## Maven三种打包方式以及区别：

* mvn clean package

完成了clean、resources、compile、testResources、testCompile、test、jar七个阶段，主要完成了项目编译、单元测试、打包功能，但没有将打好的可执行jar包部署到本地maven仓库或者远程maven私服仓库。

* mvn clean install

完成了clean、resources、compile、testResources、testCompile、test、jar、install八个阶段，主要完成了项目编译、单元测试、打包功能，并将打好的可执行jar包部署到本地maven仓库,但没有安装到远程maven私服仓库。

* mvn clean deploy

完成了clean、resources、compile、testResources、testCompile、test、jar、install、deploy八个阶段，主要完成了项目编译、单元测试、打包功能，将打好的可执行jar包部署到本地maven仓库，也安装到远程maven私服仓库。

**Maven打包实操：**

如果在项目的pom.xml文件中没有指明打包类型，则默认打成jar包。使用maven命令  mvn clean package进行项目的打包，可看到如下输出结果：

|  |
| --- |
| [INFO] Results:  [INFO]  [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  [INFO]  [INFO]  [INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ springboot ---  [INFO] Building jar: D:\Spring\springboot\target\springboot-0.0.1-SNAPSHOT.jar  [INFO]  [INFO] --- spring-boot-maven-plugin:2.1.6.RELEASE:repackage (repackage) @ springboot ---  [INFO] Replacing main artifact with repackaged archive  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time: 16.318 s  [INFO] Finished at: 2019-07-25T22:42:05+08:00  [INFO] Final Memory: 35M/252M  [INFO] ------------------------------------------------------------------------ |

可以看到   maven-jar-plugin:3.1.2:jar (default-jar) @ springboot  其实就是利用jar插件将项目打成了springboot-0.0.1-SNAPSHOT.jar，该文件会存放在target目录下。

如果要让其他的maven项目直接引用这个jar包，还要执行mvn clean install命令，输出结果为：

|  |
| --- |
| [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  [INFO]  [INFO]  [INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ springboot ---  [INFO] Building jar: D:\Spring\springboot\target\springboot-0.0.1-SNAPSHOT.jar  [INFO]  [INFO] --- spring-boot-maven-plugin:2.1.6.RELEASE:repackage (repackage) @ springboot ---  [INFO] Replacing main artifact with repackaged archive  [INFO]  [INFO] --- maven-install-plugin:2.5.2:install (default-install) @ springboot ---  [INFO] Installing D:\Spring\springboot\target\springboot-0.0.1-SNAPSHOT.jar to D:\Spring\maven-space  \repository\com\example\springboot\0.0.1-SNAPSHOT\springboot-0.0.1-SNAPSHOT.jar  [INFO] Installing D:\Spring\springboot\pom.xml to D:\Spring\maven-space\repository\com\example\sprin  gboot\0.0.1-SNAPSHOT\springboot-0.0.1-SNAPSHOT.pom  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time: 18.514 s  [INFO] Finished at: 2019-07-25T22:53:22+08:00  [INFO] Final Memory: 35M/261M  [INFO] ---------------------------------------------------------------------- |

在打包之后，我们又执行了安装任务install:install，从输出我们看到该任务将项目输出的jar安装到了Maven本地仓库中，我们可以打开相应的文件夹看到项目的pom和jar。只有将项目的构件安装到本地仓库之后，其他Maven项目才能使用它。

## Pom文件常用标签：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <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">  <!--声明项目遵循的pom模型版本-->  <modelVersion>4.0.0</modelVersion>  <!--父项目的坐标，如果没有规定某个元素的值，则默认为项目值。坐标包括groupId、artifactId、version-->  <parent>  <!--被继承的父项目的全球唯一标识符-->  <groupId>org.springframework.boot</groupId>  <!--被继承的父项目的构建标识符-->  <artifactId>spring-boot-starter-parent</artifactId>  <!--被继承的父项目的版本-->  <version>2.1.6.RELEASE</version>  <!--父项目pom.xml的相对路径。默认值是../pom.xml.maven首先在构建当前项目的地方寻找父项目的pom文件，其次在  文件系统的这个位置，然后再本地仓库，最后在远程仓库寻找父项目的pom文件-->  <relativePath/> <!-- lookup parent from repository -->  </parent>  <!--项目的全球唯一标识符，通常使用全限定的包名区分该项目和其他项目的。并且构建时生成的路径也是如此，比如com.example  对应路径为/com/example-->  <groupId>com.example</groupId>  <!--构建的标识符和groupId一起唯一标识一个构件。两个不同的项目不可能同时有相同的groupId、artifactId。maven为项目产生的构建包括jars、  源码、二进制发布、wars等 一个groupId下面可能有多个项目，就是通过artifactId区分的-->  <artifactId>demo</artifactId>  <!--项目产生的构件类型，例如：jar、war等 打包机制-->  <packaging>jar</packaging>  <!--项目的版本号，格式为：主版本.次版本.增量版本-限定版本号-->  <version>0.0.1-SNAPSHOT</version>  <!--项目名称，maven产生的文档用-->  <name>demo</name>  <!--项目主页的url，maven产生的文档用-->  <url></url>  <description>Demo project for Spring Boot</description>  <!--配置依赖版本-->  <properties>  <!--java环境设置-->  <java.version>1.8</java.version>  </properties>  <!--定义项目的依赖关系-->  <dependencies>  <!--每个依赖都对应一个jar包-->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter</artifactId>  </dependency>   <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <!--使用的插件列表-->  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |