# Maven

这个是java的项目管理工具（相当于Node.js的NPM），和svn代码管理工具有相似的地方，svn用于管理代码，maven用于管理java项目的依赖的版本，解决最难的版本不兼容的问题。另外maven还可以用构建项目，编译和发布。

Mvan的作用？

使用maven可以管理项目的依赖的版本，避免版本不兼容的问题。（说白了就是java不够完善，jar包版本不兼容，maven就是用来解决这类问题的）

使用maven之前需要先安装，就像是任何其他工具（npm，svn），并且配置好路径。

可以百度，然后还要在eclipse中进行maven的配置，参考其他文章。

## 镜像

settings.xml

<settings>

<mirrors>

<mirror>

<id>aliyun</id>

<name>aliyun</name>

<mirrorOf>central</mirrorOf>

<url>http://maven.aliyun.com/nexus/context/groups/public</url>

</mirror>

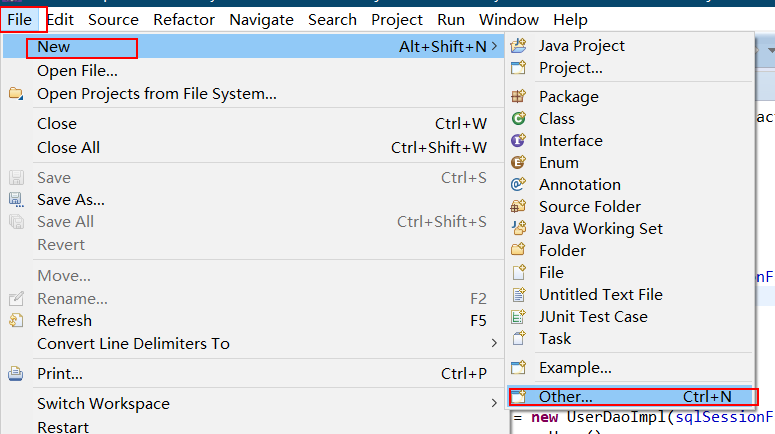
</mirrors>

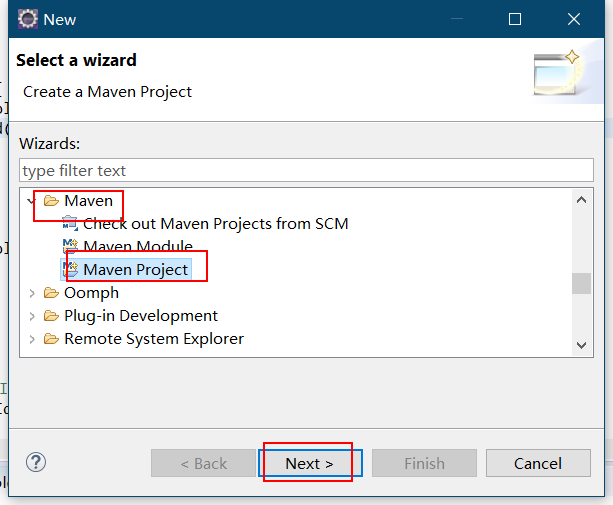
</settings>

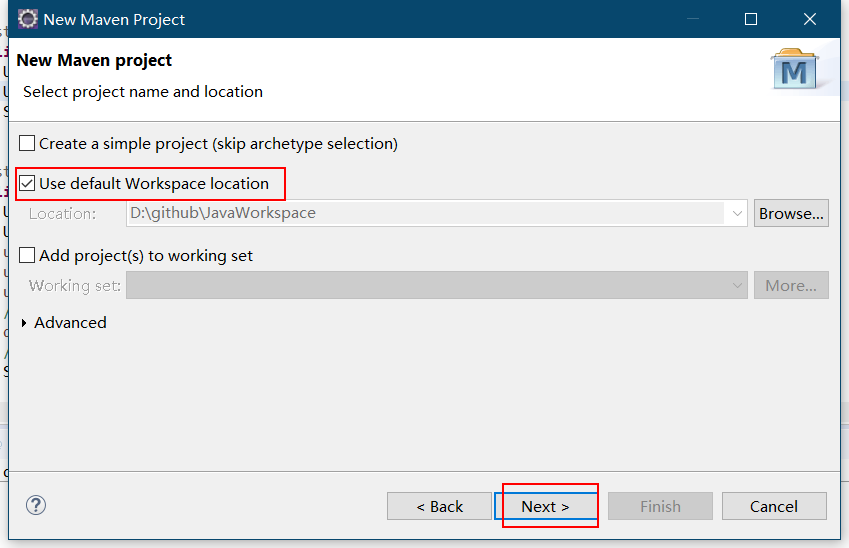
## 创建

### 新建一个maven项目

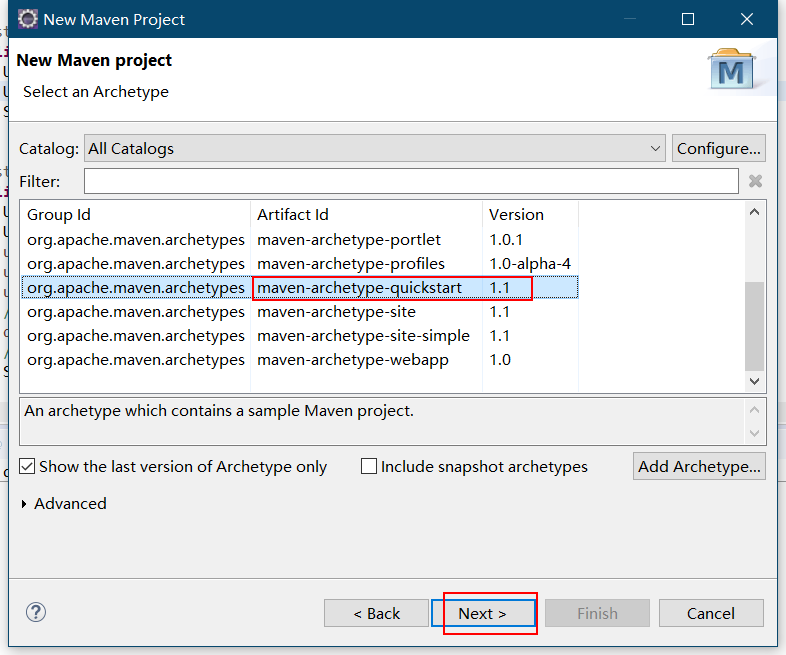
打开eclipse，文件—新建—其他—maven—maven项目







### 选择快速框架



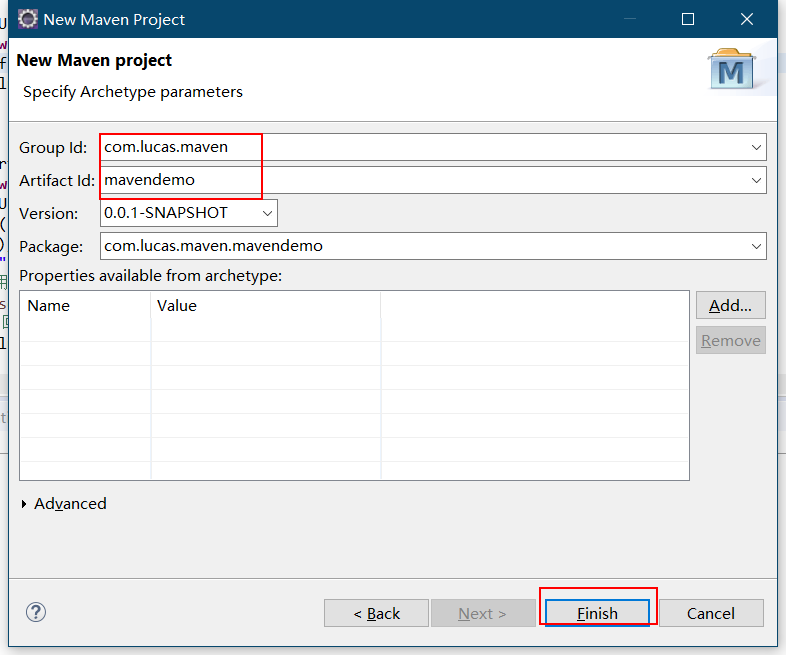
### 命名

GroupId一般是域名

artifactId 项目名

version版本号

这3个参数可以唯一定位一个jar包



第一次创建maven会自动从网络下载需要的jar包，稍微会有点慢。

## Maven管理项目

Src/main/java存放源码

Src/main/resources存储图片等资源

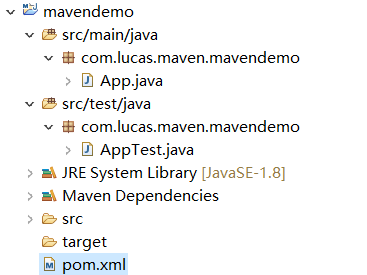
Src/test/java存储测试代码

Src/test/resources存储测试资源

Target编译输出

Pom.xml项目文件，用于版本管理

### 标准的架构



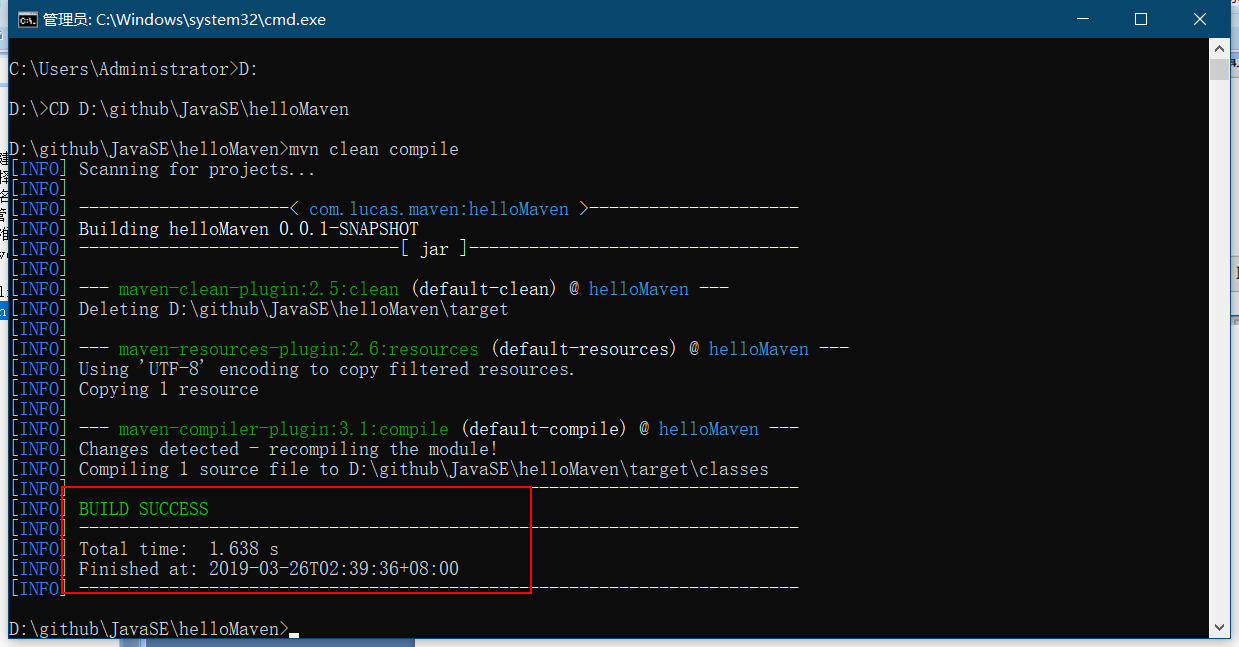
## 使用maven发布

一个maven生命周期有多个4个phase：

Clean compile test package

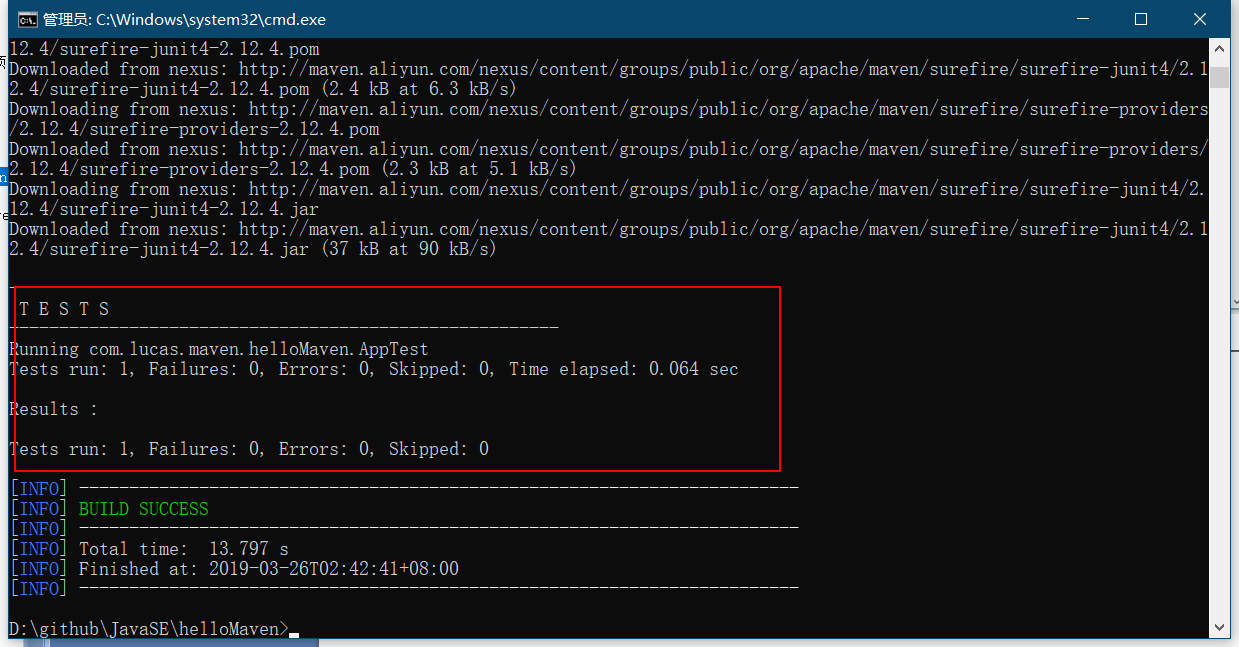
一个phase包含多个goal

执行构建在项目根目录打开命令行，执行mvn clean compile



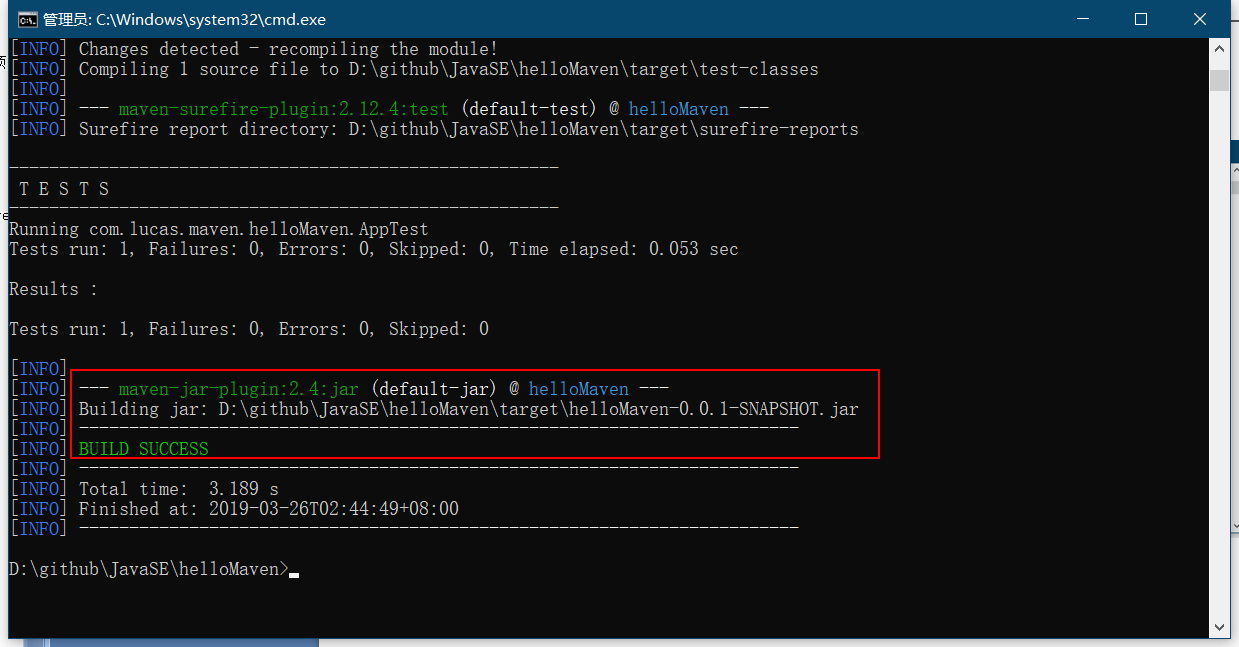
### 常用

常用的mvn clean test

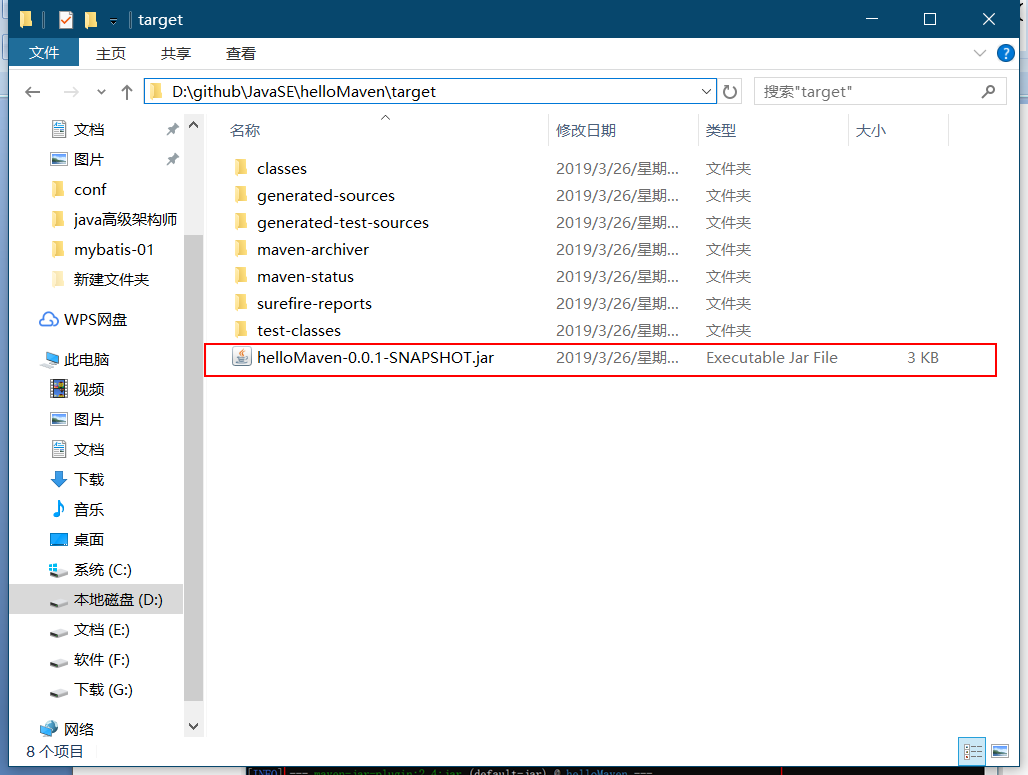


最常用当然是 mvn clean package

当出现build success，说明成功



打包的jar在target下



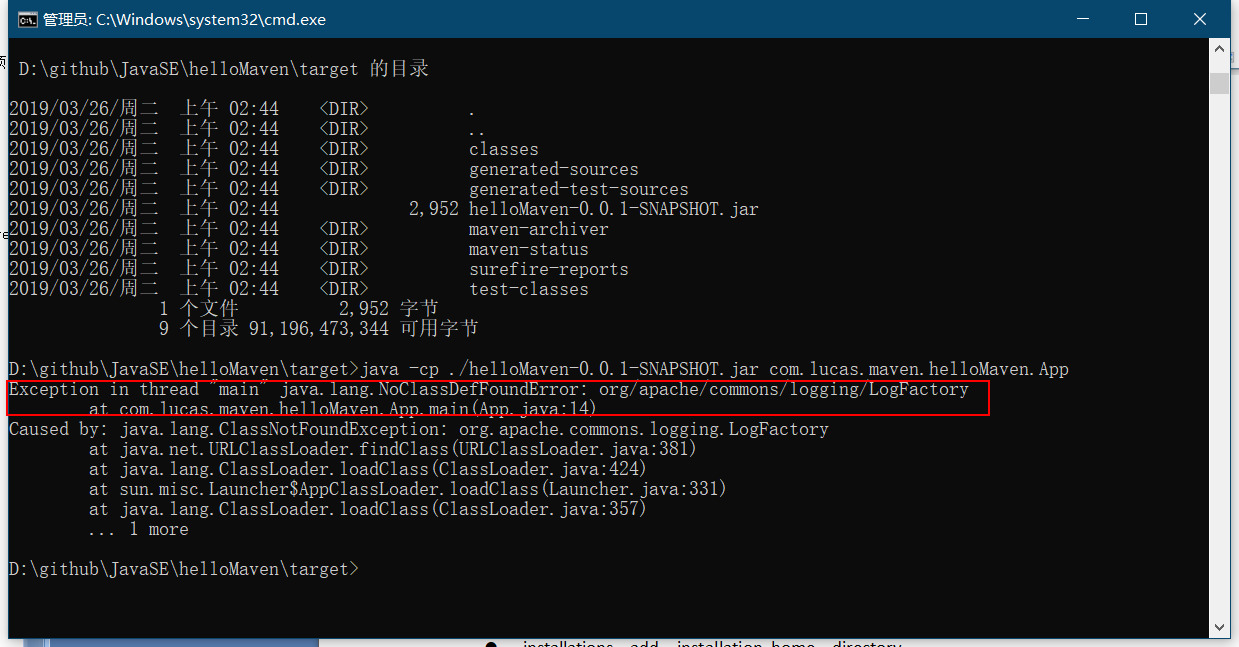
### 执行编译的jar

进入target ，cd target

输入java -cp ./helloMaven-0.0.1-SNAPSHOT.jar com.lucas.maven.helloMaven.App

发现有一个classNotFound的报错

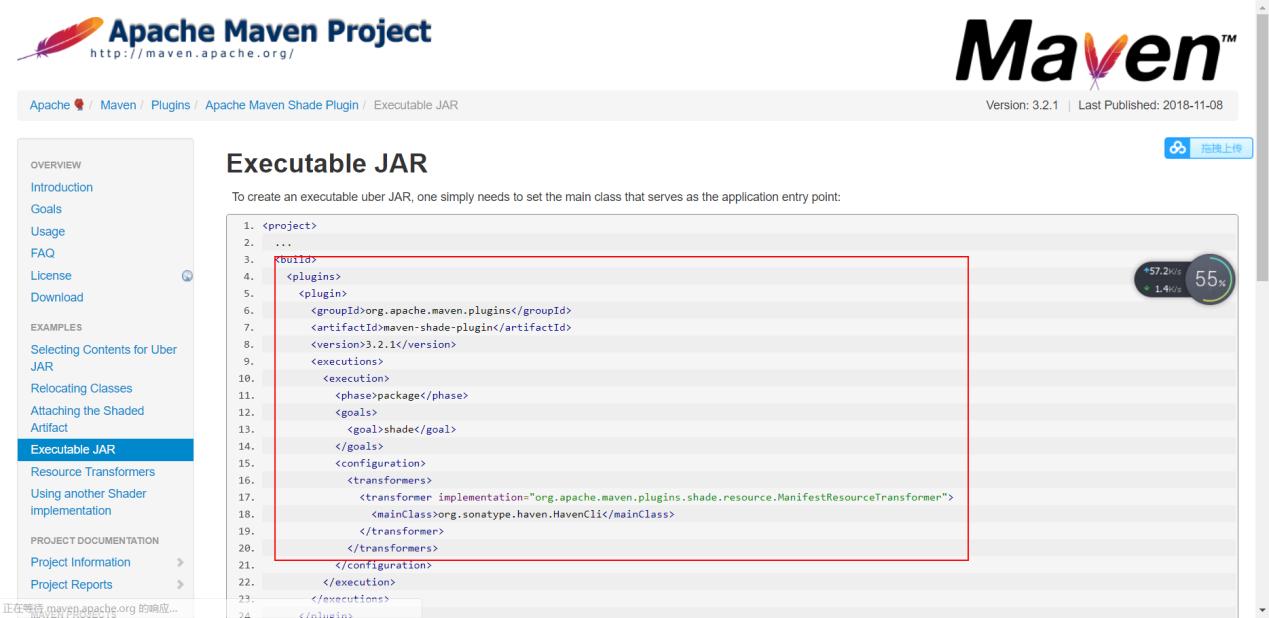
这是因为maven打包没有将log4j的jar包一起打包进入



## 可执行的jar

需要将依赖一起打包，用自定义jar插件

**浏览器输入 maven shade plugin executable jar**

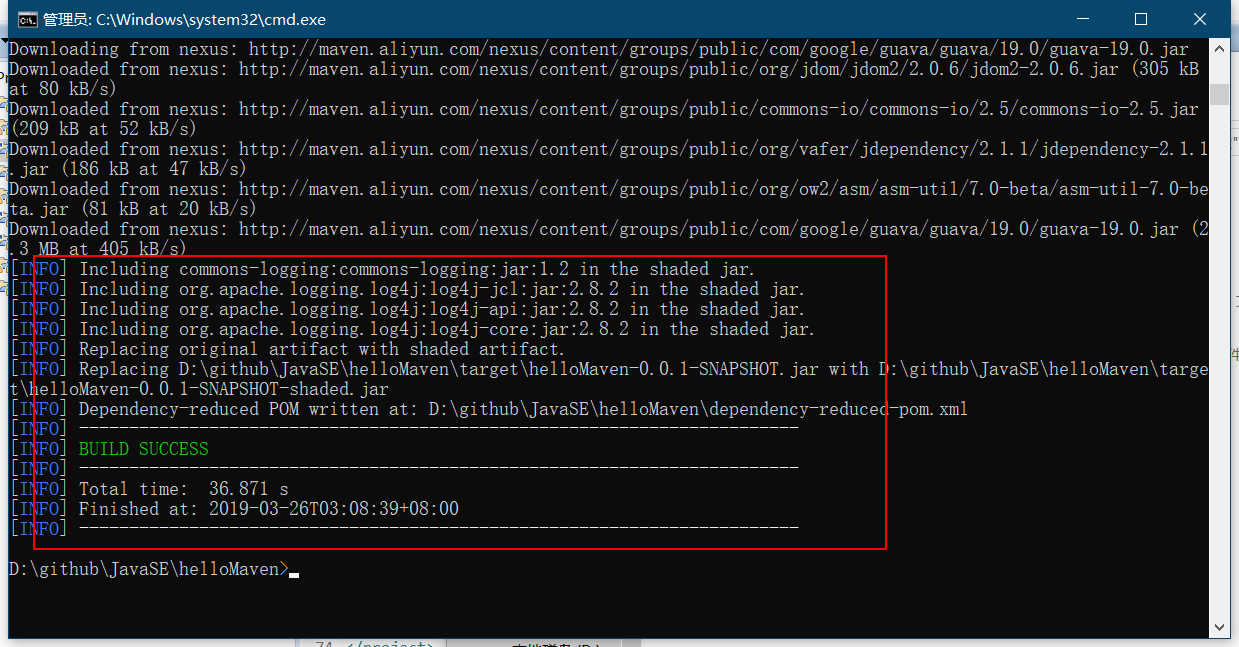


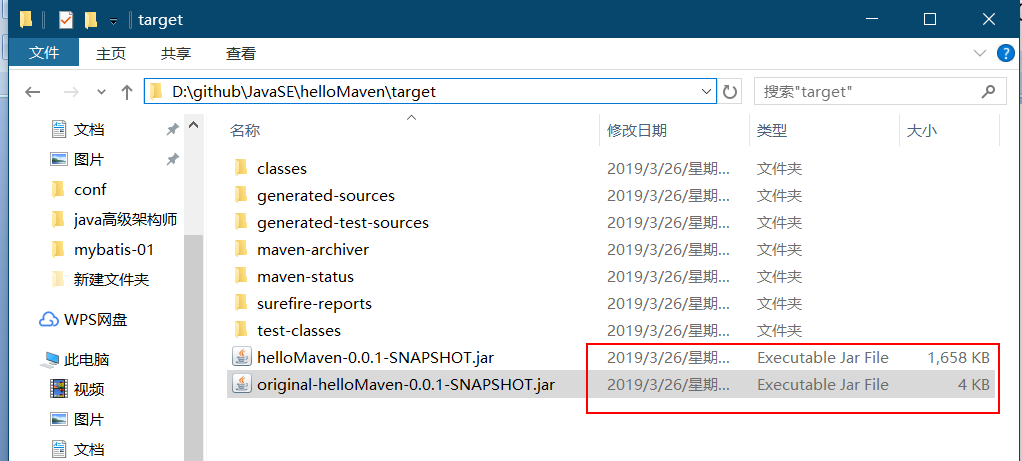
1. <build>
2. <plugins>
3. <plugin>
4. <groupId>org.apache.maven.plugins</groupId>
5. <artifactId>maven-shade-plugin</artifactId>
6. <version>3.2.1</version>
7. <executions>
8. <execution>
9. <phase>package</phase>
10. <goals>
11. <goal>shade</goal>
12. </goals>
13. <configuration>
14. <transformers>
15. <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
16. <mainClass>org.sonatype.haven.HavenCli</mainClass>
17. </transformer>
18. </transformers>
19. </configuration>
20. </execution>
21. </executions>
22. </plugin>
23. </plugins>
24. </build>

在pom.xml修改主函数的全名

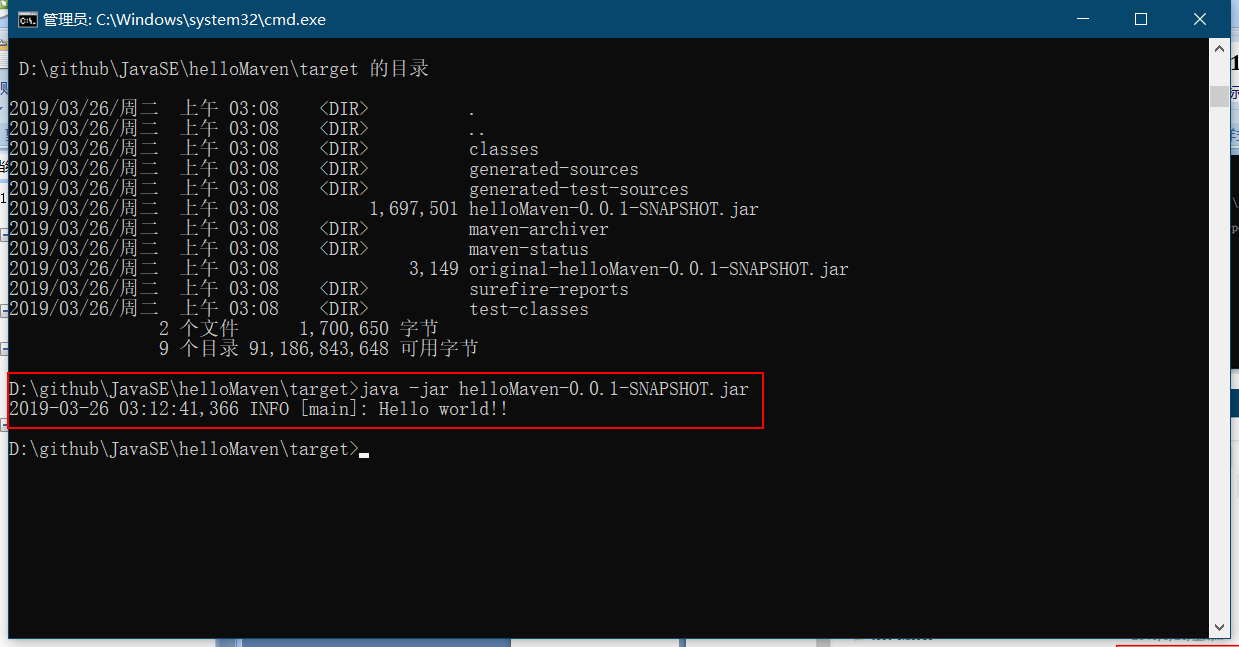


重新打包编译mvn clean package，会发现jar包很大，因为包含了依赖jar包



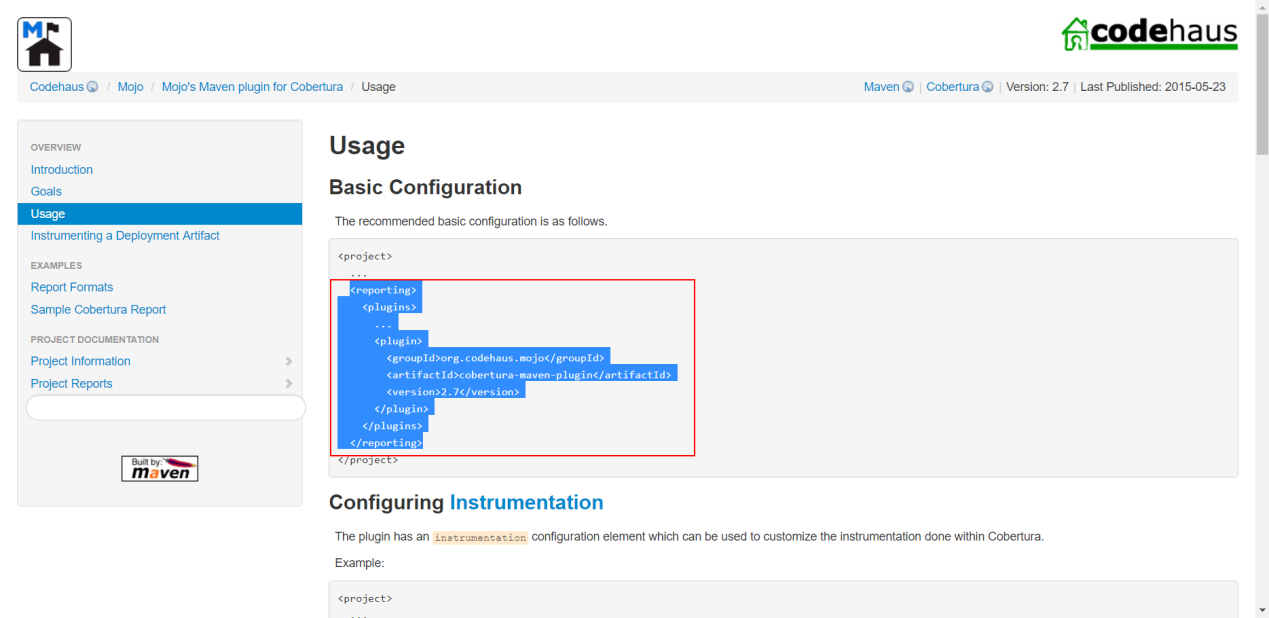


**执行java -jar helloMaven-0.0.1-SNAPSHOT.jar**



## 单元测试报告

浏览器输入 cobertura maven plugin usage



<reporting>

<plugins>

...

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>cobertura-maven-plugin</artifactId>

<version>2.7</version>

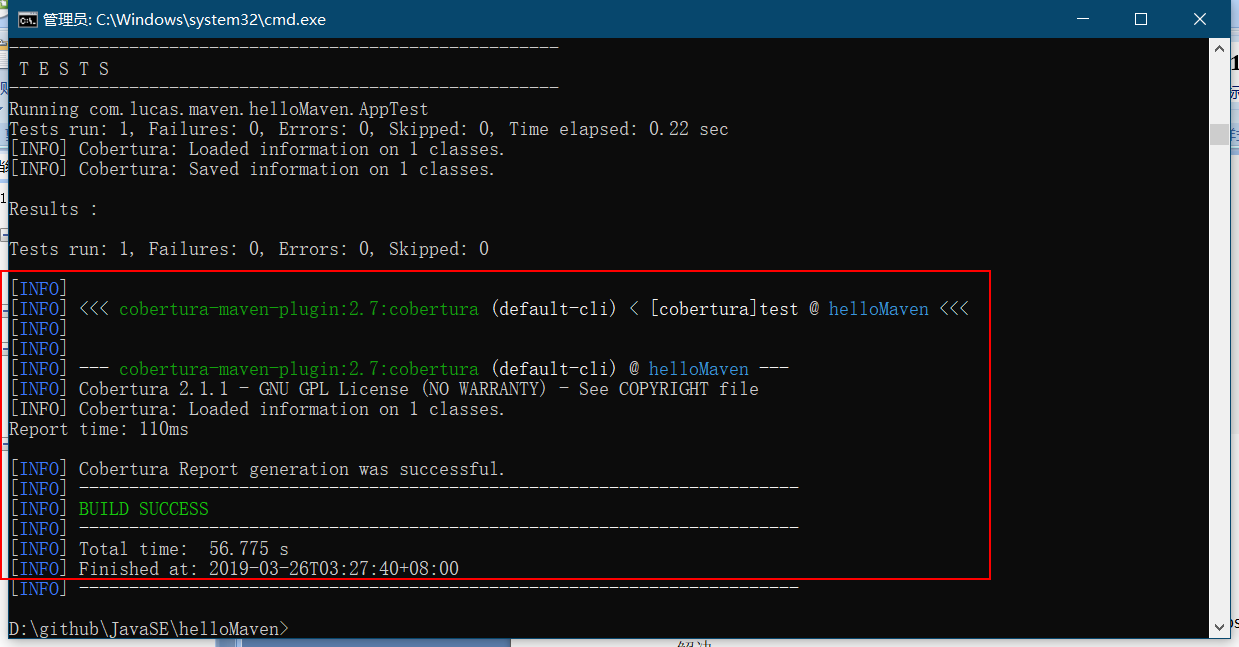
</plugin>

</plugins>

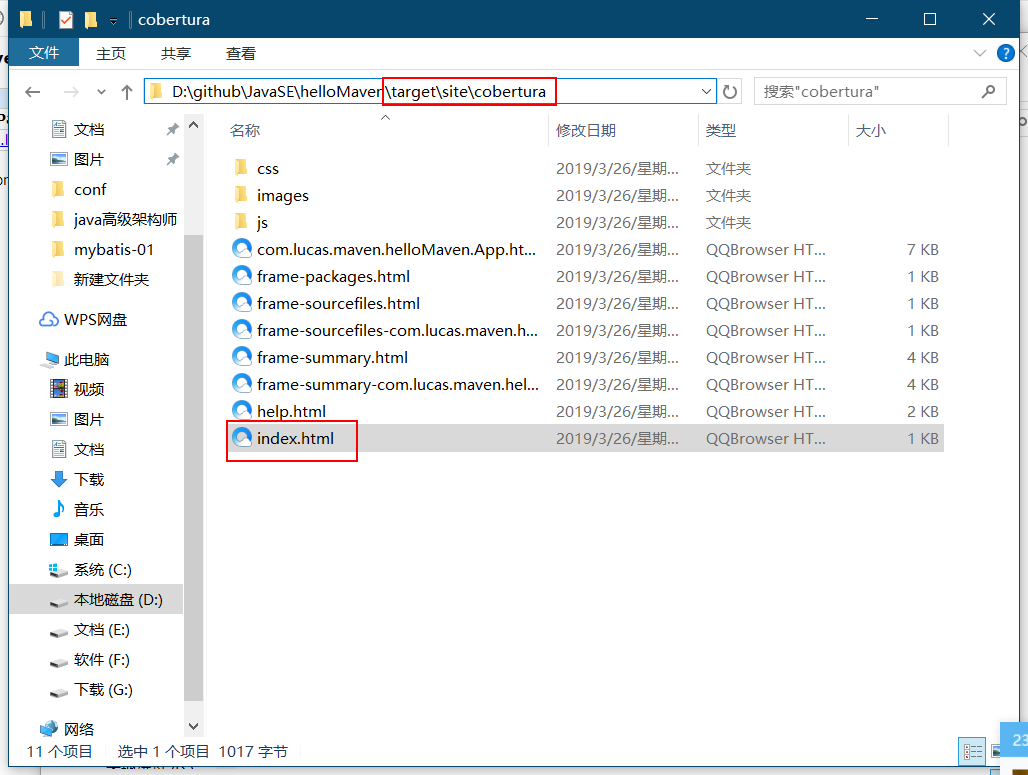
</reporting>

把这复制到pom.xml中，保存

运行 mvn cobertura:cobertura （这个是一个goal，最小执行单元）



打开index.html即可看见报告



报告覆盖率为62%



## 项目拆分

将一个项目拆分多模块管理，通过pom.xml依赖引入

如果pom.xml相似，可以提取一个项目parent，作为pom.xml的公共部分

这时单独编译helloMaven会报错，找不到greeting的依赖

**在pom.xml中添加dependency标签**

<dependency>

<groupId>com.lucas.maven</groupId>

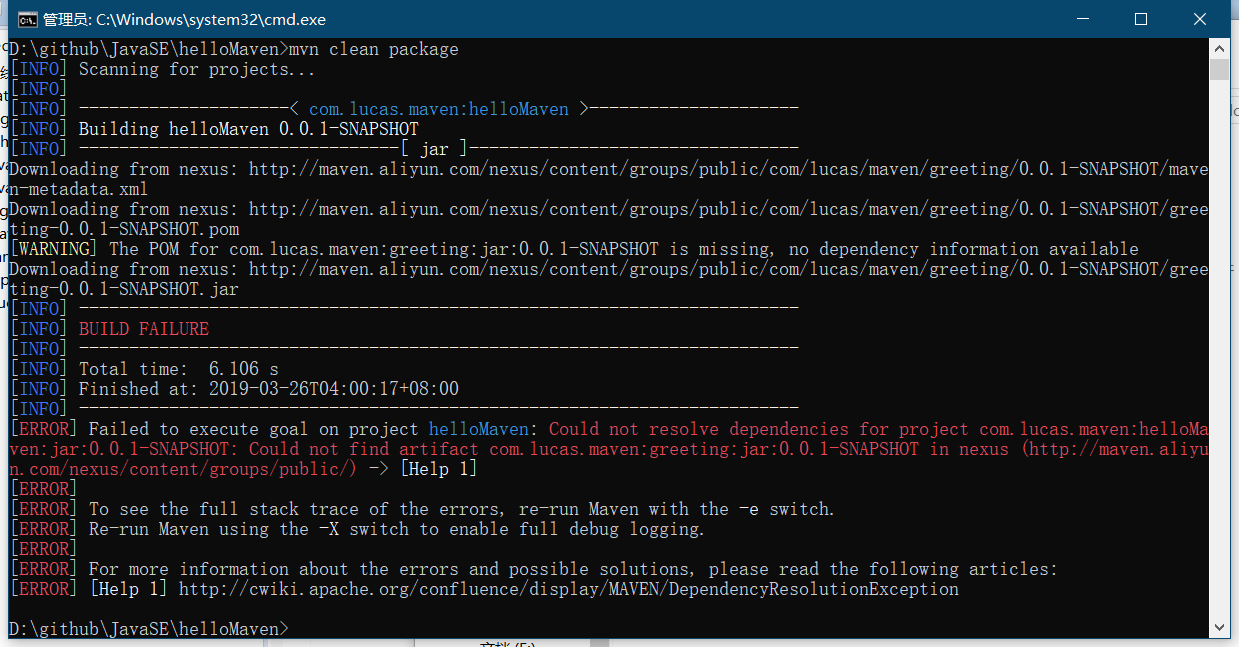
<artifactId>greeting</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

</dependencies>

但是编译的时候会有依赖的问题



**用一个build项目来管理模块化编译**

**在pom.xml中添加modules标签**

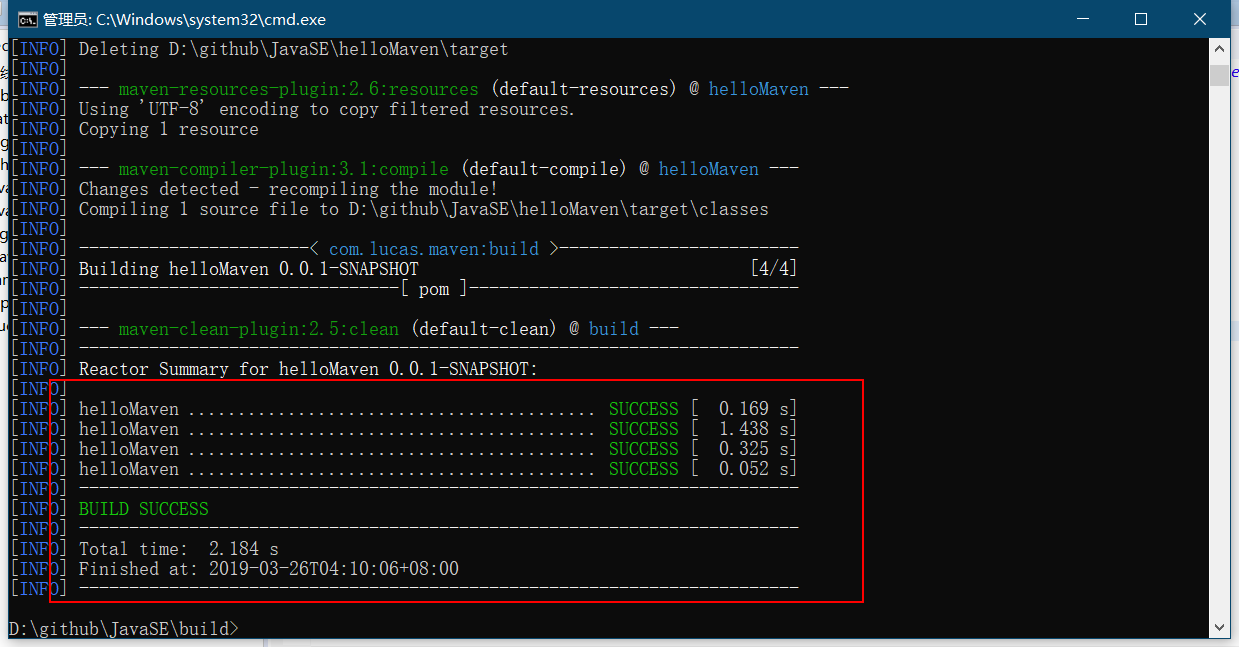
<modules>

<module>../parent</module>

<module>../greeting</module>

<module>../helloMaven</module>

</modules>



## 问题

描述：如果遇到修改pom.xml保存更新之后没有反应，可能是eclipse没有配置maven

解决：

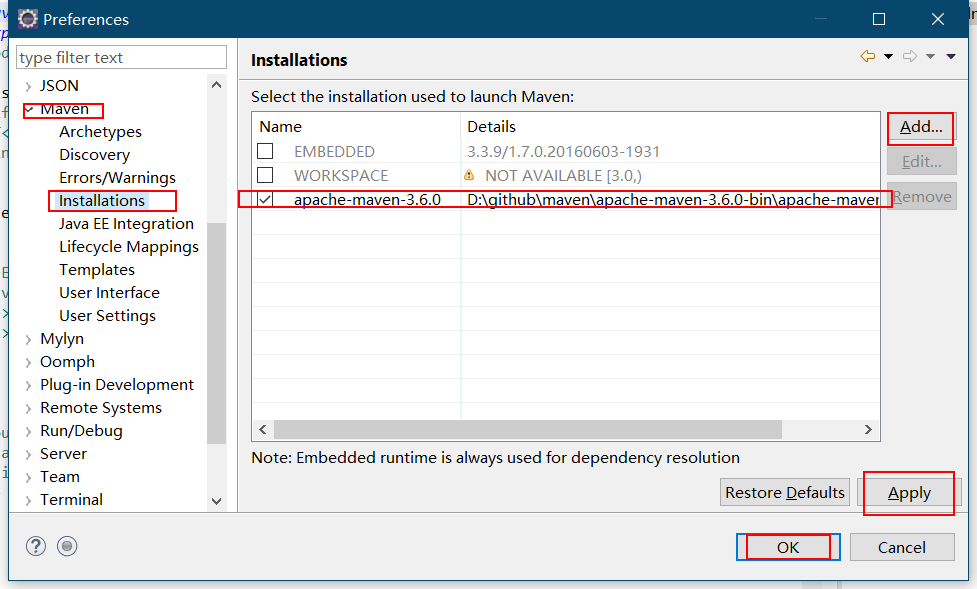
### eclipse配置maven

Windows—preferences—maven

* installations—add---installation home—directory

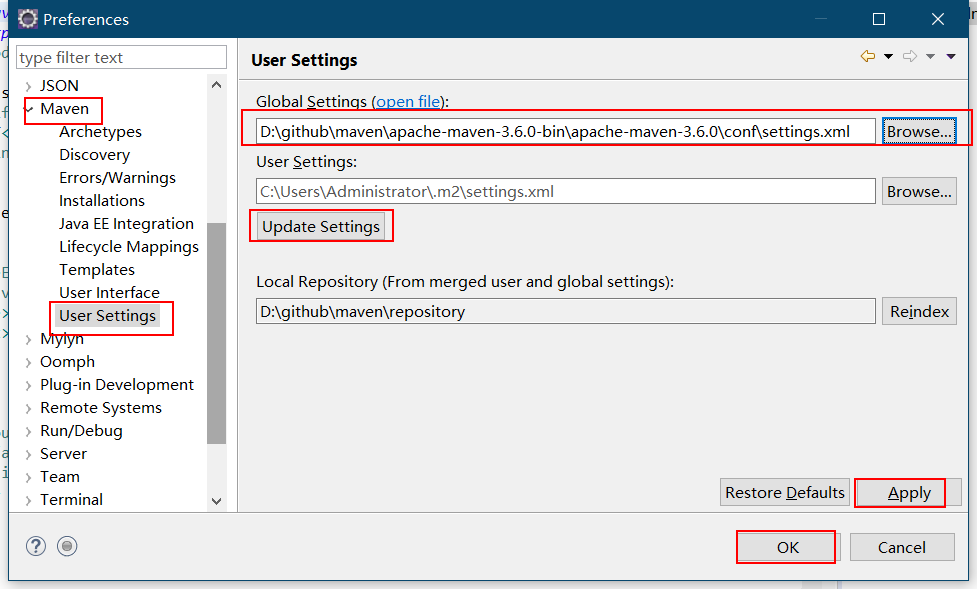
选择maven安装文件夹—finish

勾选刚才添加的maven—apply—ok



* user settings—global settings—browse

选择maven安装目录下conf文件夹里面的settings.xml-打开—update settings—apply—ok

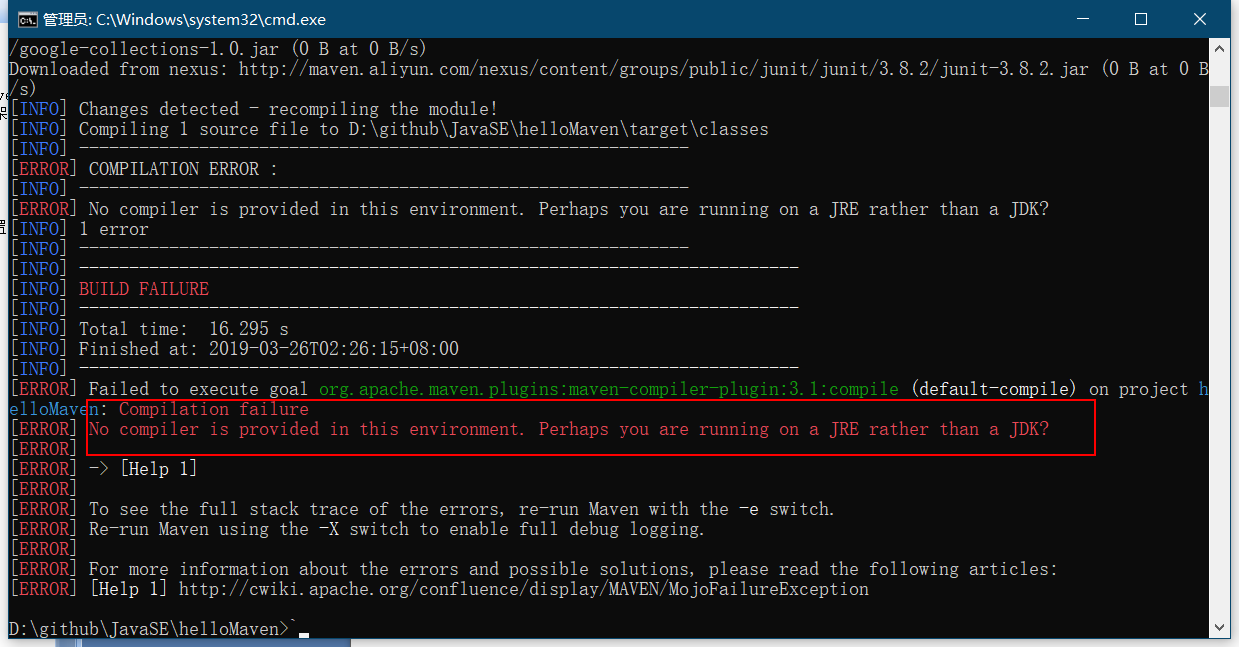


现在再一次刷新pom.xml就会有反应了

### mvn构建报错

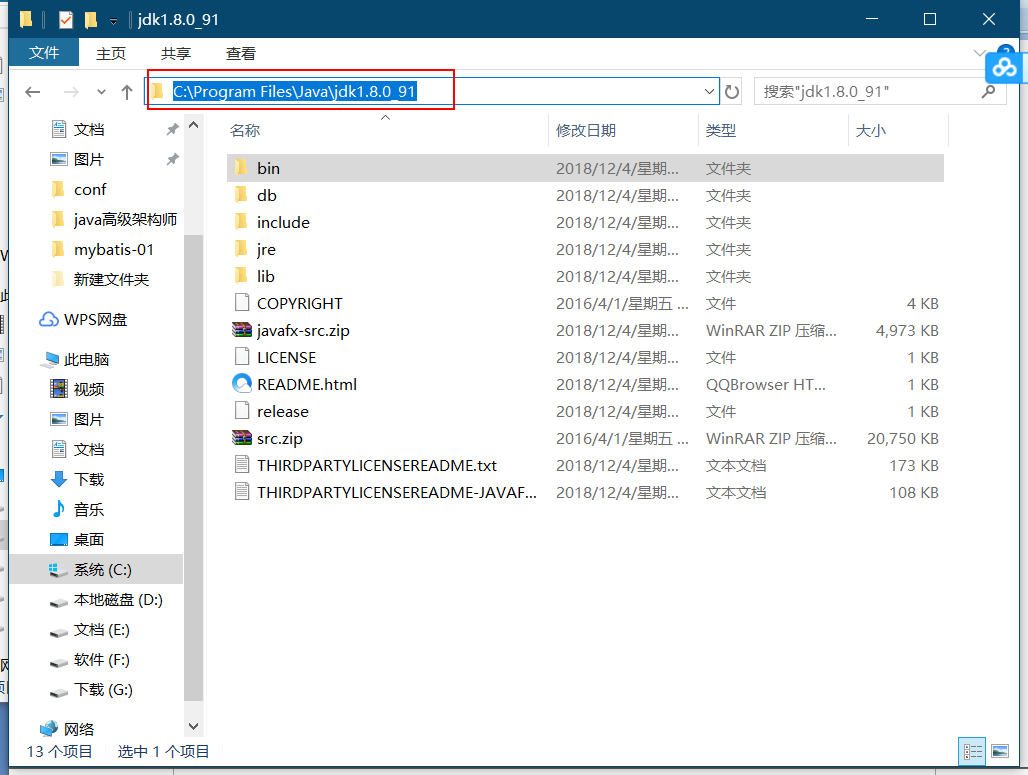
问题：

No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK?

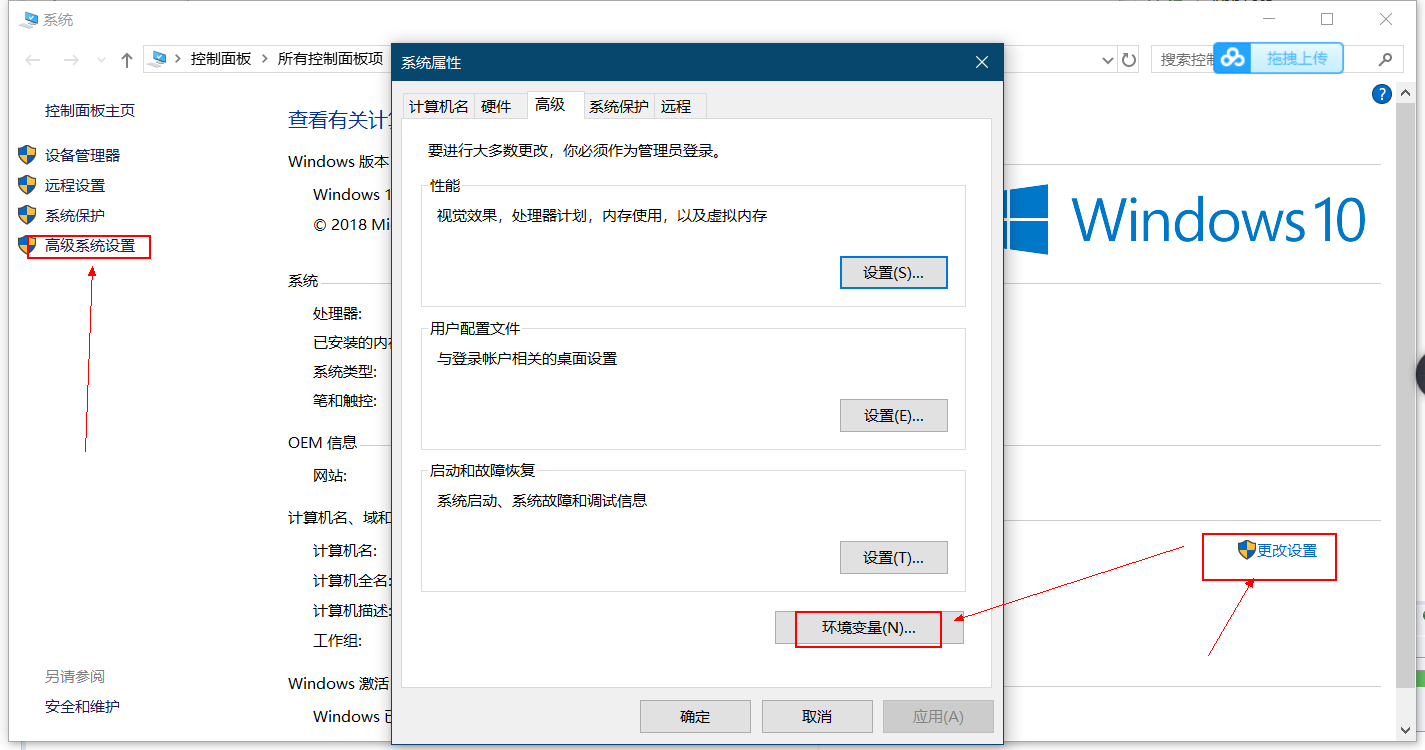


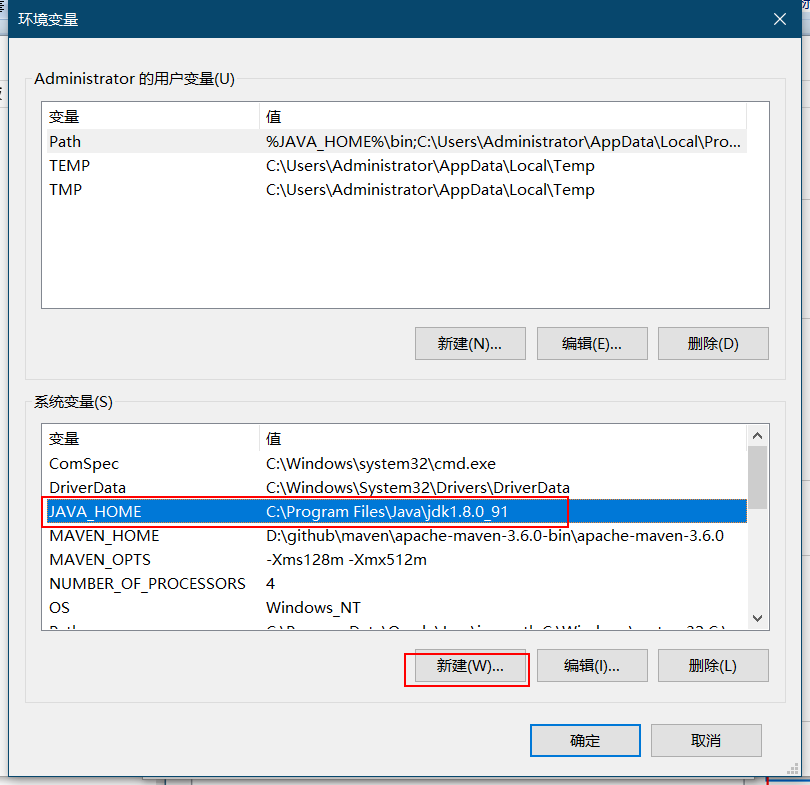
需要配置jdk

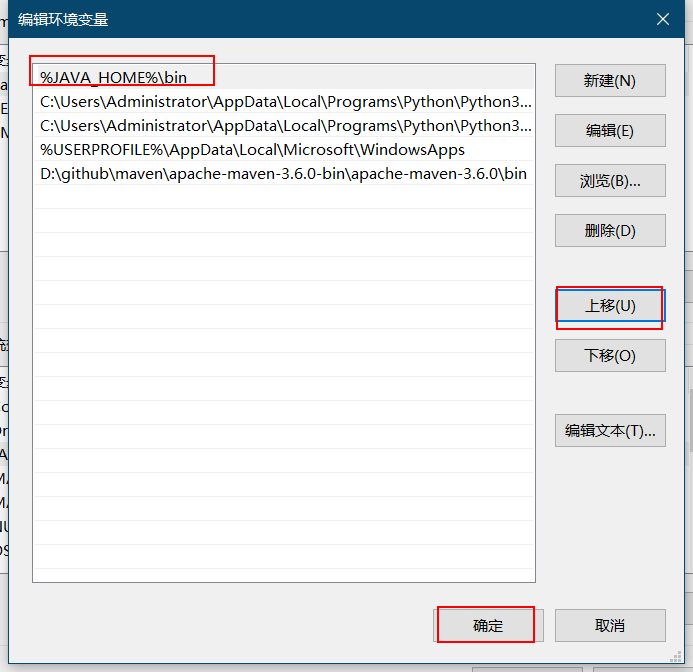
找到jdk根目录，复制路径：C:\Program Files\Java\jdk1.8.0\_91



添加环境变量JAVA\_HOME，并添加进入path







**注意：一定要保存，然后关掉cmd，重新进入**

然后再一次编译mvn clean compile

