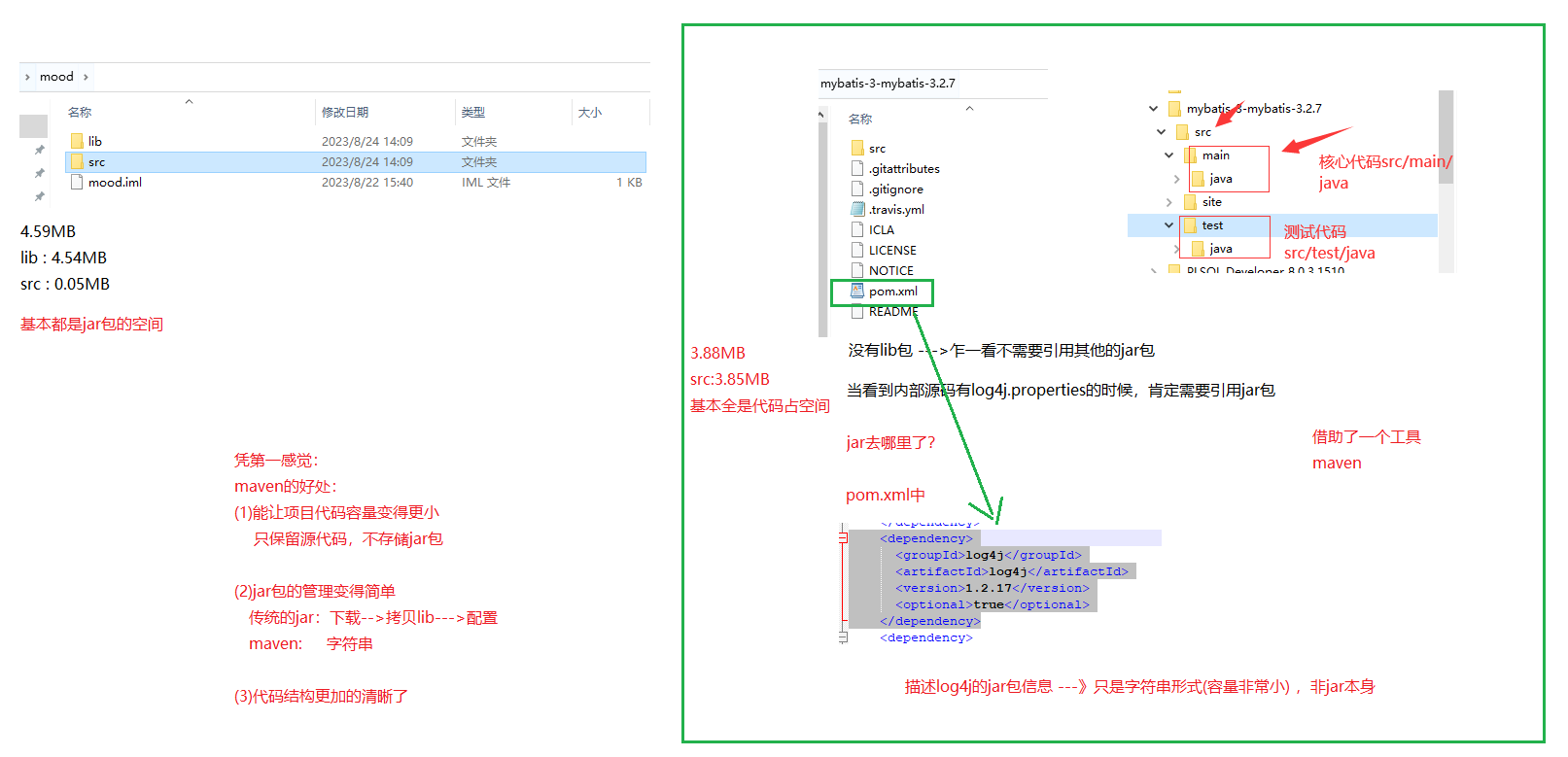
# 1.为什么学Maven

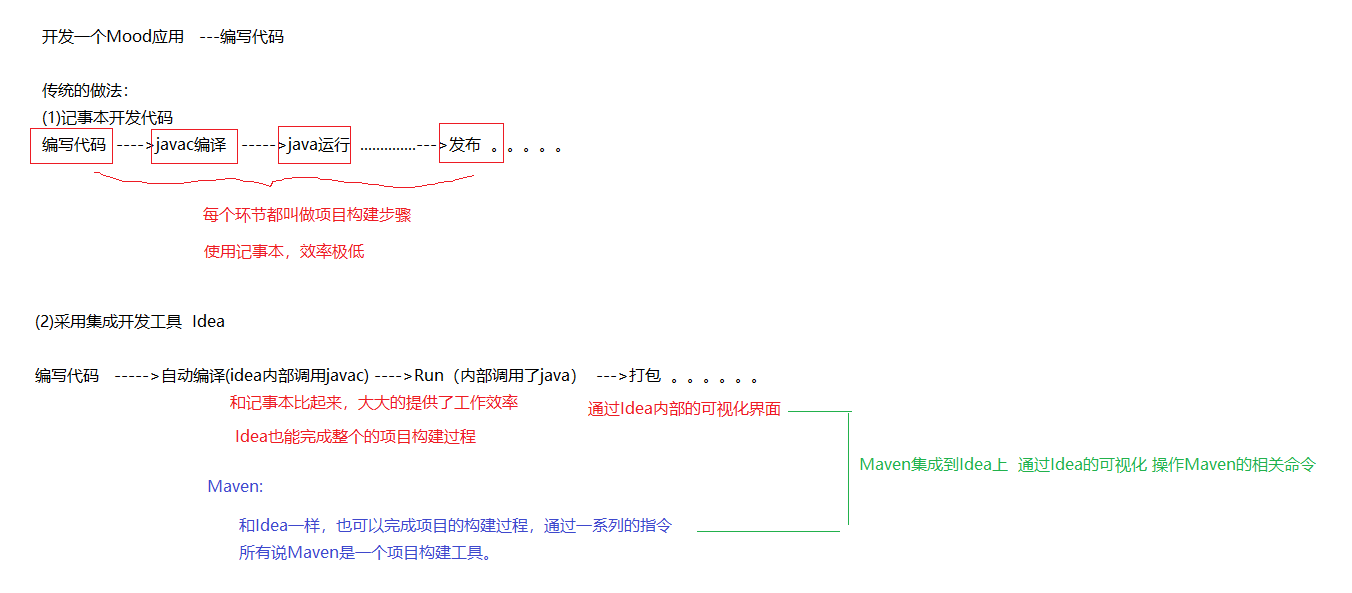


# 2.Maven简介

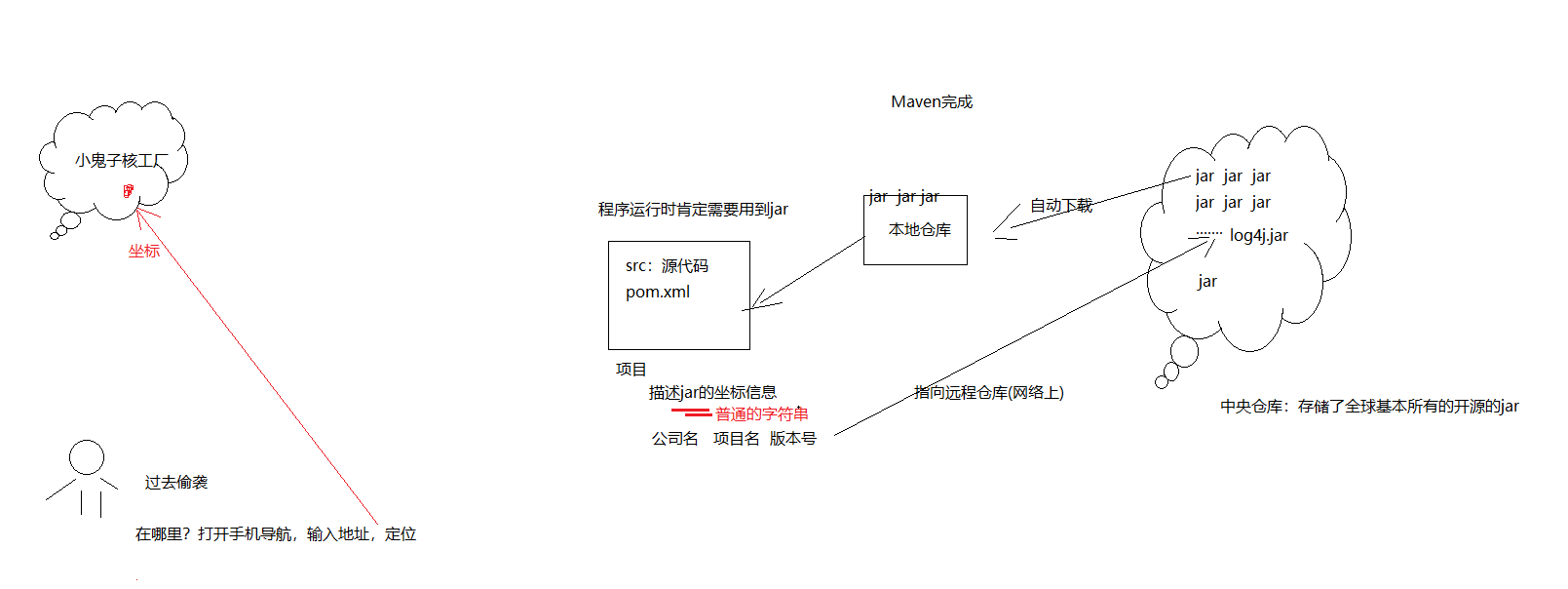
Maven是一个项目构建工具，主要用在团队开发中。

Maven有两个核心的功能

(1)项目的构建



(2)jar包的依赖管理



# 3.Maven的安装

## 3.1 安装JDK

Maven 是一个纯java开发的工具。

省略。。。。

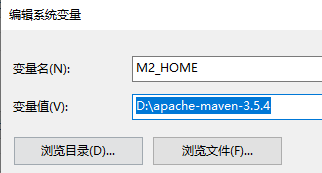
## 3.2 安装Maven

(1)解压文件,放到英文目录下

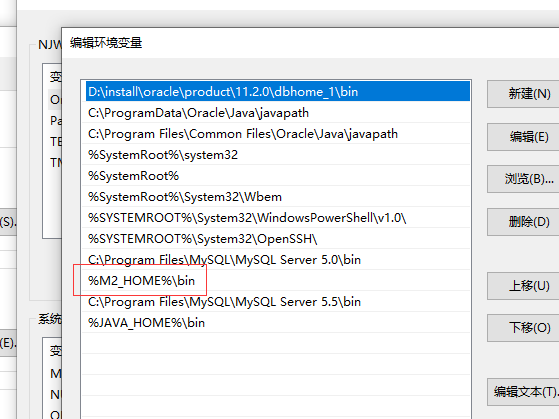


(2)配置maven的环境变量

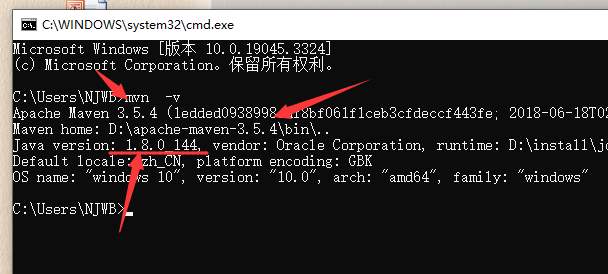
M2\_HOME : maven根目录



M2\_HOME\bin 加到path中

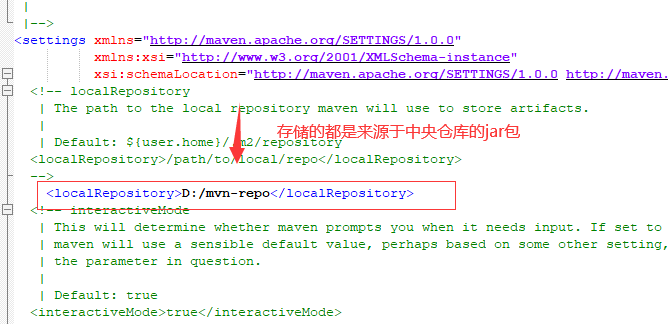


(3)检查maven安装是否成功



(4)配置本地仓库

在conf/settings.xml



(5)配置远程仓库

中央仓库的地址： https://repo.maven.apache.org/maven2/

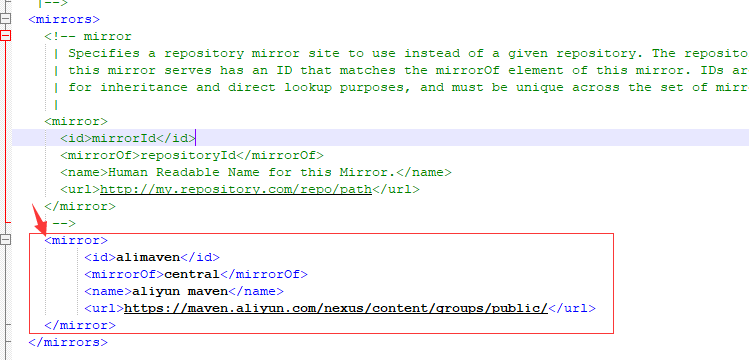


未来下载的jar都在这 ，该仓库服务器在国外，有的时候下载就非常的慢，国内一般都使用镜像地址。

配置镜像地址：

如果maven不配置镜像地址，则默认使用原来的中央仓库地址，

如果配置了镜像，则原来的中央仓库地址失效。



<mirror>

<id>alimaven</id>

<mirrorOf>central</mirrorOf>

<name>aliyun maven</name>

<url>https://maven.aliyun.com/nexus/content/groups/public/</url>

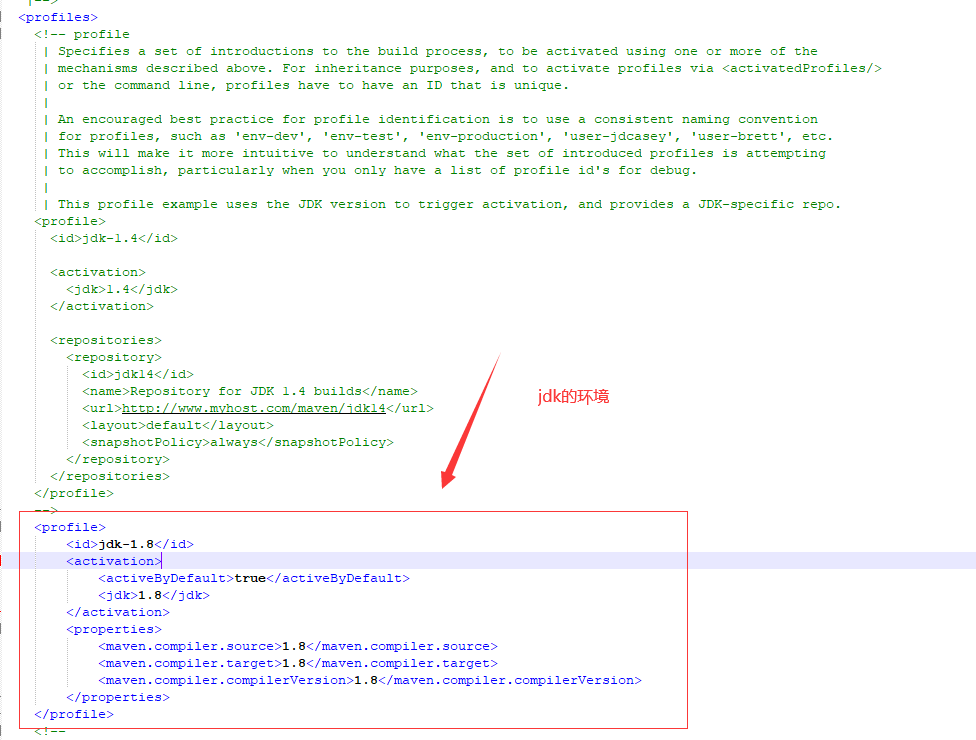
</mirror>

(6)jdk的版本号

maven做项目构建工具，代码的编译也通过maven完成

maven如何完成编译工作的：最终肯定会调用jdk的javac编译，

告诉maven该采用什么版本的jdk编译



<profile>

<id>jdk-1.8</id>

<activation>

<activeByDefault>true</activeByDefault>

<jdk>1.8</jdk>

</activation>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

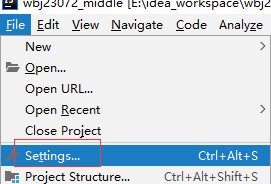
<maven.compiler.target>1.8</maven.compiler.target>

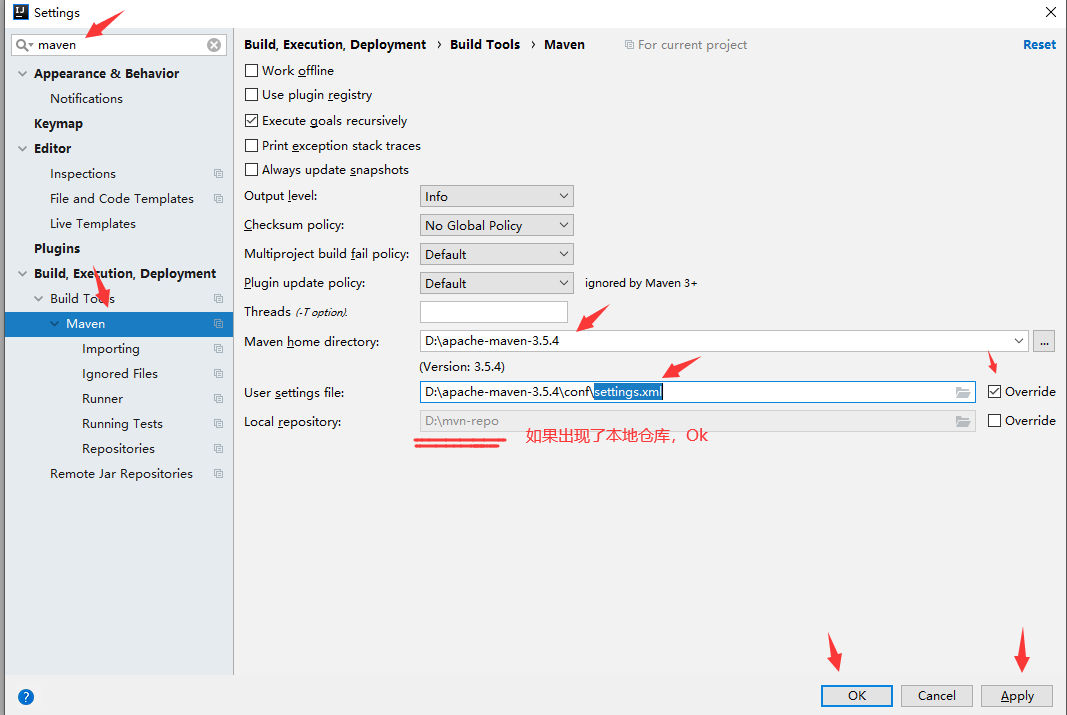
<maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>

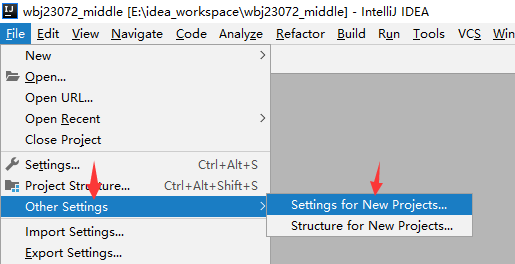
</properties>

</profile>

# 4.和Idea集成

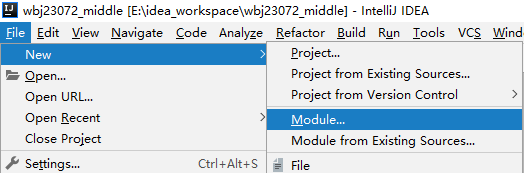


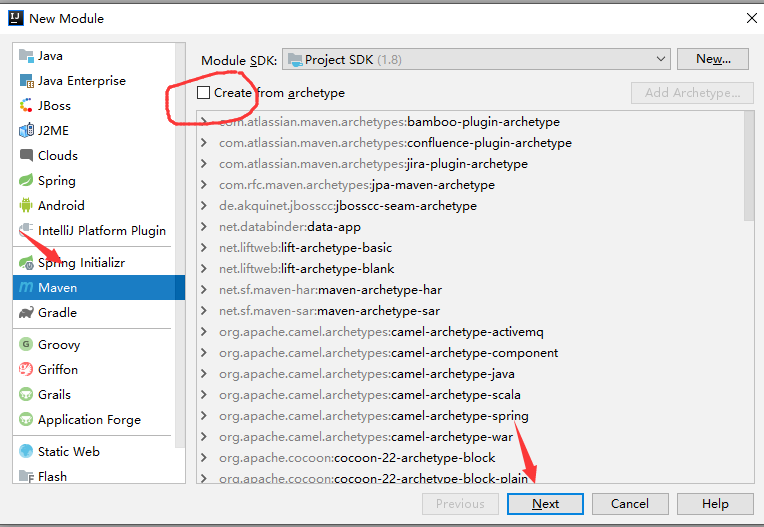


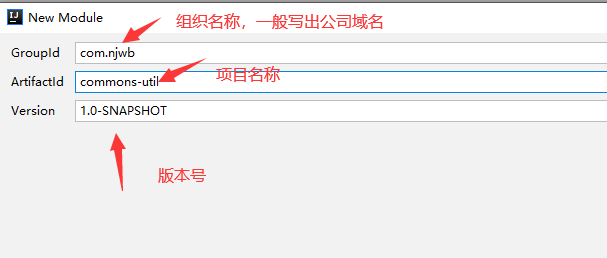


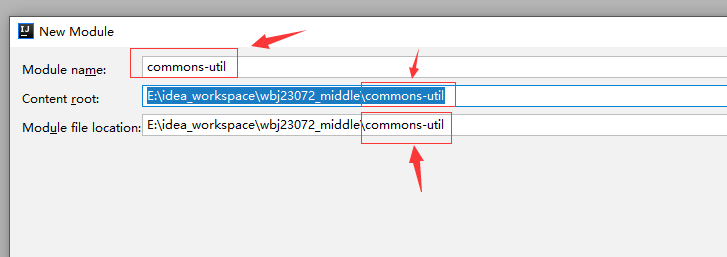
再配置一次maven

# 5.Maven创建工程





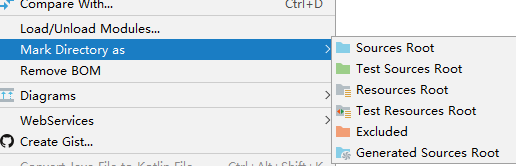




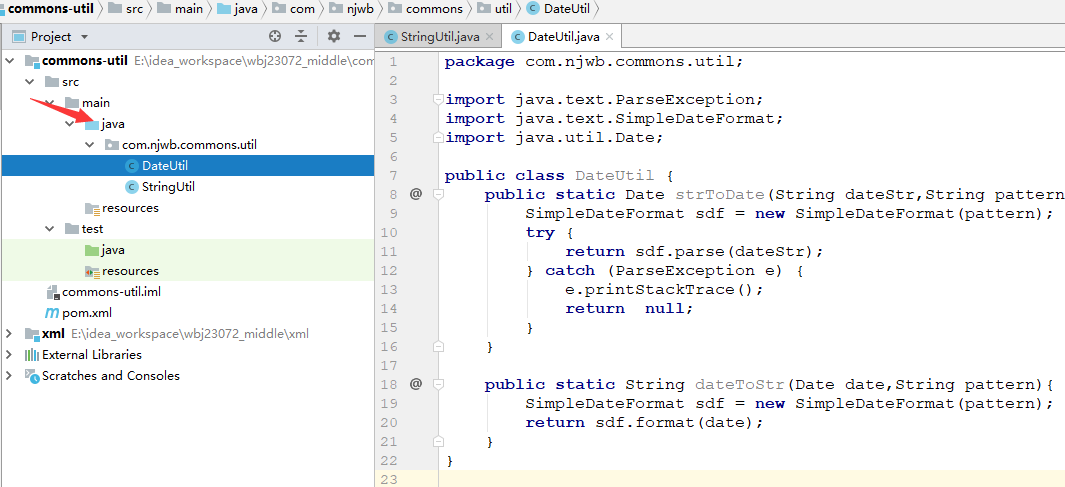


说明：如果main/java main/resources test/java test/resoures颜色不对

右键src，打好标记

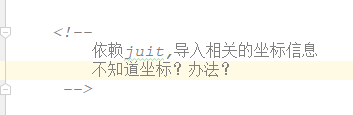


封装两个工具类DateUtil StringUtil

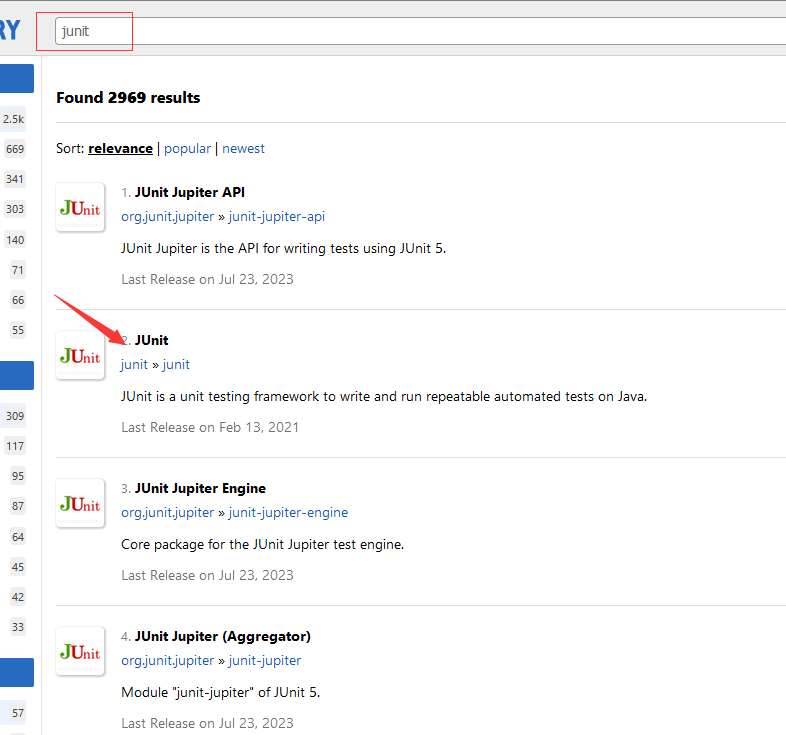


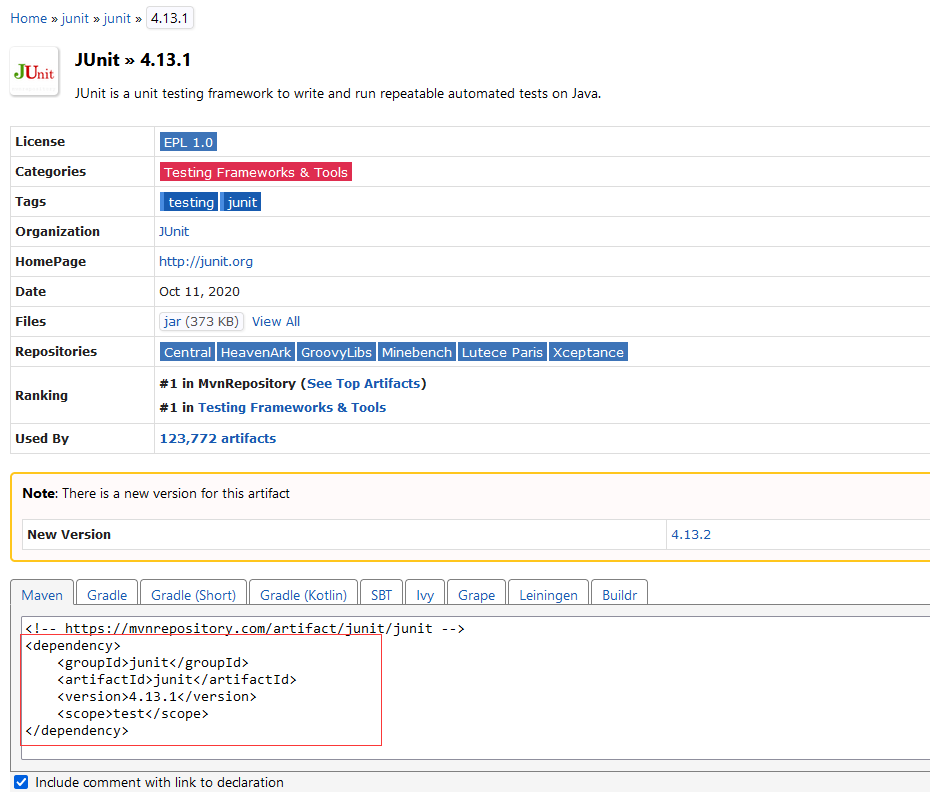
测试，采用juit测试

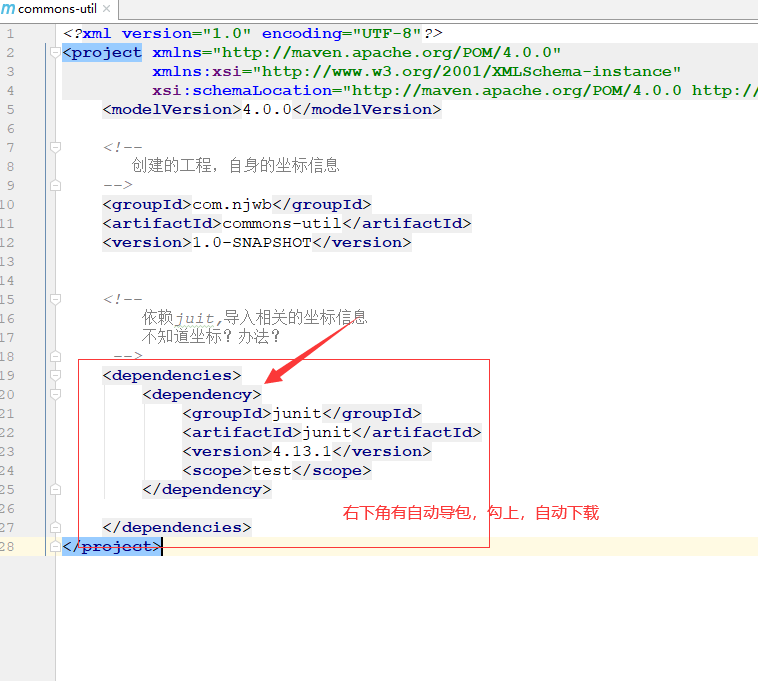
导入juit的jar包 ----》maven来管理 ----》pom.xml中定义juit的坐标信息



**https://mvnrepository.com/**



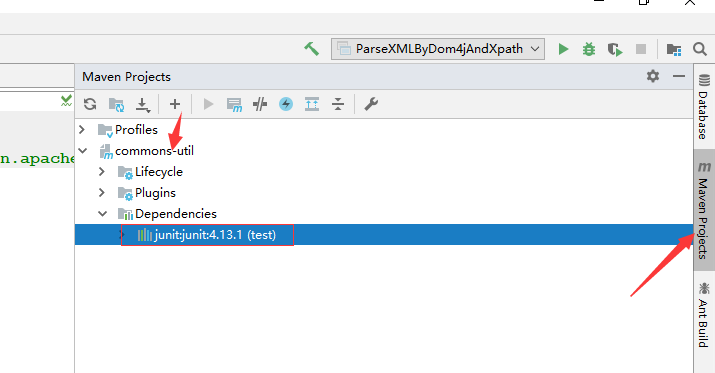




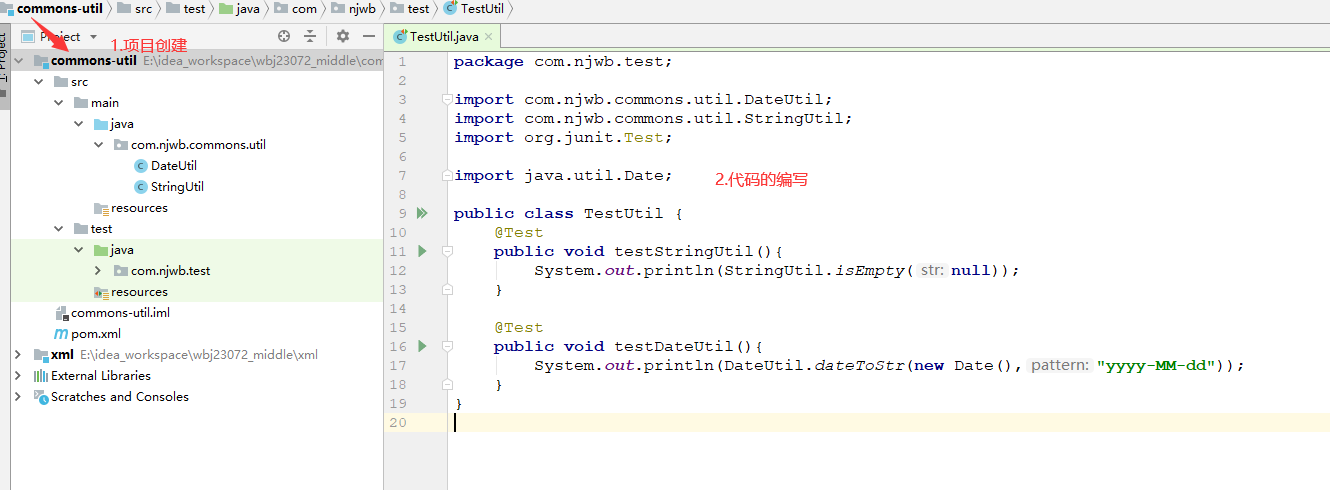
查看本地仓库

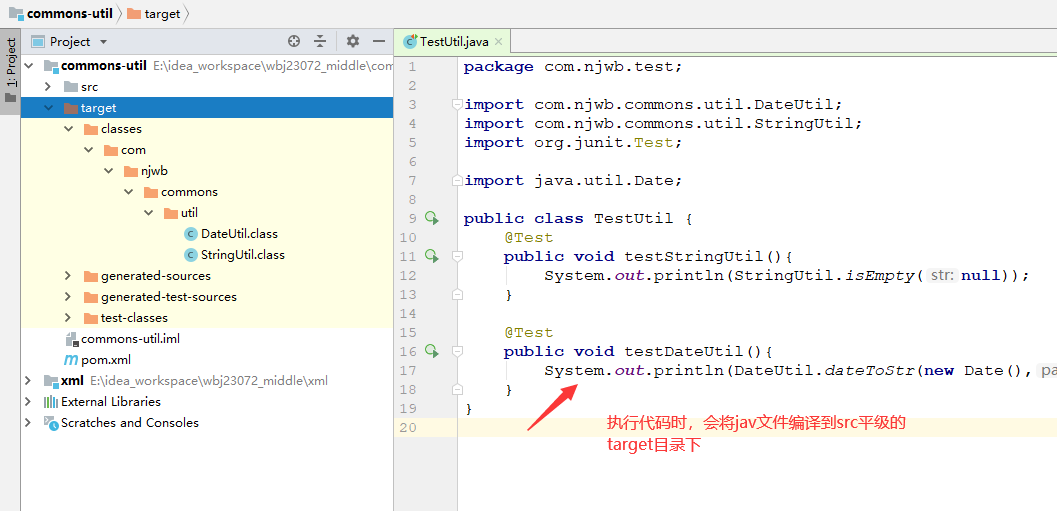
junit/junit/4.13.1/junit-4.13.1.jar

groupId/artifactId/version/artifactId-version.jar

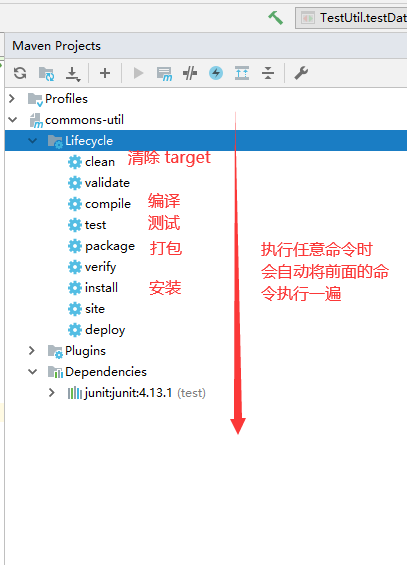


做测试

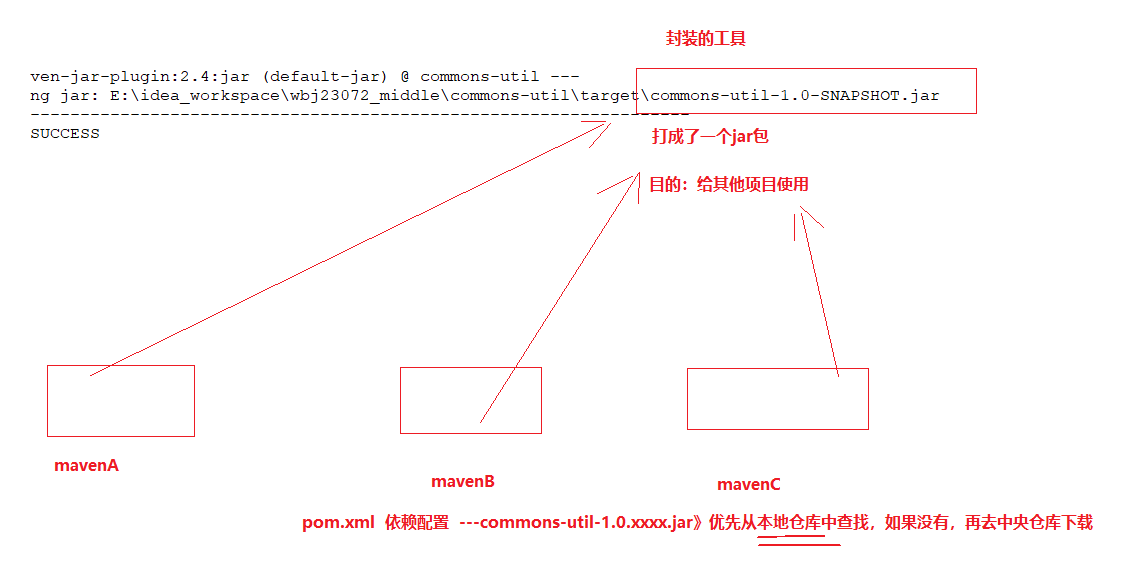




打成jar包



# 6. jar包的安装



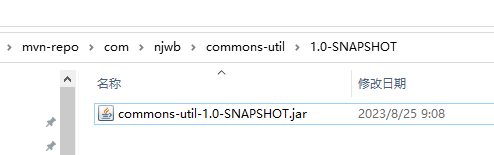


查找仓库--->没有

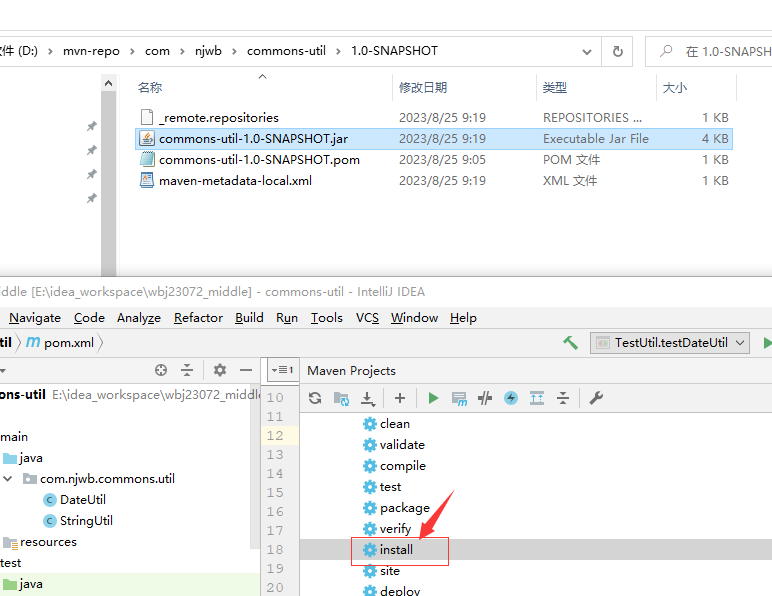
想办法把jar包安装到仓库中

方法一、手动安装

规律 安装 顺序创建目录 groupId artifactId version jar…..



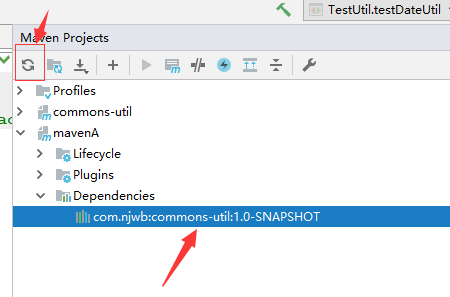
方法二、利用maven的构建功能之install --安装



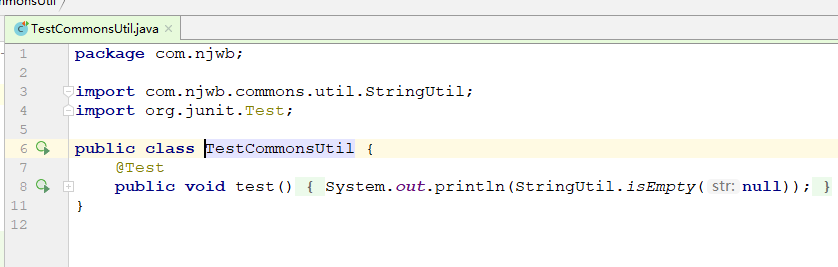
测试安装是否有效：

再创建一个maven工程 ---->pom.xml中引入 jar依赖 ---->测试代码



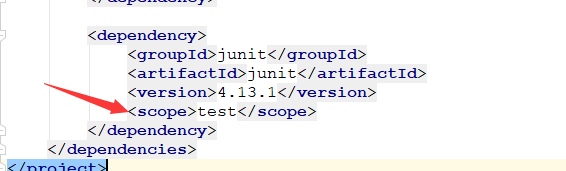


使用包中的类



# 7.依赖范围

前面在引入junit依赖包时，发现



<scope>依赖范围</scope>

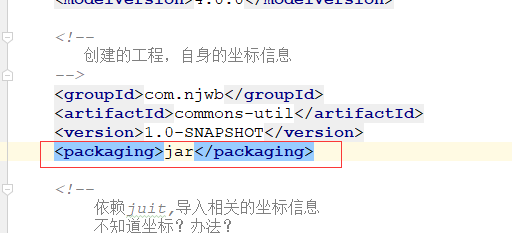


依赖范围默认为compile，

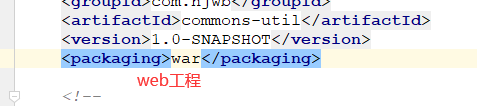
provided:运行部署到tomcat时不需要(一般用到tomcat自带jar包时需要指定)

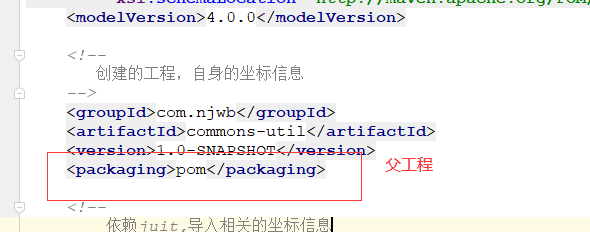
# 8.打包类型

maven默认的打包类型为jar格式

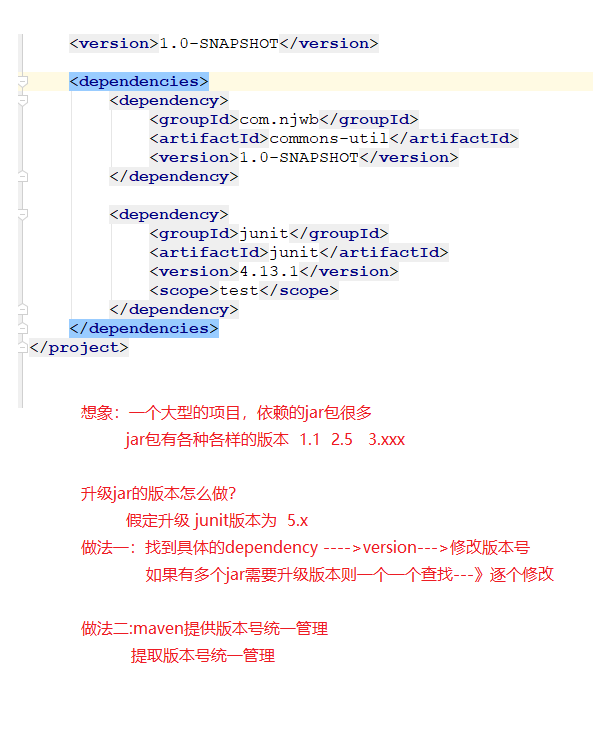


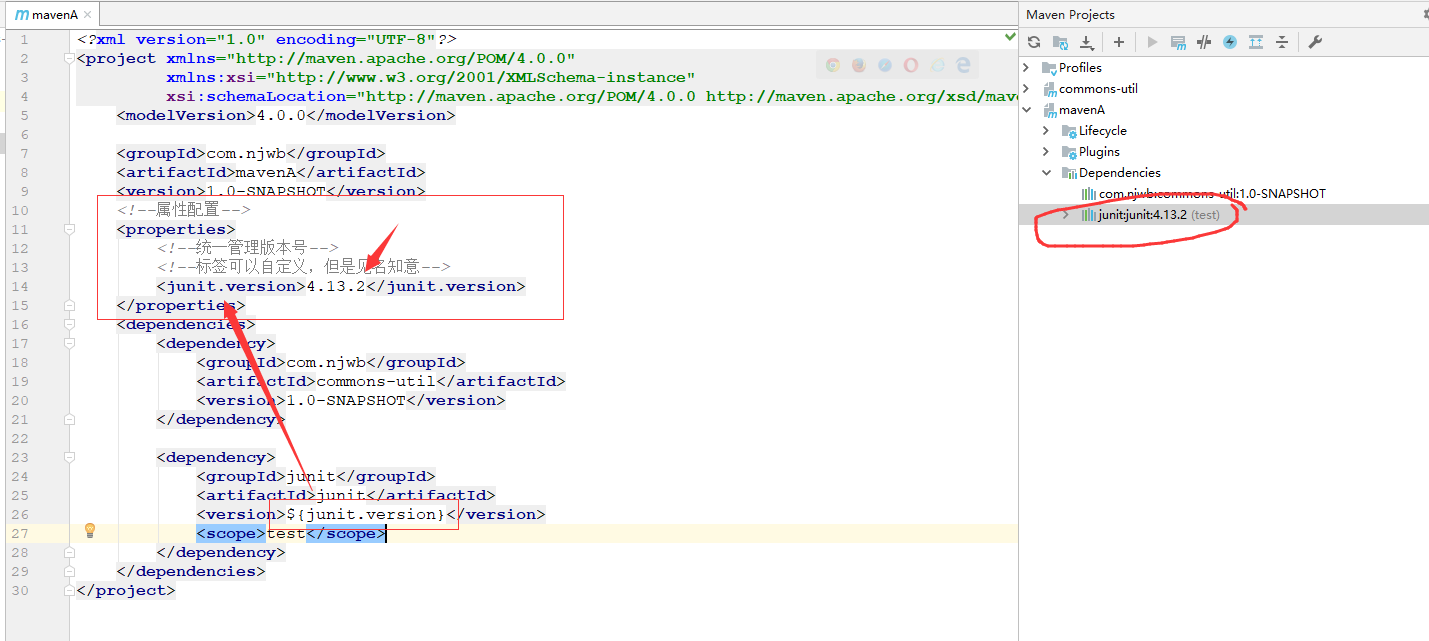
还有其他的打包方式

后期再说

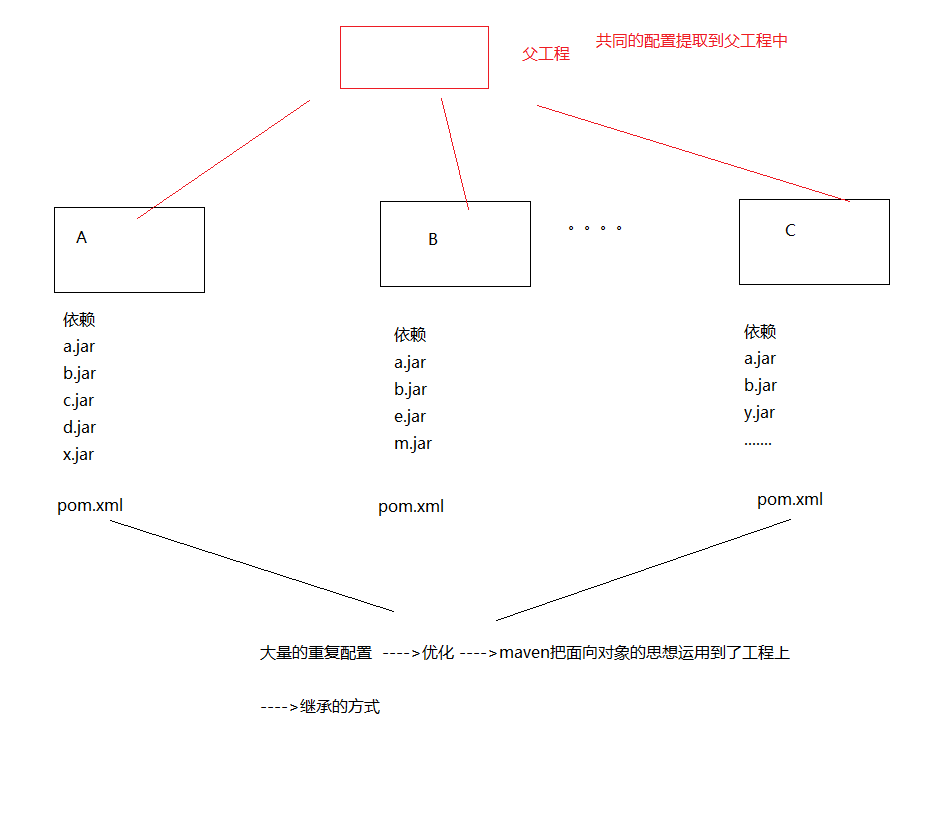


# 9.版本号控制



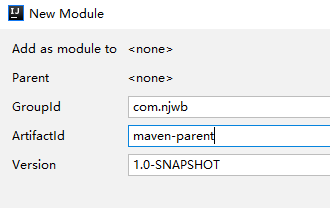


# 10.父工程



实验步骤

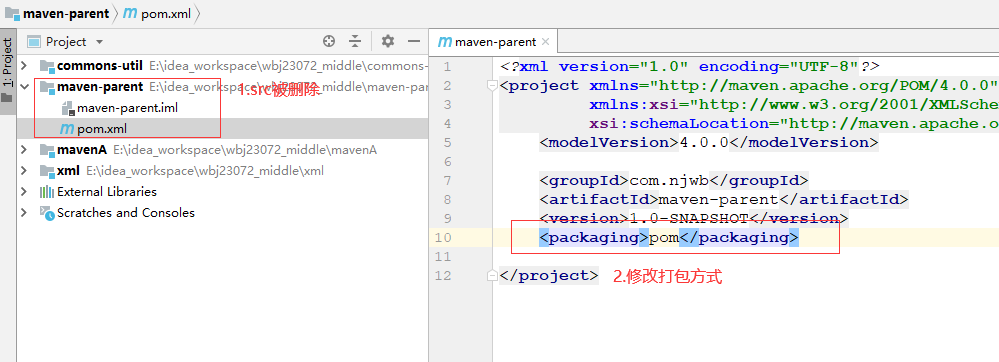
(1)定义父工程



既然叫父工程，满足两个条件

a.删除src

b.pom.xml中打包方式修改为pom方式

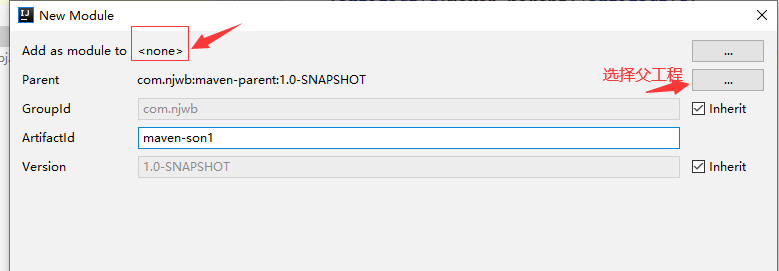


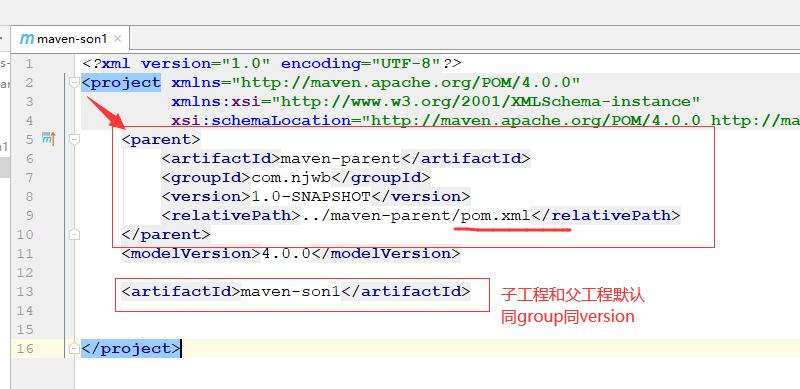
(2)在父工程中统一定义公共的jar依赖

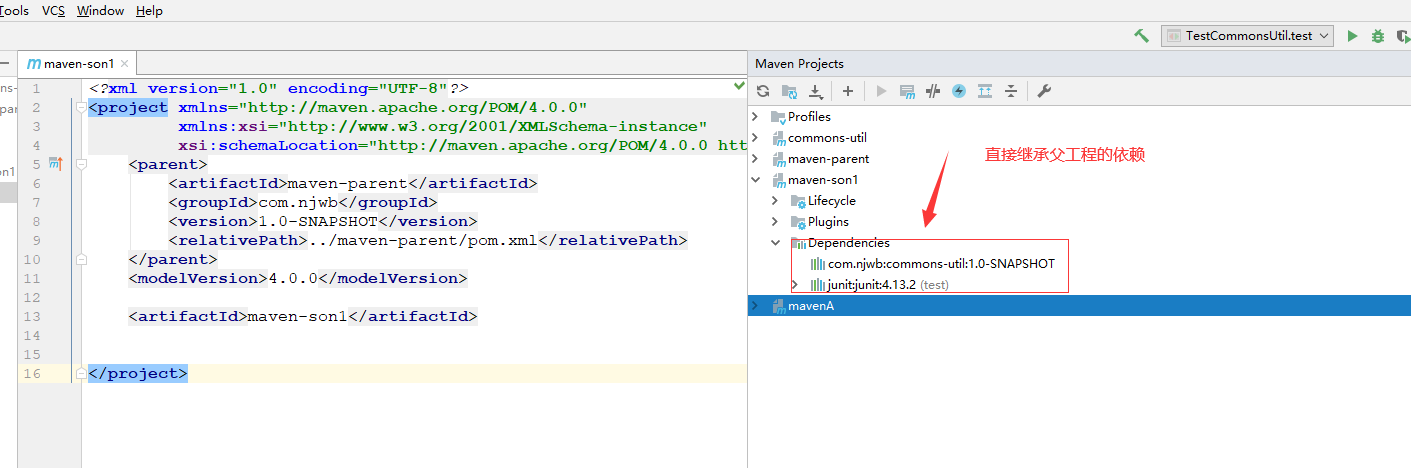


(3)创建第一个子工程

maven-son1

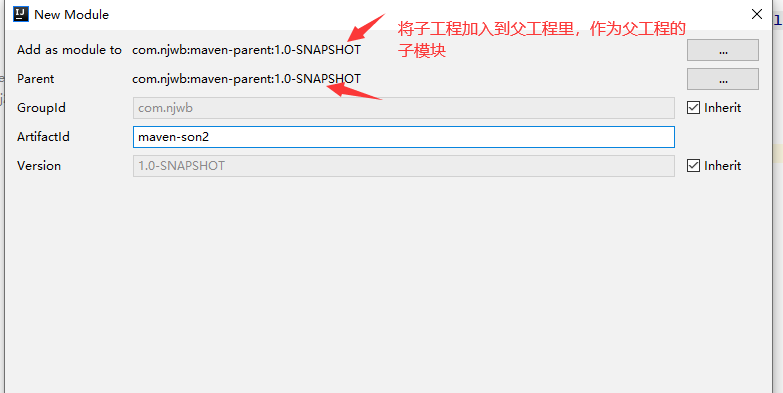


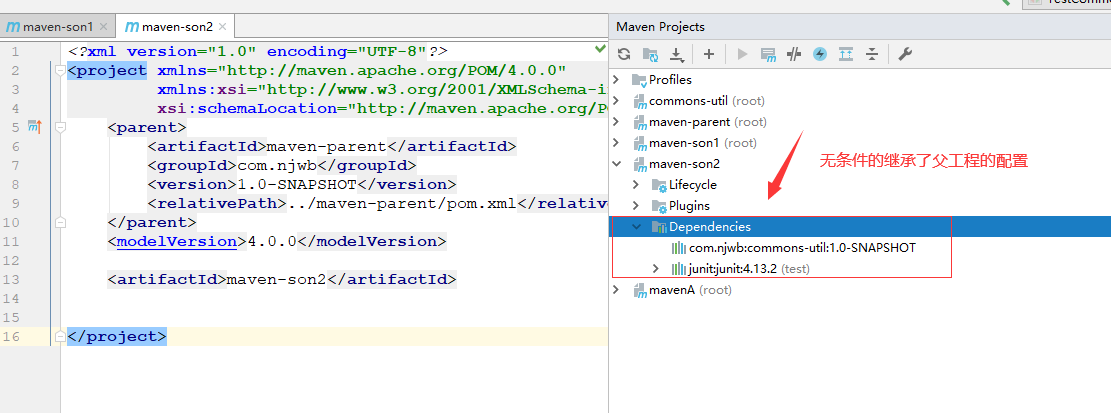




(4)创建第二个子工程

maven-son2





问题：子工程无条件的继承父工程的配置

如果：SON1只要继承commons-util的配置

SON2 只要继承junit的配置

多余的配置怎么处理？

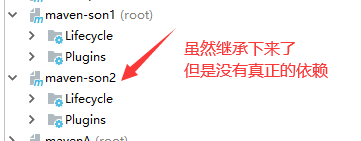
maven的做法：

父工程只负责jar的统一依赖配置管理，但是不负责真正的依赖。

修改父工程的pom配置



观察子工程的jar包依赖变化



子工程按需依赖



