

Java byte code in practice

The slide contains four sections of bytecode instructions:

- INVOKESTATIC** `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).
- INVOKEDYNAMIC** `foo ()V bootstrap`
Queries the given *bootstrap method* for locating a method implementation at runtime.
(MethodHandle: Combines a specific method and an **INVOKEX** instruction.)

At the bottom right are the VJUG logo and a photo of 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.

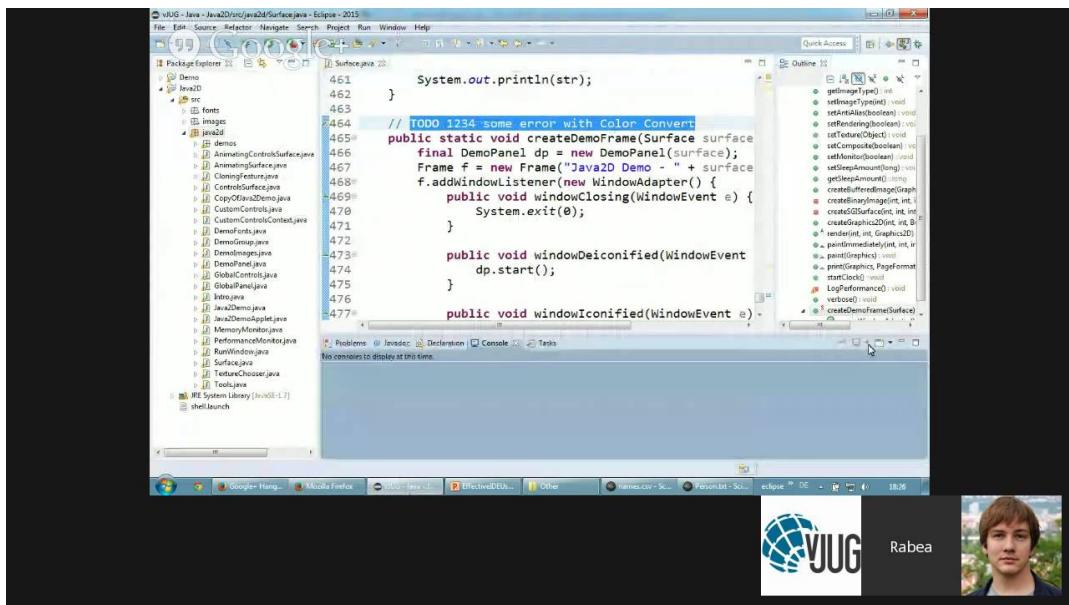
(<http://bytebuddy.net>)



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



Effective IDE Usage



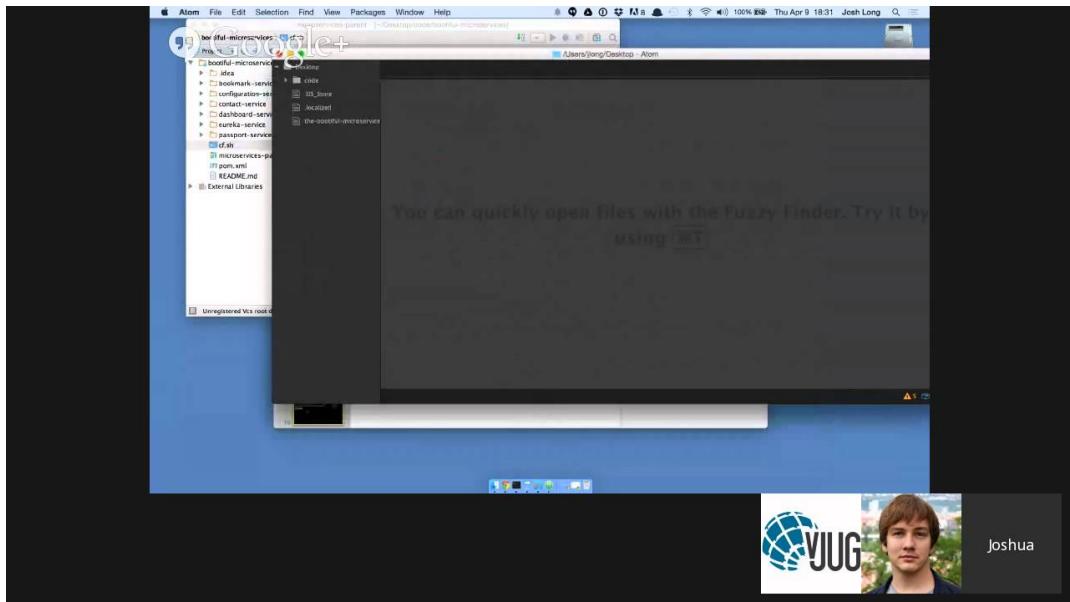
You don't want your IDE to propose `java.awt.List` as import when you need `java.util.List`? This talk will show you how to get rid of the proposal and how to use your IDE effectively to concentrate on your work.



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



Building “Bootiful” Microservices with Spring Cloud



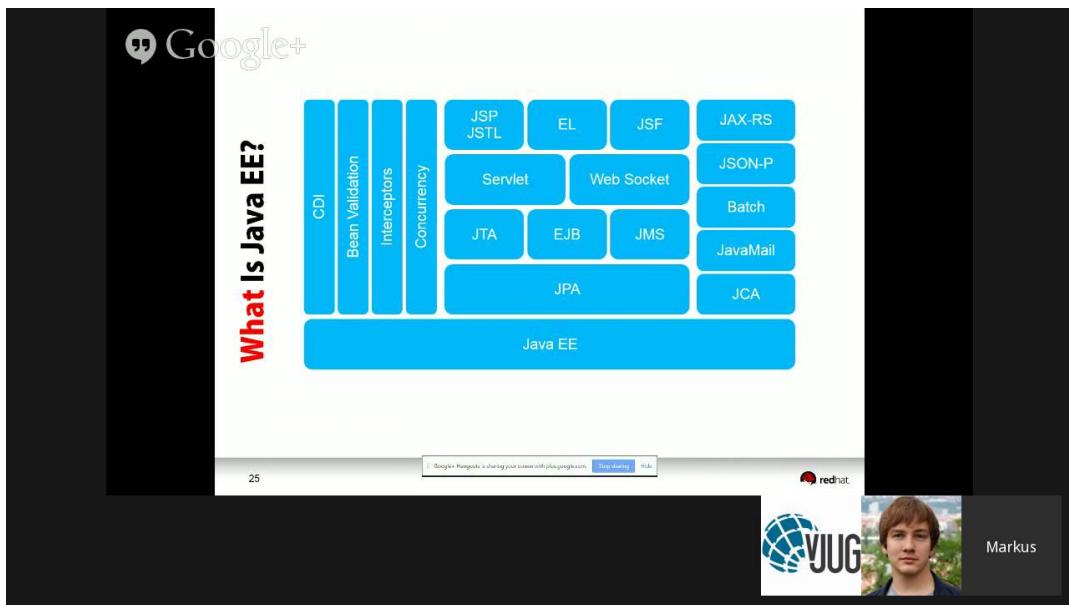
We get it already! Microservices help you build smaller, singly-focused services, quicker. They scale out. They’re more agile because individual teams can deliver them at their own pace. They work well in the cloud because they’re smaller, and benefit from elastic, horizontal scaling. But what about 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



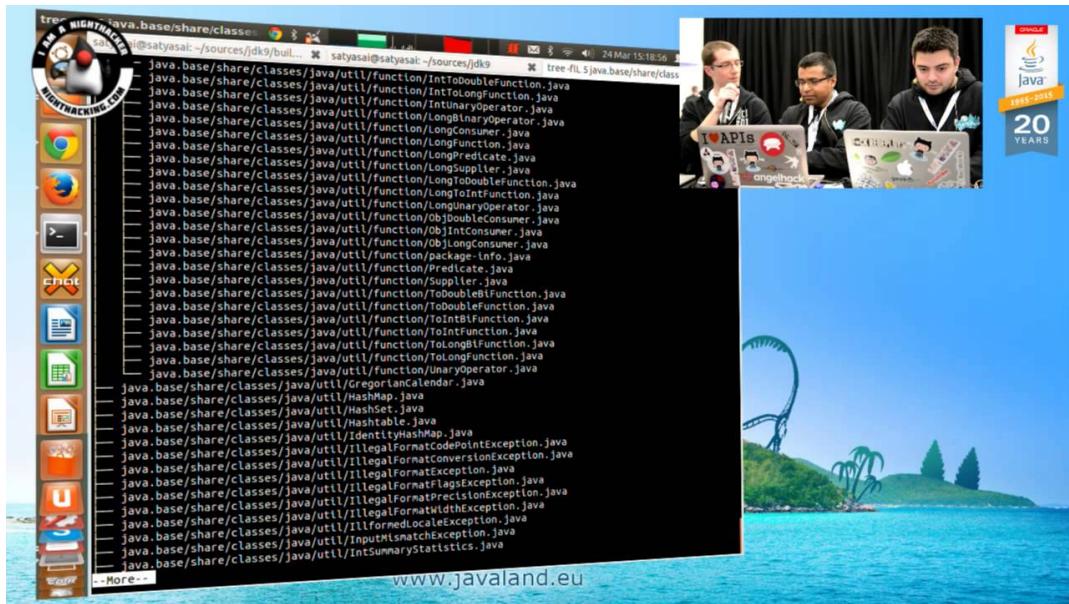
In the past I've been building component oriented applications with what I had at hand. Mostly driven by the features available in the Java EE standard to be "portable" and easy to use. Looking back this has been a perfect fit for many customers and applications. With an increasing demand for highly 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?



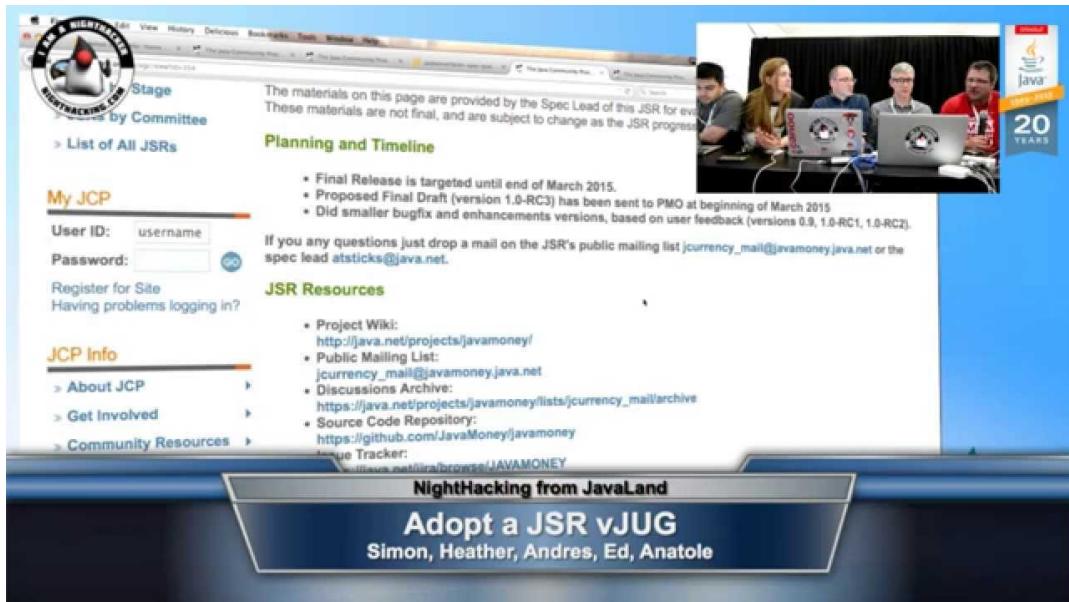
Mr Webber developer and **Ms Janet Java developer** are both developers who are interested in broadening their know-how of the Java platform. **Mr Webber developer** shares with **Ms Janet Java developer** conservations about Javaland, vJUG, Nighthacking and Adopt OpenJDK – a preview of their conversation.



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



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



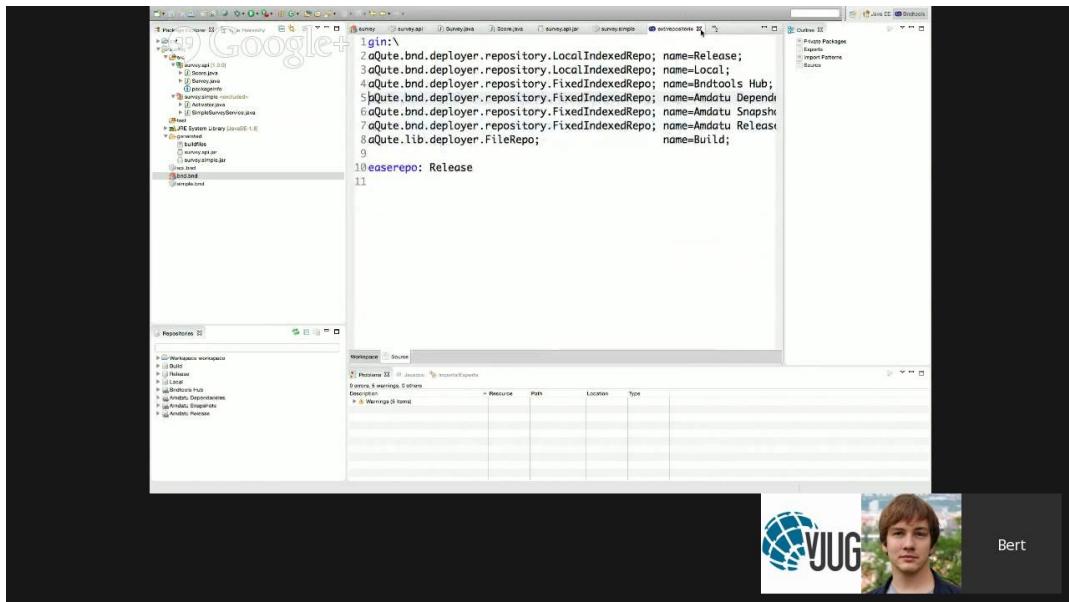
This session will take place live from the Javaland conference in Germany on the Nighthacking stage! Learn from **Heather VanCura** how you can take part in Java technology by Adopting a JSR. This session give a brief overview of the Adopt-a-JSR program and how to participate through the Virtual JUG. 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

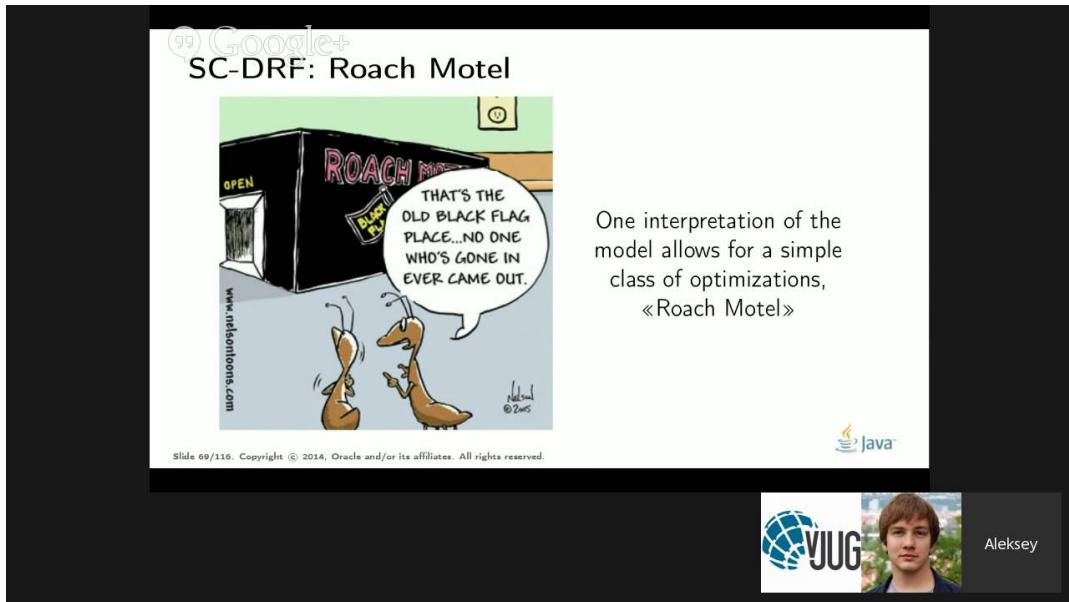
Modularity is an architectural theme that you'll hear about more and more. Being able to deal with change in a codebase is not something trivial and requires some serious thought. In this talk I will show you that it is actually pretty easy to achieve a modular architecture using OSGi, and the right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



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



Java Memory Model Pragmatics



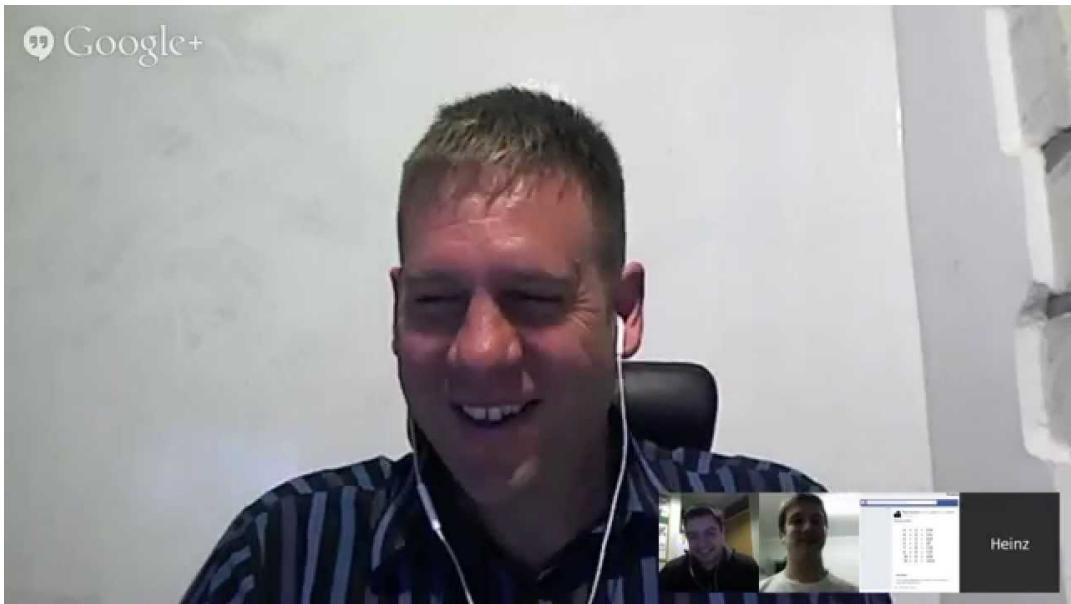
The Java Memory Model is the most complicated part of Java spec that must be understood by at least library and runtime developers. Unfortunately, it is worded in such a way that it takes a few senior guys to decipher it for each other.



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



The Live Reflection Madness



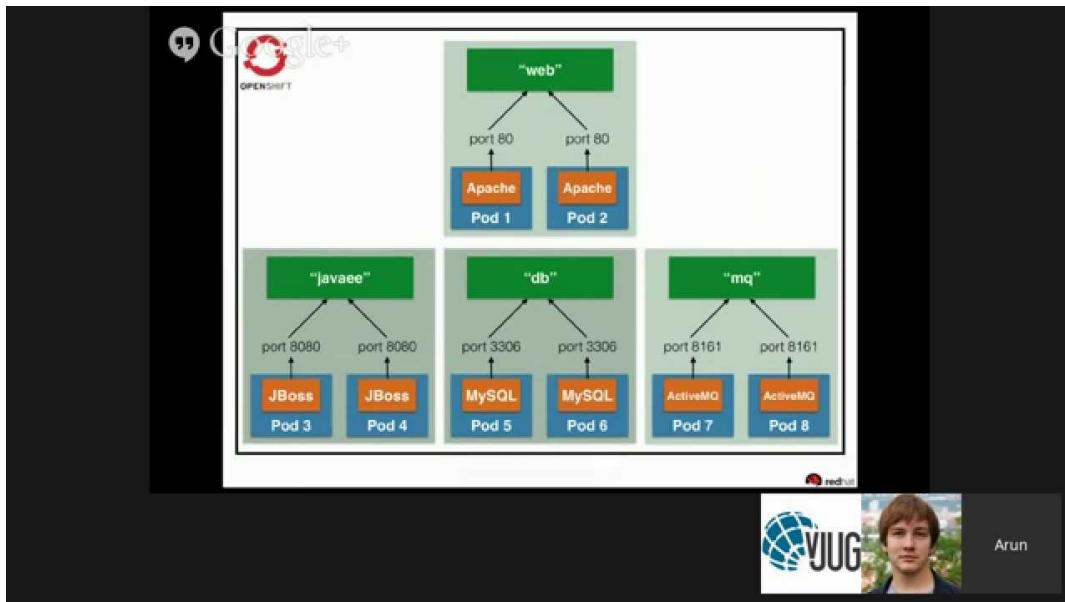
Heinz likes to compare reflection to opium. Not the perfume. The drug. In this live coding session, he will start by showing some of the powerful features available to us in Java.



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



Package your Java EE application using Docker and Kubernetes



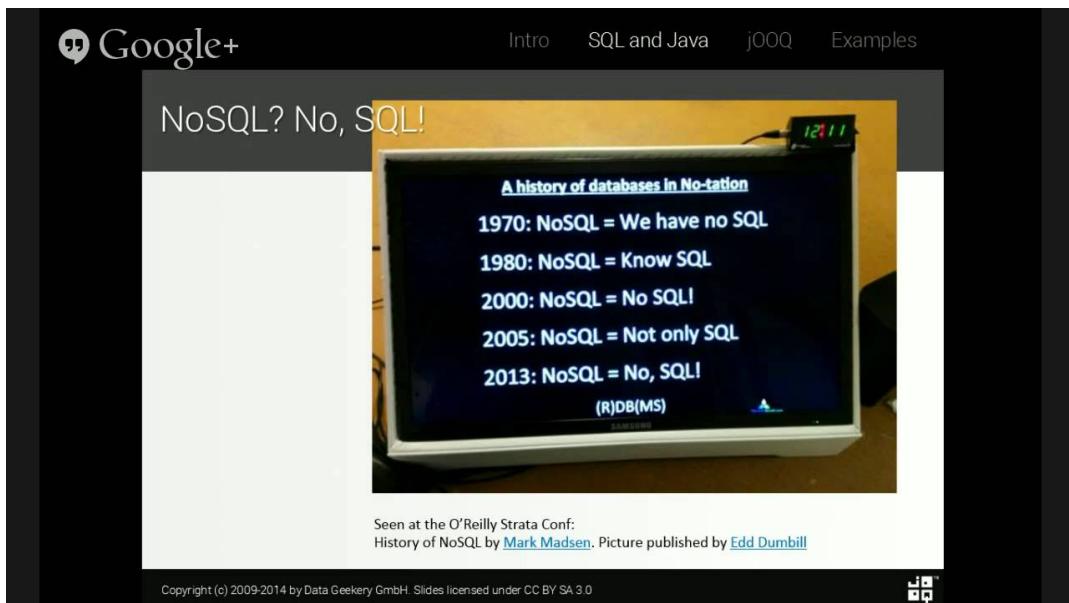
Docker simplifies software delivery by making it easy to build and share images that contain your application's operating system. It packages your application and infrastructure together, managed as one component.



<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



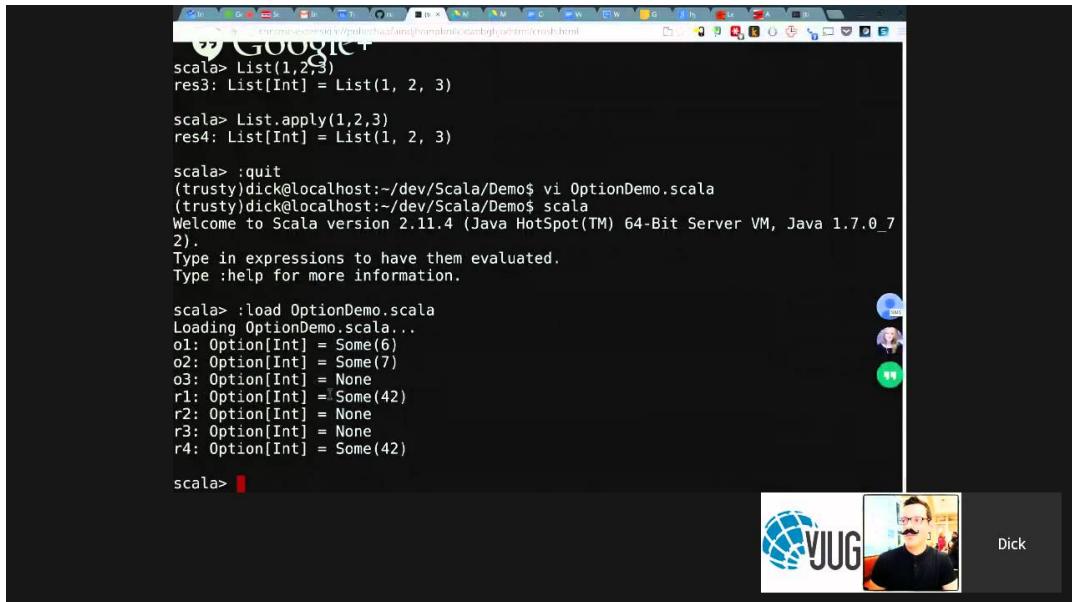
[IRC logs be be found here.](#)



<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>
```

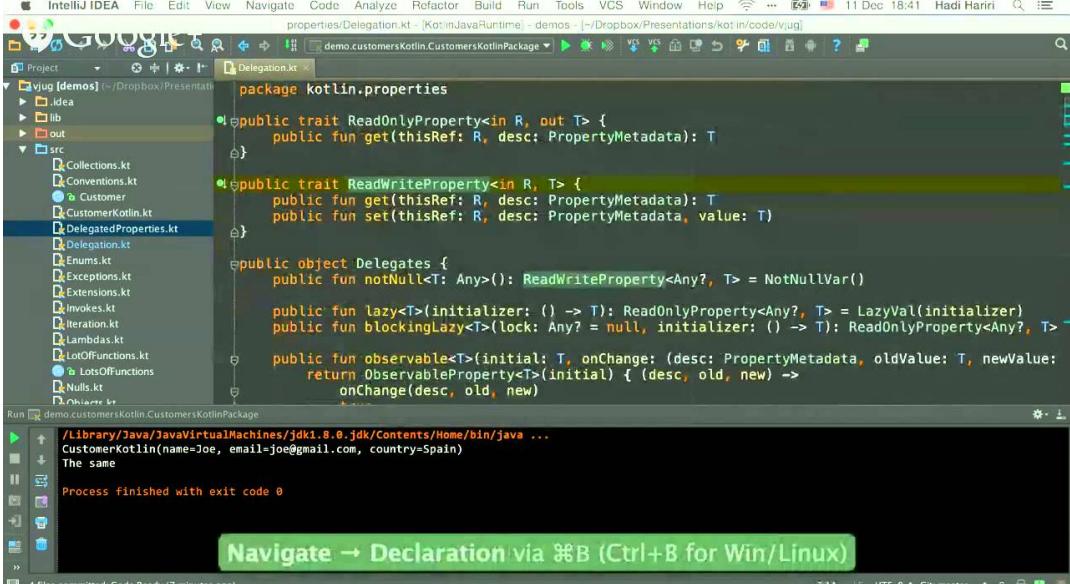
- What are the major advantages/features Scala provides
- Why should someone move from Java to Scala
- What is the future direction of Scala



<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:** The current file is "Delegation.kt" which contains Kotlin code for traits and objects related to delegation.
- 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)

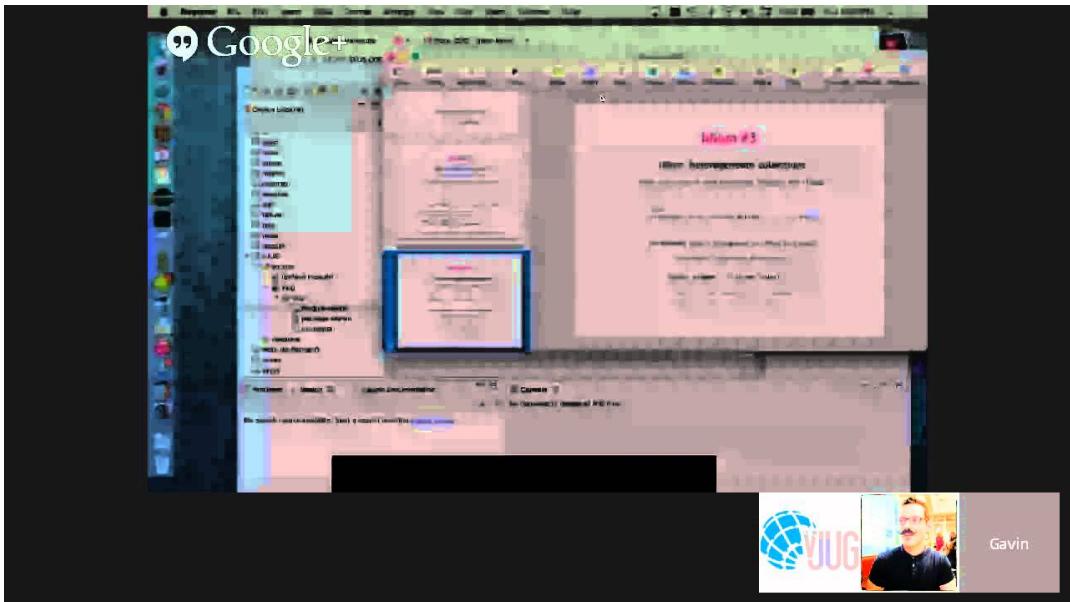
- What are the major advantages/features Kotlin provides
- Why should someone move from Java to Kotlin
- What is the future direction of Kotlin



<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 content is as follows:

Closures — Closures vs Java 8 lambdas?

```
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))
```

Use Groovy closures wherever you pass lambdas in Java 8

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

@glaforge 44 Guillaume

Guillaume

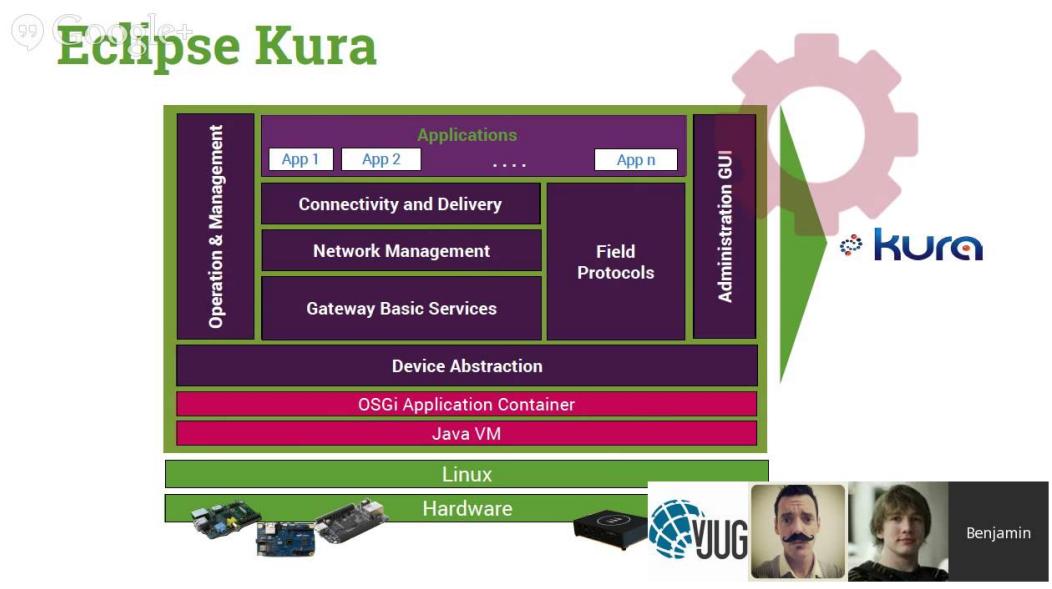
Speaker: Guillaume Laforge



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



Building the Internet of Things with Java



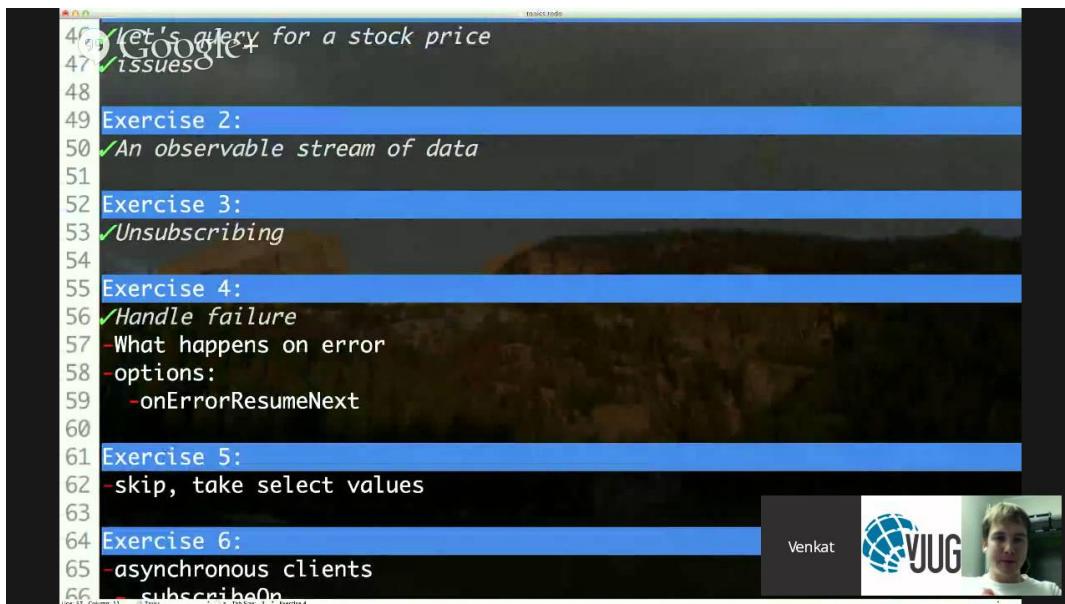
It may seem hard to get started with the Internet of Things (IoT) with so many technologies, protocols, hardware platforms, involved. In this session, Benjamin Cabé from the Eclipse Foundation will cover all you need to know



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



Reactive Programming: Creating highly responsive applications



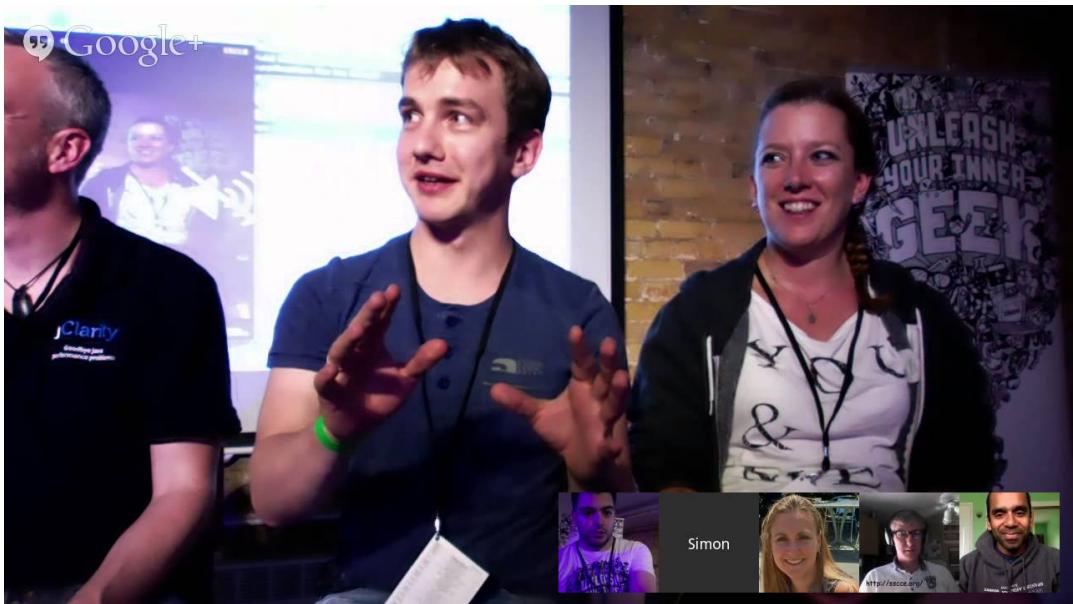
Reactive Programming is gaining a lot of attention recently, but what is it? It is a culmination of a lot of good ideas developed over the years, but brought together by the forces of recent developments



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



Shaping Java's future & vJUG party!



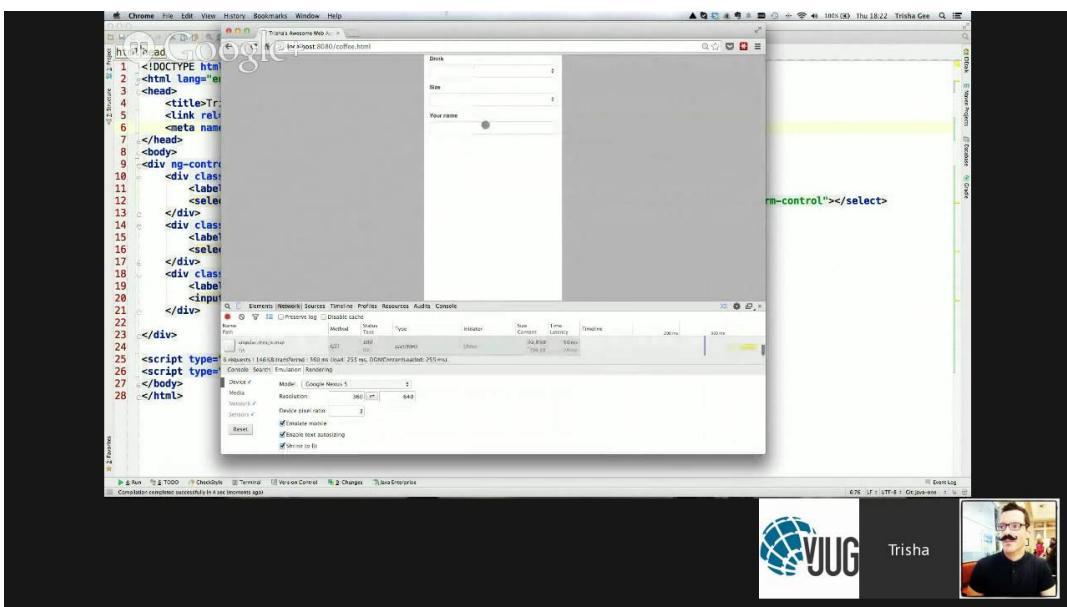
Shaping Java's future & vJUG party!



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



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



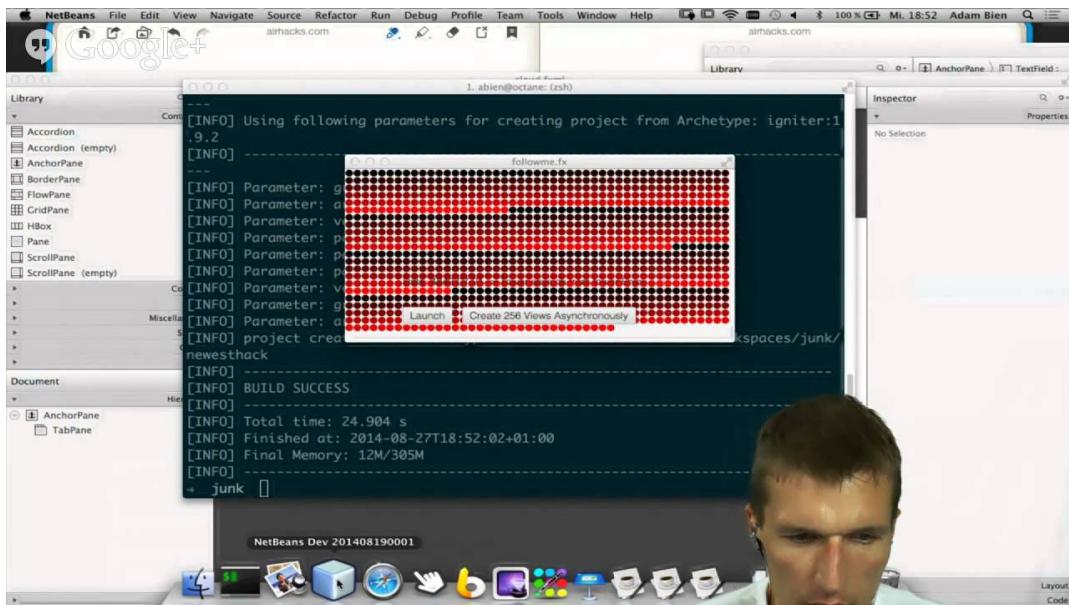
Speaker: Trisha Gee



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



Opinionated JavaFX 8



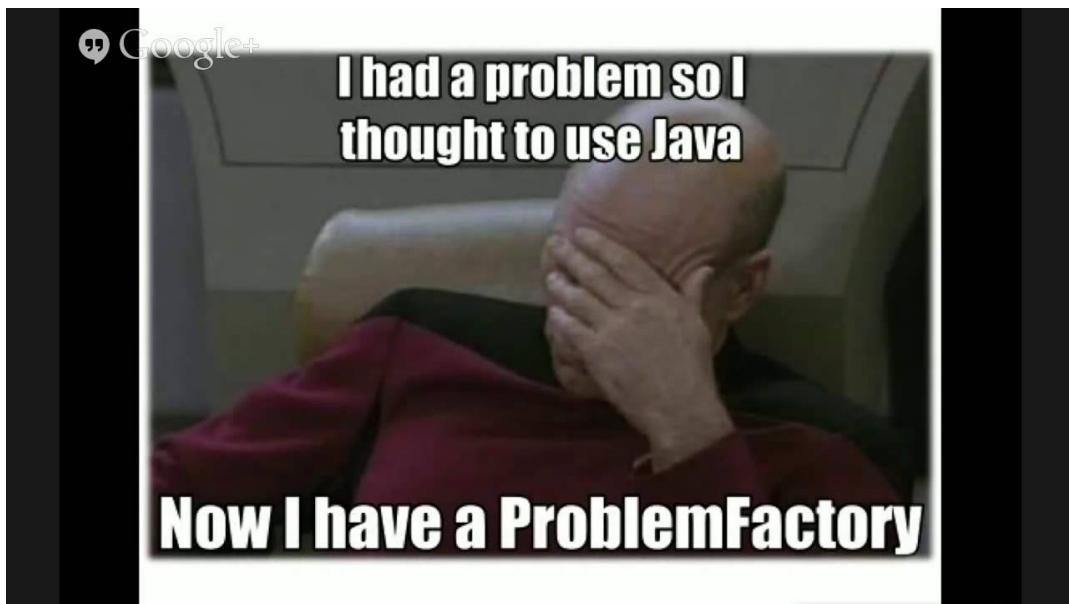
Speaker: Adam Bien



<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.



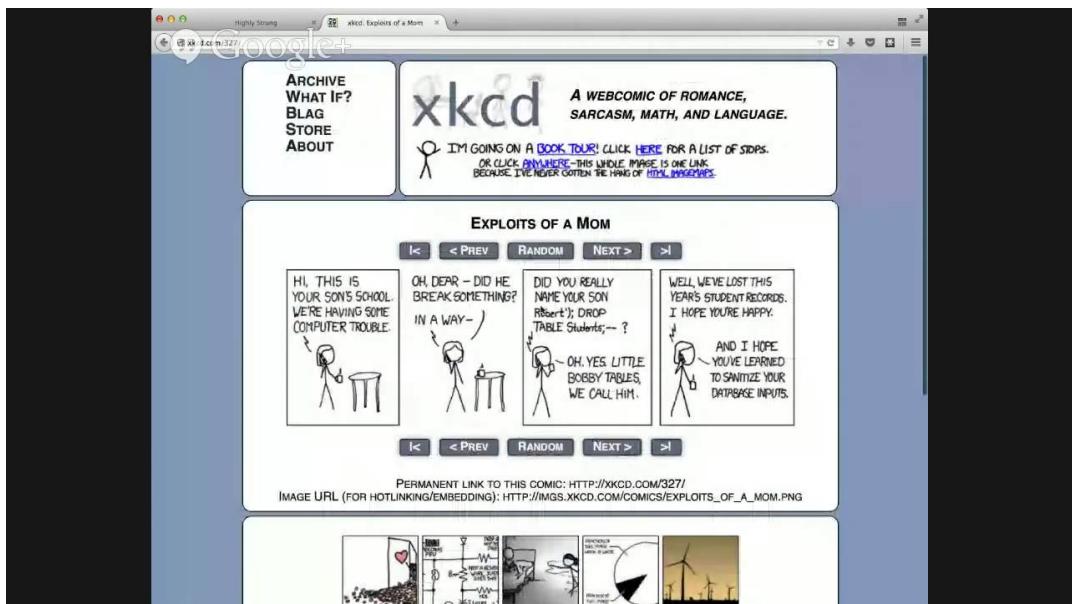
Speakers: Richard Warburton & Raoul-Gabriel Urma



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



Highly Strung: Understanding your Type System



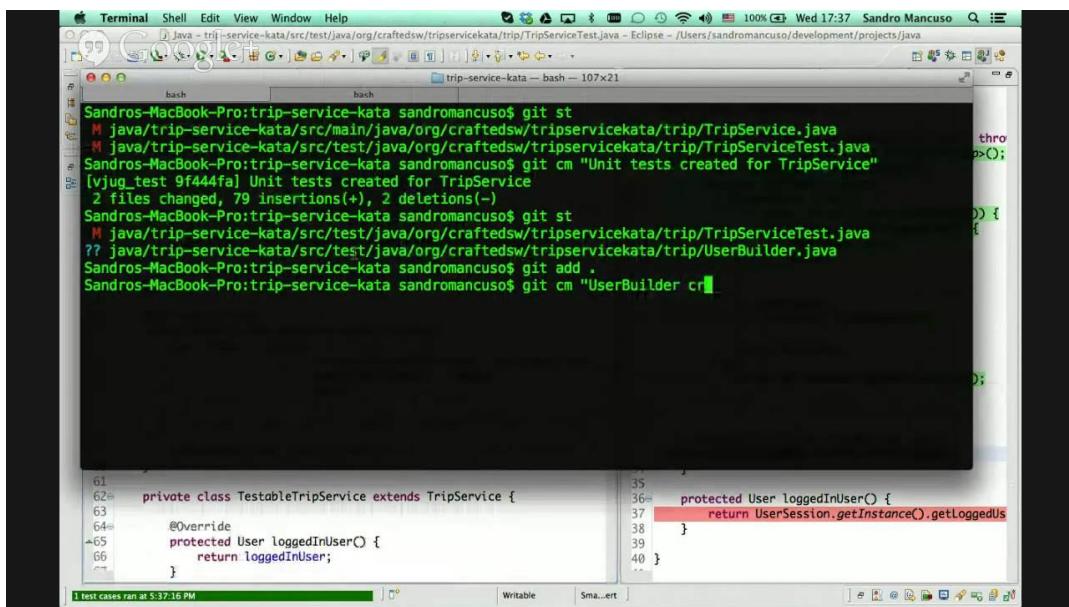
Highly Strung: Understanding your Type System



<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 tab 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 `Writable` and `SmartEdit`.

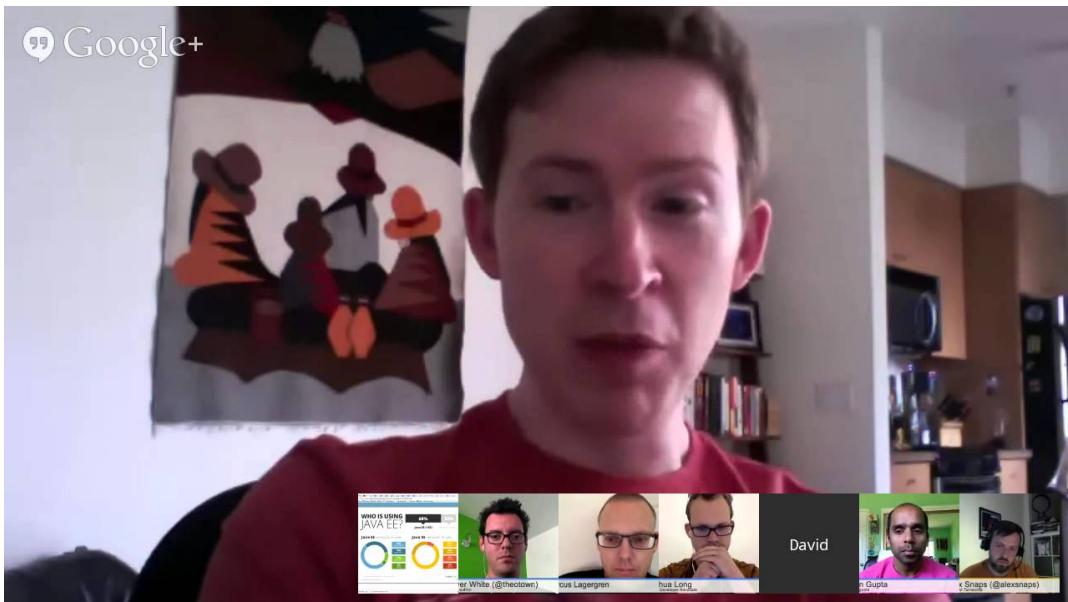
Speaker: Sandro Mancuso



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



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



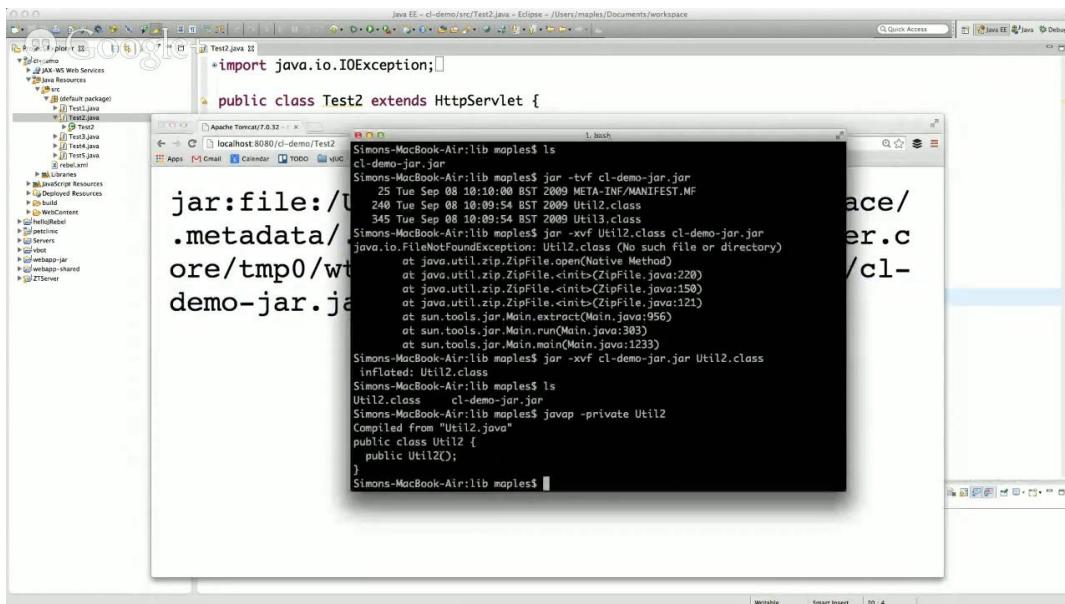
Speakers: Arun Gupta, Josh Long, Marcus Lagergren, Alex Snaps, David Blevins & Oliver White (Moderator). We look at the recent explosive publication of RebelLabs' "Java Tools and Technologies Landscape for 2014", a beautifully-designed, 56-page snapshot of what over 2000 Java developers from around the 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 Java file named 'Test.java' with the following code:

```
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 classloading events from a Mac OS X terminal session: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 also shows the current working directory as 'Simons-MacBook-Air:lib maples\$'.

Speaker: Simon Maple.



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



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



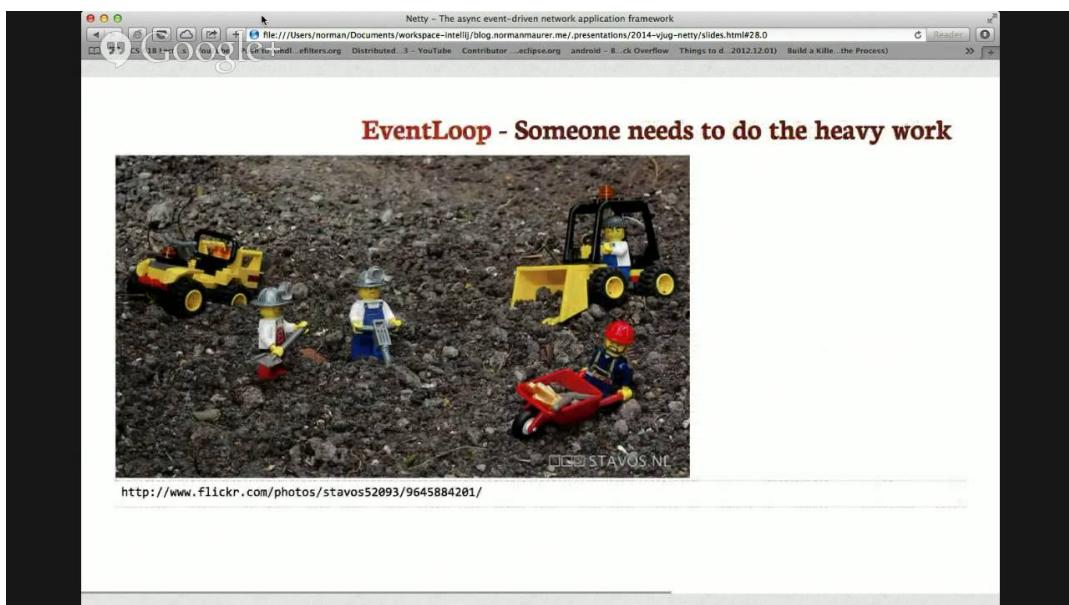
Speaker(s): Bruno Souza, Martijn Verburg and Hildeberto Mendonça, Lukas Eder & Michael Rice. (Moderated by Simon Maple)



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



Netty – The async event-driven network application framework



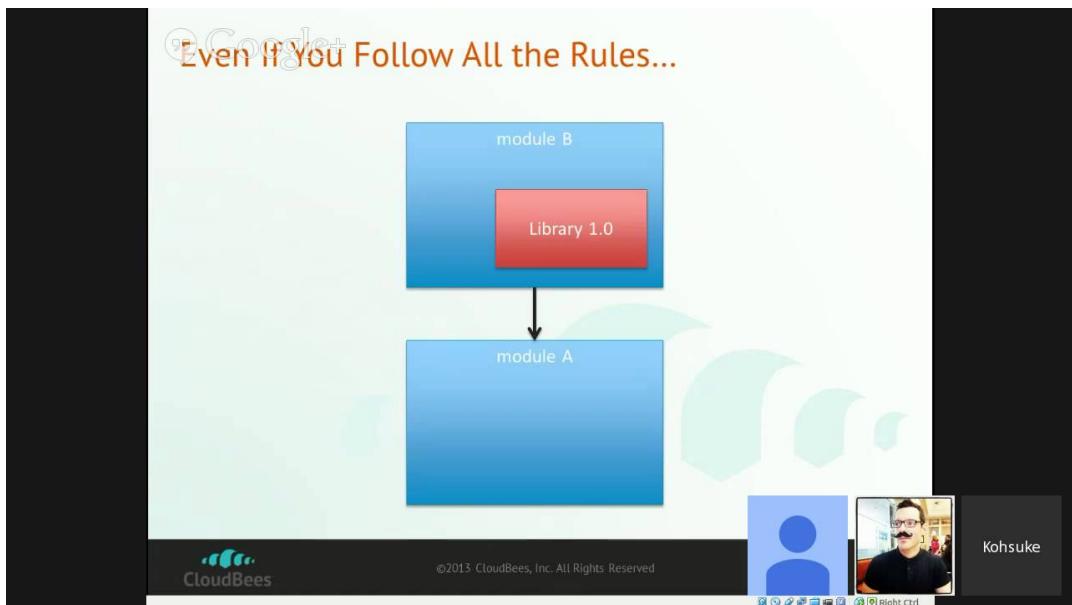
Speaker: Norman Maurer.



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



Evolving code without breaking compatibility



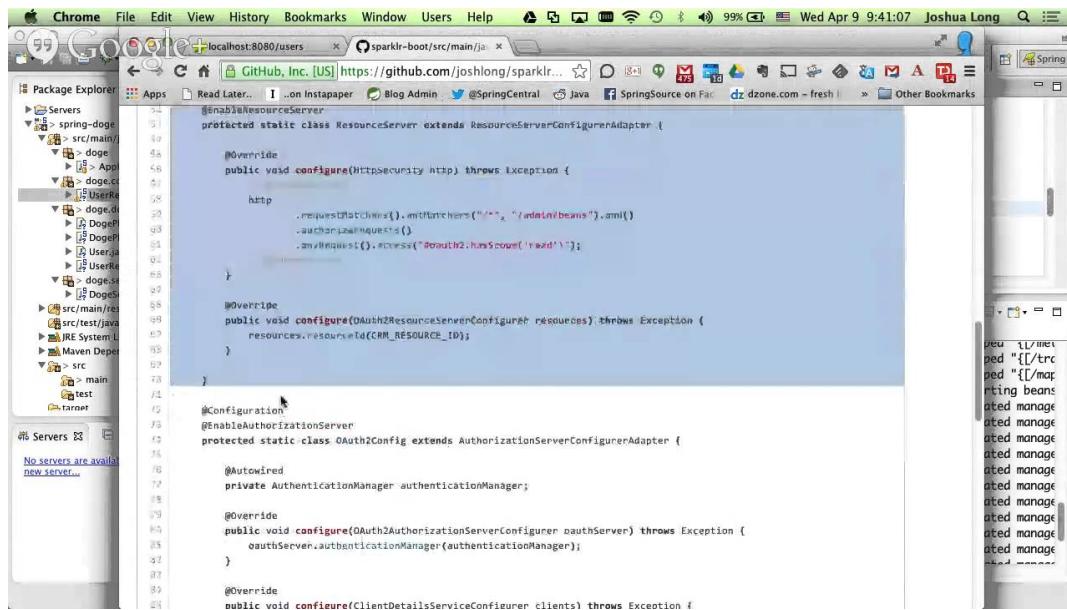
Speaker: Kohsuke Kawaguchi



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



Building Bootiful Applications with Spring Boot



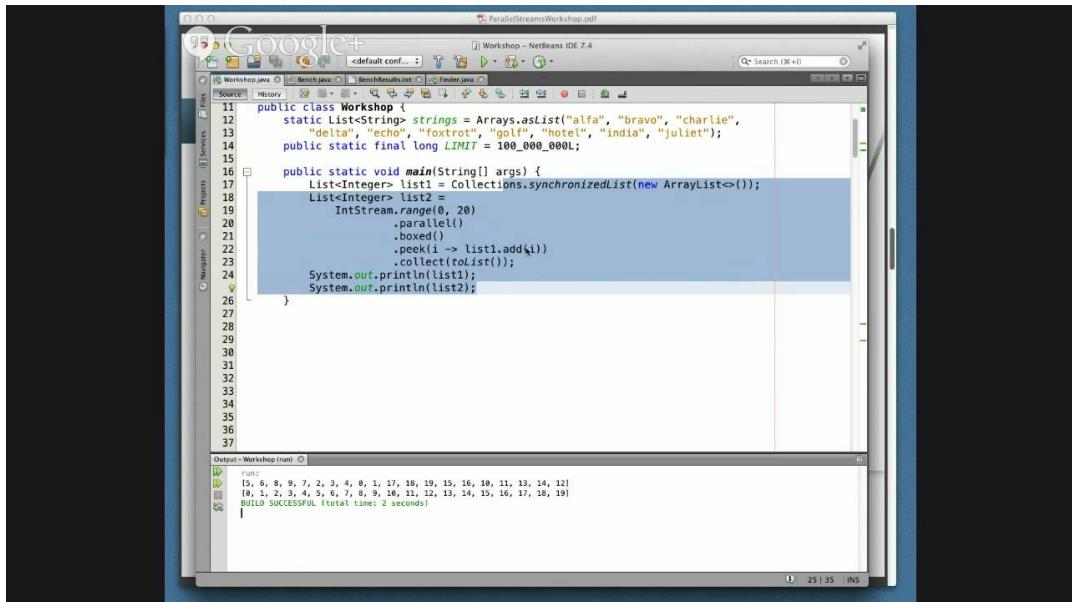
Speaker: Josh Long



<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:

```
Output - Workshop (run) [1]
[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)
```

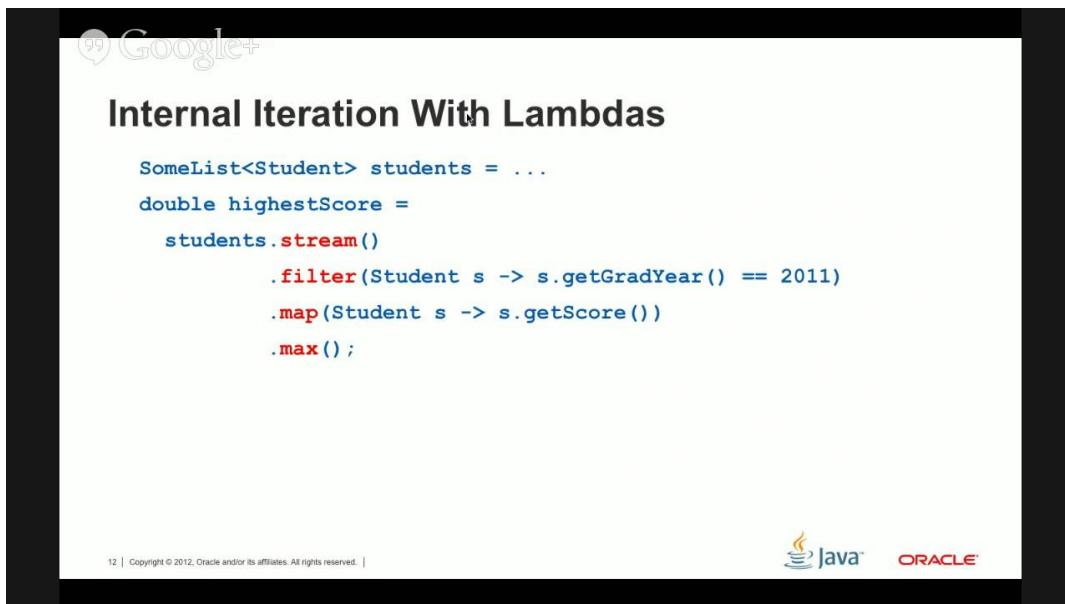
Speaker: Stuart Marks.



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



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



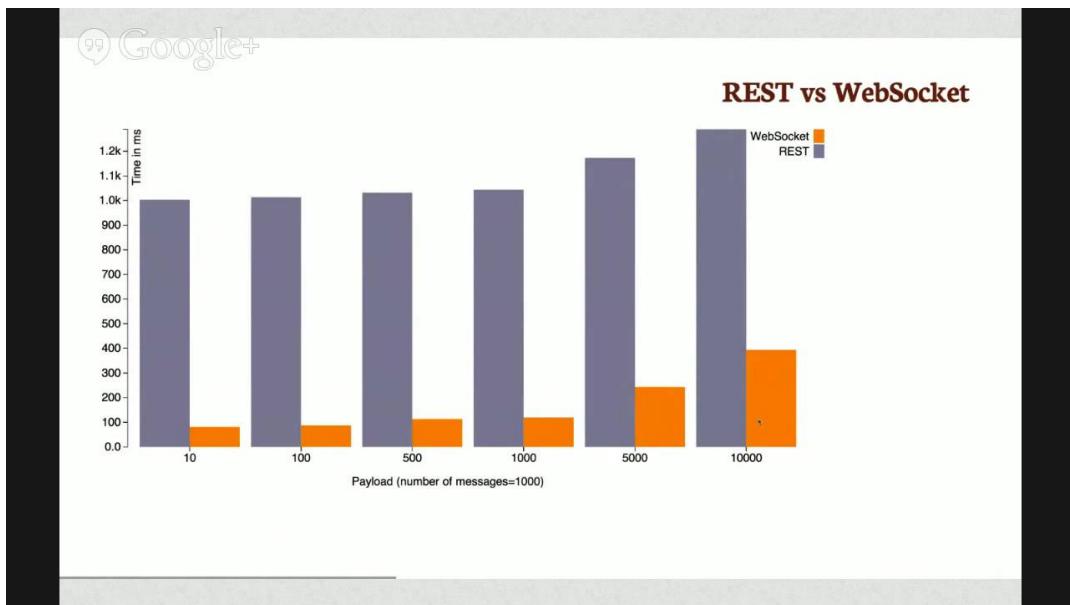
Speaker: Simon Ritter.



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



WebSocket Applications using Java EE 7



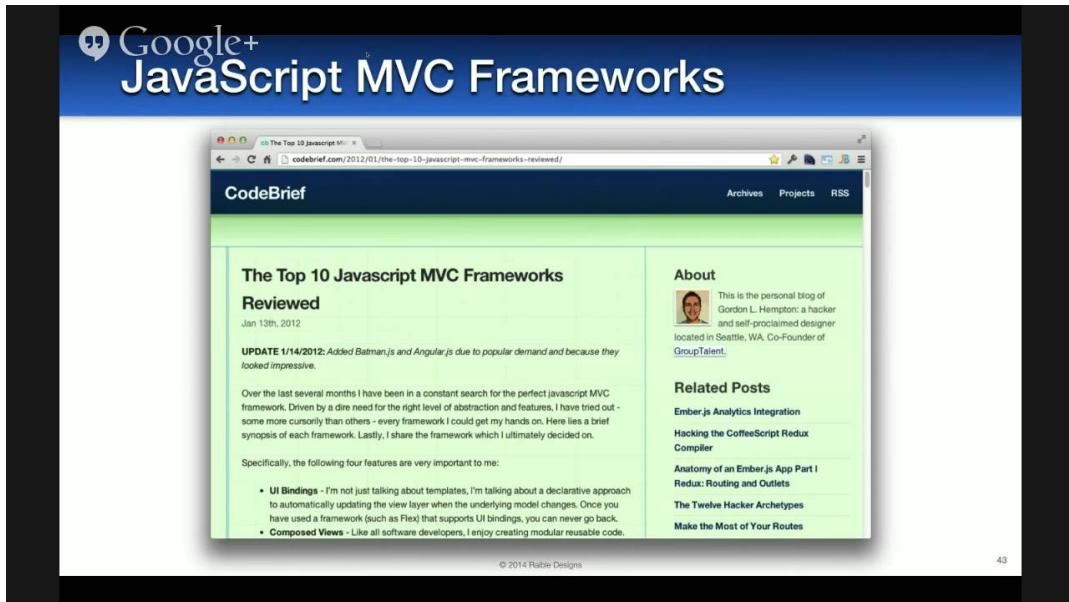
Speaker: Arun Gupta.



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



Comparing JVM Web Frameworks



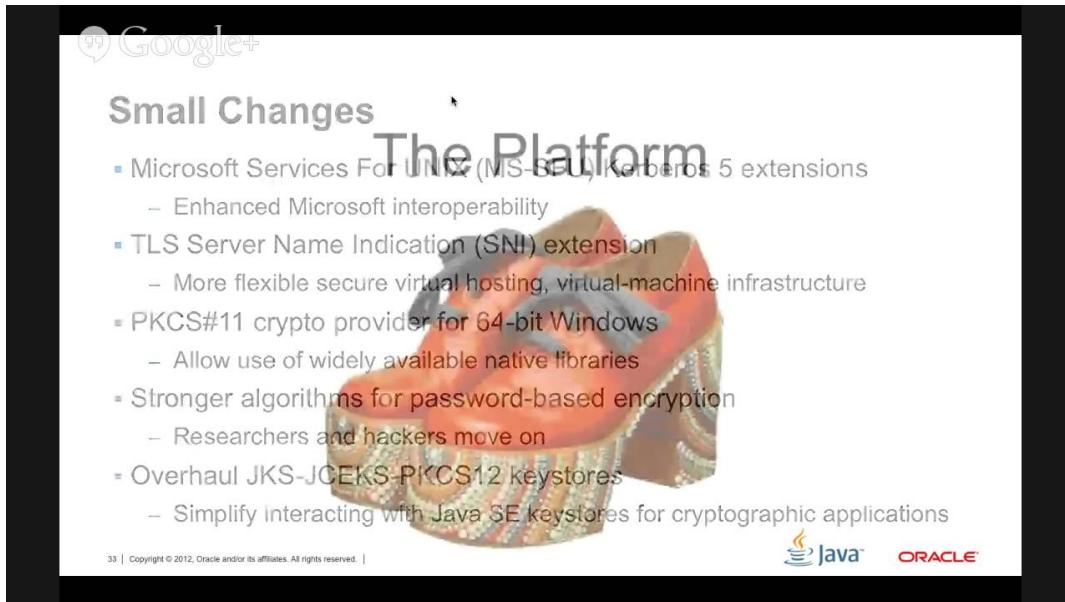
Speaker: Matt Raible



<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

Google+

Java API for RESTful Web Services 2.0

HTML5

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



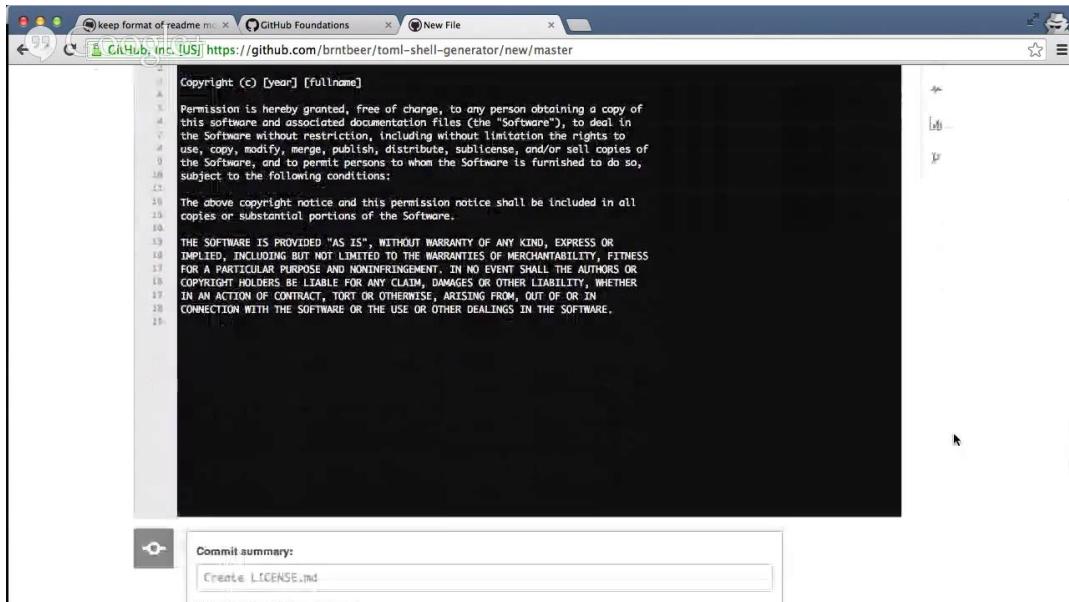
Presenter(s): Markus Eisele.



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



Drive-by Contributions



A screenshot of a GitHub commit page. The commit summary is "Create LICENSE.md". The content of the file is a standard MIT-style license notice:

```
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.
```

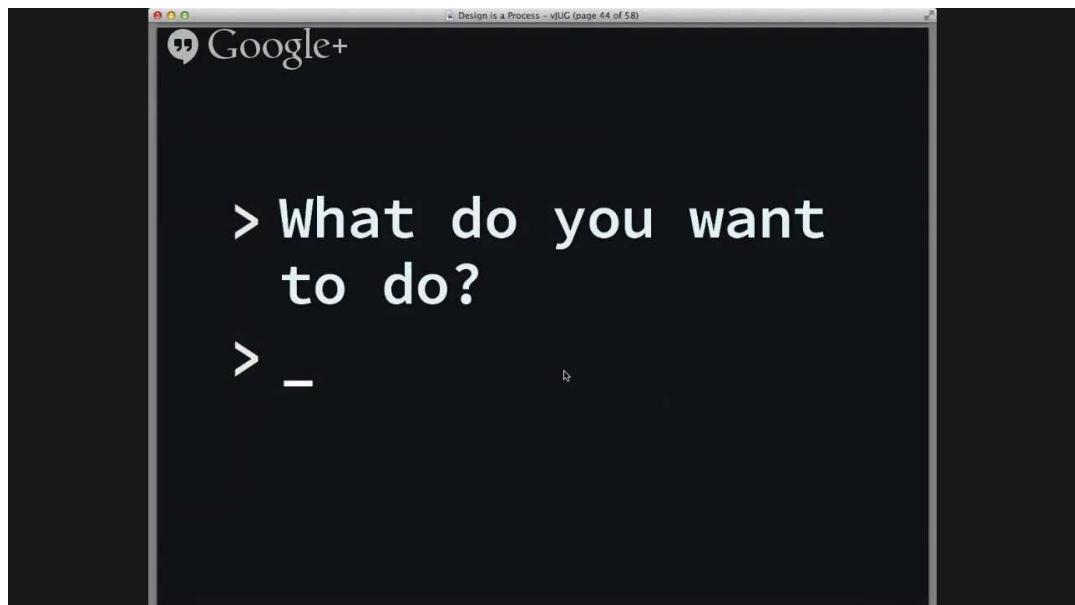
Presenter(s): Brent Beer & Matthew McCullough.



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



Design is a Process, not a Document



Presenter(s): Trisha Gee.



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



