链滴

SpringCloud+Maven 管理多模块项目遇到的一些问题

作者：bigbear

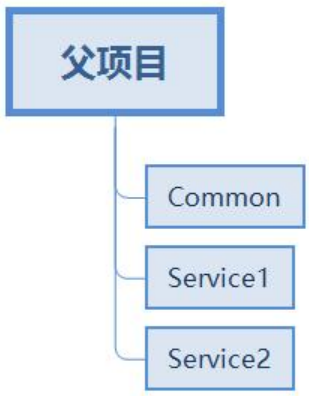
原文链接：https://ld246.com/article/1593485569481

来源网站：链滴

环境

- SpringCloud Greenwich.SR2
- Maven3.0.5
- Idea2019.3

项目结构



Service1 依赖 Common

Service2 依赖 Common

Common 无任何依赖

遇到了什么问题?

mvn package 父项目或 Service 都会报错，也就是说在 maven 多模块结构的项目中，引用了一个有程序入口的项目将会打包失败

问题分析 + 解决

下面我对两种无法通过打包的情况进行分析，并给出解决方案。

Service 项目无法打包

[ERROR] Failed to execute goal on project Service1: Could not resolve dependencies for proje t org.ccccye:Service:jar:0.0.1-SNAPSHOT: Could not find artifact org.ccccye:common:jar:0.0.1-NAPSHOT -> [Help 1]

从异常信息中可以知道两件事

1. Maven 执行 goal 失败
2. goal 执行失败是因为 Service1 项目所依赖的 common.jar 包找不到

那么问题根源应该是 Service1 项目找不到 common.jar 包造成的。通过翻阅资料和实验，发现当项单独打包时，不会主动去编译所依赖的模块项目，而是去本地仓库找 jar 包，如果用 maven install 把 common.jar 安装到本地仓库，是不是能解决问题呢？答案是可以的，在 common 项目执行 mvn install 就可以将 common.jar 安装到本地仓库了，，，等等，这里还会出现另外一个问题，接下来继续。

父项目无法打包

```
[ERROR] Failed to execute goal org.springframework.boot:spring-boot-maven-plugin:2.1.7.RELEASE:repackage (repackage) on project weather-common: Execution repackage of goal org.springframework.boot:spring-boot-maven-plugin:2.1.7.RELEASE:repackage failed: Unable to find main class -> [Help 1]
```

从异常信息中可以知道两件事

1. 同样是 Maven 的 goal 执行失败了，并且明确是 repackge 功能，这说明在将 **普通 jar** 重新打成**可执行 jar** 的过程中发生了错误
2. 异常信息中还说了报错原因是找不到 main 函数造成的

因此问题在 main 函数报错上，common 项目根本没有 main 函数并且不需要，这是难为我胖虎啊？翻了资料发现，maven 生成 jar 包的顺序是 先生成普通 jar 包，再包装成可执行 jar（普通 jar 会覆盖），**注意：**可执行程序肯定要有 main 函数的，那么我们可以通过设置 spring-boot-maven plugin 插件的 layout 参数，赋值为 NONE，maven 只打包所需的依赖，不扫描 main 函数

```
<plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
         <layout>NONE</layout>  <!--让maven不打包可执行jar，不扫描项目的main函数-->
         <classifier>exec</classifier> <!--普通jar和可执行jar不同名，普通jar为xx.jar ， 可执行jar xx-exec.jar-->
      </configuration>
</plugin>
```

如上，配置了 layout=NONE 解决了 main 函数问题，在实验过程中发现，Service1 能正常编译，运行就报异常，提示 **找不到或无法加载主类** ，原来 Service 想要的 jar 是一个纯粹的普通 jar 包，然设置了 layout=NONE，但 maven 还是会再打包一次，并且会覆盖普通的 jar，让 Service 虽然加了 jar 包却找不到 jar 包里面所包含的类，在这里我的解决方案是配置普通 jar 跟二次打包的 jar 不同，也就是配置

```
<classifier>exec</classifier>
```

到此，问题的原因和解决方案都有了，这里总结出来的知识点有

1. 子项目 package 时的依赖是去本地仓库或者远程仓库查找的，不会主动编译本地项目并依赖
2. Maven 默认会执行 repackge 这个 goal
3. repackge 将普通 jar 改成 xx.jar.original，可执行 jar 为 xx.jar，可以通过 classifier 参数来修改执行 jar 的名字后缀
4. 可执行 jar 默认寻找一个 main 函数，通过配置 layout=none 取消

第二种解决方案

还有一种解决方案，在主项目 pom 更换插件，将 spring-boot-maven-plugin 改为 maven-compile-plugin，具体的配置为

```
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.1</version>
<configuration>
    <source>${java.version}</source>
    <target>${java.version}</target>
</configuration>
```

然后在主项目运行 package 命令就可以整体编译打包了。不过要打包 Service 子项目的话，Common 项目得手动 install 安装到本地仓库才行。

关联的知识

- Maven 的插件模型

总结

主 pom	打包方式	service pom	ommon pom
spring-boot-maven-plugin	整体打包	spring-boot-maven-plugin 增加配置：1. layout=none 2 classifier=exec	
	部分打包	spring-boot-maven-plugin 增加配置：1. layout=none 2.classifier=exec。打包前需要 install	
maven-compiler-plugin	整体打包	无 build 节点	spring-boot-maven-plugin
	部分打包	build 节点，打包前需要 install	

可以看到部分打包都需要提前 install 公共项目，否则 package 过程中找不到 jar 包；使用 maven-compiler-plugin 不需要增加配置。

值得一说的是这两个插件并不是同一个类型的，只是刚好能解决问题，它们的区别如下

- spring-boot-maven-plugin 是打包插件
- maven-compiler-plugin 是编译插件

这也就不难理解，为什么 spring-boot-maven-plugin 需要增加配置了，因为它不负责编译，只能编译出来的东西进行打包，因此不能对其他项目进行编译然后打包到 jar 里；maven-compiler-plugin 可以进行编译，在编译阶段能将所依赖的其他项目按顺序编译，最后打包进 jar