Java 9 and the impact on Maven Projects

Robert Scholte (@rfscholte) - current chairman Maven
Hervé Boutemy (@hboutemy) - previous chairman Maven

“The success of Java 9 depends on the adoption by IDEs and buildtools like Maven”

Apache Maven Project

http://maven.apache.org/

Early Access releases

* Jigsaw
  * Since September 2015
  * https://jdk9.java.net/jigsaw/
  * ZIP
* Java9
  * Since April 2014
  * https://jdk9.java.net/download/
  * Executable

Challenge/Strategy Maven and Java9

- Support Maven 3.0 and above
- Only upgrades of plugins

Apache Maven Project

http://maven.apache.org/

Standard Java upgrade

* Set Java Runtime for Maven
  * JAVA_HOME=/path/to/jdk-9
* Verify source/target of maven-compiler-plugin
  * >= 6 (minimum jdk-9), okay
  * < 6, must fork to preferred JDK

* Maven JRE <> maven-compiler-plugin JDK

JEP 260: Encapsulate Most Internal APIs

Summary

Make most of the JDK's internal APIs inaccessible by default but leave a few critical, widely-used internal APIs accessible, until supported replacements exist for all or most of their functionality.

Results so far

- First java9-ea releases: close to no issues
- First jigsaw-ea release: ~99% of the Java Maven projects failed to compile.
- Cause: JavacToolProvider + (System)Classloader
- Fixed and released within 72h!
- ***zero*** lines of code changed in Maven core codebase to run on Java9

Apache Maven Project

http://maven.apache.org/

~10% of JEPs related to Maven

- 220: Modular Run-Time Images *
- 223: New Version-String Scheme
- 226: UTF-8 Property Files
- 238: Multi-Release JAR Files
- 247: Compile for Older Platform Versions
- 261: Module System *
- 282: jlink: The Java Linker *

* Part of JSR 376: Java Platform Module System (Project jigsaw)

Apache Maven Project

http://maven.apache.org/

Agenda

- 220: Modular Run-Time Images *
- 223: New Version-String Scheme
- 226: UTF-8 Property Files
- 238: Multi-Release JAR Files
- 247: Compile for Older Platform Versions
- 261: Module System *
- 282: jlink: The Java Linker *

* Part of JSR 376: Java Platform Module System (Project jigsaw)

JEP 223: New Version-String Scheme (project Verona)

Summary

Revise the JDK's version-string scheme so that it is easier to distinguish major, minor, and security-update releases.

Major (GA) Example

System property	Existing	Proposed
java.version	1.9.0	9
java.runtime.version	1.9.0-b100	9+100
java.vm.version	1.9.0-b100	9+100
java.specification.version	1.9	9
java.vm.specification.version	1.9	9

version.split("\\.")[1]

* Caused by: java.lang.ArrayIndexOutOfBoundsException: 1
* at org.codehaus.plexus.archiver.zip.AbstractZipArchiver….

- maven-archiver-3.0.1
  * maven-jar-plugin-3.0.0
  * maven-war-plugin-3.0.0
  * maven-ear-plugin-xxx
  * maven-assembly-plugin-xxx
- maven-javadoc-plugin-2.10.4
- …

JEP 247: Compile for Older Platform Versions

Summary

Enhance javac so that it can compile Java programs to run on selected older versions of the platform.

The problem

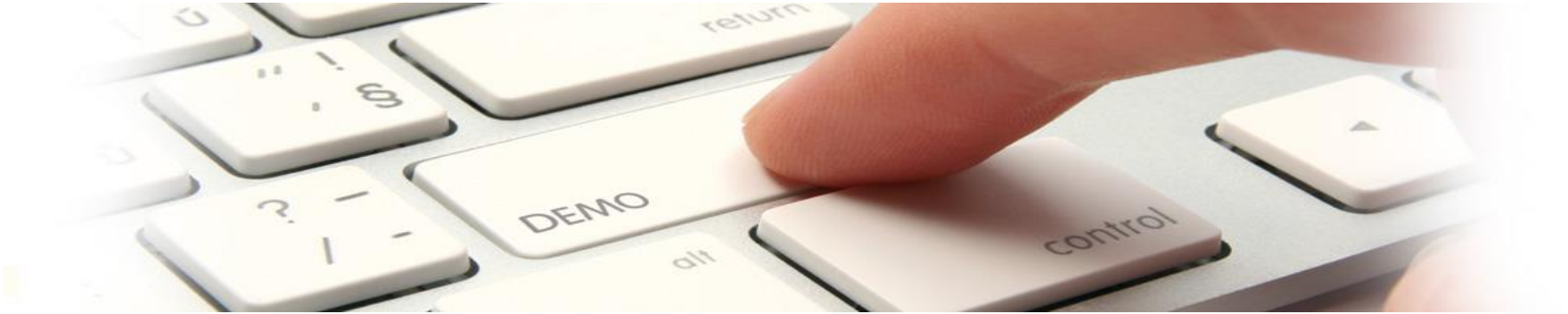
Leaking classes of the JDK

The official required javac arguments

* `-source N`
* `-target N`
* `-bootclasspath <bootclasspath-from-N>`

Apache Maven Project

http://maven.apache.org/



Apache Maven Project

http://maven.apache.org/

Available Maven Solutions (1)

- Always compile with the matching JDK version
- Configure maven-toolchain-plugin
- Configure toolchains.xml
  - `${user.home}/.m2/toolchains.xml`
  - `${maven.home}/conf/toolchains.xml` (since 3.3.1)

Available Maven Solutions (2)

- Verify code with jre signatures
- Configure animal-sniffer-maven-plugin
  - Signature for N
  - Execution-block with ‘check’ goal

Solution

* “We defined a new command-line option, -release, which automatically configures the compiler to produce class files that will link against an implementation of the given platform version. For the platforms predefined in javac, `–release N` is equivalent to `-source N -target N -bootclasspath <bootclasspath-from-N>`.”

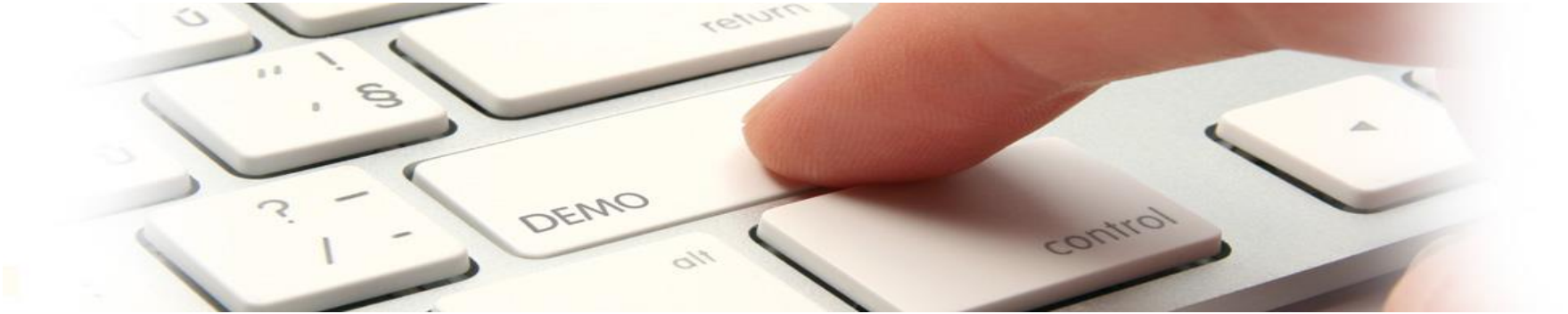
9-ea+135-jigsaw: --release (gnu-style)

Apache Maven Project

http://maven.apache.org/

maven-compiler-plugin 3.6.0

- Configuration: `<release>N</release>`
- Property: `maven.compiler.release`
- If `source/target` AND `release` are specified, `release` is used.



Apache Maven Project

http://maven.apache.org/

Forward compatibility

```
if ( javaVersion >= 1.8 ) {
  // calculation based on Date-Time API (preferred)
}
else {
  // calculation based on Date
}
```

source/target: 1.7

JDK: 1.8
JDK: 1.7 with reflection

Apache Maven Project
http://maven.apache.org/

JEP 238: Multi-Release JAR Files

Summary

Extend the JAR file format to allow multiple, Java-release-specific versions of class files to coexist in a single archive.

JAR structure

jar root
- A.class
- B.class
- C.class
- D.class
- META-INF
  - versions
    - 9
      - A.class
      - B.class
    - 10
      - A.class

1 to 1 translation

project root
 src/main/java
 - A.java
 - B.java
 - C.java
 - D.java
 src/main/java9
  - A.java
  - B.java
 src/main/java10
  - A.java

* Will work with Maven execution-blocks, not with (all) IDEs

IDE friendly POC

multimodule root
multirelease-base/src/main/java
- A.java
- B.java
- C.java
- D.java

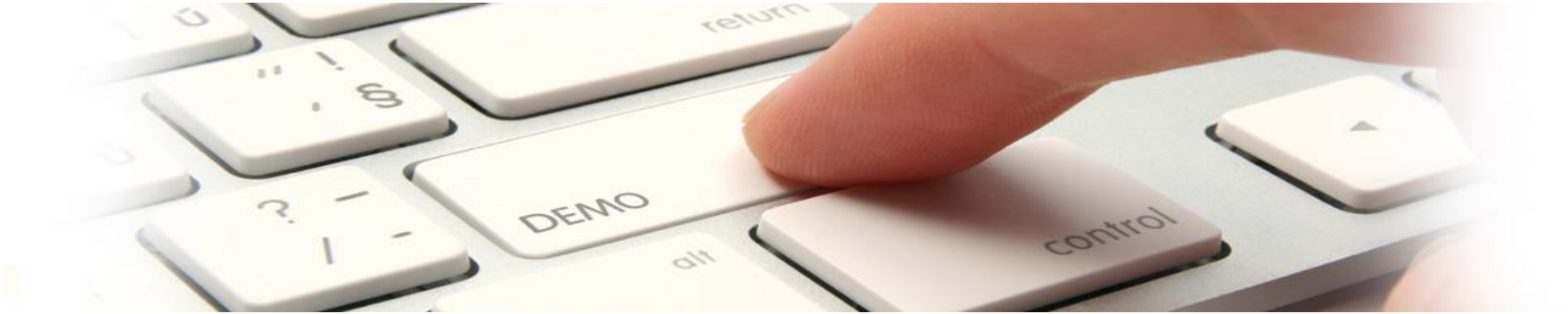
multirelease-nine/src/main/java
- A.java
- B.java

multirelease-ten/src/main/java
- A.java

multirelease/src/assembly/mvjar.xml

* https://github.com/hboutemy/maven-jep238

Apache Maven Project
http://maven.apache.org/



Apache Maven Project

http://maven.apache.org/

Improvements (in progress)

- Introduce new packaging (no install/deploy)
- Merge dependencies in JDK profiles
- Remove assembly descriptor

JEP 220: Modular Run-Time Images

Summary

Restructure the JDK and JRE run-time images to accommodate modules and to improve performance, security, and maintainability. Define a new URI scheme for naming the modules, classes, and resources stored in a run-time image without revealing the internal structure or format of the image. Revise existing specifications as required to accommodate these changes.

Removal of tools.jar

* Most Apache maven-plugins already have a lot of fallback scenarios for a long time.
* Projects that might suffer
  * custom doclettags
  * …

Tools.jar profile

```
<profile>
        <id>default-tools.jar</id>
        <activation>
                <jdk>(,9)</jdk> <!-- System.getProperty( "java.version" ) -->
        </activation>
        <dependencies>
                <dependency>
                        <groupId>com.sun</groupId>
                        <artifactId>tools</artifactId>
                        <version>1.4.2</version>
                        <scope>system</scope>
                        <systemPath>${java.home}/../lib/tools.jar</systemPath>
                </dependency>
        </dependencies>
</profile>
```

Apache Maven Project

http://maven.apache.org/

JEP 261: Module System

Summary

Implement the Java Platform Module System, as specified by JSR 376, together with related JDK-specific changes and enhancements.

In a nutshell

* module-info.java
  * Specify exposed packages
  * Specify required modules (buildtime + runtime)
  * Specify usage and implementation of SPIs

Module Declarations Example (original proposal)

```
module M.N {
  requires A.B;
  requires public C.D;
  requires static E.F;
  requires public static G.H;

  exports P.Q;
  exports R.S to T1.U1, T2.U2;
  exports dynamic PP.QQ;
  exports dynamic RR.SS to T1.U1, T2.U2;

  uses V.W;
  provides X.Y with Z1.Z2;
  provides X.Y with Z3.Z4;
}
```

Module Declarations Example (current proposal)

```
[|weak|open] module M.N {
  requires A.B;
  requires transitive C.D;
  requires optional E.F;
  requires transitive optional G.H;

  exports P.Q;
  exports R.S to T1.U1, T2.U2;

  uses V.W;
  provides X.Y with Z1.Z2;
  provides X.Y with Z3.Z4;
}
```

Requires Modifier ‘transitive’

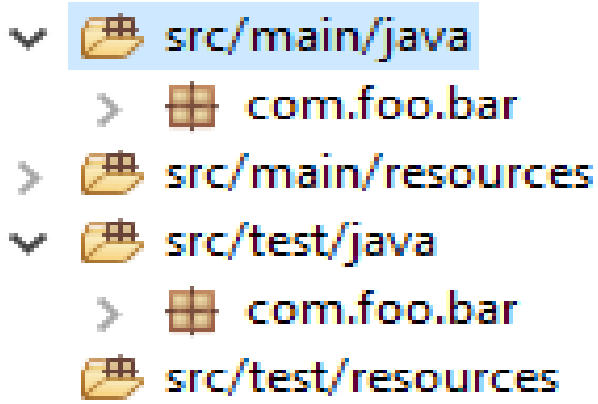
- A hint for other projects using this as dependency (no effect on this project)
- Maven best practice: don’t trust transitive dependencies; specify dependency for every used class
- Concept adopted by Java9:
  - Modules are unaware of dependency tree
  - All (non-transitive) modules MUST be specified

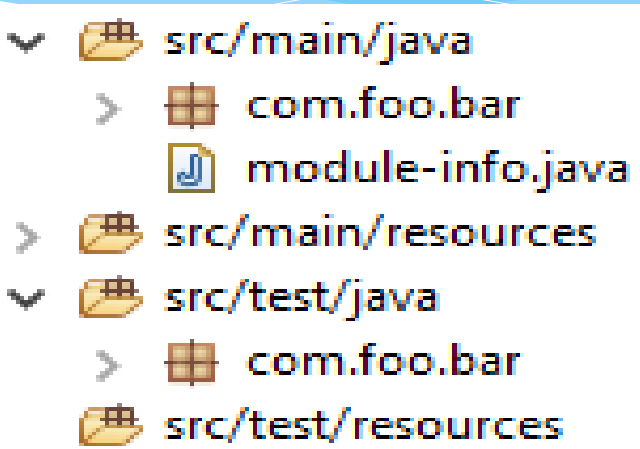
Requires Modifier ‘optional’

* comparable with dependency.optional

* Difference provided versus optional
  * Buildtime: no difference
  * Runtime:
    * provided must be available (servlet-api)
    * optional might be available (spring-boot deps)

Common usecases

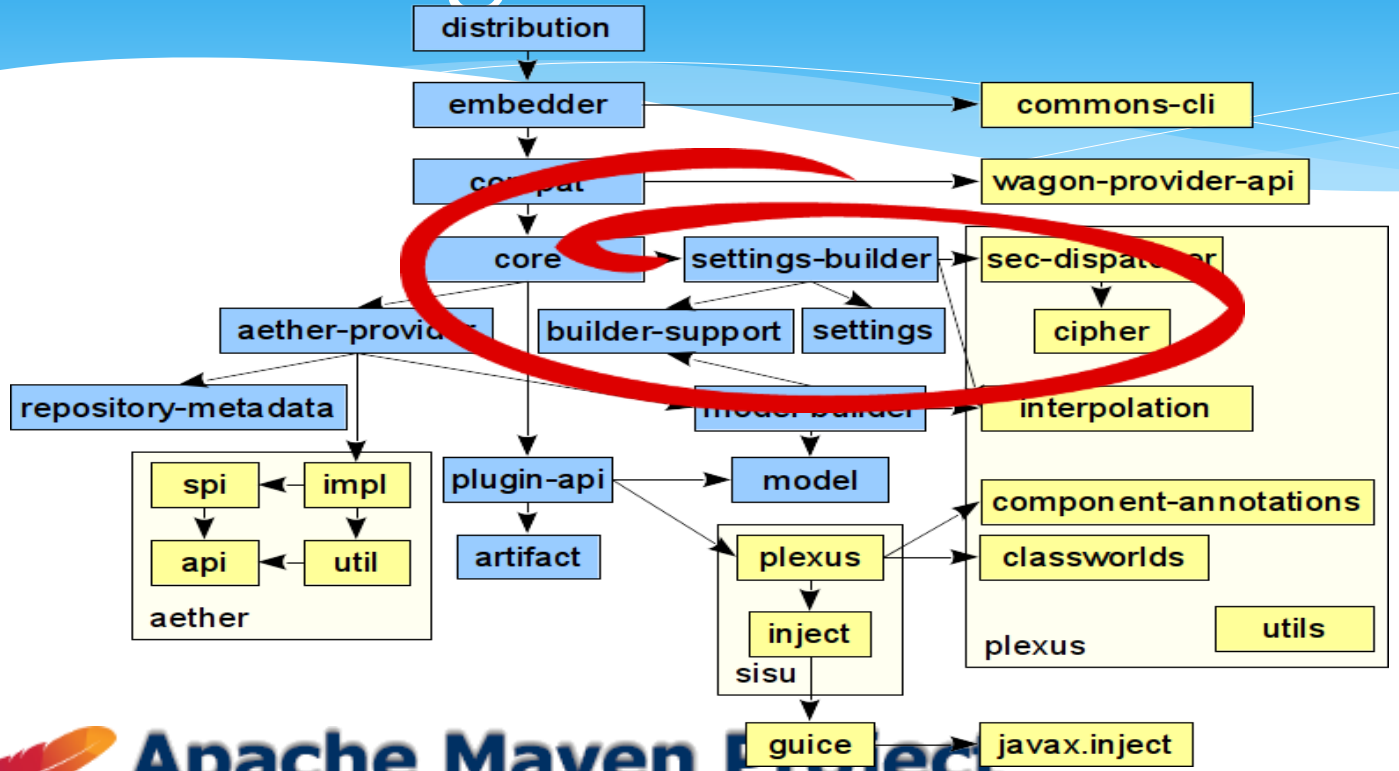


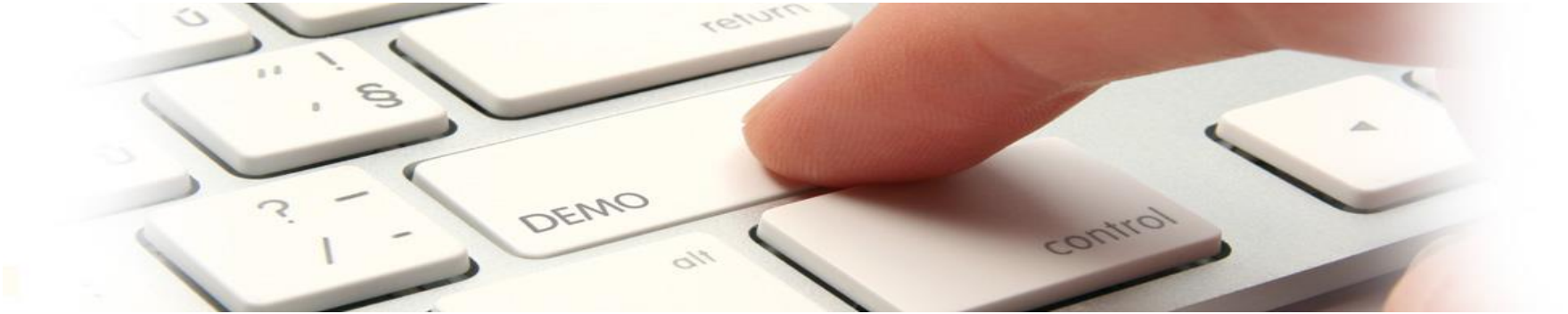


maven-compiler-plugin 3.6.0

- `:compile`, switch to modulepath when compiling module-info.java
- `:test-compile`, switch to modulepath + classpath when target/classes/module-info.class exists

Building Maven with Maven



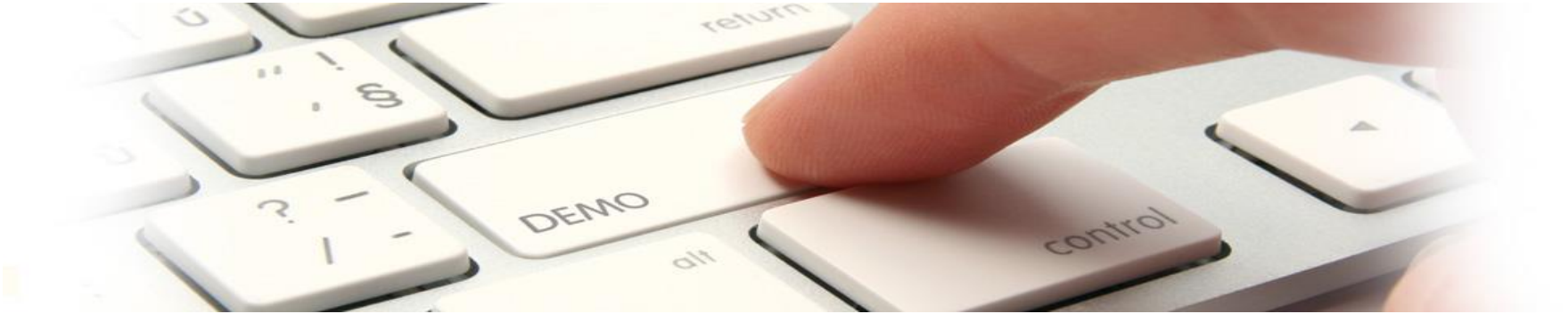


Apache Maven Project

http://maven.apache.org/

Discover moduleName

- Documentation
- Central/Repository Managers? Archiva/Artifactory/Nexus
- maven-dependency-plugin:list



Apache Maven Project

http://maven.apache.org/

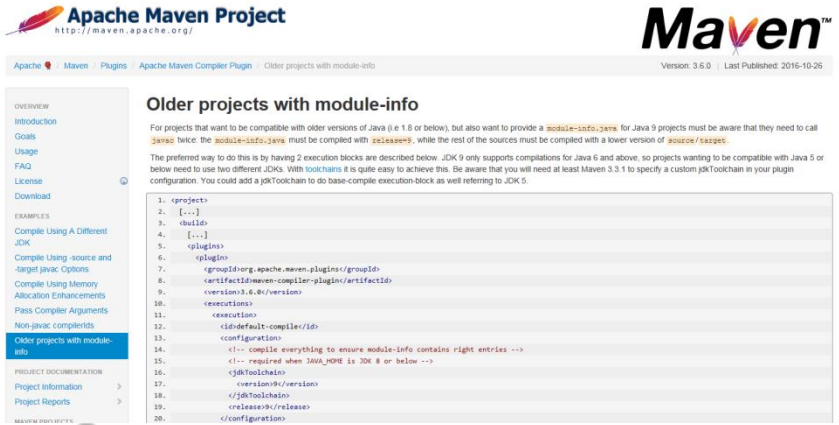
For library/framework/maven-plugin/… builders

- The lower the supported Java version, the more projects can use it
- The lower the supported Java version, the less Java features can be used.

* Can we add module-info? Yes!

Backwards compatible libraries

- module-info (-release 9)
- (other) java sources (source/target < 9)
- Maven Recipe



Dependencies and classpath

- Classpath order:
  - all direct dependencies
  - all first level indirect dependencies
  - all second level indirect dependencies
  - ...
- Locating a class:
  - iterate over classpath (in same order); first match wins

Dependencies and modulepath

- Modulepath: packages are mapped to a module
  - Duplicate packages will result in an Exception
- Locating a class:
  - find module based on package; get class from module

GOTCHA: Dependency Excludes

* ‘requires’ means it is realy required!

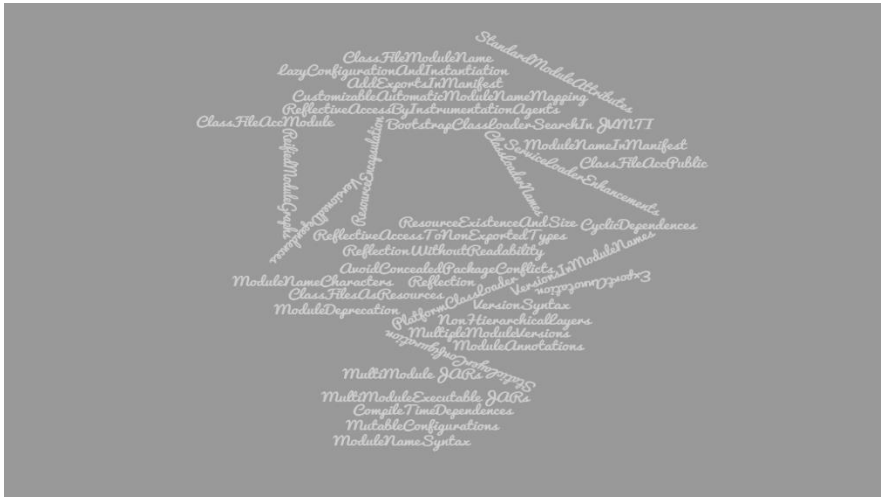
* As enduser:
  - No option to ignore requirements of third party libraries
  - No option to fix/choose package collisions between transitive dependencies

Apache Maven Project

http://maven.apache.org/

Java Platform Module System: Issue Summary

* http://openjdk.java.net/projects/jigsaw/spec/issues/



Is my project Java9-ready?

- Is it using internal classes?
  - maven-jdeps-plugin-3.0.0
- Does it have duplicate
  - extra-enforcer-rule > banDuplicateClasses
- Does it require certain Java9 features
  - Upgrade the matching plugins

Apache Maven Project

http://maven.apache.org/

The Apache Maven project tasklist

- Most features should work with Maven 3.0
- Some require Maven 3.3.1 due to improved toolchains support
- Large number of new features already developed in plugins, though not always released.
- New recipes “The Maven Way™”

Developer awareness

- Maven-compiler-plugin selecting modulePath or/and classPath
- When Java9/Jigsaw fails:
  - Usage internal APIs
  - Duplicate (exported) packages

Thank you

Give it a try!

Send feedback, issues & wishes

Apache Maven Project

http://maven.apache.org/