Informatics Large Practical: Advanced Maven

Stephen Gilmore and Paul Jackson

School of Informatics

Monday, 28th September 2020

Using Maven as a build tool

In this lecture, we use Maven for this purpose:

- compiling our Java code, and packaging it in a Java Archive file (JAR) for our project, with our code bundled together with the libraries which it depends on.

Why do we have to package our code?

- You might ask, *"My code runs fine in Eclipse, why do I have to make a JAR file for it?"*
- The answer is that we want to be able to run our code without a "human-in-the-loop" to click the Run button.
- We want to run our applications on a server, or in a script which runs every hour, without human intervention.
- Your code should run in Eclipse, but it must also be able to run outside Eclipse, even in contexts where Eclipse is not available (e.g. in the runtime system for an autonomous drone).

Automating the build process

- In addition to being able to automate the running of our code, we also want to automate the build process.
- This means that we create a build file which has precise instructions on how to compile and package our project.
  For the Maven build tool, that file is pom.xml.
- Because it is contained in a file, this information can be saved with our source code, in the project source code repository.

The concept of deployment

- The larger idea here is the idea of deployment. That is, our code is compiled and tested on one machine during development but it is deployed and run on a different machine.
- A necessary step between development and deployment is packaging of the code and its dependencies. The goal here is to ensure that our code will run on the deployment machine.
- Details of the packaging are in the project's pom.xml file.
  - *For the purposes of this project, we regard pom.xml as source code, so please don't post your pom.xml file to Piazza.*

Maven: Specifying the packaging for the artefact

XML

```xml
<groupId>uk.ac.ed.inf</groupId>
<artifactId>heatmap</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
```

- This will create target/heatmap-0.0.1-SNAPSHOT.jar, but we still need to specify what is included in the JAR file.
- This file can be renamed afterward, but your pom.xml file should create a JAR file with that precise name.

The Maven build lifecycle and its phases

- validate: check that necessary project information is available.
- compile: compile the Java source code of the project.
- test: test the compiled source code using unit testing.
- package: package the compiled code in a format such as JAR.

- verify, install, deploy: not phases that we are involved with.

- clean: remove all files generated by the previous build.

Maven: The build process uses plugins

XML

```
<build>
  <plugins>
      The Maven compiler plugin
      The Maven JAR plugin
      The Maven shade plugin
  </plugins>
</build>
```

- Note: Adding these plugins will cause JAR files to be downloaded.
  - You will need a working internet connection for this and if you are behind a firewall you may need to use a VPN here.

Note: We do not use "plugin management"

XML

```xml
<build>
    <pluginManagement> <!-- delete this start tag -->
        <plugins>
            ...
        </plugins>
    </pluginManagement> <!-- delete this end tag -->
</build>
```

- Note: our solution does not use Maven's "plugin management" feature so if your pom.xml file contains ⟨pluginManagement⟩ tags then you should delete them.

Maven plugins #1: The Maven Compiler plugin

XML

```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
      <release>11</release>
      <showWarnings>true</showWarnings>
      <compilerArgs>
            <arg>-Xlint:all,-classfile</arg>
      </compilerArgs>
    </configuration>
</plugin>
```

- Here we specify that we are using Java version 11 and ask for improved error messages (using lint).

Use of lint

- The lint tool can be used to catch programming errors which are not caught by the Java compiler.
- Visit `https://docs.oracle.com/en/java/javase/14/docs/specs/man/javac.html#extra-options` for a list of the kinds of issues caught by lint.
- For examples of many of the issues, visit `https://docs.oracle.com/en/java/javase/14/docs/specs/man/javac.html#examples-of-using--xlint-keys`

Maven plugins #2: The Maven JAR plugin

XML

```xml
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.2.0</version>
  <configuration>
    <archive>
      <index>true</index>
      <manifest>
        <mainClass>uk.ac.ed.inf.heatmap.App</mainClass>
      </manifest>
    </archive>
  </configuration>
</plugin>
```

- Here we specify the main class of our application.

Maven plugins #3: The Maven Shade plugin

XML

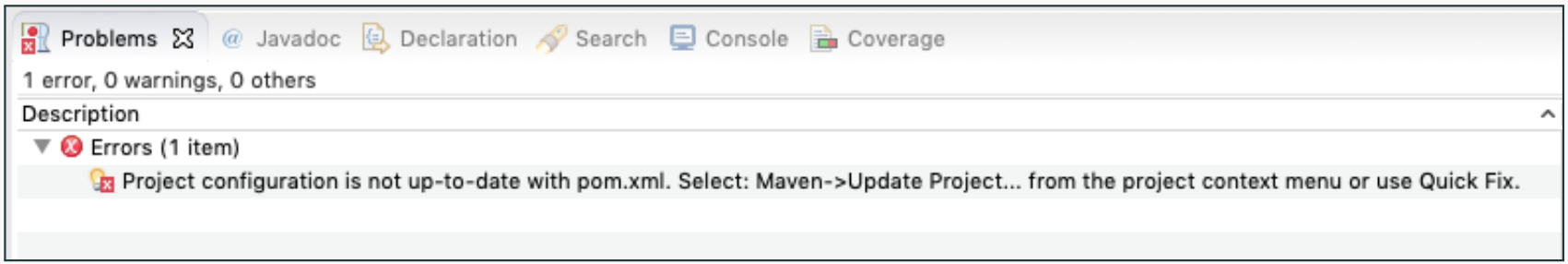
```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
    <version>3.2.4</version>

    <executions>
      <execution>
        <phase>package</phase>
        <goals>
          <goal>shade</goal>
        </goals>
      </execution>
    </executions>
</plugin>
```

The “package” phase

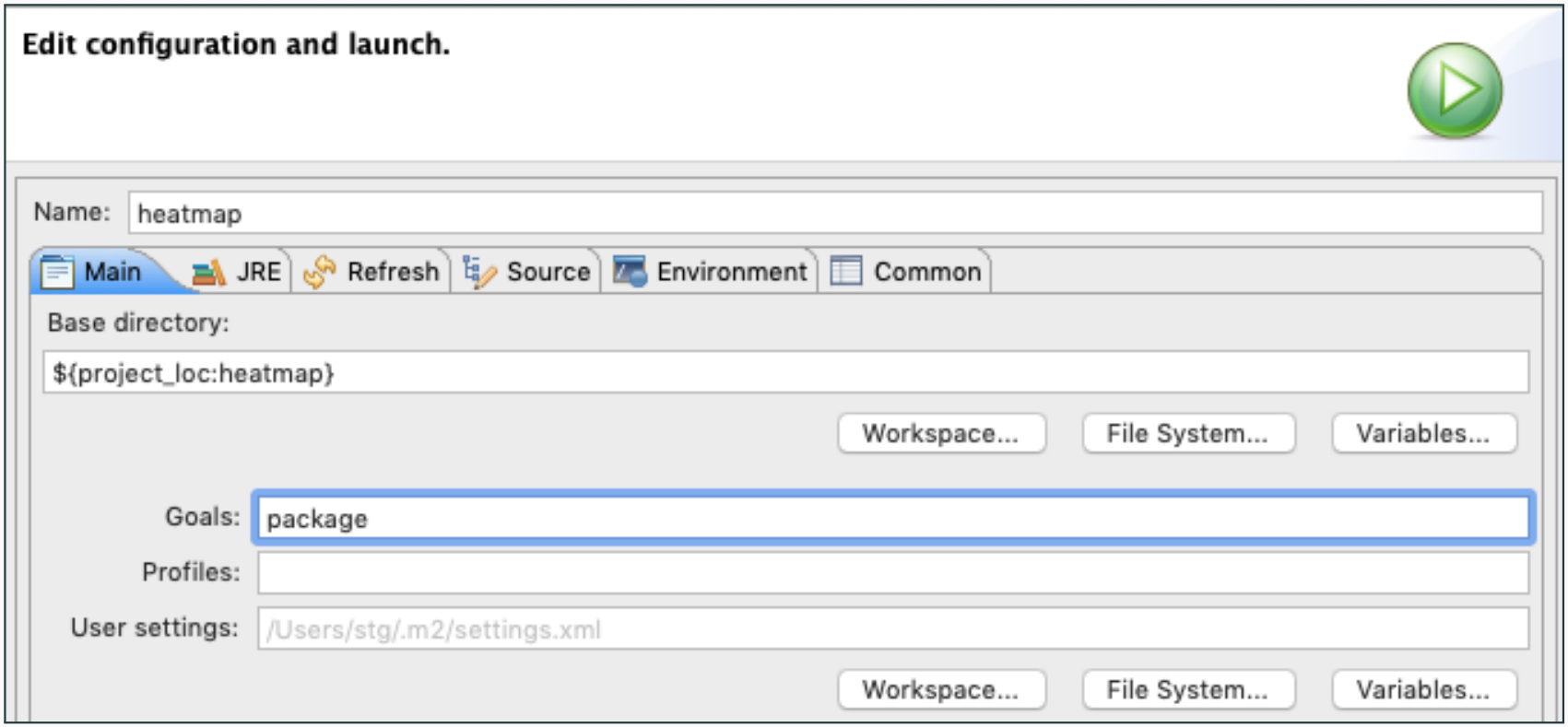
- As can be seen, shading happens in the package phase of the Maven build lifecycle. The term “shade” means largely “rename” or “hide”.
- The Maven Shade plugin takes several JAR files (such as the ones for GeoJSON and Gson) and our own code and creates an *“über JAR”* to contain all the compiled code and any attached resources.

Maven problems



- Changes to pom.xml may cause you to have to update your project, because the rules for building it have changed.
- Changes to the Maven compiler plugin version or settings will necessitate a Maven → Update Project ... rebuild.
- When you have completed your pom.xml file, you can choose Run as → Maven build instead of Run as → Java application.

"Maven build" needs a goal



- For Run as → Maven build to succeed, we must specify a goal. Our goal is package (to create a JAR file).
- It is normal for the package phase to produce warnings; these can be ignored.

Running a JAR file

Check that you can run your JAR file in the target folder with:

```
java -jar heatmap-0.0.1-SNAPSHOT.jar ../predictions.txt
```

(Assuming that your predictions.txt file is in the same folder as pom.xml.)

Thank you for listening.