BASICS OF MAVEN

BUILD TOOL

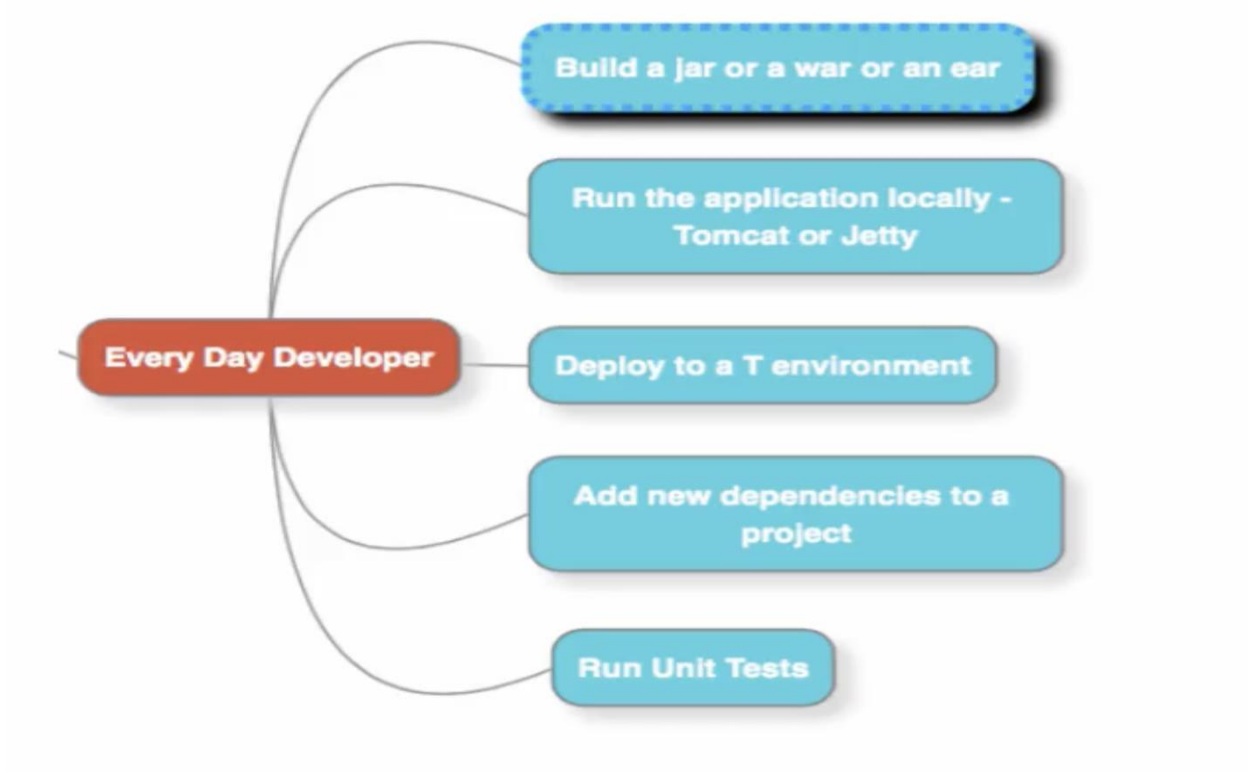
Presented By:

Devinder Singh Saggu

MAVEN VS. ANT

	Maven	**Ant**
1.	Maven is project management tool. Like black box where user don't see the complexity behind build tool.	Mainly build tool works on **build.xml** which mentions what and how .
2.	Reusable plugins.	Scripts are not reusable.
3.	Convention Over Configuration.	Not formal conventional.
4.	Declarative means POM.xml. Dependency Management (including transitive)	Procedural. What to do and when to do. Like ordering of scripts.
5.	Project Lifecycle	Do not have project life cycle.
6.	It is framework.	It is tool box.

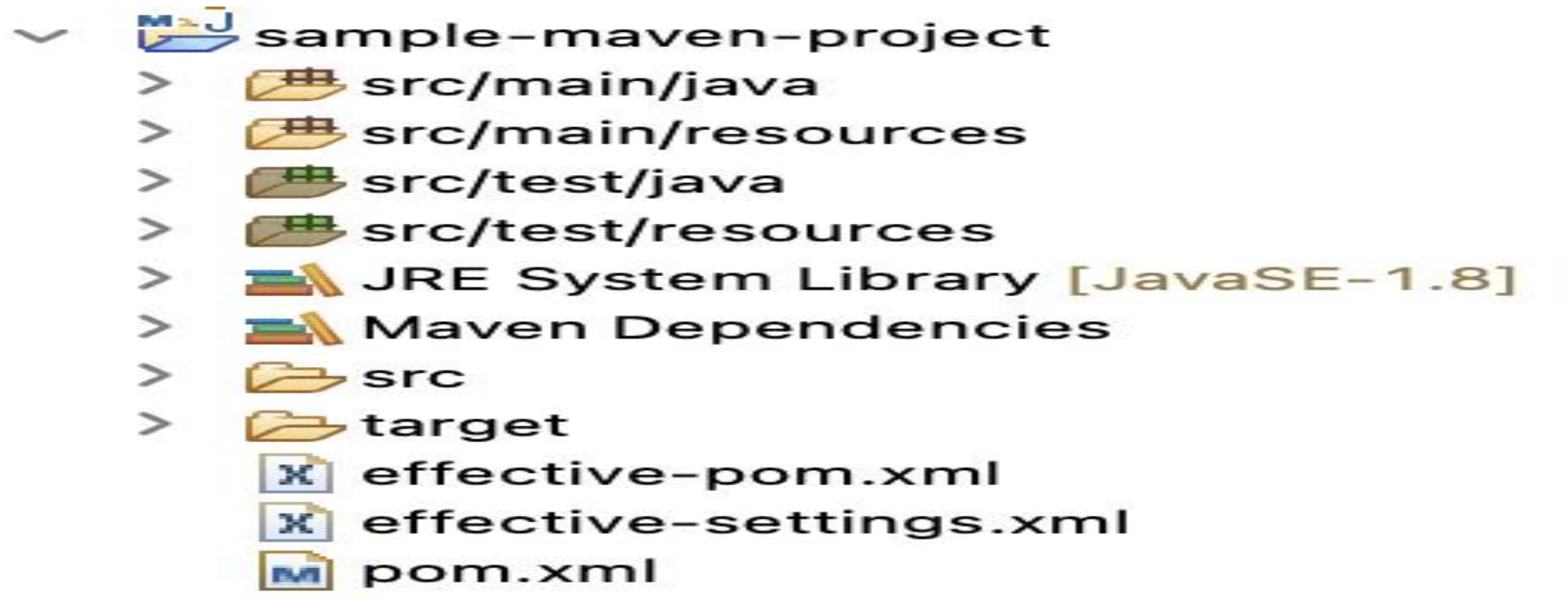
DEVELOPER TASKS


Every Day Developer
Build a jar or a war or an ear
Run the application locally - Tomcat or Jetty
Deploy to a T environment
Add new dependencies to a project
Run Unit Tests

CONFIGURE MAVEN:

- Configure on Windows:
  - https://mkyong.com/maven/how-to-install-maven-in-windows/
- Configure on Mac:
  - https://mkyong.com/maven/install-maven-on-mac-osx/
- Configure on Linux:
  - https://www.journaldev.com/33588/install-maven-linux-ubuntu

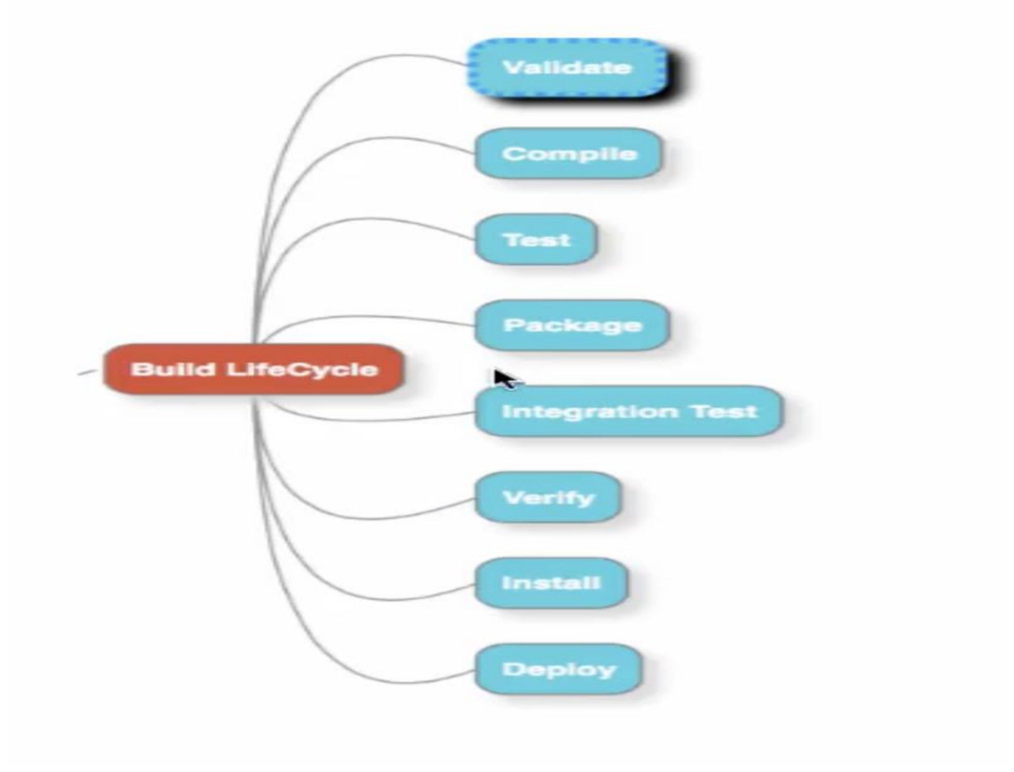
BASIC MAVEN PROJECT STRUCTURE:



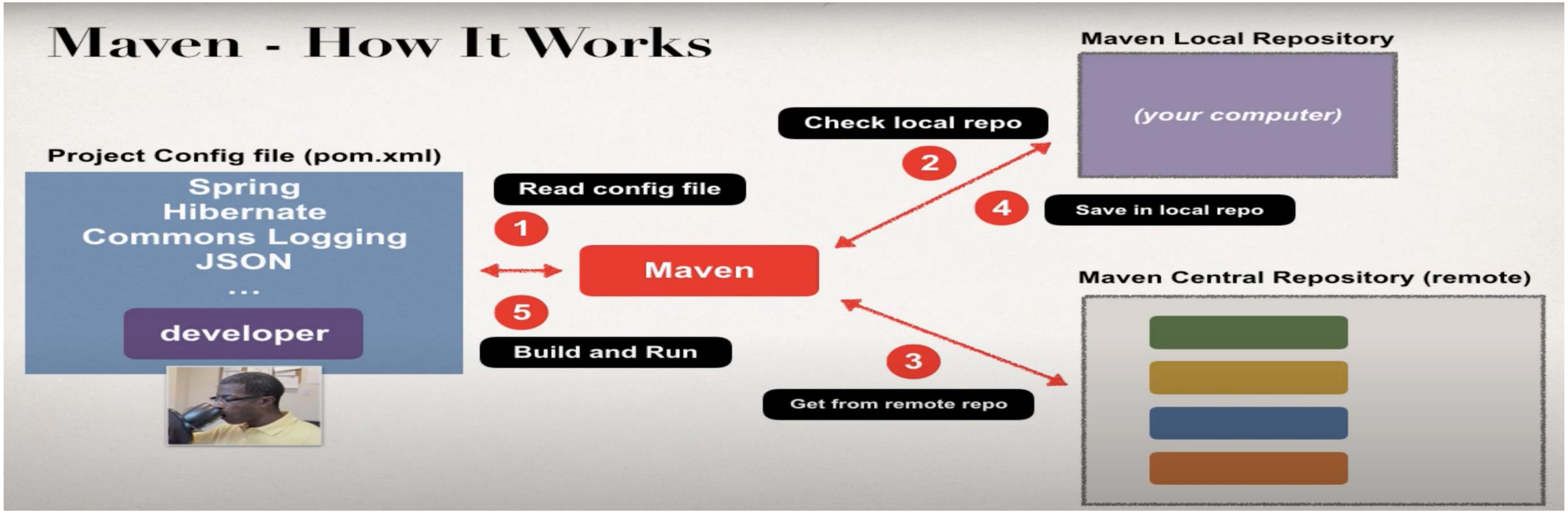
```
sample-maven-project
  src/main/java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  effective-pom.xml
  effective-settings.xml
  pom.xml
```

MAVEN BUILD LIFECYCLE:

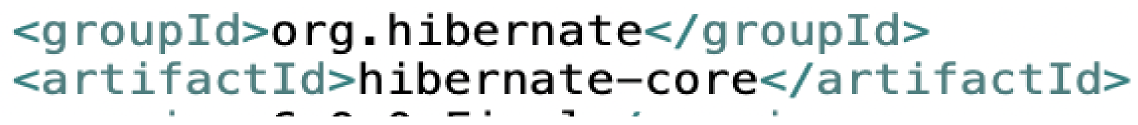
- It defines the steps maven follows when user execute command.
- On running the specific maven command the earlier steps in build life-cycle executes automatically.
- In Ant, we have to define these steps individually.



BACKGROUND WORK BY MAVEN:

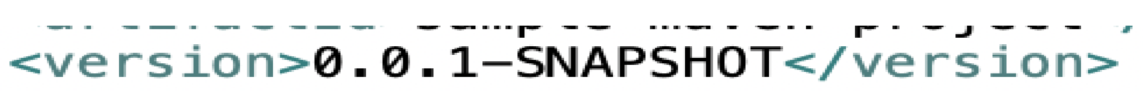


UNDERSTANDING MAVEN POM



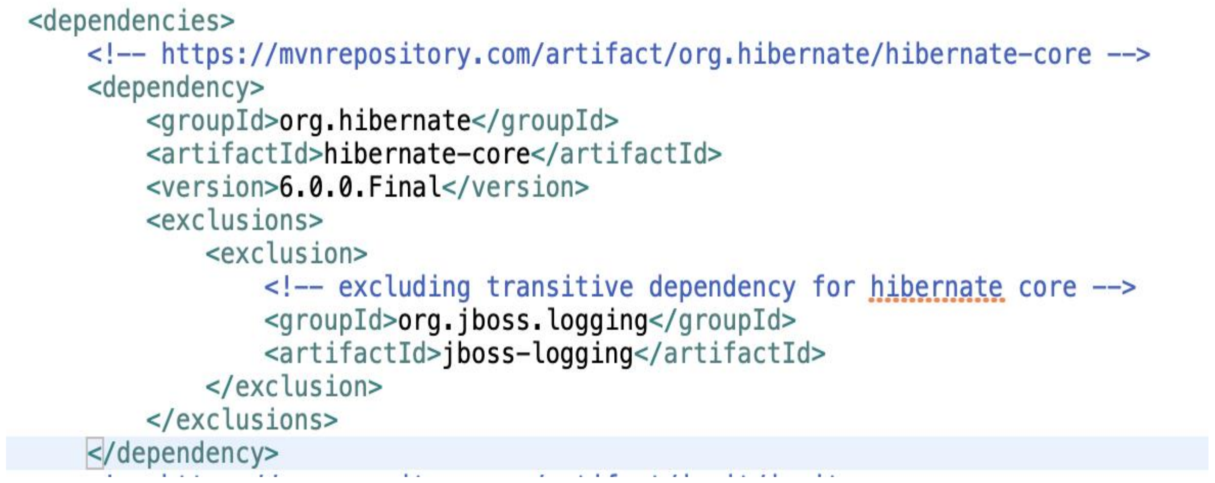
```
<groupId>org.hibernate</groupId>
<artifactId>hibernate-core</artifactId>
```

Uniquely identifies the project.



```
<version>0.0.1-SNAPSHOT</version>
```

Snapshot means underdeveloped.



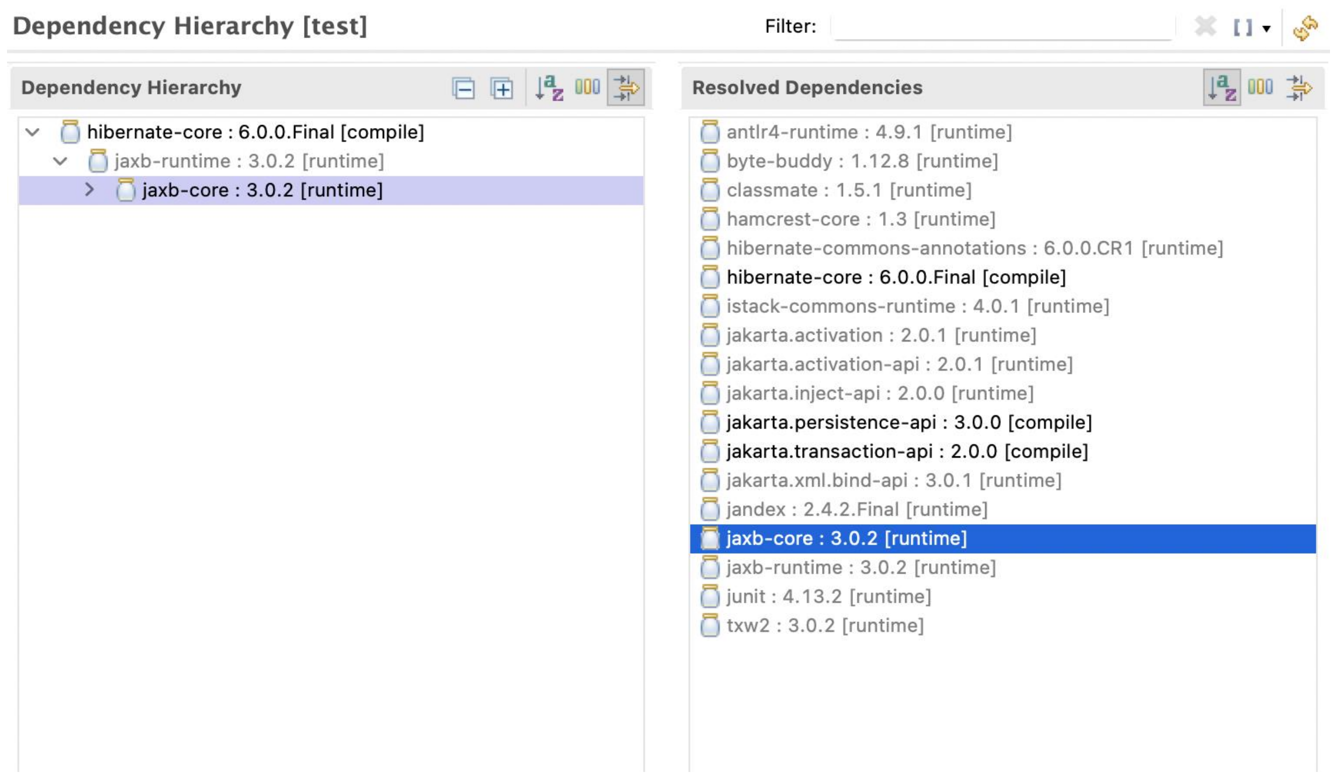
```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>6.0.0.Final</version>
        <exclusions>
            <exclusion>
                <!-- excluding transitive dependency for hibernate core -->
                <groupId>org.jboss.logging</groupId>
                <artifactId>jboss-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
```

Resolves dependencies from maven repository.

Also downloads the transitive dependencies.

Exclude the unnecessary dependencies.

TRANSITIVE DEPENDENCIES:


Dependency Hierarchy [test]

Dependency Hierarchy

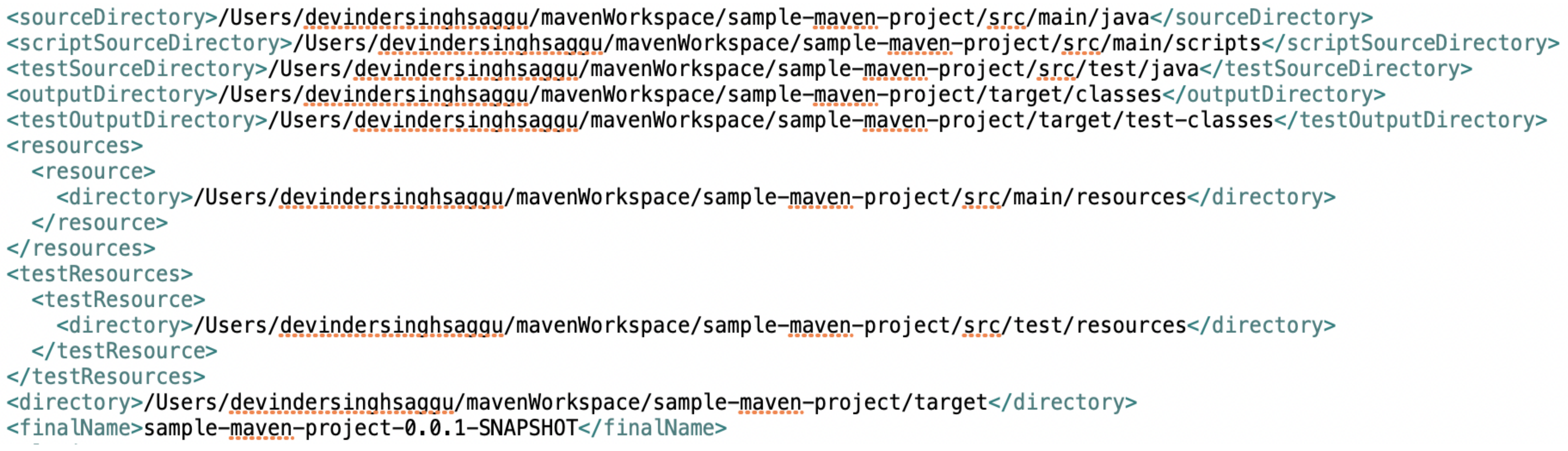
```
hibernate-core : 6.0.0.Final [compile]
  jaxb-runtime : 3.0.2 [runtime]
    jaxb-core : 3.0.2 [runtime]
```

Resolved Dependencies

```
antlr4-runtime : 4.9.1 [runtime]
byte-buddy : 1.12.8 [runtime]
classmate : 1.5.1 [runtime]
hamcrest-core : 1.3 [runtime]
hibernate-commons-annotations : 6.0.0.CR1 [runtime]
hibernate-core : 6.0.0.Final [compile]
istack-commons-runtime : 4.0.1 [runtime]
jakarta.activation : 2.0.1 [runtime]
jakarta.activation-api : 2.0.1 [runtime]
jakarta.inject-api : 2.0.0 [runtime]
jakarta.persistence-api : 3.0.0 [compile]
jakarta.transaction-api : 2.0.0 [compile]
jakarta.xml.bind-api : 3.0.1 [runtime]
jandex : 2.4.2.Final [runtime]
jaxb-core : 3.0.2 [runtime]
jaxb-runtime : 3.0.2 [runtime]
junit : 4.13.2 [runtime]
txw2 : 3.0.2 [runtime]
```

EFFECTIVE POM:

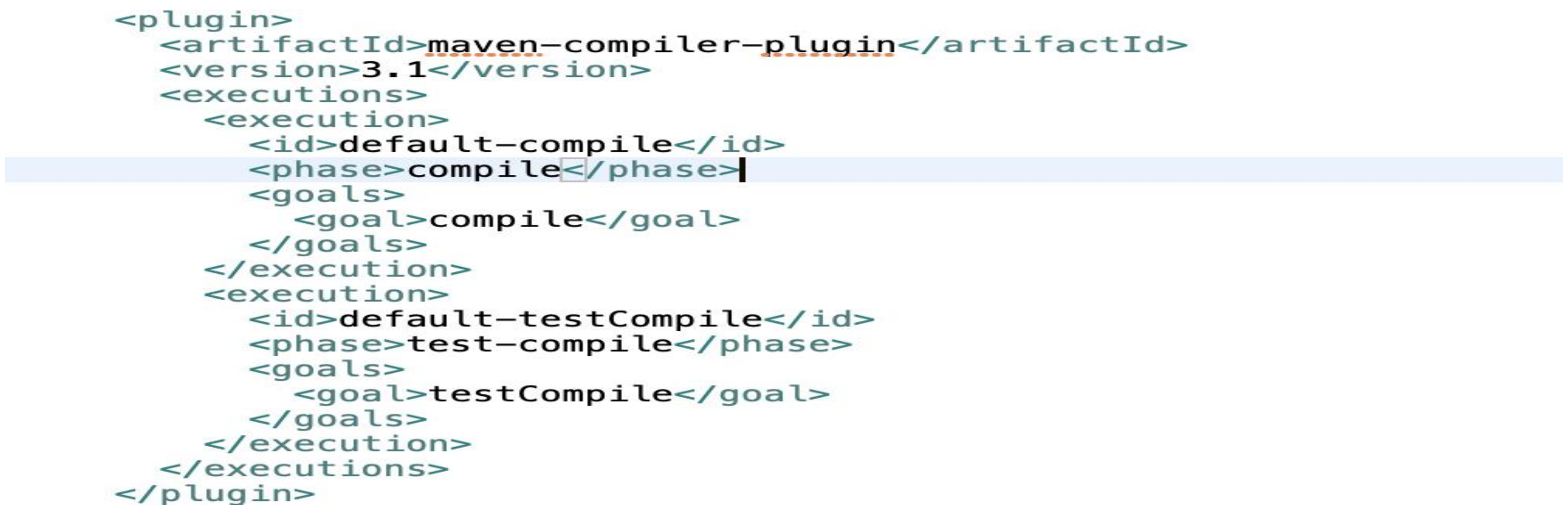
- Effective POM = POM + Super POM
  - Contains default conventional directories path.



```
<sourceDirectory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/src/main/java</sourceDirectory>
<scriptSourceDirectory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/src/main/scripts</scriptSourceDirectory>
<testSourceDirectory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/src/test/java</testSourceDirectory>
<outputDirectory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/target/classes</outputDirectory>
<testOutputDirectory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/target/test-classes</testOutputDirectory>
<resources>
  <resource>
    <directory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/src/main/resources</directory>
  </resource>
</resources>
<testResources>
  <testResource>
    <directory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/src/test/resources</directory>
  </testResource>
</testResources>
<directory>/Users/devindersinghsaggu/mavenWorkspace/sample-maven-project/target</directory>
<finalName>sample-maven-project-0.0.1-SNAPSHOT</finalName>
```

CONT…

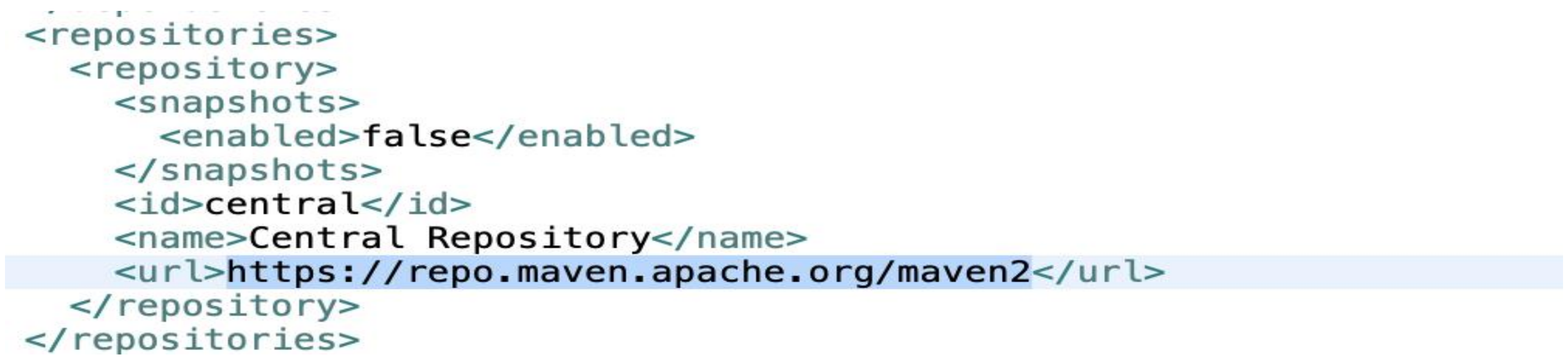
- Default plugins – Compiler, surefire (JUnit)



```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.1</version>
  <executions>
    <execution>
      <id>default-compile</id>
      <phase>compile</phase>
      <goals>
        <goal>compile</goal>
      </goals>
    </execution>
    <execution>
      <id>default-testCompile</id>
      <phase>test-compile</phase>
      <goals>
        <goal>testCompile</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

CONT…

- Central Repository

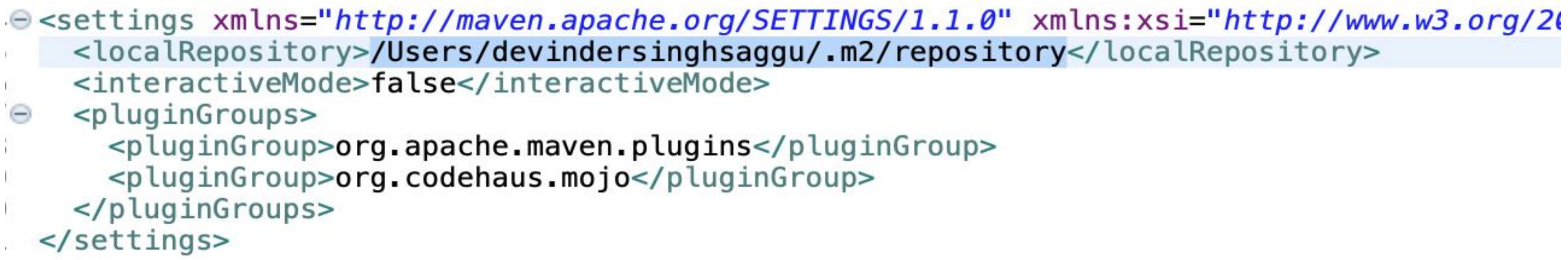


```
<repositories>
  <repository>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
    <id>central</id>
    <name>Central Repository</name>
    <url>https://repo.maven.apache.org/maven2</url>
  </repository>
</repositories>
```

- Central Maven Repository

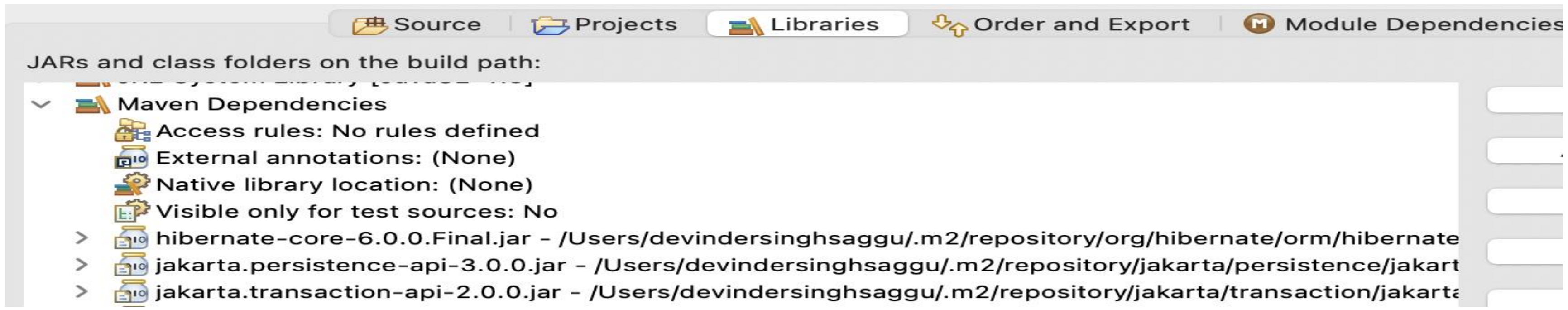
LOCAL REPOSITORY:

➢ Path for local repository can be found in **effective-settings.xml** which acts as **cache**.



```
<settings xmlns="http://maven.apache.org/SETTINGS/1.1.0" xmlns:xsi="http://www.w3.org/2
  <localRepository>/Users/devindersinghsaggu/.m2/repository</localRepository>
  <interactiveMode>false</interactiveMode>
  <pluginGroups>
    <pluginGroup>org.apache.maven.plugins</pluginGroup>
    <pluginGroup>org.codehaus.mojo</pluginGroup>
  </pluginGroups>
</settings>
```

➢ Maven dependencies in build path.



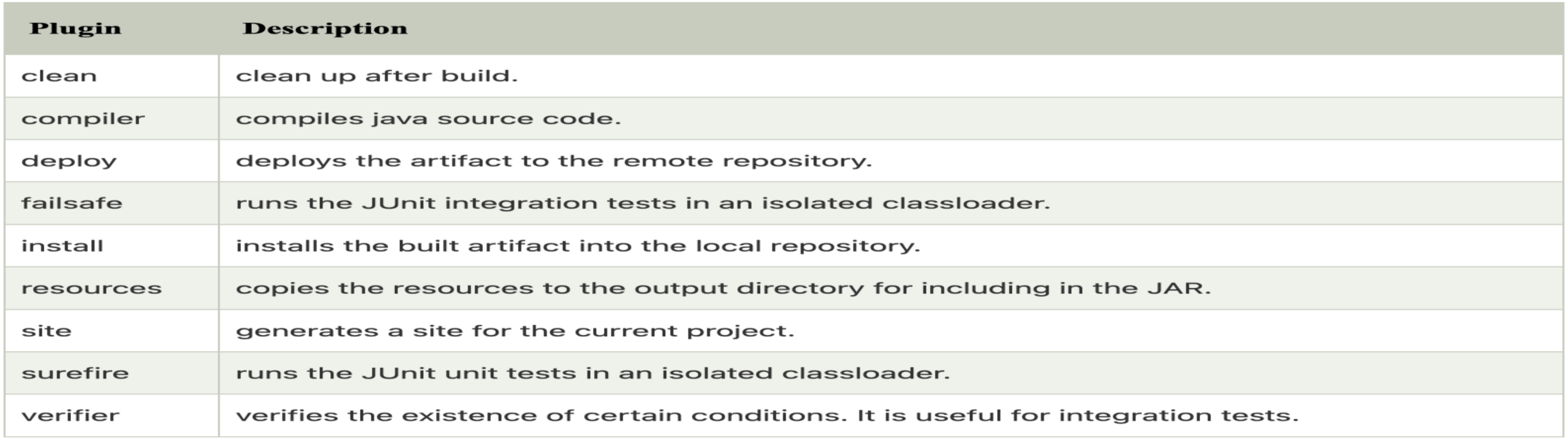
JARs and class folders on the build path:

```
Maven Dependencies
    Access rules: No rules defined
    External annotations: (None)
    Native library location: (None)
    Visible only for test sources: No
  > hibernate-core-6.0.0.Final.jar - /Users/devindersinghsaggu/.m2/repository/org/hibernate/orm/hibernate
  > jakarta.persistence-api-3.0.0.jar - /Users/devindersinghsaggu/.m2/repository/jakarta/persistence/jakart
  > jakarta.transaction-api-2.0.0.jar - /Users/devindersinghsaggu/.m2/repository/jakarta/transaction/jakarta
```

MAVEN PLUGINS

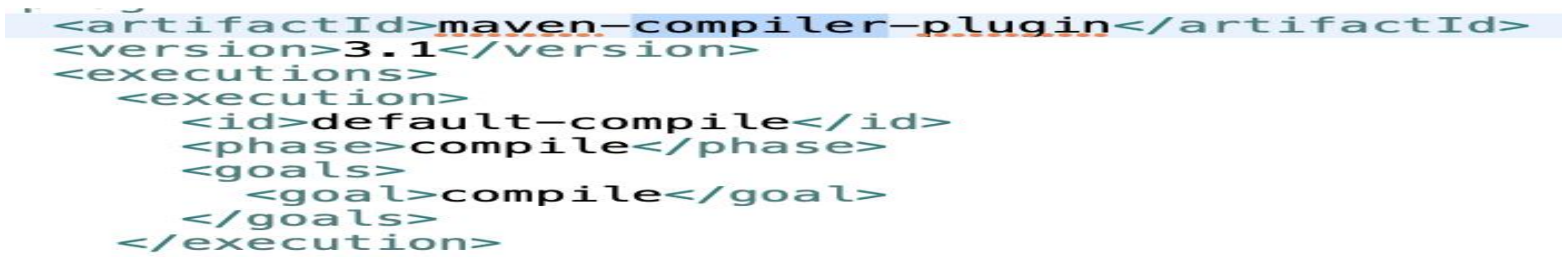
The **maven plugins** are central part of maven framework, it is used to perform specific goal.

1. **Build plugins** – executed at the time of build. These plugins should be declared inside the **<build>** element.
2. **Reporting plugins** – executed at the time of site generation. Should be declared inside the **<reporting>** element.



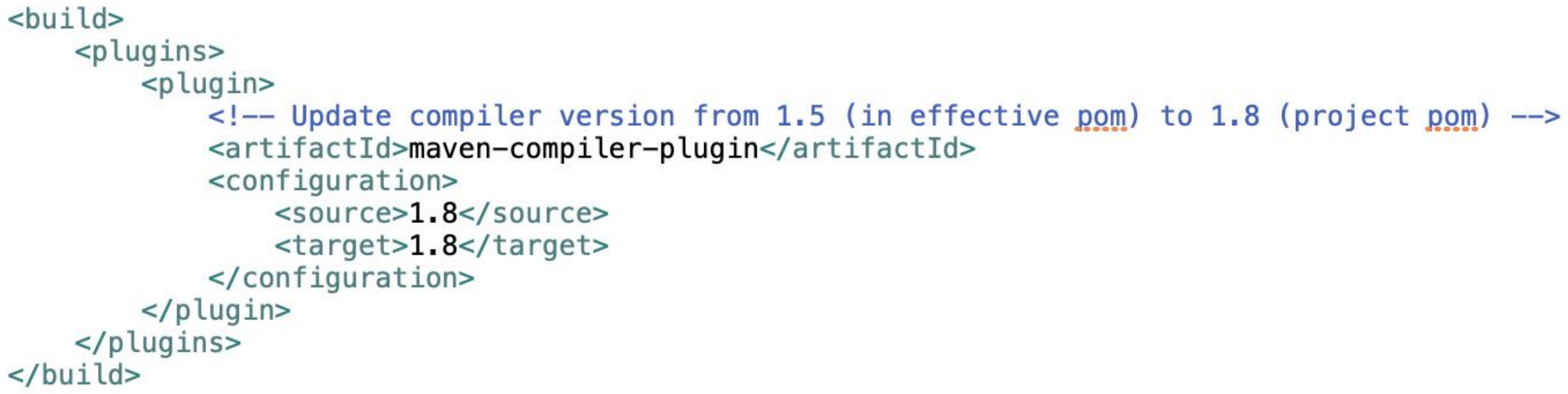
Plugin	Description
clean	clean up after build.
compiler	compiles java source code.
deploy	deploys the artifact to the remote repository.
failsafe	runs the JUnit integration tests in an isolated classloader.
install	installs the built artifact into the local repository.
resources	copies the resources to the output directory for including in the JAR.
site	generates a site for the current project.
surefire	runs the JUnit unit tests in an isolated classloader.
verifier	verifies the existence of certain conditions. It is useful for integration tests.

COMPILER PLUGIN:



```
<artifactId>maven-compiler-plugin</artifactId>
<version>3.1</version>
<executions>
  <execution>
    <id>default-compile</id>
    <phase>compile</phase>
    <goals>
      <goal>compile</goal>
    </goals>
  </execution>
```

→ Effective POM



```
<build>
    <plugins>
        <plugin>
            <!-- Update compiler version from 1.5 (in effective pom) to 1.8 (project pom) -->
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

→ Project POM

DEPENDENCY SCOPE:

- **compile** – Default scope and are propagated to dependent projects.
- **runtime** – required for execution only. Available in test and runtime classpath not in compile time classpath.
- **test –** only available for test compilation and execution phases (JUnit or Mockito).
- **import -** only supported on a dependency of type pom. It indicates the dependency is to be replaced with the effective list of dependencies. Since they are replaced, dependencies with a scope of import do not actually participate in limiting the transitivity of a dependency.
- **provided -** JDK or a container to provide the dependency at runtime. For instance, dependency on the Servlet API and related Java EE APIs because web container provides
  those classes.

MULTI MODULE PROJECT:

- Packaging should be POM. And packaging POM acts as parent POM for modules.
- Here, business and data are modules with parent project multi-module-maven-project.
- Data layer has dependencies for Spring and Hibernate.
- Business layer has dependency for Spring framework.
- Dependency management in parent POM mentions the versions of dependencies to be included in child POMs.
- Include data. layer as dependency in business layer with built-in variable for version.
- For version management of data module, mention it in parent POM.

MAVEN COMMANDS:

1. Version range for dependencies
2. help:effective-pom –Doutput=effective-pom.xml
3. mvn help:effective-settings –Doutput=effective-settings.xml
4. mvn clean install –DskipTests
5. mvn –X clean install >> sample.txt
6. mvn dependency:tree
7. mvn dependency:sources
8. mvn --help

REFERENCES:

1) https://www.youtube.com/watch?v=I9MD-IGQDoo (ANT)
2) https://maven.apache.org/what-is-maven.html (Maven)
3) https://www.youtube.com/watch?v=m_eWc9pjyg4 (How it works)
4) https://stackoverflow.com/questions/33935281/command-not-found-oh-my-zsh
5) https://www.youtube.com/watch?v=oxToe-4c6OM
6) https://repo.maven.apache.org/maven2/org/apache/maven/plugins/ (Maven Plugins)