Java One 2006 Conference Notes

By

Juixe

Wednesday, June 17

Java One Wednesday: General Session

Thomas Kurian, Senior Vice President, Oracle Server Technologies Development at Oracle gave the morning general session today. The key to note about Thomas' presentation is that Oracle is hard at work with an implementation of the EJB 3.0 specification. Thomas stated that the Oracle EJB 3.0 implementation has been donated to the Open Source community. Thomas eagerly believes that EJB 3.0 will simplify enterprise development. He said that anyone that can develop a POJO would be a J2EE developer. It seems that the EJB 3.0 spec defines an enterprise bean as a POJO whose life cycle is demarked with annotations.

Effective Java Reloaded

Joshua Bloch wrote the seminal book Effective Java. Effective Java is the one book that needs to be in your Java bookshelf. Joshua does not have a new edition of Effective Java but in this session he showed effective use of some of the new Java 1.5 features, such as Generics. Here are some blurbs from the Effective Java Reloaded session.

"Get used to typing private final."

Static factories have advantages over constructors. You should strive to minimize mutability. Use the Builder pattern to emulate named parameters. Optional, non-required, parameters will require a setter method in the builder class. Have a build method in the Builder class that will actually create the instance of the class you are building. Avoid raw object types, use generics. When using generics, prefer wildcards to type parameters. Wildcards are good in your API as parameters but it is a bad design to return a generic with a wilcard. According to Bloch, returning a wildcard generic smells like bad code. Generics and arrays don't mix well; choose generics above arrays. Here is a mind-expanding pattern: THC, that is Typesafe Heterogeneous Container. And finally, final is the new private. Joshua stated, "Get used to typing private final."

This was a great session lead by a great developer.

Web Service With Eclipse

The name for this session almost reads like a run on sentence: Create, Test, and Consume Web Services with the Eclipse Web Tools Project. This session covered new features in the Eclipse Web Tools Project. The audience was shown how to use the many web tools and wizards in Eclipse to create, test, and consume web services. Eclipse comes with a Web Service Explorer, which

can be used to test a web service… The speaker did make it seem easy but I feel that there might not be enough online documentation.

One note regarding this session, I was surprised that this speaker, speaking on the subject of web services and SOAP, did not know or heard of REST. More than anything this speaks to the amount of technology that a typical Java developer needs to be familiar with. A typical J2EE developer needs to be well versed in so many underlying technologies and web services.

Design This Container

I attended the Creating Professional Swing UIs Using NetBeans GUI Builder (Formerly Code-Named "Matisse"). I have been meaning to get started with the NetBeans GUI Builder to become a more productive UI developer. An important point for me is that Matisse supports custom components. This is important for me because at my current company we have developed a lot of custom UI components. The Matisse GUI Builder is a WYSIWYG visual builder with a drag and drop feel, which can be used to design an UI as oppose to developing one. Of course UI builders are great to quickly prototype an application.

The NetBeans UI Builder reminds me the XCode's Interface Builder. This IDE has context help, and UI Builder, like Interface Builder with alignment help. With both these tools, UI design becomes an art rather than a science. At the end of the session one of the speakers simple summed up the session as, "Hand coding bad, Matisse good."

Here is a good design practice when using the NetBeans UI Builder: Design every container as resizable.

Dos and Don'ts For Swing Apps

This session was presented by Karsten Lentzsch, founder of JGoodies. Karsten said, "It's easy to program swing … badly." According to Karsten, the average desktop project does not have a budget for UI usability analysis. This is especially true for many Open Source projects. Krasten presented some Swing dos, like do remove clutter by writing short text labels, do remove duplicate borders, do reduce design to its core essence, do use a stable aspect ration such as 1:1 or 4:5, do add negative space. Karsten believes that white space is not wasted space. In addition, he favors space to separators.

Some dont's include don't use saturated colors, don't change fonts (use native fonts and size), don't break the icon design (use a consistent icon set).

In closing, Karsten said "many people could do at least a good layout with Matisse."

"It's easy to program swing … badly."

A personal note; I saw my former instructor from SJSU at this session, Professor Horstmann. Professor Horstmann is author of Core Java.

Groovy On The JVM

I attended the Groovy = Java Tech + Ruby + Python for the JVM session. This is a simple equation that makes a lot of sense to me, especially since I write a lot of Groovy scripts at work. This session was a 101 introduction to groovy. Rob Cope of OpenLogic did a great job at describing the dynamic aspect of Groovy, Closures, Groovy Markup builders such as AntBuilder and XmlBuilder, and the the GDK extensions.

For those not familiar to Groovy, Groovy is a object-oriented dynamic language that runs on the JVM. In Groovy, static typing and semicolons are optional. Groovy has a lot of built-in features missing in Java, such as regular expressions. Yes I know Java has regular expression, but I meant that Groovy supports regular expression at the language level not as a class you import. A great feature of groovy is that Groovy compiles down to a class file that can be used in any Java application. According to Rob, "your boss can't tell if you are using Groovy."

The one feature of Groovy that I was not at all familiar with was the Scriptom. Scriptom gives you access to COM objects, in a fashion that is similar to VBScript developers. Rob showed a demo of Scriptom, he opened Microsoft Excel, added a few values, created a chart, saved the chart to the file system, and created a JFrame with a button that use the chart as an icon. More amazing is that he did this without restarting or closing or rebuilding anything.

I am seriously thinking of replacing all our VBscript dependencies from our application to Groovy/Scriptom. We use VBScript to open and print Word and Excel documents. With Groovy and Scriptom I could do this all in the JVM. The only word of warning is that if there is an error with the DLL you are invoking the Java process will die!!!

RAD Frameworks For The Java Platform

For me the last session of the day was, RAD for the Java Platform Web Tier: Frameworks Panel Discussion. The three frameworks covered for this session were Grails, previously known as Groovy on Rails, Trails, and RIFE. Both Grails and Trails are heavily influenced by Ruby on Rails. All of these frameworks where developed because a web application written in Java has been harder than they should be. According to the lead developer of Trails, once you develop your model you get your application for free. In the case of Trails, the framework heavily depends on annotations, a Java 1.5 feature.

All of these frameworks are not re-inventing the proverbial wheel; they are built on top of Java Open Source projects such as Hibernate or Spring MVC.

I am planning to attend the BOF sessions for these frameworks tomorrow. I am interested in using one of these frameworks for a project. Struts is pretty much

out of the question, but at this point all three frameworks are contenders. Maybe after tomorrow I will have a better idea of which framework to use.

Thursday, June 18

Java One Thursday: General Session

I oversleep today and missed most of the general session. I did see a partial demo of something being worked by IBM that seems to be integrating your version control, bug tracker, and build system into a more integrated development environment. I believe one of the speakers referred it as bringing the team to the tools. And of course since this is coming from IBM, this new IDE is built on top of Eclipse.

Visual Basic And The Java Platform

This session introduced Project Simplice. According to the speakers, Project Simplice intends to attract Visual Basic programmer to the Java camp. Project Simplice is inspired by Visual Basic and compiles Basic code to a Java class file that runs in the JVM. Basically, this makes the Java platform easily available and simple to use for Basic programmer. The whole project is still at an early stage. The project is not available to the general public at this time. That said they Project Simplice team made some impressive demos.

As a demo, the Project Simplice imported the Visual Basic code for a calculator example written by Microsoft with the original copyright. Once imported the session speaker hit the build button to launch a program, originally original written for the Windows platform, but now running on JVM power. Wow. Now that is some impressive compiler magic. In NetBeans, you can edit the UI in the UI Builder and update the code and have the same context help, color highlighting, and language support that you come to expect from NetBeans.

What was hard to believe was that an exe fractal application written in Visual Basic for the Windows platform ran considerably slower than when that same code was compiled and ran in the JVM. There was general disbelief and awe when that Windows version ran over 10 times slower.

It seems that Project Simplice, or whatever Sun's marketing/naming team eventually name it, is intended as a language that can be used to teach programming concepts to beginners.

Simplify Enterprise Development With Scripting

This session was all over the map in terms of scripting in an enterprise environment. At first it started as yet another Groovy session, then it turned into a general JSR 223 talk. JSR 223 intends to bring scripting to the Java platform in the next major release. What I got most from this session was some additional feature in Groovy such a GStrings. GStrings are Groovy strings

that allow you to embed a variable in a string to insert its value. Groovy also provides a poor mans ORM in the form of DataSet. I also like that fact that you can declare properties in Groovy with an @property annotation, you don't have to define setter and getter methods for simple properties.

Enterprise JavaBeans 3.0

I went Enterprise JavaBananas over this session. The original goal of the EJB specification was to simplify the enterprise application development process but what was delivered in the form of the EJB 2.x spec was a powerful yet complex technology. In EJB 2.x, the developer codes against a set of APIs intended for and required by the container.

In EJB 3.0, session and message beans are plain old POJOs with descriptive annotations. The annotations demark the beans life cycle. For example, a stateless session bean is marked with a @stateless annotation. A stateful session bean is marked as @Stateful. A bean's pre-destroy life cycle event method is denoted with the @PreDestroy annotation. The good news for EJB 2.x is that that existing applications will be forward compatible, including the deployment descriptors. For the most part, deployment descriptors are not needed in the new specification, except for certain situations, but you can still use them as you have.

Once phrase that I have already heard twice during this conference is configuration by exception. I heard it again in this session. Having experienced Ruby on Rails I know of configuration by convention. How do you configure by exception? I don't know yet but this is something to look into if you are going to be deploying your application on an EJB 3.0 container.

I was once part of a large enterprise application project. I thought that if I ever worked on an EJB project it would be too soon. Now I can't wait.

Desktop Patterns And Data Binding

This was another session by Karsten Lentzsch, founder of JGoodies. JGoodies has a data binding project and right at the start of the session Karsten cautioned those in attendance on its use. He did this not because the project is in a horrible state but because the domain has been difficult to address, in general. There were several talks during this conference that covered the issue of data binding. At this time there are several data binding projects but nothing that will solve all of the problems all of the time. According to Karsten, some projects that try to address data binding are the SwingLabs, Eclipse 3.2, and JClient from Oracle. I have to say that I am not familiar with these projects in this regard so I can't comment on this further.

During this session, Karsten also went over the Anonymous View, the Model View Presenter, and the Presentation Model pattern.

Extreme GUI Makeover: Lookin' Better

This session was more like Queer Eye for the UI Guy. The presenters, Scott Violet, Shannon Hickey, and Romain Guy (all from Sun Microsystems) spiced up a mail client not much unlike Thunderbird. It truly was as if Thunderbird got a makeover. The presenters did some very AJAXy type of UI enhancements to a Swing app, such as fade ins and outs, presenting tabular data as rich lists with images, and some animation. If you want to spice up your application you can do the following: use the Timing Framework for fade in and out animations, make use of gradients, use alternative colors for rows within a table, enbrace white space (already mentioned in Dos and Dont's For Swing Apps), use the glass pane, and add some drop shadows and translucency.

If you are interested in making over your UI, here are some projects you should look into: SwingLabs and JH Labs. Here are some class names that you might also look into: TimingController, ShadowFactory, JXCollapsablePane, and BasicGradientPainter.

After this session I felt like asking these peeps if they would pimp my UI.

AJAX Smackdown

My last session of the day was a panel discussion covering Dojo, Flex, DWR, JSF, and Swing/Java WebStart. The moderator asked a really good question. Does all the excitement over AJAX promote JavaScript as a first class language? Now, AJAX is a client pull mechanism and the moderator queried the panelist into a server push technology which some think might be needed in the future.

Chet Haase joked that the Swing API doubles as an IQ test.

The Adobe representative said that Flex is a free Flash based programming model that is more conventional to developers as oppose to the Flash timeline.

I was mostly interested in having the Swing perspective of this whole AJAX/Rich Internet application discussion. Responding to the argument that Swing is hard to learn and that it is hard to develop good UIs, Chet Haase joked that the Swing API doubles as an IQ test.

It might be worth mentioning that with HTML a layout might seem easy with all those WYSIWYG editors. Java until recently with NetBeans' Matisse didn't have a great UI Builder freely available. The key difference with HTML is that with Swing you can position components absolutely with pixel perfect precision. In HTML the best you can do is give the browser hints as to position, not edicts.

This session also touched the issue of data binding which was covered in depth in the Desktop Patterns And Data Binding session. The key idea behind data binding is that it is important to have a consistent way in the language to be able to reach out and touch your data from the UI.

Java Champions

I went to the Java Champions Bird of a Feather (BOF) just to kill some time before the Grails BOF. It is interesting, just before attending this BOF I had read the blog entry by Yakov Fain about his impressions regarding the AJAX Smackdown session (see above). Yakov is a Java Champion from the New York/New Jersey area. During this BOF, Yakov talked about the obvious benefits to being a Sun recognized Java Champion. In regards to the negatives of such recognition he said, "My clients expect me to do miracles, why? Because I'm a Java Champion." But Yakov admitted that he is a mere mortal programmer, with other priorities and a family. To those that have the passion Yakov said, "If you want to be a leader, it's easy. Just be the one."

Rapid Web Development With Grails

In this session I got to see a little bit more under the covers of Grails, previously known as Groovy on Rails. Grails is built on Spring IoC, MVC, and WebFlow. It also makes good use of Hibernate and SiteMesh. I am currently working on a few Rails projects and have a good background with that framework and with this experience fresh in my mind I have to say that I am impressed by Grails Server Pages aka GSP. The Grails template mechanism, the heart of GSP, supports dynamic tag libraries. As far as I know tag libraries are missing from the latest release of Rails, and in the very opinionated DHH thinks that erb is all that a rails developers needs. My Rails rthml files have a bunch of scriptlet code that is getting on my nerves and their is no way around the <%= @foobar %> scriptlet. In my humble opinion, Grails template is far better than Rails. But I guess that is just a matter of taste.

Some of those in attendance did note some objections to how you define your model in Grails. In Rails your model extends ActiveRecord and consists of just two lines of code, in the simplest case. In Gails your model doesn't extend any given class and needs to define properties for the table columns. These differences are rooted in philosophy rather than technology. As a hacking project for anyone that is interested, it should be easy to hack a Grails version of the ActiveRecord if you feel that is what you need.

According to the speaker, Grails is not just for 'Greenfield' applications. The speaker was willing to put money to quell questions regarding performance because the Groovy code that makes up Grails is a small part of the application. The Grails platform is backed by the performance of Hibernate, Spring IoC, etc. that make up a large part of the framework. In Grails, your Groovy code is just a facade.

Wednesday, June 17

Java One Friday: General Session

Scott McNealy made an appearance during the last general session of JavaOne 2006. He joked about Jonathan Schwartz's ponytail by saying that all of management is trying to grow one. Now, I have to say that was funny but there is nothing funnier (at first then it gets really annoying) than a bunch of nerds and techies cracking up at some bad jokes. There is a lot of snorts and unconventional sounding laughter when the audience is made up of mostly male geeks. I felt like programmatically shutting the lips of the guy next to me, because Java is everywhere and everything you do in JavaOne involves code. We all run on Java!

"You are cursed with the opportunity to make a difference in this planet."

The hall was jam packed with thousands of developers that had come here to listen to Scott and James Grosling. There was a powerful techie vibe throughout the place that kindled something in me. I mean, some of the world's best Java minds, hackers, and developers were sitting in this hall, and I was there too!!

Scott said that we are living in the Participation Age where users are not only consuming content but also producing and sharing it. We as a species have evolved through the Ice Age, the Industrial Age, the Information Age, and now have arrived at the Participation Age. I can only imagine what is the next age. The Global Warming Age? Maybe. Scott made references of gobal warming during his short talk. Speaking to all those present he said, "You are cursed with the opportunity to make a difference in this planet."

James Grosling, like a true hacker at heart, spent his time demoing some projects. In particular I was interested in Project Jackpot. The demo didn't work just right because a goat wasn't sacrificed to the to demo demigods but from what I gather Project Jackpot is a powerful refactoring tool that can modify your whole code base. For example, you can configure your IDE to replace a deprecated method call from your code base with the correct snippet of code. This sounds like something to keep an eye out for.

Javax Scripting

The Java community is really excited about JSR 223, which will bring scripting to the Java platform. My first session of the day went over the new Scripting API in Java SE 6, the next major release of Java. By the time I entered this session I had already attended several sessions that focused on the Scriptability of the Java platform so there wasn't a lot of new information in this session. What I noted from this session was the large number of scripting languages now available to the average Java developer. Scheme, Pnut, JRuby, Jython, JavaScript, and Groovy are just some of the script languages available

in the JVM. I also discovered that Java SE 6 will introduce a new command line application, jrunscript.exe, that will be used to invoke a script file from the command line. At this point, jrunscript.exe in still designated as experimental and is intended to be available only with the next major release of the JDK.

Rich Client Platforms

The Test Driving the Rich Client Platform session by Mikael Boman was really informative. In my mind, one of the better sessions this year. I have been looking into the Eclipse RCP and have also been reading a lot about the NetBeans Platform. In this session, Mikael discussed the NetBeans Platform, the Eclipse RCP, the Spring RCP, and the remaining alternatives for building a rich client application.

A Rich Client Platform like the Eclipse RCP helps an application developer by proving a proven application UI framework with most of the plumbing already wired. The Eclipse RCP has a large user base and has been well tested by a myriad of applications. By getting all this for free you, as an application developer, have more time to focus on the business logic. If you think about it, the Eclipse IDE is just a collection of plug-ins. The IBM RAD tool is basically Eclipse with a ton of plug-ins. As a positive note, the Eclipse RCP counts on the SWT native UI library and hundreds of available plug-ins. The SWT is also one of the negative points for the Eclipse RCP. SWT is a competing technology to Swing and it is not 100% Java. According to Mikael, you can use Swing with an Eclipse RCP application but this will introduce some difficulty in the configuration.

According to Mikael, the NetBeans Plaform benefits from being 100% Java. A ton of existing modules already exist for the NetBeans Platform that you might be able to reuse in your own application. As for the negative aspects of the NetBeans Platform you can include the lack of documentation and the complexity of the platform.

The Spring RCP provides data binding and validation to the framework. The Spring RCP supports Swing and is feature rich. The problem with Spring RCP is that there is no documentation available for it at this time. The Spring RCP framework is not mature and at this time if you use it you will have to do a lot of hand coding. Basically Mikael recommends that only the brave use the Spring RCP.

Mikael's final conclusion was to "build on what you know." If you are using Eclipse and are developing SWT then the Eclipse RCP is for you. If you are developing Swing then maybe the NetBeans Platform will be a viable option.

I stopped at the JavaOne bookstore after this session and noticed that you can get started with the Eclipse RCP with the help of three books. The SWT: The Standard Widget Toolkit, Eclipse: Rich Client Platform, and Eclipse: Building Commercial-Quality Plug-ins. These books are from the Eclipse Series by Addison-Wesley. I didn't see any books that will help you get started with the NetBeans Platform and that to me speaks volumes.

Using UML 2.0

I attended the How to Represent the Architecture of Your Enterprise Application Using UML 2.0 and More session on the last day of JavaOne 2006. According to the speaker, UML 2.0 provides "a lot of flexibility which is an euphemism for ambiguity." A diagram is not enough. The speaker recommended that every diagram have a key. Along with your diagrams and key, add a glossary and acronym list. Record the rationale behind your design decisions including relevant rejected alternatives. It is also useful to gather the requirements and design constraints in these documents. To facilitate design and analysis documents, standardize on templates and use them. I myself have written a lot of use case documents and feature specifications and when I use a template I just fill in the gaps in the template.

API Desing

I attended the How to Write APIs That Will Stand the Test of Time session on the last day of JavaOne 2006. The speakers were from the NetBeans team and they stated that everyone is in the API business. Today a typical application is not built by hand but assembled with a hodgepodge of open source projects, each with its own API. Since these guys are from the NetBeans team they recommended that an application be broken down into modular chunks, a la NetBeans modules or Eclipse Plug-ins. If your application is broken down into modular chunks then each module can have a separate team, life cycle, and/or schedule. This approach required a dependency management and I feel that this advice only works if you application is complex and your development team large.

An API is used to communicate not with the machine but with people, albeit developers. We all know that an API is a contract between the service and the client. The speakers said that an API is more than a contract, it is a promise. As a developer you need to make an API, as a promise, that you can keep.

Here are some quick tips from this session. Do not expose more than you have to. Factory methods gives you more freedom, because you are not tied to a specific implementation. Avoid calling foreign code in critical sections. An API is not just a method signature, it is a contract. Design to last.

Personally, I believe that a good API is one that is not noticed, that is not intrusive, that is self descriptive, that rolls off the tongue, and that doesn't require me to remember anything useless or require me to bend backwards to use.

Copyright