Java One 2007 Conference Notes

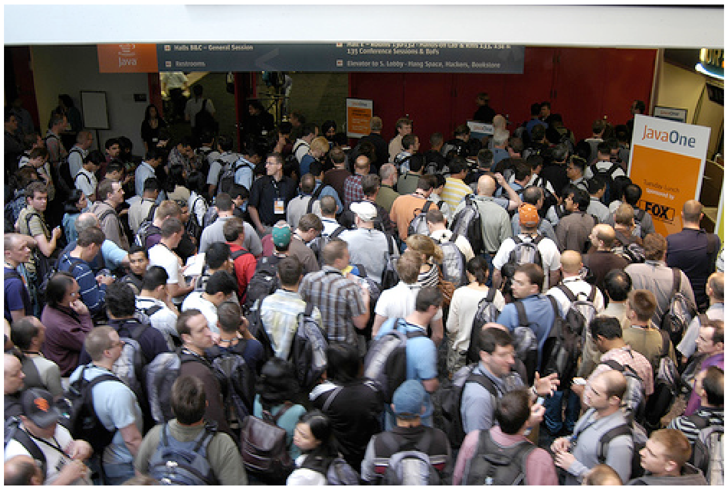
By

Juixe

Monday, May 7

Welcome to CommunityOne 2007

For several years now, Sun has put on NetBeans Day one day prior the start of JavaOne. This year, they rebranded NetBeans Day and its spin off GlassFish Day into CommunityOne. CommunityOne is an open and free event, a mini-JavaOne if you will. This year, the featured speaker for the General Session was Tim O'Reilly. Rich Green, Vice President of the Software division at Sun, welcomed the attentive audience by talking about the new open source culture at Sun. Rich said, "Simplicity and access is far more important than technological perfection." After a brief introduction Rich welcomed Tim O'Reilly to the stage.



Tim started by asking the crowd, "How many of you use linux? How many of you use Google?" Tim feels that Google, Craigslist, Yahoo, Ebay, Amazon, and all those web applications share a common thread. He feels that they are "data aggregators, not packaged software." They are not so much important as applications but more so as services. Just having the source code to Google's Page Rank algorithm does not mean you will replicate the success of Google. These services use the network effects of User Generated Content to gain market share. Tim reminded the audience that value has moved from hardware to software, and how it is now moving from software to services. But with these services end users aren't just entities that mindlessly contribute content, they become co-contributers and this has been made self-evident by the recent Digg Revolt. Digg users went up in arms regarding the censorship of stories listing a HD DVD key.

Tim O'Reilly said that Web 2.0 is about bionic software powered by people. Digg, Flickr, Delicious, and other Web 2.0 social-based sites are powered by people and leverage an architecture of participation as described in The Cornucopia of the Commons.

In describing the architecture of participation, Tim noted the tendency for web services to be stuck in perpetual beta. Microsoft might have a better idea of perpetual beta when they think of their online servies as live software. Online web applications grow and evolve organically. Small and agile teams are more able to evolve software organically. Tim said that Amazon has a rule of thumb for limiting the size of a team to the number of techies that can be feed by two pizzas. Although not described as such, Google uses two pizza teams on their products such as Google Calendar.

There are two other comments made by Tim that I though interesting. The first comment I wanted to capture was that "we are coming to the end of cheap outpouring." Then he quoted a colleague that you can make money on the long tail, but not in the long tail.

Getting Started and What's New in GlassFish v2

The first technical session I attended at CommunityOne 2007 was about GlassFish, the JEE5 reference implementation for JSR 244. According to the session speaker, GlassFish is more than a reference implementation. The value of GlassFish is that it is open source, community driven, and production quality JEE5 Application Server. JEE5 is a simplification of J2EE, the previous incarnation of the Java Enterprise Edition. JEE5 simplifies enterprise development by leveraging Plain Old Java Objects (POJO) described with annotations. Annotations are first class language construct in java 5.

The current release of GlashFish, v2, provides better performance, faster startup time, out of the box load balancing, cluster management, and built-in fail-over. Under the covers, the GlassFish Application Sever is powered by Grizzly, EJB 3.0, and JPA. GlassFish is really a community effort with many supporting side projects such as jMaki, Slynkr, Blogapps, and Phobos.

Lunch with the Java Posse

The JavaPosse podcast is hosted by Tor Norbye, of Sun, Carl Quinn and Dick Wall, of Google, Joe Nuxoll, of Apple. The regular format for the show is for the posse to run down trough and pontificate the current news in the Java world. For this live recording of the Java Posse podcast the posse adopted the Top Ten count down made famous by David Letterman. Some of the posse's top predictions for JavaOne 2007 included mobile mayhem, desktop Java push, and that applets will no longer suck.

According to the posse, the top missing features in Java include the missing LINQ, cross platform solutions for recording and playing back audio, device support for USB and FireWire, browser level Swing HTML rendering, language level properties and events, component model for the desktop, component model for the web, component model for the mobile, and more importantly a consistent component model for all. Additional missing features in the language include true generics, regexp literals ala Groovy, multiple return values or parallel assignment, closures, @Nullable annotation, type inference ala Scala, invoke dynamic, and modules and super packages.

The posse then listed the top APIs to nuke from Java which include anything related to date and calendar, java.awt.List, java.awt.*, java.beans.*, java.swing.LookAndFeel, java.util.Stack, java.util.hashtable, the close method if it throws an exception, java.io.File since it is just a path, cloneable, binary serialization, Enumeration, and CORBA.

The top software industry pet peeves according to the Posse include the not invented here syndrome, paper architects (if you can't code, retire), developers with nothing left to learn, IDE zealots, framework zealots, snap judgment of technology, frivolous patentry, lack of User Interface sex appeal, coding styles, meetings, tHE cAPS lOCK mUST dIE, unskippable intros, and general lack of manners.

Here are some memorable quotes from the posse members.

The definition of Hell is working with dates in Java, JDBC, and Oracle. Every single one of them screw it up. -- Dick Wall

This project is running late, quick lets have a meeting. -- Dick Wall

The recommendation I would give you is not to use Swing, but to be using a rich application platform like NetBeans. -- Tor Norbye

Ajax Applications Made Easy with jMaki and Scripting

This technical session at CommunityOne 2007 introduced me jMaki, a project under the GlassFish umbrella. Project jMaki is a lightweight AJAX framework that wraps JavaScript libraries and in such provides a common interface to any and all JavaScript code. jMaki proposes a convention, over configuration, to wrap any third-party or custom JavaScript library as a jMaki widget. In other words, jMaki is a widget model for creating, using, and wrapping existing JavaScript functionality. jMaki is also available as a helper in Rails, and as a function in PHP, as well as a component in JSF.

Out of the box, jMaki comes with layout and theme support. In addition to layout, jMaki has a XMLHttpProxy (XHP) feature that allows you to bypass the pesky same-origin policy restriction.

Should you use jMaki? Yes. jMaki does all the hard work so that you can mix YUI!, Scriptaculous, Spry, and any other Ajax library together seamlessly. jMaki provides visual development support with plugins for Eclipse and NetBeans. jMaki also provides a common widget model for JavaScript libraries so that you don't have to learn each more than you have to to work with any one of them.

Swing GUI Building with Matisse: Chapter II

This session was presented by Joshua Marinacci, Shannon Hickey, Hans Muller and other Sun technical leads behind NetBeans, Swing Application Framework, and the Beans Binding specification. This session was practically a demo of NetBeans GUI Builder improvements and support for JSR 296: Swing Application Framework and JSR 295: Beans Binding.

The logic behind the Swing Application Framework is to provide a framework to help novice swing developers get up and running. The desktop client API stack is pretty large and intimidating when starting a Swing application from scratch.

Swing Application Framework can be explained in an hour, and is small and simple enough for small to medium size applications.

The demo demonstrated how to use GUI Builder to construct a Swing Application Framework-based application mashup that downloads images from Flickr based on a given search criteria. They tied the model with the UI with Beans Binding. Beans Binding keeps two properties from two objects in sync. Shannon Hickey, the specification lead of JSR 295 said that "Beans Binding is going to be great for Swing when finished." Beans Binding describes the source properties using Java Expression Language. The Beans Binding will accomodate objects that don't follow the beans pattern, such as maps which will be treated as dynamic beans.

These guys built a Flickr client using NetBeans, Data Binding, Swing Application Framework in four minutes flat.

JRuby: Understanding the Fuss

Fellow Sun engineers Tor Norbye and Charles Nutter presented on the past, present, and future fuss of JRuby at CommunityOne 2007. To understand where JRuby is heading, the presenters painted a clear picture of the Ruby programming language. Ruby is a dynamically typed pure object-oriented language originally written by Yukihiro 'Matz' Matsumoto. The Ruby programming language is an expressive, powerful, and easy to learn, read, and write.

Tor and Charles gave a quick tour of the Ruby language features such as string substitution, modules and mixins, open classes, meta-programming, duck typing, literals for arrays and maps, and code blocks. Ruby is an incredible flexible and expressive language. The more expressive the language, the less code you'll end up writing and maintaining.

Tor also demoed his work with NetBeans support for JRuby. NetBeans includes code highlighting, code completion, and documentation support for JRuby. While demoing NetBeans and thinking of JPA Tor Norbye said, "I wish they had annotations in this language."

Up the Stack

The last session of CommunityOne 2007 was titled Up the Stack. The main concern of this session was the whole software development stack, from the Operating System to the database and all the way to the web framework. For the most part, developers these days working on the next great Web 2.0 pay no heed to limitations and strengths of the OS.

This session dealt with performance and profiling considerations spanning the whole stack from using DTrace on Solaris to using caching in your application. Dtrace is dynamic tracing utility made available in Solaris that will help you discover bottlenecks in your application by analyzing the whole process.

Another session dealt with GlassFish. GlassFish is an enterprise ready Java EE 5 Application Server with easy management tools and clustering support. GlassFish supports RIFE, Rails, Struts, Wicket and just about every other Java web application framework under the sun, no pun intended.

Tim Bray of Sun moved up the stack and talked about web technologies such as PHP and Rails. Tim stated that PHP is easy to learn and quick to develop with. PHP has a share nothing architecture that is great for scaling but is historically known for the tons of security holes, SQL injection, and cross-site scripting attacks. Tim noted that some developers would trade the security of JEE for the speed of development of PHP to get first to market. Tim also mentioned that Don't Repeat Yourself, Convention Over Configuration and the expressiveness of Ruby while talking about Rails. The Ruby programming language allows for Rapid Agile Development. The big knock against Rails is its lack luster performance and it's multi-headed mongrel deployment story. PHP or Rails are a good solution for many of the CRUD applications that babysit a database.

The last session of the talked compared Ehcache and memcached. Memchached is said to be used in LiveJournal and Slashdot as well as many Ruby on Rails applications. Ehcache distributed peer-based caching in Java sync/async operations used in Spring and Hibernate.

G2One

The G2One, Groovy and Grails mini-pre-JavaOne conference, was put together by the folks behind No Fluff, Just Stuff conference. The G2One mini-conference brought together Guillaume LaForge, the Groovy project lead, Graeme Rocher, Grails project lead, and Groovy in Action author Dierk Koenig amongst other Groovy developers and evangelist.

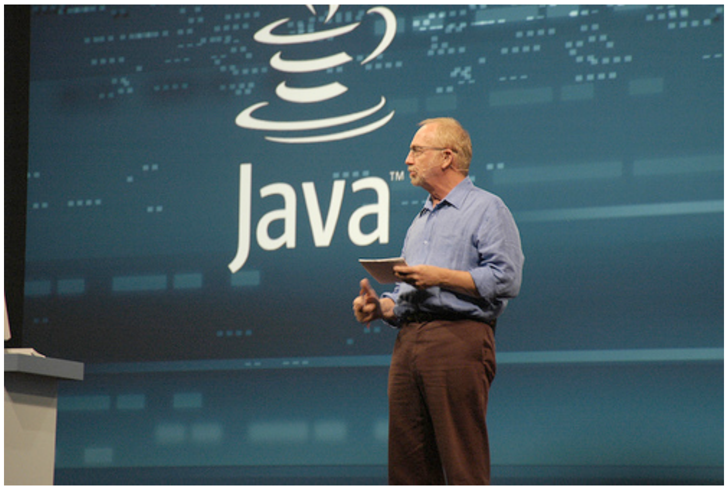
Guillaume started the event by proving a quick introduction of Groovy and going into the details of the upcoming release. Groovy provides semantic sugar and sweetness to the Java VM. Groovy is Java-like and compiles down to Java to byte code but with the added bonus of the GDK, the Groovy Development Kit. You can mix and match Java/Groovy classes seamlessly, no bridge/connector required. I think that Groovy is what Java 3 should and might be. For example, Groovy provides string substitution, array and map literals, and regex literals. There are Eclipese plugins for Groovy that come with code completion, navigation, and highlighting. Groovy 1.1-beta has support for annotations.

Graeme talked about the currently available Grails 0.5 release. Grails builds on top of Groovy as the language used in the models and controllers, Spring for the Inversion of Control container, Hibernate for the Object Relational Mapping, Quartz for task scheduler, and Sitemesh for templeting. Grails comes with an embedded Jetty servlet container and embedded HSQLDB database for quick turn around agile development. Graeme stated that most of the behavior in Grails is provided by plugins. So what is new in Grails 0.5? Command Objects in controllers. List and map support in GORM. Cascading validation. Script event hooks. Converter plugin to automatic convert models too XML, JSON, or RSS.

There was a series of rapid-fire demonstrations by Guillaume, Graeme, Dierk, and other Groovy developers. One demo used the SwingBuilder groovy class to mashup GMaps and Flickr data. The demo used XMLSlurp to consume XML RSS feeds. Remixing and mashing up Google Maps and Flickr is like the 'Hello, World' first program of Web 2.0 mashups. When demoing the Swing-based mashup, Dierk Koenig said of the aesthetics of his demo, "This is what happens when you program in Swing, the first attempt is usually ugly."

Tuesday, May 8

Tuesday General Session



John Gage, Sun co-founder and Chief Researcher, kicked off JavaOne 2007 by saying that there were 81-hours until the end of JavaOne. He recommended that we all take the time and meet the fellow developers and engineers sitting to each other at each session. Engineers, as a breed, tend to be introverts but John reminded us to not be shy. He recommended that you forget that you are Swedish; forget you are British, polite, and reserved. He said, "For the next 81 hours you are Brazilian." To this declaration, the Brazilian contingent raised the roof with their cheers. John said that in that conference hall room are all the authors of every Java book, leader of many Open Source projects, and innovators in just about every front. He recommended that we all take advantage of that fact.

The big announcement at JavaOne made by Sun have has JavaFX Script, formerly known as Form Follows Function (F3). I think of JavaFX Script as a scripting Domain Specific Language for Rich Internet Applications which begs the question, why a new language? Why not accomplish the same functionality using Groovy, JRuby, or a unified and simplified API? Recently Sun also made inroads into the mobile space by purchasing the SavaJe mobile phone technology. At JavaOne, Sun announced JavaFX Mobile. JavaFX Mobile is an iPhone-inspired Java-powered software system for mobile devices. Rich Green, Vice President of the Software division at Sun, said that JavaFX Mobile is some "serious mobile eye candy."

The general session is also meant to inspire all those present to share ideas, contribute code, and to participate. To this end, Rich Green said, "This is not a read-only life style." Scott McNealy, former CEO of Sun, made a brief appearance on stage to announce his current project, Curriki, an elementary to high school curriculum wiki. Scott McNealy also had time to joke that Rich Green is a 'short sleeve version of Steve Jobs.'

JRuby on Rails - Agility for the Enterprise

Charles Nutter and Thomas Enebo, Sun employees and key contributors to the JRuby project, gave the typical introduction into the Ruby programming language, JRuby implementation, and JRuby on Rails. Ruby is a pure Object-Oriented programming language with plenty syntactic sugar for arrays, maps, regular expressions, anonymous blocks, duck typing, and meta-programming. JRuby is a first class citizen in the JVM.

A common theme by the scripting folks, is their common dislike for regular expression in Java and how the leverage the libraries already available in Java. JRuby, in short combines the libraries and power of Java with the syntax and expressiveness of Ruby.

Charles stated that the benefits of JRuby over Ruby include better scaling, will soon be faster, native unicode support, Java libraries, easy adoption in the enterprise environment and existing applications. The benefits of JRuby over Java include Ruby language features and Ruby applications like Rails, Rake, Rave, and RSpec.

Charles Nutter, the lead on JRuby did state the the current focus is 'compatibility over performance.' By this, his focus is for JRuby to be equivalent to the C implementation of Ruby 1.8. Ruby does not have an official language specification; the C implementation is as close as you get to a language spec. Charles Nutter has stated that the lack of an official specification as a pain point in the development in JRuby.

There are some JRuby extras on RubyForge, include a Web Application Resource plugin recently renamed to Goldspike. Originally named Rails Intregation, the Goldspike plugin allows you to create a WAR file for a JRuby on Rails project, which can be deployed on Tomcat, Jetty, or any other Servlet container. Other interesting projects include ActiveRecord-JDBC and a Java port of RMagick, the popular image package used in Rails applications.

The speakers also mentioned on the precepts and philosophy behind Rails such as Convention over Configuration, Don't Repeat Yourself (DRY), and agile development. The typical quote around the JRuby on Rails peeps is that there is "less Rails code than Java application configuration."

There was a question from an audience member about how JRails compares with Grails. The JRuby folks feel that Ruby on Rails currently has more books, more focus, and more developers. Grails was heavily inspired from Rails, and both projects have borrowed ideas from each other. I personally strongly recommend either Grails, and JRuby on Rails over some of the other available web application frameworks.

Evolutionary Java - General Session

Evolutionary Java was a general session held at noon on the first day of JavaOne 2007. This general session delved into some proposed features for Java SE 7. Some proposed features for Java SE 7 include modularization which will introduce the concept of super package, new byte code and careful evolution of the Java language, and better support for multiple and dynamic languages in the JVM.

During this session there where more demos including one of JRuby on Rails running a standard Rails application, Mefisto. Tor Norbye gave a demo of the new JRuby/Rails support in a beta version of NetBeans with code completion, code highligthing, and wizards for creating new projects. There was a great demo of GL Studio for Java. Basically the demo mashed up Google Earth-like earth simulation provided by NASA Ames and DiSTI 3D models of a realistic F-16 jet fighter. One of the best demos was Iris, a flickr mashup with some great eye candy.

There was an additional demo and information on JavaFX Script (JFX). JavaFX Script is an object-oriented programming language with a declarative syntax. Think of JFX as a mini-DSL for effects, animations, and rollovers for the JVM. JFX is a programming language especially designed for effects and animations. It is interesting to note that most of the demos of JFX are ports of Flash applications. A common question amongst those present was, why introduce another language? Why not just a Java-based library?

Java Puzzlers

The complete name of this JavaOne 2007 session was Java Puzzlers, Episode VI: The Phantom-Reference Menace/Attack of the Clone/Revenge of the Shift. This technical session was presented by perhaps the most effective Java developer Joshua Bloch and fellow puzzler William Pugh. Joshua and William presented 8 short programs with curious behavior. This session was basically a 'what does this program print?' interactive question and answer discussion.

The puzzles where like Google interview questions, a bunch of brain teasers which intended to preach a few moral and teach a few tips regarding interesting behavior at the language and JDK level. For example here are some tips I took away from the session, use URI instead of URL because the hashCode and equals method in URL is broken. Don't mix short, long, int primitives and objects because autoboxing with knock you out. Autoboxing happens, when you least expect it. JUnit does not support concurrency. Wrapped primitives aren't primitives. Watch out for circular class initialization when you mix statics and constructors, remember static code is process from top to bottom before constructors. If an API is broken, wrap it. For API designers, don't violate the principle of least astonishment. Math.abs doesn't guarantee non-negative results, for example Math.abs(Integer.MIN_VALUE) because Integer.MIN_VALUE == -1*Integer.MIN_VALUE. Avoid mixing types in the ?: ternary operator. Use find bugs, acording to Joshua and William Find bugs warned all of the questionable behavior.

Lastly, William said that the "Collections API where designed for what is type-safe, not what is sensible."

Using jMaki in a Visual Development Environment

The jMaki project started as a wrapping utility for JS libraries and widgets. The j stands for JavaScript and maki comes from the Japanese word for wrapping, maku. The intent of jMaki is to promote clean seperation between content, style, and javascript and do so in an abstraction layer that lends itself well to a component model. jMaki is a client-server framework with support in PHP, Rails, or Java web applications.

jMaki has nice IDE support in NetBeans with plenty of wizards to get you started. Since jMaki provides a nice component model you can use the visual editor to wire together the UI of a web application in your IDE. Since a big part of web application development is getting the layout right, jMaki comes with several layouts right out of the box, all you do is select the template you want to use from the IDE.

To wrap a custom JavaScript library with jMaki you will need to create a jMaki widget. A jMaki widget uses the convention, instead of configuration, which is basically a folder containing three files, a HTML template, a CSS style, and JavaScript behavior. jMaki already has support for popular JavaScript libraries such as ExtJS, YUI!, Dojo, etc.

Greg Murray stated that he developed jMaki because he "wanted the reload button to be the redeploy button." Well, since HTML, CSS, and JS aren't compiled, that is easy to do. One button redeploy is not the stregth of jMaki, what jMaki has going for itself is that you can visual develop a web application using NetBeans and that this wraps several libraries in a easy to use fashion. jMaki also allows you to mix and match different JavaScript libraries together and communicate with each other using Glue. jMaki provides a common ground for integrating multiple of the currently existing AJAX libraries.

There is a jMaki on Rails plugin available and a PHP library.

Java Persistence API - Best Practices and Tips

This JavaOne 2007 session was packed the rim but I soon discovered I should have gone elsewhere. Everything in Java has some type of context and JPA is no different, the speaker was talking about the Persistence Context, blah, blah, Entity Manager, something or other. I found this session a bit hard to follow, a bit too technical and really dry. The slides looked like endless paragraphs from Tolstoy's The Death of Ivan Ilyich. A JPA junkie snapped pictures of the slides so that he can decipher and analyze them at a later time, or so I imagined, with the help of Java nonetheless.

Here are some general rules that I was able to grasp from this session... You need to manage the relationship of the parent and child object explicitly for each object individually. You should try to use left joins to fetch data in your queries. Using the @Version annotation attribute will help to detect parallel updates to a entity. You should use named queries whenever possible because they are cached by the JPA provider. Use fully qualified names for your queries such as 'Employee.findByName' to avoid named collisions. Concatenating strings in an attempt to make dynamic queries is not a best practice because it is open to SQL injection. Use named parameters in queries to avoid SQL injection. Try to avoid native SQL queries so that your application can be more database portable. That said, native queries might be required for vendor specific functionality or stored procedures.

Developing a Real-World Web Application with NetBeans 5.5 Visual Web Pack

David Botterill walk the audience through the design, analysis, issues, and resolutions of creating the NetBeans plugin portal</A> site using NetBeans, of course, and JavaServer Faces (JSF). Some of David's best practices for JSF development include are to come up with a preliminary set of requirements, use best guess at page design and page flow in prototype to get feedback, solidify the UI during analysis (the UI will effect the architecture), use Firefox Firebug to debug tricky HTML, CSS, and JavaScript.

Many of the issues described in by David seemed to have been web pack bugs that have since been fixed. That said, he did suggest some common pitfalls and how to avoid them. David suggested to start with Plain Old Java Objects, POJOs, not with relational tables and foreign keys, and joins, etc. David also stated that getting started with JPQL queries was a bit difficult, if for no other reason than it is something new that he had to go through. David found it difficult to have a button for a form which you can just press the enter button to submit it because there was no submit button on the form. Be aware of session timeouts, don't make assumption in your software about the state of the session, and always test for it. Be diligent about separating your CSS in separate files from your HTML. If you have a web application that relies heavily on the GET method then reevaluate using JSF because it makes it difficult.

Finally, David believes that for web applications resiliency is better than efficiency.

Grails, Sails, and Trails - Rails Through a Coffee Filter

This JavaOne 2007 Bird of a Feather (BOF) session seemed like a brief history of web application development. In the beginning there was pain. In the second day XML moved forth. And on the last day there was Rails, and developers thought it was good. Ruby on Rails' most mentioned philosophical innovations include Convention Over Configuration, Don't Repeat Yourself, Opinionated Software, Test Driven Development, and the 80/20 rule. The 80/20 rules indicates that Rails is not all things for all web developer. Rails does one thing and it does CRUD applications well.

Rails growth coincides with the disillusion of EBJ 2.x, and the painful write, compile, deploy, and restart cycle.

Sails models leverage Hibernate, view uses custom template engine Viento, and the controller is rigged using custom dependency injection library that provides Convention Over Configuration. The Viento templating system provides Ruby-like features such as method missing, mixin, and reopening classes.

The second Java-based framework heavily influenced by Rails mentioned during this session was Trails. In addition to Rails, Trails was influenced by the Naked Object pattern, and domain driven design. Trails uses Tapestry, Spring, Hibernate, and Maven to tie everything together.

Grails was originally named Groovy on Rails and as such it was heavily influenced by Rails' Convention Over Configuration, MVC, dynamic finders, and agile web development. Grails is powered by Hibernate, Spring, Sitemesh, Quartz, and the Groovy programming language.

A key benefit of Grails models over Rails models is that your model does not need to inherit some ORM ActiveRecord-like class, like in Rails. Grails embraces legacy enterprise systems with more complex relationships by leveraging Hibernate. Grails allows you to configure Hibernate and Spring when needed. A possible draw back, depending how you see it, is that in Grails you need to declare the properties in your model, which will be mapped to your database table.

Groovy code compiles down to Java byte code. A Groovy class is a Java class, and vice versa.

According to the speaker both Trails and Sails don't solve enough pain points while Grails is worth a shot. It is worth to note that as soon as some performance improvements have been made, JRuby on Rails will soon be a legitimate platform for your next web application. The benefits of Rails are that there are more books, more documentation, more inertia, and perhaps more developers for Ruby on Rails.

Rapid Seam Application Development with the NetBeans IDE

Michael Yaun, JBoss/Red Hat evangelist and co-author of JBoss Seam: Simplicity and Power Beyond Java EE, presented on the Seam development using NetBeans. Michael started by stating the state of the art in web applications which include animations, effects, partial page upload, AJAX, RESTful URLs, back button support, etc. Any modern framework needs to have support for all these features. JBoss Seam takes a page from Ruby on Rails.

JBoss Seam is a complete Java EE stack made up of JavaServer Faces (JSF), Hibernate, and Spring-like which adheres to Convention over Configuration and generators for rapid web application development.

JBoss Seam provides code generation, configuration by exception, testing is crucial, simplify simple cases (don't over engineer), and adhere to agile web development. A key feature of JBoss Seam is that is follows the Java way, is standards based, scalable, allows for reusable components, and provides a choice in enterprise components.

JBoss Seam has code generators that can create a whole scaffold web application based on your database schema. For each HTML page you have two files, one XHTML and one XML use for page navigation. From what I gather, Seam seems more verbose than Grails or Rails and even Yuan himself said, "If you are not familiar with Seam it might seem really intimidating with all those XML files."

JBoss Seam can be easily integrated with JBoss Rules engine and all those popular AJAX libraries such as Dojo and GWT. You can use the Red Hat Developer Studio for Eclipse will make Seam development easier.

Wednesday, May 9

Wednesday General Session

Thomas Kurian, Senior Vice President of Development in Oracle, gave the Wednesday morning general session. Thomas said that Oracle is following four key technological trends including JEE 5.0, Server-Oriented Architecture and Event Driven Architecture, Web 2.0, and grid computing. As Thomas spoke I keep thinking of JEE 5.0 as a MVC framework with EJB3/JPA as the model, JSF/AJAX as the view, and JSF as the controller.

Thomas talked about having JSF components that would generate AJAX or Flash widgets just as naturally as HTML. Thomas also talked a bit about enterprise mashups as corporations begin to bridge together AJAX, JSF, SMS, RSS, wiki, blogs, and social applications with business applications.

Thomas also positioned Oracle as a large and committed Open Source contributer and mentioned EclipseLink, Oracle recent code donation to the Eclipse Foundation.

While Thomas spoke about Oracle's position in the middleware, a thought occurred to me. Plain Old Java Objects (POJO) as used in EJB 3 and JPA are not as plain as you are lead to believe, but these objects might well be renamed to Plain Annotated Java Objects (PAJO).

Swing Vector Graphics

Luan O'Carroll, director of Xeotrope and lead developer of the XUI project, talked about how to improve your average pizza box-like business User Interface with Swing and Vector Graphics. According to Luan, the basic application UI is horrible with lack of style. It is hard to enhance or create a custom Swing Look and Feel so most developers don't bother and instead hope and wait that Sun will make a push for the desktop.

Luan provides a prescription for better User Interfaces that include using Synth, SVG, Swing, Java2D, Swinglabs, Timing Framework, and the Painter API.

Synth provides support for application UI skins and themes. Synth is basically a configurable L&F, which can easily allow for skins. Synth requires less knowledge for customing the UI that creating a new L&F from scratch. Luan recommends you use SVG instead of rasterized images with Synth because one size does not fit all with raster images.

SVG is an XML-based standardized format for describing vector graphics. Gaphic illustrators and graphics tools are familiar with this format. SVG provides a rich feature set such as animations, layers, overlays, and effects. There are seveal Java toolkits for working with SVG, such as Apache Batik and SVG Salamander.

You can breakout of the square Swing component with SVG graphics. Custom SVG/Swing components can be animated and support mouse events such as clicks and rollovers. It is possible to have Jave2D interact with SVG and vice versa so you can get the best of both worlds.

Luan mentioned the Synth SkinBuilder application that helps to customize the Synth Look and Feel. As of this writing, SkinBuilder does not seem to be available check Luan's blog for updates.

Effective Java Reloaded - This Time It's for Real

Joshua Bloch presented on his upcoming book Effective Java Reloaded. This was a repeat of his presentation given during last years JavaOne. I was astounded on the line to get into the room for this presentation. The line snaked the pavilion and back.

The following are some the suggestions given by Joshua during the Effective Java Reloaded session.

For object creation, Joshua recommends using the static factory pattern. The static factory has advantages over constructors, such as allowing flexibility for returning subtypes, caching, and in Java 5 you can do type inference.

Use the builder pattern to overcome the numerous telescoping constructor signatures or the bean style getter/setter methods. The build() method of the builder class constructs a object in a consistent state and can check for inconsistencies.

The following tips deal with Java generics.

Joshua suggested avoiding raw type in new code. Use generics for compile time checking. He recommends that you fully understand each warnings caused by generics and try to eliminate them if possible. If you can't eliminate the warnings, suppress them at the lowest possible scope using the @SuppressWarnings annotation.

Use bounded wildcards to increase applicability of APIs. Use <? extends T> when parameterized instance is a T producer. <? super T> when the API is a T consumer.

Don't confuse bounded wildcards with bounded type variables. A bounded type variable, <T extends Number>, restricts the actual class to a subclass of the typed parameter, in this case Number.

As a rule of thumb, if a type variable appears only once in a method signature, use wildcard instead. It is usually best to avoid bounded wildcards in return types. Don't overuse wildcards; it might water down the benefits of generics.

Generics do not mix well with arrays or variable length arguments. If given a choice, choose generics.

Joshua also talked about a cool mind expanding pattern, the Typesafe Heterogeneous Container (THC) pattern, which he talked about last year.

Joshua said that generics are tricky but well worth learning as they make your code better and safe.

Joshua also recommended the use of the @Override annotation every time you want and think you are overriding a method because sometimes you are overloading a method instead.

And finally, final is the new private. Minimize mutability where ever possible, but be caution when working with serialization or cloneable.

Building JavaServer Faces Applications with Spring and Hibernate

This technical JavaOne 2007 session was presented by Kito Mann, author of JavaServer Faces in Action, and Chris Richardson, author of POJOs in Action. Kito and Chris talked about the holy trinity of Java-based web application development, JavaServer Faces (JSF), Spring, and Hibernate.

JSF is a server-side UI component/event model with a basic set of UI components out of the box, and a simple MVC-style framework with a basic

Dependency Injection container. The standard UI component model of JSF enables third party and open source market place for additional components.

The Spring framework simplifies JEE development with AOP, ORM support, and Dependency Injection. Spring Beans are objects created and managed by Spring. Spring AOP allows modular implementation of cross cutting concerns, such as security, logging, and more.

Hibernate is a super set of Java Persistence API (JPA). JPA is a standardized ORM, providing transparent persistence, tracks changes of objects, manages identity, and maintains concurrency. Hibernate improves productivity, performance, maintainability, and portability.

If you are going to be doing web application development with JSF, Spring, and Hibernate then you should look into JBoss Seam. JBoss Seam integrates JSF, Spring, and Hibernate with a set of generators and conventions. Seam is greater than the sum of its parts.

Extreme GUI Makeover 2007

This was one of the most enjoyable sessions in all of JavaOne. Last year's Extreme GUI Makeover, the gang made over a typical looking mail client. This year Christopher Campbell, Shannon Hickey, Hans Muller, and Romain Guy presented transformed a pizza box-like business User Interface (UI) into a visually stunning UI with Swing and Java 2D. The first type presented by the extreme makeover folks is to use a modern Look and Feel like Nimbus.

According to Romain, it makes a good idea to add a splash screen, and in Java 6 this is easier than ever with the new -splash JVM option. You should also make good use of non-rectangular UI splash screen with a drop shadow, and a progress indicator. The demo here faded the splash screen image from sepia to a full color image to indicate the loading progress.

They demoed JGoodies form validation, which paints a little warning icon on the text field. Another demonstration showed a modal dialog with rounded corners ala Web 2.0 which blurred the main application window below. They rounded the border by overriding the paintBorder method on the Swing component. They use of gradients everywhere. They demonstrated a set of thumbnails with reflection using ReflectionRender that you can find in SwingX project from the SwingLabs. The demo also demonstrated custom table cell rendering. The speakers said that the default cell renderer is a JLabel but that you can use any JCoponent with any layout.

The demo had some animated effects for sorting and filtering table rows by extending a TableRowSorter available in Java 6. The demo used the Timing Framework to drive the animation and used the Glass Pane to paint the animation.

One of the best moments of the demo was when Romain Guy joked, "One thing I don't like about UI is that it is constrained inside the frame of the window." Why stick square components inside to the bounds of a window? Romain demoed a translucent non-rectangular dialog window that floated outside the main application window. To get this effect he said he used the JNA API, which I understood him to be described as JNI for Swing.

In speaking about boring business application UIs Romain joked, "You want to call the police to arrest the guy that wrote that UI."

In summary, if you want to makeover your Swing UI, use Java 2D, Timing Framework, SwingX, and the Nimbus L&F.

Anatomy of an Eclipse RCP Application

Wayne Beaton, Eclipse Foundation evangelist, gave an overview of the Eclipse Rich Client Platform (RCP). As he spoke, I thought of Eclipse RCP as a Fat Client minus the lard. Wayne described a Rich Client Platform as a system that provides a rich user experience, is platform independent and extensible, and provides a component model.

The Eclipse IDE is itself a collection of plugins, or components, on top of the base Eclipse platform. A big aspect of Eclipse RCP is Equinox, the OSGi based module system used to discover Eclipse plugins. In this context, plugins, modules, and components are used interchangeably.

Wayne gave a quick overview of the plugin structure of Eclipse. Basically Eclipse provides extension or plugin points at its core level. Adding or removing functionality is as simple as adding and removing files from the plugin directory in the Eclipse installation directory.

Wayne also mentioned Mylar. Mylar monitors and remembers which resources a developer is interested in for a recorded task and hides the rest. Wayne said, "You need to look into Mylar. Mylar will change your life."

Tricks and Tips with NIO

This JavaOne 2007 technical session, Tricks and Tips with NIO – Using the Grizzly Framework, provided some general advice when working with the NIO classes. The most important piece of advice was to use the Grizzly Framework. The Grizzly project hides the programming complexity of using NIO so that developers can focus on implementing large-scale servers. Grizzly is in use in the GlassFish Application Server.

For those not working with the Grizzly Framework, Jean-Francois Arcand of Sun gave a quick overview of NIO, or the new IO library introduced in Java 1.4. Jean-Francois reminded the audience that traditionally Java sockets are blocking but NIO channels can be either blocking or non-blocking. The talked described some very technical tricks and tips when working with NIO. The last tip and most important piece of advice was to use the Grizzly project when working with NIO.

Dive into the GlassFish Aquarium

During this Bird of a Feather session Eduardo Pelegri-Llopart, Distinguished Engineer at Sun, covered similar material as the slides presented during the GlassFish Day at Community One. The key take aways from the BOF was that GlassFish is a production quality application server. GlassFish also enjoys plenty of support from third-party vendors and a large collection of accompanying projects such as jMaki, Phobos, Blogapps, Hudson, Rome, and more. Eduardo announced the first ever GlassFish Champions in recognition of their key contributions to the community.

Seamless Web Browser Integration

The complete title for this JavaOne 2007 BOF was Ingredients for a Killer Application - Adding Mojo to Your Swing and Ajax Applications with Seamless Web Browser Integration. Wow, what a mouthful. It might not be immediately clear from the title but the focus of this BOF was the use of the JDesktop Integration Components (JDIC) Web Browser embedded in a Swing application. As we all know Swing does not a complete Java-based web browser component, although there is a good XHTML Renderer. With the JDIC you can embed a native browser (IE, Firefox, or Safari) in your Swing application. Unlike the XHTML Renderer, the native browser allows you to run JavaScript code, apply AJAX/Web.2.0 effects, manipulate the DOM, and more.

Integrating JDIC into your project is as simple as adding the jar. A key demonstration of this BOF was to show how to have bi-directional native browser communication with your Swing application. To use the JDIC native browser, just create a WebBrowser object and invoke the executeScript method passing some JavaScript code in a string to communicate from the JVM to the browser.

To work your way from the broswer to the Java process you need to listen to the statusTextChange method on the WebBrowserListener. The statusTextChange method would be called when you set a string value to the browser window, as in the following JavaScript code.

```
window.status = "message passed to the JVM";
window.status = "";
```

The speaker did warn the audience that using the status text to listen for processing instructions from the browser was a total and complete hack. The speaker recommended that as soon set the status to communicate with the JVM that you clear the command/message. The speaker talked about having your own mini-protocol to pass data and instructions to the JVM, I would recommend you use a standard like JSON.

Someone in the audience asked about having access to the x, y location of DOM elements in the browsers to combine with Java 2D overlays and effects, but the speaker was not sure how to go about implementing such a task.

Using the JDIC native web browser in your Swing application you can take advantage of Web 2.0 services such as Google Maps and Flickr in a Swing/Web 2.0 mashup. I have been looking into the JDIC web browser so that I can create a offline standalone web-based application by embedding a browser, Jetty, HSQLDB, and JRuby on Rails. Embedding all these components is easy enough to integrate, although you might have to write some glue code to make into a fully stacked framework.

Putting a Swing Front End on a Web Application

During this JavaOne 2007 BOF, David Wroton shared his experienced of porting a JavaServer Faces (JSF) front-end to a Swing desktop client. David started with the whole browser vs fat client debate. David listed his issues with the browser, such as HTML Scrolling, back button, Funky JSF URLs. He said, "The back button is something that is breaking society." The benefit of developing a HTML web application is that it is easier to get started.

The issues listed by David as holding back a rich Swing desktop include installation, maintenance, and getting started from scratch. To work around installation issues David used Java Web Start and instead of writing the desktop client from scratch he looked into using a Rich Client Platform such as Eclipe, NetBeans or Spring. Regarding the choice for which RCP to use he said, "There was a review last year, the review said, man, they are all hard." Ultimately he selected Spring RCP, mostly because of his previous exposure to Spring. According to the speaker, Spring RCP has good form, data binding, and validation support. The major draw back to Spring RCP is the lack of documentation, an issue descried in last years' review of Rich Client Platforms.

In closing, the David told the audience that the desktop is far more rich than any web based application, but AJAX goes a long way making a web application more user friendly and for many applications that is enough. Finally, he said there is still no clear lightweight/agile solution for a Swing framework.

Thursday, May 10

JavaOne 2007: Thursday General Session

Padmasree Warrior, Motorola Executive Vice President and Chief Technology Officer, gave the Thursday general session at JavaOne. As you can imagine, her theme was the mobile network. In a presentation entitled Platform Disturbia she stated that in the mobile device world each network, each device, each spacial domain is its own platform. She also presented some figures where she states the there are more cell phone devices than cars, or televisions, or any other electronic consumer good. Padmasree enthusiastically said, "The whole world is a hot spot."

Padmasree had a quick panel discussion regarding which mobile technology will win, mobile AJAX vs J2ME. The panel discussion was with mobile tech writers and pundits Ajit Jaokar and C. Enrique Ortiz. In the end, I think it was Enrique that said that both have their strengths and weaknesses. To me, the debate sounded a lot like the same one between the desktop and web applications.

One thing that Padmasree said that stuck with me was that software development is about the user experience not the user interface. Sometimes as engineers we do over think a solution. We often design a system that ends up looking and behaving like it was designed by a software engineer, not a real person.

Motorola also took the time to promote their Open Source initiatives and their Developer Studio.

Being Productive with Swing

Ben Galbraith, founding blogger at Ajaxian, gave an insightful presentation at JavaOne 2007 on Swing. Ben started off by saying that Swing isn't in the 'Wow' game like WPF, Cocoa, or Flex/Apollo. According to him, JavaFX Script might fill that gap but not just yet. In software development, productivity is king. Productivity is the currency of IT development. According to Ben, the Swing Application Framework does not address productivity but instead focuses on basic architectural concerns.

A theme that I picked up from Ben was that there is too much freedom when working with Swing and everybody builds GUIs in their own fashion. Since there is no consistency in UI development even within a project and since there is no solidified framework, Swing applications are hard to maintain and support.

Basically, there is no opinionated software in Swing to provide a standard, convention over configuration, pattern for building Swing-based applications. The key concerns a Swing framework needs to address are data binding, actions, listeners, activation, consistent form, and life cycle events amongst others.

One invaluable recommendation made by Ben is to use the JGoodies Forms layout if you hand code your UIs. Otherwise stick with your favorite UI Builder. He recommended the UI builder over fidgeting with UI code by hand. He said that coding a UI by hand is like hand writing a 50-page TPS report. He also reminded the audience to not muck or edit generated UI code. Ben said to let the UI builder own the code.

Ben recommended JForm Designer as an excellent commercial GUI builder and NetBeans as an Open Source alternative. Without a UI builder, it is difficult to attain UI consistency while fudging with spacers, fonts, insets, etc. Having a framework helps to attain a consistent layout for you desktop application.

Ben also had time to mention his framework, Swing Clarity. Clarity is a Swing Rich Client Platform that borrows heavily from Ben's experience as a Ajax/web developer. Clarity has a Prototype-like method, $(), which can find named UI components using CSS-like selectors. With Clarity you can bind behavior to components by the variable name of the component with the help of custom @FormUIField annotation. I will definitely look into this framework as it introduces some ideas that make web development exciting. Clarity also provides di-directional data binding and allows for some amount of skinning.

As an interesting side note, Ben called Karten Lentzsch, JGoodies project lead, the God father of Swing experts.

Technical Overview of GlassFish v2

This was the third and last JavaOne 2007 session that I attended that dealt with the GlassFish Application Server. This was a technical overview of what's new the latest release of GlassFish. The new features in v2 include clustering support, high availability (five nines or that is 99.999), memory replication, and Java business integration just to name a few.

Just as a community service announcement, if you attend a technical session and the speaker starts off by saying, "I have been heads down on this feature..." you should walk out. Anybody that is head down for a year on some feature will surely bore you to death if you ask him about it, even if just in passing. Well this guy I am talking about went into how GlassFish uses JXTA, peer-to-peer technology, to achieve memory replication.

The latest release of GlassFish also has a server usage profiling tool, new administration console, performance enhancements, lazy loading of providers/containers such as EJB, JMS, and HTTP. It is makes good use of the Grizzly project.

JavaFX Script

This JavaOne 2007 session was originally titled Form Follows Function (F3), after the code name for what was marketed as JavaFX Script. JavaFX Script (JFX) has been one of the most hyped up technology announcements here at JavaOne. According to the marketing literature from Sun, JavaFX Script is a "highly productive scripting language that enables content developer to leverage blah blah ..." What Sun PR means to say is that JavaFX Script is Sun's too little and perhaps too late response to Microsoft Silverlight and Adobe Flex/Flash.

In an eat your own dog food sort of demo, Chris Oliver presented his presentation slides on a PDF viewer application he created using JFX. The PDF viewer demo had great zoom support and even thumbnails of the PDF documents with reflection. JavaFX Script have me at hello, world.

As Chris described, he started working on what eventually became JFX because he wanted to provide an answer to the following set of questions, why does it take so long to write GUI applications? How do you avoid the "Ugly Java GUI" stereotype? Why is it easier to write a web application than a Swing application? Why don't Swing programmers use fancy images, graphics, effects, and animations like the Web 2.0 designers?

JavaFX Script is a new programming language that runs on the Java VM. JFX is an object-oriented language with declarative syntax, data binding, and statically type with type inference. JFX allows for custom painting, transformation, grouping, transparencies, effects, clippings and animation. JFX allows you to read and write images. It seemed to me that SVG can do just about the same thing that JavaFX Script intends to do. JavaFX also reminded me of the Processing project.

As mentioned SVG, Silverlight, OpenLazlo, and Flex provide similar functionality and effects but they introduce a mix of XML-based markup and some degenerate scripting language. SVG is declarative like JFX but Chris doesn't like the JavaScript/DOM programming model. Chris stated that he rather use a real programming language and a programming model that software engineers understand. The folks behind JFX intend to provide full language support such as refactoring, code completion, code search, code high lighting in NetBeans. In addition to static type efficient compilation JFX provides compile time error and warnings.

Why Spaghetti is Not Tasty

Jasper Potts, Sun, gave an insightful presentation at JavaOne 2007 on Why Spaghetti is Not Tasty: Architecting Full-Scale Swing Apps. Jasper started the technical session by describing the first typical architecture for a Swing application. The typical first architecture is to pass everything to everybody. The second level of abstraction in a Swing architecture is to use a singleton object that functions as the application's service/resource finder/locator which provides lookup methods for an implementation of a given interface. For large applications, Jasper recommends using a framework and he recommends Spar. Spar is a Swing Rich Client Platform framework written by Jasper and based on his research on best practices. As Jasper described, Spar is aimed for applications with over 50K lines of code.

Jasper also talked about software modularity, of breaking up an application down into manageable chunks. Classes, packages, and libraries provide modularity in Java but Jasper, and many other people here at JavaOne, is talking about allowing for modular plugins for extending a given application, such as Eclipse's plugins of NetBeans' modules. There are currently a few specifications regarding modularity such as OSGi and JSR 291: Dynamic Component Support for Java SE. Jasper recommended several implementations, in particularly Eclipse Equinox and Knopflerfish. Additional JSRs along these lines also include JSR 277: Java Module System and JSR 294: Improve Modularity Support in Java. These JSRs intend to provide a superpackage modular system at the language level. A module helps to separate public API from its implementation, not every class should be visible to the whole application and the visibility identifier can only do so much.

Jasper also made another recommendation that would clean up your code from inner classes. As described by Jasper, a typical Swing application uses the communication pattern typically used by listener classes, which inevitable produce huge interconnections across the whole application and a ton of inner classes littered in the source code. The solution solution recommended by Jasper is to use a message bus for Swing events. This allows the developer to set message rate limit.

The key advice given from Jasper is to break up an application into modules, limit direct coupling, and employ best practices such as plugins, IoC, and event message bus frameworks for developing your next Swing application.

Beans Binding

In this aptly named technical session, Shannon Hickey and Hans Muller tag teamed to talked about the current state of JSR 295: Beans Binding. Right off the bat, Shannon said that the Beans Binding implementation is far from final and if any body has any thoughts on how beans binding should be done Shannon suggested they participate in the discussion.

The Beans Binding JSR intends to replace the property change and action listeners that litter Swing applications. In essence as Shannon described Beans Binding "keeps two properties in two objects in sync." Beans Binding also has support for objects that don't adhere to the bean pattern, such as collections. Beans Binding also has the ability and added value that it can convert and validate values of as it sync and binds them to the target object.

The end goal of Beans Binding is to make it trivial to bind an application model to a Swing UI component.

Write a 3D Game in Java

The complete name for this JavaOne 2007 technical session was Write a 3D Game in the Java Programming Language in Less than 50 Minutes. This was not a hands-on session but instead Erick Hellman spoke on his experience writing a 3D game in Java. In full disclosure Erik informed those present that he never before written a game, has basic knowledge of OpenGL, lacks artistic talent, and suffers from color blindness. He said, "I need a ruler to draw a stick figure," but he reassured the audience that a ruler wasn't required because game development is made simple in Java.

Erik presented a basic recipe for game development. The recommended tools of the trade include Blender for 3D modeling, Gimp for image manipulation, JOGL for OpenGL bindings, JInput for game control, Vectmath for Java 3D coordinates, and XMLBeans for loading and persistent the models.

The architecure of the game revolves around the scene graph. The scene graph is the core of modern games and everything builds around it. He also briefly touched on the core mathematical concepts of 3D graphics, which are matrix transformations such as rotate, scale, and translate. For those that didn't do well in linear algebra, or don't even know what that is, the speaker say that OpenGL abstracts most of these details and that what a game developer is concerned with is the graphics pipeline.

Erik also mentioned some good 3D model formats. Lightwave, 3D Studio Max is a closed formats but he also recommended Collada and X3D as open format alternatives. The speaker also described a bit the ideas and algorithms used for collision detection. The technique he used is to enclose the character in a virtual, invisbile, bounding box. Elements of the game, such as barriers, need to be outside ot the avatars bounding box. He also listed some common issues when developing a multiplayer game. These issues include latency, pocket loss, and cheating.

In summary, Erik listed the key components for a simple game. He assumed the audience that with a scene graph, model, navigation, and collision detection any one there could in fact write a 3D game in 50 minutes. Talk about rapid game development!

Web 3.0 - This is the Semantic Web

This JavaOne 2007 BOF was about the coming web, deemed the Semantic Web. The way I would describe a semantic enabled site is to talk about microformats. You can use microformats to annotate a given snippet of HTML to describe the data contained in the HTML. For example, a typical site uses HTML to structure the data, CSS to style the page, but if you want to annotate and describe your data you can use a microformat and add additional attributes to the HTML tags. Simply speaking, you can use a microformat in a web page to describe the meaning and relationship of data. For example, blogs that have a blogroll can add XHTML Friends Network (XFN) data to links to describe the relationship to the link, whether they link to friends you met or coworkers. But instead of using HTML, the Semantic Web is based on Resource Description Framework (RDF). And instead of using XFN, you use Friend of a Friend (FOAF) to describe human relationships. RDF is a way to meaningfully describe your data.

Another format like FOAF is the Description of a Project (DOAP). DOAP is used to describe an Open Source projects with a name, description, SVN info, etc. so that this information can be crawled, indexed, scrapped, and later successfully searched. RDF, like XML, is open in such a away that you can define your own ontologies, and there are many existing ones that will be standardized. Competing ontologies will be naturally selected and standardized by the community. Another interesting RDF format is Beatle, bug and enhancement tracking language.

The speaker started the session by describing a timeline of recent modern computing. He narrowed the PC Era to the desktop of the '80s, the Web 1.0 was the original World Wide Web of the 90's, he described the Web 2.0 as the Social Web we currently live in, Web 3.0 will be the Data Web or Semantic Web, and Web 4.0 which is slated for 2020 will be the NetOS or Intelligent Web. In terms of search, he stated that we started Web 1.0 with natural language search, we will soon move to semantic search, then associative search, simple reasoning, and finally to intelligent agents in the Web 4.0.

The speaker said that the Semantic Web and RDF can be used to search, archive, and retrieve online content in new ways. He said that the open web would be treated like a massive distributed database. The web is made up of data and RDF can define your data into a database that can be located, reference, and related to other data with URLs.

The speaker also touched on the growing number of semantic tools, a large portion of which are written in Java. The speaker spoke of RDF databases and repositories and SPARQL, an SQL-like query language to search RDF repositories.

The Java 3D API and Java Binding for OpenGL

This JavaOne 2007 BOF covered both the Java 3D and JOGL APIs. The session started with the news that Java 3D 1.5.0 was released back in December 2006 and includes a lightweight JCanvas3D and several bug fixes. This session had a ton of kewl demos, mostly of scientific applications and Web 2.0 style image reflections. They also demoed Wonderland, a Second-Life-like virtual and immersive world developed in Java using the Darkstar game server. The speaker described Wonderland as a "virtual collaborate environment," where your avatar can peer-program in NetBeans with fellow engineers telecommuting half way around the world!

The speaker also posed an open question to the Java 3D community. He wondered if the community in general should focus on version 2.0 or 1.x, where 1.x would be fully backward compatible. Key features he said he would like to see are to allow a mix of Java 3D and JOGL, making lightweight components first class citizens, ease of use utilities for sound, and a plugin architecture.

The Java OpenGL portion of the BOF skipped the PowerPoint presentation and went straight for the demo. NASA Ames' World Wind application was demoed. World Wind is an open source, Java-based, extensible and mashup-able, and mind boggling Google Earth like application. They also demoed some Java 3D/2D integration with JOGL where some Java 2D based text graphics was shaded and animated by JOGL code.

Glossitope - An Open-Source Java-based Widget Container

Joshua Marinacci and Robert Cooper presented on Glossitope, a Java-based widget Container. Glossitope was originally named AB5k. Joshua is also the founder and lead of the XHTML Renderer project, which is code named Flying Saucer. As good of a coder Joshua is, he is horrible at naming projects!

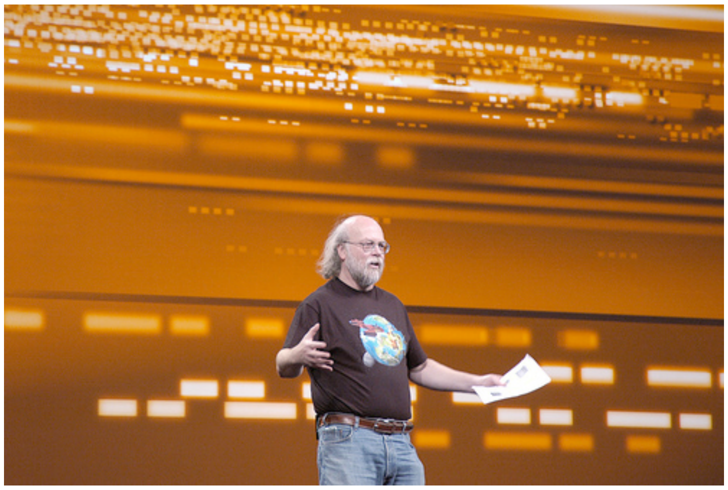
A widget, also known as gadget, is a small and simple program that lives and runs in a manager container. As Joshua said, they do "a lot of useful awesome little things." The OS X Dashboard widget system comes with a calendar, date, and calculator widget. Independent developers have released all times of widgets such as RSS reader and weather widgets that there is a growing widget economy.

There are a few other widget systems other there such as Google Gadgets for the web, Dashboard for the OS X, and Yahoo! provides widgets for the desktop as a third-party application. Even with all these widget systems, Joshua and Robert see issues with them all, such as they use an XML-based markup and lack a real programming language. These 'issues' sound surprisingly similar to the ones presented by Chris Oliver for his motivation for JavaFX Script. Another common reason used for both the JavaFX Script and Glossitope projects is large number of Java libraries. Joshua also reminded the audience that Glossitope has immediate support for a myriad of languages such as JRuby, JavaFX Script, and French (via Unicode).

Joshua also mentioned that it is his hope to be able to support widgets written for other systems like Yahoo! or Google. He also hopes that Glossitope will be the standard widget system for Linux distributions. As a call for action, he asked the audience to write more widgets targeted for Glossitope. To demonstrate the limitless ability of Glossitope, Joshua demoed a 3D widget using JOGL! Glossitope is very alpha and still throws the occasional exception here and there but still worth the look.

Friday, May 11

JavaOne 2007: Friday General Session



Friday's general session, the last day of JavaOne 2007, is generally known as James Gosling's Toy Show. The Toy Show is just a series of demonstrations of kewl applications and technologies that make good use of Java. To paraphrase Gosling, when all the announcements have been made "What do you do at the end? Inspire you!"

Gosling talked a bit about Network.com, Sun's pay-per-use grid computing network. Sun's grid computing is offering 200 free hours. Tor Norby, the 'Demo Stud' as Gosling introduced him, showed off NetBeans 6 and demoed NetBeans' Ruby on Rails integration and local file history. My favorite demo was of Project Wonderland. Wonderland is a virtual collaborate environment. You can run FireFox, NetBeans, and just about any other application in 3D in

Wonderland. In wonderland, Gosling joked, is a virtual world where you could have your own window office. Gosling also talked about a meat scale with a JVM and browser that can communicate to a centralized database. Gosling also had robots, Robosapiens, literally running on Java and dancing on stage.

Bringing Life to Swing Desktop Applications

Kirill Grouchnikov and Alexander Potochkin presented on advance Swing techniques at this JavaOne 2007 technical session. They talked about advanced painting and effects such as non-rectangular components, translucency, layering, image filtering and animation.

Custom components, especially non-rectangular components, should override the contains and paint methods. You can implement any effect on custom components by overriding these methods. By default, components are opaque. You can use opacity for transition effects, such as fading in and out of a container when tabs as selected. The reason you want to use visual effects is to indicate to the user a change in state, by nature Swing changes the UI immediately on an event.

In addition to adding effects to custom components and playing with opacity, Kirill talked about using a custom RepaintManager, and drawing on the glass pane. Customizing the RepaintManager allows you to control Swing's component painting, for example to force a container to repaint child components. Drawing on the glass pane allows you to paint over all components. Kirill also suggested using JXPanel and JXLayer of the SwingX project from SwingLabs, these containers supports translucency, Painters API, image filtering, and non-rectangular components.

Kirill and Alexandar also talked about layering with UI delegates. Swing UI delegates are the classes responsible for painting Swing components, there exists one UI delegate for each component. As of current Java versions there are 41 UI delegates.

Kirill demoed some eye candy Swing effects, something he calls the Rainbow, a SVG explorer application that demonstrates many of the techniques discussed in this session.

Ajax and JavaServer Faces Tooling in Eclipse

This JavaOne 2007 technical session started off with Cameron Bateman describing how to use the Eclipse Web Tools Platform (WTP) Project to create JavaServer Faces (JSF) applications and custom tools for Eclipse. This was a discussion better suited for JSF component developers. The Eclipse web tools provides a web page editor, JSF configuration model, editor, and wizards. Eclipse and the WTP project are extensible, the WTP in particular allows developers to leverage design-time tag rendering, expression language support, and the meta-data framework. Customizing Eclipse and the WTP project allows you to provide design-time support for your custom tag libraries in the Eclipse IDE.

This was like two sessions in one because Philippe Ombredanne then presented on the Ajax Toolkit Framework (ATF) tools in Eclipse. The ATF project can help to develop AJAX applications in Eclipse. The most interesting aspect of the ATF project is the Eclipse Mozilla mashup. The ATF project embeds Mozilla in Eclipse via XULRunner and JavaXPCOM. In addition to embedding a browser in the Eclipse IDE, the ATF also provide a rich feature set for AJAX developer such as a JavaScript editor and debugger, DOM and XMLHttpRequst inspector, and plain simply faster AJAX development. You can debug a remote website in Eclipse with the ATF project similarly how you would step through a Java project in the debug perspective. Embedding the Mozilla "enables web browser integration beyond the capabilities of the standard SWT browser widget" and brings AJAX applications to the rich client desktop.

Bytecode Manipulation Techniques for Dynamic Applications for the JVM

This JavaOne 2007 technical session seemed like a panel discussion between Eugene Kuleshov and Tim Eck of Terracotta, Tom Ware of Oracle/TopLink, and Charles Nutter of Sun/JRuby. The session started off by describing the Java Virtual Machine, the Java bytecode, and the ASM framework. The Java Virtual Machine (JVM) is a proven and reliable platform aimed at high-performing applications. The JVM is designed for statically-typed languages but provides class loading and a reflection API for dynamic languages. The discussion also delved into the class file format. The class file format simply consists of field and method names, string literals and constants, and debug information.

The ASM bytecode framework is a simple, small, and fast library for adding dynamism to your Java application. The ASM framework is useful for Java code generation and modification. Terracotta, TopLink, and JRuby use ASM to dynamically inject code into an existing class.

Charles Nutter described how the JRuby team is using the ASM framework to support Ruby's dynamic and open class nature. Speaking of JRuby's move to ASM Charles said, "To be slower than one of the slowest dynamic languages was embarrassing." Using ASM, recent version of JRuby are performing better than the C implementation of Ruby in some cases.

As a word of warning, you should always document use of code generation because it is hard to debug and maintain if you don't know what is going on.

Filthy-Rich Clients - Talk Dirty to Me

Romain Guy, of Google, and Chet Hasse, of Sun, presented on their forth coming book Filthy Rich Clients which is based on a previous JavaOne technical session of the same name. They described this session as "the presentation based on the book based on the presentation."

The session began with a slide listing Data Binding as a goal for the talk. Romain then said, "Data Binding is the most critical development of Swing today, but is it not here yet." The next slide crossed out Data Binding and listed Applications Framework which is subsequently scratched out for 'Cooler Applications.'

The ingredients, or agenda, for developing cooler applications according to these filthy rich engineers is based on graphics, performance, animation, and effects. Romain suggested that you override the paintComponent method instead of the paint method for custom components. Overriding the paint method may clobber some painting code that should really happen, like appropriately calling paintBorder, paintChildren, etc. The session titled Bring Life to Swing Desktop Applications talked about overriding the paint method.

For the performance portion of the talk, Romain was mostly concerned with the performance of scaling images, such as creating thumbnails, applying effects, etc. For these type of image operations quality and performance matter most. Romain explain how there are simply too many options to scale images in Java. The best performing methods to scale an image down is to use the drawImage method using the default interpolation (NEAREST), the second best method is to use drawImage with BILINEAR, followed with BICUBIC.

Another common sense piece of advice offered here was to request repaints only for the area where the UI has changed, this is generally known as clipping. To repaint just what you need you can use the repaint(x, y, w, h) method on the component.

In regards to animating your Swing application, Romain and Chet recommend you use the Timing Framework from the SwingLabs for scheduling and running animations. The Timing framework has an Animator class that can run a instance of a TimingTarget which has the animating code.

For effects they recommend the blur, drop shadow, spring (ghost effect), morphing, and animated transitions. The SwingLabs has a GaussianBlurFilter, ShadowRenderer, Morphing2D class to achieve blurs, drop shadows, and morphings.

The idea of using animated transitions such as fading in and out is to lead your users, not leave them. Romain and Chet are working on an Animated Transitions project related to the book.

Writing Games with Project Darkstar

For the last session of JavaOne 2007, I attended the Writing Games with Porject Darkstar technical session presented by Chris Melissinos, Sun's Chief Gaming Officer, and Jeffrey Kesselman, Chief Darkstar Architect. Project Darkstart is Sun's Gaming Server (SGS). Chris told the crowd of gamers that there are huge demands for online game developed at lower costs. He also described how it is easier to make 15 games that generate $1M than to develop one game that will generate $15M. With that said, Chris elaborated that the purpose of Project Darkstar is to make practical, massively multi-player online (MMO) games. Current MMO games scale only by zoning users into different shards. The design goals for Darkstar are similar to those of GlassFish, Darkstar is to be a distributed, persistent, fault-tolerant server that is economical to develop for and administer.

There was also time for a quick discussion on how to develop a game on Darkstar. Someone spoke about their experience writing Bunny Hunter Online, a 2D multiplayer action game

Copyright