <renderer-class>
com.thepeninsulasedge.components.MapRenderer
  </renderer-class>
 </renderer>
</render-kit>
```

Tag Definition

The JSP tag representing the component is defined by a subclass of the javax.faces.webapp.UIComponentTag class (Listing 4) and a tag library descriptor (TLD) — an XML file that provides metadata for the tag (Listing 5). The TLD should be registered in the web.xml file shown as follows:

```
    <jsp-config>
        <taglib>
            <taglib-uri>http://thepeninsulasedge.com/jsf</taglib-uri>
            <taglib-location>/WEB-INF/pc.tld</taglib-location>
        </taglib>
    </jsp-config>
```

Developing Mashup Components with JSF and Shale

As previously mentioned, the intent of

Figure 2 Hello World with Shale

Figure 3 Component Classes

Figure 4 Rendered Map Component

this article is to build a JSF component that encapsulates a mashup service and provides the ability to link JavaScript events that represent interactions with the encapsulated service to methods associated with a managed bean. Now that you have a basic understanding of the core technologies involved, let's take a look at an example of a mashup component.

```
<tpe:map id="gmap"
   initLat="#{mapbean.initLat}"
   initLng="#{mapbean.initLng}"
   zoomLevel="#{mapbean.initZoom}"
   inlineStyle="width:500px;height:500px;"
            key="#{mapbean.key}"
            model="#{mapbean}"/>
```

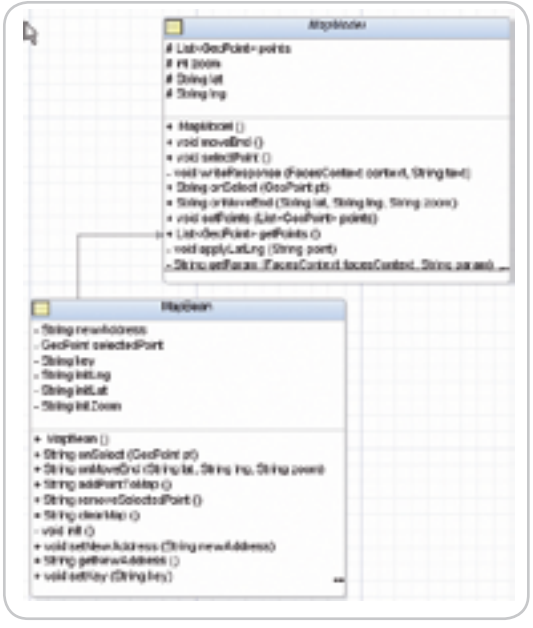


Figure 5 MapModel Diagram

Figure 6 Info Window

Figure 7 Coordinates and Zoom

The tpe:map component represents a wrapper for the Google Maps API, and it generates the HTML and JavaScript shown in Listing 6. The rendered component (shown in Figure 4) provides a generic interface to the API and can easily be changed to represent another mapping API such as Yahoo! Maps. To do so, simply replace the renderer class (Listing 3) provided in the example with one that generates the HTML and JavaScript required to display a map from Yahoo! Maps. User interactions with the Google Maps API, such as zooming, panning, and clicking, are trapped by JavaScript events, which in turn request a URL that is mapped to a method associated with a managed bean.

The map component has two required attributes: key and model. The key attribute is the account identifier for the Google Maps service. You can request a key at http://www.google.com/apis/maps/signup.html. The model attribute requires a subclass of the com.thepeninsulasedge.components.model.MapModel abstract object (Listing 7). A concrete implementation of this object represents the model for the map component. The object contains geospatial points in the form of com.thepeninsulasedge.components.model.GeoPoint objects (Listing 8) that represent visual markers on the map. The object also contains methods or event handlers to respond to events fired by the map.

The MapModel and MapBean

The MapModel object (Listing 7) defines two abstract methods: onSelect and onMoveEnd. The onSelect method is called when a user selects a marker on the map, and the onMoveEnd method is invoked after a user completes a zoom or pan operation.

```
abstract public String onSelect(GeoPoint pt);

abstract public String onMoveEnd(String lat,
String lng, String zoom);
```

Both methods represent steps in a simple template or behavior pattern in which behavior is implemented by a subclass of the MapModel object (Figure 5).

The onSelect and onMoveEnd methods are called by the selectPoint and moveEnd template methods, respectively. Both methods extract values for parameters from a request string, pass those values onto the methods that inject behavior (for example, onMove or onSelect) into the encapsulated method, and return a response using the writeResponse method.

The moveEnd and selectPoint methods are invoked by the faces/dynamic/mapbean/moveEnd and faces/dynamic/mapbean/selectPoint URLs, respectively. The addSelectPointListener and addMoveEndListener JavaScript functions defined in the mapScript.js file (Listing 9) associate a GEvent listener with the current GMap object. When called, a listener posts a request to the URL that invokes either the moveEnd or selectPoint method. In this example, Gevent listeners are generated on marker selections or pan and zoom operations. For more information on Gevent listeners or other facets of the Google Maps API, refer to the Google Maps API documentation (http://www.google.com/apis/maps).

The concrete implementation of the MapModel object used in this example is com.thepeninsulasedge.view.managed.MapBean (Listing 10). The class defines an onSelect method that extracts a message or string contained in a GeoPoint object, which is associated with a marker on the map. The extracted message is then written as a response to the initial XMLHttpRequest object using the writeResponse method, and displayed in an info window on the map (Figure 6).

The onMoveEnd method updates a display that shows the coordinates of the map's center and the current level of magnification (Figure 7).

Besides providing methods to respond to JavaScript events, the MapBean also defines three action methods — addPointToMap, removeSelectedPoint, and clearMap — that are executed by command components. These methods demonstrate how the tpe:map component can be integrated with existing JSF components. For example, the addPointToMap method adds a new point to the map by manipulating the collection of GeoPoint objects contained in the MapBean. The method creates a new GeoPoint instance from an address and adds the point to the MapBean's current list of points. The method is executed by h:commandButton and the address is provided by an h:inputText field (See Listing 11).

GeoPoints

Each GeoPoint object in the MapBean is associated with a marker on a map and located by latitude and longitude. (To better understand the relationship between GeoPoint objects and the MapBean, see the diagram in Figure 8.)

REGISTER TODAY AND SAVE!

Rich Internet Applications: AJAX, Flash, Web 2.0 and Beyond...

www.AjaxWorldExpo.com

AJAXWORLD™ EAST
CONFERENCE & EXPO

NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

SYS-CON Events is proud to announce the AjaxWorld East Conference 2007!

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

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

BEING HELD MARCH 19 - 21, 2007!

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

Figure 8 Map Component Model

Thus, when a user clicks a marker, the corresponding GeoPoint object is located in the MapBean by finding a GeoPoint object with a matching set of coordinates. The following Predicate (see the Apache Commons Collection at http://jakarta.apache.org/commons/collections/) is used to perform the evaluation between latitude and longitude and a corresponding GeoPoint object. The Predicate performs this evaluation while iterating through a collection of GeoPoint objects.

```
public static Predicate
findPredicate(final String lat,
final String lng) {
    return new Predicate() {
                public boolean
evaluate(Object obj) {
          if (!(obj instanceof GeoPoint))
                        return false;
          GeoPoint pt = (GeoPoint)obj;
          if (pt.getLat().equals(lat) &&
pt.getLng().equals(lng)) {
                return true;
              } else {
                return false;
            }
          }
        };
    }
```

Listing 1

```
    public void welcomeUser(){
        FacesContext facesContext =
FacesContext.getCurrentInstance();
        String username = getParam(facesContext, "user-
name");
        writeResponse(facesContext, "Welcome " + username
+
" and hello world!");
    }

    private static String getParam( FacesContext facesCon-
text,
String param ){
        String value = (String)facesContext
            .getExternalContext()
            .getRequestParameterMap().get(param);
        return value;
    }

    private static void writeResponse(FacesContext con-
text,
  String text) {
        if(context == null || text == null) return;
        ResponseWriter writer =
  (new ResponseFactory()).getResponseWriter(context,
"text/plain");
        try {
            writer.startDocument();
            writer.writeText(text, null);
            writer.endDocument();
            writer.close();
            context.responseComplete();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

The GeoPointUtil class also provides a convenient way to create new instances of GeoPoint objects using the Yahoo! Geocoding API. The technique simply parses coordinates from an XML result generated by a request and uses the coordinates to create a new instance. The result is parsed with the Apache Commons Digester (http://jakarta.apache.org/commons/digester/).

The MapRenderer

The HTML and JavaScript in Listing 6 is the code used to create a new instance of the GMap2 object — the JavaScript object that represents a map in the Google Maps API. The code also ties the JavaScript used to consume the Google Maps API to the Java code in the MapBean. The com.thepeninsulasedge.components.MapRenderer class (Listing 3) is responsible for producing the markup shown in Listing 6. To understand which method in the MapRenderer class produced a specific snippet of JavaScript or HTML, look at the comments generated by the MapRenderer class in Listing 6. Note that only the load function used to instantiate the GMap2 object and a set of variables are dynamically generated in the MapRenderer class. This is done to ensure that each tpe:map component has a unique load function and set variables, allowing multiple tpe:map components to be used in a single page. Uniqueness is guaranteed by appending the component's client identifier to the name associated with the dynamically generated variable or function. According to JSR 127, each server-side component in JSF is guaranteed a unique client identifier.

The MapRenderer class relies on functions defined in the mapScripts.js file (Listing 9). These functions limit the need to hardcode JavaScript into the class and define essential functions such as addSelectPointListener and addMoveEndEvent. For the JavaScript generated by the MapRenenderer class to use functions defined in mapScripts.js, the file must first be imported. This is done with Shale Remoting. The linkJavascript method in org.apache.shale.remoting.XhtmlHelper is used for the import statement for the mapScripts.js file:

```
<script type="text/javascript" src="/
GoogleMapsAndDWR-JSFView-context-root/faces/
static/com/thepeninsulasedge/components/
scripts/mapScripts.js"></script>
```

The script import provides a reference to the mapScript.js file relative to the Web applications class path. Thus, the file specified is located under the package structure com/thepeninsulasedge/components/scripts/. A similar import is generated for the prototype.js file.

Conclusion

Consuming mashup services in the enterprise is a reality, and encapsulating mashup APIs with custom JSF components provides an elegant solution for packaging and reusing these services. The example in this article consolidated the complex markup and script shown in Listing 6 into a simple and concise component that amounted to a single XML tag. Imagine that — the power of Google Maps packaged in a simple component. Moreover, the componentization of the Google Maps API, as well as the linkage between JavaScript events and the JSF managed bean facility, makes it relatively easy to tie Google Maps to J2EE services such as EJB 3.0. The result of this blending of external mashups, Shale, and JSF is an enterprise-ready mashup service in very accessible package. To learn more about the technologies referenced in this article, please refer to the references provided.

References

- Jonas Jacobi and John R. Fallows. *Pro JSF and AJAX: Building Rich Internet Components.*
- Chris Schalk and Ed Burns. *JavaServer Faces: The Complete Reference.*
- The Google Maps API documentation: http://www.google.com/apis/maps/documentation/
- The Shale Framework documentation: http://shale.apache.org/documentation.html

JavaOne℠

Sun's 2007 Worldwide Java Developer Conference℠

May 8-11, 2007

The Moscone Center
San Francisco, CA

JavaOne™ Pavilion: May 8-10, 2007

IT'S AN EXCITING TIME FOR THE ENTIRE JAVA™ TECHNOLOGY COMMUNITY.

With the evolution of the Java platform, new opportunities are emerging for developers, thought leaders, and entrepreneurs. That's why for 2007 the Conference is featuring an expanded program that embraces technologies outside the core Java platform while keeping Java™ technology at the center. You'll get the same in-depth content we have always focused on, plus more topics and issues relevant to today's evolving market.

LEARN MORE ABOUT:*

- > Java Technology
- > Scripting
- > Open Source and Community Development
- > Integration and Services-Oriented Development
- > Web 2.0
- > Compatibility and Interoperability
- > Business Management
- > And More

Java™

PLATINUM COSPONSORS

intel ORACLE

GOLD COSPONSOR

NAVTEQ

SILVER COSPONSORS

InterSystems PARASOFT We make software work.

SAVE $200**

REGISTER BY APRIL 4, 2007

Please use priority code: J7PA1JDJ

*Content subject to change.
**Offer not available on site.

Register Today @ java.sun.com/javaone

Sun microsystems

SWT: Ship Happens

Insights from the SWT community

Interview by Joe Winchester

The Standard Widget Toolkit (SWT) is the GUI toolkit used by Eclipse. The same folks that worked on the Common Widget (CW) library for IBM/Smalltalk developed it, this time for Java. Now, it's maintained as part of the Eclipse Platform project and distributed under an open source license, the Eclipse Public License (EPL). One key design point of SWT is that it uses native functionality on each operating system and, at the same time, presents a common, portable API. Joe Winchester, Desktop Java Editor for *Java Developer's Journal*, asked Steve Northover (SWT Team Lead) recently whether he'd be happy to answer some questions about SWT and, after talking to his colleagues and a few developers, here is the result.

JDJ: SWT supports many different widget toolkits with a common programming API. What's the hardest thing about making all this work?

Steve: Specifying an API that can be implemented natively on a variety of different platforms is very challenging. If you make the wrong choice, you end up with API that is difficult or impossible to implement. To avoid this problem, you need to approach API design with an open mind. Smart programmers want to get the job done and don't care too much about how they do it. Our goal is to get out of the way and get functionality to the programmers. We've been pretty successful doing this and keeping the API reasonable at the same time.

For the implementation, complexity becomes a big issue. It's easy to die the death of 1000 cuts, implementing a native widget toolkit. A sure way to do this is to over-engineer things. We often use the "just do the work" pattern (a favorite of mine). Given two solutions to a problem, I will always choose the one with the fewest classes and lines of code.

Carolyn: The hardest thing is saying "no" to some of the features. Implementation-wise, though, the "devil is in the details."

Silenio: Sometimes it's hard to keep the behavior of the widgets consistent between platforms and still have platform-specific features. For example, on the Macintosh the menu bar is detached from the shell, which is quite different from the other platforms. We needed to come up with a consistent way of integrating this Macintosh feature into the API.

JDJ: When you're dealing with a feature that must be built to work across the different implementations, you have a choice between doing the lowest common denominator, or something you do natively on some platforms and emulated on others. When do you decide which to use, and do you regret any of these decisions?

Steve: There is no decision. If the operating system offers a feature, we make use of it. There's nothing to regret either. We just go ahead and implement whatever is necessary and move on to the next problem.

I'd like to talk about "lowest common denominator" for a minute though. Lowest common denominator is kind of a negative statement. A more positive way to think about it is "highest common multiple." The operating systems provide a lot of functionality that's common, but the native API is different. We expose this functionality, raising the bar rather than lowering it.

JDJ: Do you wish you'd used a different API as your base rather than the Windows one, in particular the way in which window parents can't be changed after construction? For example, if this feature became supported in Windows in the future, it might seem a bit of SWT legacy, whereas if you'd coded reparenting in the C code that SWT sits on top of, you could provide a higher level of API. AWT, for example, allows reparenting by having a wrapper around the peer that can recycle the underlying widget.

Steve: First, I'd like to challenge the notion that SWT is based on the Windows API. It isn't. If you go to MSDN, find the documentation for something like TreeView32 or HDC and check out the SWT API that makes use of these things – you'll see it doesn't look anything like the Windows API. People might get this idea from things like style bits that are found on Windows, but many things in SWT are patterned after other operating systems. For example, the keyboard and mouse API is based on X. We are familiar with many different windowing systems and make API decisions keeping all of them in mind.

Back to reparenting: it's X/Motif that doesn't allow the parent to be changed after a widget is created, not Windows. Whether you are coding in Java, C, or both, either the operating system supports reparenting or it doesn't. If the operating system doesn't support this feature, hiding it in a peer layer doesn't really help that much

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

joewinchester@sys-con.com

(actually, it makes it worse because it increases the complexity of the toolkit implementation and adds a level of indirection).

Carolyn: By not having peers, we simplified the toolkit tremendously. Also, peers are somewhat slower and take up more space. And setParent() [mostly] works.

JDJ: What's your favorite native platform to work with, and which do you most loathe having to code on?
Silenio: My favorite platform to work with is the one I have been working on most at the moment. That's because it's the platform I understand best and I can achieve faster results.

Carolyn: Windows has the best doc, so that makes it "friendlier" to work with. We use Google for GTK doc. I don't loathe coding on any of the platforms – the variety is what makes it interesting.

Kevin: Every platform presents unique challenges to SWT so it's difficult to single out any as being better or worse than the others. My favorite platform really depends on the problem that I'm trying to solve.

JDJ: At JavaOne you were walking around with the letters 67384 tatoo'd on your arm. What's the story behind this particular bug?
Florian: SWT contains a piece of code called the SWT_AWT bridge that lets you embed an AWT/Swing component in an SWT shell and vice versa. Prior to 3.2, this didn't work on the Mac due to architectural difficulties. Basically, SWT uses the Carbon API while AWT is implemented in terms of Cocoa, with implications on how UI events are handled. Making the event threads of the two widget toolkits cooperate smoothly and avoiding deadlocks proved to be a rather tricky issue that required changes to both SWT and Apple's Java implementation. The discussion about all of this took place in the aforementioned bug report and spanned a ton of comments from various users. While some people vented their frustration or put forth conspiracy theories, others actually presented ideas on how to fix the problem. In the end, Scott Kovatch, an excellent engineer at Apple, worked out and implemented the necessary steps in cooperation with the SWT team, allowing us to finally mark that bug as "RESOLVED FIXED". Obviously everyone is very happy about that. One user on the bug report even went as far as articulating his excitement over the fix in a rather unique way. For more details, see Bugzilla.

Steve: Although Eclipse and most other SWT applications didn't use AWT/Swing, there were some applications that did and this was holding them back. I hate that because we take pride in helping people ship, not telling them which technology they should use. SWT normally integrates really well with native code but the Mac supports only one GUI thread and both toolkits expected to have their own. That was the technical issue. Somehow, Scott and Silenio got it to work.

Carolyn: "SWT_AWT not implemented for Mac" https://bugs.eclipse.org/bugs/show_bug.cgi?id=67384#c170. It was very exciting to have this one fixed… <g>

Contributors

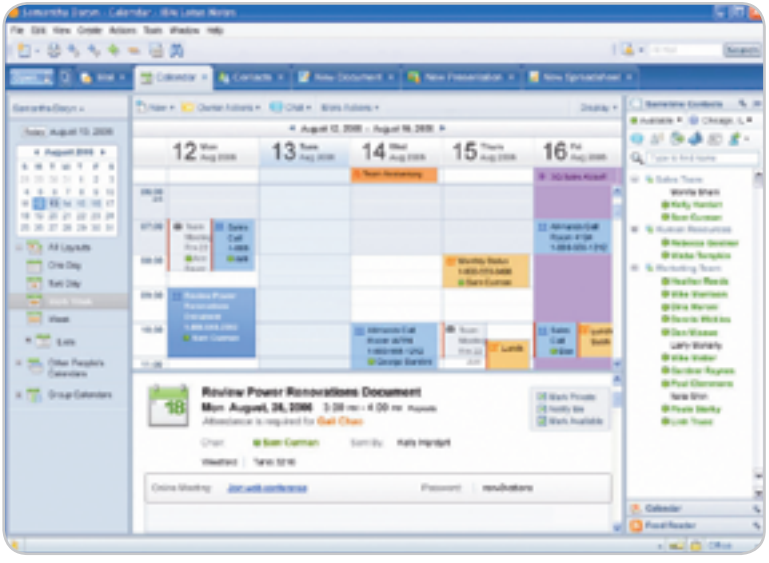
Benjamin Pasero – *RSS Owl*, an RSS Reader
Carolyn MacLeod – *IBM*, committer
John Kellerman – *IBM*, Product Manager, Eclipse
Kevin Barnes – *IBM*, contributor
Felipe Heidrich – *IBM*, committer
Florian Priester – contributor
Grant Gayed – *IBM*, committer
Matthew Hatem – *Lotus*, Lotus Notes, Advisory Software Engineer
Olivier Chalouhi – *Azureus*, a BitTorrent client
Silenio Quarti – *IBM*, SWT Technical Lead, committer
Steve Northover – *IBM*, SWT Team Lead, committer

JDJ: What's the story behind the naming of the classes CoolBar and CoolItem?
Steve: They're cool. Personally, I hate them.

Carolyn: That's what Microsoft called them. (I know, their control is called a "Rebar," but they used the term CoolBar when describing the control. See this article: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwui/html/msdn_rebar.asp.)

Silenio: Usually when we add a new widget to the toolkit, we decide its name by taking into consideration the names used by all platforms as well as our own ideas. We choose the one that best describes the widget and that's most known by everyone. In this particular case, there were two main options: CoolBar (MFC) and ReBar (Win32). I must admit that CoolBar is strange, but it's a bit better than the other option.

JDJ: When SWT started did you think about using the AWT model, which is essentially a native toolkit but with a wrapper delegating to the peer? If SWT had extended AWT, then widget interoperability, not to mention karma in the Java community, might have been better.
Steve: For starters AWT is free-threaded. That can't be changed because it's built into the toolkit. Also, there are a



Calendar

number of other technical issues with AWT that I won't discuss here that made this option unattractive.

I think that most of the strong feeling comes from people who are passionate about their technology and get carried away. For our part, we had different design constraints that led us to a different solution. Being native was a requirement. At the time, although we had quite a good reputation in the Smalltalk community, we were largely unknown in the Java world. Extending or rewriting AWT was not an option; we had built portable native widget toolkits in the past and there was no time to argue over philosophy. Nobody had any idea that Eclipse would take off like it did and that so many people would want to use SWT outside of it. That part is amazing but it goes to show that there is a real demand for native technologies.

JDJ: *SWT has no pre-req on AWT at all, even for seemingly trivial classes like Rectangle. This looks like a conscious decision to have SWT be able to run without AWT having to be present. Are there places SWT can run where the AWT packages aren't available?*

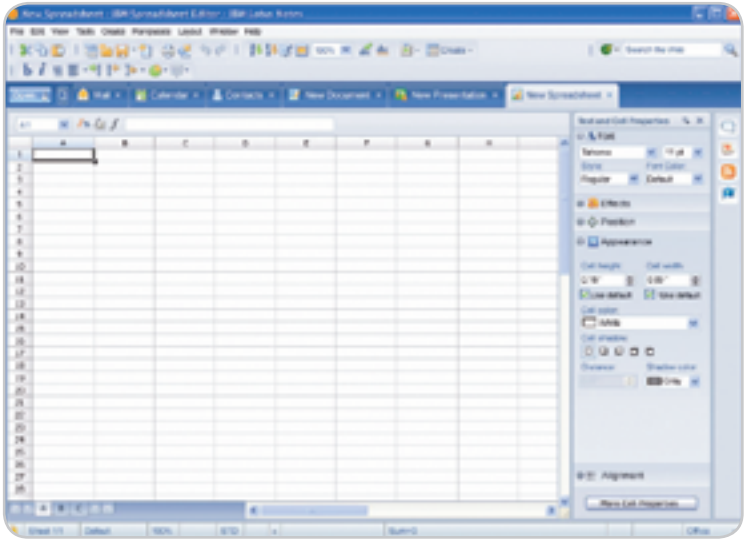
Steve: SWT runs on QNX Photon where AWT does not exist.

JDJ: *Why isn't there an org.eclipse.swt.graphics.Dimension class? It always seems odd when getSize() and getLocation() both return a Point, rather than a Dimension and a Point.*

Silenio: That was a design decision made to keep the number of classes down.

Steve: The only way I know for sure to make something small is to start out that way. Libraries always grow and programmers have a habit of referencing every line of code you've ever written. Attempts to strip out classes later are painful and never really work that well. In SWT, what you see is what you get. For example, by design, there are very few inner classes in the implementation of the toolkit (almost none in the widgets or graphics package).

Carolyn: Why fill up the toolkit with classes that don't pull their weight?



ODF Editor

JDJ: *Constructor style bits have always puzzled me. I used to tell people that they were everything that couldn't be changed after a widget had been constructed, and that this was everything that affected the size of the trim. Therefore things like BORDER, V_SCROLL, and so forth that change the trim/client area ratio are style bits. However, there seem to be places where style bits are used for things that also have a perfectly good getter and setter and can clearly be changed post-construction. What's the story behind style bits and why and when they are and aren't used on an API?*

Steve: There are very few features that are style bits that can be changed after the constructor. These were added later when we discovered that some properties that we had originally thought were create-only could be changed later.

Carolyn: Yes, style bits are for "things that can't be changed after construction." Most of them are things you wouldn't ever want to change after construction. But some – like read-only/editable – occasionally needed to be changed later, so setters were added.

Silenio: Style bits are also used to reduce the number of classes. For example, rather than having a Separator class, Label can display static text or be a line separator, depending on the style bit.

JDJ: *Why isn't there a native rich text widget in SWT? The StyledText control used by the Java editor in Eclipse is emulated, so it's just a big canvas with a lot of paints on it. This seems to go against the grain of the SWT philosophy of being a thin layer on top of the platform.*

Carolyn: We originally used a native RICHEDIT on Windows (i.e., the same control that MS WordPad uses). It didn't fit the requirements for the Eclipse text editor. It makes sense to use a custom text control here for an IDE.

Felipe: Initially we had to go with an emulated rich text widget because Motif didn't have a native option available. Another reason is that rich text widgets can have too many features, making it very hard to define a common API that can be implemented in all platforms. The door is still open for providing a native rich text editor in the future.

JDJ: *Several of the last Eclipse releases have seen more things being introduced that seem to be more emulated than native: the UI forms toolkit, user painting in Table and Tree Items, and the rounded gradient tab item titlebar used by Eclipse. Is there a danger of SWT becoming Swingish, where instead of native functionality an SWT program is doing all of its work building the UI in the middle of a paint event?*

Steve: People need to ship applications and whatever they decide is fine by me. For example, the UI designers for Eclipse decided they wanted a certain look for Eclipse so they built it. It's not a question of philosophy or Swingishness.

Felipe: Probably not; the customer would really have to overuse the custom draw capabilities of SWT to cause the application to look non-native. It's important that SWT offers these features to allow branding and custom UIs.

Grant: There are contexts where doing some painting is useful and is not necessarily a non-native practice. For instance, custom drawing of table items can be used to draw an item's image to the right of its text, and any programmer using the native Windows table would have to do this anyway. However, the underlying control is still native and maintains its native behavior.

In general, SWT will not provide non-native implementations for natively available widgets. Exceptions to this, such as the CTabFolder, were created for Eclipse branding purposes, but the programmer has a choice. So there is not necessarily a trend toward SWT doing increased painting. Certain widgets have been painted all along.

JDJ: *What do you think of things like Nebula and other efforts that seem to be putting more emulated controls into SWT?*
Silenio: They are doing great work and making the toolkit richer.

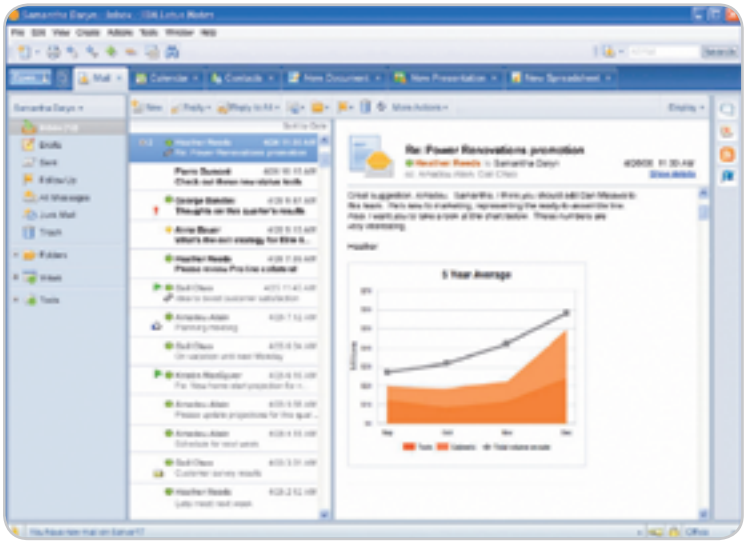
Grant: Nebula is interesting to watch because its mandate is to create controls that do not overlap with existing ones in SWT. It's not surprising that these widgets are emulated since they often implement functionalities that are not available natively. I think it's great that users have a common place to share controls like these that they find useful. This is good for Eclipse and SWT.

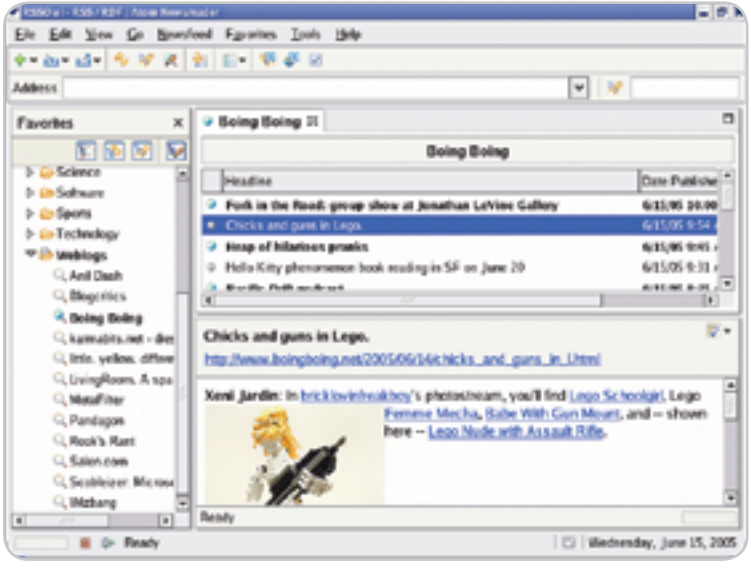
JDJ: *Swing used to get beat up quite a bit about not looking and behaving like a native widget toolkit. In Java 6.0 they are using platform API calls to determine the correct way to paint their widgets, so presumably the difference to the user between a Swing app and an SWT one will be indistinguishable. If this had been present at the point when SWT was being discussed, do you think history would have taken a different path?*
Steve: Possibly, but Swing wasn't up to the task when we needed it. We were shifting gears from Smalltalk to Java and Swing didn't meet our needs. We couldn't have waited until now for widgets that we could use, otherwise, no Eclipse. Also, with all due respect to Java 6.0 and Swing, there is more to being native than drawing that way, assuming of course that you can get it right for every widget. Eclipse and other SWT applications are tightly integrated with the desktop and get all sorts of benefits from this type of integration.

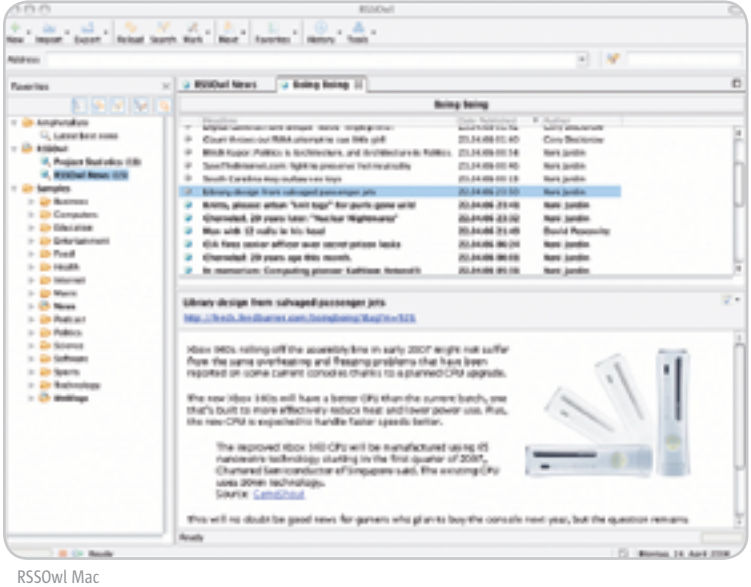
Carolyn: It's not just the painting; it's the "feel" of a control, too. We are constantly getting stuff from the operating system "for free," like native drag-and-drop support, native accessibility, etc.

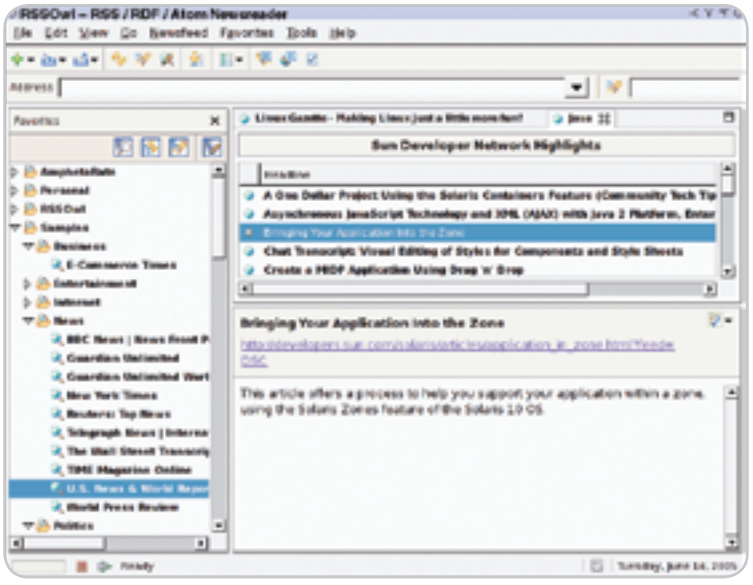
JDJ: *Following on from the previous question, if Swing hadn't had problems with size, platform fidelity, and reliability, do you think it would have been adopted by IBM as the widget toolkit for Eclipse?*
Steve: That's a funny question. If Swing had met our design goals and requirements, could we have used it? Yes, but at the time, it didn't. If you are really asking whether there is value in native widget toolkits, there is. Things just feel and work right. Desktop settings, painting and drawing, key bindings, animations, fonts, input method editors, third-party tools, accessibility, the list goes on. With SWT, Java is a first class citizen on the desktop.

JDJ: *Apart from the previous two, what's the single question you get most tired of being asked about SWT?*
Steve: Nothing really stands out. Explaining design decisions like style bits, constructors, and threading can become tiresome. But really, people can't know the answers to these questions. If they have never done any operating system programming and have a

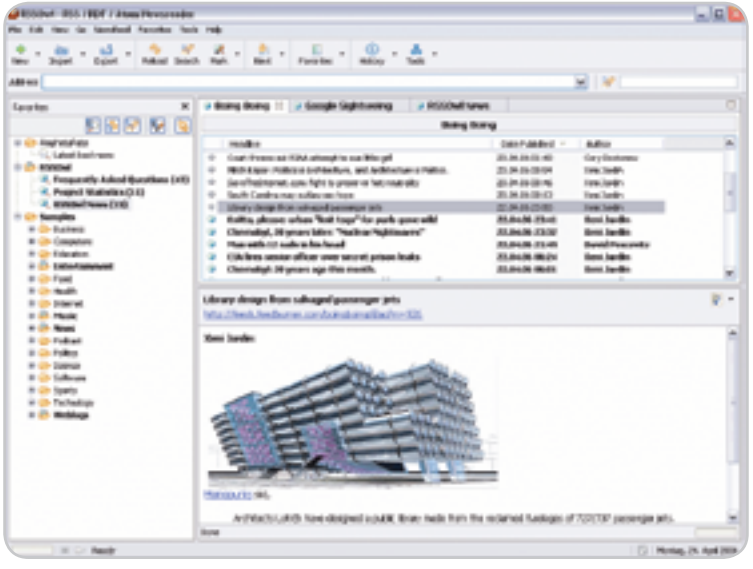

Mail


RSSOwl Linux


RSSOwl Mac



RSSOwl Solaris



RSSOwl Win

Java-centric view of the world, then asking these sorts of questions is quite natural.

How about, "Why did you do SWT? Are you trying to fracture the community?" The answer is "Anything but."

Silenio: Why doesn't SWT use finalization? I'm sick of that one.

JDJ: There was a JavaLobby story recently about some folks who'd managed to get the SWT API to work with Swing classes, and even had screenshots of Eclipse running under Swing. Does this kind of effort a) amaze, b) frighten, or c) bore you?
Steve: None of the above. I think this is really cool. One of the challenges of SWT is developing an API that can run on all sorts of different platforms. If you consider Swing as just another platform, these guys ported to it. I think our position with respect to other technologies is pretty consistent. I don't expect that to change. Interestingly, this port reaffirms many of the API design decisions we've made.

Carolyn: a)

JDJ: The Linux folks seem to complain about print support for SWT. Is this an issue and something being worked on?
Steve: Fixed > 20060717.

Carolyn: Printing support was added to GTK+ with the release of GTK+ 2.10 in July. We added GTK printing to SWT practically the next day, and it went into Eclipse 3.3M1. The main point is that we were waiting on this from GTK.

JDJ: What's the most exciting thing going on in SWT at the moment, both within the committers and development team, and also the larger community with its usage of SWT?
Steve: For me, it's Vista.

Silenio: WPF port, animation API, theme drawing API, Windows port for 64-bit …

Kevin: The SWT community is fun to work with because they're a very dedicated bunch who really want to contribute. There are an amazing number of bug reports filed and the reporters are always willing to work with the team to provide more information, testing, and the feedback that we need to make SWT better. It takes a significant commitment to become a committer. For example, I have been an Eclipse committer on another project for over three years but despite that, I still need to earn my commit rights fixing problems and learning the SWT code base.

Grant: Not 64-bit XP. Mozilla everywhere?

JDJ: What's in the pipeline for SWT in the future?
Steve: It's too early to tell. We're looking at lots of things. Right now, it's a very interesting time for user interface technologies. Never has this space been so fragmented. We can't even agree on the computer language let alone the platform. On Windows, it's C/C++ for Win32 and C# or VB for .NET. On the Mac, C for Carbon and Objective-C for Cocoa. Linux systems support GTK+ and Qt. Many of the older workstations are still running X/Motif. Then there are the browsers. Do you use XML or AJAX to program them? Flash is a pretty cool technology. What about PHP? If you choose AJAX, what widget library do you use? You can use Dojo, but there's also GWT, J2S, and a dozen more.

One thing is certain: rather than fight technology, we will embrace it and continue to help programmers build and ship products. That's the interesting part.

JDJ: What do you think of efforts to have a declarative way of describing an SWT GUI in something like XML?
Steve: If the world goes declarative, then we will too. One thing I know for sure though, you will always need an API to manipulate widgets.

JDJ: If you had to do SWT all over again, what would you do differently with the benefit of hindsight?
Steve: Not much really. We made a few API mistakes, arguments in the wrong order and that sort of thing, but nothing major stands out. A really good indicator of this is that almost

nothing in the toolkit is deprecated. In this world of bloatware and complexity, I'm really proud of what we did in terms of getting the API right and keeping the size of the toolkit down. For example, the class hierarchies for graphics, widgets, the browser, printing, and drag and drop all fit on one slide without using a tiny font. It's amazing considering the functionality that's packed in there.

JDJ: Why are there so few interfaces in SWT? Classes like org.eclipse.swt.widgets.Layout might be a good candidate for an interface rather than an AbstractClass.

Silenio: Interfaces have a big drawback: they can never change. Once an interface ships, nothing can change, otherwise it breaks binary and source compatibility. I believe API always evolves and abstract classes make this evolution easier, without leaving a trail of dead code behind (Interface1, Interface2, etc.).

Carolyn: You have to get an interface exactly right the first time, because you can't change it without breaking binary compatibility. That's why you get silly names like "ISomething2", or, for example, "IDispatchEx"…

JDJ: Why is LayoutData typed to java.lang.Object, whereas a marker interface might help to do things like validate at compile time that the argument was actually valid? Instead, things like Assert and casts have to be done within layouts, which presumably is more expensive for performance and also a less clear API than using typed arguments. For example, instead of layoutData as an attribute on Control, you could have had Layout.setLayoutData(Control,Object) that was overloaded on each implementation, e.g., GridLayout.setLayoutData(Control ,GridData).

Silenio: There should be a common class (no interface, see above) for the layout data objects. This would allow some sharing of some common properties. On the other hand, having declared it as an Object does not necessarily mean bad performance, since usually the layout algorithm performs caching and the validation of the objects only happens when the cache is flushed.

JDJ: Likewise for arrays as arguments. Things like setSelection(String[]) or Widget[] getItems() instead of using lists and collections.

Carolyn: I like Arrays <g>. But the correct answer is that collection classes are not always available (or not completely implemented) in CLDC classes. SWT still runs on Java – what, Steve – 1.2? (I know it runs for sure on 1.4.2_06 – I use that all the time.)

Silenio: SWT runs on JDK 1.1 and collections were only introduced in JDK 1.2.

JDJ: If org.eclipse.swt.widgets.List were called ListBox, for example, there wouldn't be naming clashes with java.util.List. Do you wish you'd named List differently, and are there other places like that where you think the names used clash with base classes in an "annoying" way when doing imports.

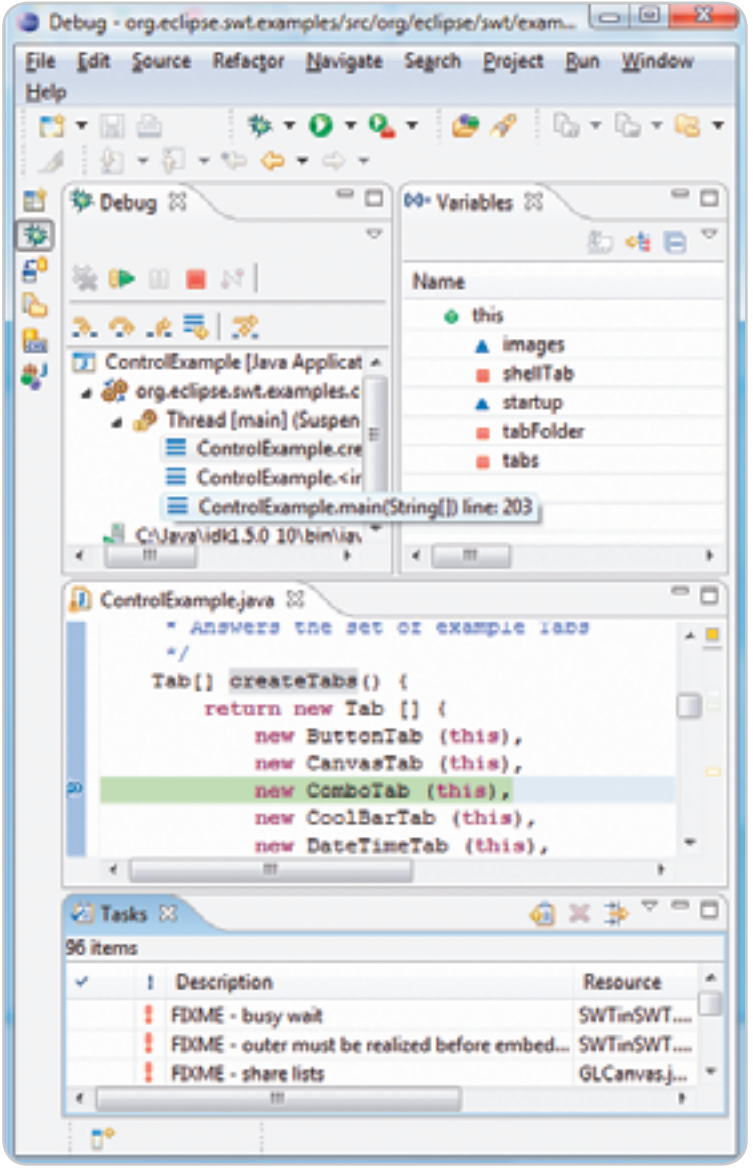
Steve: Both the AWT and SWT List existed before java.util. List. At the time, we had a big debate over whether or not we should prefix the SWT classes. I forget which side I was on. I suppose that had we prefixed them, name clashes would have been avoided, but Java has a mechanism to resolve clashes.

Carolyn: It's annoying when I'm trying to open class Button in Eclipse, and the AWT version is always at the top of the list!

JDJ: How is the Vista work going? Are there any problems with it, and also are there any cool new things you're going to put into SWT to take advantage of such as 3D support, some of the stuff they use graphics card APIs for transparency, fading, animation and so forth.

Steve: Vista is going well. The very first step is to get the Win32 port running well and taking advantage of some of the new Vista features. That's what I'm working on right now.

Silenio: It's going well. We are attacking two fronts. First, we're making sure that the Win32 port works well and has the right



Vista Window

> "One key design point of SWT is that it uses native functionality on each operating system and, at the same time, presents a common, portable API."

look and feel for Vista. This is well under way. Second, we're writing a port to WPF. It's still early to comment on this, but there is good work in progress.

JDJ: What's the coolest SWT application you've seen, and what inspires you most about how people are using SWT?
Silenio: I still have to say that Eclipse is the coolest SWT application. It's certainly the biggest and most famous. But there are others out there.

Steve: Really, I interact with people mostly through the bug system. Sometimes I see their applications and sometimes I don't. I'm not an evangelist or anything like that. I use RSS Owl. It works well. Interesting things are happening at NASA JPL, JP Morgan, and Lotus.

JDJ: What's the ugliest SWT application you've seen, and what horrifies you most about how people are using SWT?
Silenio: I haven't seen anything yet that puts a knife through my heart.

JDJ: SWT, like the rest of Eclipse, is open source. How do you think this has been a benefit for both the development of the toolkit and clients who build applications with it? Would you say that there are also some disadvantages when compared to closed source?
Olivier: I like the fact that it's open source, because when we experience a bug, or some weird behavior, we can look into the source code and try to understand why SWT behaves that way. I really don't see any disadvantages to it being open source, on the contrary. The fact that the community can help fix bugs is really something great, and so far the development of SWT has been good with new and improved features.

Florian: Stuff like having users send crash logs, then attaching them to bug reports, allowing the SWT team to debug things more directly; plus providing nightly downloads, which in turn lets developers quickly release updates by repackaging the patched JARs with their applications (as opposed to having to wait for a JRE update, and then for all users to catch up, as would be the case with other closed source systems).

Olivier: I like the fact that SWT is *not* an MVC framework, and that SWT and JFace are separated. Performance is great, and native widgets make Java desktop applications a reality. It is also a great thing for open source projects, as they can be compiled/interpreted with the GCJ compiler, which made it possible for Azureus to be included in the latest Fedora, for example.

Benjamin: I began RSS Owl in the summer of 2003 with the intention of learning and using SWT. At that time I had been using Eclipse 2.1 for only a short time and was very impressed by the fast and good-looking UI. After a couple of months I found out about the release process of Eclipse. Every Tuesday, the SWT team delivered a new integration build of SWT, including a detailed log about the changes from the past week. I became a happy reader of this change log and adopted the latest integration build as soon as it was released. Kudos to the stability of these builds. I only had to step back a few times to a previous milestone build.

The bigger RSSOwl became, the more features I wanted to add. The introduction of the Browser widget in early SWT 3.0 was exactly what I was looking for. I was finally able to render news content that included HTML. Of course there were quite a few bugs and some missing features in the early days of the Browser widget. I became a frequent user of the Eclipse Bugzilla. Up to today I have filed 152 bugs and feature requests, of which 106 have been closed/fixed. A lot of missing functionality has been added during the various major releases: Eclipse 3.0, 3.1, and 3.2. During each release there were a couple of new widgets and APIs. It's great to have the SWT library provided as open source with the SWT team doing an awesome job keeping the quality at this high level. Looking forward to what comes during the 3.3 release cycle!

Matthew: I am a huge fan of Java and open source software. Often, I find myself reading the Eclipse source code more than I read the documentation. Eclipse and SWT are quite well done. Reading the Eclipse source code has made me a better developer. I can't say this about every open source project.

I have enjoyed watching and participating in the evolution of the Eclipse project. Eclipse and SWT get better with every release. The dedicated contributors, whether they are reporting bugs or committing patches to fix bugs, are what make Eclipse and SWT so great. I have used many toolkits and SWT is my toolkit of choice.

John: The decision to launch eclipse.org and open source Eclipse was based on our business goals. We wanted to establish an open integration platform, get ISVs on board, capture the hearts and minds of developers, and, in general, create a community. We, IBM, also wanted a vehicle with which we could compete against the growth of Microsoft and Visual Studio. As a programmer, which would you prefer: an open platform driven by the needs of the larger community, or a closed, proprietary one under the control of a single vendor? As an ISV making an investment decision, which terms would you prefer: an open source license or a commercial license agreement with a single vendor? We felt that doing Eclipse as an open source project was the best way to accomplish our goals. Eclipse has succeeded, better than we ever could have imagined.

Steve: That's telling them.

JAVA DEVELOPER'S JOURNAL Advertiser Index

Advertiser	URL	Phone	Page
AjaxWorld East Conference 2007	www.ajaxworldexpo.com	201-802-3022	23
Altova	www.altova.com	978-816-1600	Cover II
Extentech	www.extentech.com/java		9
IBM	ibm.com/takebackcontrol/flexible		Cover IV
ILOG	www.ilog.com/jdj/ajax		17
Infragistics	www.infragistics.com/jsf	800-231-8588	10-11
InterSystems	www.intersystems.com/jalapeno1p		4
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	33
JavaOne	www.java.sun.com/javaone		25
Jinfonet Software	www.jinfonet.com/live	240-477-1000	7
Northwoods Software Corp.	www.nwoods.com	800-434-9820	15
SOA and EOS 2007 Conference & Expo	www.soaworld2007.com	201-802-3020	19
Software FX	www.softwarefx.com	800-392-4278	Cover III

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.

Behavioral & Philosophical Aspects of Communities

Onno Kluyt

Whether it's a prescriptive environment like the JCP or a less prescriptive one like OpenJDK and other open source software forums, communities have a lot in common.

Companies, organizations, and individual developers join or participate in certain communities – and not in others – driven by expectations of benefiting from the effort, influencing and/or leading it. The idea of joining the JCP could be motivated by the desire to make your work part of a standard, actively leading an industry in a certain direction and being recognized as a thought leader or to have one's products as closely aligned with the emergence of new standards and be seen as a market leader.

In the case of open source software communities one could want to benefit from the fruits of the commons and be better able to focus your key resources on your product's differentiation.

In either case, the active participant will enjoy advantages over those that don't participate or are passive in their participation. In the case of the JCP your advantage as spec lead and expert group member may come from being able to predict the direction of a new standard better than your competition and make product roadmap decisions accordingly. In the case of open source communities you may enjoy faster time-to-market and a better ability to react to changing market conditions due to the momentum of open source development.

While all communities attempt to attract active participation by enabling these and other benefits for those who contribute, these communities also draw boundaries around these benefits. The JCP does that through its requirement on using all the intellectual property gathered that's applied to developing compatible implementations and by timing the release of use rights. Open source communities that use the GPL have a similar concept: your derivative work must use the same terms as the community is developing the software under. Over time this balance

between benefit and obligation may move about. The JCP used to assume that Sun would have final copyright ownership, and it doesn't anymore. The JCP used to assume that all implementations derived from Sun's, and it doesn't anymore. Similarly, the Apache software license has changed over time and the Free Software Foundation is discussing version 3 of the GPL license.

A community attracts interest in part because of the structure outlined above. And the amount of active participation has a lot to do with timing. Spec leads in the JCP have learned that their JSRs often progress as fast through the process as they go. In other words, the spec lead must lead: by the rate at which she generates working drafts, the expert group will respond, act. and react. For open source software communities the attention of and access to those with committer status determines the feel of the community – and the ease of knowing where to start.

Where does the community need help, where are my skills best aligned with the work that needs to be done? The spec lead and expert group for JSRs, the current developers of an open source project must both guide newcomers. There's a risk of being too nice. "We need help everywhere!" may not be a good answer to a newly interested party trying to decide how to get involved. What do you want reviewed, what is the "to do" list for the project? Getting involved in any community has its own learning curve. Those with a steep curve may be seen as elitist because of the effort required to get into the club.

A frequent question that spec leads in the JCP get is: "When will you be ready?" The broad emotion behind this query is one that also features in open source projects: "Is anything happening here?" To show a pulse and a heartbeat is key to the success of community efforts. Communities survive on volunteers: a spec lead can develop that document elsewhere, a developer can spend his time elsewhere. Participants try to persuade other participants to "work" for them, for free. For such a social contract to work, expectations must be met. If you ask for feedback you should get it. If you provide feedback you should expect to be drawn in. If you get feedback you should expect to act on it.

As usual you can send your thoughts and comments to onno@jcp.org.

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

onno@jcp.org

> "Communities try to persuade people to "work" for them, for free. For the social contract to work, expectations must be met"