About The Conference

The Eclipse Ecosystem is a LOT MORE than just the Eclipse IDE. Its hosts 340+ projects: MQTT, Jetty, Hudson, Vert.x, AspectJ, GeoMesa, StatET, RCP/SWT, Paho, Babel, EGerrit and many more...
Eclipse and Java™ 9

Daniel Megert
Platform and JDT Lead
IBM Research - Zurich
Eclipse and Java™ 9

- Schedule
- Running on Java 9
- What's new in Java 9
- Java 9 Features in Eclipse
Schedule

- **JSR 379 (Java SE 9 Release Contents)**
  - Early Draft Review: January 2017
  - Public Review: March 2017
  - Proposed Final Draft: May 2017
  - Final Release: July 2017

- **Feature Extension Complete: December 22, 2016**

- **Eclipse Java 9 support for Oxygen?**
Run on Java™ 9 (Demo)

- `-XX:+IgnoreUnrecognizedVMOptions`
- `--add-modules=java.se.ee`
  - Make sure to use two dashes
- `jdeps.exe (available since Java 8)`
  - e.g.: `jdeps -jdkinternals yourJar/Folder`
Install Eclipse Java™ 9 Support

- Latest Oxygen (4.7) Eclipse SDK integration build
- Install Market Place Client if not yet installed
- Go to the Marketplace and search for “Java 9 Support“
Install Eclipse Java™ 9 Support

Java 9 Support (BETA) for Oxygen

Java™ 9 support has not yet landed in our standard download packages. But you can add an early access preview to your existing Eclipse Oxygen (4.7) install.

The Eclipse Java™ 9 Support (BETA) contains the following:

- ability to add JRE and JDK 9 as installed JRE
- support for JavaSE 9 execution environment
- ability to create Java and Plug-in projects that use a JRE or JDK 9
- ability to compile modules that are part of a Java project

Notes:

- You need the most recent Oxygen (4.7) Eclipse SDK integration build
- You need a recent Java™ 9 JDK, from https://jdk9.java.net/
- Recent Java 9 previews require you to add `--add-modules=java.se.ee` (and `modules java.se.ee` for older Java 9 builds) to the vmargs when launching Eclipse. see https://www.eclipse.org/eclipse/news/4.6/platform.php#java-9
What's New in Java™ 9

- Milling Project Coin
- Modularized JDK
- Platform Module System
- HTTP2/Client
- Platform Logging API and Service
Milling Project Coin (JEP 213)

- @SafeVarargs can be applied to private methods
- final or effectively final variables in try-with-resources
- Allow diamond with anonymous classes if the argument type of the inferred type is denotable
- Disallow '_' as a one-character identifier (error)
- private methods in interfaces
Modularized JDK

- Many attempts have been made before
- The source code has been rearranged in the repository
- The JDK classes are now built as modules
- There are new file formats like JMOD and JIMAGE
- The runtime layout has been changed
- There is no longer a JDK/JRE distinction
Platform Module System

- **High level goals:**
  - Reliable configuration of applications
    - Replace the class-path with a new mechanism allowing components to declare explicit dependencies between each other
  - Strong encapsulation
    - Allow components to declare which of its **public** types are accessible
Platform Module System

- Modules are defined by a new type of program component
  - The runtime recognizes classes, interfaces, packages, and now modules

- There are updates to the Java language and JVM specifications
  - Introducing new keywords to declare modules, and new behaviors for VM accessibility

- Modules are named in a familiar convention, e.g. com.example.app
Platform Module System

- Modules are defined in module-info.java
  - Lives in the module root directory
  - Compiler produces module-info.class with
    - attribute_name_index
    - explicit module info
    - requires table and requires_index
    - exports table
    - exports_to_count
    - internal Packages
Module vs. OSGi Bundle

- Module information and MANIFEST can co-exist
- But: there's overlap in the metadata
- Most of the metadata can be synchronized
- OSGi still more powerful
- Same JAR can be used in both scenarios
Evaluate the Sessions
Sign in and vote at eclipsecon.org

- 1 0 + 1