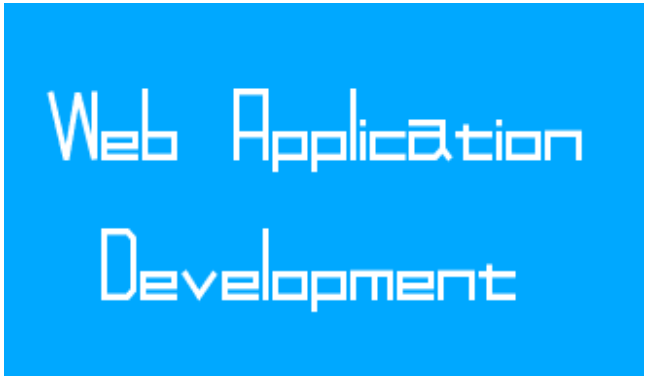
互联网应用开发技术

Web Application Development

第7课
WEB后端-MAVEN

Episode Seven

Maven



陈昊鹏

chen-hp@sjtu.edu.cn

Overview

- Maven
- Maven in IntelliJ IDEA

What is Maven?

- Maven,
  - a Yiddish word meaning *accumulator of knowledge*,
  - began as an attempt to simplify the build processes in the Jakarta Turbine project.
  - We wanted a standard way to build the projects, a clear definition of what the project consisted of, an easy way to publish project information, and a way to share JARs across several projects.
  - The result is a tool that can now be used for building and managing any Java-based project. We hope that we have created something that will make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

Maven's Objectives

- Maven's primary goal
  - is to allow a developer to comprehend the complete state of a development effort in the shortest period of time.

- In order to attain this goal, Maven deals with several areas of concern:
  - Making the build process easy
  - Providing a uniform build system
  - Providing quality project information
  - Encouraging better development practices

Maven's Objectives

- **Making the build process easy**
  - While using Maven doesn't eliminate the need to know about the underlying mechanisms, Maven does shield developers from many details.

- **Providing a uniform build system**
  - Maven builds a project using its project object model (POM) and a set of plugins.
  - Once you familiarize yourself with one Maven project, you know how all Maven projects build.
  - This saves time when navigating many projects.

Maven's Objectives

- **Providing quality project information**
  - Maven provides useful project information that is in part taken from your POM and in part generated from your project's sources. For example, Maven can provide:
    - Change log created directly from source control
    - Cross referenced sources
    - Mailing lists managed by the project
    - Dependencies used by the project
    - Unit test reports including coverage
    - Third party code analysis products also provide Maven plugins that add their reports to the standard information given by Maven.

Maven's Objectives

- **Providing guidelines for best practices development**
  - Maven aims to gather current principles for best practices development and make it easy to guide a project in that direction.
  - For example, specification, execution, and reporting of unit tests are part of the normal build cycle using Maven. Current unit testing best practices were used as guidelines:
    - Keeping test source code in a separate, but parallel source tree
    - Using test case naming conventions to locate and execute tests
    - Having test cases setup their environment instead of customizing the build for test preparation
  - Maven also assists in project workflow such as release and issue management.
  - Maven also suggests some guidelines on how to layout your project's directory structure.

The first Maven project

- To create our first Maven project we are going to use Maven's archetype mechanism.
  - An archetype is defined as *an original pattern or model from which all other things of the same kind are made*. In Maven, an archetype is a template of a project which is combined with some user input to produce a working Maven project that has been tailored to the user's requirements.

- On to creating your first project! In order to create the simplest of Maven projects, execute the following from the command line:

```
mvn -B archetype:generate -DgroupId=com.mycompany.app
    -DartifactId=my-app
    -DarchetypeArtifactId=maven-archetype-quickstart
    -DarchetypeVersion=1.4
```

The first Maven project

- You will notice that a directory named my-app has been created for the new project,
  - and this directory contains a file named pom.xml:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>my-app</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
      ... lots of helpful plugins
    </pluginManagement>
  </build>
</project>
```

The first Maven project

- pom.xml contains the Project Object Model (POM) for this project:
  - **project** This is the top-level element in all Maven pom.xml files.
  - **modelVersion** This element indicates what version of the object model this POM is using.
  - **groupId** This element indicates the unique identifier of the organization or group that created the project.
  - **artifactId** This element indicates the unique base name of the primary artifact being generated by this project.
  - **version** This element indicates the version of the artifact generated by the project.
  - **name** This element indicates the display name used for the project. This is often used in Maven's generated documentation.
  - **url** This element indicates where the project's site can be found.
  - **properties** This element contains value placeholders accessible anywhere within a POM.
  - **dependencies** This element's children list dependencies. The cornerstone of the POM.
  - **build** This element handles things like declaring your project's directory structure and managing plugins.

The first Maven project

- After the archetype generation of your first project you will also notice that the following directory structure has been created:

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

Compile Maven project

mvn compile

```
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.mycompany.app:my-app >----------------------
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory <dir>/my-app/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ my-app ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to <dir>/my-app/target/classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.899 s
[INFO] Finished at: 2020-07-12T11:31:54+01:00
[INFO] ------------------------------------------------------------------------
```

Compile Test Source & Run Test


```
mvn test
```

```
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.mycompany.app:my-app >----------------------
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory <dir>/my-app/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory <dir>/my-app/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ my-app ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to <dir>/my-app/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ my-app ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.mycompany.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.025 s - in com.mycompany.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.881 s
[INFO] Finished at: 2020-07-12T12:00:33+01:00
[INFO] ------------------------------------------------------------------------
```

Create & Install a Jar(War)

- Making a JAR file is straight forward enough and can be accomplished by executing the following command:

  ```
  mvn package
  ```

  – You can now take a look in the ${basedir}/target directory and you will see the generated JAR file.

- Now you'll want to install the artifact you've generated (the JAR file) in your local repository (${user.home}/.m2/repository is the default location). To do so execute the following command:

  ```
  mvn install
  ```

- There are plenty of other standalone goals that can be executed as well, for example:

  ```
  mvn clean
  ```

  – This will remove the target directory with all the build data before starting so that it is fresh

Use Plugins


```
...
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.3</version>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
      </configuration>
    </plugin>
  </plugins>
</build>
...
```

Add resources to JAR

- The simple rule employed by Maven is this:
  - any directories or files placed within the ${basedir}/src/main/resources directory are packaged in your JAR with the exact same structure starting at the base of the JAR.

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   |-- java
    |   |   `-- com
    |   |       `-- mycompany
    |   |           `-- app
    |   |               `-- App.java
    |   `-- resources
    |       `-- META-INF
    |           `-- application.properties
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

Add resources to JAR

- If you unpacked the JAR that Maven created for you and took a look at it you would see the following:

```
|-- META-INF
|   |-- MANIFEST.MF
|   |-- application.properties
|   `-- maven
|       `-- com.mycompany.app
|           `-- my-app
|               |-- pom.properties
|               `-- pom.xml
`-- com
    `-- mycompany
        `-- app
            `-- App.class
```

Filter resource files

- To have Maven filter resources when copying,
  - simply set filtering to true for the resource directory in your pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <filtering>true</filtering>
      </resource>
    </resources>
  </build>
</project>
```

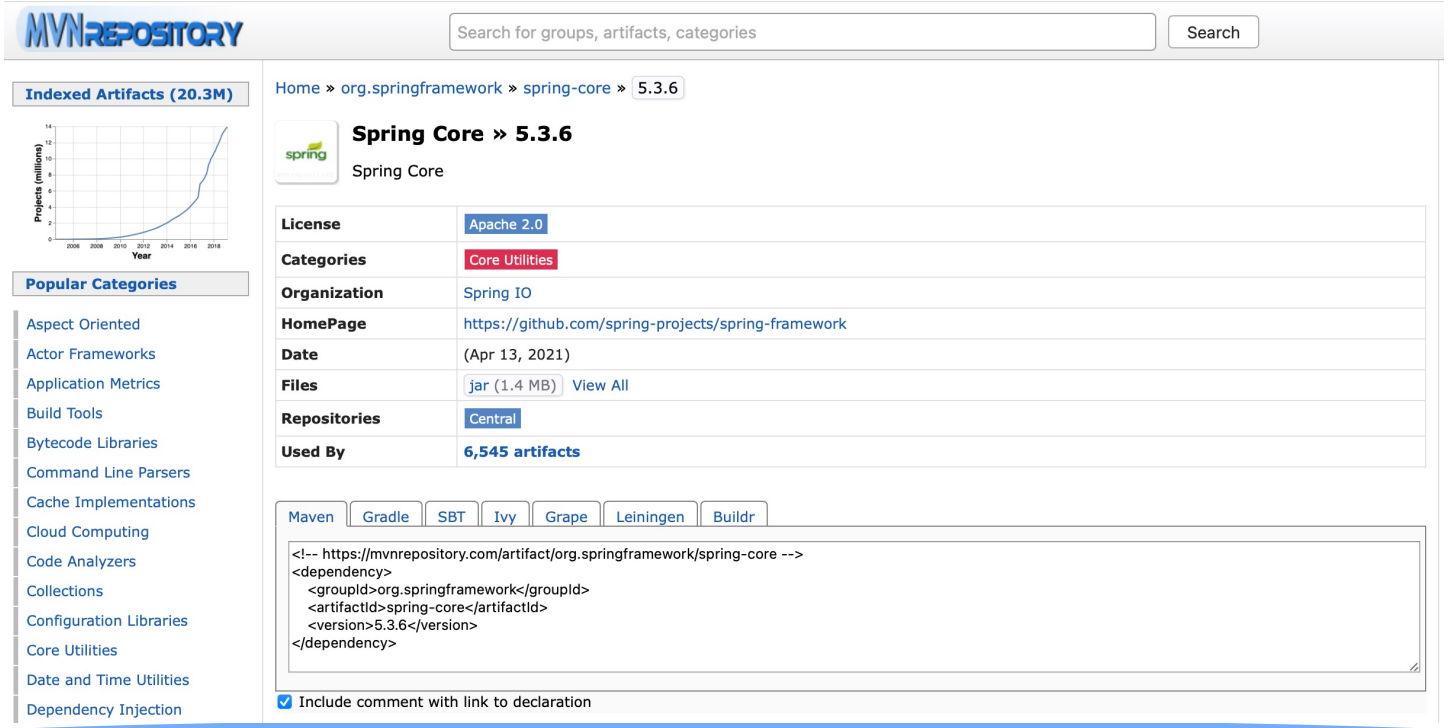
Add External Dependencies

- The dependencies section of the pom.xml lists all of the external dependencies that our project needs in order to build

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.12</version>
      <scope>compile</scope>
    </dependency>
  </dependencies>
</project>
```

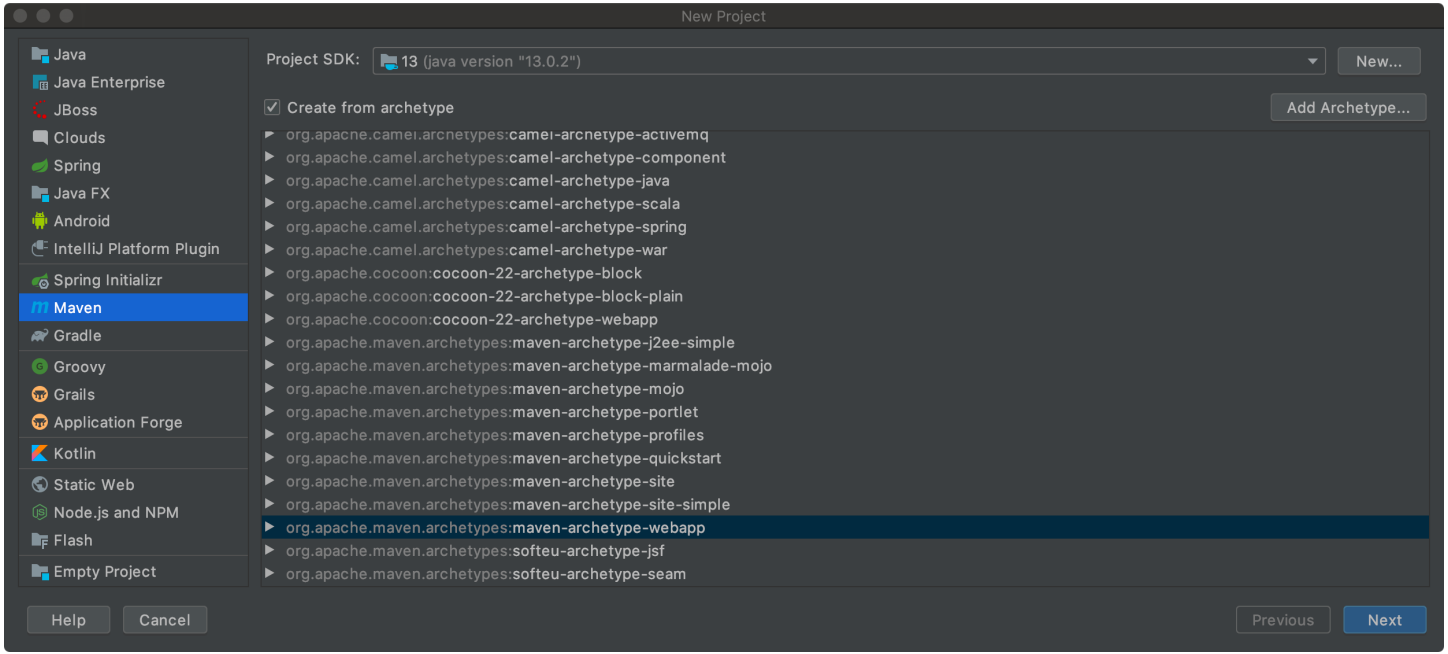
MVNREPOSITORY

- https://mvnrepository.com



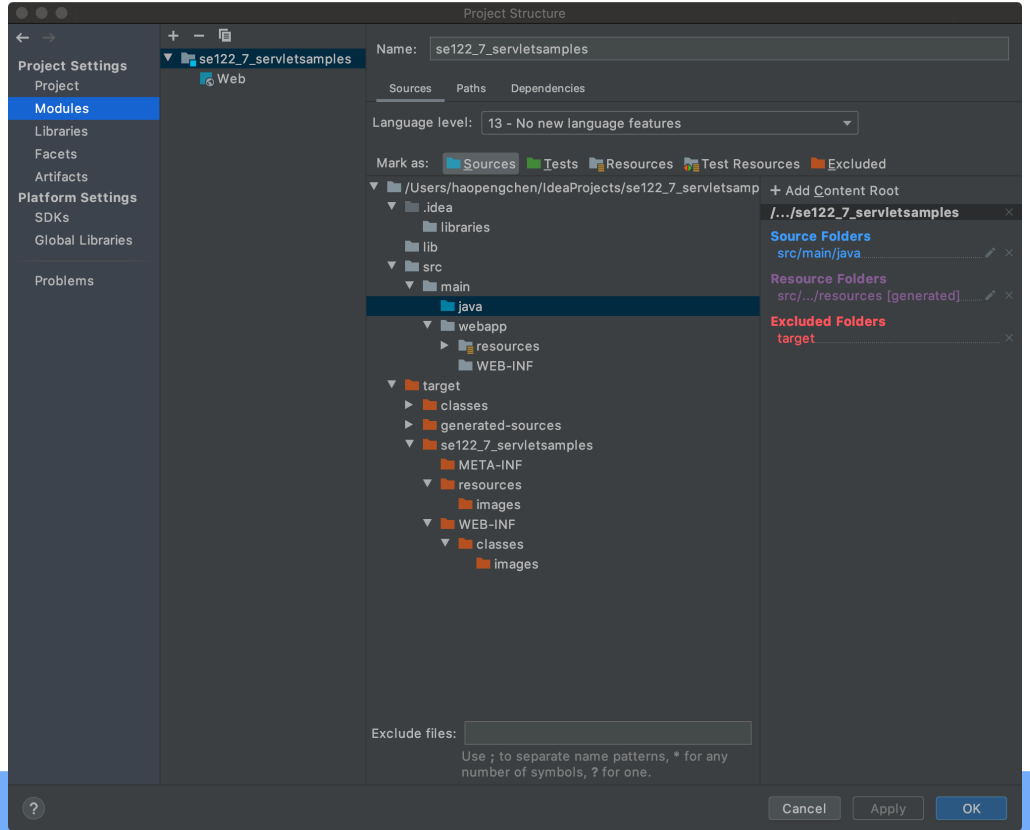
Maven in IntelliJ IDEA

- New Maven Project – Create from archtetype

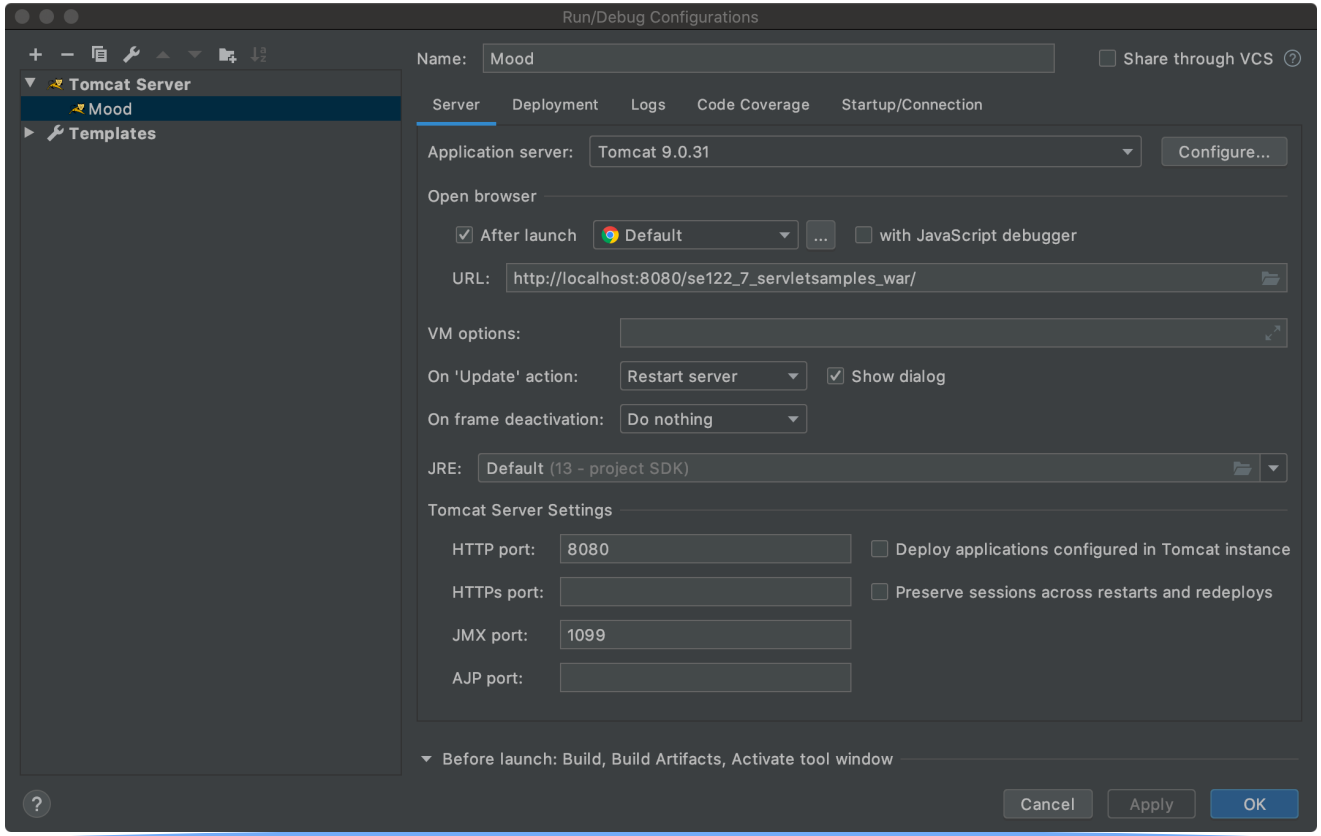


Maven in IntelliJ IDEA

- Create `source` and `resource` folders



Maven in IntelliJ IDEA



References

- What is Maven?
  - http://maven.apache.org
  - http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
  - http://maven.apache.org/guides/getting-started/index.html
  - http://maven.apache.org/pom.html#
- Maven in IDEA
  - https://www.jetbrains.com/help/idea/maven.html
  - https://www.jetbrains.com/help/idea/maven-support.html
- MVNREPOSITORY
  - https://mvnrepository.com

- *Web开发技术*
- *Web Application Development*

Thank You!