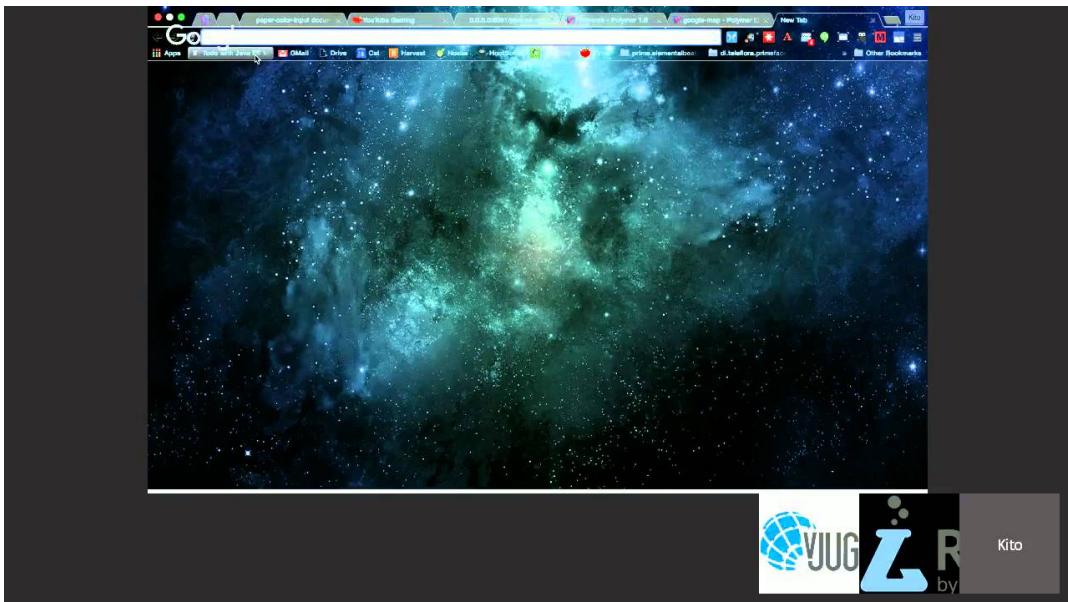


# Table of Contents

- [Modern Web Apps with HTML5 Web Components, Polymer, and Java EE MVC 1.0](#)
- [Streams: The Real Powerhouse in Java 8](#)
- [Programming Language Development Made Easy](#)
- [Mutation testing with pitest](#)
- [Getting C/C++ performance out of Java](#)
- [Java 8 in Anger!](#)
- [Java Profiling from the Ground Up!](#)
- [Generics: Past, Present and Future](#)
- [Flavors of Java Concurrency](#)
- [Resilience is by design](#)
- [Java Concurrency Under the Hood](#)
- [Getting Started with Minecraft Modding](#)
- [Value in Relationships – How Graphs make databases fun again](#)
- [So why would I use a distributed database like Apache Cassandra?](#)
- [Live From Devoxx UK Hackergarten: From vJUG virtuality to Devoxx UK real](#)
- [Live From Devoxx UK: Apache TomEE from Dev to Ops](#)
- [State of the art data access with Spring Data](#)
- [Gradle: hot or not?](#)
- [Java byte code in practice](#)
- [Effective IDE Usage](#)
- [Building “Bootiful” Microservices with Spring Cloud](#)
- [Architecting Large Enterprise Java Projects](#)
- [JavaLand Session: How is Java/JVM built?](#)
- [JavaLand Session: What’s coming in Java.Next?](#)
- [Building Modular Java Applications in the Cloud](#)
- [Java Memory Model Pragmatics](#)
- [The Live Reflection Madness](#)
- [Package your Java EE application using Docker and Kubernetes](#)
- [jOOQ: Get Back in Control of Your SQL](#)
- [Java and the Wave Glider, by James Gosling](#)
- [Scala for Java Developers](#)
- [Kotlin for Java Developers](#)
- [Ceylon for Java Developers](#)
- [Groovy for Java Developers](#)
- [Building the Internet of Things with Java](#)
- [Reactive Programming: Creating highly responsive applications](#)
- [Shaping Java’s future & vJUG party!](#)
- [HTML5, AngularJS, Groovy, Java and MongoDB all together – what could go wrong?](#)
- [Opinionated JavaFX 8](#)
- [3 years of backend testing at Shazam \[the stuff we got wrong\]](#)
- [Pragmatic Functional Refactoring with Java 8](#)
- [Highly Strung: Understanding your Type System](#)
- [Testing and Refactoring Legacy Code](#)
- [vJUG panel: Review of 2164 Survey Responses on Java Tools and Technology](#)
- [Java Classloaders: The good, the bad and the WTF.](#)
- [vJUG Panel: What do the Oracle/Google shenanigans mean to the Java Developer?](#)
- [Netty – The async event-driven network application framework](#)
- [Evolving code without breaking compatibility](#)
- [Building Bootiful Applications with Spring Boot](#)
- [Java 8 Parallel Streams Workshop](#)
- [Project Lambda: Functional Prog. Constructs and Simpler Concurrency in Java SE 8](#)

- [WebSocket Applications using Java EE 7](#)
- [Comparing JVM Web Frameworks](#)
- [55 New Features in Java SE 8](#)
- [How To Do Kick-Ass Software Development](#)
- [Getting started with Java EE 7](#)
- [Don't be that guy! Developer Security Awareness](#)
- [Drive-by Contributions](#)
- [Design is a Process, not a Document](#)

# Modern Web Apps with HTML5 Web Components, Polymer, and Java EE MVC 1.0



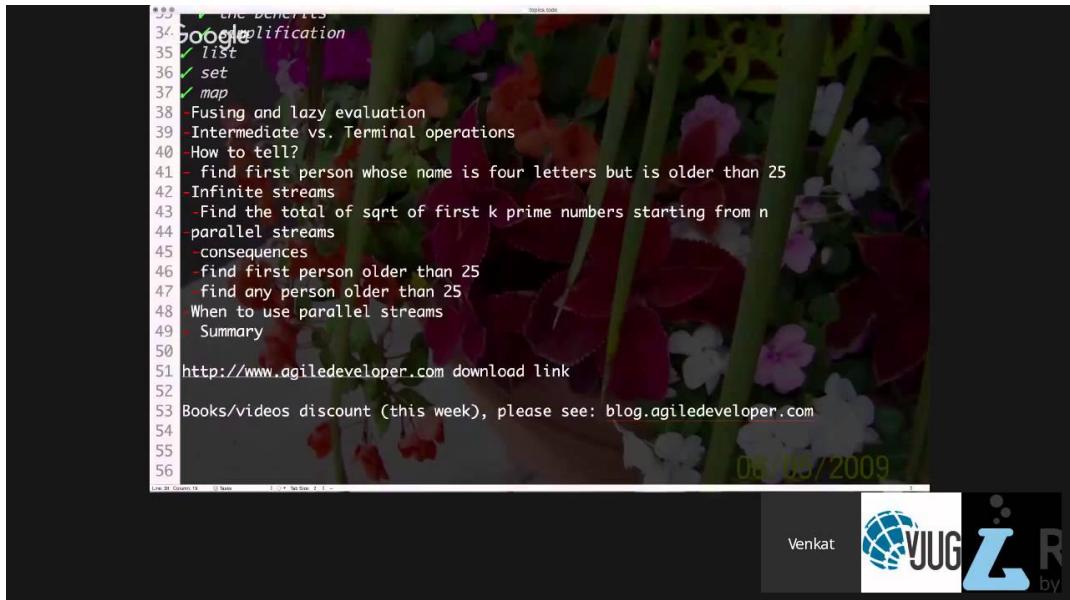
In the upcoming Java EE 8 release, there is a new kid on the block: Model-View-Controller (MVC) 1.0. It's not a replacement for the other web framework, JavaServer Faces (JSF), but it offers an alternative, more traditional, action-based approach more like Spring MVC. Since MVC doesn't provide a component model like JSF, you can use it with any front-end technology you like. One of the emerging standards for building rich UIs in the browser is HTML5 Web Components, and Google's Polymer project, which implements it and provides additional features for building powerful applications. This session shows how to build a modern web application using these bleeding edge technologies together.



<http://virtualjug.com/?p=1915>



# Streams: The Real Powerhouse in Java 8



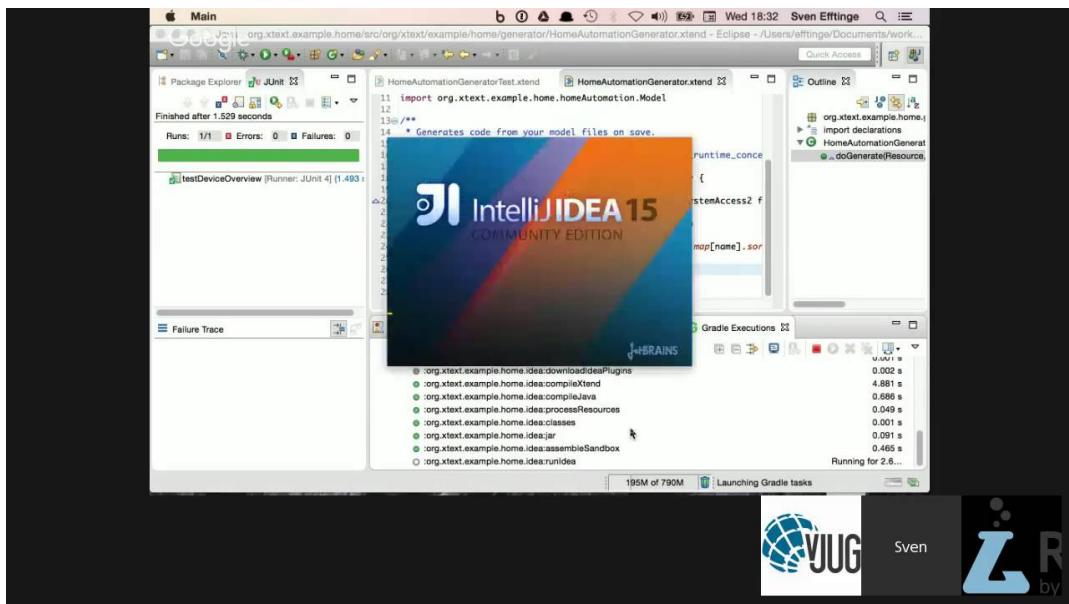
At first glance, we get excited about lambdas. But soon, as we begin to program in Java 8, we begin to realize that the real power is packed into the streams. This presentation focuses on streams, discusses what they are, and looks at how they differ from the common collections. You will also learn about the usage patterns and the benefits of designing APIs with streams.



<http://virtualjug.com/?p=1908>



# Programming Language Development Made Easy



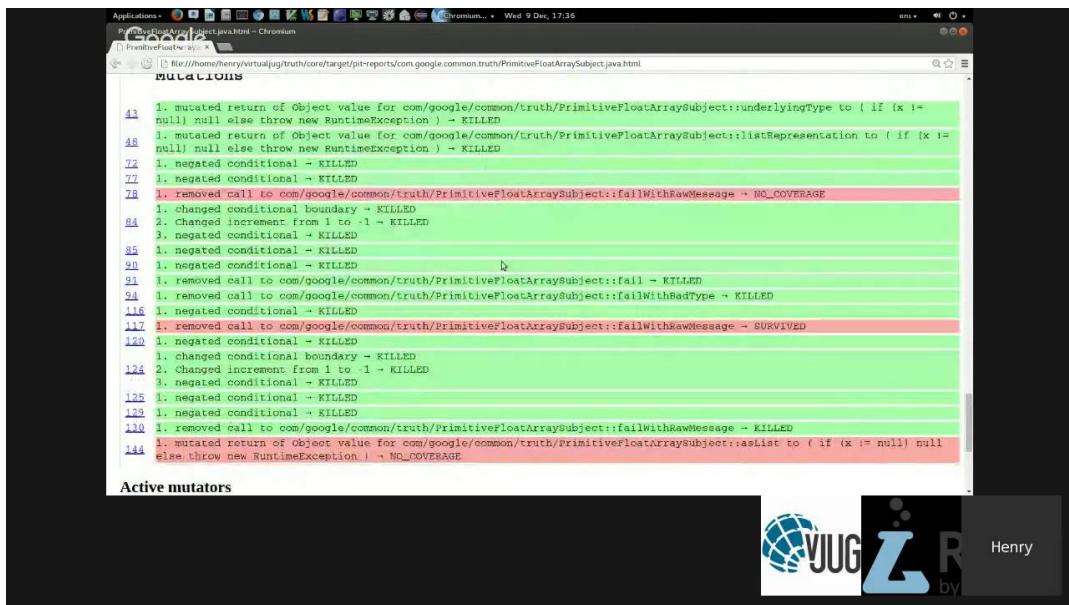
Xtext let's you build fully featured programming languages as well as simple domain specific languages (DSL) easily. Based on a syntax description in a powerful grammar language, the framework not only gives you a generated parser and AST but supports the full stack of infrastructure that is needed. This includes compiler aspects, like linkers, type checkers as well as rich editor support for Eclipse, IntelliJ and web. Also a deep integration with the common build tools Maven, Gradle, and Ant comes out of the box.



<http://virtualjug.com/?p=1894>



# Mutation testing with pitest



The screenshot shows a mutation report for the file `PrimitiveFloatArraySubject.java.html`. The report lists 144 mutations, categorized into survived (green) and killed (red) sections. The mutations are numbered 43 to 144 and include various changes like removing calls to `failWithRawMessage`, changing conditional boundaries, and negating conditions. A sidebar on the left shows 'Active mutators'.

Mutation ID	Description	Status
43	1. mutated return of Object value for com/google/common/truth/PrimitiveFloatArraySubject::underlyingType to ( if (x != null) null else throw new RuntimeException )	KILLED
48	1. mutated return of Object value for com/google/common/truth/PrimitiveFloatArraySubject::listRepresentation to ( if (x != null) null else throw new RuntimeException )	KILLED
72	1. negated conditional	KILLED
77	1. negated conditional	KILLED
78	1. removed call to com/google/common/truth/PrimitiveFloatArraySubject::failWithRawMessage - NO_COVERAGE	KILLED
84	1. changed conditional boundary	KILLED
85	1. changed increment from 1 to -1	KILLED
86	1. negated conditional	KILLED
90	1. negated conditional	KILLED
91	1. removed call to com/google/common/truth/PrimitiveFloatArraySubject::fail - KILLED	KILLED
94	1. removed call to com/google/common/truth/PrimitiveFloatArraySubject::failWithBadType - KILLED	KILLED
116	1. negated conditional	KILLED
117	1. removed call to com/google/common/truth/PrimitiveFloatArraySubject::failWithRawMessage - SURVIVED	SURVIVED
120	1. negated conditional	KILLED
124	1. changed conditional boundary	KILLED
125	2. changed increment from 1 to -1	KILLED
126	3. negated conditional	KILLED
129	1. negated conditional	KILLED
130	1. negated conditional	KILLED
138	1. removed call to com/google/common/truth/PrimitiveFloatArraySubject::failWithRawMessage - KILLED	KILLED
144	1. mutated return of object value for com/google/common/truth/PrimitiveFloatArraySubject::asList to ( if (x != null) null else throw new RuntimeException ) - NO_COVERAGE	KILLED

Sometimes it takes a little while for a good idea to catch on. A student paper written in 1971 is just now starting to influence how we approach unit testing 40 years later.



<http://virtualjug.com/?p=1892>



# Getting C/C++ performance out of Java

Google HOW DOES IT PERFORM? C24

Compare classic Java binding to binary...

	Classic Java version	Binary Java version	Improvement
Bytes used	328	39	840%
Serialization size	668	85	786%
Custom Serialization	668	40	1,670%
Time to Serialize/ Deserialize	41.1 $\mu$ S	4.17 $\mu$ S	10x
Batched Serialize/ Deserialize	12.3 $\mu$ S	44nS	280x

33



John by

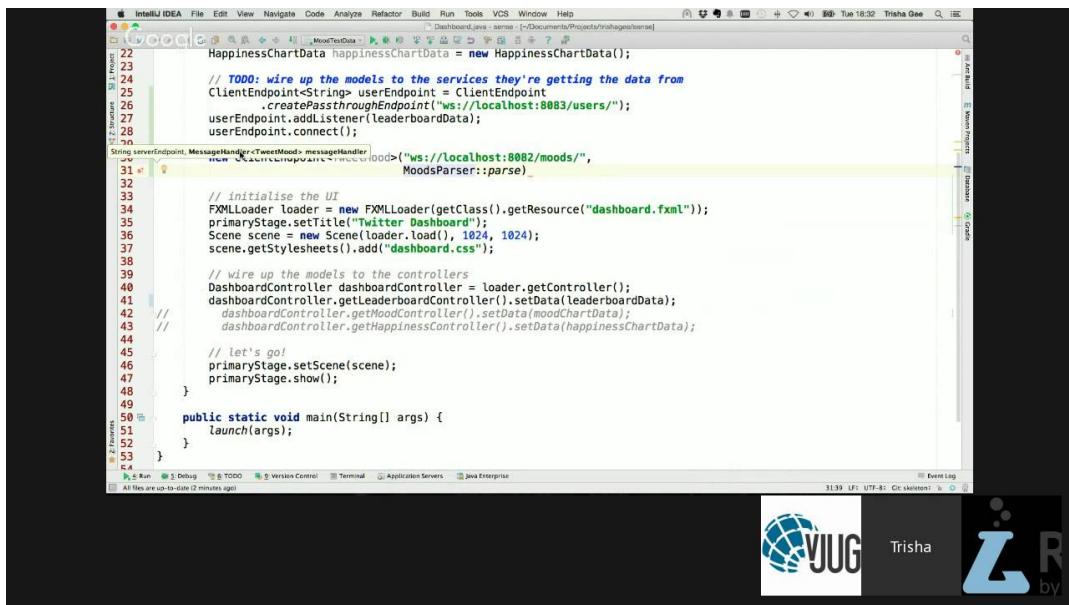
Today everything is connected, from the wearables to internet banking, every business is global and technology revolves around messaging. Data volumes have gone through the roof, “Big Data” is the buzzword of the era and we’ve been forced into distributed architectures and in-memory databases to handle the demand. We are now being held back by Java, its inefficient memory handling, and atrocious serialisation. By re-applying some of the lessons we learnt in the days of C/C++ we can significantly speed it up, and we’re not talking 50%, but 50 times.



<http://virtualjug.com/?p=1878>



# Java 8 in Anger!



The screenshot shows the IntelliJ IDEA interface with a Java file named `Dashboard.java` open. The code implements a `MoodTextData` class and a `DashboardController`. It uses Java 8 features like streams and lambdas. The code includes annotations for FXML loading and controller setup. At the bottom, there's a `public static void main` method.

```
22 // HappinessChartData happinessChartData = new HappinessChartData();
23 // TODO: wire up the models to the services they're getting the data from
24 ClientEndpoint<String> userEndpoint = ClientEndpoint
25     .createPassthroughEndpoint("ws://localhost:8083/users/");
26 userEndpoint.addHandler(leaderboardData);
27 userEndpoint.connect();
28
29 String serverEndpoint, MessageHandler<TweetMoods> messageHandler;
30 new LeaderboardController().moodChartData("ws://localhost:8082/moods/",
31     MoodsParser::parse);
32
33 // initialise the UI
34 FXMLLoader loader = new FXMLLoader(getClass().getResource("dashboard.fxml"));
35 primaryStage.setTitle("Twitter Dashboard");
36 Scene scene = new Scene(loader.load(), 1024, 1024);
37 scene.getStylesheets().add("dashboard.css");
38
39 // wire up the models to the controllers
40 DashboardController dashboardController = loader.getController();
41 dashboardController.getLeaderboardController().setData(leaderboardData);
42 // dashboardController.getMoodController().setData(moodChartData);
43 // dashboardController.getHappinessController().setData(happinessChartData);
44
45 // let's go!
46 primaryStage.setScene(scene);
47 primaryStage.show();
48 }
49
50 public static void main(String[] args) {
51     launch(args);
52 }
53 }
```

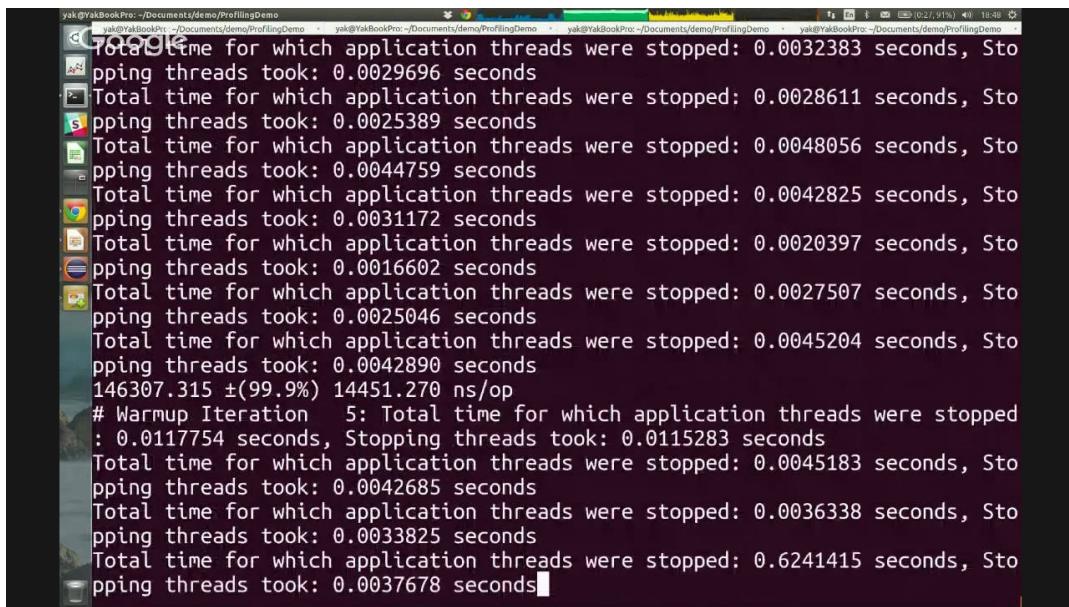
Java 9 is just around the corner, but many of us developers have yet to use Java 8 features in an application. This presentation moves beyond slide-sized examples of streams and lambdas, and shows how to build a fully working end-to-end application using minimal external dependencies and the latest version of Java.



<http://virtualjug.com/?p=1857>



# Java Profiling from the Ground Up!



A screenshot of a terminal window titled "Google" showing Java profiling output. The output includes several iterations of thread stopping times and a warmup iteration summary. The text is as follows:

```
Total time for which application threads were stopped: 0.0032383 seconds, Stopping threads took: 0.0029696 seconds
Total time for which application threads were stopped: 0.0028611 seconds, Stopping threads took: 0.0025389 seconds
Total time for which application threads were stopped: 0.0048056 seconds, Stopping threads took: 0.0044759 seconds
Total time for which application threads were stopped: 0.0042825 seconds, Stopping threads took: 0.0031172 seconds
Total time for which application threads were stopped: 0.0020397 seconds, Stopping threads took: 0.0016602 seconds
Total time for which application threads were stopped: 0.0027507 seconds, Stopping threads took: 0.0025046 seconds
Total time for which application threads were stopped: 0.0045204 seconds, Stopping threads took: 0.0042890 seconds
146307.315 ±(99.9%) 14451.270 ns/op
# Warmup Iteration 5: Total time for which application threads were stopped : 0.0117754 seconds, Stopping threads took: 0.0115283 seconds
Total time for which application threads were stopped: 0.0045183 seconds, Stopping threads took: 0.0042685 seconds
Total time for which application threads were stopped: 0.0036338 seconds, Stopping threads took: 0.0033825 seconds
Total time for which application threads were stopped: 0.6241415 seconds, Stopping threads took: 0.0037678 seconds
```

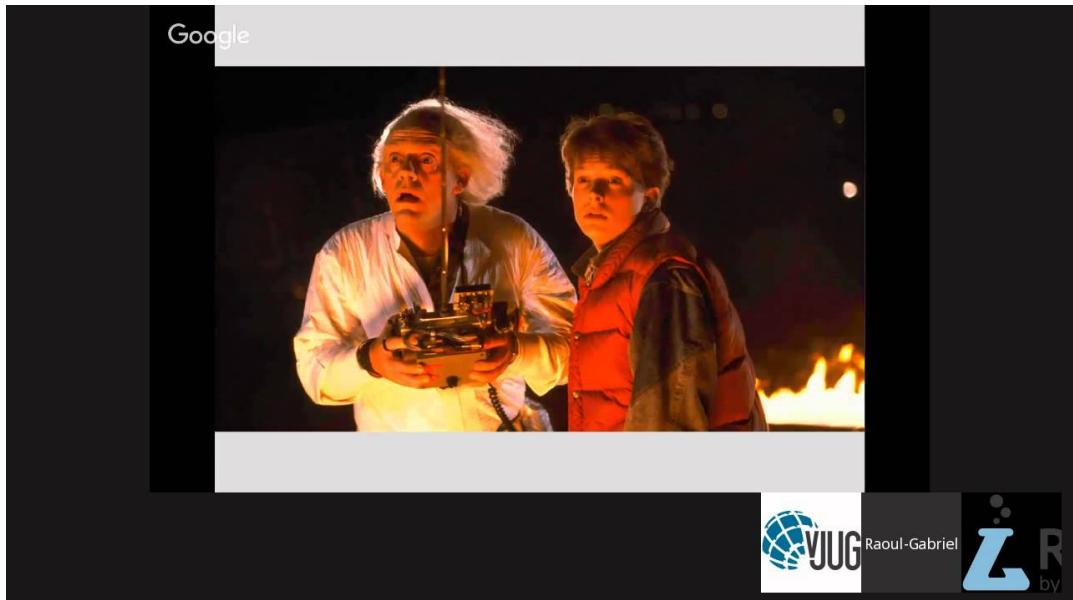
We will take a deep dive into the guts of Honest-Profiler, an unbiased sampling CPU profiler for the JVM, and into the JVM internals which enable it to work.



<http://virtualjug.com/?p=1846>



# Generics: Past, Present and Future



Generics are one of the most complex features of Java. They are often poorly understood and lead to confusing errors. Unfortunately, it won't get easier. Java 10, release planned for 2018, extends Generics. It's now time to understand generics or risk being left behind.



<http://virtualjug.com/?p=1844>



# Flavors of Java Concurrency



Writing concurrent code that is also correct is unbelievably hard. Naturally, humanity has developed a number of approaches to handle concurrency in the code, starting from basic threads that follow the hardware way to do concurrency to higher level primitives like fibers and work-stealing solutions. But which approach is the best for you?



<http://virtualjug.com/?p=1827>



# Resilience is by design



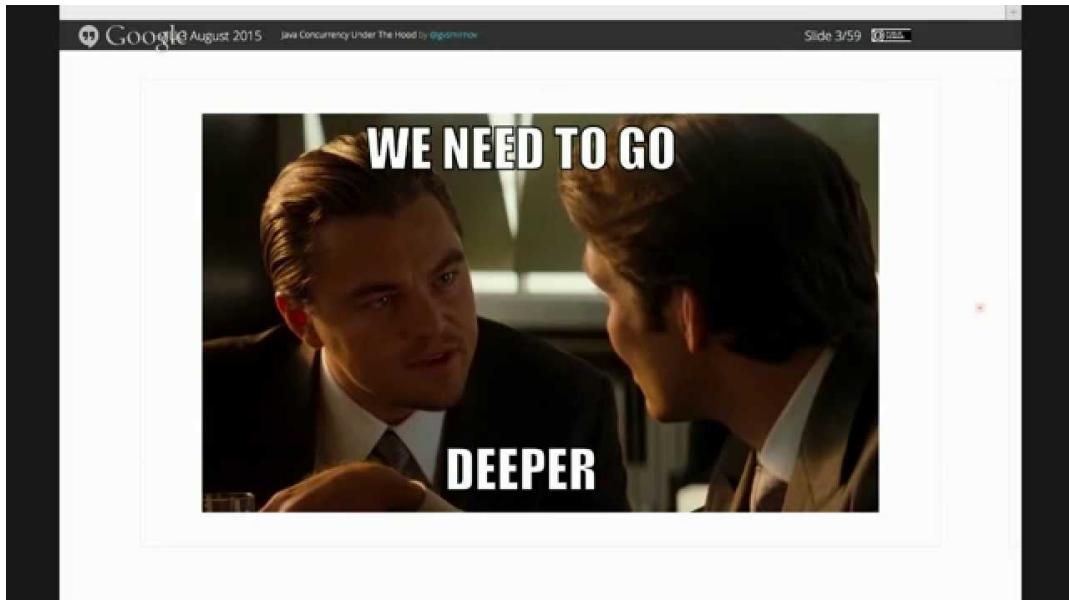
Resilience; most developers understand what the word means, at least superficially, but way too many lack a deeper understanding of what it really means in the context of the system that they are working on now. I find it really sad to see, since understanding and managing failure is more important today than ever. Outages are incredibly costly—for many definitions of cost—and can sometimes take down whole businesses. In this talk we will explore the essence of resilience. What does it really mean? What is its mechanics and characterizing traits? How do other sciences and industries manage it? We will see that everything hints at the same conclusion; there is no “happy path”, failure is an option and resilience is by design. In this talk we will explore how.



<http://virtualjug.com/?p=1807>



# Java Concurrency Under the Hood



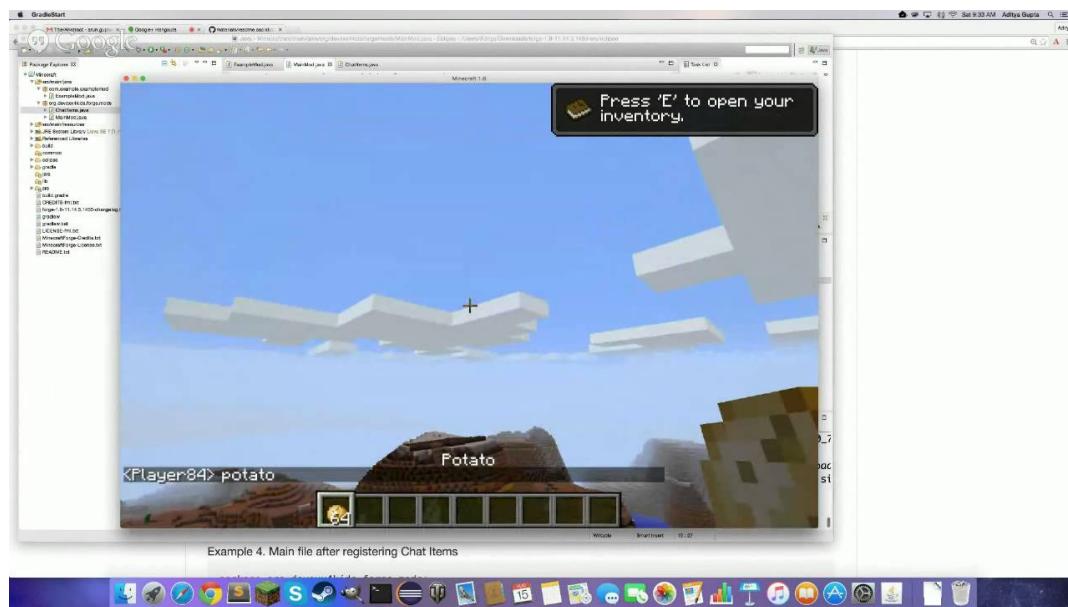
In this age when parallelism matters, being able to write proper concurrent code is paramount. While Java hides lots of implementation details by its ‘Write Once, Run Anywhere’ motto, all abstractions will eventually leak. When they do, you will have to go deeper and see how that thing actually works.



<http://virtualjug.com/?p=1799>



# Getting Started with Minecraft Modding



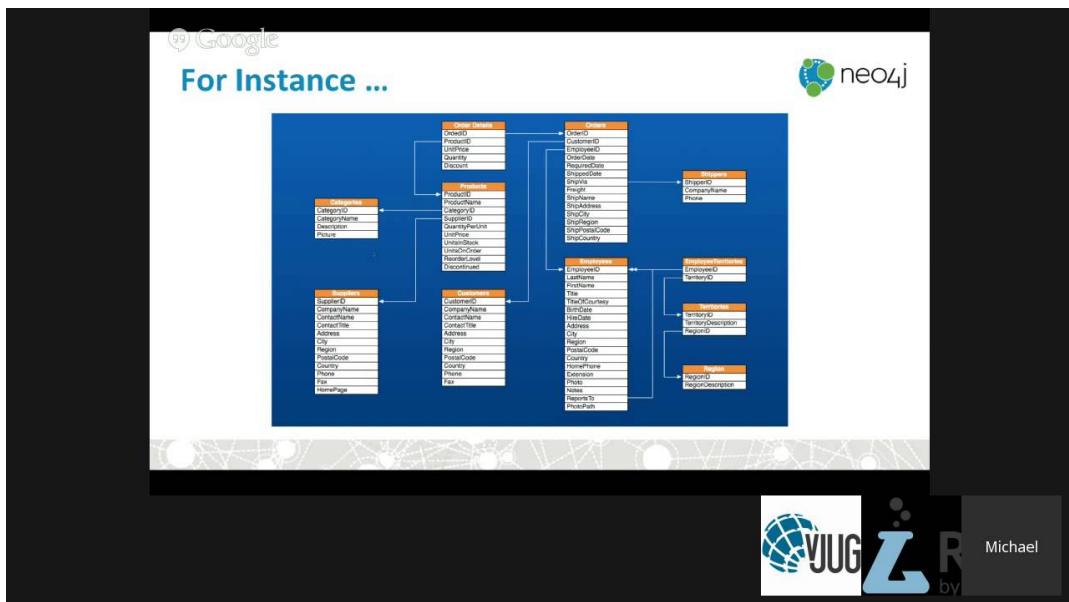
In this session, we'll show parents and kids how to get started building Minecraft mods with Minecraft Forge. We'll show you how to setup your computer with little fuss, as well as walk you through the process of creating your first mod. You'll also learn essential Java programming skills. If you're a kid searching for new ways to have fun with the game, or a parent looking to nurture your kids' creativity through code, you won't want to miss this exciting, hands-on tutorial.



<http://virtualjug.com/?p=1794>



# Value in Relationships – How Graphs make databases fun again



Looking at the world around us – society, social, science, economy and tech we can't see any isolated pieces of information. Instead everything is densely connected and a lot of the valuable information lives in the relationships between your entities. In the past and present databases always had a hard time to manage highly connected and semi-structured information in an efficient manner.



<http://virtualjug.com/?p=1745>



# So why would I use a distributed database like Apache Cassandra?



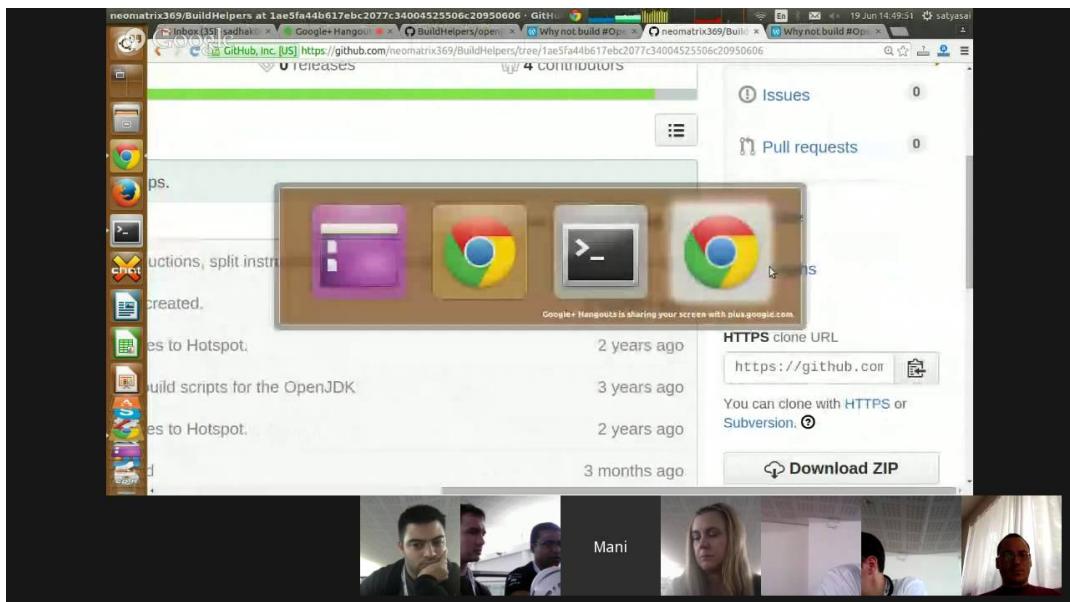
A new “database” seems to appear every other week. Most of them are “NoSQL” so they must be cool. All these new tools make it really hard for developers to cut through the fluff and know which type of data store to use and why.



<http://virtualjug.com/?p=1716>



# Live From Devoxx UK Hackergarten: From vJUG virtuality to Devoxx UK real



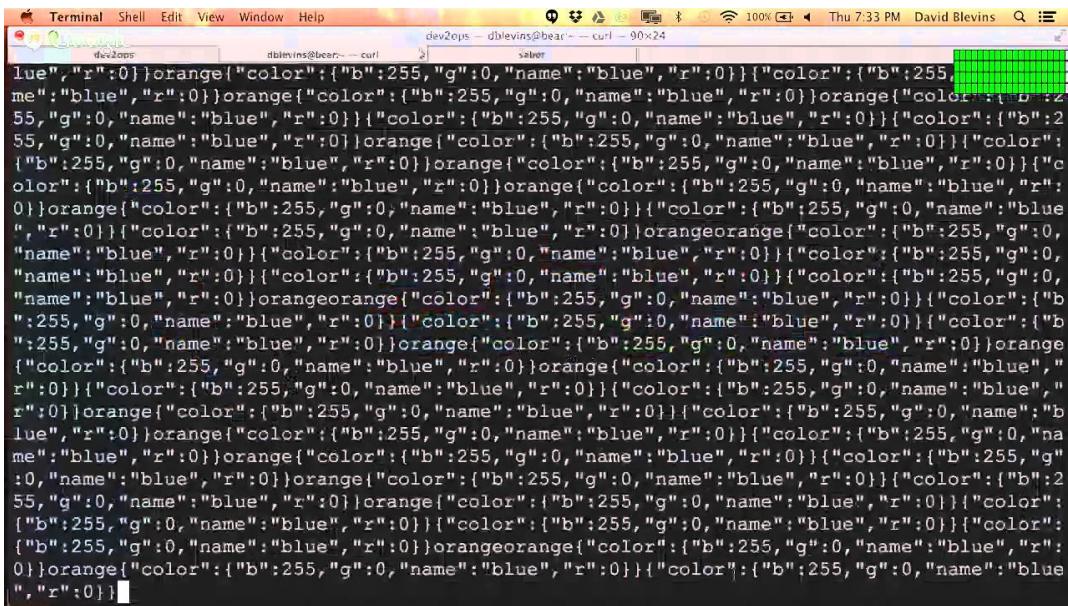
One dev: Did you know what happened at a recent Java conference few months ago ?



<http://virtualjug.com/?p=1714>



# Live From Devoxx UK: Apache TomEE from Dev to Ops



```
Terminal Shell Edit View Window Help
dev2ops - dblevins@bear: ~ curl -s http://virtualjug.com/vjUG/ -o vjUG.html
dblevins@bear: ~ curl -s http://virtualjug.com/vjUG/ -o vjUG.html
```

The terminal window shows the command `curl -s http://virtualjug.com/vjUG/ -o vjUG.html` being run. The output of the curl command is displayed in the terminal, consisting of a large amount of JSON-like data representing a color palette or similar data structure.

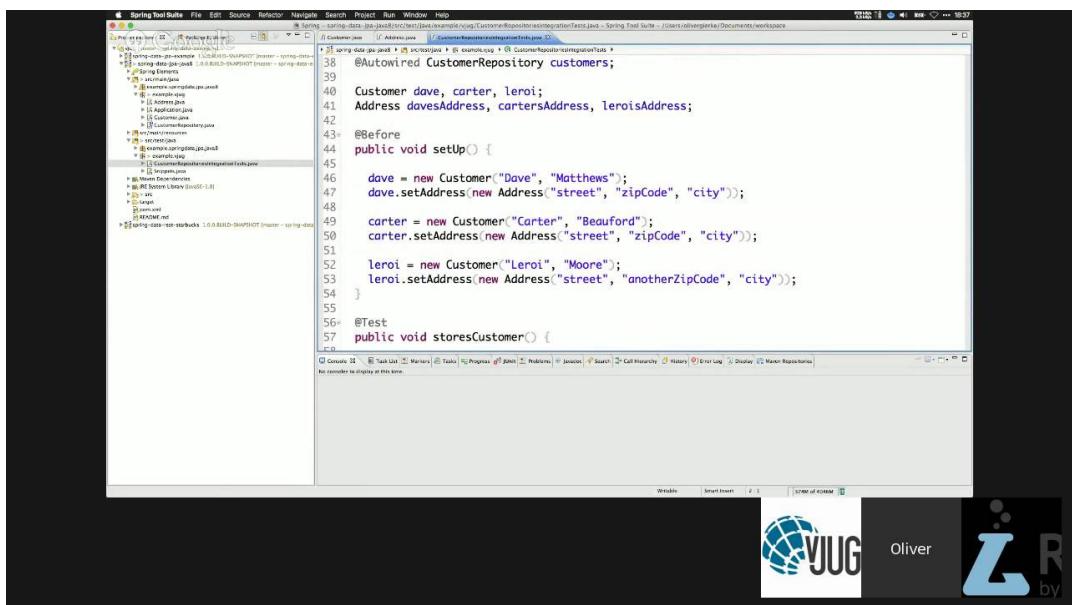
Apache TomEE is the Java EE distribution of Apache Tomcat. This live vJUG session goes beyond the basics and explores some fun features both TomEE-specific and JavaEE-portable for supercharging your application development, runtime and maintenance. Have a huge pile of DAOs? Use TomEE's abstract bean concept. Need to configure your application for many different environments? CDI and portable-extensions to the rescue. Want to create secured microservice distributions without any fuss? Nothing beats the TomEE Maven Plugin. Looking for a way to get detailed stats from your code? Hello annotation-driven monitoring support. Ever wish you could make your own management API? Check out the portable SSH Connector. The perfect session for any TomEE or Java EE enthusiast looking for cool toys for both developer and operations bliss.



<http://virtualjug.com/?p=1711>



# State of the art data access with Spring Data



Even with the invention of JPA, implementing data-access layers in Java has been a tedious job for developers, often resulting in a lot of boilerplate code. Spring Data is an umbrella project that provides a convenient and consistent interface-based programming model to implement repositories. It can be used on top of JPA as well as NoSQL stores like MongoDB and Neo4j.



<http://virtualjug.com/?p=1709>



# Gradle: hot or not?



Maven has been the preferred build tool of many Java based projects for years however times are a-changing, there's a new build tool in town and it promises to speed up build times, deliver build consistency, easier CI setup, extensibility and more. This tool is Gradle. Prominent open source projects have switched to Gradle already; organizations around the world are evaluating it too or made the switch already. So what makes Gradle tick? Join us to figure out the details! After all, it's not a "should I change to Gradle" question, rather "\_when\_ should I change to Gradle".



<http://virtualjug.com/?p=1704>



# Java byte code in practice

The screenshot shows a slide from the Java Virtual Machine Specification. It lists five bytecode instructions:

- INVOKESSTATIC** `pkg/Bar foo ()V`  
Invokes a static method.
- INVOKEVIRTUAL** `pkg/Bar foo ()V`  
Invokes the most-specific version of an inherited method on a non-interface class.
- INVOKESPECIAL** `pkg/Bar foo ()V`  
Invokes a super class's version of an inherited method.  
Invokes a "constructor method".  
Invokes a private method.  
Invokes an interface default method (Java 8).
- INVOKEINTERFACE** `pkg/Bar foo ()V`  
Invokes an interface method.  
(Similar to INVOKEVIRTUAL but without virtual method table index optimization.)
- INVOKEDYNAMIC** `foo ()V bootstrap`  
Queries the given *bootstrap method* for locating a method implementation at runtime.  
(MethodHandle: Combines a specific method and an `INVOKE*` instruction.)

At the bottom right, there is a logo for VJUG and a photo of a man named Rafael.

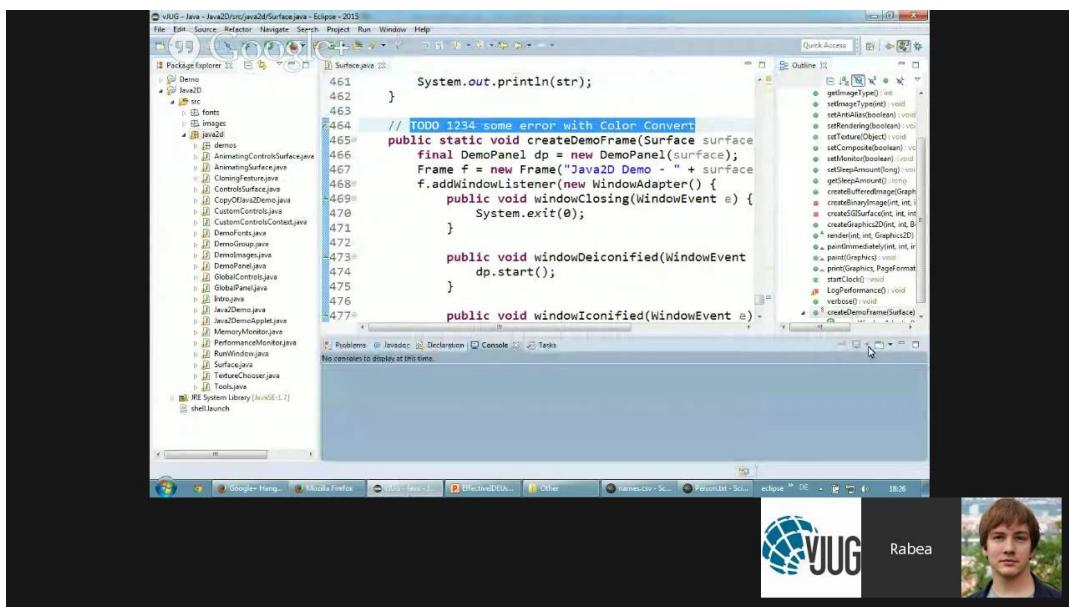
At first glance, Java byte code can appear to be some low level magic that is both hard to understand and effectively irrelevant to application developers. However, neither is true. With only little practice, Java byte code becomes easy to read and can give true insights into the functioning of a Java program. In this talk, we will cast light on compiled Java code and its interplay with the Java virtual machine. In the process, we will look into the evolution of byte code over the recent major releases with features such as dynamic method invocation which is the basis to Java 8 lambda expressions. Finally, we will learn about tools for the run time generation of Java classes and how these tools are used to build modern frameworks and libraries. Among those tools, I present Byte Buddy, an open source tool of my own efforts and an attempt to considerably simplify run time code generation in Java.  
(<http://bytebuddy.net>)



<http://virtualjug.com/?p=1673>



# Effective IDE Usage



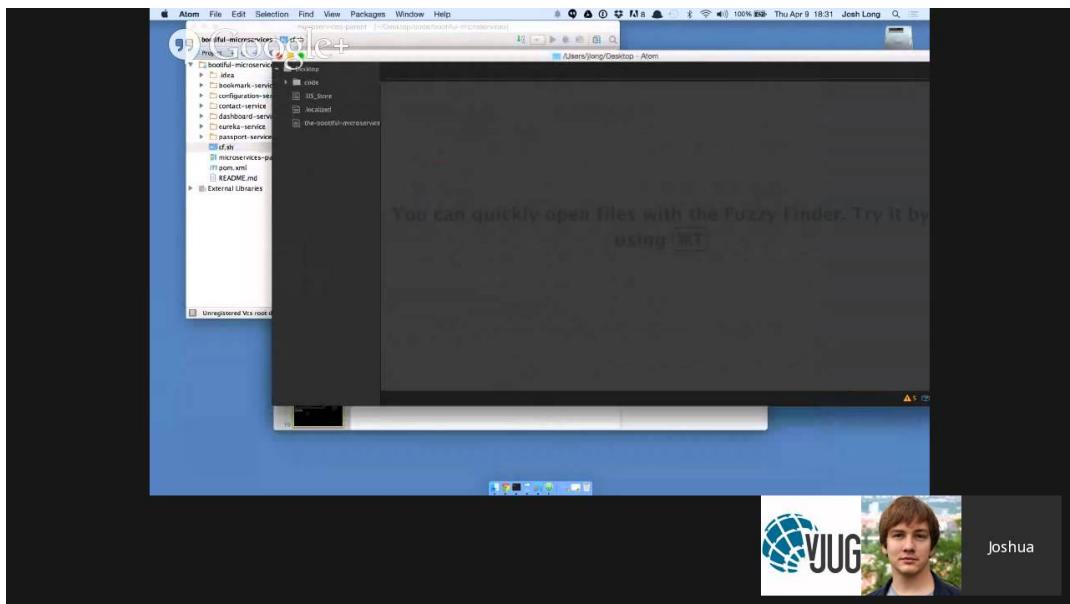
You don't want your IDE to propose `java.awt.List` as import when you need `java.util.List`? This talk will show you how to get rid of the proposal and how to use your IDE effectively to concentrate on your work.



<http://virtualjug.com/?p=1658>



# Building “Bootiful” Microservices with Spring Cloud



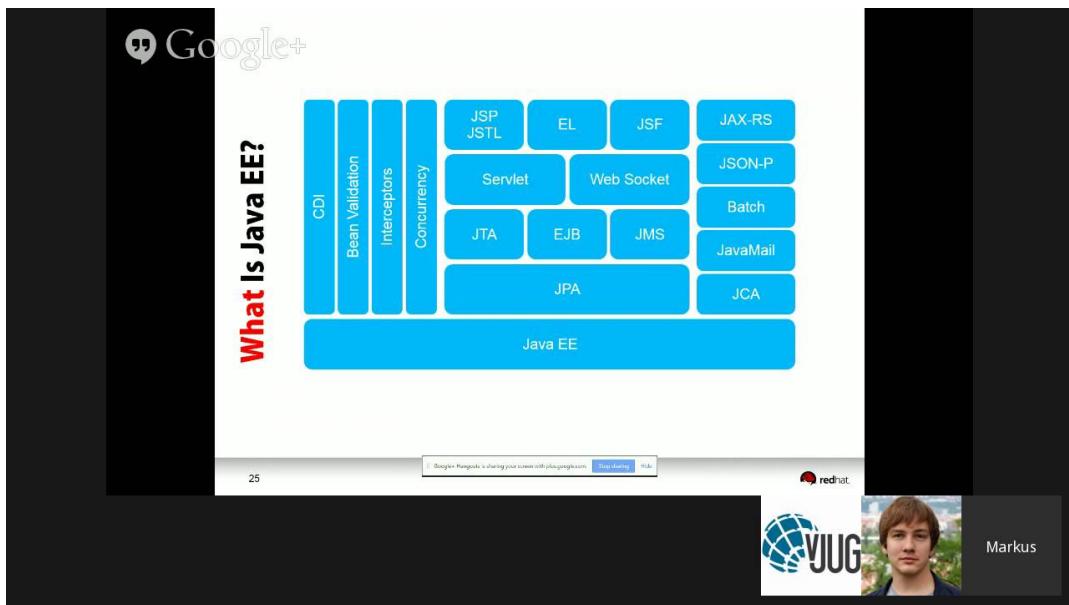
We get it already! Microservices help you build smaller, singly-focused services, quicker. They scale out. They’re more agile because individual teams can deliver them at their own pace. They work well in the cloud because they’re smaller, and benefit from elastic, horizontal scaling. But what about the complexity? There’s a cost associated with adding services and coordinating the interactions between them. In this talk, we’ll look at Spring Cloud, which builds atop Spring Boot and the Netflix OSS stack, and see how it lets you easily integrate service-discovery, security, reliability patterns like the circuit breaker, and centralized and journaled property configuration (and more) to build resilient microservices that scale.



<http://virtualjug.com/?p=1552>



# Architecting Large Enterprise Java Projects



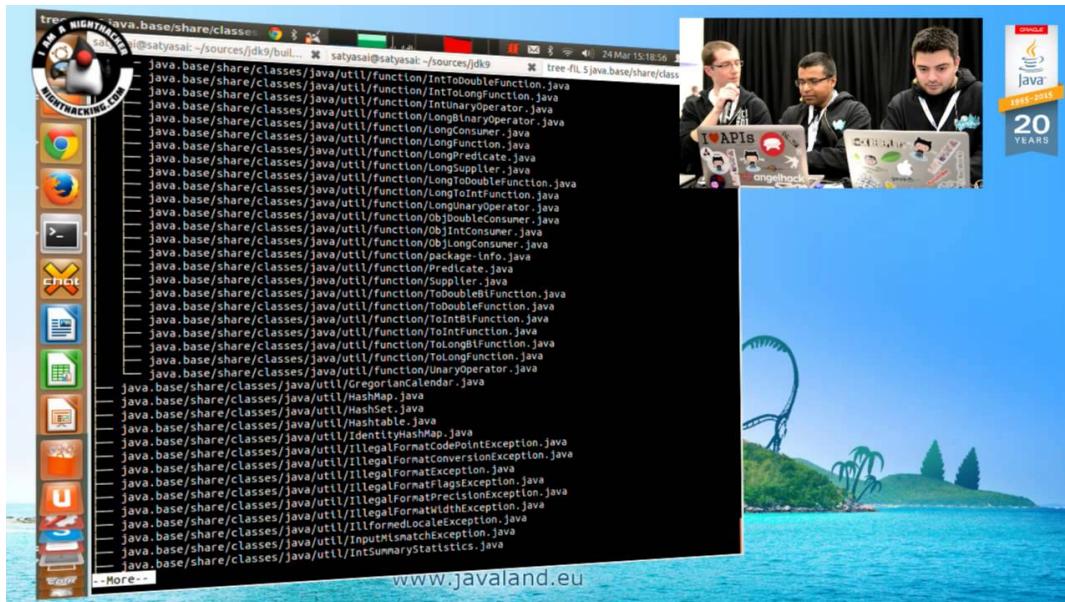
In the past I've been building component oriented applications with what I had at hand. Mostly driven by the features available in the Java EE standard to be "portable" and easy to use. Looking back this has been a perfect fit for many customers and applications. With an increasing demand for highly integrated applications which use already available services and processes from all over the place (departmental, central or even cloud services) this approach starts to feel more and more outdated. And this feel does not come from a technology perspective but from all the requirements around it. Having this in mind this post is the starting point of a series of how-to's and short tutorials which aim to showcase some more diverse ways of building (Java EE) applications that fit better into today's requirements and landscapes.



<http://virtualjug.com/?p=1546>



# JavaLand Session: How is Java/JVM built?



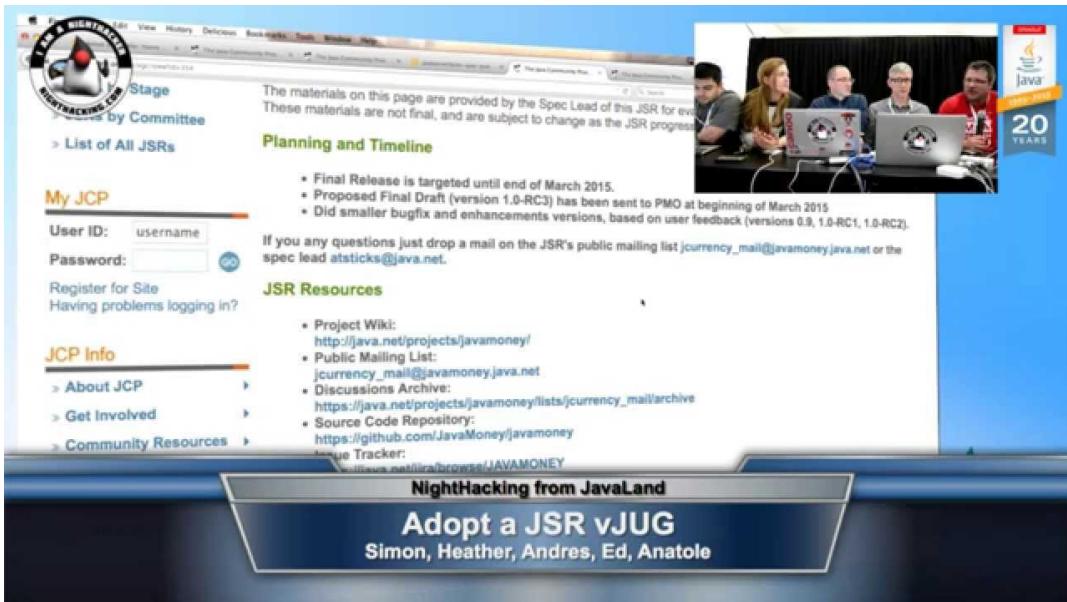
**Mr Webber developer** and **Ms Janet Java developer** are both developers who are interested in broadening their know-how of the Java platform. **Mr Webber developer** shares with **Ms Janet Java developer** conservations about Javaland, vJUG, Nighthacking and Adopt OpenJDK – a preview of their conversation.



<http://virtualjug.com/?p=1533>



# JavaLand Session: What's coming in Java.Next?



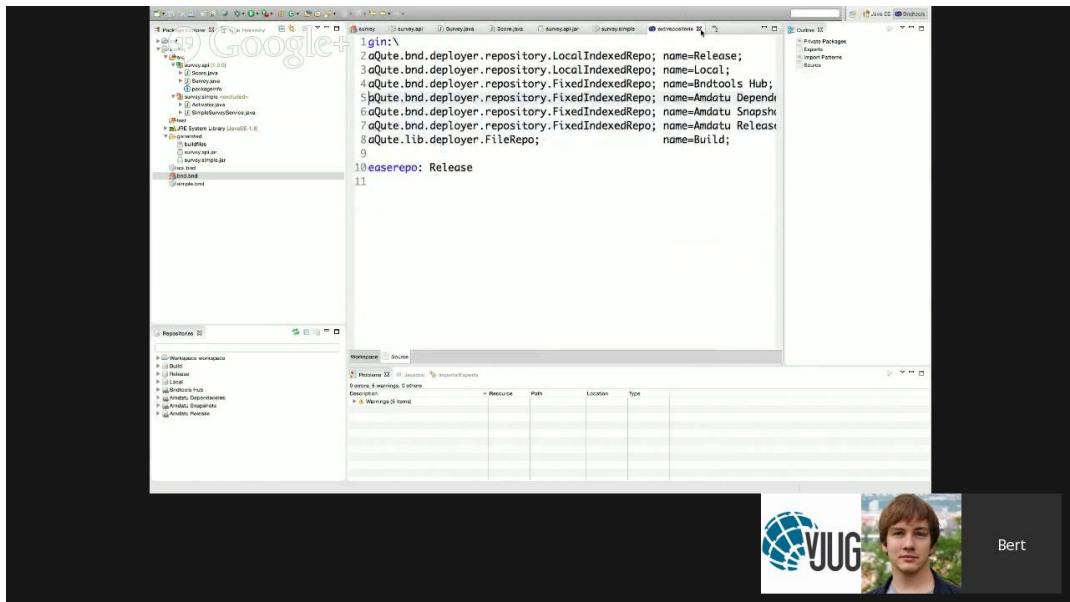
This session will take place live from the JavaLand conference in Germany on the Nighthacking stage! Learn from **Heather VanCura** how you can take part in Java technology by Adopting a JSR. This session give a brief overview of the Adopt-a-JSR program and how to participate through the Virtual JUG. We will meet and discuss with three current JCP Spec Leads to find out how their JSRs could benefit from vJUG Adopt-a-JSR participation.



<http://virtualjug.com/?p=1532>



# Building Modular Java Applications in the Cloud



Bert

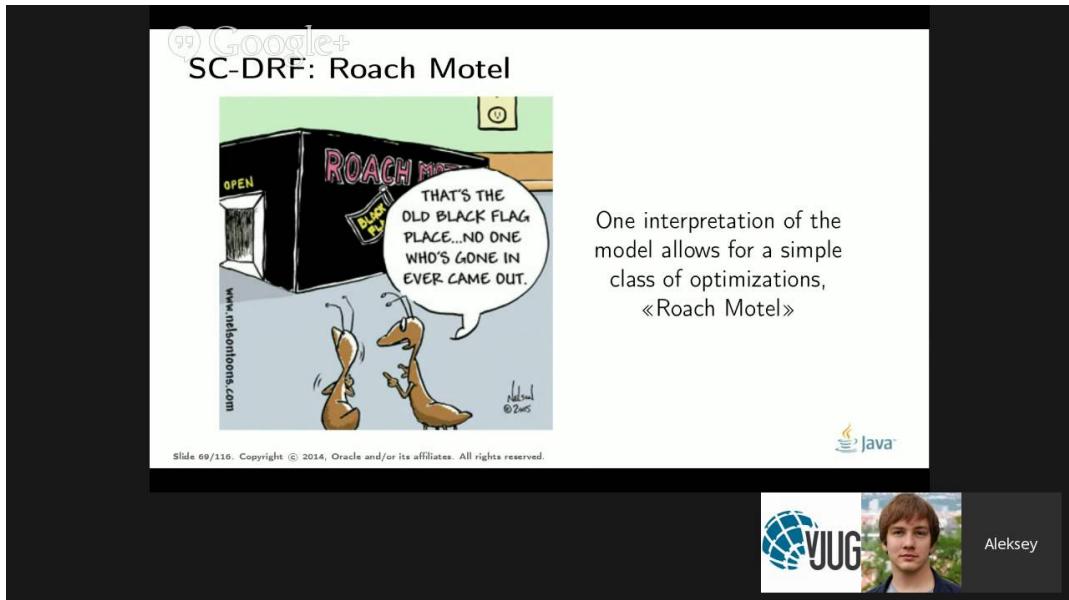
Modularity is an architectural theme that you'll hear about more and more. Being able to deal with change in a codebase is not something trivial and requires some serious thought. In this talk I will show you that it is actually pretty easy to achieve a modular architecture using OSGi, and the right set of tools. Of course everything will be demonstrated using live coding!



<http://virtualjug.com/?p=1436>



# Java Memory Model Pragmatics



The Java Memory Model is the most complicated part of Java spec that must be understood by at least library and runtime developers. Unfortunately, it is worded in such a way that it takes a few senior guys to decipher it for each other.



<http://virtualjug.com/?p=1388>



# The Live Reflection Madness



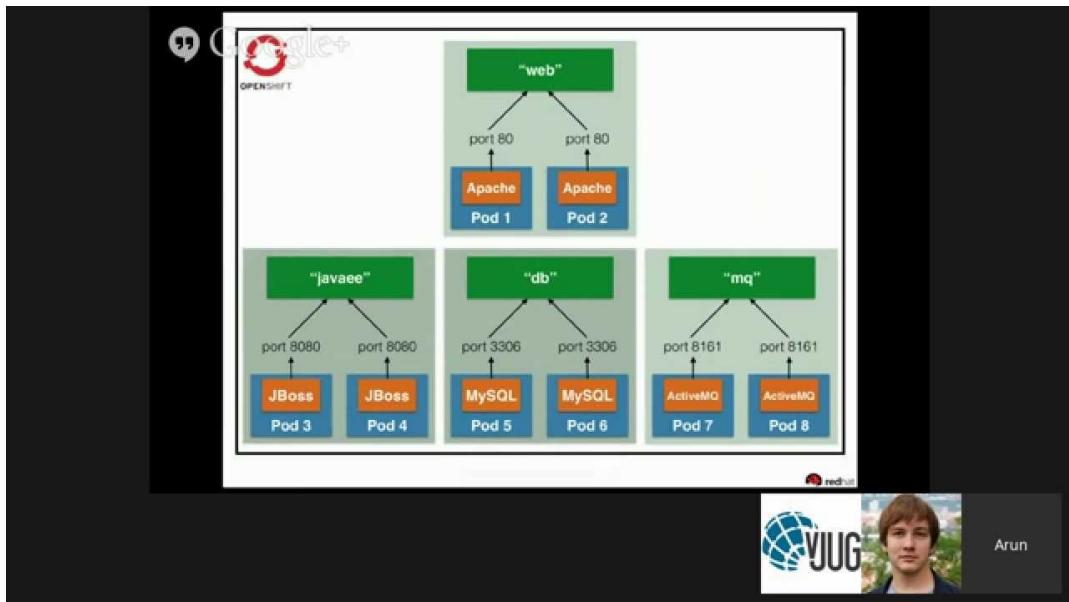
Heinz likes to compare reflection to opium. Not the perfume. The drug. In this live coding session, he will start by showing some of the powerful features available to us in Java.



<http://virtualjug.com/?p=1347>



# Package your Java EE application using Docker and Kubernetes



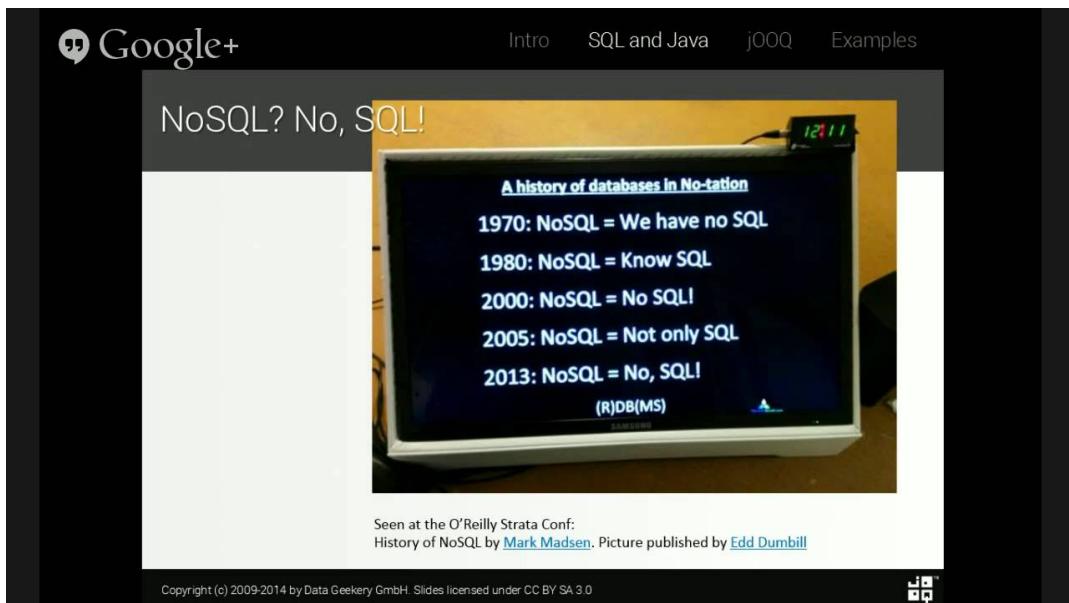
Docker simplifies software delivery by making it easy to build and share images that contain your application's operating system. It packages your application and infrastructure together, managed as one component.



<http://virtualjug.com/?p=1343>



# jOOQ: Get Back in Control of Your SQL



SQL is a powerful and highly expressive language for queries against relational databases. SQL is established, standardised and hardly challenged by alternative querying languages.



<http://virtualjug.com/?p=1337>



# Java and the Wave Glider, by James Gosling



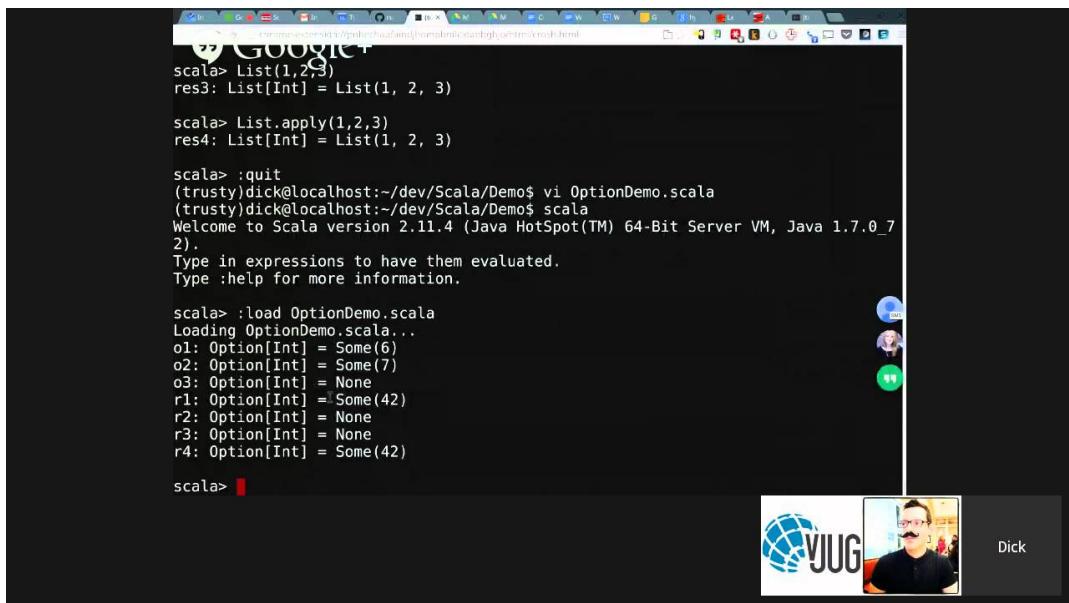
[IRC logs be be found here.](#)



<http://virtualjug.com/?p=1325>



# Scala for Java Developers



```
scala> List(1,2,3)
res3: List[Int] = List(1, 2, 3)

scala> List.apply(1,2,3)
res4: List[Int] = List(1, 2, 3)

scala> :quit
(trusty)dick@localhost:~/dev/Scala/Demo$ vi OptionDemo.scala
(trusty)dick@localhost:~/dev/Scala/Demo$ scala
Welcome to Scala version 2.11.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_7
2).
Type in expressions to have them evaluated.
Type :help for more information.

scala> :load OptionDemo.scala
Loading OptionDemo.scala...
o1: Option[Int] = Some(6)
o2: Option[Int] = Some(7)
o3: Option[Int] = None
r1: Option[Int] = Some(42)
r2: Option[Int] = None
r3: Option[Int] = None
r4: Option[Int] = Some(42)

scala> 
```

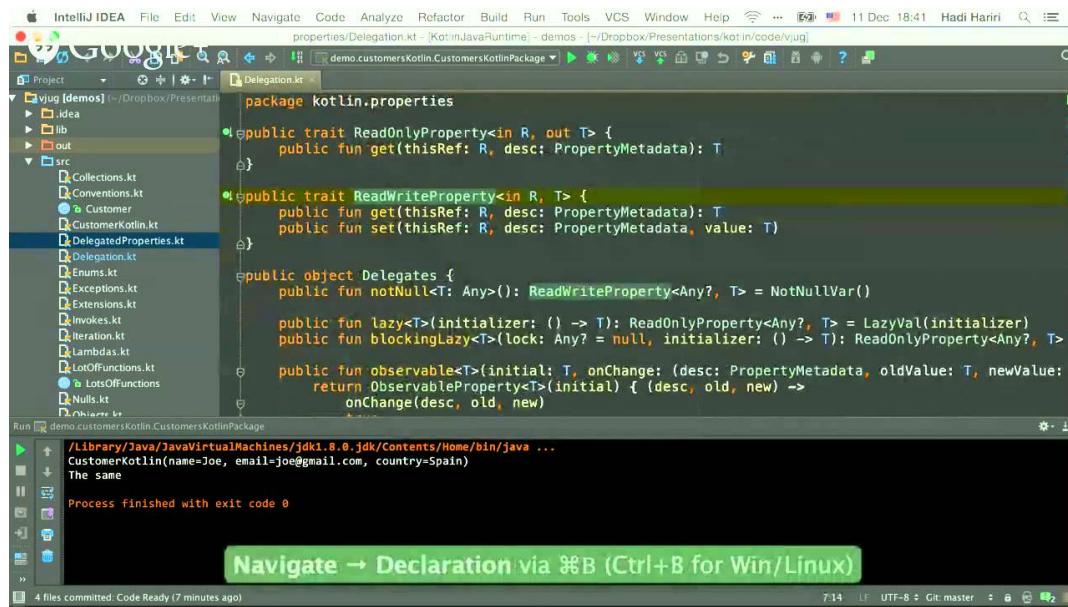
- What are the major advantages/features Scala provides
- Why should someone move from Java to Scala
- What is the future direction of Scala



<http://virtualjug.com/?p=1302>



# Kotlin for Java Developers



The screenshot shows the IntelliJ IDEA interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Toolbar:** Standard toolbar items like New, Open, Save, etc.
- Status Bar:** 11 Dec 18:41, Hadi Hariri, 7:14, UTF-8, Git: master.
- Project View:** Shows a project named "vjug [demos]" with a "src" folder containing various Kotlin files like Collections.kt, Conventions.kt, CustomerKotlin.kt, etc.
- Code Editor:** Displays the content of Delegation.kt, which includes traits for ReadOnlyProperty and ReadWriteProperty, and an object Delegates.
- Run Tab:** Shows a successful run of "CustomerKotlin" with parameters name=Joe, email=joe@gmail.com, country=Spain.
- Bottom Status:** Process finished with exit code 0.
- Message Bar:** "Navigate → Declaration via ⌘B (Ctrl+B for Win/Linux)"
- Bottom Footer:** 4 files committed: Code Ready (7 minutes ago)

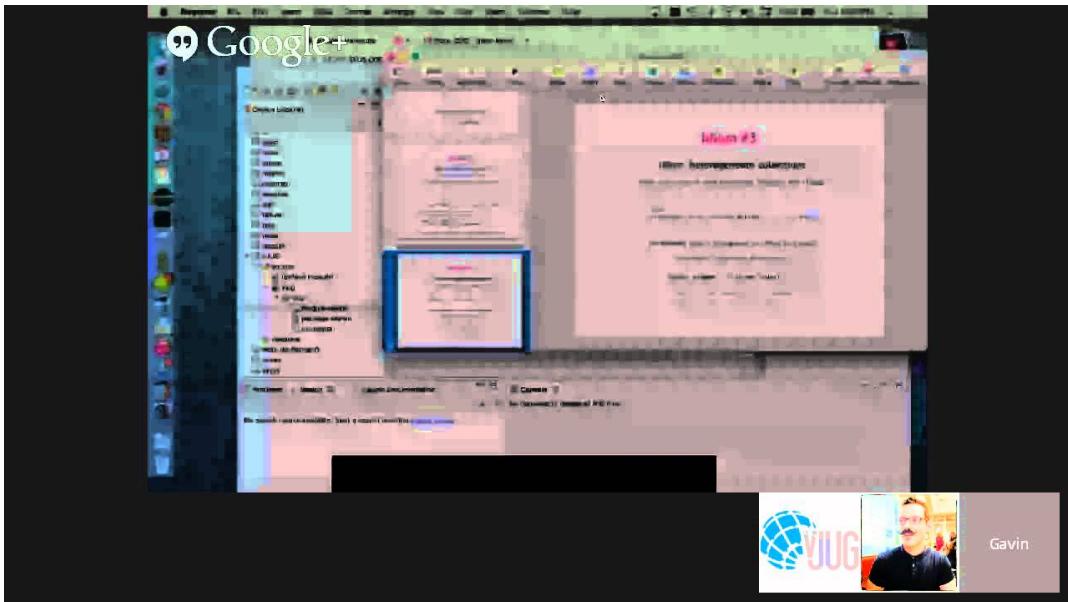
- What are the major advantages/features Kotlin provides
- Why should someone move from Java to Kotlin
- What is the future direction of Kotlin



<http://virtualjug.com/?p=1297>



# Ceylon for Java Developers



- What are the major advantages/features Ceylon provides
- Why should someone move from Java to Ceylon
- What is the future direction of Ceylon



<http://virtualjug.com/?p=1294>



# Groovy for Java Developers

The slide contains two code snippets side-by-side:

```
IntStream.range(1, 100).forEach(s -> System.out.println(s));  
Files.lines(Paths.get('README.adoc'))  
    .map(it -> it.toUpperCase())  
    .forEach(it -> System.out.println(it))
```

```
IntStream.range(1, 100).forEach { println it }  
Files.lines(Paths.get('README.adoc'))  
    .map { it.toUpperCase() }  
    .forEach { println it }
```

A red callout bubble with white text points from the Java 8 code towards the Groovy code, containing the text: "Use Groovy closures wherever you pass lambdas in Java 8".

At the bottom left is the handle @glaforge. At the bottom right is the number 44, a small image of a person with a mustache, and the text "Guillaume".

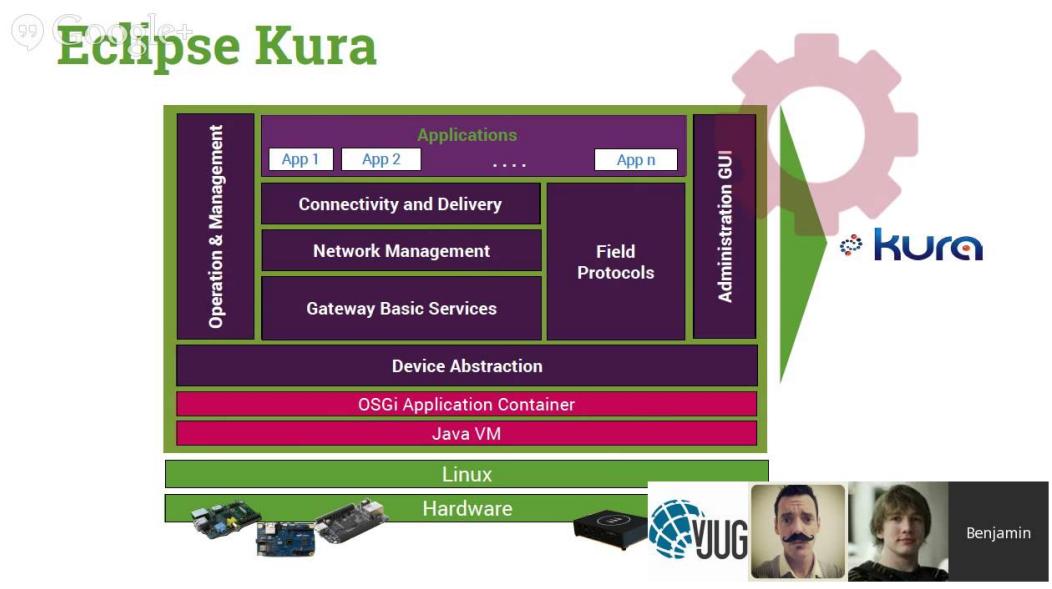
Speaker: Guillaume Laforge



<http://virtualjug.com/?p=1288>



# Building the Internet of Things with Java



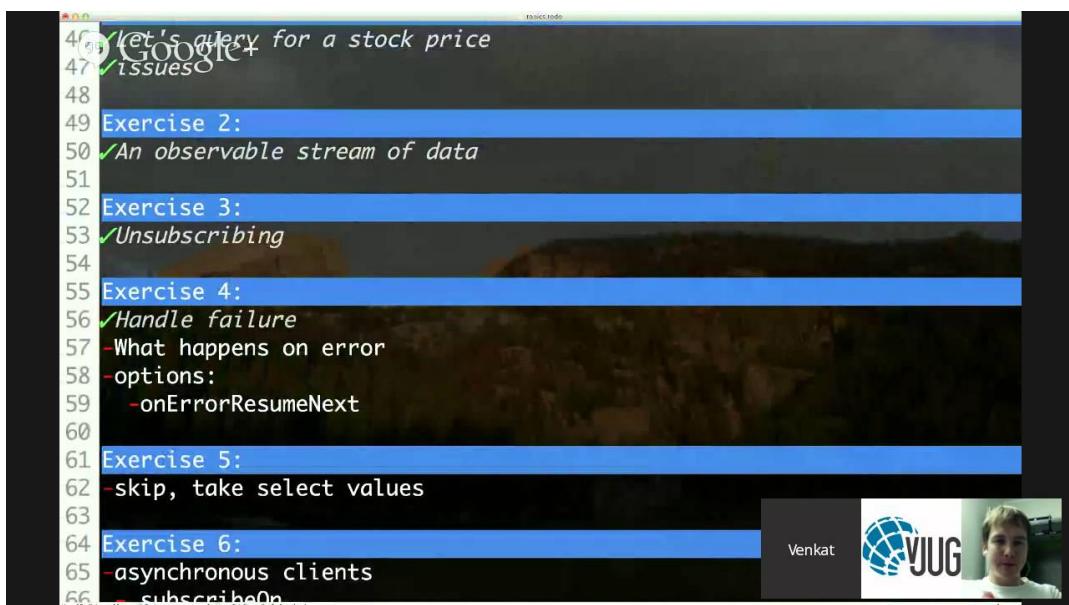
It may seem hard to get started with the Internet of Things (IoT) with so many technologies, protocols, hardware platforms, involved. In this session, Benjamin Cabé from the Eclipse Foundation will cover all you need to know



<http://virtualjug.com/?p=1195>



# Reactive Programming: Creating highly responsive applications



```
46 Let's query for a stock price
47 ✓Issues
48
49 Exercise 2:
50 ✓An observable stream of data
51
52 Exercise 3:
53 ✓Unsubscribing
54
55 Exercise 4:
56 ✓Handle failure
57 -What happens on error
58 -options:
59   -onErrorResumeNext
60
61 Exercise 5:
62 -skip, take select values
63
64 Exercise 6:
65 -asynchronous clients
66   -subscribeOn
```

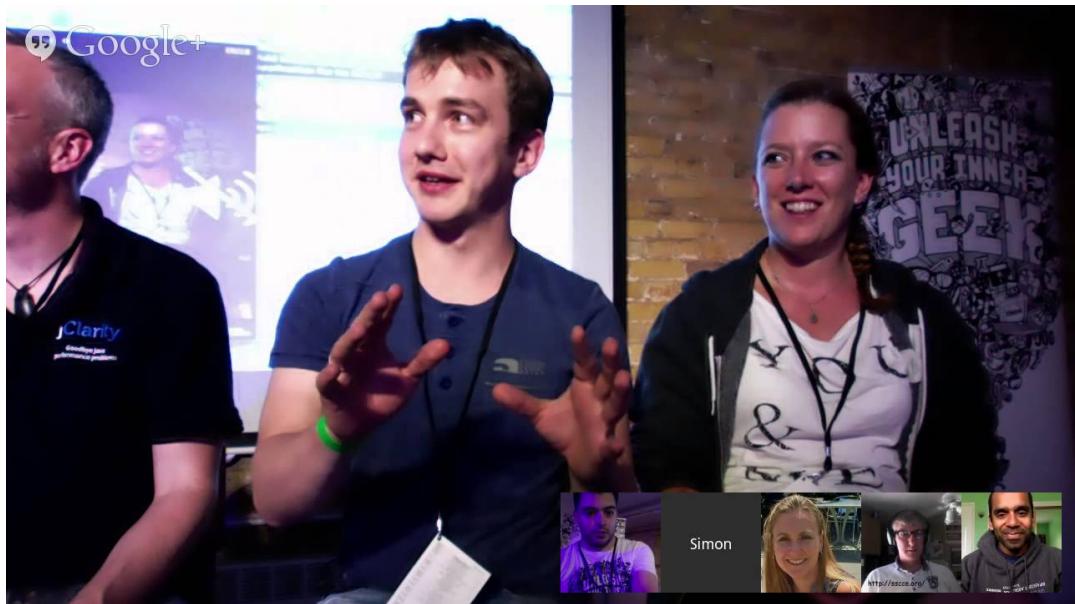
Reactive Programming is gaining a lot of attention recently, but what is it? It is a culmination of a lot of good ideas developed over the years, but brought together by the forces of recent developments



<http://virtualjug.com/?p=1193>



# Shaping Java's future & vJUG party!



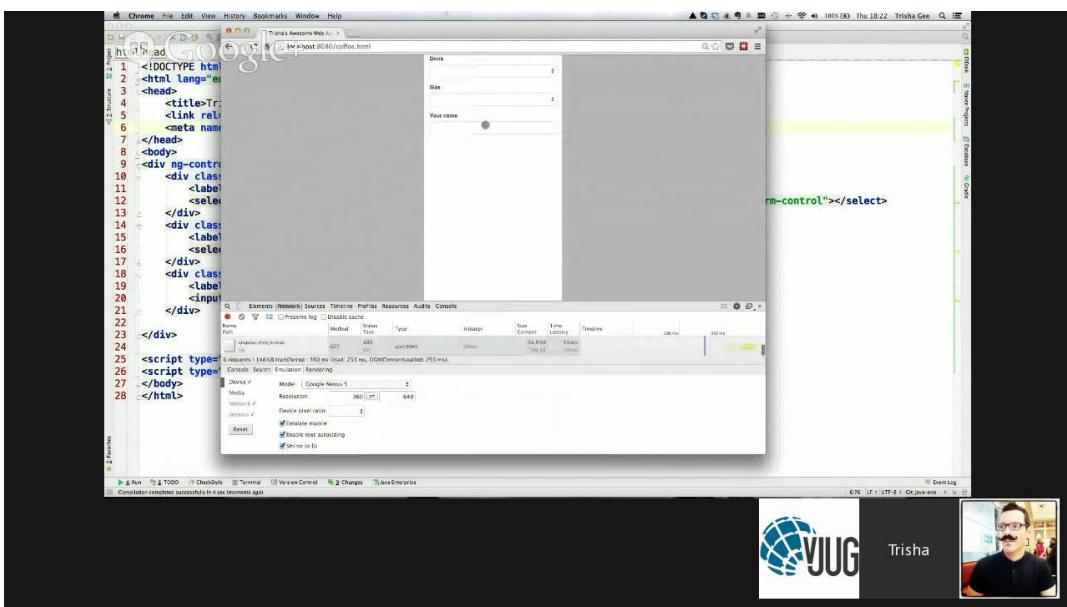
Shaping Java's future & vJUG party!



<http://virtualjug.com/?p=1191>



# HTML5, AngularJS, Groovy, Java and MongoDB all together – what could go wrong?



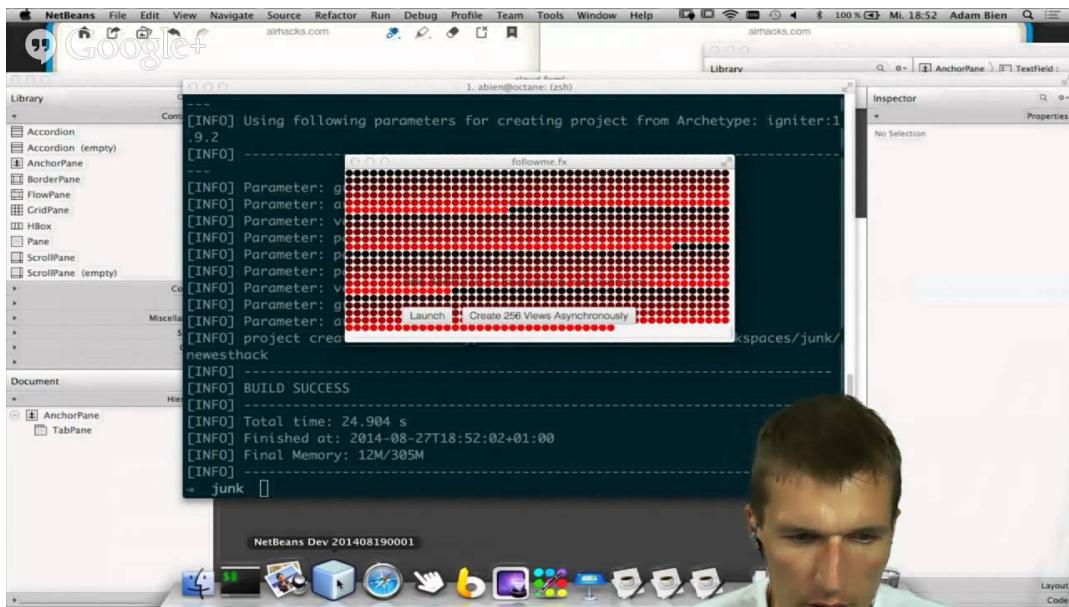
Speaker: Trisha Gee



<http://virtualjug.com/?p=1188>



# Opinionated JavaFX 8



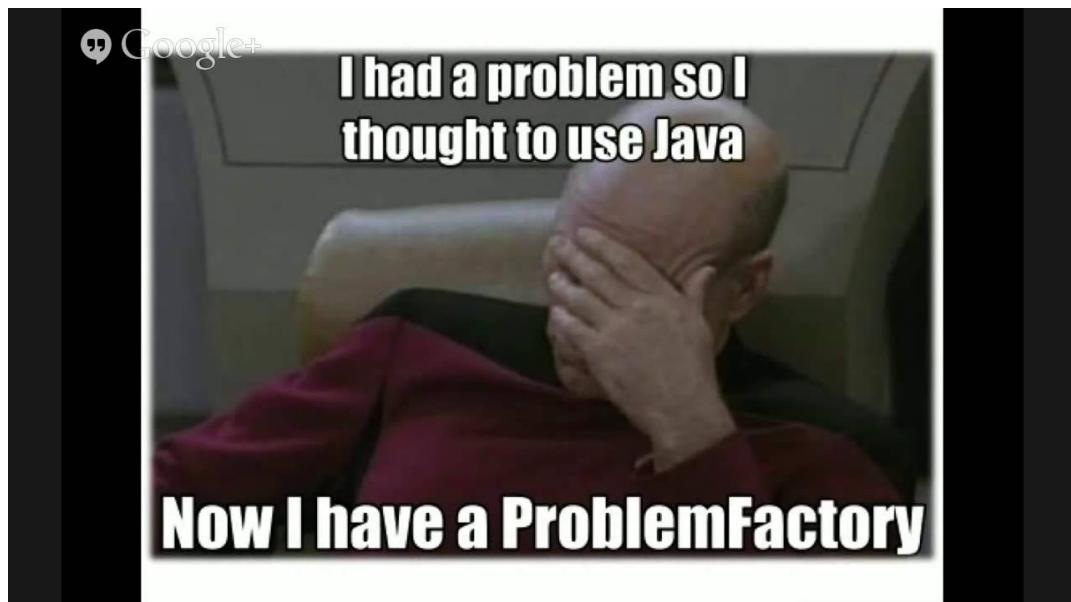
**Speaker: Adam Bien**



<http://virtualjug.com/?p=1157>



# 3 years of backend testing at Shazam [the stuff we got wrong]



Speaker: Colin Vipurs



<http://virtualjug.com/?p=1155>



# Pragmatic Functional Refactoring with Java 8



Scenario: be able to link together train journeys  
to form longer journeys.



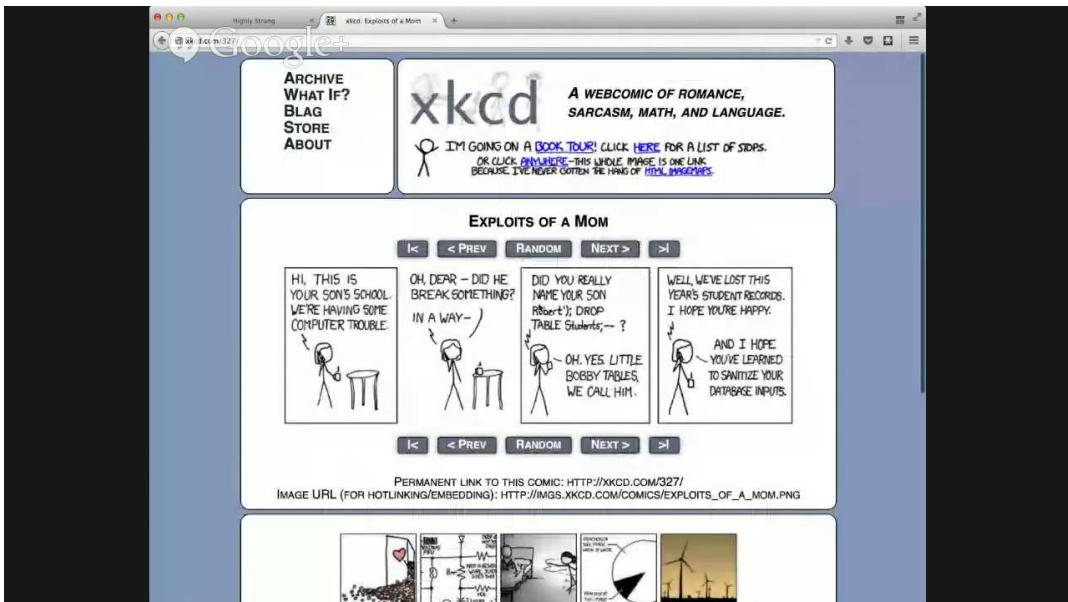
**Speakers:** Richard Warburton & Raoul-Gabriel Urma



<http://virtualjug.com/?p=1152>



# Highly Strung: Understanding your Type System



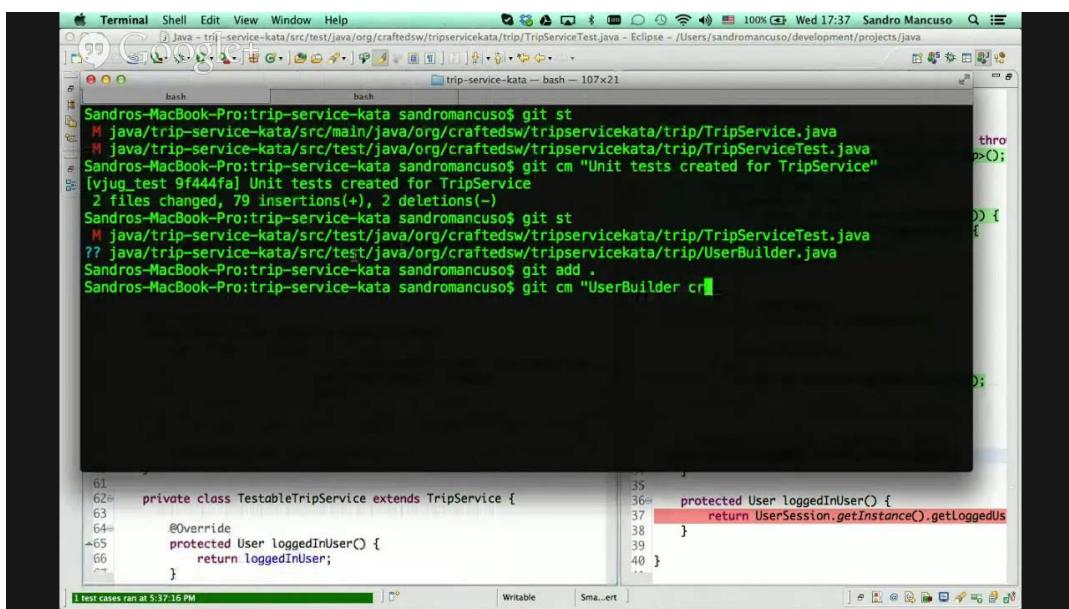
## Highly Strung: Understanding your Type System



<http://virtualjug.com/?p=1121>



# Testing and Refactoring Legacy Code



The screenshot shows a Mac OS X desktop environment with an Eclipse IDE window open. The terminal tab in the top bar shows the command-line history:

```
Sandros-MacBook-Pro:trip-service-kata sandromancuso$ git st
M java/trip-service-kata/src/main/java/org/craftedsw/tripservicekata/trip/TripService.java
M java/trip-service-kata/src/test/java/org/craftedsw/tripservicekata/trip/TripServiceTest.java
S Sandros-MacBook-Pro:trip-service-kata sandromancuso$ git cm "Unit tests created for TripService"
[vjug_test 9f444fa] Unit tests created for TripService
 2 files changed, 79 insertions(+), 2 deletions(-)
Sandros-MacBook-Pro:trip-service-kata sandromancuso$ git st
M java/trip-service-kata/src/test/java/org/craftedsw/tripservicekata/trip/TripServiceTest.java
?? java/trip-service-kata/src/test/java/org/craftedsw/tripservicekata/trip/UserBuilder.java
S Sandros-MacBook-Pro:trip-service-kata sandromancuso$ git add .
Sandros-MacBook-Pro:trip-service-kata sandromancuso$ git cm "UserBuilder cr"
```

The code editor shows a Java class named `TestableTripService`:

```
61  private class TestableTripService extends TripService {
62+     @Override
63+     protected User loggedInUser() {
64+         return loggedInUser;
65+     }
66 }
```

```
35
36+     protected User loggedInUser() {
37         return UserSession.getInstance().getLoggedUs
38     }
39 }
40 }
```

The code editor interface includes tabs for `test cases ran at 5:37:16 PM`, `Writable`, and `SmartEdit`.

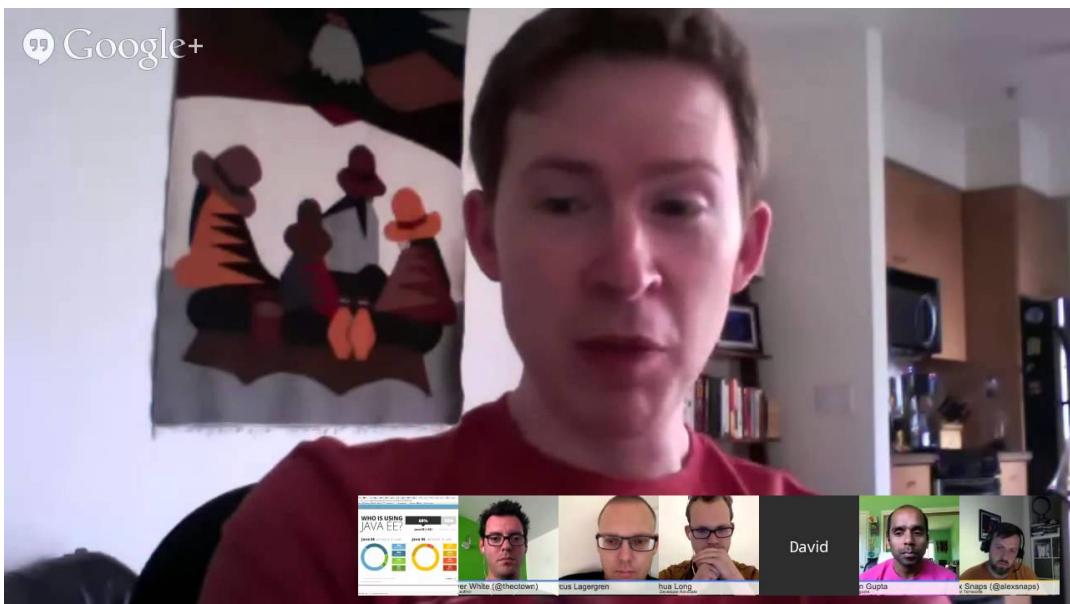
Speaker: Sandro Mancuso



<http://virtualjug.com/?p=993>



# vJUG panel: Review of 2164 Survey Responses on Java Tools and Technology



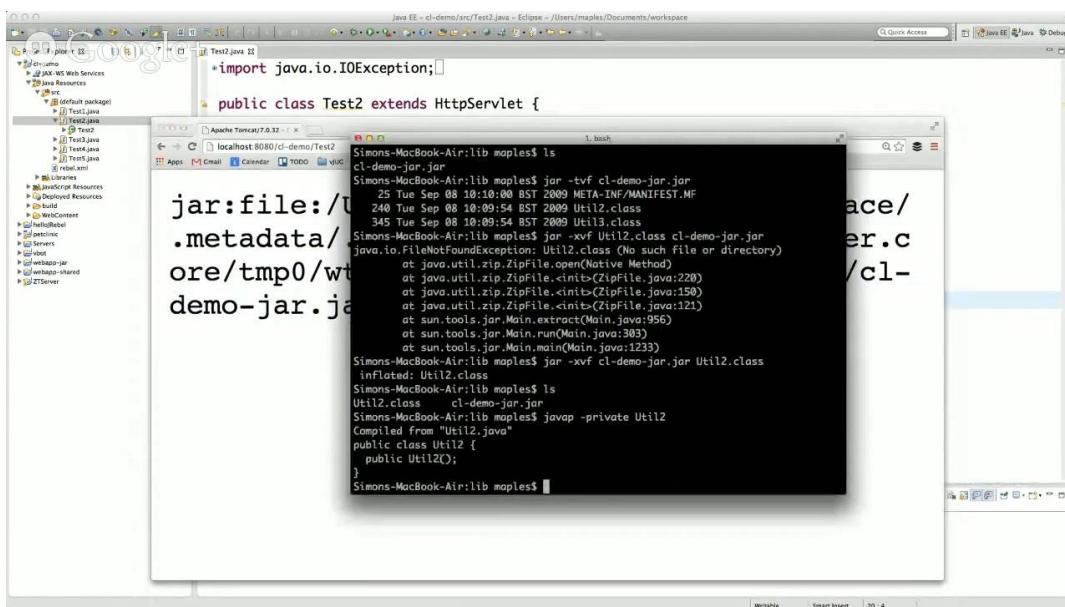
Speakers: Arun Gupta, Josh Long, Marcus Lagergren, Alex Snaps, David Blevins & Oliver White (Moderator). We look at the recent explosive publication of RebelLabs' "Java Tools and Technologies Landscape for 2014", a beautifully-designed, 56-page snapshot of what over 2000 Java developers from around the world are using in their daily development.



<http://virtualjug.com/?p=938>



# Java Classloaders: The good, the bad and the WTF.



The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays a Java project structure with packages like 'JAX-WS Web Services' and 'Java Resources'. In the center, the code editor shows a file named 'Test2.java' with the following content:

```
*import java.io.IOException;*
public class Test2 extends HttpServlet {
```

To the right of the code editor is a terminal window titled 'Terminal' showing command-line output. The output is a log of class loading events from a Mac OS X terminal window:

```
Simons-MacBook-Air:lib maples$ ls
cl-demo-jar.jar
Simons-MacBook-Air:lib maples$ jar -tvf cl-demo-jar.jar
25 Tue Sep 08 10:10:00 BST 2009 META-INF/MANIFEST.MF
240 Tue Sep 08 10:09:54 BST 2009 Util2.class
345 Tue Sep 08 10:09:54 BST 2009 Util3.class
Simons-MacBook-Air:lib maples$ jar -xvf Util2.class cl-demo-jar.jar
java.io.FileNotFoundException: Util2.class (No such file or directory)
at java.util.zip.ZipFile.open(Native Method)
at java.util.zip.ZipFile.<init>(ZipFile.java:220)
at java.util.zip.ZipFile.<init>(ZipFile.java:150)
at sun.tools.jar.Main.extract(Main.java:956)
at sun.tools.jar.Main.run(Main.java:303)
at sun.tools.jar.Main.main(Main.java:1233)
Simons-MacBook-Air:lib maples$ jar -xvf cl-demo-jar.jar Util2.class
inflated: Util2.class
Simons-MacBook-Air:lib maples$ ls
Util2.class  cl-demo-jar.jar
Simons-MacBook-Air:lib maples$ javap -private Util2
Compiled from "Util2.java"
public class Util2 {
    public Util2();
}
```

The terminal window title bar says 'Terminal' and the status bar at the bottom indicates 'Writable' and 'Smart Insert'.

Speaker: Simon Maple.



<http://virtualjug.com/?p=936>



# vJUG Panel: What do the Oracle/Google shenanigans mean to the Java Developer?



Speaker(s): Bruno Souza, Martijn Verburg and Hildeberto Mendonça, Lukas Eder & Michael Rice. (Moderated by Simon Maple)



<http://virtualjug.com/?p=865>



# Netty – The async event-driven network application framework



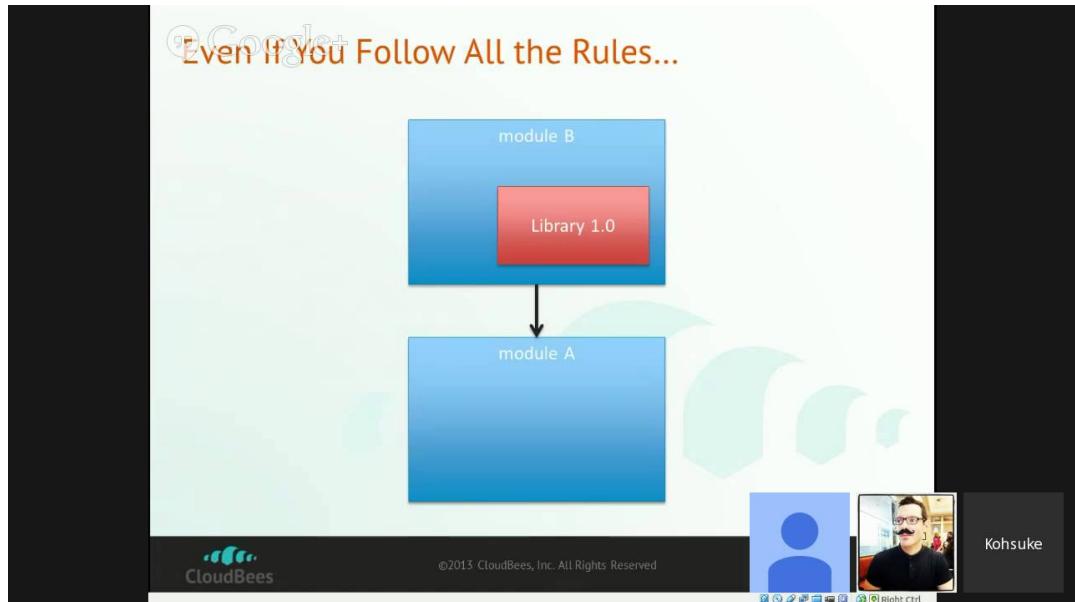
Speaker: Norman Maurer.



<http://virtualjug.com/?p=862>



# Evolving code without breaking compatibility



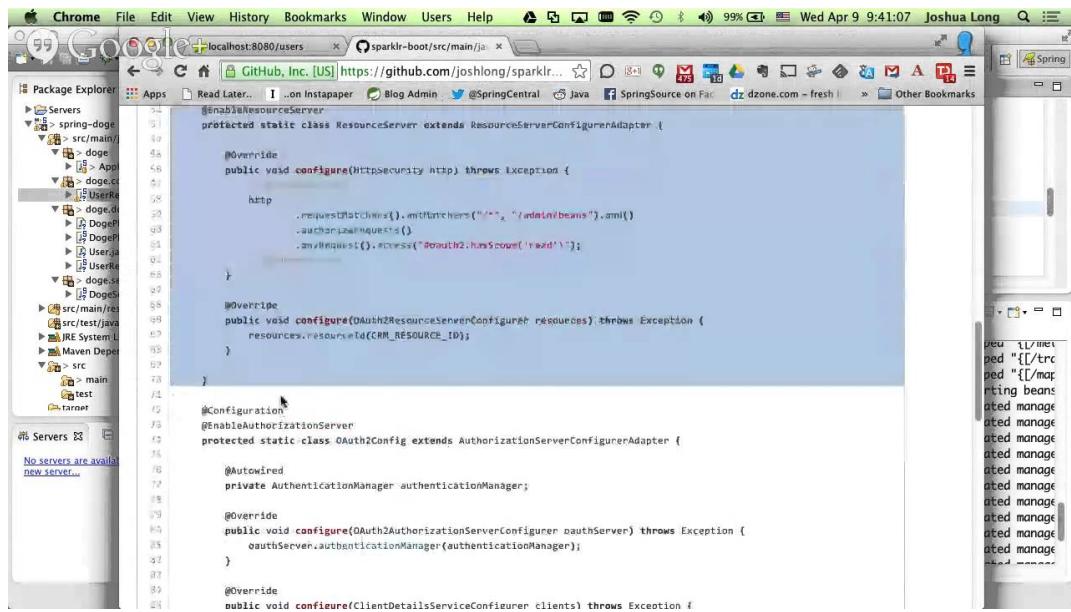
Speaker: Kohsuke Kawaguchi



<http://virtualjug.com/?p=159>



# Building Bootiful Applications with Spring Boot



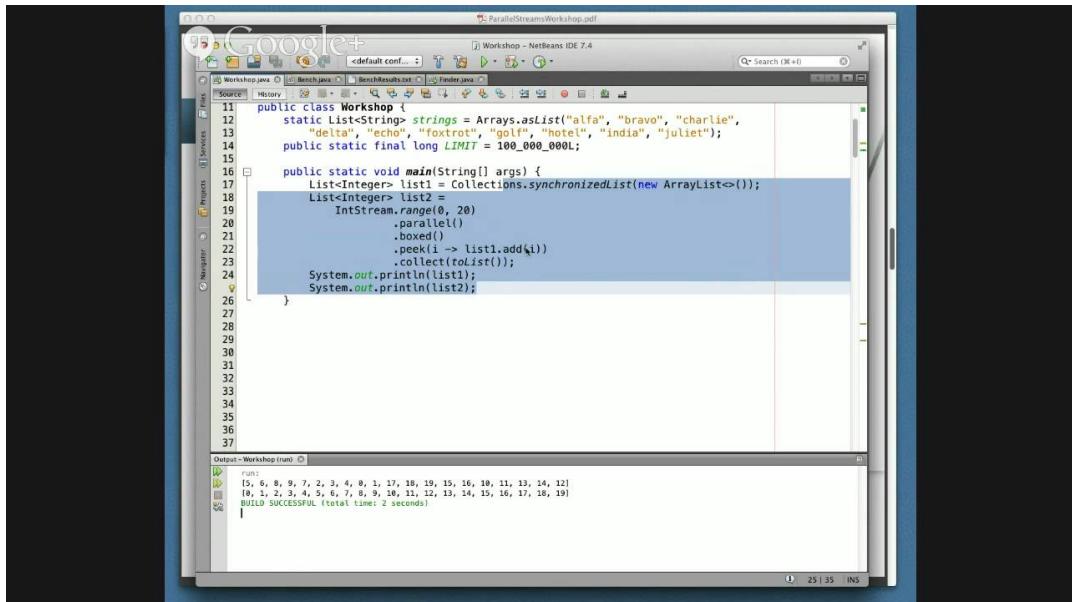
Speaker: Josh Long



<http://virtualjug.com/?p=157>



# Java 8 Parallel Streams Workshop



The screenshot shows the NetBeans IDE interface. The code editor displays a Java file named 'Workshop.java' containing the following code:

```
11  public class Workshop {
12      static List<String> strings = Arrays.asList("alfa", "bravo", "charlie",
13          "delta", "echo", "foxtrot", "golf", "hotel", "india", "juliet");
14      public static final long LIMIT = 100_000_000L;
15
16      public static void main(String[] args) {
17          List<Integer> list1 = Collections.synchronizedList(new ArrayList<>());
18          List<Integer> list2 =
19              IntStream.range(0, 20)
20                  .parallel()
21                  .boxed()
22                  .peek(i -> list1.add(i))
23                  .collect(toList());
24          System.out.println(list1);
25          System.out.println(list2);
26      }
27
28
29
30
31
32
33
34
35
36
37}
```

The output window below shows the results of the run:

```
Run:
[5, 6, 8, 9, 7, 2, 3, 4, 9, 1, 17, 18, 19, 15, 16, 18, 11, 13, 14, 12]
[9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
BUILD SUCCESSFUL (total time: 2 seconds)
```

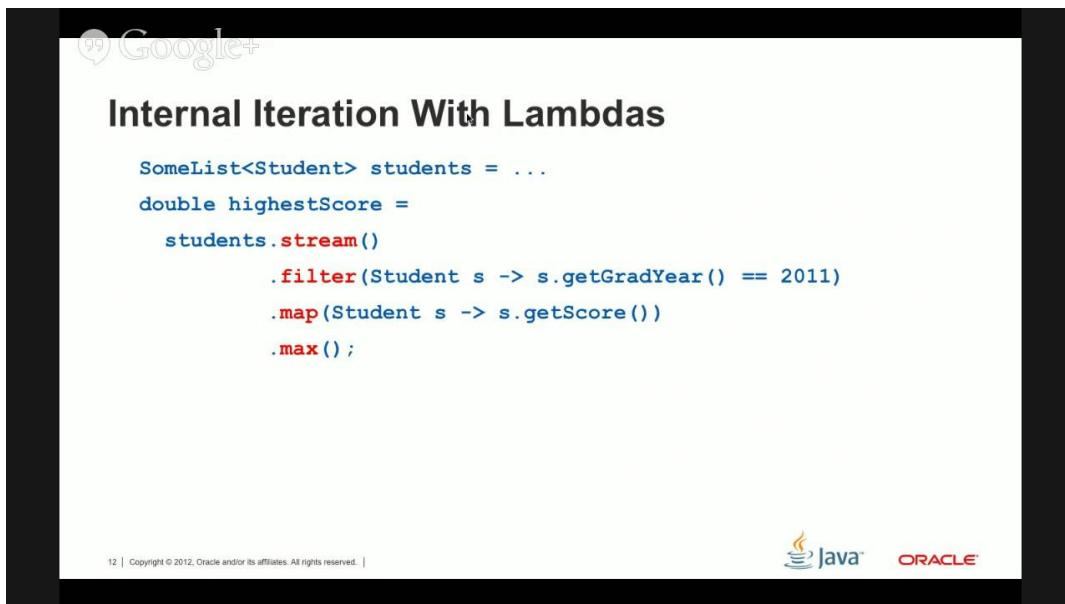
Speaker: Stuart Marks.



<http://virtualjug.com/?p=155>



# Project Lambda: Functional Prog. Constructs and Simpler Concurrency in Java SE 8



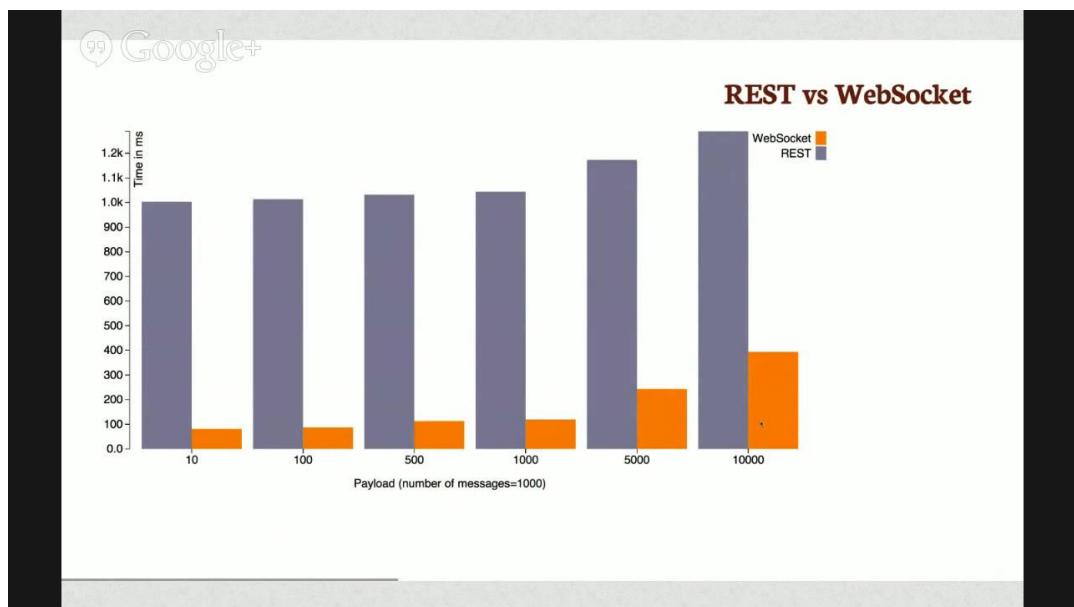
Speaker: Simon Ritter.



<http://virtualjug.com/?p=153>



# WebSocket Applications using Java EE 7



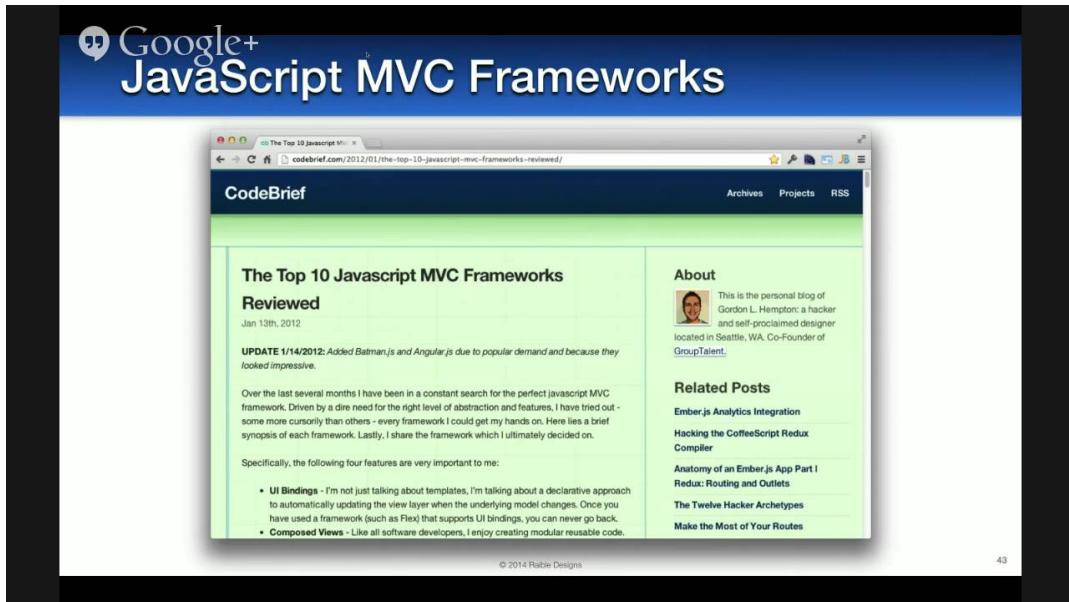
Speaker: Arun Gupta.



<http://virtualjug.com/?p=151>



# Comparing JVM Web Frameworks



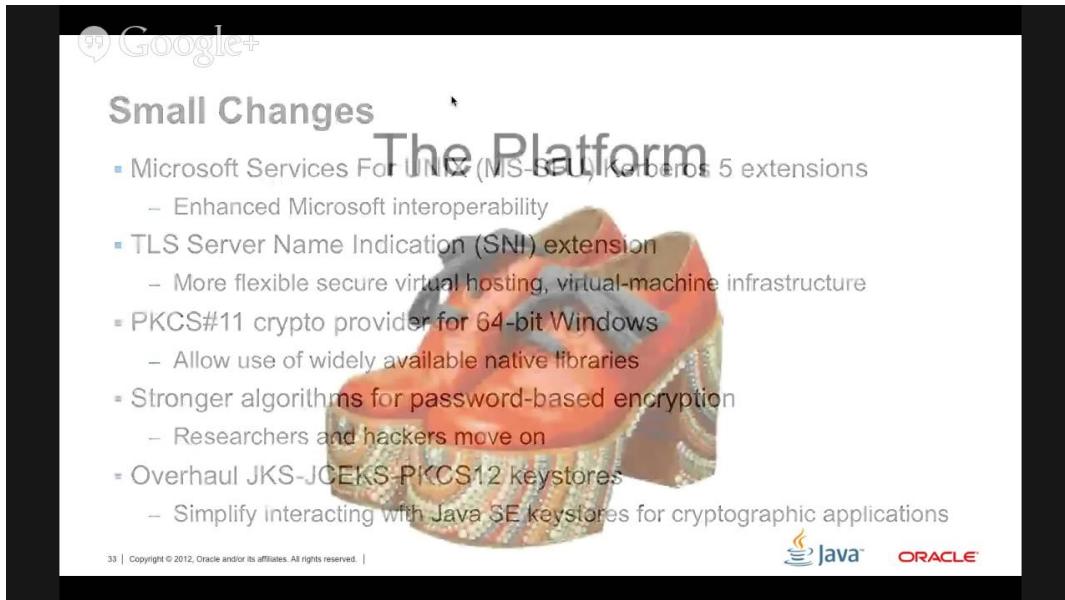
Speaker: Matt Raible



<http://virtualjug.com/?p=149>



# 55 New Features in Java SE 8



A Google+ slide titled "Small Changes" under "The Platform". The slide lists several new features:

- Microsoft Services For UNIX (MS-SUL) Kerberos 5 extensions
  - Enhanced Microsoft interoperability
- TLS Server Name Indication (SNI) extension
  - More flexible secure virtual hosting, virtual-machine infrastructure
- PKCS#11 crypto provider for 64-bit Windows
  - Allow use of widely available native libraries
- Stronger algorithms for password-based encryption
  - Researchers and hackers move on
- Overhaul JKS-JCEKS-PKCS12 keystores
  - Simplify interacting with Java SE Keystores for cryptographic applications

33 | Copyright © 2012, Oracle and/or its affiliates. All rights reserved. |  

Speaker: Simon Ritter



<http://virtualjug.com/?p=147>



# How To Do Kick-Ass Software Development



Speaker: Sven Peters



<http://virtualjug.com/?p=144>



# Getting started with Java EE 7

The slide features a central white rectangular area surrounded by black vertical bars on the left and right. In the top-left corner of the white area is the Google+ logo. In the top-right corner is the HTML5 logo, which consists of a stylized orange '5' with the word 'HTML' above it and a small figure below. In the bottom-right corner of the white area is the Red Hat logo. The central white area contains the following text:  
Java API for RESTful Web Services 2.0  
Client API  
Message Filters and Entity Interceptors  
Asynchronous Processing – Server and Client  
Common Configuration

19

redhat.

Presenter: Arun Gupta.



<http://virtualjug.com/?p=142>



# Don't be that guy! Developer Security Awareness



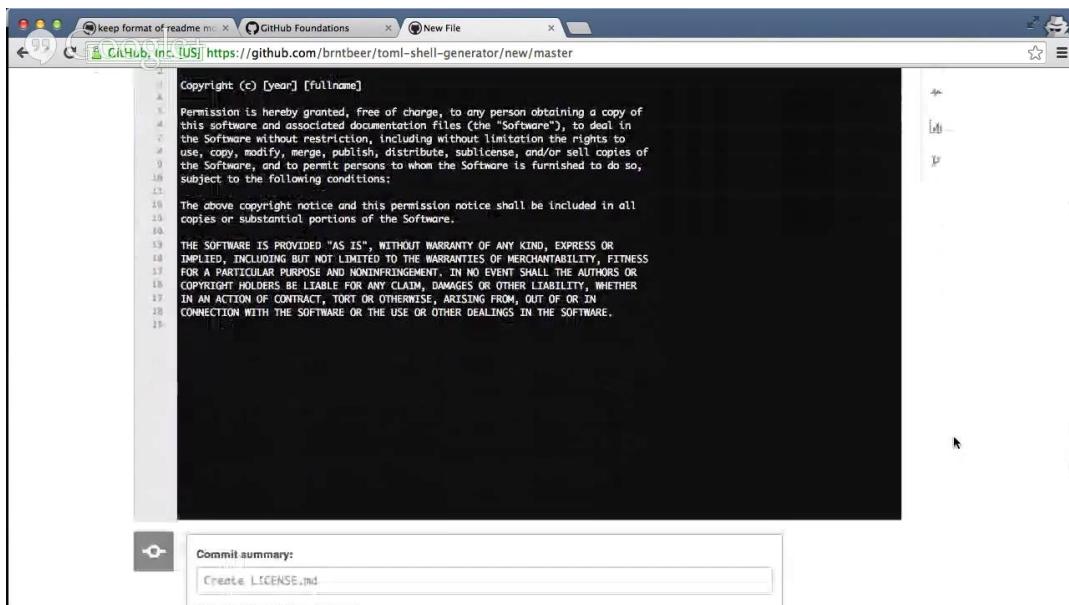
Presenter(s): Markus Eisele.



<http://virtualjug.com/?p=139>



# Drive-by Contributions



A screenshot of a GitHub commit interface. The main area shows a file named 'LICENSE.md' containing a standard MIT-style license notice. Below the code editor, there is a 'Commit summary:' field with the placeholder 'Create LICENSE.md'.

```
Copyright (c) [year] [fullname]

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

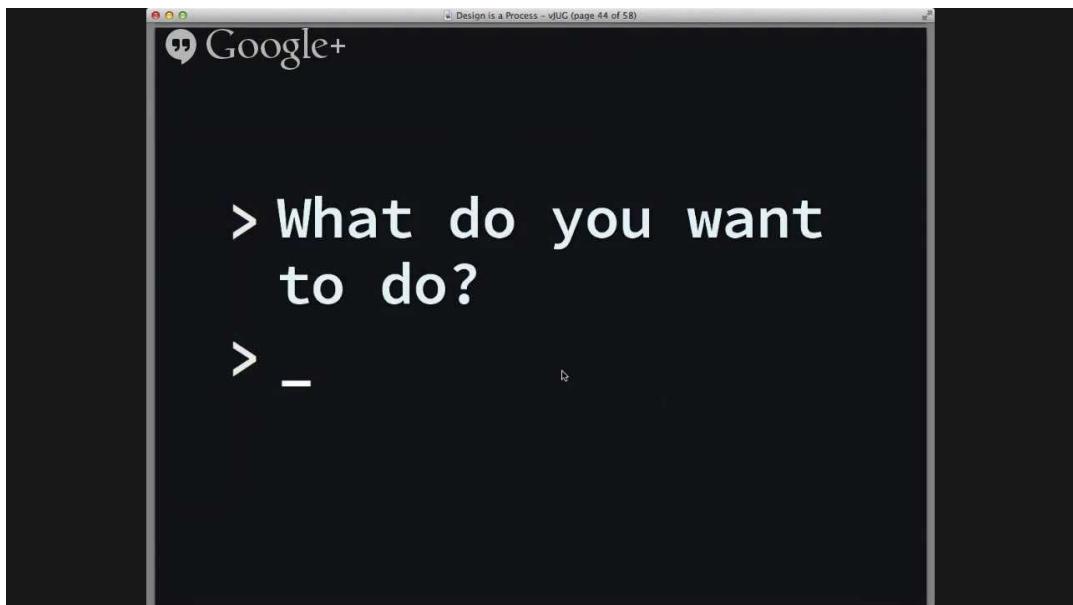
Presenter(s): Brent Beer & Matthew McCullough.



<http://virtualjug.com/?p=137>



# Design is a Process, not a Document



Presenter(s): Trisha Gee.



<http://virtualjug.com/?p=132>



