

# maven 打包可运行 jar 包

## 目录

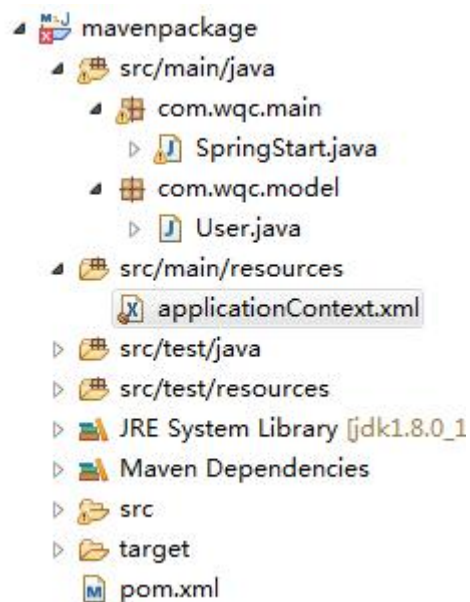
### 1、前提

Maven 可以使用 `mvn package` 指令对项目进行打包, 如果使用 `Java -jar xxx.jar` 执行运行 jar 文件, 会出现“no main manifest attribute, in xxx.jar”（没有设置 Main-Class）、`ClassNotFoundException`（找不到依赖包）等错误。

要想 jar 包能直接通过 `java -jar xxx.jar` 运行, 需要满足:

- 1、在 jar 包中的 META-INF/MANIFEST.MF 中指定 Main-Class, 这样才能确定程序的入口在哪里;
- 2、要能加载到依赖包。

示例项目:



pom.xml 基本配置:

```
<properties>
```

```
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>4.3.9.RELEASE</spring.version>

</properties>

<dependencies>

    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-context</artifactId>

        <version>${spring.version}</version>

    </dependency>

</dependencies>
```

使用 Maven 有以下几种方法可以生成能直接运行的 jar 包，可以根据需要选择一种合适的方法。

## 方法一：使用 maven-jar-plugin 和 maven-dependency-plugin 插件打包

在 pom.xml 中配置：

```
<build>

    <plugins>

        <plugin>

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

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

<version>2.6</version>

<configuration>

  <archive>

    <manifest>

      <addClasspath>true</addClasspath>

      <classpathPrefix>lib/</classpathPrefix>

      <mainClass>com.wqc.main.SpringStart</mainClass>

    </manifest>

  </archive>

</configuration>

</plugin>

<plugin>

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

  <artifactId>maven-dependency-plugin</artifactId>

  <version>2.10</version>

  <executions>

    <execution>

      <id>copy-dependencies</id>

      <phase>package</phase>

      <goals>

        <goal>copy-dependencies</goal>

      </goals>

    </execution>

  </executions>

</plugin>
```

```

        <configuration>

            <outputDirectory>${project.build.directory}/lib</outputDirectory>

        </configuration>

    </execution>

</executions>






</plugin>

</plugins>







</build>

```

执行 maven package 命令后生成:

	classes	2018/2/5 10:34	文件夹
	lib	2018/2/5 10:34	文件夹
	maven-archiver	2018/2/5 10:34	文件夹
	maven-status	2018/2/5 10:34	文件夹
	mavenpackage-1.0.0.jar	2018/2/5 10:34	Executable Jar File 5 KB

执行 maven install 命令后生成: install 和 package 命令的区别将在下一篇说明

	classes	2018/2/5 10:34	文件夹
	lib	2018/2/5 10:34	文件夹
	maven-archiver	2018/2/5 10:34	文件夹
	maven-status	2018/2/5 10:34	文件夹
	test-classes	2018/2/5 10:22	文件夹
	mavenpackage-1.0.0.jar	2018/2/5 10:22	Executable Jar File 5 KB

然后在文件夹中按 shift 鼠标右键点击打开 cmd 窗口,然后输入:java -jar mavenpackage-1.0.0.jar

完全能运行....

maven-jar-plugin 用于生成 META-INF/MANIFEST.MF 文件的部分内容, com.xxg.Main 指定 MANIFEST.MF 中的 Main-Class, true 会在 MANIFEST.MF 加上 Class-Path 项并配置依赖包, lib/ 指定依赖包所在目录。

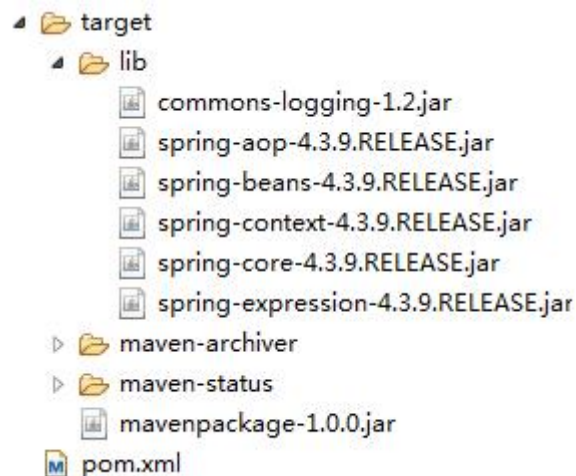
例如下面是一个通过 maven-jar-plugin 插件生成的 MANIFEST.MF 文件片段:

```
Class-Path: lib/commons-logging-1.2.jar lib/commons-io-2.4.jar

Main-Class: com.xxg.Main
```

只是生成 MANIFEST.MF 文件还不够, maven-dependency-plugin 插件用于将依赖包拷贝到 \${project.build.directory}/lib 指定的位置, 即 lib 目录下。

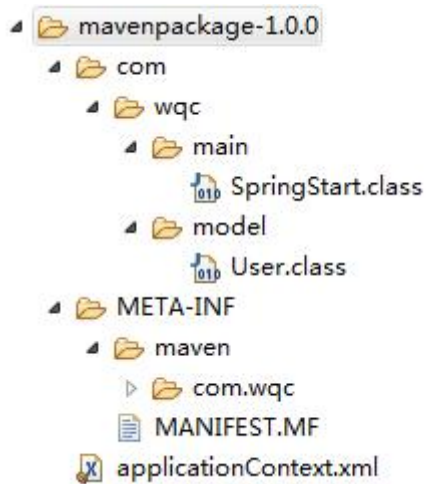
配置完成后, 通过 mvn package 指令打包, 会在 target 目录下生成 jar 包, 并将依赖包拷贝到 target/lib 目录下, 目录结构如下:



指定了 Main-Class, 有了依赖包, 那么就可以直接通过 java -jar xxx.jar 运行 jar 包。

这种方式生成 jar 包有个缺点, 就是生成的 jar 包太多不便于管理, 下面两种方式只生成一个 jar 文件, 包含项目本身的代码、资源以及所有的依赖包。

将打包成的 jar 包解压后:



## 方法二：使用 maven-assembly-plugin 插件打包

在 pom.xml 中配置：

```
<build>

  <plugins>

    <plugin>

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

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

      <version>2.5.5</version>

      <configuration>

        <archive>

          <manifest>

            <mainClass>com.wqc.main.SpringStart</mainClass>

          </manifest>

        </archive>

      </configuration>

    </plugin>

  </plugins>

</build>
```

```

        <descriptorRefs>

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

        </descriptorRefs>

    </configuration>

</plugin>

</plugins>

</build>

```

打包方式：

这种必需要加上 `assembly:single`，只会生成一个 `mavenpackage-1.0.0.jar`，且运行时会报错：jar 中没有主清单属性

```
mvn package assembly:single
```

•

1

打包后会在 `target` 目录下生成一个 `xxx-jar-with-dependencies.jar` 文件，这个文件不但包含了自己项目中的代码和资源，还包含了所有依赖包的内容。所以可以直接通过 `java -jar` 来运行。

此外还可以直接通过 `mvn package` 来打包，无需 `assembly:single`，不过需要加上一些配置：

```

<build>

    <plugins>

        <plugin>

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

```

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

<version>2.5.5</version>

<configuration>

  <archive>

    <manifest>

      <mainClass>com.wqc.main.SpringStart</mainClass>

    </manifest>

  </archive>

  <descriptorRefs>

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

  </descriptorRefs>

</configuration>

<executions>

  <execution>

    <id>make-assembly</id>

    <phase>package</phase>

    <goals>

      <goal>single</goal>

    </goals>

  </execution>

</executions>

</plugin>
```



```
</plugins>
```

```
</build>
```

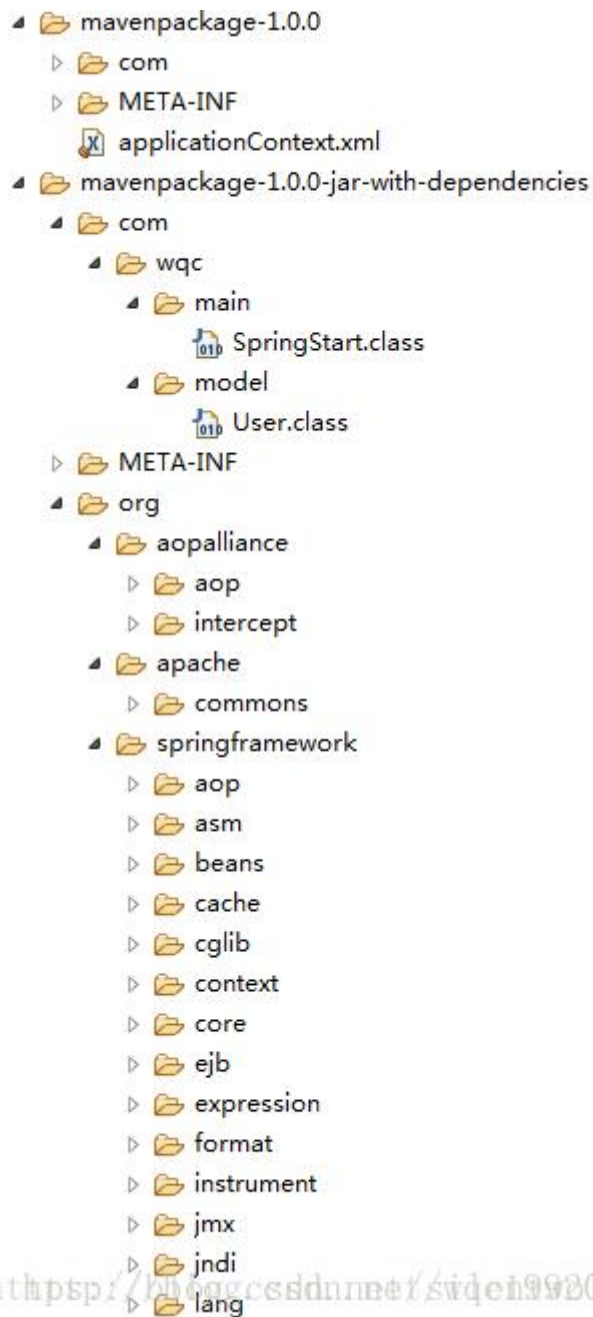
其中 package、single 即表示在执行 package 打包时，执行 assembly:single，所以可以直接使用 mvn package 打包。

不过，如果项目中用到 spring Framework，用这种方式打出来的包运行时可能会出错，但实验中没有出现这种情况。使用下面的方法三可以处理。

执行 maven install 后的 target 文件目录：

名称	修改日期	文件类型
archive-tmp	2018/2/5 10:59	文件夹
classes	2018/2/5 10:59	文件夹
maven-archiver	2018/2/5 10:59	文件夹
maven-status	2018/2/5 10:59	文件夹
test-classes	2018/2/5 10:59	文件夹
mavenpackage-1.0.0.jar	2018/2/5 10:59	Executable Jar File
mavenpackage-1.0.0-jar-with-dependencies.jar	2018/2/5 10:59	Executable Jar File

将打包好的 jar 包解压后：



### 方法三：使用 maven-shade-plugin 插件打包

在 pom.xml 中配置：

```
<build>
```

```
<plugins>
```

```

<plugin>

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

  <artifactId>maven-shade-plugin</artifactId>

  <version>2.4.1</version>

  <executions>

    <execution>

      <phase>package</phase>

      <goals>

        <goal>shade</goal>

      </goals>

      <configuration>

        <transformers>

          <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

            <mainClass>com.wqc.main.SpringStart</mainClass>

          </transformer>

        </transformers>

      </configuration>

    </execution>

  </executions>

</plugin>








</plugins>

```

```
</build>
```

配置完成后，执行 `mvn package` 即可打包。在 `target` 目录下会生成两个 jar 包，注意不是 `original-xxx.jar` 文件，而是另外一个。和 `maven-assembly-plugin` 一样，生成的 jar 文件包含了所有依赖，所以可以直接运行。

执行 `maven install` 后 `target` 文件夹如下所示：

	classes	2018/2/5 11:04	文件夹	
	maven-archiver	2018/2/5 11:04	文件夹	
	maven-status	2018/2/5 11:04	文件夹	
	test-classes	2018/2/5 11:04	文件夹	
	mavenpackage-1.0.0.jar	2018/2/5 11:04	Executable Jar File	3,6
	mavenpackage-1.0.0-shaded.jar	2018/2/5 11:04	Executable Jar File	3,6
	original-mavenpackage-1.0.0.jar	2018/2/5 11:04	Executable Jar File	

在 `cmd` 中可以直接 `java -jar mavenpackage-1.0.0.jar` 即可运行，执行 `java -jar mavenpackage-1.0.0-shaded.jar` 也能执行。`original-mavenpackage-1.0.0.jar` 只是对源码的打包，只有本身的 `class` 文件，不包含依赖的 jar 的 `class` 文件。

将生成的三个 jar 文件解压：

- └─ mavenpackage-1.0.0
  - └─ com
  - └─ META-INF
  - └─ org
  - └─ applicationContext.xml
- └─ mavenpackage-1.0.0-shaded
  - └─ com
  - └─ META-INF
  - └─ org
  - └─ applicationContext.xml
- └─ original-mavenpackage-1.0.0
  - └─ com
  - └─ META-INF
  - └─ applicationContext.xml

如果项目中用到了 Spring Framework,将依赖打到一个 jar 包中,且运行时会出现读取 XML schema 文件出错。原因是 Spring Framework 的多个 jar 包中包含相同的文件 spring.handlers 和 spring.schemas, 如果生成一个 jar 包会互相覆盖。为了避免互相影响, 可以使用 AppendingTransformer 来对文件内容追加合并:

```
<build>

  <plugins>

    <plugin>

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

      <artifactId>maven-shade-plugin</artifactId>

      <version>2.4.1</version>

      <executions>

        <execution>

          <phase>package</phase>

          <goals>

            <goal>shade</goal>

          </goals>

          <configuration>

            <transformers>

              <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

                <mainClass>com.xxg.Main</mainClass>

              </transformer>

            </transformers>

          </configuration>

        </execution>

      </plugin>

    </plugins>

  </build>
```

```

        </transformer>

        <transformer
implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">

            <resource>META-INF/spring.handlers</resource>

        </transformer>

        <transformer
implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">

            <resource>META-INF/spring.schemas</resource>

        </transformer>

    </transformers>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

```

这样生成的 jar 包就只有两个了，mavenpackage-1.0.0-shaded.jar 合并到了 mavenpackage-1.0.0.jar 中去了