

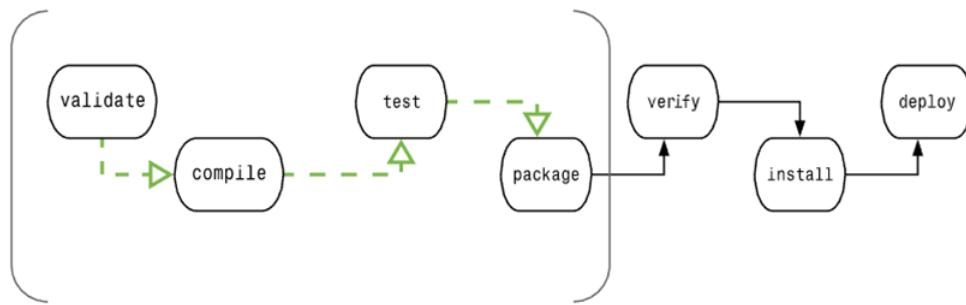
[CS304] Tutorial 6 - Maven & Build Systems

Part 1. Maven Intro & Setup

Maven is a software project management and comprehension tool, can be used for building and managing any Java-based project. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

1. Making the build process easy
2. Providing a uniform build system
3. Providing quality project information
4. Encouraging better development practices

A Build Lifecycle is Made Up of Phases. Each of these build lifecycles is defined by a list of build phases.



When the default lifecycle is used, Maven will first validate the project, then will try to compile the sources, run those against the tests, package the binaries (e.g. jar, war), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository. For example, the default lifecycle comprises of the following phases:

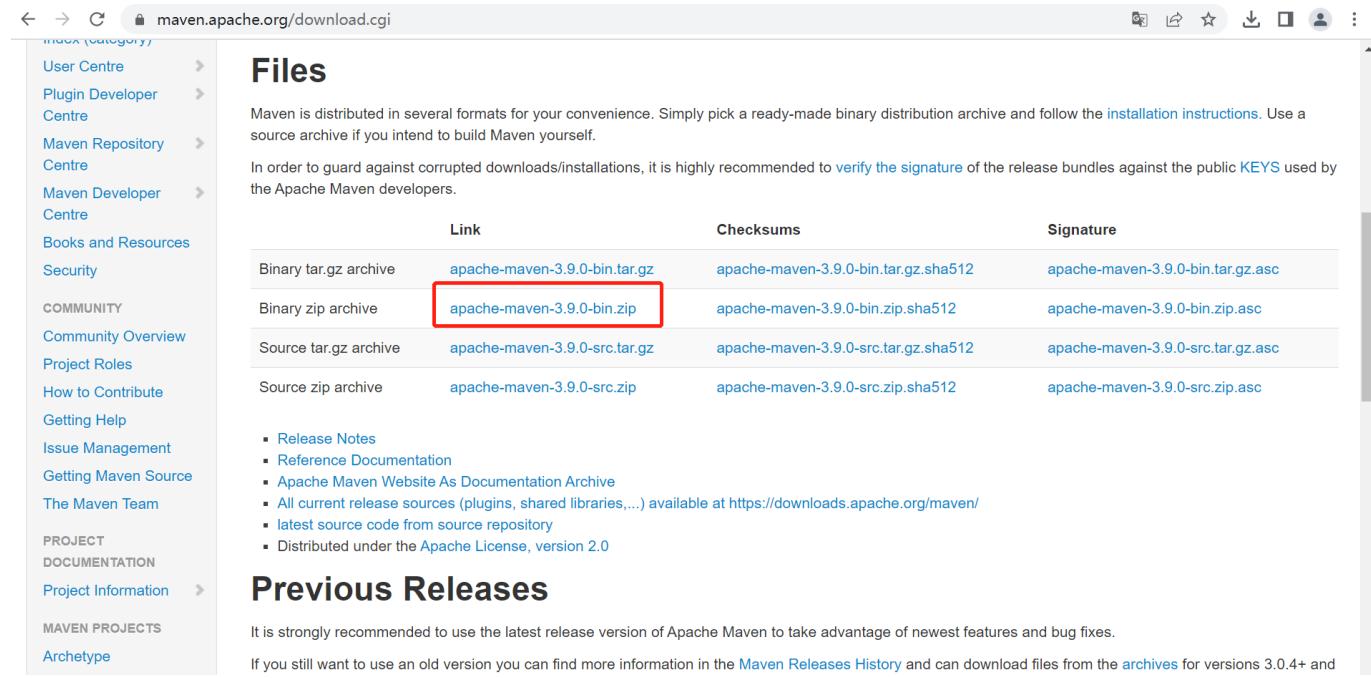
```
validate - validate the project is correct and all necessary information is available
compile - compile the source code of the project
test      - test the compiled source code using a suitable unit testing framework.
These tests should not require the code be packaged or deployed
package   - take the compiled code and package it in its distributable format, such as a JAR.
verify    - run any checks on results of integration tests to ensure quality criteria are met
install   - install the package into the local repository, for use as a dependency in other projects locally
deploy    - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.
```

These lifecycle phases (other lifecycle phases not shown here) are executed sequentially to complete the default lifecycle.

Maven official website:

<https://maven.apache.org/index.html>

Download and install [here](#):



The screenshot shows the Apache Maven download page. On the left is a sidebar with links like User Centre, Plugin Developer Centre, Maven Repository Centre, etc. The main content area has a heading 'Files'. It says Maven is distributed in several formats. Below is a table with columns: Link, Checksums, and Signature. The 'Link' column shows four options: Binary tar.gz archive (apache-maven-3.9.0-bin.tar.gz), Binary zip archive (apache-maven-3.9.0-bin.zip), Source tar.gz archive (apache-maven-3.9.0-src.tar.gz), and Source zip archive (apache-maven-3.9.0-src.zip). The 'apache-maven-3.9.0-bin.zip' link is highlighted with a red box. Below the table is a list of notes about release sources and licenses.

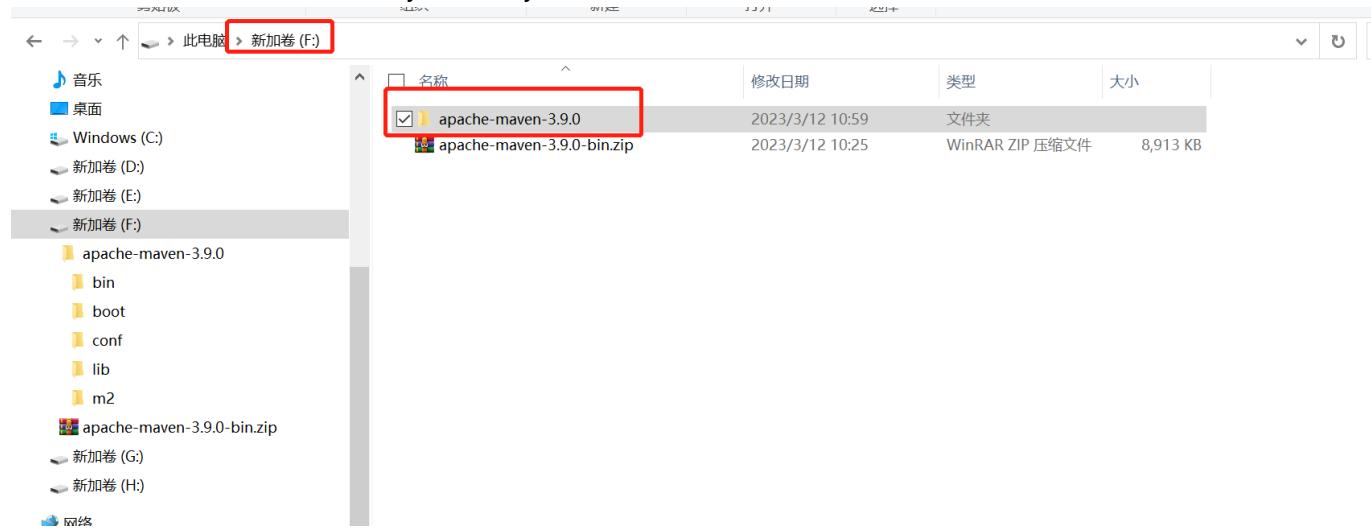
Link	Checksums	Signature
Binary tar.gz archive apache-maven-3.9.0-bin.tar.gz	apache-maven-3.9.0-bin.tar.gz.sha512	apache-maven-3.9.0-bin.tar.gz.asc
Binary zip archive apache-maven-3.9.0-bin.zip	apache-maven-3.9.0-bin.zip.sha512	apache-maven-3.9.0-bin.zip.asc
Source tar.gz archive apache-maven-3.9.0-src.tar.gz	apache-maven-3.9.0-src.tar.gz.sha512	apache-maven-3.9.0-src.tar.gz.asc
Source zip archive apache-maven-3.9.0-src.zip	apache-maven-3.9.0-src.zip.sha512	apache-maven-3.9.0-src.zip.asc

▪ Release Notes
▪ Reference Documentation
▪ Apache Maven Website As Documentation Archive
▪ All current release sources (plugins, shared libraries,...) available at <https://downloads.apache.org/maven/>
▪ latest source code from source repository
▪ Distributed under the Apache License, version 2.0

Previous Releases

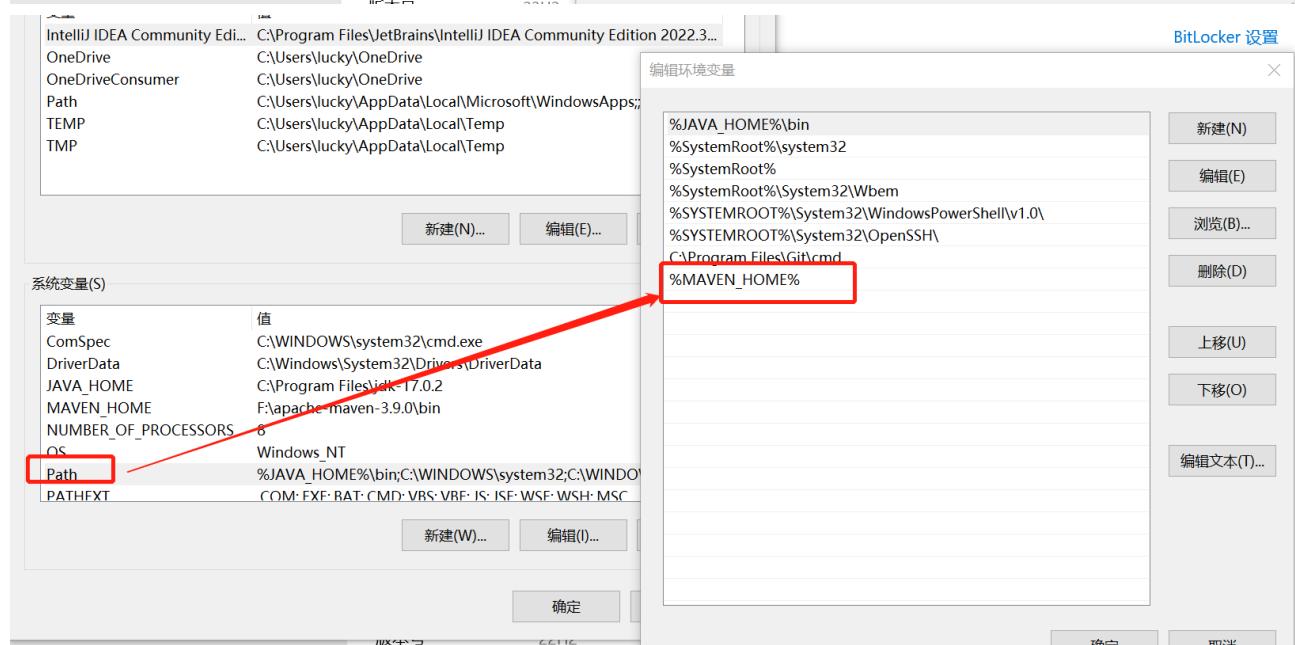
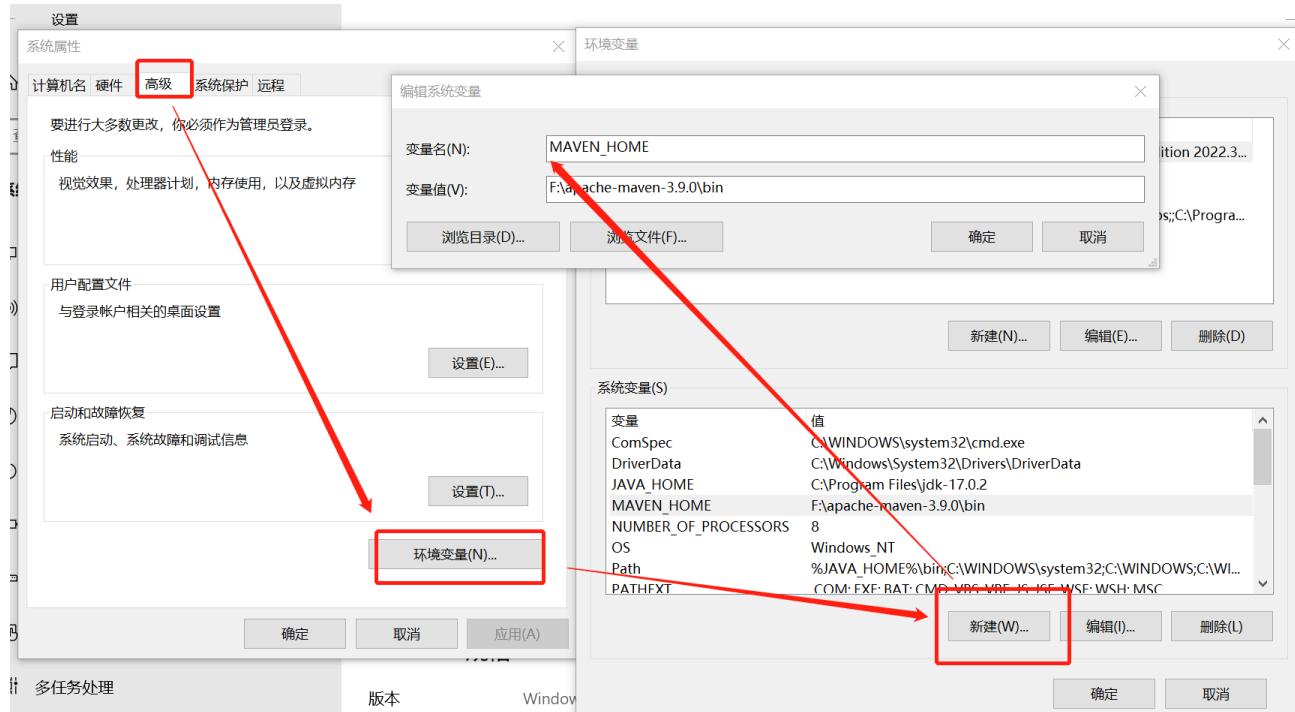
It is strongly recommended to use the latest release version of Apache Maven to take advantage of newest features and bug fixes.
If you still want to use an old version you can find more information in the [Maven Releases History](#) and can download files from the [archives](#) for versions 3.0.4+ and

Extract distribution archive in any directory:



The screenshot shows a Windows File Explorer window. The address bar shows '此电脑 > 新加卷 (F:)'. The left sidebar lists drives: 音乐, 桌面, Windows (C:), 新加卷 (D:), 新加卷 (E:), 新加卷 (F:). The main area shows a file list with two items: 'apache-maven-3.9.0-bin.zip' (WinRAR ZIP 压缩文件, 8,913 KB) and 'apache-maven-3.9.0' (文件夹). The 'apache-maven-3.9.0' folder is selected, indicated by a checked checkbox in the header. The file list header includes '名称', '修改日期', '类型', and '大小'.

Config environment variables: MAVEN_HOME



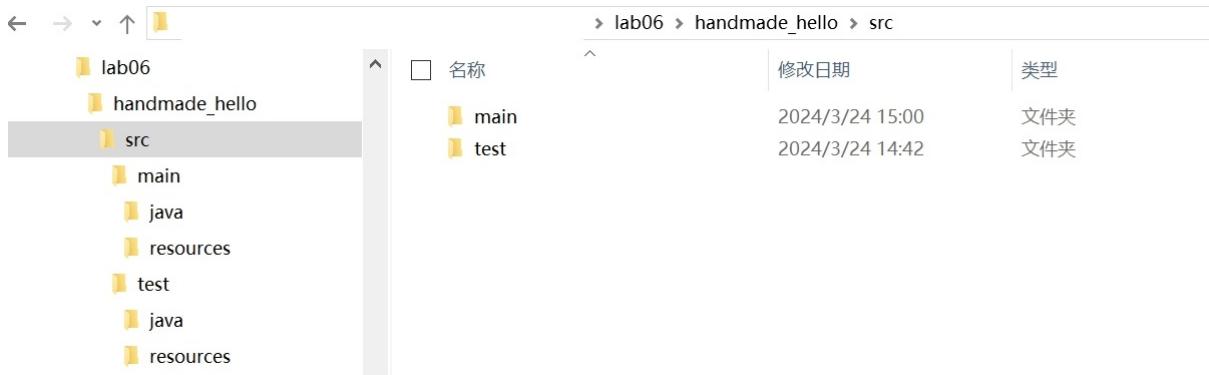
```
C:\> mvn -v
Apache Maven 3.9.0 (9b58d2bad23a66be161c4664ef21ce219c2c8584)
Maven home: F:\apache-maven-3.9.0
Java version: 17.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\jdk-17.0.2
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
C:\>
```

Part 2. Hello Maven & Basic Commands

1. Create a simple Java project

1. Create files and `pom.xml` below:

The `src/main/java` directory contains the project source code, the `src/test/java` directory contains the test source, and the `pom.xml` file is the project's Project Object Model, or POM.



2. Create `hello.java` (in `\src\main\java\lab06\helloMaven`) and `helloTest.java`(in `\src\test\java\lab06\helloMaven`):

```
package lab06.helloMaven;

public class hello {
    public static void main(String[] args) {
        hello hi=new hello();
        hi.echo("CS304");
    }

    public String echo(String name){
        String echo="hello,"+name;
        System.out.println(echo);
        return echo;
    }
}
```



```

package lab06.helloMaven;

import org.junit.Test;
import org.junit.Assert;

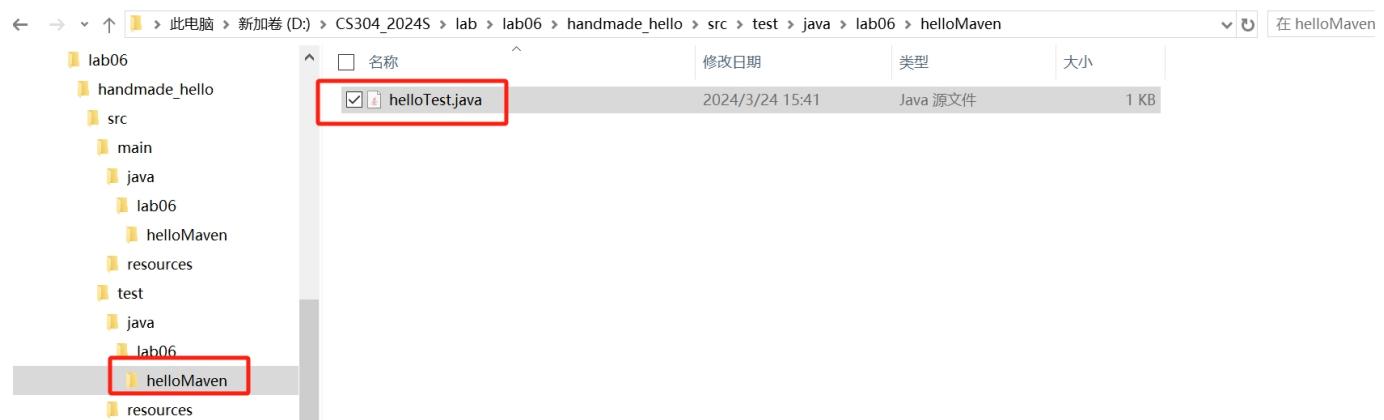
public class helloTest {

    @Test
    public void testEcho(){
        hello hi_test=new hello();
        String echoTest=hi_test.echo("CS304");
        Assert.assertEquals("hello,CS304",echoTest);

    }

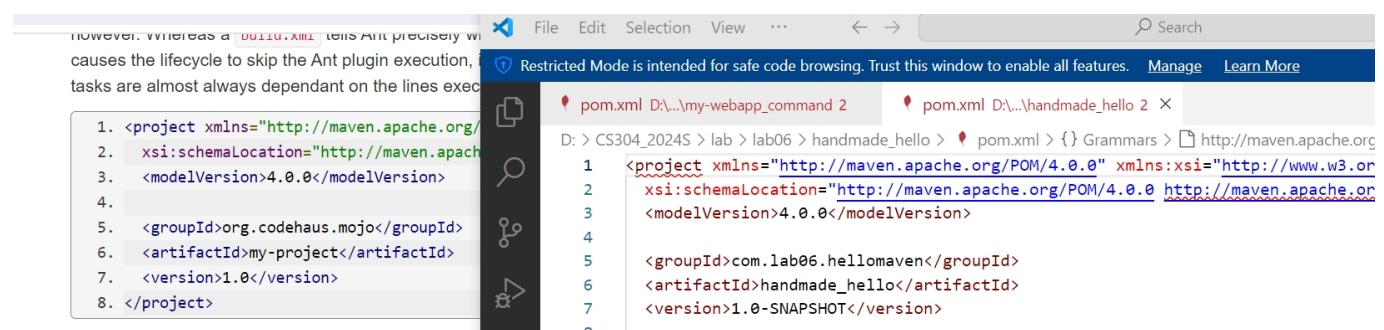
}

```



3. Maven Coordinates

1. modelVersion: pom model version.
2. groupId: This is generally unique amongst an organization or a project.
3. artifactId: The artifactId is generally the name that the project is known by.



Update our maven coordinates:

```

<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.lab06.hellomaven</groupId>
<artifactId>handmade_hello</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
</properties>

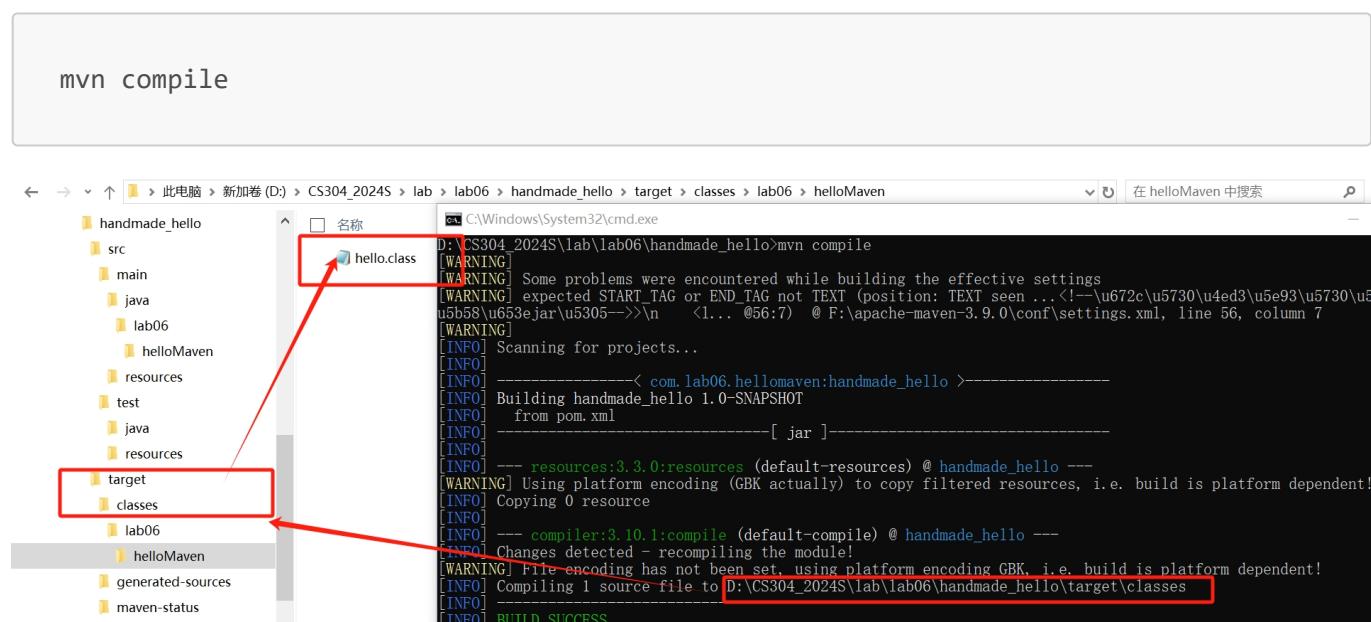
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
</dependencies>

</project>

```

4. Run basic maven commands:

1.



```

mvn compile

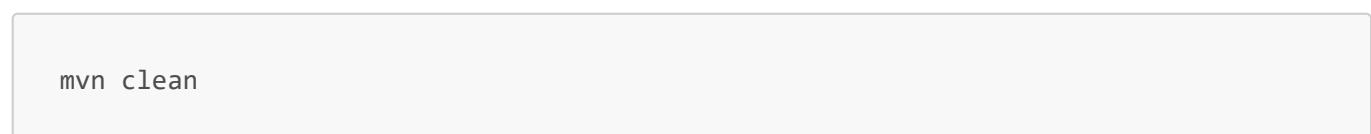
```

```

D:\CS304_2024S\lab\lab06\handmade_hello>mvn compile
[WARNING] Some problems were encountered while building the effective settings
[WARNING] expected START_TAG or END_TAG not TEXT (position: TEXT seen ...<!--\u672c\u5730\u4ed3\u5e93\u5730\u5
u5b58\u653ejar\u5305-->\n      <... @56:7)  @ F:\apache-maven-3.9.0\conf\settings.xml, line 56, column 7
[WARNING]
[INFO] Scanning for projects...
[INFO] -----
[INFO] < com.lab06.hellomaven:handmade_hello >-
[INFO] Building handmade_hello 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----
[INFO] --- resources:3.3.0:resources (default-resources) @ handmade_hello ---
[WARNING] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- compiler:3.10.1:compile (default-compile) @ handmade_hello ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding GBK, i.e. build is platform dependent!
[INFO] Compiling 1 source file to D:\CS304_2024S\lab\lab06\handmade_hello\target\classes
[INFO]
[INFO] BUILD SUCCESS

```

2.



```

mvn clean

```

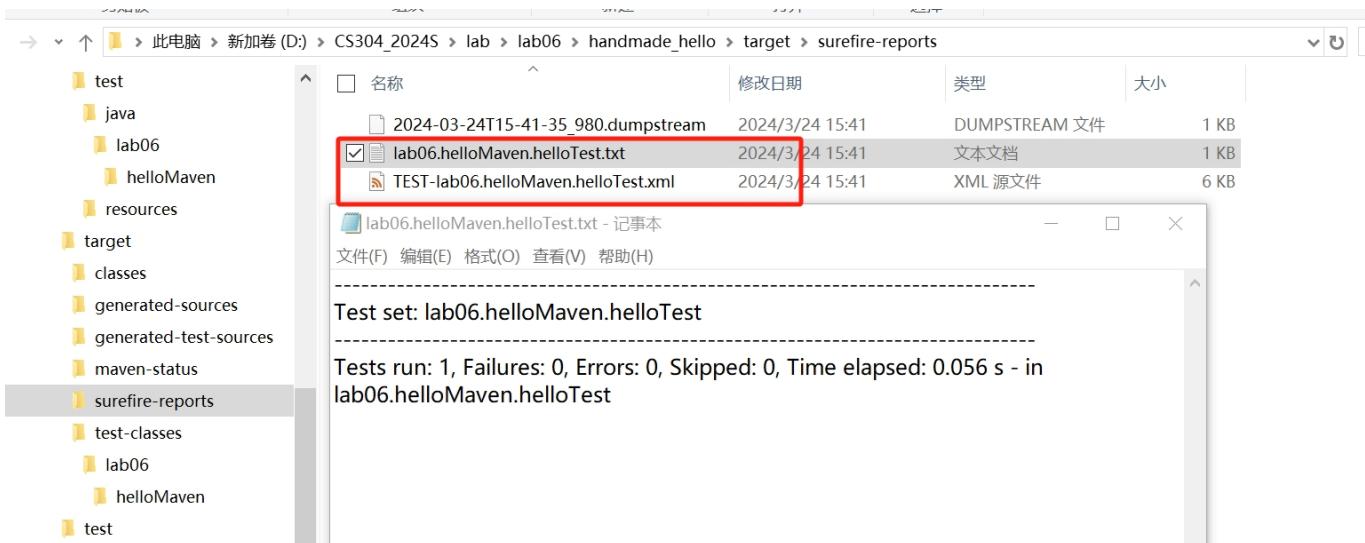
```
D:\CS304_2024S\lab\lab06\handmade_hello>mvn clean
[WARNING] Some problems were encountered while building the effective settings
[WARNING] expected START_TAG or END_TAG not TEXT (position: TEXT seen ...<!--\u672c\u5730\u5b58\u653ejar\u5305-->\n <... @56:7) @ F:\apache-maven-3.9.0\conf\settings.xml, lin
[WARNING]
[INFO] Scanning for projects...
[INFO] < com.lab06.hellomaven:handmade_hello >
[INFO] Building handmade_hello 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO]   [ jar ]
[INFO]   clean:3.2.0:clean (default clean) @ handmade_hello
[INFO] Deleting D:\CS304_2024S\lab\lab06\handmade_hello\target
[INFO] BUILD SUCCESS
[INFO] Total time: 0.484 s
[INFO] Finished at: 2024-03-15 22:44:28
[INFO]
```

3.

```
mvn test
```

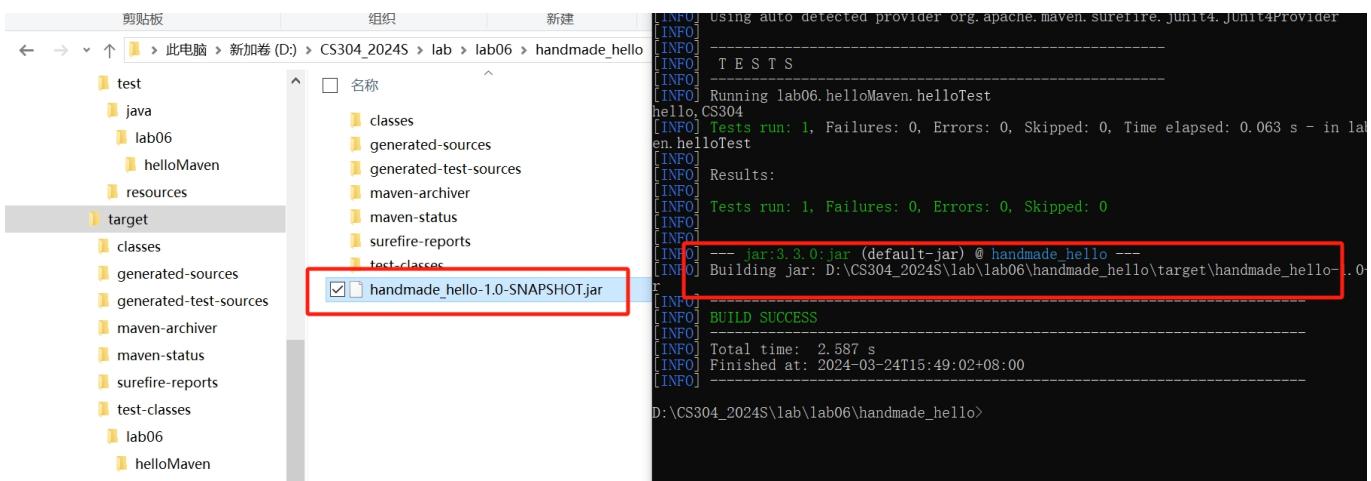
```
[INFO] < com.lab06.hellomaven:handmade_hello >
[INFO] Building handmade_hello 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO]   [ jar ]
[INFO]
[INFO] --- resources:3.3.0:resources (default-resources) @ handmade_hello ---
[WARNING] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- compiler:3.10.1:compile (default-compile) @ handmade_hello ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.3.0:testResources (default-testResources) @ handmade_hello ---
[WARNING] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- compiler:3.10.1:testCompile (default-testCompile) @ handmade_hello ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding GBK, i.e. build is platform dependent!
[INFO] Compiling 1 source file to D:\CS304_2024S\lab\lab06\handmade_hello\target\test-classes
[INFO]
[INFO] --- surefire:3.0.0-M8:test (default-test) @ handmade_hello ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit4.JUnit4Provider
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/surefire-junit4/3.0.0-M8/surefire-junit4-3.0.0-M8.jar
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/surefire-junit4/3.0.0-M8/surefire-junit4-3.0.0-M8.jar (18 kB at 12 kB/s)
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/surefire-junit4/3.0.0-M8/surefire-junit4-3.0.0-M8.pom
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/surefire-junit4/3.0.0-M8/surefire-junit4-3.0.0-M8.pom (2.9 kB at 7.2 kB/s)
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/common-junit4/3.0.0-M8/common-junit4-3.0.0-M8.pom
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/common-junit4/3.0.0-M8/common-junit4-3.0.0-M8.pom (3.0 kB at 6.5 kB/s)
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/common-junit4/3.0.0-M8/common-junit4-3.0.0-M8.jar
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/surefire/common-junit4/3.0.0-M8/common-junit4-3.0.0-M8.jar (26 kB at 49 kB/s)
[INFO]
[INFO]
[INFO] T E S T S
[INFO]
[INFO] Running lab06.helloMaven.helloTest
Hello,CS304
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.056 s - in lab06.helloMaven.helloTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
```

Check the test result:



4.

```
mvn package
```



```
java -cp target/handmade_hello-1.0-SNAPSHOT.jar lab06.helloMaven.hello
```

```
D:\CS304_2024S\lab\lab06\handmade_hello>java -cp target/handmade_hello-1.0-SNAPSHOT.jar lab06.helloMaven.hello
hello, CS304
```

5.

```
mvn install
```

```
[INFO] [INFO] Running lab06.helloMaven.helloTest
hello,CS304
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.06 s - in lab06.helloMaven.helloTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ handmade_hello ---
[INFO]
[INFO] --- install:3.1.0:install (default-install) @ handmade_hello ---
[INFO] Installing D:\CS304_2024S\lab\lab06\handmade_hello\pom.xml to F:\apache-maven-3.9.0\m2\repository\com\lab06\hello
maven\handmade_hello\1.0-SNAPSHOT\handmade_hello-1.0-SNAPSHOT.pom
[INFO] Installing D:\CS304_2024S\lab\lab06\handmade_hello\target\handmade_hello-1.0-SNAPSHOT.jar to F:\apache-maven-3.9.0\m2\repository\com\lab06\hellomaven\handmade_hello\1.0-SNAPSHOT\handmade_hello-1.0-SNAPSHOT.jar
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 4.613 s
[INFO] Finished at: 2024-03-24T15:56:09+08:00
[INFO]

D:\CS304_2024S\lab\lab06\handmade_hello>
```

my local repo

2. Standard Directory Layout

Maven has its own standard directory layout: <https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html> <https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html>

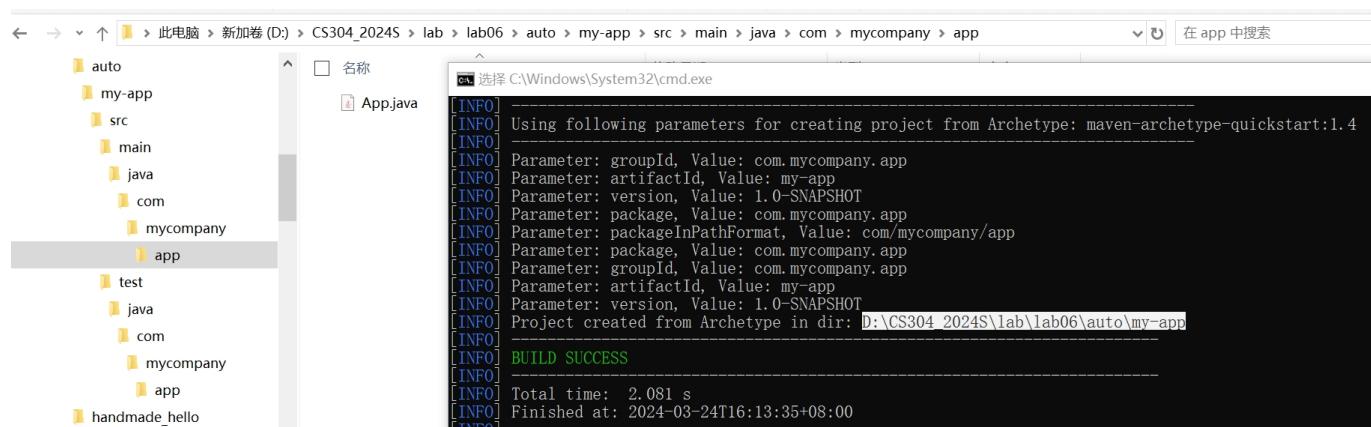
Directory	Purpose
src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/filters	Test resource filter files
src/it	Integration Tests (primarily for plugins)
src/assembly	Assembly descriptors
src/site	Site
LICENSE.txt	Project's license
NOTICE.txt	Notices and attributions required by libraries that the project depends on
README.txt	Project's readme

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

We use this command to create a directory and start it. This `archetype:generate` goal created a simple project based upon a maven-archetype-quickstart archetype.

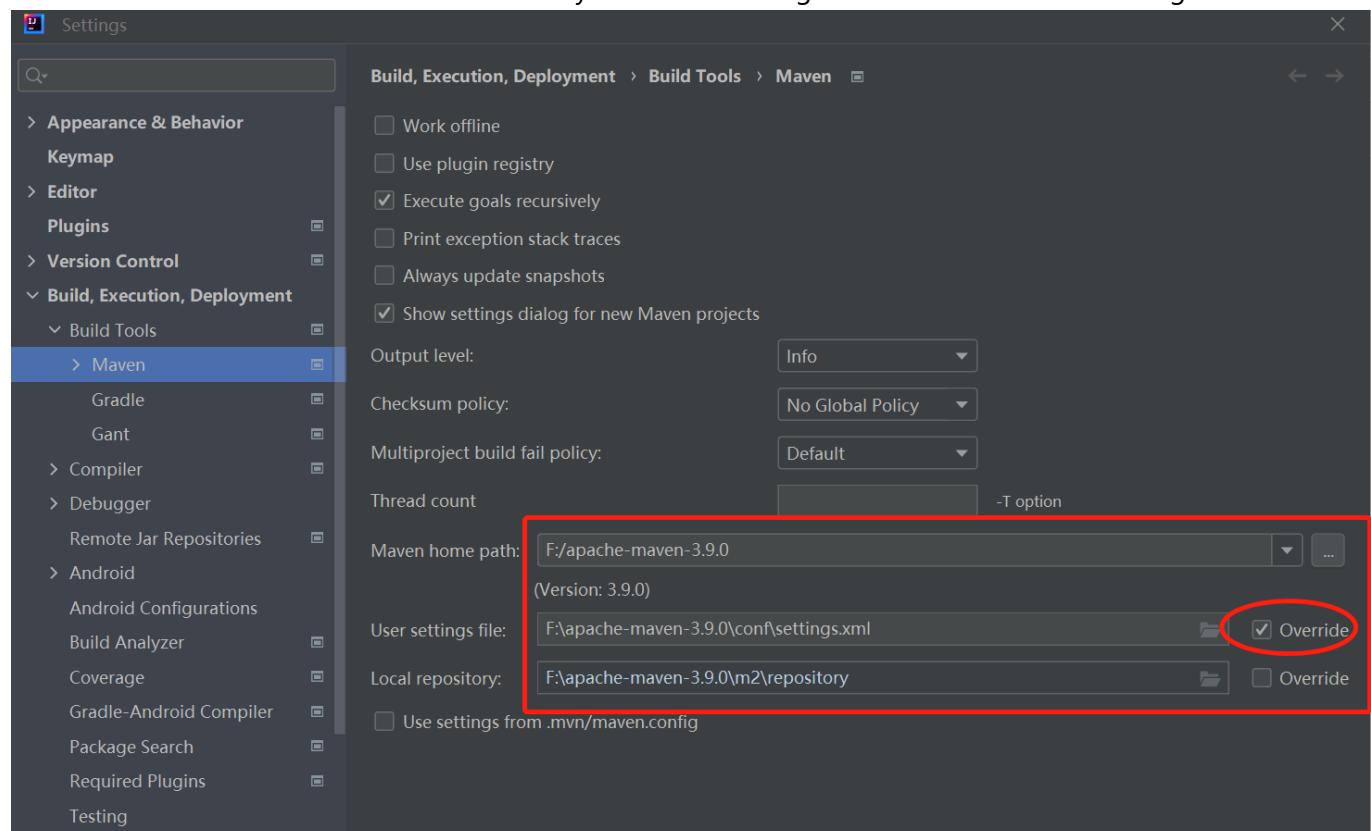
You can rename as you like:

```
-DgroupId=com.mycompany.app
-DartifactId=my-app
```

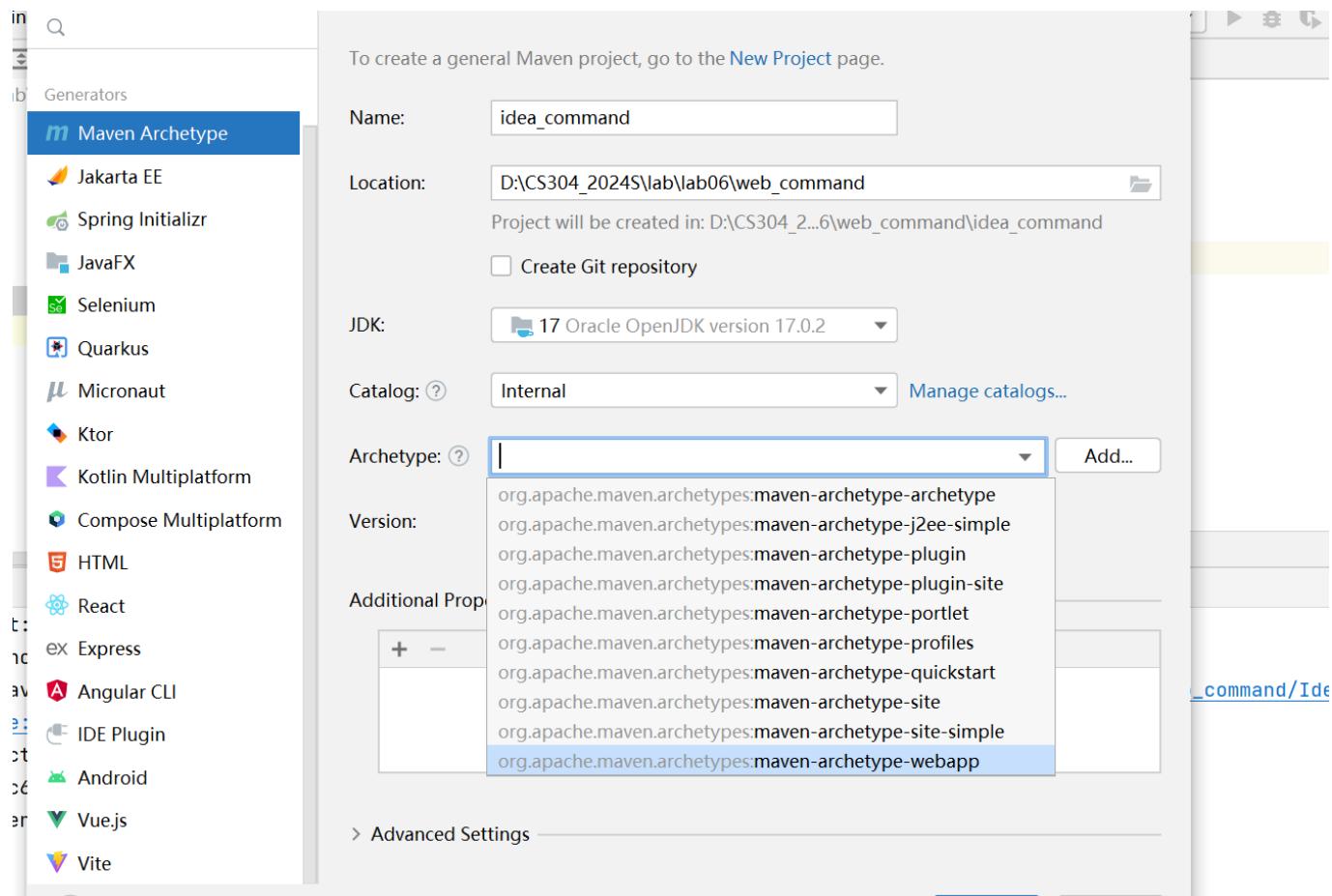


Part 3. Use Maven in IDEA & Run a web app with Jetty

Maven is bundled in IDEA. You can also modify the Maven configuration in IDEA: File -> Settings:



1. Create a Maven project



2. Add the java servlet dependency



3. Add jetty plugin:

```

<plugin>
    <groupId>org.eclipse.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>11.0.14</version>
</plugin>

```

4. Run jetty server:

```
mvn install
mvn jetty:run
```

The screenshot shows an IDE interface with the following details:

- Project View:** Shows a tree structure of files and folders. Under "my-webapp_command", there are ".idea", "src" (containing "main" and "webapp" folders), "target", and "pom.xml".
- Terminal:** Displays Maven command-line output. The output includes logs about Jetty version 11.0.14, session names, and server connectors. A red box highlights the line "[INFO] Started ServerConnector@5d71b500{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}".
- Code Editor:** Shows the content of the pom.xml file, specifically the pluginManagement section which defines the Jetty plugin.

5. Open your browser:



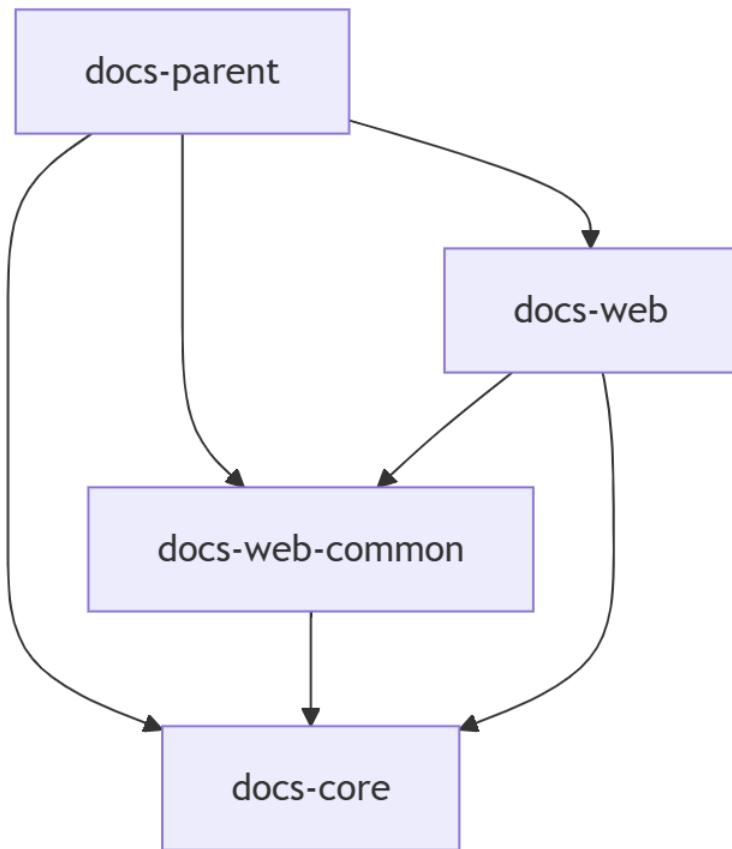
Part 4. Explore Teedy's dependencies

You may view project dependencies using maven command:

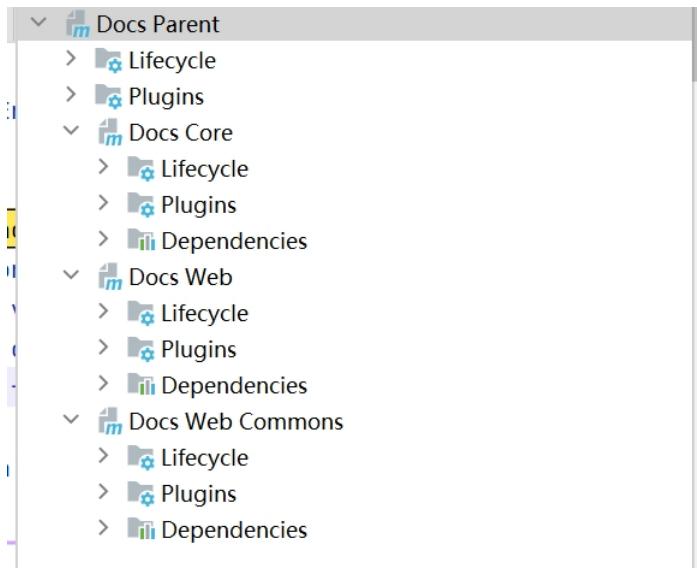
```
mvn dependency:tree
```

or in IDE, right click and select "Show Diagram".

Here is Teedy's module dependency:



Each module has its own `pom.xml`:



1. Root POM

The root `pom.xml` serves as the parent POM for the entire project

1. Module declaration

```

<!--manage shared configurations and dependencies for child modules.-->
<packaging>pom</packaging>

<!--Module list-->
<modules>
  
```

```
<module>docs-core</module>
<module>docs-web-common</module>
<module>docs-web</module>
</modules>
```

2. Dependency Management

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.sismics.docs</groupId>
      <artifactId>docs-core</artifactId>
      <version>${project.version}</version>
    </dependency>
  <dependencies>
    .....
<dependencyManagement>
```

2. Docs-core

This is a core module of a document management system project. Here are the main dependency categories:

- Persistence Layer:
 - Hibernate Core - For ORM (Object-Relational Mapping)
 - PostgreSQL - Database driver for production
 - H2 Database - For testing purposes
- Utility Libraries:
 - Joda-Time - For date/time handling
 - Google Guava - General utility library
 - Apache Commons (Compress, Lang3, Email) - Various utility functions
 - FreeMarker - Template engine
 - BCrypt - For password hashing
- Search & Text Processing:
 - Apache Lucene (multiple modules) - For search functionality:
- Document Processing:
 - PDFBox - PDF manipulation
 - etc.
- Network/Communication:
 - OkHttp - HTTP client
 - Apache Directory API - For LDAP integration

- Commons Email - For email functionality
- Logging:
 - Log4j
 - SLF4J (multiple modules)
- Testing:
 - JUnit - For unit testing

```
<dependency>

    <groupId>com.h2database</groupId>

    <artifactId>h2</artifactId>

    <scope>test</scope>

</dependency>
```

Teedy defines 2 profiles: dev and prod, which correspond to different configurations.

```
<profiles>
    <!-- Development profile (active by default) -->
    <profile>
        <id>dev</id>
        <activation>
            <activeByDefault>true</activeByDefault>
            <property>
                <name>env</name>
                <value>dev</value>
            </property>
        </activation>
    ....
    <!-- Production profile -->
    <profile>
        <id>prod</id>
    </profile>
</profiles>
```

They could be used as:

```
mvn clean install -P prod
mvn clean install -Denv=dev
```

3. Docs-web-common

This is a common web module that provides shared web functionality, REST API capabilities, and utilities that can be used across other web modules in the project. It sits between the core module and web-specific implementations.

4. Docs-web

The docs-web module is packaged as a WAR file and serves as the main web application interface for Teedy. The module integrates with docs-core for business logic and docs-web-common for shared functionality.