# Maven Jar插件

该插件提供了构建Jar包的能力。

## 1.可用的Goal

jar:jar => 为项目创建jar

jar:test-jar => 为测试代码创建jar

## 2.Usage：用法

<http://maven.apache.org/plugins/maven-jar-plugin/usage.html>

在pom.xml配置maven-jar插件，并且需要将打包方式设置为jar：

<packaging>jar</packaging>

<plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <version>2.6</version>  
</plugin>

## 3. Archiver配置

Jar插件使用Maven Archiver来处理jar内容和清单配置。

<archive>  
 <!--  
 是否在归档包的路径META-INF/maven/${groupId}/${artifactId}包含下面两个文件：  
 pom.xml  
 pom.properties  
 默认值：true  
 -->  
 <addMavenDescriptor/>  
 <!-- 是否进行压缩，默认值：true -->  
 <compress/>  
 <!-- 是否强制重新创建存档，默认值：true -->  
 <forced/>  
 <!-- 创建的归档文件是否包含INDEX.LIST文件，默认值是false。 -->  
 <index/>  
 <!-- -->  
 <manifest>  
 <!-- 是否创建一个Class-Path清单条目，默认false -->  
 <addClasspath/>  
 <!--  
 如果设置为true，则manifest将会包含下面内容，默认false  
 Implementation-Title: ${project.name}  
 Implementation-Version: ${project.version}  
 Implementation-Vendor-Id: ${project.groupId}  
 Implementation-Vendor: ${project.organization.name}  
 Implementation-URL: ${project.url}  
 -->  
 <addDefaultImplementationEntries/>  
 <!--  
 如果设置为true，则manifest将会包含下面内容，默认false  
 Specification-Title: ${project.name}  
 Specification-Version: ${project.artifact.selectedVersion.majorVersion}.${project.artifact.selectedVersion.minorVersion}  
 Specification-Vendor: ${project.organization.name}  
 -->  
 <addDefaultSpecificationEntries/>  
 <!-- 是否创建扩展列表清单条目,默认值是false。 -->  
 <addExtensions/>  
 <!-- 格式化Class-Path时需要使用的布局类型，有效值为:simple/repository/custom  
 默认值：simple  
 -->  
 <classpathLayoutType/>  
 <!-- 是否使用repository进行格式化，默认false -->  
 <classpathMavenRepositoryLayout/>  
 <!-- 作为所有Class-Path前缀的文本,默认值"" -->  
 <classpathPrefix/>  
 <!-- -->  
 <customClasspathLayout/>  
 <!-- 设置manifest清单里面的Main-Class属性 -->  
 <mainClass/>  
 <!-- 设置manifest清单里面的Package属性 -->  
 <packageName/>  
 <!-- 是否使用唯一的时间戳快照版本而不是-SNAPSHOT版本，默认true -->  
 <useUniqueVersions/>  
 </manifest>  
 <!-- 添加到manifest清单的key/value对 -->  
 <manifestEntries>  
 <key>value</key>  
 </manifestEntries>  
 <!-- 定义自己的manifest文件 -->  
 <manifestFile/>  
 <manifestSections>  
 <!-- -->  
 <manifestSection>  
 <!-- section的名字 -->  
 <name/>  
 <!-- 添加到manifest清单文件的key/value对 -->  
 <manifestEntries>  
 <!-- -->  
 <key>value</key>  
 </manifestEntries>  
 </manifestSection>  
 </manifestSections>  
 <!-- 使用它来覆盖自动创建的pom.properties文件（只有在addMavenDescriptor设置为true的情况下） -->  
 <pomPropertiesFile/>  
</archive>

## 4. Examples：例子

<http://maven.apache.org/plugins/maven-jar-plugin/>

为构建的jar排除或保护文件：

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

<configuration>

<includes>

<include>\*\*/service/\*</include>

</includes>

</configuration>

</plugin>

构建的jar包含测试类：

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

<executions>

<execution>

<goals>

<goal>test-jar</goal>

</goals>

</execution>

</executions>

</plugin>

从项目中创建附加的jar包：

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>jar</goal>

</goals>

<configuration>

<classifier>client</classifier>

<includes>

<include>\*\*/service/\*</include>

</includes>

</configuration>

</execution>

</executions>

</plugin>