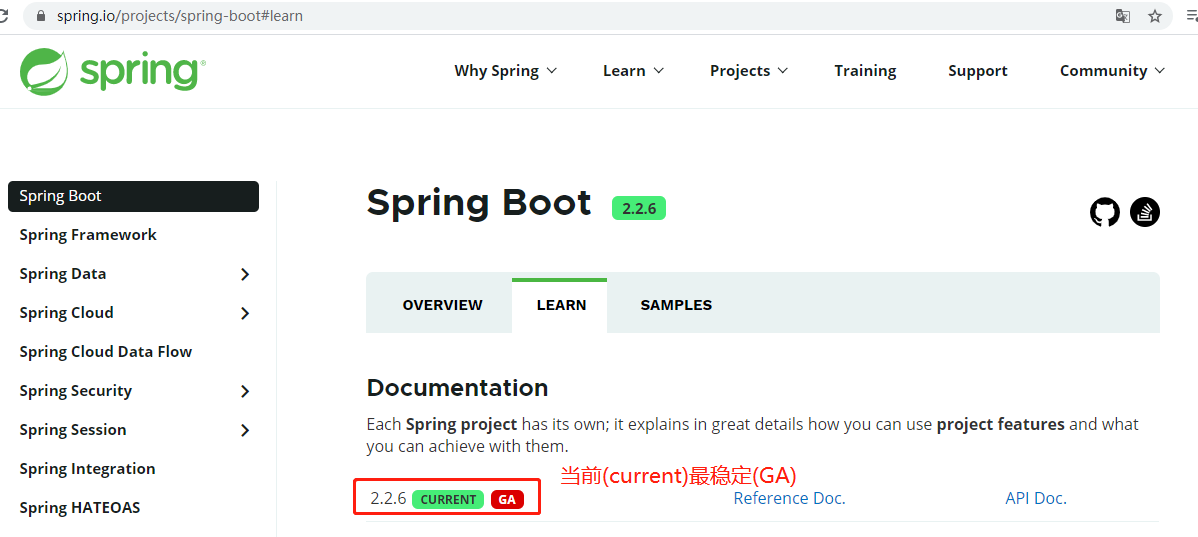
# 版本选择

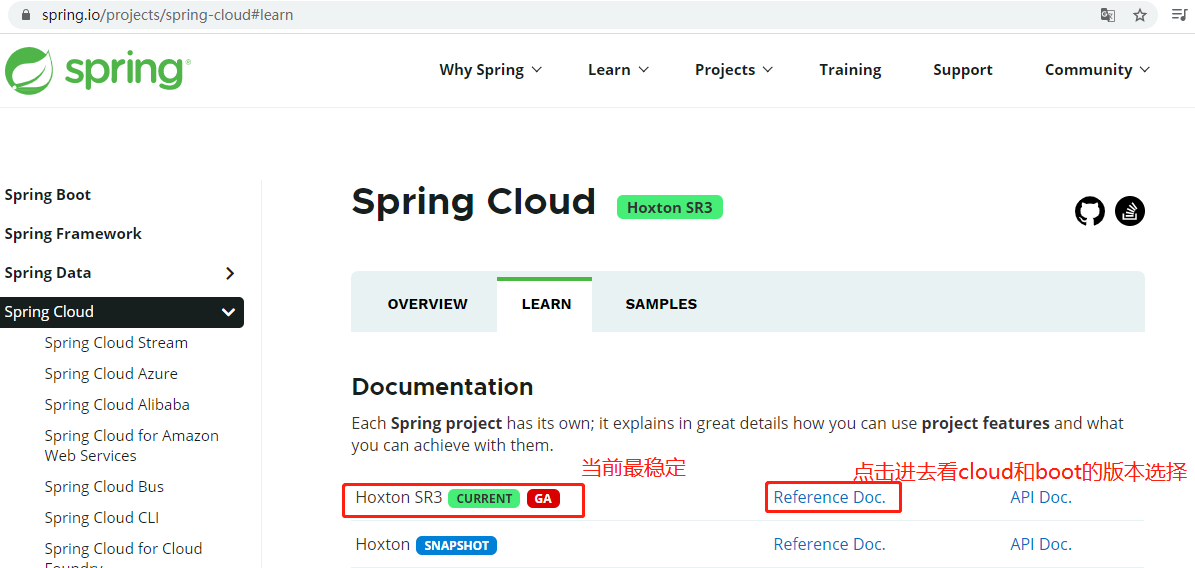
## springBoot

<https://spring.io/projects/spring-boot>



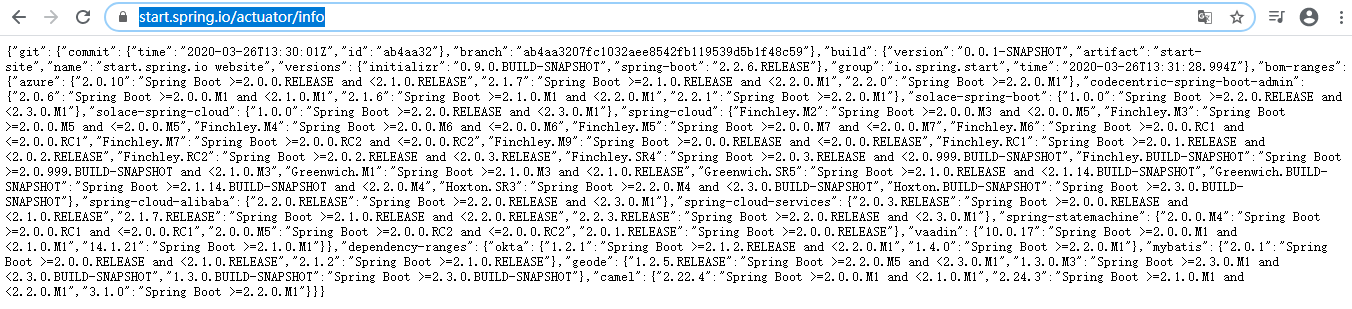
## SpringCloud

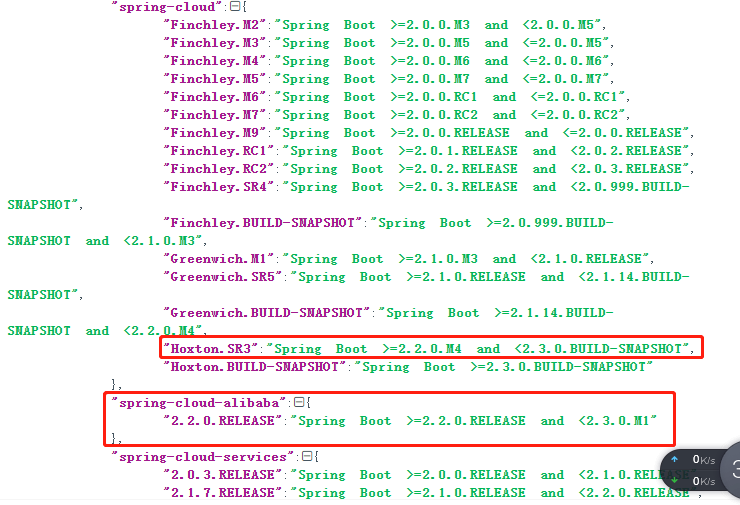
<https://spring.io/projects/spring-cloud>



## 1.3 springBoot和springCloud版本对应查看

<https://start.spring.io/actuator/info>

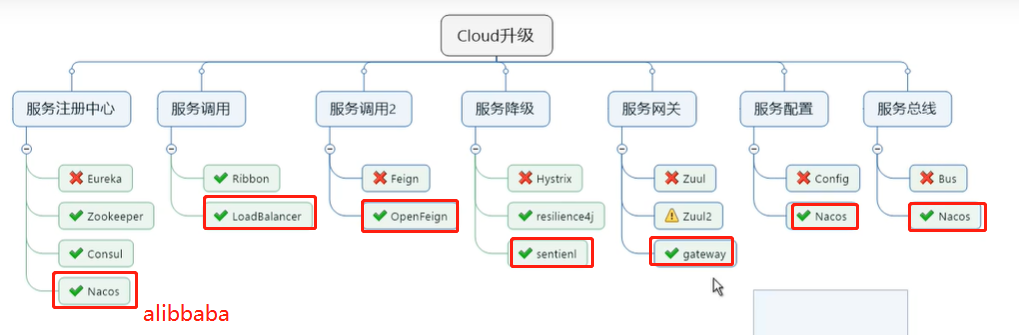




## 1.4 本次版本选择

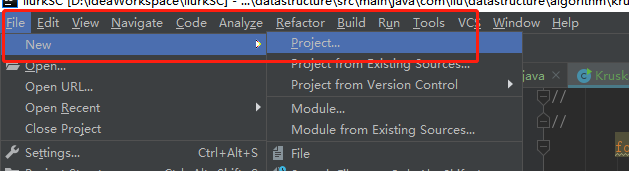
|  |  |
| --- | --- |
| springCloud | Hoxton.SR1 |
| springBoot | 2.2.2.RELEASE |
| Cloud alibaba | 2.1.0.RELEASE |
| Jdk/java | 1.8/java8 |
| Maven | 3.5及以上(3.6.0) |
| Mysql | 5.7及以上(5.8) |

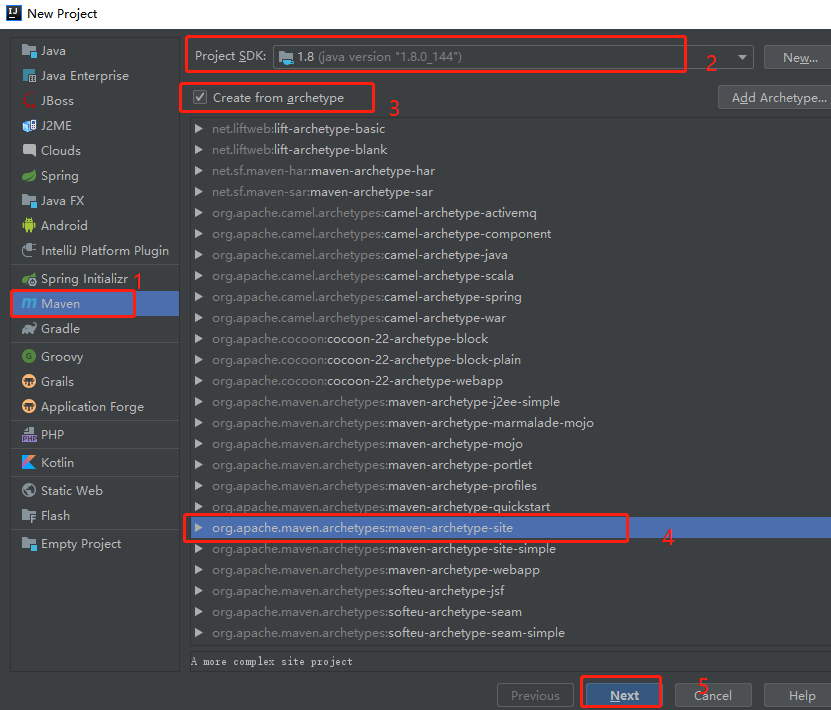
# springCloud各组件技术选型

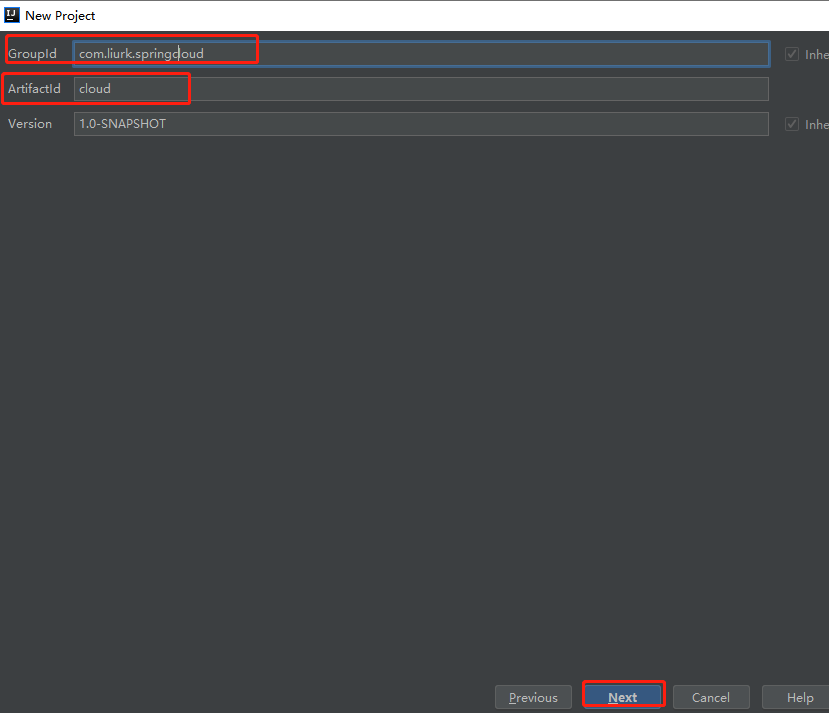


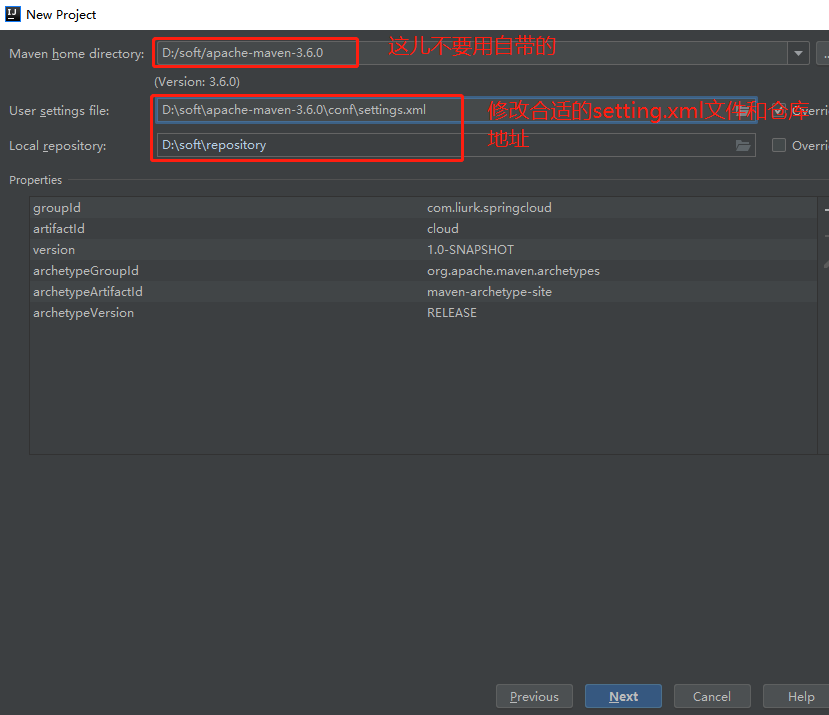
# 创建父工程

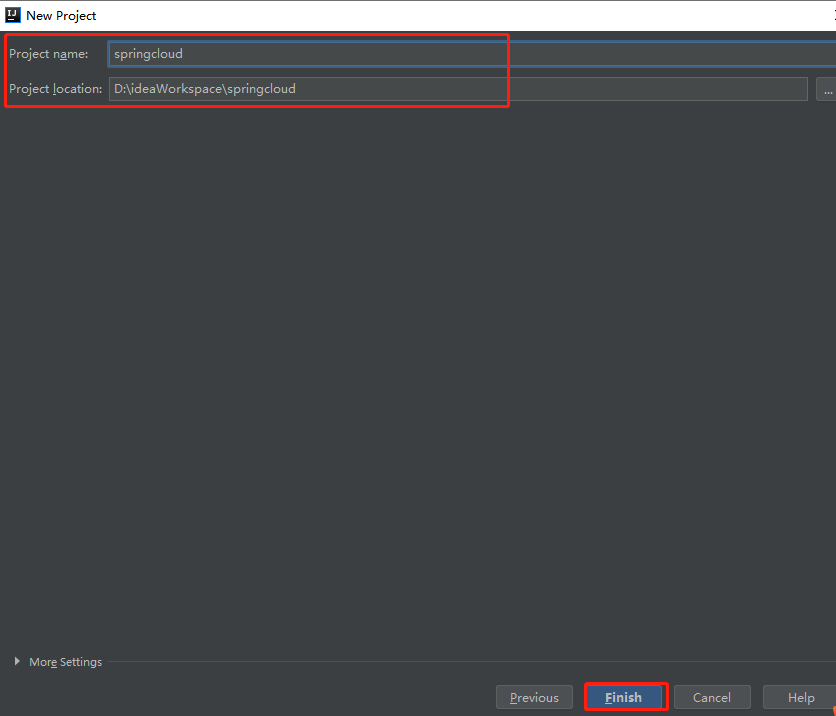
## 3.1 创建springcloud父工程

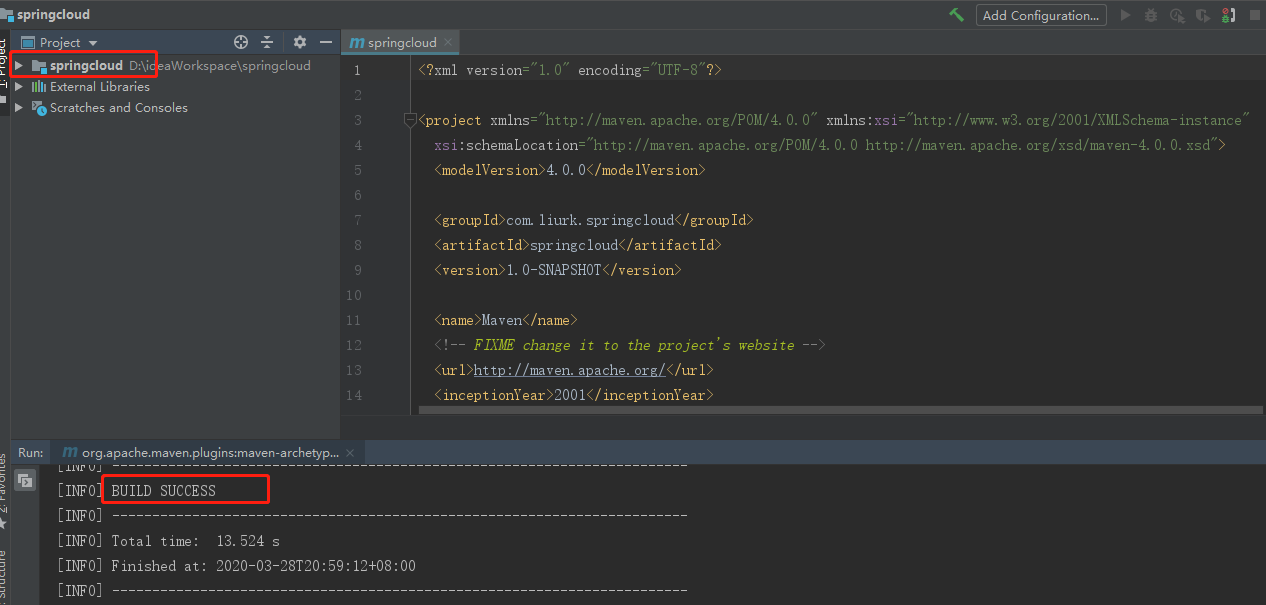




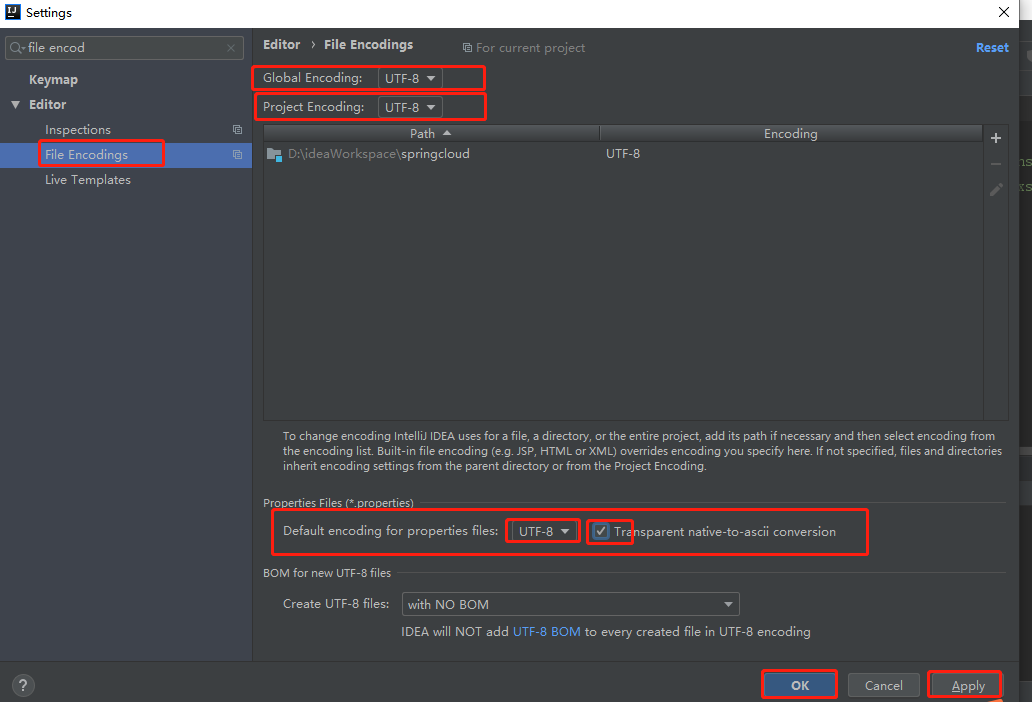


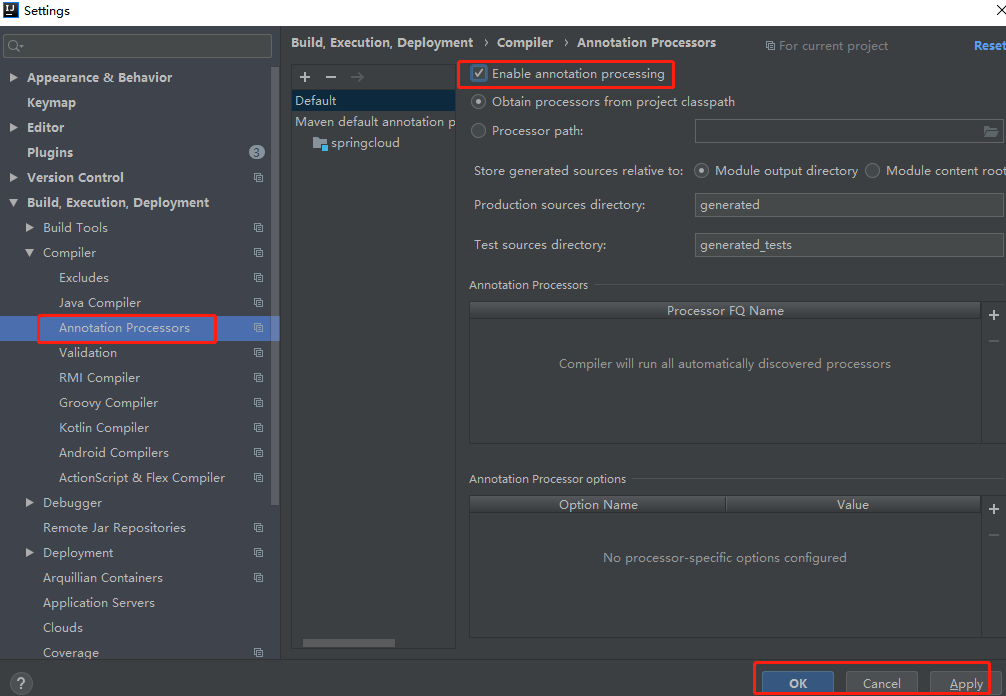


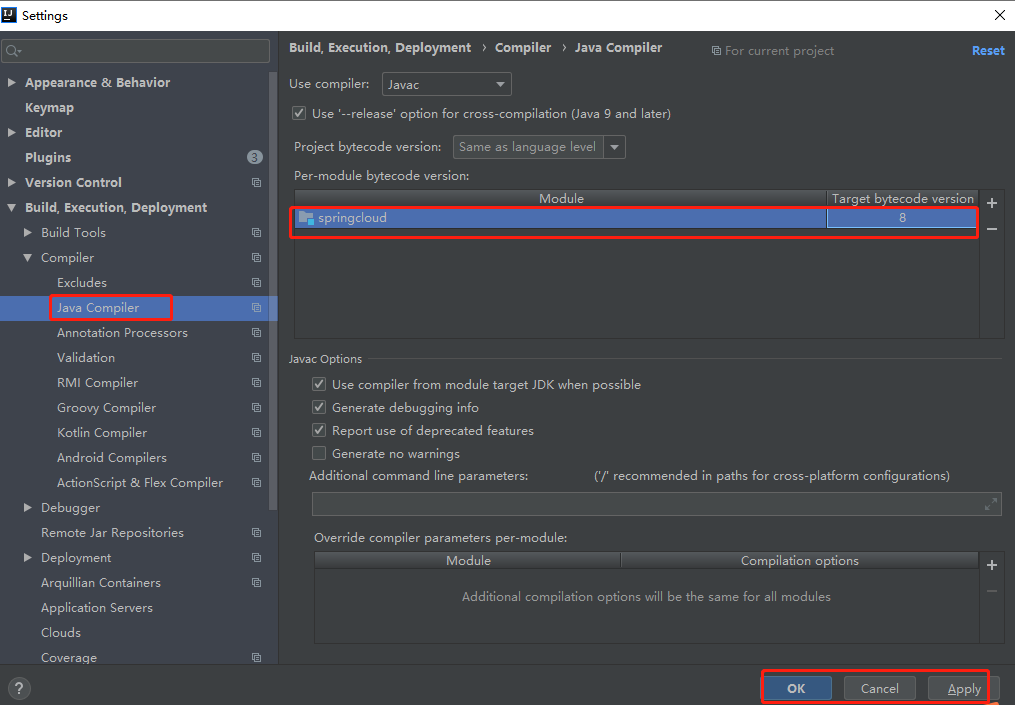




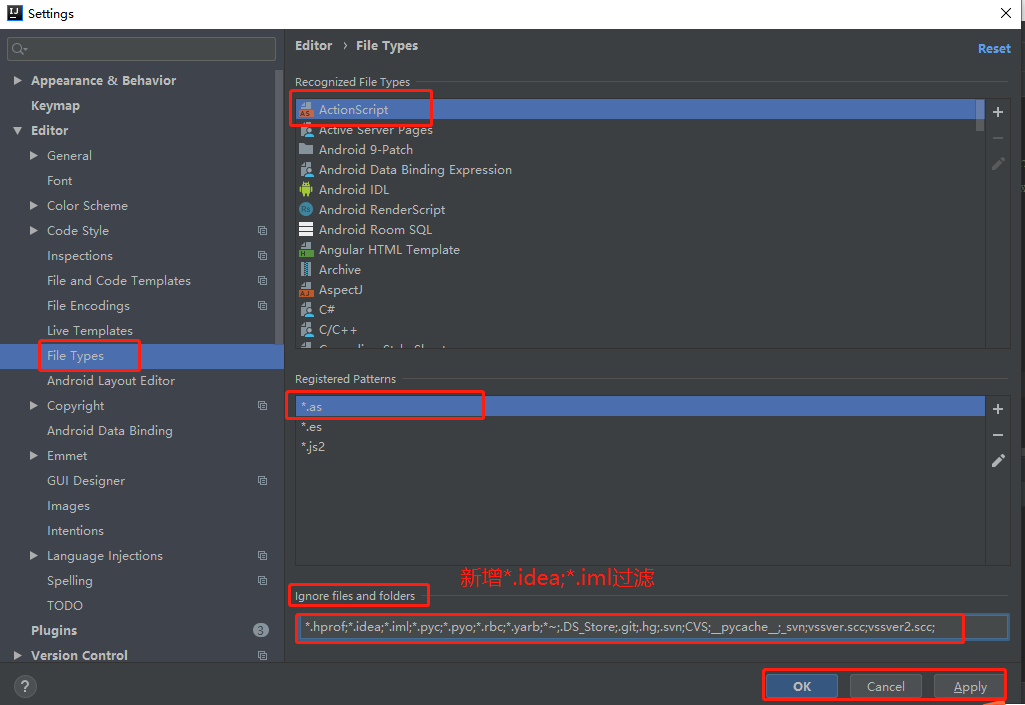
## 3.2 字符编码、注解激活生效、java编译版本





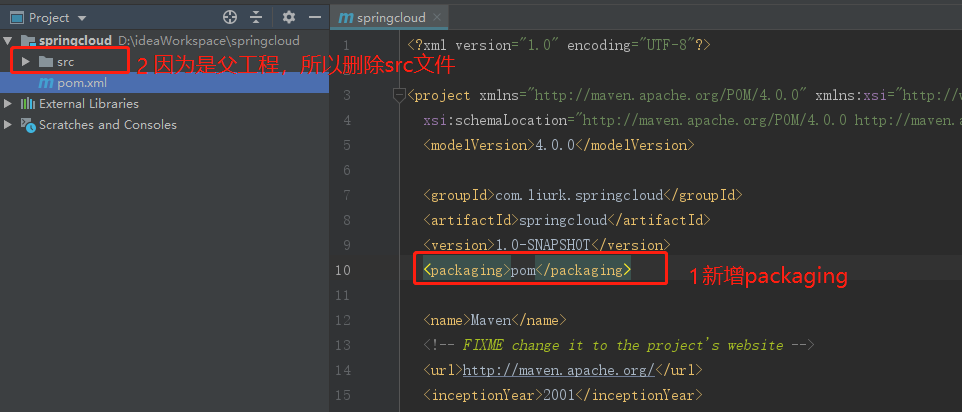


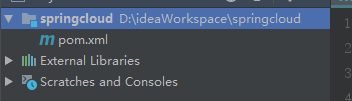
## 3.3 File Types过滤(\*.idea;\*.iml;)



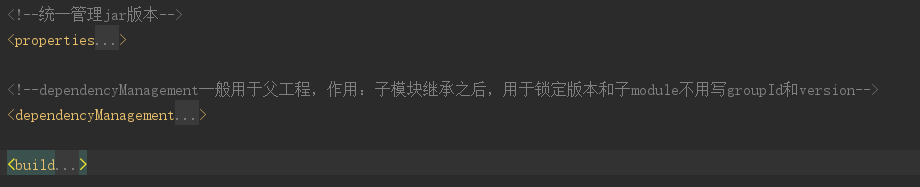
## 3.4 父工程pom.xml

### 3.4.1 修改pom.xml和删除src文件





### 3.4.2 pom.xml添加依赖



### 3.4.3 dependencyManagement和dependencies的区别

（1）dependencyManagement

maven使用dependencyManagement元素来提供了一种管理依赖版本号的方式

一般会出现在项目的父工程

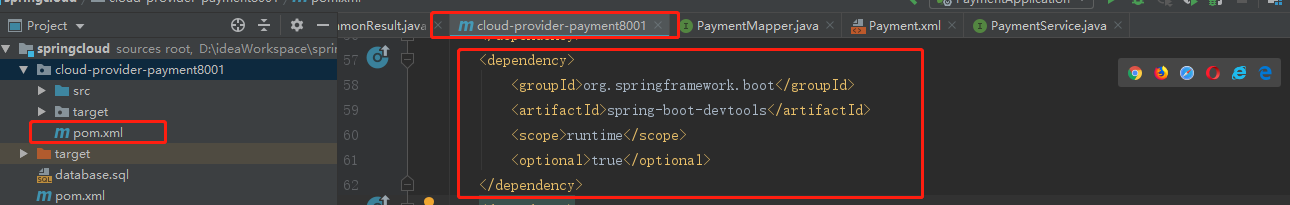
使用pom.xml中的dependencyManagement元素能让所有子项目中引用一个依赖而不显式的列出版本号，maven会沿着父子层次向上走，直到找到一个拥有dependencyManagement元素的项目，然后就会使用dependencyManagement元素中指定的版本号

这样做的好处是：如果有多个子项目都引用同一依赖，则可以避免在每个使用的子项目里都声明一个版本号，这样想升级或者切换另一个版本时，只需要在顶层父容器里更新，而不需要一个一个修改子项目。当某个子项目需要另外的一个版本时，只需要声明version即可，如果子项目中制定了版本号，那么会使用子项目中指定的jar版本

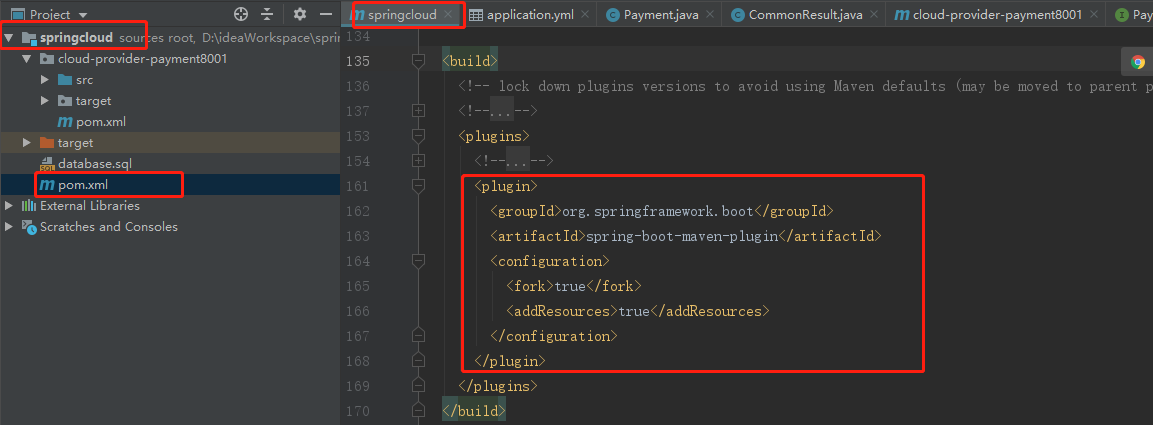
注意：dependencyManagement里只是声明依赖，并不实现引入，因此子项目需要显示的声明需要用的依赖

# 热部署Devtools

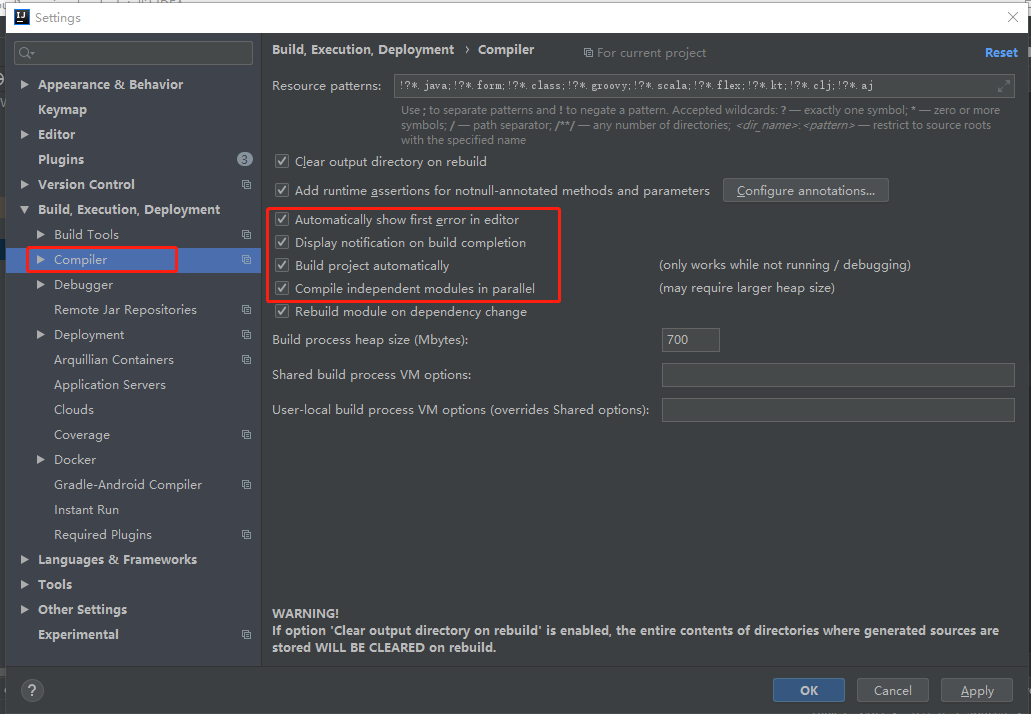
## 在子module添加依赖



## 在父工程中添加maven插件

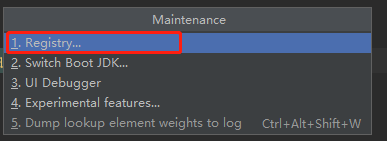


## 开启自动编译

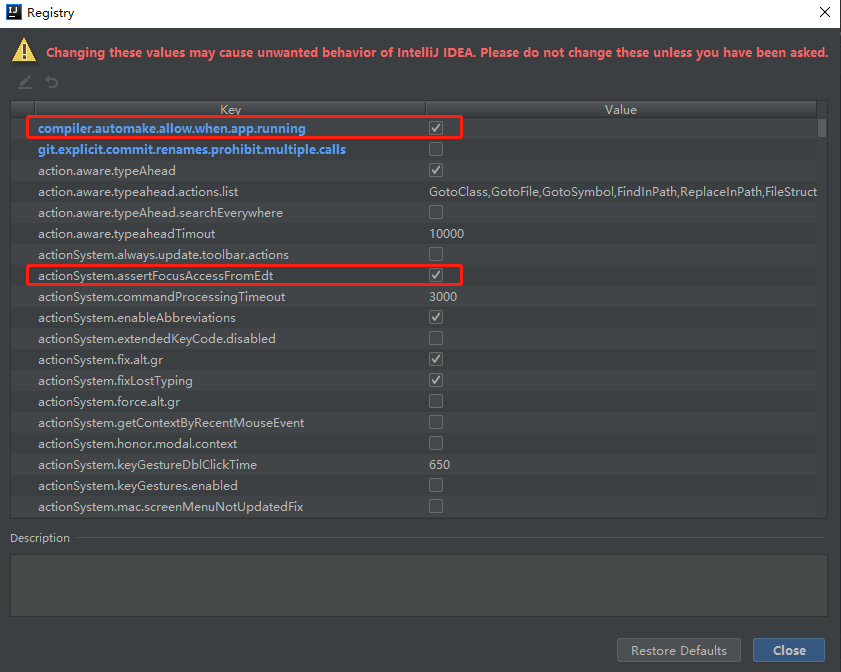


## Update the value of

快捷键：ctrl+alt+shift+/ 会有个弹框，点击Registry

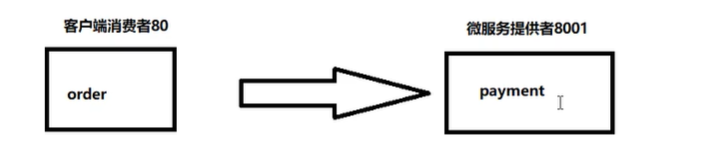


勾选如下两个



## 重启idea

# Restful微服务工程构建

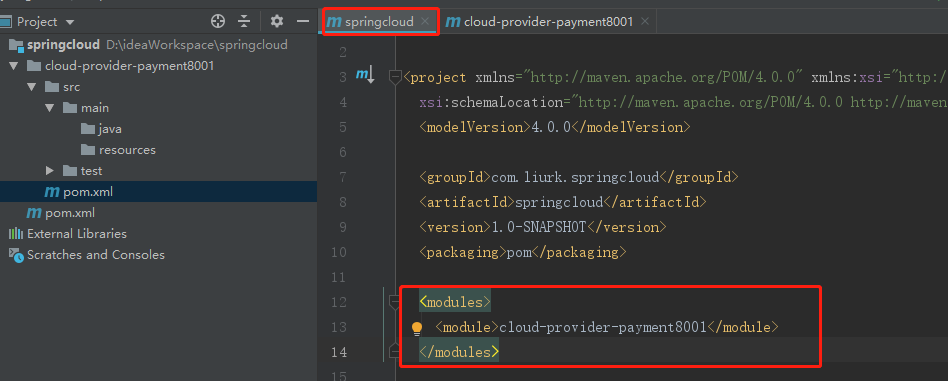


## 新建支付模块payment

### 创建module



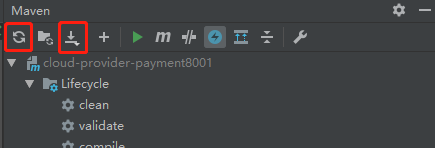
创建好后再查看父工程的pom.xml，多了modules模块



### 修改pom文件

### 编写application.yml

如果driver-class-name: com.mysql.jdbc.Driver报错的话，可能是jar包没有下载下来，需要手动下载和刷新一下依赖



### 启动类

### 业务类

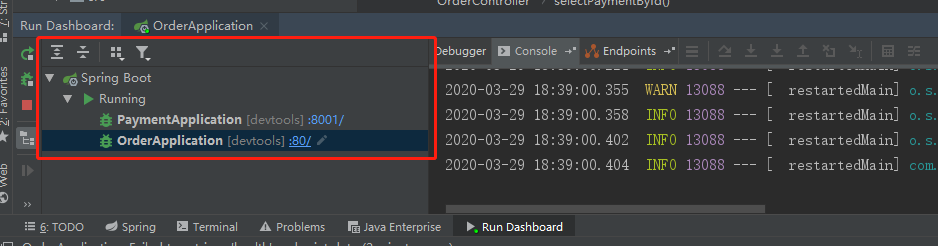
## 消费者订单模块

cloud-cousumer-order80

# RunDashboard

当微服务多了之后，为了方便查看微服务，需要使用RunDashboard

当启动多个微服务后RunDashboard会自动出现

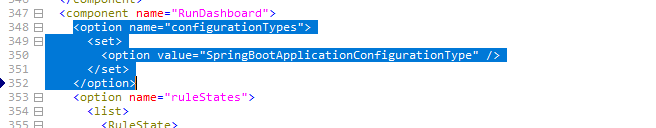


如果没有自动出现的话，修改workspace.xml文件(.idea下面)



添加如下配置

|  |
| --- |
| <option name="configurationTypes">  <set>  <option value="SpringBootApplicationConfigurationType" />  </set>  </option> |



添加后重启idea，就会出现RunDashboard，方便微服务的启动管理

可以右键debug启动对应的微服务

# 工程重构-提取公共内容到common模块

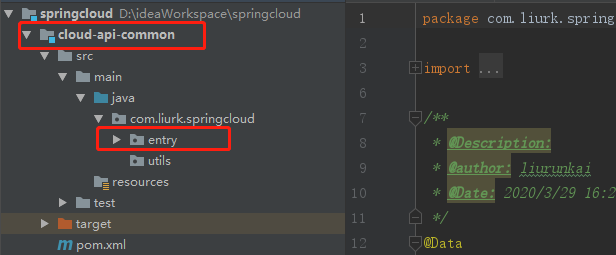
## 新建cloud-api-common模块

用于存放公共的东西：实体类、工具类等

## 修改pom文件

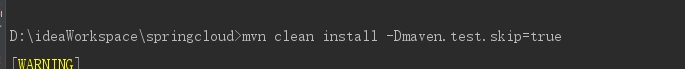
## cloud-api-common模块添加公共内容

将支付模块cloud-provider-payment8001和订单模块cloud-cousumer-order80的entry拷贝到cloud-api-common模块



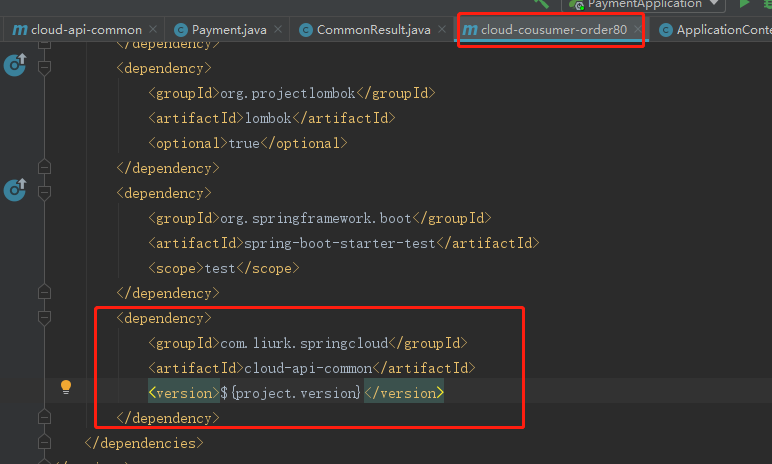
## 将cloud-api-common模块clean install

mvn clean install -Dmaven.test.skip=true



## 支付模块和订单模块改造

1. 将支付模块cloud-provider-payment8001和订单模块cloud-cousumer-order80的entry删除
2. 在对应的pom.xml文件中添加cloud-api-common依赖



## 再次使用postman测试

测试支付模块cloud-provider-payment8001

和订单模块cloud-cousumer-order80

# 服务注册-Eureka

1. 订单
2. 订单
3. p15