

Guide

To JAR Hell and Back

Show that it works on Java 8:

```
mvn clean install
java -cp 'app/*' monitor.Main
```

In a different terminal:

```
watch -n 1 curl http://localhost:4567/stats/xml
```

Migration

Building on Java 9

1. start building with Java 9
 - a. show `mvn -v`
 - b. set `JAVA_HOME` in `.mavenrc`

```
JAVA_HOME="/opt/jdk-9"
```

- c. show `mvn -v`

2. create a profile for Java 9+

```
<profiles>
  <profile>
    <id>java-9+</id>
    <activation>
      <jdk>[9,)</jdk>
    </activation>
  </profile>
</profiles>
```

3. set `release` to 9 in profile

```
<properties>
  <maven.compiler.release>9</maven.compiler.release>
</properties>
```

Internal APIs

1. `mvn clean install`
2. fix use of `sun.misc.BASE64Encoder`

```
Base64.getEncoder().encodeToString(content.getBytes());
```

3. note warning message in *monitor.utils*
4. fix warning messages in *UtilsTest* in *monitor.utils*
 - a. create a profile for Java 9
 - b. show that `illegal-access=deny` disallows access
 - c. configure Surefire to use `--add-opens`

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <configuration>
        <argLine>
          --illegal-access=deny
          --add-opens=java.base/java.lang=ALL-UNNAMED
        </argLine>
      </configuration>
    </plugin>
  </plugins>
</build>
```

- d. better, yet: update Mockito to `2.8.47`

Java EE Modules

1. note error `package javax.xml.bind.annotation is not visible`
 - a. open `StatisticsEntity`
2. add comment `// from module java.xml.bind:` above `import XmlElement`
3. add `java.xml.bind` module for annotations in *monitor.rest*
 - a. create a profile for Java 9
 - b. configure compiler to use `--add-modules java.xml.bind` to fix `package javax.xml.bind.annotation is not visible`

```

<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <compilerArgs>
          <arg>--add-modules=java.xml.bind</arg>
        </compilerArgs>
      </configuration>
    </plugin>
  </plugins>
</build>

```

Patch Modules

1. observe *error: cannot find symbol* for missing type `Generated`
 - a. open `StatisticsEntity`
2. add `java.xml.ws.annotation` to `monitor.rest` to fix missing type `Generated`
 - a. observe that error contains no hint of the containing module
 - b. look up module in documentation
 - c. add comment `// from module java.xml.ws.annotation:` above `import Generated`
 - d. add `java.xml.ws.annotation`

```
<arg>--add-modules=java.xml.ws.annotation</arg>
```

- e. observe new missing types problem
3. patch `java.xml.ws.annotation` with JSR classes

```

<arg>--patch-
module=java.xml.ws.annotation=${settings.localRepository}/com/google/code/findbugs/
jsr305/3.0.2/jsr305-3.0.2.jar</arg>

```

4. replace `maven.compiler.release` with `maven.compiler.source` and `maven.compiler.target` in parent POM

Running on Java 9

Start running with Java 9:

```
java9 -cp 'app/*' monitor.Main
```

1. fix cast to `URLClassLoader` by replacing code in `logClassPathContent` with

```
String[] classPath = System.getProperty("java.class.path").split(":");
String message = Arrays.stream(classPath)
    .map(url -> "\t" + url)
    .collect(joining("\n", "Class path content:\n", "\n"));
System.out.println(message);
```

2. add module `java.xml.bind`

```
java9 --add-modules java.xml.bind -cp 'app/*' monitor.Main
```

3. note that `--add-exports`, `--add-opens`, `--add-modules`, `--patch-module` usually carry from build to compile time

Using Java 10

1. edit `.mavenrc` to point to Java 10
2. show `mvn -v`
3. limit parent POM profile for Java 9+ to 9 and copy paste to create one for 10

```
<profile>
  <id>java-10</id>
  <activation>
    <jdk>10</jdk>
  </activation>
  <properties>
    <maven.compiler.source>10</maven.compiler.source>
    <maven.compiler.target>10</maven.compiler.target>
  </properties>
</profile>
```

4. to use `var` somewhere, activate java-10 profile in IntelliJ and reimport
5. build with `mvn clean install`
6. fix error in `monitor.utils` by updating Mockito to 2.18.3
7. in parent POM's Java 10 profile update ASM to preempt compiler problems

```

<build>
  <plugins>
    <!-- if compilation fails, try newer version of ASM
         https://stackoverflow.com/q/49398894/2525313
    -->
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <dependencies>
        <dependency>
          <groupId>org.ow2.asm</groupId>
          <artifactId>asm</artifactId>
          <version>6.1.1</version>
        </dependency>
      </dependencies>
    </plugin>
  </plugins>
</build>

```

8. run with

```
java10 --add-modules java.xml.bind -cp 'app/*' monitor.Main
```

Using Java 11

1. edit `.mavenrc` to point to Java 11
2. show `mvn -v`
3. in parent POM update compiler plugin to 3.8.0
remove ASM update in profile `java-10`
4. in parent POM copy paste profile to create one for 11

```

<profile>
  <id>java-11</id>
  <activation>
    <jdk>11</jdk>
  </activation>
  <properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>
</profile>

```

5. build with `mvn clean install`
6. replace `--add-modules` with third-party dependencies in `monitor.rest`'s POM:

```

<dependency>
  <groupId>javax.annotation</groupId>
  <artifactId>javax.annotation-api</artifactId>
  <version>1.3.1</version>
</dependency>
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.0</version>
</dependency>
<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-core</artifactId>
  <version>2.3.0</version>
</dependency>
<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-impl</artifactId>
  <version>2.3.0</version>
</dependency>

```

7. run with

```
java11 -cp 'app/*' monitor.Main
```

8. observe that **xml** endpoint does not return anything, but **json** does

9. activate **java-11** profile, reimport, and launch from IntelliJ

10. in **MonitorServer** change **catch (JAXBException ex)** to **catch (Throwable ex)** and observe the error

11. add this dependency:

```

<!-- HOW?! -->
<dependency>
  <groupId>com.sun.activation</groupId>
  <artifactId>javax.activation</artifactId>
  <version>1.2.0</version>
</dependency>

```

(Was not needed before because **java.xml.bind** requires it)

Modularization

Preparation

- set compiler source and target to Java 11
- remove version-specific profiles

Get an Overview

1. see artifact dependencies

```
jdeps11 -s -R -cp 'app/*' app/main.jar
```

2. limiting to *Monitor* classes

```
jdeps11 -include 'monitor.*' -s -R -cp 'app/*' app/main.jar
```

3. create a diagram

```
jdeps11 -include 'monitor.*' -s -R --dot-output . -cp 'monitor/target/libs/*'  
monitor/target/main.jar  
dot -Tpng -O summary.dot  
gwenview summary.dot.png
```

4. clean up graph

```
sed -i '/java.base/d' summary.dot  
sed -i 's/.jar//g' summary.dot  
dot -Tpng -O summary.dot  
gwenview summary.dot.png
```

Start Bottom-Up

1. start with *monitor.utils*
 - a. modularize *monitor.utils*

```
module monitor.utils {  
    exports monitor.utils;  
}
```

- b. build with Maven
- c. observe that *monitor.utils* is module:


```
jar11 --describe-module --file monitor/target/libs/utils.jar
```

- d. observe that the module works on the class path:

```
java11 -cp 'app/*' monitor.Main
```

- e. make it work on module path

- i. add `mv mv app/utils.jar mods` to `move-modules.sh`
- ii. try to run

```
./move-modules.sh
# fails
java11 -cp 'app/*' --module-path mods monitor.Main
# too noisy
java11 -cp 'app/*' --module-path mods \
    --show-module-resolution monitor.Main
# no utils
java11 -cp 'app/*' --module-path mods \
    --show-module-resolution monitor.Main
| grep utils
# yes utils!
java11 -cp 'app/*' --module-path mods \
    --add-modules monitor.utils \
    --show-module-resolution monitor.Main
| grep utils
# launch
java11 -cp 'app/*' --module-path mods \
    --add-modules monitor.utils \
    monitor.Main
```

2. continue with observers

- a. modularize *monitor.observer*:

```
module monitor.observer {
    exports monitor.observer;
}
```

- b. modularize *monitor.observer.alpha*:

```
module monitor.observer.alpha {
    requires monitor.observer;
    exports monitor.observer.alpha;
}
```

c. modularize *monitor.observer.beta*:

```
module monitor.observer.beta {  
    requires monitor.observer;  
    exports monitor.observer.beta;  
}
```

d. in *monitor.observer.alpha*, show with `mvn -X compiler` class/module path

e. add `mv app/observer* mods` to `move-modules.sh`

f. run

```
java11 -cp 'app/*' --module-path mods \  
    --add-modules monitor.utils,monitor.observer.alpha,monitor.observer.beta \  
    monitor.Main
```

3. modularize statistics as preparation for *monitor.rest*

a. modularize *monitor.statistics*

```
module monitor.statistics {  
    requires monitor.observer;  
    exports monitor.statistics;  
}
```

b. add `mv app/statistics.jar mods` to `move-modules.sh`

c. run

```
java11 -cp 'app/*' --module-path mods \  
    --add-modules  
    monitor.utils,monitor.observer.alpha,monitor.observer.beta,monitor.statistics \  
    monitor.Main
```

4. modularize *monitor.rest* in face of unmodularized dependencies

a. create initial module descriptor

```
module monitor.rest {  
    requires java.xml.bind;  
  
    requires monitor.utils;  
    requires monitor.statistics;  
  
    exports monitor.rest;  
}
```

- b. use `jar11 --describe-module-file --file spark-core` etc to determine module names
- c. add `requires` for `spark.core`, `jackson.core`, `jackson.databind` to module descriptor
- d. identify split package between `jsr305` and `java.annotation`
- e. add `requires java.annotation`
- f. patch package split:

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <compilerArgs>
          <arg>--patch-
module=java.annotation=${settings.localRepository}/com/google/code/findbugs/jsr3
05/3.0.2/jsr305-3.0.2.jar</arg>
        </compilerArgs>
      </configuration>
    </plugin>
  </plugins>
</build>
```

- g. add to `move-modules.sh`:

```
mv app/jaxb-api.jar mods
mv app/javax.activation.jar mods
mv app/javax.annotation-api.jar mods
mv app/spark-core.jar mods
mv app/jackson-core.jar mods
mv app/jackson-databind.jar mods
```

- h. run

```
java11 -cp 'app/*' --module-path mods \
  --add-modules monitor.rest,monitor.observer.alpha,monitor.observer.beta \
  monitor.Main
```

- i. mention that `--patch-module` is not needed because annotations are not evaluated at run time
- j. observe run-time error in `watch` tab
- k. add `opens monitor.rest to java.xml.bind;` to `monitor.rest`