**自动打jar上传小工具**

**适用人群：**

**经常要给其它项目打jar包的开发**

目录

[需求背景 2](#_Toc518638564)

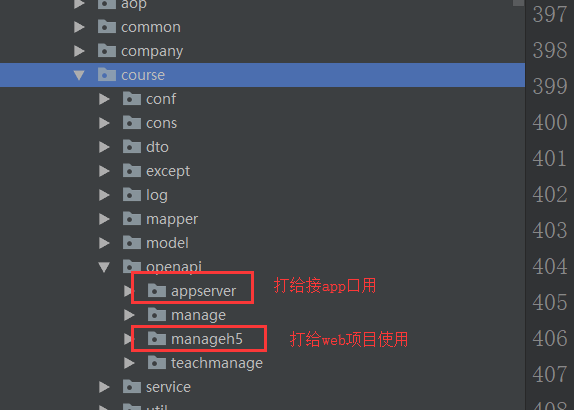
[环境依赖（限windows）： 7](#_Toc518638565)

[使用步骤 8](#_Toc518638566)

[conf.json 配置 10](#_Toc518638567)

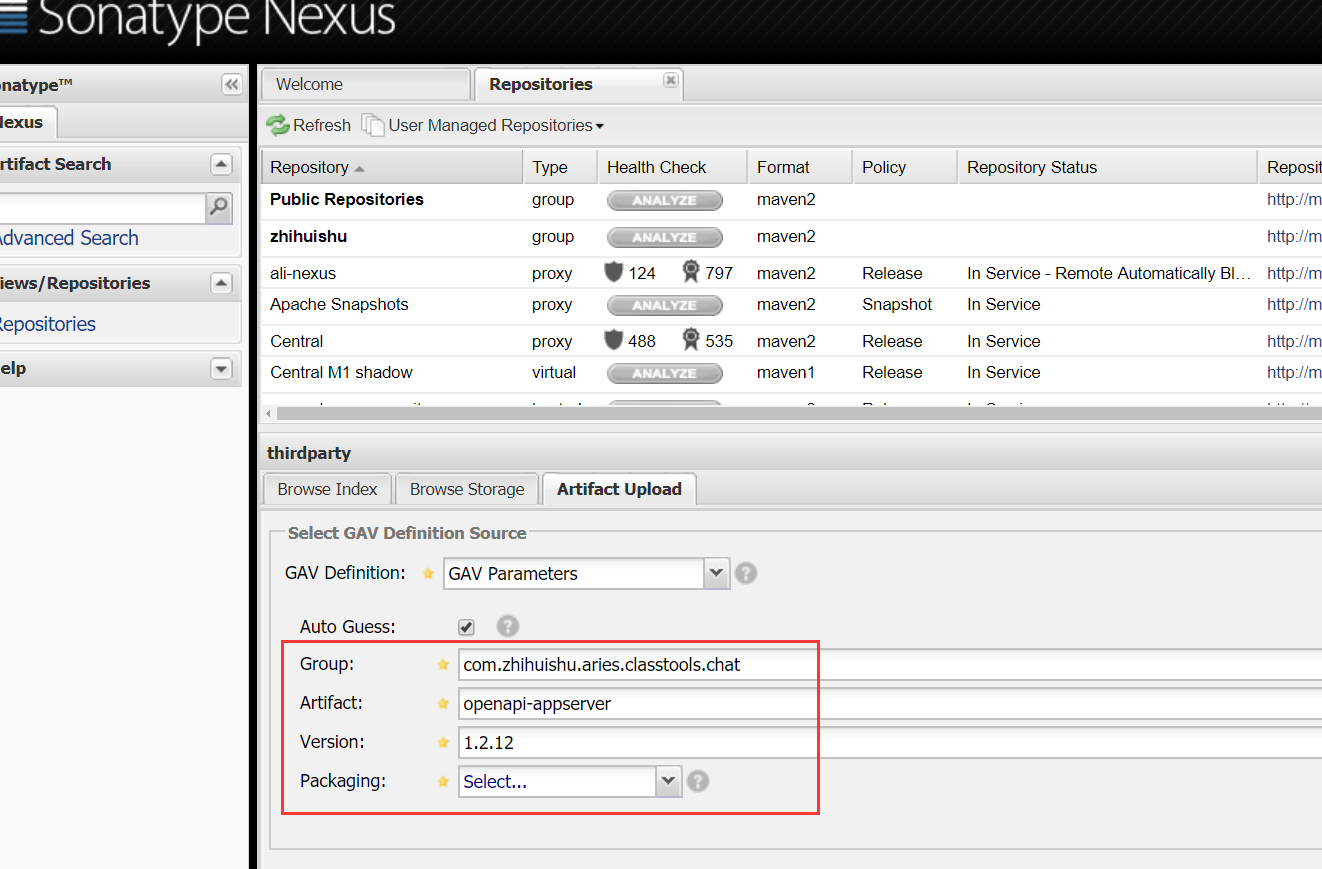
[其它说明 17](#_Toc518638568)

# 需求背景

由于微服务架构的盛行，使用dubbo的同学经常需要打出jar文件给调用方使用。

像上图一样经常出现一个项目要给不同的项目打jar包的场景

打了jar包还不够，还要上传到maven库



上传又不可避免地要填上面这些参数，这个操作非常重复，效率也非常低，也很容易出错。

传完还有最后一步，把依赖坐标发给调用方

<dependency>

<groupId>com.zhihuishu.aries.classtools.chat</groupId>

<artifactId>openapi-appserver</artifactId>

<version>1.2.12</version>

</dependency>

整个操作执行下来少说也有3分钟，遇上接口变动频繁，这个3分钟就会成倍地增长，对业务研发者来说无疑是一种肉体+精神+智商的摧残。

因此，急需一样工具能自动搞定这件事情，它要能做到：

1. 自动打出需要的jar包，如jar包内包含哪些class文件，是否打入java源码
2. 自动传到maven服务器自己配置的坐标内，坐标中的version要根据服务器最新版本号自动计算出下一个版本
3. 生成最后能发给调用方的依赖坐标，因为自己懒得去拼出来

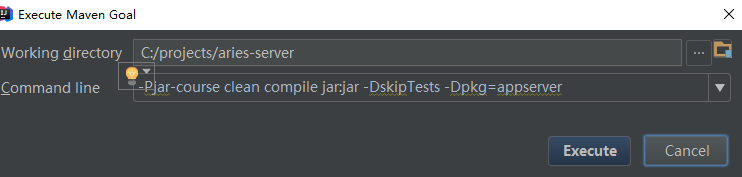
期望经过初期的配置之后一劳永逸，双击脚本一键搞定

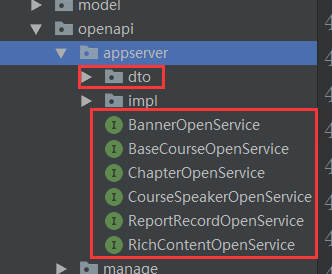
第一点的实现依赖于pom自带的打jar功能，可以看下pom.xml中的配置

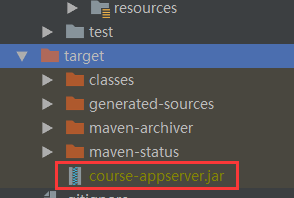
<profile>  
 <id>jar-course</id>  
 <build>  
 <resources>  
 <resource>  
 <directory>src/main/java</directory>  
 <includes>  
 <include>\*\*/\*.java</include>  
 </includes>  
 </resource>  
 </resources>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <version>2.4</version>  
 <configuration>  
 <finalName>course-${pkg}</finalName>  
 <includes>  
 <include>com/zhihuishu/aries/course/openapi/${pkg}/\*</include>  
 <include>com/zhihuishu/aries/course/openapi/${pkg}/dto/\*</include>  
 </includes>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</profile>

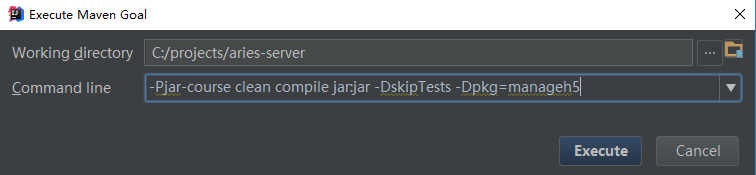
在配置了这样一个profile之后

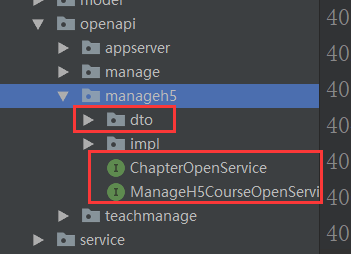
当我执行

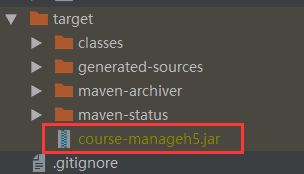


会打入这些class

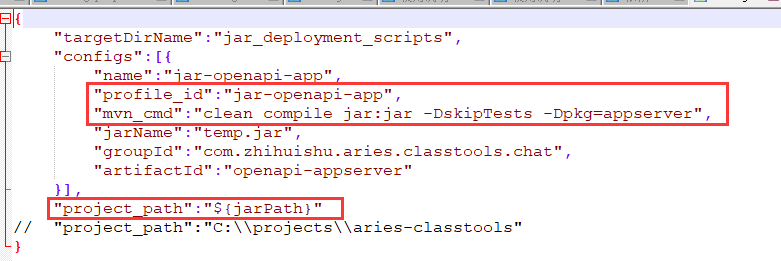
target目录下生成

当我执行

会打入这些class

target目录下生成

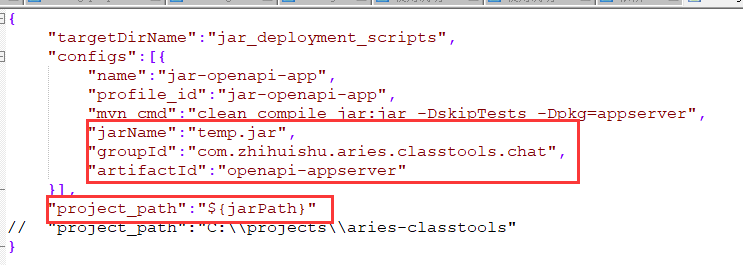
第一个要求（生成jar文件）算是通过maven的插件功能解决了

本工具通过下面几项配置保证第一步能按自己的需要执行

project\_path配了后脚本才能cd到这个路径下进行打包操作

第二步的上传，第三步的生成依赖文本这些没有现成解决方案，只能靠自己搭桥

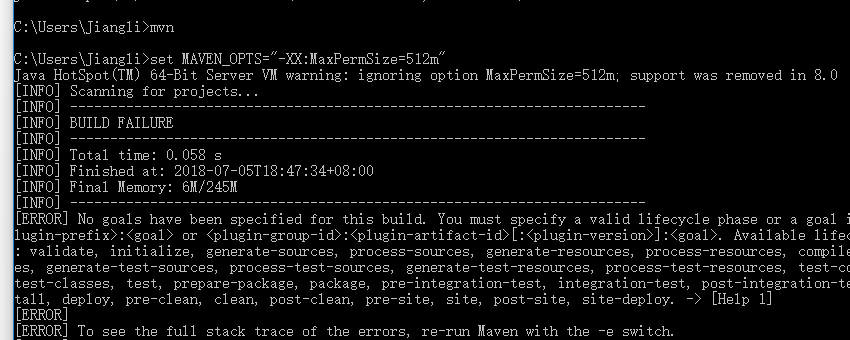
这两步都用到了如下参数

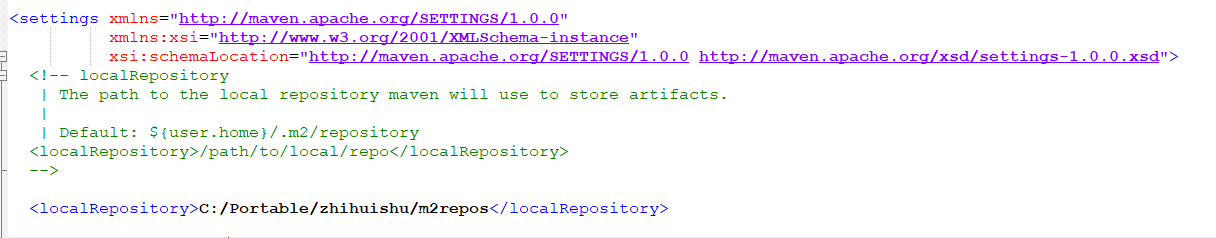
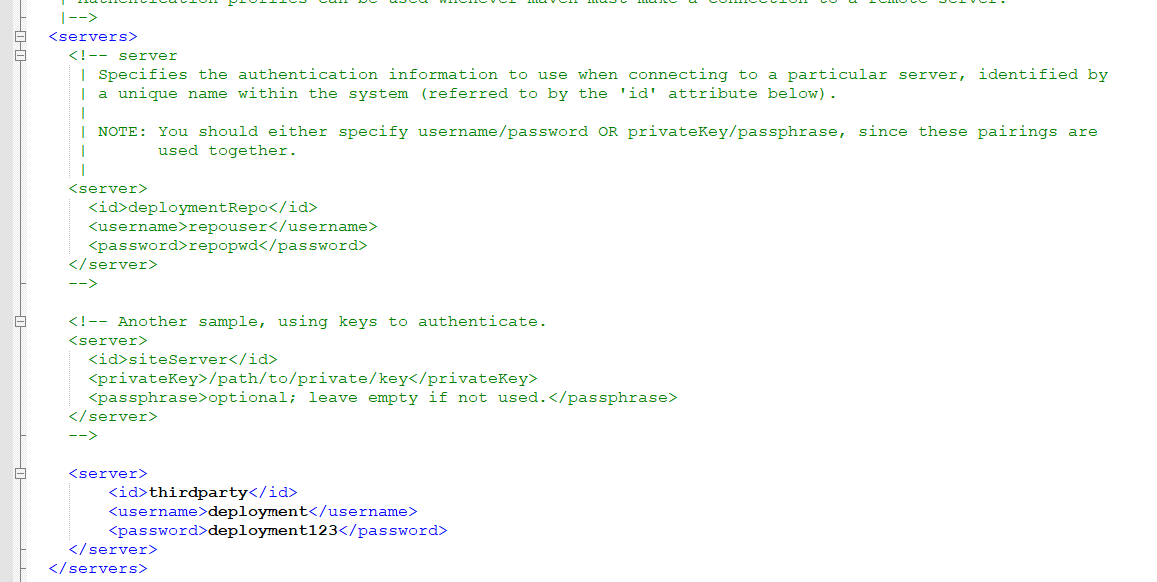


jarName需要指定是因为脚本会从project\_path目录下的target目录拷出jar包出来再上传

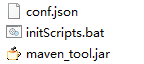
后两个参数会在第二步上传到指定的坐标和第三步生成依赖文本时用到，version也是通过这两参数请求服务器解析出最大版本号，然后计算出来的。

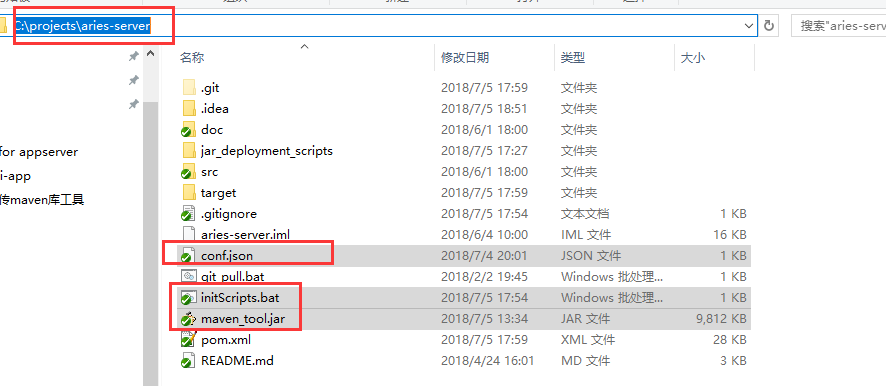
# 环境依赖（限windows）：

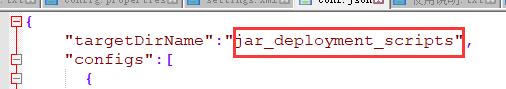
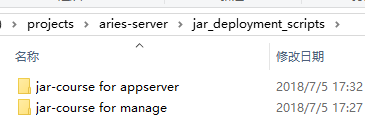
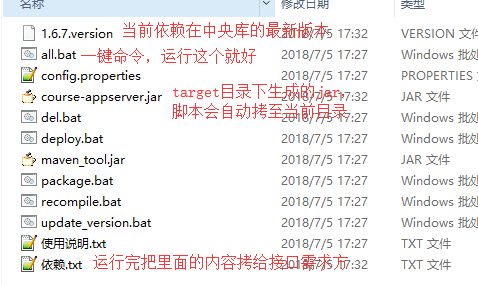
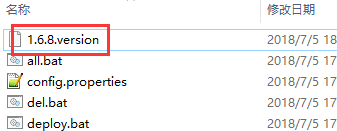
1. 需配置jre环境变量（可执行java -version），且为1.8以上
2. 需配置maven环境变量（可执行mvn）
3. 在%maven\_home%/conf/settings.xml配置了服务端的用户名和密码，如下：

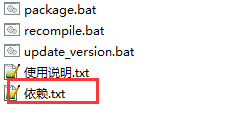


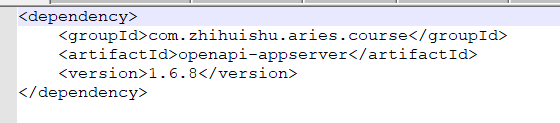
# 使用步骤

1. 将拷到需要执行自动打包的工程根路径下

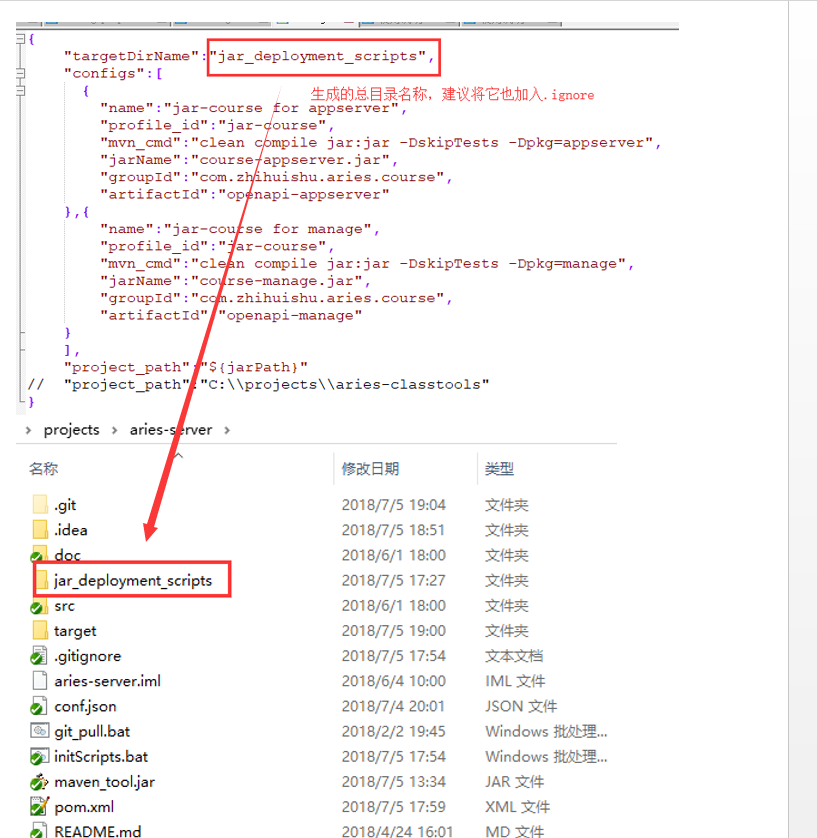
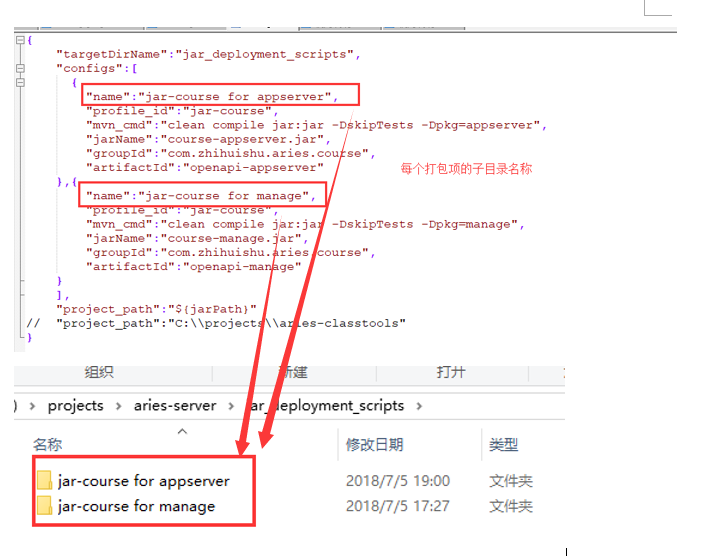
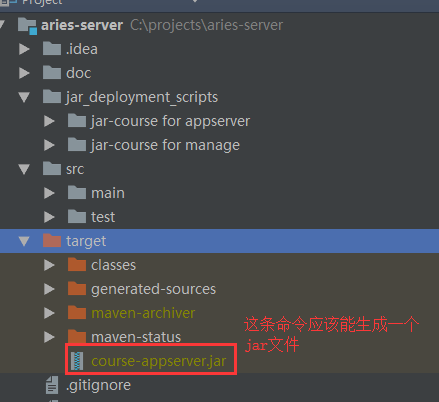
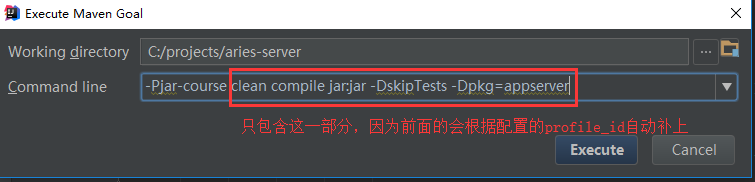
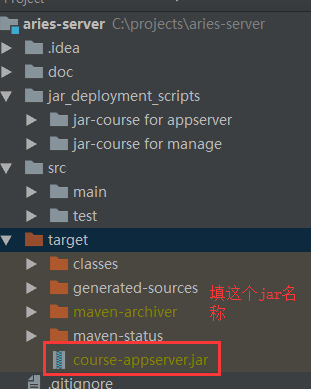
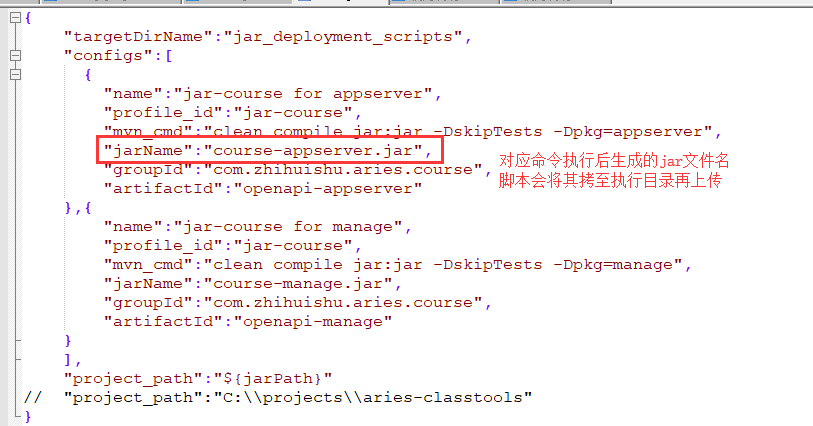
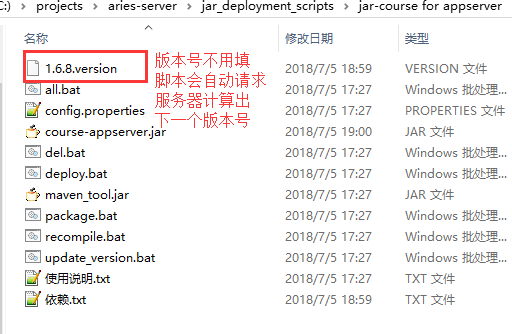
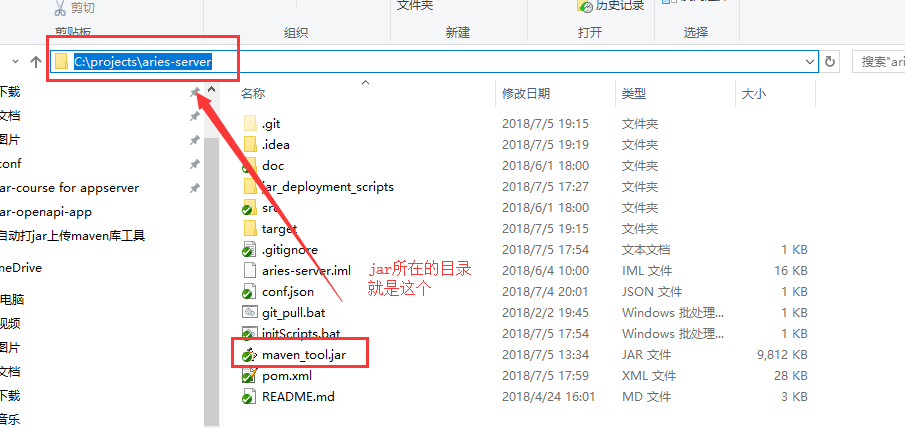


1. 根据需求配置conf.json【见[conf.json 配置](#_conf.json_配置)章节】
2. 执行initScripts.bat 将生成如下配置项指定的脚本根目录
3. 进入目录后可看到不同的打包项
4. 进入自己需要项的文件夹内，可看到
5. 双击运行all.bat 可看到版本通过请求和计算自动得到下一个版本号
6. 当脚本执行完关闭时，依赖已经可用
7. 打开依赖.txt文件将maven坐标拷给接口需求方

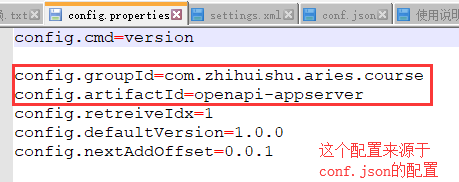


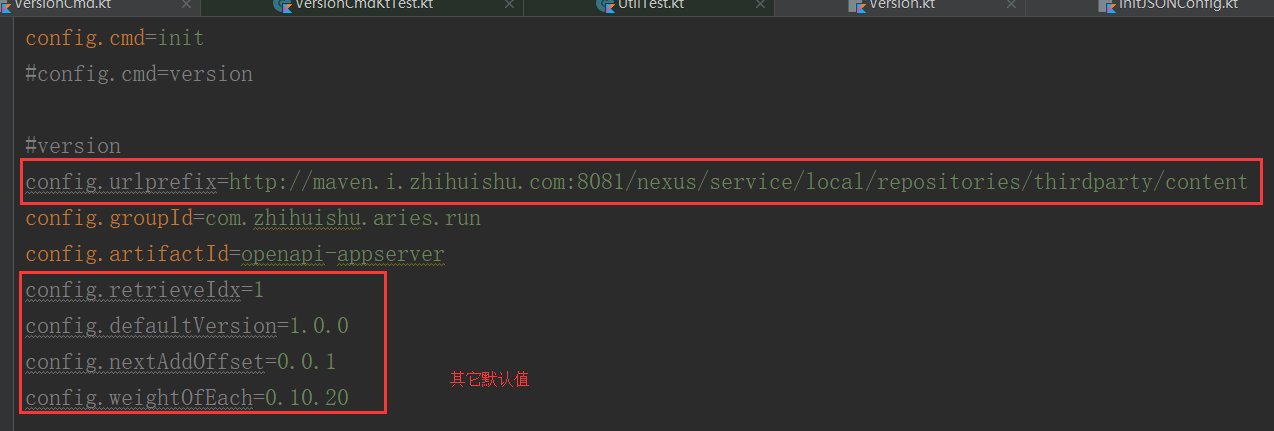


# conf.json 配置

1. 
2. 
3. 
4. 
5. 
6. 
7. 

# 其它说明

1. 关于计算下一个版本号



其中 config.weightOfEach=0.10.20

代表每一位满多少进1

按上面的配置

1.2.18的下一个版本为1.2.19

1.2.19的下一个版本为1.3.0

1.2.99的下一个版本为1.3.0

1.9.1 的下一个版本为1.9.2

1.9.19 的下一个版本为2.0.0

1.99.199 的下一个版本为2.0.0

10.99.199 的下一个版本为11.0.0