<https://blog.csdn.net/qq_32736999/article/details/93395246>

在Maven中，主要有3个插件可以用来打包：

maven-jar-plugin，默认的打包插件，用来打普通的project JAR包；

maven-shade-plugin，用来打可执行JAR包，也就是所谓的fat JAR包；

maven-assembly-plugin，支持自定义的打包结构，也可以定制依赖项等。

我们日常使用的以maven-assembly-plugin为最多，因为大数据项目中往往有很多shell脚本、SQL脚本、.properties及.xml配置项等，采用assembly插件可以让输出的结构清晰而标准化。

要使用该插件，就在项目pom文件中加入以下内容。

<build>

<plugins>

<plugin>

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

<artifactId>maven-assembly-plugin</artifactId>

<version>${maven-assembly-plugin.version}<version>

<executions>

<execution>

<id>make-assembly</id>

<!-- 绑定到package生命周期 -->

<phase>package</phase>

<goals>

<!-- 只运行一次 -->

<goal>single</goal>

</goals>

</execution>

</executions>

<configuration>

<!-- 配置描述符文件 -->

<descriptor>src/main/assembly/assembly.xml</descriptor>

<!-- 也可以使用Maven预配置的描述符

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs> -->

</configuration>

</plugin>

</plugins>

</build>

assembly插件的打包方式是通过descriptor（描述符）来定义的。

Maven预先定义好的描述符有bin，src，project，jar-with-dependencies等。比较常用的是jar-with-dependencies，它是将所有外部依赖JAR都加入生成的JAR包中，比较傻瓜化。

但要真正达到自定义打包的效果，就需要自己写描述符文件，格式为XML。下面是我们的项目中常用的一种配置。

<assembly>

<id>assembly</id>

<formats>

<format>tar.gz</format>

</formats>

<includeBaseDirectory>true</includeBaseDirectory>

<fileSets>

<fileSet>

<directory>src/main/bin</directory>

<includes>

<include>\*.sh</include>

</includes>

<outputDirectory>bin</outputDirectory>

<fileMode>0755</fileMode>

</fileSet>

<fileSet>

<directory>src/main/conf</directory>

<outputDirectory>conf</outputDirectory>

</fileSet>

<fileSet>

<directory>src/main/sql</directory>

<includes>

<include>\*.sql</include>

</includes>

<outputDirectory>sql</outputDirectory>

</fileSet>

<fileSet>

<directory>target/classes/</directory>

<includes>

<include>\*.properties</include>

<include>\*.xml</include>

<include>\*.txt</include>

</includes>

<outputDirectory>conf</outputDirectory>

</fileSet>

</fileSets>

<files>

<file>

<source>target/${project.artifactId}-${project.version}.jar</source>

<outputDirectory>.</outputDirectory>

</file>

</files>

<dependencySets>

<dependencySet>

<unpack>false</unpack>

<scope>runtime</scope>

<outputDirectory>lib</outputDirectory>

</dependencySet>

</dependencySets>

</assembly>

id与formats

formats是assembly插件支持的打包文件格式，有zip、tar、tar.gz、tar.bz2、jar、war。可以同时定义多个format。

id则是添加到打包文件名的标识符，用来做后缀。

也就是说，如果按上面的配置，生成的文件就是artifactId−artifactId−{version}-assembly.tar.gz。

fileSets/fileSet

用来设置一组文件在打包时的属性。

directory：源目录的路径。

includes/excludes：设定包含或排除哪些文件，支持通配符。

fileMode：指定该目录下的文件属性，采用Unix八进制描述法，默认值是0644。

outputDirectory：生成目录的路径。

files/file

与fileSets大致相同，不过是指定单个文件，并且还可以通过destName属性来设置与源文件不同的名称。

dependencySets/dependencySet

用来设置工程依赖文件在打包时的属性。也与fileSets大致相同，不过还有两个特殊的配置：

unpack：布尔值，false表示将依赖以原来的JAR形式打包，true则表示将依赖解成\*.class文件的目录结构打包。

scope：表示符合哪个作用范围的依赖会被打包进去。compile与provided都不用管，一般是写runtime。

按照以上配置打包好后，将.tar.gz文件上传到服务器，解压之后就会得到bin、conf、lib等规范化的目录结构，十分方便。