SYS-CON

INTELLIGENT GUIs SHOULD REQUIRE NO THOUGHT TO OPERATE PAGE 3

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

JDJ

JDJ.SYS-CON.COM VOL.12 ISSUE:5

3-DAY EVENT!

11th International 2007 SOAWORLD™ CONFERENCE & EXPO

PLUS 2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

www.SOAWorld2007.com

2007 VIRTUALIZATION CONFERENCE+EXPO www.virtualizationconference.com

SEE PAGES 25 and 32

THE HOLY GRAIL OF DATABASE INDEPENDENCE

RETAILERS PLEASE DISPLAY UNTIL JULY 31, 2007

$5.99US $6.99CAN

06>

0 09281 01751 6

PLUS...

- Flow Analysis: Static Analysis on Steroids
- Browser Wars and Swing on the Desktop
- Evaluating Options for Persisting Java Objects

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 (XMLSpy, MapForce, StyleVision, etc.)
- 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

Join Altova at JavaOne, San Francisco **Booth #716**

ALTOVA®

Intelligent GUIs Should Require No Thought to Operate

Joe Winchester

In Bernard J. Baar's book "A Cognitive Theory of Consciousness," he describes the brain as having a single conscious area that can be occupied by one thought at a time. The unconscious part of the brain stores memories and experiences and, like the conscious brain, is capable of performing tasks; however, it does so automatically, unlike the conscious area that requires the intervention of the "self." The first time we are given a new input, sensation, or experience to deal with, the conscious brain is responsible for analyzing it, comparing it to something that has occurred before, and dealing with the action accordingly. Repeated exposure to the same input drives the response into the unconscious area of the mind, so the next time the same experience is encountered, an automatic reply can be recalled without requiring conscious intervention.

An example of this is driving a car; an extremely complex function to perform that deals with multiple inputs – many different feedback loops to control speed, direction, and so forth. Only once you have mastered how to control the mechanics of a car, via repetition and practice, can you begin to drive it around using unconscious thought and start to concentrate on inputs such as road signs, other drivers, and assorted day-to-day driving hazards. A very real problem exists that when driving the same route over and over, what should be a conscious thought process becomes unconscious, leading to more traffic accidents occurring in your own neighbourhood on familiar roads than, ironically, far from home where conscious thought needs to be engaged. The fact that the brain can only single task can be exacerbated by functions such as talking on the phone while driving; advanced driving schools recognize this when they teach better road skills by making drivers repeat to themselves everything they are seeing as they drive, helping to ensure that it's the conscious and not unconscious brain behind the wheel.

This model of how the brain performs tasks causes problems for GUI designers in two ways: the wrong consciousness is active when input or response is required, and too much conscious thought slows down the user, creating slow and costly context switches.

An example of where things break down is with dialogs that pop up and require a response. From a design point they're the equivalent of a road user driving along happily in an unconscious mode, thinking about what they're going to do at their destination, when a deer runs in front of the car. Immediately, conscious thought is required to create a response, and as the driver creates the correct response, their locus of attention is changed from their day dreaming to the more pressing situation at hand. When you delete a file in Windows Explorer it puts it in the recycle bin asking if you to confirm you're sure. From here you can retrieve it later if you want to. If you don't want to fill up the recycle bin you can press Shift+Delete together to do a "delete and don't recycle". This too asks if you're sure with a "Yes/No" response, however the problem is that because pressing Yes to the modal dialog that follows delete has become an automatic part of deletion you don't read the dialog, and don't read and process the Shift+Delete different, potentially more severe with no recovery, question. The problem is that conscious actions, through repetition, can become unconscious, so each time a user sees the dialog they will become more accustomed to just pressing OK to dismiss it, so that dealing with the "Are you sure you want to delete?" just becomes an automatic part of the delete action. For an application that uses dialogs to grab the user's attention, which they have learned as part of the normal operation, what then must it do to really get the user woken up ? One such way is to actually stop the user from proceeding, a technique used by wizards, for example, that disables the Finish button until enough information has been completed. Again,

–continued on page 13

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

joewinchester@sys-con.com

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

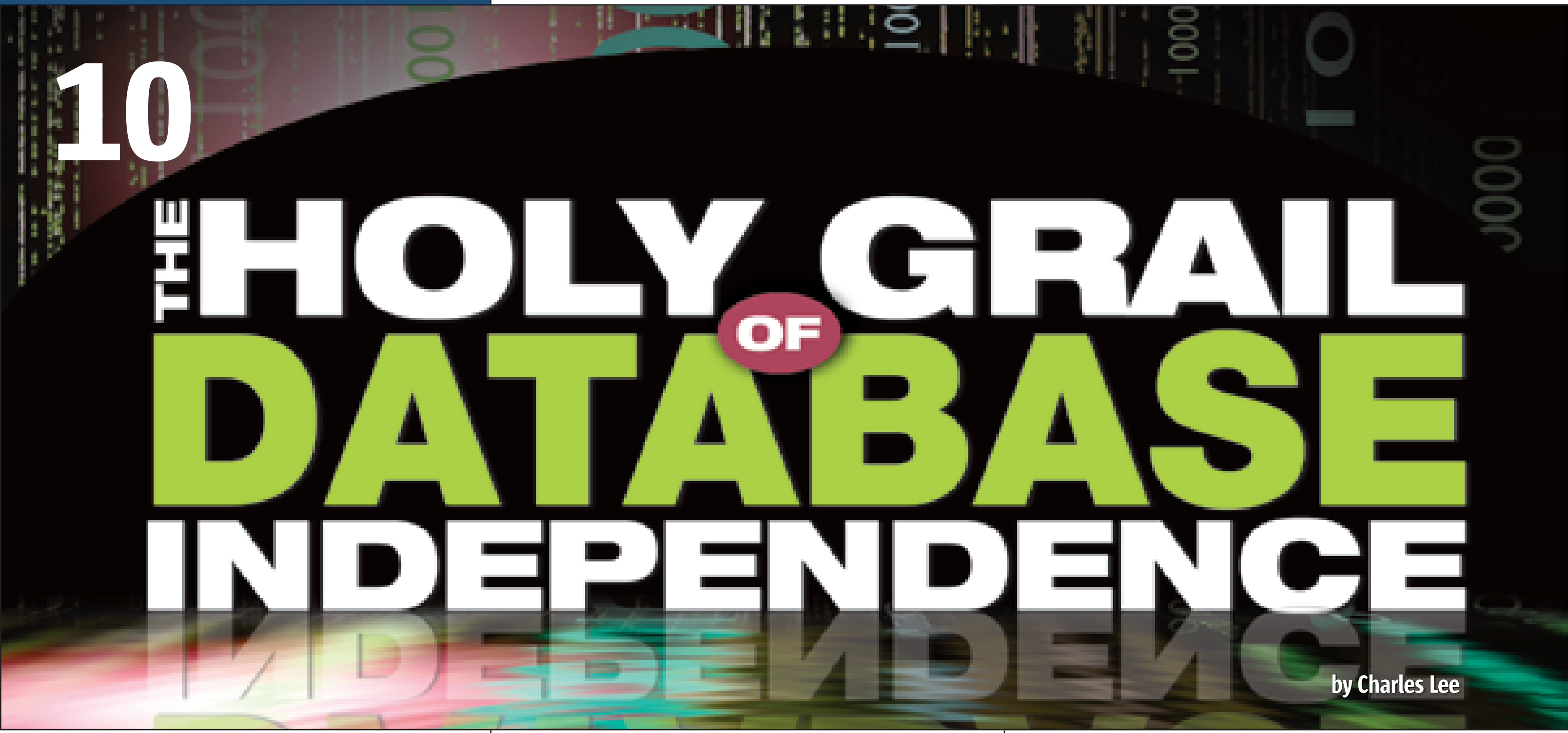
Visit us at JavaOne 2007 in booth #420.

ILOG®

Changing the rules of business™

MAY 2007 VOLUME:12 ISSUE:5

JDJ contents



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.

Nigel Cheshire
Guest Editor

Never Mind the Quality, **Feel the Width!**

"Never Mind the Quality, Feel the Width" was the title of a British TV sitcom in the late 60's (yes, I really am that old), which has nothing to do with Java software development. Or does it?

The more I talk to people about the issue of Java software quality, the more I am reminded of the name of that seemingly ridiculous TV show. It seems to me that however much we talk about the need for quality in software development, it's an issue that takes a backseat to the "width" – by which I mean the number of feature requests that get crammed into our development projects.

Many years ago, I worked for a company called Data General. Anyone with more than one or two gray hairs may remember DG as a minicomputer manufacturer (guess what – before Eclipse was an open source IDE, it was the name of a minicomputer made by DG). DG finally passed away in 1999, when what was left of it was acquired by EMC Corporation, but many people remember that back in the '80s, DG had quite a reputation for innovation, as captured in the 1981 book by Tracy Kidder, *Soul of a New Machine*.

Anyhow, the point of all this reminiscing is to illustrate that, back when I worked in the software R&D group at DG, we took software quality seriously. If we found a bug in a released software application, we'd run around like crazy until we fixed it, and send out a patch (a real patch, back in those days) to fix the problem, pronto.

Once I left DG in 1989, and moved into the wacky world of Windows/386, I was astonished at how buggy released software could be. That was my introduction to the great quality/feature tradeoff debate, and it's fascinating to me that today, almost 20 years later, that issue still exists.

Joe Winchester wrote a nice piece about quality versus features in January's JDJ; just a single page, that you could easily have flipped past if you weren't looking for it, but good stuff. Joe was mostly talking about usability in his piece, but that's just as much a part of software quality as bugginess. "It disheartens me," he said, "that the basic task of analyzing, understanding, and tooling for the user's most simple scenarios have been overtaken by an obsession to cram as much functionality as possible into the hardware."

–continued on page 8

"The more I talk to people about the issue of Java software quality, the more I am reminded of the name of that seemingly ridiculous TV show"

Nigel Cheshire is CEO of Enerjy Software, a division of Teamstudio Inc. He oversees product strategy and has been driving the company's growth since he founded it in 1996. Prior to founding Teamstudio, Inc., Nigel was co-founder and principal of Ives & Company, a CRM solutions consultancy. He holds a Bachelor of Science degree in computer science from the University of Teesside, England.

nigel_cheshire@enerjy.com

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
Advertising Sales Director:
Megan Mussa *megan@sys-con.com*
Associate Sales Manager:
Corinna Melcon *corinna@sys-con.com*

Events
Events Manager:
Lauren Orsi *lauren@sys-con.com*
Events Associate:
Sharmonique Shade *sharmonique@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*
Alicia Nolan *alicia@sys-con.com*

Innovations by InterSystems

Make Java Applications More Valuable

For Java developers, Caché offers a wealth of benefits.

When you embed Caché in your applications, they become more valuable. Caché dramatically improves speed and scalability while decreasing hardware and administration requirements. This innovative object database runs SQL queries faster than relational databases. And with InterSystems' JALAPEÑO™ technology for Java developers, Caché eliminates object-relational mapping. Which means Caché doesn't just speed up the *performance* of applications, it also accelerates their development. Caché is available for Unix, Linux, Windows, Mac OS X, and OpenVMS – and it is deployed in more than 100,000 systems ranging from two to over 50,000 users. Embed our innovations, enrich your applications.

InterSystems CACHÉ®

Visit us in Booth #520 at JavaOne, May 8th - 11th, San Francisco, CA

Download a free, fully functional, no-time-limit copy of Caché, or request it on CD, at **InterSystems.com/JavaOne2007P**

"So it's not surprising that software quality isn't seen as an important boardroom issue – we as an industry have trained our customers to expect poor quality"

–continued from page 6

In my own company, we spend a lot of time talking to people in the software industry about quality issues. When I first started out in this area, the biggest mistake I made was assuming that people at senior levels of management would be deeply concerned about the quality of the software being written in their companies. How wrong was I about that! It's not that people don't care about software quality; it's that the people who care, for the most part, are developers, technical managers, and architects, who don't necessarily have the power to bring the necessary resources to bear on solving the problem.

And these people are frustrated that they can't do more to improve quality practices, because at higher levels in the organization, or in the lines of business, there are often low expectations of the IT group's performance.

In February of this year, Forrester Research published a paper titled "Closing the CEO-CIO Gap" and subtitled "CEOs and CIOs Need Loftier Aspirations For IT's Contribution." The report indicates that CEO satisfaction with the performance of IT in their firm is high: 59% of CEOs surveyed were "satisfied or very satisfied" and 27% were "neutral." On the face of it, that seems good, until you look at the CEO's *expectations* of the IT department: only 28% of CEOs saw their IT departments as proactive sources of business innovation, and that number dropped in smaller companies of less than 1,000 employees.

So it's not surprising that software quality isn't seen as an important boardroom issue – we as an industry have trained our customers to expect poor quality. It's a bit like asking me whether I'm satisfied with the customer service provided by my cell phone network provider – well, yes, I suppose so, but that's only because I know that all cell phone network providers provide miserable customer service – so I never call them.

So, what can we do to turn this situation around? We want to do a better job, we really do – but we are constantly under pressure from the business to add more features, and to do it all in less time. So something has to give, and that something is usually unit testing, coding standards, usability testing, refactoring to make the application easier to maintain – all those things we know we should be doing but that we don't absolutely *have* to do to get the application out the door.

What we can do is start to defend our projects against the feature squeeze. Unless we start to measure what we are spending time on and the value that work delivers to the project, we have no leverage for pushing back when the pressure is on to get more features done in less time.

In his paper "Controlled Chaos: Living On The Edge" written back in 1996, Ken Schwaber made the distinction between defined processes, which perform the same way every time, and empirical processes, which by their nature are "chaotic and unrepeatable, requiring constant measurement and control through intelligent monitoring." And, you guessed it – software development is an empirical process, requiring constant measurement and control – otherwise, you get unpredictable results.

So, for those of us who care about the quality issue, and think we can do a better job, my recommendation is to look into those things that can be measured for leverage: basic quality metrics such as defect rates, unit test coverage, and coding standards violations that, in conjunction with the adoption of an agile methodology such as Schwaber's Scrum, can help bring some order to the chaos, and measure the impact of trying to cram more features in without flexing project timescales.

Then, when the customer comes by to say that he or she needs the delivery date to move forward by a month, but still needs all the same features in the finished product, we can have some way of explaining why that can't happen unless something else gives.

After all, as an industry, we should care about the quality as well as the width.

KEEPING THE $Ø IN $ØFTWARE

- ✓ Open Standards, 100% Java
- ✓ Cross-Platform, Modular, Scalable, Extensible
- ✓ Application Prototyping Utilizing NetBeans GUI Builder Matisse
- ✓ Free & Open Source Application Development Framework

NetBeans Wins Jolt Awards (http://www.joltawards.com/). NetBeans nabbed four Jolt Awards, also known as "Tech Oscars", at the 17th Annual Jolt Product Excellence Awards ceremony. NetBeans took top honors in the following categories:

Best Development Environment:
NetBeans IDE

Collaboration Tools (Productivity Winner):
NetBeans IDE

Mobile Development Tools (Productivity Winner):
NetBeans Mobility Pack

Web Development (Productivity Winner):
NetBeans Visual Web Pack 5.5

NetBeans: Improved Productivity Through Innovation
Find out more at NetBeans.org

NetBeans

The Holy Grail of Database Independence

Part 3: When your application has to support more than one database type, it makes sense to find a tool that can create the schema regardless of database type

by Charles Lee

The hope of using any persistence framework is absolute database independence. Database independence means that you can focus on your job as an application developer and not a DBA. However, no framework can fully make this claim. There's much more to running an application on a database than simply issuing compatible SQL queries and getting back the query results as expected. In my last article, I detailed the process by which we converted existing Enterprise Java Beans 2 (EJB2) Entity beans to Hibernate Plain Old Java Objects (POJOs). This article is less about our conversion process and more about the tools and methods we chose to work with for the Hibernate implementation and the backend databases (Oracle and PostgreSQL) supported by Hyperic HQ.

Creating the Database Schema

The relational database management system (RDBMS) is the foundation of any application. After all, the point of having a persistence layer is to map from the object model to the data model. The creation of the database schema is the most common task, and certainly there are plenty of point tools for various types of RDBMS around. However, when your application has to support more than one database type, it makes sense to find a tool that can create the schema regardless of database type. EJB2 provided no native tools for such a task, so we built our own tool, plainly named DBSetup. DBSetup is integrated with Ant and can easily incorporate into our build or installer (both of which rely on Ant). Its architecture is straightforward. There are base classes called Table, Column, Index, View, etc., that know how to generate the SQL commands to create themselves. Most RDBMS have proprietary extensions in their Data Definition Language (DDL), which allow control over non-standard features of the database system. If a RDBMS requires non-standard commands, you just subclass the base class, for example, the OracleTable class that can return the correct SQL to create a table in Oracle. We defined our own XML file format for the database schema, a proprietary DDL, if you will. DBSetup generates the sequence of database-specific commands in a single script and piped it to the database to create the schema.

Charles Lee is vice president of engineering at Hyperic.

For example, here's how we would define a table named SUBJECT:

```
<table name="SUBJECT">
  <column name="ID"
          default="sequence-only"
          initial="10001"
          primarykey="true"
          required="true"
          type="INTEGER"/>
  <column name="NAME"
          required="true"
          size="100"
          type="VARCHAR2"/>
  <column name="FIRST_NAME"
          required="false"
          size="100"
          type="VARCHAR2"/>
  <column name="LAST_NAME"
          required="false"
          size="100"
          type="VARCHAR2"/>
  <column name="FSYSTEM"
          type="BOOLEAN"
          default="FALSE"/>
  <index name="SUBJECT_NAME_KEY"
         unique="true">
    <field ref="NAME"/>
  </index>
</table>
```

The syntax is self-explanatory. DBSetup worked fine for us, but it meant that for any new database type that we want to support, we'd have to analyze that database's command syntax and create subclasses as needed. As I had mentioned in my last article, we went through supporting Oracle, Pointbase, Cloudscape, InstantDB, MySQL, and PostgreSQL databases, and maintaining DBSetup to be compatible with all of them was tedious. Besides, there was no association between the tables being created here and the entity beans used by the application.

Hibernate provides a better tool for schema population, *hbm2ddl*. It is also integrated with Ant. It lets you run the task against your Hibernate mapping files (HBMs) and

Help is here.

At last!

- *Lightweight*
- *Java™ browser SDK*
- *Based upon Mozilla™*
- *Faithful robust rendering*

Swing Edition also features:

- *Pure Swing lightweight rendering*
- *Web Standards compliance*
- *W3C DOM*
- *Easy to embed*
- *Java Applet and plugin support*
- *Extensive API*

All WebRenderer products support the following standards: HTML, XHTML, CSS 1 & 2, JavaScript (inc AJAX), DOM, HTTP, HTTPS, XML, XSL, plugins etc. with faithful and always predictable rendering. Check out our other products:

PLUS Desktop Edition

- Supports *Internet Explorer™*, *Mozilla™* & *Safari™* spawns
- Fast and robust SDK
- Micro edition available
- Native font support
- Faithful, predictable rendering

PLUS Server Edition

- Server optimized and scalable
- J2EE support
- Leading server-side Java browser
- Render to thumbnail image
- Headless print support SDK

JadeLiquid™ Software

DOWNLOAD FREE 30 DAY TRIALS NOW! WEBRENDERER.COM

generate the resulting data definition language (DDL) in a file or to be exported directly into the database. Since we chose not to use the annotation feature with Hibernate, we hard-coded the HBM files. (Note that even if you were to use annotations, you can still use Hibernate tools to create the schema with your mapping.) We were able to convert our DBSetup schema files to HBM files relatively fast due to the structural similarities. Furthermore, we simplified our manually maintained Hibernate configuration files by offloading our defaults into an XSLT transformation process at build time, so our HBM files can be as minimal as possible.

Hibernate has classes that support various database dialects, so all of a sudden we've gained the ability to create schemas for a wide variety of databases without doing additional work ourselves. Hibernate's Web site lists the supported database types:

- Oracle 8i, 9i, 10g
- DB2 7.1, 7.2, 8.1
- Microsoft SQL Server 2000
- Sybase 12.5 (JConnect 5.5)
- MySQL 3.23, 4.0, 4.1, 5.0
- PostgreSQL 7.1.2, 7.2, 7.3, 7.4, 8.0, 8.1
- TimesTen 5.1, 6.0
- HypersonicSQL 1.61, 1.7.0, 1.7.2, 1.7.3, 1.8
- SAP DB 7.3
- InterSystems Cache' 2007.1

It's good to know that we have options.

Populating the Data

Most applications will require that some of the database tables be initialized with data. HQ certainly does. The criteria for our data population is the same as schema creation, it has to be database-independent to support the various databases we support (or once supported). Hibernate doesn't have any tools to support this task. You can, of course, do this programmatically through the application using Hibernate POJOs. As a developer, you wouldn't have to use a separate tool or learn another configuration format. However, this is not the most straightforward approach, you'll probably write more code than you need. There are tools available for such things. DbUnit Framework will easily populate a database with an XML file containing the initialization data. As for us, we had rolled our data initialization functions into DBSetup.

We began building HQ in 2001; there was nothing available in open source. In fact, we even considered creating an open source project for DBSetup, but decided that we didn't have the bandwidth to maintain the project in the open source world. We defined the data in a database-independent format in XML files that are separate from the aforementioned schema XML files, and DBSetup would, of course, generate the SQL according to the database column types. An insertion into the SUBJECT table looks like this:

```
<table name="EAM_SUBJECT">
  <data id="1"
        name="admin"
        first_name="System"
        last_name="User"
        factive="TRUE"/>
</table>
```

Since we designed HQ as a conglomerate of subsystems (authorization, measurement, control, etc.), we organized the database initialization by having a schema XML file and a data XML file for each of the subsystems. After our conversion to Hibernate, we kept the data population function of DBSetup and switched over to hbm2ddl for the schema creation.

Upgrading the Database Schema

Hyperic has some loyal customers, some of whom date back to HQ 1.0 (thanks for sticking with us). However, this means that we have to carry forward the customers' data from some very different schemas from previous versions. Upgrading the schema is an area that is the least explored by existing tools, but represents one of the greatest challenges that we faced. hbm2ddl has an *update* option that attempts to create an upgrade script that calculates the delta between what's in the database and the mapping files. However, there's a fairly unsettling warning in Hibernate's tools documentation regarding this operation:

*(Do *not* use against production databases, no guarantees at all that the proper delta can be generated or that the underlying database can actually execute the needed operations.)*

In our development, we found that changes to existing columns and new indexes were not created by the update option, not to mention that it didn't support alteration of the stored data as a part of the migration. Clearly, this is inadequate for our purposes, since we have customers that rely on our software to manage their production machines. We had previously developed another database tool, also integrated with Ant, called DBUpgrader. Unfortunately, DBUpgrader isn't the most elegant of solutions. It supports some basic database functions, such as change column type, add new column, and data insertion. It's fairly cumbersome to maintain, because RDBMSes were different in their SQL variations, particularly for alterations and index creation. So there were a lot of direct SQL statements (we tag each statement for a specific database type as needed). After creating a fresh schema through hbm2ddl from our Hibernate mapping files, we saw that we had a lot of work to do to migrate our previous schema versions to match what Hibernate created. The process was pretty brute force, we simply dumped the new database schema to a file and used it as a reference to hand-code just about every new table, column, index, constraint, as well as transform and migrate the existing data and make column type changes. Perhaps there will be a better tool someday, but we stuck to a tool that worked for us, as cumbersome as it was (the entire upgrade XML file clocks in at about 5000 lines).

Container Managed Relationships (CMR)

Bidirectional CMRs are one of the very few instances where one might consider EJB2 superior to Hibernate. Take the simple example of the parent/child relationship.

In EJB2, you can set the association from either side of the relationship. This means that you can either do child.setParent(parent) or parent.getChildren().add(child) in the application, and expect the framework to update the appropriate entities and automatically persist the foreign key relationship to the database. However, this is not the case with Hibernate. Hibernate doesn't have "true" bidirectional association. The issue is with the data integrity, and being able to create a data schema in the RDBMS with the correct constraints that mirror the object model. In this case, a child couldn't exist without a parent, so we want to put a non-null foreign key constraint on the *PARENT* column on the *CHILD* table in the database. However, this means that you'd be prevented from performing an operation like parent.getChildren().remove(child), because the child would be orphaned without a parent, which is against the model. For EJB2, it means you forgo that constraint, and actually allow nulls in the child's *PARENT* column. However, it made coding much easier. In Hibernate, you'd define one side of the relationship as read-only using the attribute inverse="true." Hibernate persists the association only through the non-inverse side of the relationship. What does that mean? If we were to define the inverse attribute in the parent's mapping (one-to-many), it means that the parent's children collection is read-only. Hibernate doesn't update the association through any modification to the parent's children collection.

Consider the following code:

```
Parent alice = new Parent();
Parent jane = new Parent();
Child mary = new Child();
mary.setParent(alice);
jane.getChildren().add(mary);
```

Who ends up being mary's parent? Since we defined the inverse attribute on the parent, it means that Hibernate persists the association only when we invoke setParent() on the child. In the end, mary's parent is alice. The effect of this is that we end up having to code the association on both sides to be safe. To delete a child, you have to make sure to delete the child POJO and from its parent, otherwise you'd get the dreaded "deleted object would be re-saved by cascade (remove deleted object from associations)" exception, because the parent object would still retain the child object in its collection and not remove it automatically. We certainly had to work through plenty of those problems before the application functioned correctly as before, since we became reliant on the flexibility of being able to associate entities from either side of the relationship. However, having a stricter set of constraints also guarantees that you don't end up with mysteriously orphaned rows in the database and a whole different set of problems (which we had experienced with EJB2).

Successful Quest?

Obviously, the end result is not a fully database-independent application. As of today, Hyperic HQ only supports Oracle and PostgreSQL backends. However, the migration to Hibernate improved portability on many fronts. There's also stronger mapping between Java objects and the database persistence. The tools aren't complete, and we are maintaining some tools that leave much to be desired. We'll soon put this recipe to the test, though. By the time you read this, we will probably have ported Hyperic HQ to run on additional databases. In my next article, we'll move into the application layer and examine how Hibernate queries the database and compare the merits between SQL, EJBQL, and HQL.

Intelligent GUIs Should Require No Thought to Operate

by Joe Winchester, *Desktop Java Editor*

–continued from page 3

though, the desire to spend as little time being interrupted leads the user to learn all that is required is the minimum set of input to make the Finish button enabled and get the dialog to dismiss itself. Despite this, I find that it's a rare GUI designer who will avoid the knee-jerk decision to use a modal dialog to report a situation that, while exceptional to the application, is just a distraction and annoyance to the user, and for modal screens to be used for dialogs such as property pages, creation wizards, and so forth.

The second problem with mapping brain function to GUIs is that the mind is not a pre-emptive parallel multi-tasking machine, but it is basically capable of serial thought. Switching between one thought and another requires moving the locus of conscious attention, rather like moving an imaginary cursor inside the mind to a new position. This takes time to do and, for the task being left behind, this will either fade from short-term memory in a few seconds, or else must be moved to unconscious thought by repetition or reinforcement.

GUIs that work well are those that embrace the way the mind works, and only invoke conscious thought for the experienced user as and when required. The problem that GUIs often encounter is that they're sold on the basis of how they look, the sparkle and up-front razz, and how quickly they can simplify tasks they're engineered specifically to demo well. However, the true test comes when a skilled user can operate them with unconscious efficiency to mirror the speed at which touch typists can capture information into a console-based text app. I've worked on several failed applications for financial institutions where we wrote superb GUI applications that the users just rejected out of hand because there were too many screens presented in a manner that, while familiar to IDE users' unconscious brains, were just over complex to end users who reverted to spreadsheets to capture all their data and create outputs. To get the edge over basic tools it needs to replace, GUI design must engage the user's brain on their terms, not that of the designer or the framework and language on which it runs.

Flow Analysis: Static Analysis on Steroids

by Nada daVeiga

How and why to add flow analysis to your existing testing strategies

There are three main types of software bugs:

- ***Poorly implemented requirements*** – The software doesn't operate as expected because the functionality defined in the requirements was implemented incorrectly.
- ***Missing or incomplete requirements*** – The software doesn't perform necessary operations or handle feasible scenarios because the stakeholders/designers didn't anticipate the need for such functionality and didn't specify requirements for it, or because the developers failed to implement a specified requirement.
- ***Confused user*** – The software was designed in a way that lets confused users take unexpected paths.

Building a robust regression suite is the best way to identify poorly implemented requirements, and performing negative testing is the best way to identify confused user errors. However, finding missing requirements is difficult because it's not clear what you're looking for. Flow analysis, which basically analyzes paths through the code without executing it, is the only known automated testing technique that leads you to such problems. For instance, assume that flow analysis identified a NullPointerException in a Java application. If you examine the path that led to the exception, then consider the conditions under which the program might end up going down this path, you might find a missing requirement – for instance, the exception could be caused by a certain situation that's feasible but was never anticipated during the design/planning phase. Such problems wouldn't be noticed if testing focused solely on writing tests that verify requirements.

Besides pointing to missing requirements, flow analysis also can expose construction problems and logical flaws in an application. It's designed to be used as part of a comprehensive regression test suite that also includes pattern-matching static analysis, JUnit tests, Cactus tests, HttpUnit tests, and any other tests you use to verify the software. Running the complete regression test suite – including flow analysis and all the other tests – automatically and regularly is the most effective way to determine if code modifications/additions introduced new problems, broke existing functionality, or caused unexpected side effects.

This article examines why and how to add flow analysis to your existing testing strategies. After introducing the general concept and benefits of flow analysis, it demonstrates how flow analysis helps you find critical runtime bugs without executing code.

Nada daVeiga is the Product Manager of Java Solutions at Parasoft, where she has been a senior member of Professional Services team for two years. Nada holds a bachelors degree in computer science from the University of California, Los Angeles (UCLA).

nada_daveiga@parasoft.com

Static Code Flow Analysis - Background

The term *static code analysis* means different things to different people in the software industry. There seems to be two main static analysis approaches: (1) program execution or flow-based analysis and (2) pattern-based analysis.

For program execution adherents, *static analysis* means trying to logically execute the program – sometimes symbolically – to uncover code problems such as memory corruption, leaks, and exceptions. This type of testing largely focuses on identifying code problems without creating test cases. It provides developers with the "instant feedback" they need to address defects and security vulnerabilities quickly on the desktop – while they're still working on the code and it's fresh in their minds – and it prevents defects and vulnerabilities from making their way further downstream in the software development process, which is where they're much more expensive to identify and remediate.

Flow analysis can be done using automated technologies that determine whether the application's execution paths match "suspicious behavior" profiles. For each defect found, a hierarchical flow path details the complete execution path that leads to the identified defect, ending with the exact line of code where the bug manifests itself. In some cases, the analysis configurations can be customized to make the analysis process more flexible and tailored to your unique project needs. As a result, flow analysis can even be used to detect violations bound to the use of very specific APIs.

Using flow analysis, development teams gain the following key benefits:

- ***Perform more comprehensive testing with existing resources:*** Flow analysis complements other testing techniques by letting you find problems that would otherwise require the development, execution, and maintenance of complex test cases. The defects exposed by flow analysis would be very difficult and time-consuming to find through manual testing or inspections, and would be exponentially more costly to fix if they weren't detected until runtime. Flow analysis lets developers quickly find, diagnose, and fix classes of software errors that can evade pattern-based static analysis and/or unit testing. Exposing these defects early in the software development lifecycle saves hours of diagnosis and potential rework.
- ***Automatically identify defects that pass through multiple classes:*** Most developers have done thorough testing on a class, corrected all the apparent problems, integrated the code then later encountered problems, such as NullPointerExceptions, that took days to diagnose because they resulted from an obscure or complex execution path that passed through multiple methods or even multiple packages. Using flow analysis, the same problem can be identified in seconds.
- ***Focus on actual defects and misuses:*** Flow analysis results typically indicate actual misuse (as opposed to the possible/hypothetical misuse that might be reported during unit testing). For example, flow analysis shouldn't report a violation for the following code unless there was a method in the source code calling strlen and passing it a null value, but unit testing could report a problem regardless by passing null to the strlen method in the test:

```
 public int strlen(String str)
{
    return str.length();
}
```

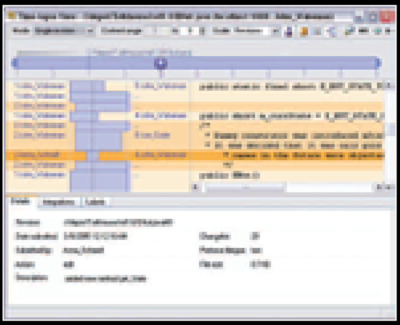
Perforce | *Fast* Software Configuration Management

MEET PERFORCE AT
JAVAONE BOOTH
#415

SEE A DEMO OF THE
PERFORCE PLUG-IN
FOR ECLIPSE

Introducing Time-lapse View,

a productivity feature of Perforce SCM.



Perforce Time-lapse View

Time-lapse View lets developers see every edit ever made to a file in a dynamic, annotated display. At long last, developers can quickly find answers to questions such as: 'Who wrote this code, and when?' and 'What content got changed, and why?'

Time-lapse View features a graphical timeline that visually recreates the evolution of a file, change by change, in one fluid display. Color gradations mark the aging of file contents, and the display's timeline can be configured to show changes by revision number, date, or changeset number.

Time-lapse View is just one of the many productivity tools that come with the Perforce SCM System.

PERFORCE
SOFTWARE

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

Method Name	TestField.java	TestLocal.java
falsePositive1	X	X
falsePositive2	X	X
falsePositive3	X	X
falsePositive4	X	X
ifalsePositive1	X	X
truePositive1	✓	✓
truePositive2	✓	✓
truePositive3		
truePositive4	✓	✓
truePositive5	✓	✓
truePositive6	✓	✓
itruePositive1	✓	✓
itruePositive2	✓	✓
itruePositive3	✓	✓

Table 1 Defects identified by Jtest's flow analysis. X indicates that a flow analysis violation wasn't reported in the method and [check mark] indicates that a flow analysis violation was reported in that method.

Running Flow Analysis

To better understand the types of defects that flow analysis can expose, consider how it can be applied to two sample Java classes. For our purposes, flow analysis will be done with the BugDetective technology featured in Parasoft Jtest.

One sample class involves a class instance field that can be null (Listing 1 – TestFields class) and the second one involves the same class with a local variable that can be null (Listing 2 – TestLocal class). Both classes call a LocalHelper class. The goal is to demonstrate how flow analysis handles (1) intra-procedural calls, and (2) inter-procedural calls (a) within one class and (b) that cross class boundaries.

Both of the examples (see below) contain instance field and local variable variations of the same defects. The methods named "falsePositive" contain false positives and the methods named "truePositive" contain true positives.

To do the flow analysis, I selected the two sample classes in my IDE (Eclipse) then ran a "BugDetective" Test Configuration. This flow analysis flagged the following defects in the two files (see Table 1).

All false positives are marked in blue and all true positives are marked in red. X indicates that a flow analysis violation wasn't reported in the method and ✓ indicates that a flow analysis violation was reported in that method.

Taking a closer look at the results, notice that no false positives were flagged in these examples. Also notice that Jtest's flow analysis found the defects in the *truePositive3* method to be false positives even though other technologies may report them as true errors.

Consider the following code from the TestFields class:

```
Object x; //NPE origin
TestFields(Object x) {
  this.x = x;
}

int truePositive3(boolean b) {
       Object y = null;
        if (x != null)
            y = new Object();
        if (y != null)
            return x.hashCode() +
y.hashCode();
        else
            return x.hashCode(); //NPE
}
```

The instance variable x is initially initialized to null, but it gets reassigned to the value of argument x in the constructor call.

This violation wasn't flagged during flow analysis because when simulating execution paths through the code, the flow analysis technology saw a potential violation point on the path (the line marked with //NPE) but it didn't see a path from the violation origin statement (the line marked with //NPE origin) to that line without going through a constructor. This wasn't reported as a violation because the flow analysis didn't find a line where x is initialized to null. The code didn't find a path in the source code that contains the following sequence of steps:

```
TestFields tf = new TestFields();
tf.truePositive3(true|false);
```

Nor did it find a path such as this:

```
TestFields tf = new TestFields(null);
tf.truePositive3(true|false);
```

However, assume that the following method is added to the TestFields class:

```
void callerTruePositive3() {
 TestFields tf = new TestFields(null);
tf.truePositive3(true);

}
```

Flow analysis now flags this violation since it sees the violation origin and violation point, as well as a code path that leads from one to the other.

Conclusion

Flow analysis helps software development teams find critical runtime bugs without executing code. Since it tries to check whether potential problems could actually be triggered by real application paths, it reports an extremely high ratio of true positives to false positives. This means that you'll be alerted to problems that are likely to occur at runtime – but you won't need to waste time reviewing an overwhelming number of false positives. This is especially helpful if you need a fast way to zero in on critical defects in a large code base.

When flow analysis is applied as part of a comprehensive regression test suite, it helps development teams to:

- ***Increase team development productivity*** by identifying and addressing defects from the earliest phases of the development cycle – when fixing them requires minimal effort and rework.
- ***Achieve more with existing development resources*** by automatically vetting known coding issues so developers and QA can spend more time on tasks that require human intelligence.
- ***Build on the code base with confidence*** by efficiently constructing, continuously executing, and maintaining a comprehensive regression test suite that detects whether updates break existing functionality.
- ***Decrease time to market*** by building an efficient, consistent, and controlled team workflow for applying best practices that reduce testing time, testing effort, and the number of defects that reach QA.
- ***Reduce support costs*** by automatically performing negative testing on a broad range of potential user paths to uncover problems that might otherwise surface only in "real-world" usage.
- ***Quickly expose problems in complex, difficult-to-test systems*** by automatically exposing many critical bugs in software for SOA and Java EE without involving staging systems.

–Listings 1 and 2 are on pages 18 & 20

Embedded Reporting and Analytics

Empower Java Applications with Operational Business Intelligence.

Only JReport has what it takes to make both developers and users happy.

Developers are happy to embed reports with sophisticated layout options, Web controls and integrated security. JReport makes their life easier with top performance, scalability and seamless Java EE deployment.

Users are happy to get their reports pre-scheduled or on demand; view them with cascading parameters, dynamic drilling, sorting, filtering and precise exporting. JReport's self-service ad hoc reporting lets users create analytic reports intuitively and work with them interactively.

The bottom line: JReport makes the business more efficient and more productive. And that makes everyone happy.

See for yourself. Visit www.jinfonet.com/j to learn more about the power of JReport.

See us at JavaOne Booth 738

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

JReport®
JINFONET SOFTWARE

Note: The two sample classes were adapted from the code featured in David Hovemeyer and William Pugh's article "Finding More Null Pointer Bugs, But Not Too Many," which is available at http://findbugs.cs.umd.edu/papers/MoreNullPointerBugs07.pdf.

Listing 1

```
public class TestFields {

   Object x;
   TestFields(Object x) {
         this.x = x;
   }
    int falsePositive1(int level) {
        x = null;
        if (level > 0)
            x = new Object();
        if (level > 4)
            return x.hashCode();
        return 0;
    }

    int truePositive1(int level) {
        x = null;
        if (level > 0)
            x = new Object();
        if (level < 4)
            return x.hashCode();
        return 0;
    }
    int falsePositive2(boolean b) {
        x = null;
        if (b)
            x = new Object();
        if (b)
            return x.hashCode();
        return 0;
    }
    int truePositive2(boolean b) {
        x = null;
        if (b)
            x = new Object();
        if (!b)
            return x.hashCode();
        return 0;
    }

    int falsePositive3(boolean b) {
        Object y = null;
        if (x != null)
            y = new Object();
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return 0;
    }
    int truePositive3(boolean b) {
          Object y = null;
        if (x != null)
            y = new Object();
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return x.hashCode();
    }
    int falsePositive4(boolean a, boolean b) {
          x = null;
        Object y = null;
        if (a) x = "x";
        if (b) y = "y";
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return 0;
    }

    int truePositive4(boolean a, boolean b) {
          x = null;
        Object y = null;
        if (a) x = "x";
        if (b) y = "y";
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return x.hashCode();
    }

    int truePositive5() {
          if (x == null) return x.hashCode();
          return 0;
    }

    int truePositive6() {
          if (x == null) {
                  Object y = x;
                  return y.hashCode();
          }
          return 0;
    }

    int ifalsePositive1(boolean b) {
        x = null;
        if (!b)x = new Object();
        return LocalHelper.helper1(x, b);
    }

    int itruePositive1(boolean b) {
        x = null;
        if (b) x = new Object();
        return LocalHelper.helper1(x, b);
    }

    int itruePositive2() {
          x = null;
        return LocalHelper.helper2(x);
    }

    int itruePositive3(boolean b) {
          x = null;
        if (b) x = "x";
        return LocalHelper.helper3(x);
    }

}
```

Listing 2

```
public class TestLocal {

    int falsePositive1(int level) {
        Object x = null;
        if (level > 0)
```

Listing 2 continued on page 20

Build Geography Into Your Applications

Population demographics analysis application

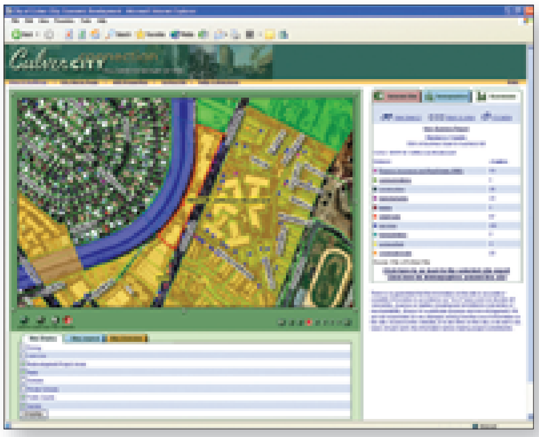
Give Your Users the Complete Picture to Help Them Make Better, Faster Decisions.

Applications that incorporate geographic information system (GIS) technology give users a visual way to analyze their data and make more informed decisions. With ESRI® developer solutions, you can quickly and cost-effectively bring geography and mapping capabilities into your applications, regardless of whether you are building desktop, client/server, mobile, or Web applications.

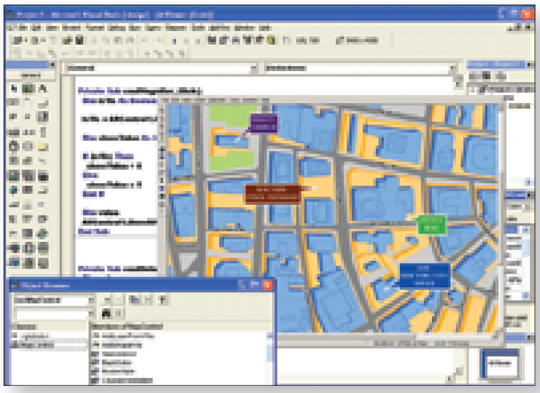
ESRI developer solutions enable you to

- Quickly and cost-effectively integrate GIS capabilities into your new and existing applications.
- Select the developer tools that fit best with your architecture (ESRI's developer products encompass GIS components, servers, and Web services).
- Use the development environment of your choice, including Java™, .NET, COM, and C++, and deploy applications on a variety of platforms.
- Access and manipulate data in multiple formats.

To learn more about the ESRI developer solutions that are right for you, visit **www.esri.com/develop**.



Web-based property management system



Using GIS components within a commercial IDE

ESRI

1-888-288-1277

www.esri.com/develop

info@esri.com

```
            x = new Object();
        if (level > 4)
            return x.hashCode();
        return 0;
    }

    int truePositive1(int level) {
        Object x = null;
        if (level > 0)
            x = new Object();
        if (level < 4)
            return x.hashCode();
        return 0;
    }

    int falsePositive2(boolean b) {
        Object x = null;
        if (b)
            x = new Object();
        if (b)
            return x.hashCode();
        return 0;
    }

    int truePositive2(boolean b) {
        Object x = null;
        if (b)
            x = new Object();
        if (!b)
            return x.hashCode();
        return 0;
    }

    int falsePositive3(Object x, boolean b) {
        Object y = null;
        if (x != null)
            y = new Object();
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return 0;
    }

    int truePositive3(Object x, boolean b) {
          Object y = null;
        if (x != null)
            y = new Object();
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return x.hashCode();
    }

    int falsePositive4(boolean a, boolean b) {
          Object x = null;
        Object y = null;
        if (a) x = "x";
        if (b) y = "y";
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return 0;
    }

    int truePositive4(boolean a, boolean b) {
          Object x = null;
```

```
        Object y = null;
        if (a) x = "x";
        if (b) y = "y";
        if (y != null)
            return x.hashCode() + y.hashCode();
        else
            return x.hashCode();
    }

    int truePositive5(Object x) {
          if (x == null) {
                    return x.hashCode();
          }
          return 0;
    }

    int truePositive6(Object x) {
          if (x == null) {
                    Object y = x;
                    return y.hashCode();
          }
          return 0;
    }

    int ifalsePositive1(boolean b) {
        Object x = null;
        if (!b) x = new Object();
        return LocalHelper.helper1(x, b);
    }

    int itruePositive1(boolean b) {
        Object x = null;
        if (b) x = new Object();
        return LocalHelper.helper1(x, b);
    }

    int itruePositive2() {
        return LocalHelper.helper2(null);
    }

    int itruePositive3(boolean b) {
          Object x = null;
        if (b) x = "x";
        return LocalHelper.helper3(x);
    }

}

public class LocalHelper {

//  Bug when x is null and b is false
    public static int helper1(Object x, boolean b) {
        if (b) return 0;
        return x.hashCode();
    }

    public static int helper2(Object x) {
        return x.hashCode();
    }

    public static int helper3(Object x) {
        return x.hashCode();
    }

}
```

JavaOne℠

Java™

OPEN POSSIBILITIES

May 8–11, 2007
The Moscone Center, San Francisco, CA
JavaOne Pavilion: May 8–10, 2007

java.sun.com/javaone

> JAVA™ TECHNOLOGY IS NOW OPEN–AND SO ARE THE POSSIBILITIES

The 2007 JavaOne℠ conference has expanded and is definitely one conference you won't want to miss. With the decision to open source Java™ technology, 2007 marks a major milestone for the Java platform. Whether your passion is scripting languages, open source, SOA, Web 2.0, mashups, or the core Java platform, this is a conference that has something for almost all technology developers.

LEARN MORE ABOUT*:

> Scripting
(JavaScript™ Programming Language, PHP, Ruby on Rails, Python, and More)
> Open Source and Community Development
> Integration and Service-Oriented Development
> Web 2.0 Development
> AJAX
> Java Technology and the Core Java Platforms (EE/SE/ME)
> Compatibility and Interoperability
> Business Management

SAVE $100**
Register Today!

Please use priority code: J7PASC

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

Attend the JavaOne conference, and you will have many opportunities over the course of four days to network with like-minded developers; attend in-depth technical sessions; engage with your peers in Hands-on Labs and BOFs; and experience general sessions featuring speakers from Intel Corporation, Motorola, Inc., Oracle Corporation, and Sun Microsystems, Inc. Meet face-to-face with leading technology companies, and test-drive the latest tools and technologies shaping the industry.

PLATINUM COSPONSORS

intel MOTOROLA ORACLE

GOLD COSPONSORS

AMD Smarter Choice · bea Think liquid. · IBM · NAVTEQ · NOKIA · SAP

SILVER COSPONSORS

InterSystems · Sprint Together with NEXTEL · PARASOFT We make software work. · TERRACOTTA

Sun microsystems

Evaluating Options for Persisting Java Objects

Hibernate, DB4O, and Caché Database with Jalapeño

by Richard Conway

We live in a relational world – which is too bad since we develop with objects. Since most non-trivial applications require information to be persisted and retrieved in what is generically called a database, we need to find efficient methods for persisting our objects and retrieving them. Historically, this has been done with relational databases and lots of code that flattens the objects and maps them to the relational tables. This can be done in Java or with object-relational mapping tools like Hibernate.

While most books and articles about object-oriented software development discuss the benefits of using objects to describe the problem space, currently most development is done in a more convoluted manner, working from both the Java and database ends and using various technologies to bridge the gap between them. Hibernate has emerged as one of the most popular ways to address this development challenge.

Instead of starting with a database schema and building objects from the tables and data therein, this article will focus on starting with the Java objects and evaluate a few of the many options for persisting them. We will compare and contrast the methods, as well as discuss challenges and problems specific to object persistence management (see Table 1).

Options for Persisting Java Objects

There are many options for persisting your Java objects. Some are best used when your persistence needs are simple (such as saving program state between sessions) and some are better when your needs are complex (such as saving lots of data over long periods of time). Some applications simply need to load data for use in configuring the application, while others require sophisticated tools for searching and filtering object sets. The latter is typical for applications that use a database for persistence and will be the primary focus here. For each project you work on, you should evaluate which persistence strategy best fits your needs based on the project requirements. The following persistence mechanisms will be discussed in this article:

- Hibernate
  - The most commonly used object-relational mapping tool
- DB4Objects DB4O
  - A small simple embeddable object database
- InterSystems' Caché Database with Jalapeño
  - An enterprise database that lets you store POJOs via its Jalapeño technology – but also provides a JDBC/SQL interface to the objects stored in the database

Please note that there are many other object-relational mapping solutions and object databases available, the purpose of this article is simply to provide some insight as to the benefits of using these technologies versus a JDBC/DAO implementation, how they compare and to how you can use them for your projects.

Definition: **DataStore** – A system for storing and retrieving data. Can be relational or object-oriented.

Object Persistence Mechanism Considerations

For each of the persistence strategies outlined above, I will review the following:

- Ease of Implementation
  - How much preparation and/or configuration is required to persist your objects?
- Ease of Persisting Objects
  - How do you persist an object or objects?
  - Is it straightforward and intuitive?
  - Is it better/simpler/faster than using SQL and writing DAOs?
- Ease of Retrieving Objects
  - What mechanisms are available for finding and retrieving the objects in the datastore?
- Control over Object Depth
  - How many objects do you want to save or retrieve at the same time?
- Control over Object Property Breadth
  - How many properties do you need access to? Can you only return those properties?

	Typical/Object Relational	Desired/Object Oriented
1	Define the problem space using Nouns and Verbs to identify Objects and Methods	Define the problem space using Nouns and Verbs to identify Objects and Methods
2	Identify all the data that must be persisted as Objects and/or Tables, and document the relationships between the OBJECTS	Identify all the data that must be persisted AS OBJECTS and document the relationships between the OBJECTS
3	Develop a relational database schema to contain the data, representing the Objects as one or more Tables in a database. Work towards 3rd Normal Form	Create the database based on the Object Schema. Ideally, you have a one-to-one mapping between your business objects and your database object store
4	Write SQL Code, create Data Access Objects, or use JDO to persist and load data between the POJOs and the database	Avoid writing Data Access Objects or the equivalent. Deal with objects exclusively

Table 1 The software development process

Richard Conway is a software developer and technology consultant with more than 15 years of technology, project management, and information services experience. He has extensive experience developing Java/Struts-based web applications. He started focusing more on Swing based developments at the beginning of 2005 and has just finished a Swing-based client/server asset management project. He lives in Miami with his wife Patricia, is currently working on an EMR application, and plays sand volleyball in his spare time.

reconway@egrok.com

- Object Tree Traversal
  - Can you access all related objects and their properties simply by traversing the object tree?
- Enforcing Referential Integrity
  - How do you ensure you don't delete an object that other objects depend on?
    - * For example, can you delete a department if employees still exist?
  - Does it support cascade deletes/updates?
- Enforcing Uniqueness
  - How do you ensure that specific properties are unique in the database, such as Social Security numbers?
- Support for Indices
  - Can you define indices that will enhance the performance of your queries?
- Property Constraints
  - Can you control/limit the values that will be entered into the datastore?
- Security and Access Control
  - How do you control who has access to the data and what they can do with it?

Ease of Implementation

One of the things we're trying to get away from is the effort to write and maintain DAOs. So let's look at what's involved in setting up your datastore and prepping your POJOs to be persisted for each of the persistence mechanisms identified.

Hibernate

Hibernate has the most complex setup of the solutions discussed here, but it's not that bad and Hibernate does provide a lot of tools to make your life easier. There are multiple ways to implement Hibernate, but since we are starting the POJOs, we'll only consider the case of adding annotations to the Java class to support Hibernate persistence. If you're using Hibernate 3.2 with Java 1.5 or above, you can use annotations to map your POJO properties to your Table columns. Using annotations is much less verbose than defining your mappings in XML files and has the additional benefit of reducing the number of files you must keep track of. In addition to annotating your POJOs you need to provide the fully qualified name of the annotated class as a <mapping> element in the hibernate.cfg.xml file.

The minimal steps for preparing to persist your objects with Hibernate are:

1) Set up a database and create a username and password for Hibernate to access it with
2) Annotate your Java Classes
3) Add your class mappings, database, and user/login information to the hibernate.cfg.xml file
4) Run the Hibernate hbm2ddl to create the schema in the database

An example of mapping in the hibernate.cfg.xml file:

```
<mapping class="com.egrok.hibernate.Person"/>
```

Since with Hibernate, you're typically mapping to a relational database, complex objects may need to be persisted to multiple tables, adding some complexity to the implementation.

To save you the effort of annotating everything, Hibernate provides many default behaviors. For example, Hibernate by default will map your class to a Table of the same name so you don't have to use the @Table annotation unless the class name and table name differ. The same goes for class properties. They will be mapped to columns of the same name in the Table. If you want to override this behavior, use the @Basic annotation. This type of intelligent behavior minimizes the work you have to do to persist a class. For our Person class example, this means that it will be persisted to the database Table named "Person," which will have three columns (id, firstName, and lastName) and the id column will be the primary key. Hibernate provides excellent control over how the primary key is assigned, but that's beyond the scope of this article.

An example of a minimally annotated POJO:

```
import javax.persistence.Entity;
import javax.persistence.Id;
@Entity
public class Person{
 @Id
 public Integer id;
 public String firstName;
 public String lastName;
}
```

An example of how a one-to-many relationship is defined:

```
  // In the Department class -
  @OneToMany(cascade=ALL,mappedBy = "department_id")// Requires the foreign key
"department_id" in the Person class
  public ArrayList<Person> getEmployees(){// Method to return all employees
 return employees;
  }

  // In the Person class
  @ManyToOne
  @JoinColumn(name = "department_id")
  public Department getDepartment(){ // Method to return the department
 return department;
  }
```

As a final time and labor saver, you can now use Hibernate's hbm2ddl tool to generate the database schema for you. While you may be splitting objects to save them in multiple tables, which is not required for the object databases, you do get very good control over the mapping process. And as we'll see later, there are also circumstances where the relational approach has advantages over a pure object approach.

DB4O

Implementing persistence with DB4O is dead easy. You don't need to create a database ahead of time or prep your Java Classes at all. Simply call Db4o.openFile() and provide the path to your database file as the parameter. If the database doesn't exist, it will be created for you. Then instantiate an Object and call db.Save(object) to persist it. It doesn't get any simpler.

```
ObjectContainer db=Db4o.openFile("C:\db4o\test.yapp");
try {
 Person person = new Person("James","Hogan");
 db.set(person); // It's now saved!
}
finally {
db.close();
}
```

Since you're storing actual Java objects, there' no mapping required. There's also no need to annotate relationships, but you must provide properties in the objects on both sides of a relationship if you want to be able to traverse the object tree bidirectionally.

```
  // In the Department class
  public ArrayList<Person> employees;

  // In the Person class
  public Department department;
```

Caché

Setting up persistence with Caché and Jalapeño falls between Hibernate and DB4O in complexity. No mapping is required, however, as InterSystems correctly points out in their Jalapeño documentation: "Databases define concepts such as constraints, relationships (with referential integrity), and indices, which have no equivalent within a Java class definition." To address and support these concepts, InterSystems' Jalapeño provides database operation-specific annotations for use in your Java Classes, much as Hibernate does. However, they are only used by the Jalapeño SchemaBuilder to create the Object Storage in the Caché database, and aren't required (other than for documentation purposes) after the Object Storage has been created.

The minimal steps for preparing to persist your objects with Jalapeño/Caché are as follows:

1) Create a Namespace and Empty Database using the Caché System Management Portal
2) Run the Jalapeño SchemaBuilder to create the Object Storage in the database

An example of a minimally annotated POJO that can be persisted using Jalapeño/Caché:

```
public class Person{
 public String firstName;
 public String lastName;
}
```

Like DB4O, since Caché stores objects, there's a one-to-one relationship between your POJOs and the Caché Object Classes defined in the database – so no mapping is required.

Note: For any non-trivial object schema you'll have to add annotations to support database specific functionality – such as relationships and indices. Annotating your Java Classes for Jalapeño works like the way it works for Hibernate. InterSystems is releasing NetBeans, Eclipse, and IntelliJ plug-ins for Caché 2007 that makes the process of adding annotations a point-and-click operation. Thus you can quickly define or modify the Object Storage in the Caché database.

An example of how a relationship is defined:

```
  // In the Department class
  @Relationship(type=RelationshipType.ONE_TO_MANY,inverseClass="Person")
  public ArrayList<Person> getEmployees(); // Method to return all employees

  // In the Person class
  @Relationship(type=RelationshipType.MANY_TO_ONE,inverseClass="Department")
  public Department getDepartment(); // Method to return the department
```

Ease of Persisting Objects

Once you have your databases created and configured, persisting your objects with any of these methods is very simple. Any of these is a distinct improvement over using JDBC/SQL and DAOs.

Hibernate

Once your Hibernate mappings are configured, persisting an object is very straightforward:

```
try{
SessionFactory factory = new Configuration().configure().buildSessionFactory();
Session session = factory.openSession();
Transaction tx = session.beginTransaction(); //optional
Person person = new Person("James","Hogan");
Long personID = (Long) session.save(person);
tx.commit();
session.close(); //optional
}catch(Exception e) {

}
```

DB4O

Once again, DB4O makes it extremely easy to persist your objects. Simply instantiate an object and then call db.set() on it.

```
ObjectContainer db=Db4o.openFile("C:\db4o\test.yapp");
try {
 Person person = new Person("James","Hogan");
 db.set(person); // It's now saved!
// commit is called implicitly when you close the container,
// so you don't really need to call it here.
db.commit(); //optional
}finally {
db.close();
}
```

Caché

Caché also makes it extremely simple to save your objects.

```
try {
/* Connect to this machine, in the SAMPLES namespace */
ObjectManager objectManager = connect (connectiontype, host, username,
password);
 Person person = new Person("James","Hogan");
      Object id = objectManager.save (person, false);
      objectManager.close ();
} catch (Exception ex) {

}
```

Ease of Retrieving Objects

Storing data is only half the equation. It must be easy to access your objects as well. Once again all three solutions simplify the process of getting an object.

Hibernate

You can use HQL or SQL to retrieve objects using Hibernate. For example:

```
package hello;
import java.util.*;
import org.hibernate.*;
import persistence.*;
public class HibernateExample {
public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();
Transaction newTransaction = session.beginTransaction();
List people = newSession.createQuery("from Person m order by m.name asc").list();
System.out.println( people.size() + " people found:" );

for ( Iterator iter = people.iterator();
iter.hasNext(); ) {
Person person = (Person) iter.next();
System.out.println( person.getName() );
}
newTransaction.commit();
```

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

3-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 25-27, 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 2007 SOAWORLD™ CONFERENCE & EXPO

2nd Annual ENTERPRISE> 2007 OPENSOURCE 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.

Enerprise 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

```
newSession.close();
// Shutting down the application
HibernateUtil.shutdown();
}
}
```

DB4O

DB4O provides three different ways to retrieve your objects: Query By Example, Native Queries, and SODA Queries. Each has its pros and cons.

The simplest and most limited is QBE. With this method, you create a prototype of the object you're looking for using one of the object's constructors, and DB4O returns the matching records.

```
// QBE Example – Retrieve Person By Name
Person proto=new Person("Ben","Franklin",0);
ObjectSet result=db.get(proto);
listResult(result);
```

Native Queries are the preferred query method and are typesafe, compile-time checked, and refactorable.

Caché

Cache provides an object oriented query mechanism that uses SQL for selection and which returns an iterator you can use to traverse the returned objects. You also have the option of using JDBC and SQL to perform complex queries over multiple related objects (SQL JOINS) or VIEWS. The results of these complex queries can also be accessed as objects, as long as the Object ID is returned as part of the query.

```
import com.intersys.pojo.ApplicationContext;
import com.intersys.pojo.ObjectManager;

try {
      ObjectManager objectManager;
String url="jdbc:Caché ://localhost:1972/" + namespace;
      objectManager = ApplicationContext.createObjectManager (url,
                  username, password);

      String sql = "Name %startsWith ?"; // Search for people by name
      if ("null".equalsIgnoreCase (query)){
            query = null;
}
      String[] qargs = {query};
      Iterator people = objectManager.openByQuery (Person.class, sql,
qargs);
      while (people.hasNext()){
      IPerson person = (Person) people.next();
            System.out.print ("Name: " + person.getName() );
 }
      objectManager.close ();
} catch (Exception ex) {
       System.out.println( "Caught exception: " + ex.getClass().getName() +
": " + ex.getMessage() );
       ex.printStackTrace();
}
```

Controlling Object Depth

One of the challenges of storing objects is to control how many objects are stored and retrieved at one time. For example, consider a company that contains 100 departments and each department contains 25 to 50 employees. Using Java Serialization, serializing the Company object would also persist all the Department and Employee objects – and instantiating the Company would also instantiate the related Department and Employee objects. If all you need to persist is the Company Object, you've done a lot of unnecessary work! There's considerable impact on performance and memory requirements for every object instantiated. Furthermore, if you send this object over the network, you want to be as efficient as possible – and therefore only send the Company Object.

What's needed is a mechanism for controlling how deeply you traverse the object tree when instantiating objects.

Hibernate

Hibernate handles this well because it's based on a relational database structure. It is easy to control the depth of the data (or the depth of the objects) returned by specifying LAZY or EAGER loading and if you switch to JDBC, you have total control using SQL JOIN statements.

When you specify LAZY loading, the POJO is actually replaced with a proxy object that will load properties and collections via the Hibernate session as needed.

DB4O

DB4O provides excellent control over object depth. You can easily specify exactly how many levels you want to retrieve or update. You can also turn on cascading updates/deletion – which traverses the entire object graph:

```
Db4o.configure().objectClass(Car.class).cascadeOnUpdate(true);
```

Or you can specify the activation depth when selecting/updating/deleting:

```
SensorReadout readout=car.getHistory();
while(readout!=null) {
db.activate(readout,2); // Activate the next 2 object levels!!!
System.out.println(readout);
readout=readout.getNext();
}
```

Thus you can specify: "Get the next three levels only" if desired or "Get everything!"

CAUTION: DB4O doesn't enforce referential integrity, so be very careful when deleting with cascade delete enabled. You can delete objects that are still pointed to by other objects in the database.

Caché

Caché provides good control over object depth as well. You can specify FetchType = Eager or Lazy like Hibernate. Calling the "detach(Object)" method ensures that all data in the given object (including all objects reachable from it by following references) can be accessed without a connection to the database.

Once again, if you switch to JDBC/SQL, you have total control over object depth via SQL JOIN statements, but can still open the objects referenced in the resultset.

Controlling Object Property Breadth

You may also want to limit how much data you retrieve from a given object. A complex object may consist of 20, 30, or even 50 or more properties, including embedded objects and lists or arrays of objects. What if you only need access to one or two of those properties? Isn't it overkill to instantiate the entire object, populating 50 properties in order to get two of them? If you're retrieving a list of such objects, you could end up with an array of 200 objects – along with all 50 of their properties – when all you need or want is one or two properties per object.

What's needed is a mechanism for controlling object property coverage.

One approach is to define a POJO that only defines a subset of the properties in the original object and populate it. This is where a relational database has an advantage over an object database. You can use JDBC to retrieve just the data you want and populate the POJO. "SELECT firstName, LastName FROM PERSON WHERE ID = 1"

Hibernate

Since Hibernate is typically backed by a relational database, it provides excellent control over your object property coverage. You can define a SELECT statement that only retrieves the properties you're interested in and provides them to you as a Java List or a List of Object Arrays.

DB4O

DB4O is a pure object database, so you must instantiate the object to access its properties. DB4O provides no help for you on this score.

Caché

Besides the Object interface that Caché provides, it also provides an SQL projection or interface. This lets you access objects as if they were tables and columns in a relational database. Using this method, Caché provides excellent control over your object property breadth.

Object Tree Traversal

Once you have your objects, you want to be able to traverse the object tree. For example, if you start with an employee, you want to be able to access the company name as follows:

```
employee.getDepartment().getCompany().getName().
```

This is one of the most powerful features of object-oriented development and one of the strongest arguments for using an object database.

Hibernate

Unfortunately relational databases provide virtually no support for this type of functionality – normally you'd have to issue SQL SELECTs to retrieve additional data as needed and create the associated POJOs. Fortunately, Hibernate provides this functionality for you by providing a proxy object to fetch additional mapped objects as needed. This works as long as you have a valid Hibernate session object available.

DB4O and Caché

Since they are object databases to begin with, DB4O and Caché handle this with aplomb. As you make calls to related objects, they're automatically retrieved from the database. Thus you can access your objects as follows:

```
employee.getDepartment().getCompany().getName();

List myEmployees = department.getEmployees();
```

And so on.

Enforcement of Referential Integrity

How can you ensure you don't delete an object that other objects depend on? For example, can you delete a department if the employees still exist? Does it support cascade deletes/updates?

Hibernate

While you can define relationships using Hibernate annotations, the

THE | POWER | TO CREATE | REALITY™

Visit us at JavaOne Booth # 1602

Make Java Fly!

DiSTI, the global leader in human machine interface development, announces GL Studio for Java.

GL Studio for Java delivers state-of-the-art, 3D, fully interactive virtual reality—*as reusable Java components.* Whether you are a web developer, a simulation-based courseware designer, or need powerful tools to create maintenance training solutions, GL Studio for Java gives you the rich, dynamic content you need to reach entirely new heights of realism and interaction.

GL Studio for Java...
Giving *you* the power to create reality.

For a free trial download,
visit ***www.simulation.com/jdj***

GL studio for Java

DiSTI

www.simulation.com

11486 Corporate Blvd. Suite 190, Orlando, FL 32817 • PH: 407-206-3390 actual support and enforcement of referential integrity is dependent on the database used on the back-end.

DB4O

DB4O currently doesn't enforce referential integrity.

Caché

Caché provides full support for maintaining referential integrity, as well as for performing cascade updates and deletes. This can be controlled based on the way you define the relationships. One-to-many relationships enforce referential integrity, but don't perform cascade updates and deletion. Parent-Child relationships enforce referential integrity and provide cascade update and deletion functionality as well.

Enforcement Of Uniqueness

How do you ensure that specific properties are unique in the database, such as Social Security numbers?

Hibernate

Support for marking properties unique and enforcing it is supported by most databases used as a back-end for Hibernate.

DB4O

DB4O currently doesn't provide any support for enforcing uniqueness. This feature is currently undergoing beta testing and the tentative release date is round mid-summer 2007.

Caché

You can mark as many properties unique as you want.

Support for Indices

Can you define indices that will enhance the performance of your queries?

Hibernate

Use the annotation @Index to cause an index to be created for the specified column or columns.

DB4O

You can define indexes in your DB4O configuration method before you open the object container. For example:

```
Db4o.configure().objectClass(Foo.class).objectField("bar").indexed(true);
```

Caché

Caché provides powerful indexing options. Besides specifying that the property must be unique, you can specify the following index types:

```
type = "" (default standard index), bitmap, bitslice, index, and key. For
example:

  @Indices({
    @Index(name="IndexOnName", columnNames={"Name"}),
    @Index(name="IndexOnSSN", type="bitmap", columnNames={"SSN"})
  })
```

BitMap indices provide extremely high performance filtering for columns that have a limited number of possible values (such as categories) or which have a fixed number of characters (such as Social Security Numbers).

Enforcement of Property Constraints

Can you limit or control the values that will be saved to the database?

Hibernate

With Hibernate, you are limited to specifying that a property be NOT NULL or UNIQUE, although you may be able to specify constraints in the underlying database.

DB4O

DB4O doesn't provide any support for constraints.

Caché

Caché provides @PropertyParameter and @PropertyParameters annotations so you can control the values entered into a property. You can specify a maximum value, minimum value, and even an input pattern.

```
@PropertyParameter (name = "PATTERN", value = "3N1\"-\"2N1\"-\"4N")
public String ssn;

@PropertyParameter (name = "MINVAL", value = "0")
public float balance;
```

Security and Access Control

How do you control who has access to the data and what they can do with it?

Hibernate

This can be performed programmatically or via the underlying database.

DB4O

You can encrypt and password-protect the database file, but there are no other user access controls. Once you have access to the database, you have access to all the data in it. If you want to control access to specific data or objects in the database, this can only be done programmatically.

Caché

This can be done programmatically or via the Caché System Management Portal. You can specify which objects the user has access to as well as the level of access (ALTER, SELECT, INSERT, UPDATE, DELETE, and REFERENCES)

Portability

How portable is the solution? Is there a vendor tie-in?

Hibernate

By definition and purpose, Hibernate helps make your application database-independent – so long as you stick to standard SQL and don't use database-specific functionality. This can be useful if you want to prototype your application on a lightweight database and move it to a more robust database later at production, however I feel that it's typically better to match your development environment to the production environment as much as possible. I've also seen very few instances where an application has been migrated to another database except in extreme legacy systems.

DB4O

With DB4O, you could say there's a vendor tie-in, but you can always add Hibernate annotations to your POJOs and run a script to read in your data from DB4O and save it to your Hibernate persistence layer.

Caché/Jalapeño

Caché stores objects using sparse arrays, so it's not your typical relational database. However, you can access data as objects or via Caché's SQL projection – which makes it look and act like a relational database (to your JDBC applications at any rate). Interestingly enough, you can also use Jalapeño to export your Caché class schema to a DDL file that can be imported into a relational database. You can then use Hibernate to map your objects to the new relational schema – or continue to use the Jalapeño Object Manager to interact with the new data source. The Object Manager automatically uses object persistence methods (Open, Save, New, Delete) when accessing Caché, and relational persistence methods (Select, Update, Insert, Delete) when it's configured to connect to a relational database.

Conclusion

While you can eliminate mapping your objects to relational tables altogether, using DB4O or Caché, for example, it appears that some work must always be done if you want to take advantage of advanced database/datastore features such as enforcing referential integrity and uniqueness.

Hibernate has come a long way since it was first released. It has a bewildering number of options for configuring your object persistence mappings and behavior – as well as great tools to make it if not painless then at least not so painful.

If you want to quickly persist your objects for a small project and you can manage uniqueness and referential integrity within your application code - look no further than DB4O. It just doesn't get any easier.

The Caché/Jalapeño combination provides a compelling option for quickly persisting your Java objects with a minimum of effort while providing excellent control over database-specific functionality.

While you were busy programming your last tour de force, your peers and technology vendors have been busy building tools that enable you to do things that were previously impossible. You owe it to yourself and to your clients to pause once in a while and survey the state-of-the-art in databases and development tools to see where new entries can save you time and effort. For a comparison of features, go to the online version of this article at http://jdj.sys-con.com.

Resources

- Comparative Study of Persistence Mechanisms for the Java Platform. http://research.sun.com/techrep/2004/abstract-136.html
- Mark Weisfeld. *The Object-Oriented Thought Process*. SAMS Publishing. This is an excellent analysis of what object-oriented design is all about and how it compares to procedural programming.

Java Serialization:
- Discover the secrets of the Java Serialization API. http://java.sun.com/developer/technicalArticles/Programming/serialization/
- Bruce Eckell. *Thinking in Java*. http://www.mindview.net/Books/TIJ/

Hibernate:
- Web site: http://www.hibernate.org/
- Dave Minter and Jeff Linwood. *Beginning Hibernate*. Apress. 2006.

InterSystems' Caché Database:
- Web site: http://www.intersystems.com/Cache
- Jalapeño: http://www.intersystems.com/Jalapeno

DB4O:
- Web site: http://www.db4o.com/
- Simple Object Persistence with the db4o Object Database. http://www.onjava.com/pub/a/onjava/2004/12/01/db4o.html

END MEDIA PLAYER FRUSTRATION...
VX30 -- THE JAVA-BASED,
UNIVERSAL STREAMING VIDEO SOLUTION!

VX30 VOTED #1 STREAMING VIDEO SOLUTION BY INDEPENDENT UNIVERSITY STUDY*

Platform Independent - Works with any combination of hardware architecture and operating system

Client Agnostic - Stream to PCs, MACs and *NIX workstations without upgrading or maintaining any client software

Cost Effective - Provides the lowest total cost of ownership of any streaming solution on the market today

Sun Partner Advantage

URL: www.VX30.com **EMAIL:** sales@VX30.com **TOLL FREE:** (866) 661-5699

VX30

*Dr. Edgar Huang - Assoc. Professor Of Informatics Indiana University-Purdue University. - Indianapolis Research Paper: "Searching For An Ideal Streaming Technology". August, 2006

Browser Wars and Swing on the Desktop

by Anthony Scotney

Is the WebRenderer Swing Edition a significant development?

The WebRenderer Swing Edition changes the face of Java Swing applications and the rendering of Web content within Java. Before we jump into that, let's take a look back at Web content display in Java desktop applications including the generational changes and Java's very own "Browser Wars."

The date is August 26, 1997. The Netscape Corporation announces[1] that they are working with Sun Microsystems to release "a 100% Pure Java version of Netscape Navigator client software by 1998." The Java community rejoices. Along comes 1998 and nothing but silence. Was it just me who missed the release of Netscape Navigator Java Edition? Who pulled the plug and why? A decent Java browser SDK back in 1998 would have gone a long way toward popularizing Java on the desktop. Instead, we're left with more than a few beers short of a six-pack.

Java on the desktop has always been somewhat lagging. First, Swing was plagued by a bad image surrounding speed. Then there was poor platform L&Fs, bugs in essential components such as the File chooser, etc., and now in the year 2007 Swing still lacks pure Swing industrial-strength Web content rendering. This fact alone has driven many Java rich-client developers to either seek out third-party components or change the direction of their desktop application development and write their rich client applications in an alternative language that supports standards-compliant rendering, or (and this is the worst) utilize non-standards-compliant Java browser SDKs and craft the Web content input around the failings of the Web content rendering engine. This paradox was always confusing. Java Web content rendering engines were written to

display Web content, but in the end the Web content had to be modified to work around the failings of the Java browser SDKs. Figure that one out!

While Web standards have improved the look and feel of Web sites through CSS and JavaScript and interactivity has increased through AJAX, Java enterprise Web applications have boomed, but Java still lacks quality Web content rendering for its Swing toolkit. How could this important detail be omitted in the decade of the Web boom?

Through all this mess, over the last decade a budding community of ISVs and large software vendors have taken the challenge with open arms and developed their own Java browser SDKs of varying architectures and with mixed results spawning their very own "Browser Wars."

First-Generation Web Content Rendering

Web content rendering in Java started life as a clean room, pure-Java implementation of an HTML parsing engine and utilized Swing drawing routines to render the graphics. The Sun default Java implementation (JEditorPane) was, and still is, basic and did not support any standards outside basic HTML, so various companies developed "commercial-strength" pure Java browser SDKs. These SDKs followed the same paradigm as the default Java implementation, but extended this to support JavaScript and other Web standards. The key limitation of pure Java browsers is they do not live up to the rendering standards delivered by Firefox and Internet Explorer.

The sheer scale of Web standards meant that small private companies in the Web content rendering space had to hire small armies to develop anything that was remotely standards compliant. Then there were issues with dynamic content such as JavaScript, and additional Web styles such as CSS 1 and 2. The task of developing a pure Java browser was and still is

Third-Generation Web Content Rendering

The WebRenderer Swing Edition is a pure Swing Java browser SDK based upon native parsing and layout of the Mozilla engine (the exact same parsing and layout engine as utilized by Firefox). What does this mean? It means that now Java Swing applications will have commercial-strength Web content rendering in pure Swing. Every line, polygon, image, text, and more will be drawn in pure Swing. The WebRenderer Swing Edition has removed the heavyweight rendering of the native layout and rendering engine. This will provide Java Swing applications with Swing integration and industry-standard rendering.

The rendering quality of Mozilla is paired with the lightweight drawing of Swing, including support for Java Look and Feels (L&F). This means seamless integration into Swing applications with the standards compliance of Mozilla. No longer is there a great divide between Swing and real-world commercial strength Web content rendering. WebRenderer Swing Edition is the first of the third generation of Java browser SDKs. Not only is HTML, CSS 1 & 2, JavaScript (including AJAX) supported, but Java applets and third-party plugins such as Flash are supported. Enterprise technology mash-ups including Java, Web content, AJAX, Flash, and Swing are now possible in a single Java Swing application.

Anthony Scotney is the founder and CEO of JadeLiquid Software. With Anthony at the helm JadeLiquid Software has grown exponentially since inception and received several awards in recognition of its success. He was awarded the prestigious 2004 Pearcey Award for individual pioneering achievement and contributions to research and development in information technology. In 2005 he was named Tasmanian Young Achiever of the year. Anthony studied science at the University of Tasmania, majoring in computer science.

ascotney@jadeliquid.com

"Web content in Java started life as a clean room, pure-Java implementation of an HTML parsing engine and utilized Swing drawing routines to render the graphics"

monumental. A standards-compliant pure Java browser is not a commercially viable prospect. The first-generation Java browsers ended up with a lightweight implementation that often "broke" on real-world Web content. Due to this fact, first-generation Java browsers are slowly dying out with the major vendors withdrawing support, paving the way for a new wave of Java browser SDKs.

Second-Generation Web Content Rendering

This first round of Web content rendering engines can be referred to as first-generation rendering engines. The next step in the development of Java browser SDKs was brought to life by JadeLiquid Software, which developed the WebRenderer Desktop Edition, a commercial-strength, native-based Java browser SDK. The release of the first second-generation, Web content rendering engine occurred back in 2002 with the release of the WebRenderer Desktop Edition. The development affected the face of Web content display in Java by bringing the market-leading native browsers (Mozilla, Internet Explorer, and Safari) through to the Java environment for use in Java rich client applications. Many copycat developments, including some from the Java industry giants, have followed and the trend for Java browsers has headed toward the architecture pioneered through the WebRenderer Desktop Edition. The key limitation of second-generation native Java browser SDKs is that they are heavyweight. Being heavyweight, on many occasions they don't play well with the Swing toolkit. While the rendering is phenomenal and standards compliant, Swing integration can be somewhat tricky. Standard JFrames, JPanels, etc., are fine. The integration issues rear their head with JInternalFrames, JTabbedPanes, Glasspanes, JMenus, and the like. The heavyweight–lightweight mixing issue has always created issues for Swing. The WebRenderer Desktop Edition provides a compatibility layer to help with Swing integration but, by design, lightweight and heavyweight components don't mix well together. Despite the integration issues in the commercial space, the second generation of Java browser SDKs have proven themselves as the architecture of choice for commercial grade Web content rendering. Now that you have the history, where to from here? Well, JadeLiquid, the developer of the WebRenderer family of products, has just released a new product that will change the Java browser SDK space and create the third generation.

The Final Frontier

Now that the Java "Browser Wars" are moving toward a conclusion, with the native architecture base winning out for enterprise desktop applications, it will be interesting to watch this space and see what companies, such as Sun Microsystems, do to embrace or "reinvent the wheel" in the eye of the third-generation Java browser SDK, WebRenderer Swing Edition. Sun's latest attempt at building a Java browser SDK (the JDIC program) has been flailing with mixed results. The architecture direction is blatantly wrong and major issues are a dime a dozen. Through the architecture of JDIC Swing compatibility is limited (see second generation) and with time the second generation is destined to end up like the first generation. Word from the Java rumor mill is that Sun's JEditorPane is undergoing some tweaking. Without years of development and a team larger than the entire J2SE team, this is a losing and some would say pointless battle. At best some JEditorPane revisions will result in yet another substandard first-generation browser component or at worst a better version of a broken wheel.

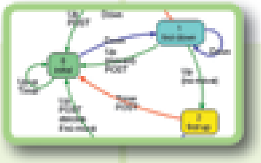
Is WebRenderer Swing Edition the only commercial strength option for "real-world" Web content support within Java Swing applications? You decide.

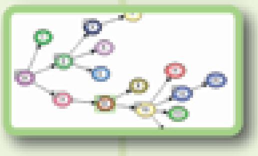
References

- http://web.archive.org/web/20031014110817/http://java.sun.com/pr/1997/august/pr97826-02.html
- http://www.webrenderer.com

Build interactive diagrams easier than you ever imagined

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





FREE Download With Full Support!

GoDiagram

Interactive diagram components

www.nwoods.com

2007 VIRTUALIZATION CONFERENCE + EXPO
www.virtualizationconference.com

Register Now!
EARLY-BIRD SAVINGS!
$200

ROOSEVELT HOTEL
NEWYORK CITY
June 25-27, 2007

Virtualization: Solutions for the Enterprise.

Delivering The #1 i-Technology Educational and Networking Event of the Year!

As today's CIOs and IT managers rise to the challenge of using their enterprise computing infrastructure as cost-effectively as possible while remaining responsive in supporting new business initiatives and flexible in adapting to organizational changes, Virtualization has become more and more important.

A fundamental technological innovation that allows skilled IT managers to deploy creative solutions to such business challenges, Virtualization is a methodology whose time has come. The fast-emerging age of Grid Computing is enabling the virtualization of distributed computing, of IT resources such as storage, bandwidth, and CPU cycles.

But Virtualization can also apply to a range of system layers, including hardware-level virtualization, operating system level virtualization, and high-level language virtual machines.

Register Online!
www.VirtualizationConference.com

THE FIRST MAJOR VIRTUALIZATION EVENT IN THE WORLD!

HURRY, FOR EXHIBITOR AND SPONSORSHIP OPPORTUNITIES
CALL 201-802-3020

— BROUGHT TO YOU BY —
ENTERPRISE OpenSource MAGAZINE   SOAWORLD MAGAZINE

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

JAVA DEVELOPER'S JOURNAL Advertiser Index

Advertiser	URL	Phone	Page
Altova	www.altova.com	978-816-1600	Cover II
DiSTI	www.simulation.com/jdj	407-206-3390	27
ESRI	www.esri.com/develop	888-288-1277	19
ILOG	www.ilog.com/jdj/ajax		4
Infragistics	www.infragistics.com/jsf	800-231-8588	Cover IV
InterSystems	www.intersystems.com/javaone2007p		7
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	33
JadeLiquid	www.webrenderer.com		11
JavaOne	www.java.sun.com/javaone		21
Jinfonet Software	www.jinfonet.com/j	240-477-1000	17
Northwoods Software Corp.	www.nwoods.com	800-434-9820	31
Perforce Software	www.perforce.com		15
SOA and EOS 2007 Conference & Expo	www.soaworld2007.com	201-802-3020	25
Software FX	www.softwarefx.com	561-999-8888	Cover III
Sun Microsystems	www.netbeans.org		9
Virtualization Conference & Expo 2007	www.virtualizationconference.com	201-802-3020	32
VX30	www.vx30.com	866-661-5699	29

Getting Ready to Choose the Year's Best

Onno Kluyt

Every year the process of choosing the community's best starts with nominations in five categories: Member of the Year, Most Outstanding Spec Lead for Java Standard Edition/Enterprise Edition, Most Outstanding Spec Lead for Java Micro Edition, Most Innovative JSR for Java Standard Edition/Enterprise Edition, and Most Innovative JSR for Java Micro Edition. This year the JCP adds a new one: JCP Participant of the Year. At the time of writing the JCP Executive Committees (EC) representatives selected three to four nominees in each category and have another 10 days to vote for the winners. It's become a tradition to announce them at the community event the JCP organizes at JavaOne, which this year will host the fifth edition of the JCP Annual Awards.

And the Nominees Are!

JCP Member of the Year

With this award the JCP recognizes the corporate or individual member who has made the most significant positive impact on the community in the past year. When choosing nominees, the EC members look at leadership qualities, breadth and depth of effort put in the community, and innovation contributions. The nominees in this category are the Apache Software Foundation, Nokia, and Orange France. The Apache Software Foundation will present at JavaOne on the topic of The Apache Harmony Project and will join Sun, Red Hat, the Free Software Foundation, and Max Planck Institute for Computer Science on the Java Technology Libre Panel. As usual, Nokia has a strong presence at JavaOne, presenting and co-presenting in at least 10 technical sessions and BOFs combined. Orange presents in two technical sessions, one of which is about "Tackling Java ME Device Fragmentation: Orange and Sun Collaboration."

JCP Participant of the Year, New Category in 2007

This award rewards the corporate or individual member participant (individual name) who has made the most significant positive impact on the community in the past year. Leadership, technical contribution, and innovation are some of the qualities that EC members look for in voting for this award.

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

onno@jcp.org

The nominees are Wayne Carr, Jean-Marie Dautelle, and Doug Lea. For brief bios of the three nominees go to http://jcp.org/en/participation/committee and search for their names under the respective Executive Committees. You can also catch Jean Marie Dautelle at JavaOne on the panel of the Java ME BOF-5697, Take the Guessing out of the Java Platform, Micro Edition (Java ME) Future: Latest JSRs Predict Exciting Technology Developments Ahead.

Most Outstanding Spec Lead for Java SE/EE

This award goes to the person who has brought together such qualities as technical savvy, the ability to build consensus in spite of diverse corporate goals, and focus on efficiency and execution. The nominees this year are Alan Bateman (JSR 203), Nasir Khan (JSR 289), and David Nuescheler (JSR 283). If you want to listen to Alan Bateman at JavaOne, make note of his BOF Troubleshooting and Diagnostic Utilities in JDK Release 5 and 6. To find out more about the work these Spec Leads put into developing these JSRs, go to the public pages of these projects at http://jcp.org, and do a search for the JSR number.

Most Outstanding Spec Lead for Java ME

Similarly, the EC members look for the same strong qualities when they choose nominees for this category. This year the nominees for the most outstanding Spec Lead for Java ME are Shai Gotlib (JSR 190), Mike Milikich (JSR 271), and Antti Rantalahti and Ivan Wong (JSR 272). In September 2006 Shai Gotlib interviewed with Artima Developer; to check out what he had to say about the API he's driving, go to http://www.artima.com/lejava/articles/mobile_events.html.

Most Innovative JSR for Java SE/EE

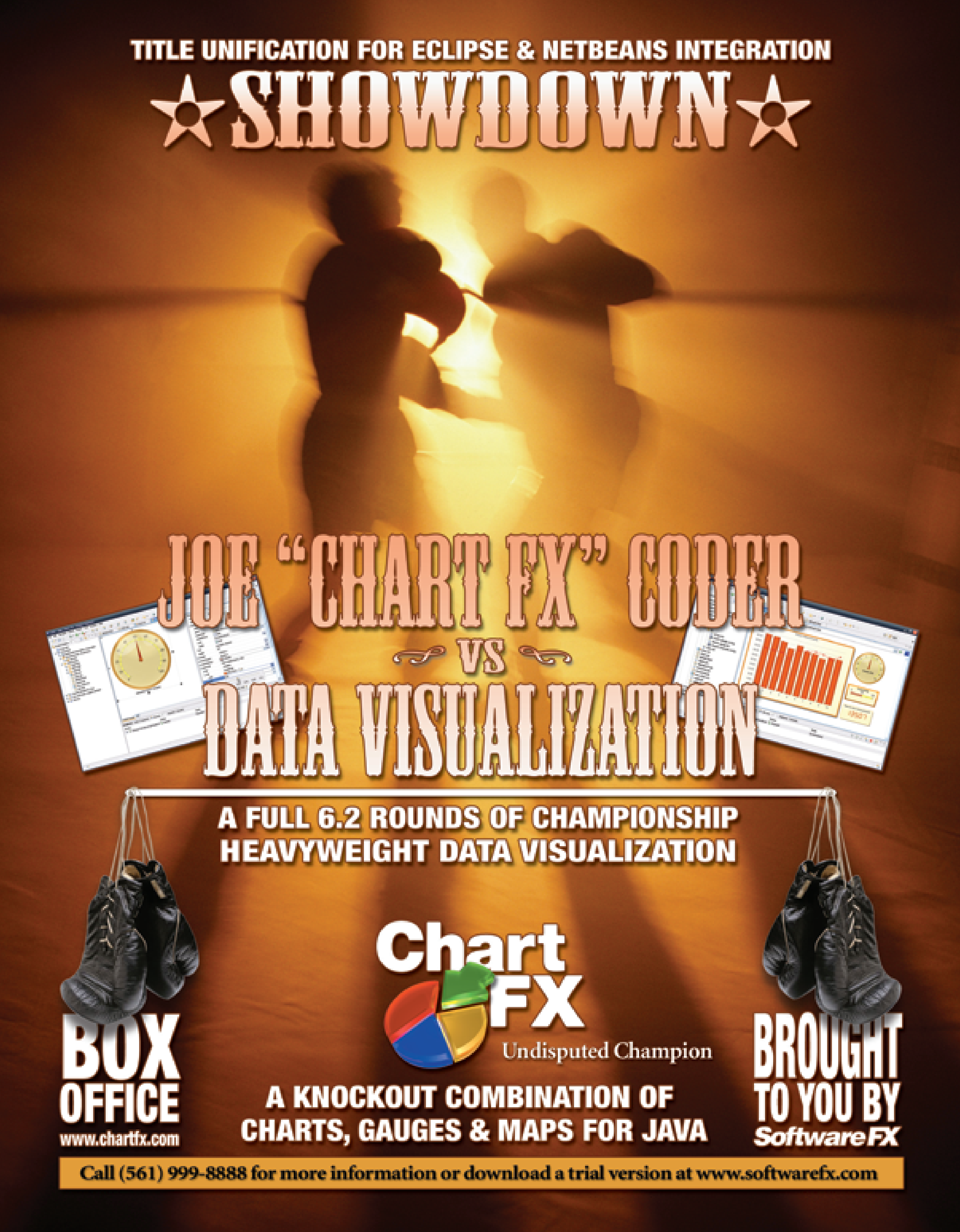
Innovation is key to the success of the JCP program and helps ensure that the JCP remains a fresh and vibrant community. This award recognizes the Spec Lead and Expert Group that have introduced the most innovative new JSR for the Java SE or Java EE communities in the past year. The candidates put forward are JSR 299, Web Beans; JSR 308, Annotations on Java Types; and JSR 309, Media Server Control API. If you're looking for a presentation on JSR 299, Web Beans, bookmark the technical session TS-4089, Web Beans Update, given by Gavin King, JBoss, and Bob Lee, Google. For more information on the JSRs I recommend you go to the respective public pages at http://jcp.org.

Most Innovative JSR for Java ME

A similar award is offered for Java ME for which innovation is as important as for Java SE/EE. The JSRs nominated by the EC members this year are JSR 298, Telematics API for Java ME; JSR 300, DRM API for Java ME; and JSR 307, Network Mobility and Mobile Data API.

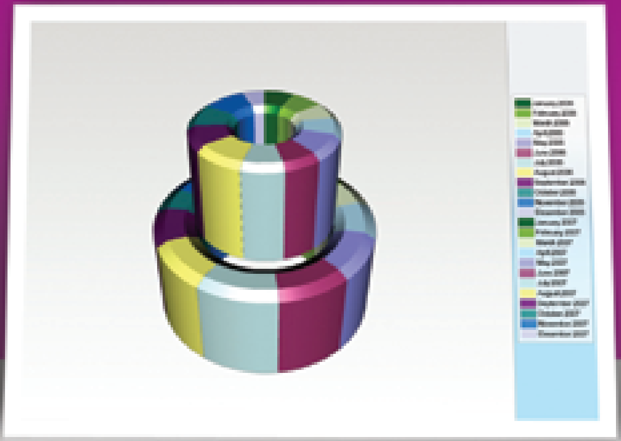
The grand finale is scheduled to take place at the community event at JavaOne organized by the JCP on May 9. The winners will be officially announced at the Fifth JCP Annual Awards ceremony. Don't miss this community event that the JCP brings to JavaOne every year. If you can't make it to the ceremony, check the next JSR column; I'll be introducing the winners to you. If you are interested in more JCP events at JavaOne, check out the JCP Events Calendar at http://jcp.org/en/whatsnew/calendar.

For members of the press and analysts the JCP organizes a round table on Wednesday, May 9, The Java Standards Advantage, from 3:15 p.m. – 4:15 p.m. at the Moscone Conference Center, Room 123. Participants include some of the nominees for the Fifth JCP Annual Awards. (journalists or analysts, send your request for round table details to corina@jcp.org).

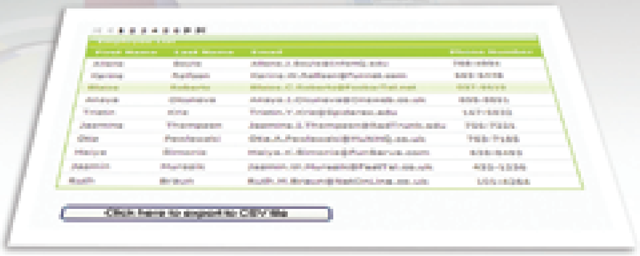

TITLE UNIFICATION FOR ECLIPSE & NETBEANS INTEGRATION
SHOWDOWN
JOE "CHART FX" CODER
VS
DATA VISUALIZATION
A FULL 6.2 ROUNDS OF CHAMPIONSHIP HEAVYWEIGHT DATA VISUALIZATION
Chart FX
Undisputed Champion
BOX OFFICE
www.chartfx.com
A KNOCKOUT COMBINATION OF CHARTS, GAUGES & MAPS FOR JAVA
BROUGHT TO YOU BY
SoftwareFX
Call (561) 999-8888 for more information or download a trial version at www.softwarefx.com

"I bring meaning and comprehension to my corporate data using advanced visualization tools."

- Application Developer



WebChart™ - 3D Overlay Donut Chart



WebGrid™ Export to CSV Format

NetAdvantage® for JSF 2007 Volume 1

Consistent Multi-Platform User Experience

WebCharting – Add high impact 2D/3D visualizations with over 70 different chart types and views

Improved Data Reporting – Easily export data within a grid to any application that supports the import of CSV such as spreadsheets and databases

Application Performance – Leverage our AJAX framework to turbo-charge your Web applications

Accessibility – Support of US Code Section 508 across all Infragistics controls

learn more: infragistics.com/jsf

Infragistics®

Powering The Presentation Layer

Infragistics Sales - 800 231 8588

Infragistics Europe Sales - +44 (0) 800 298 9055

WINDOWS® FORMS ASP.NET WPF JSF

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