# 目录

[目录 1](#_Toc76067399)

[一、创建maven工程 2](#_Toc76067400)

[二、创建dubbo-nacos-mybatisplus-provider子模块 3](#_Toc76067401)

[2.1 创建 3](#_Toc76067402)

[2.2 添加依赖 5](#_Toc76067403)

[2.3 application.yml配置 5](#_Toc76067404)

[2.4 mybatisplus逆向工程 6](#_Toc76067405)

[2.5 修改代码 7](#_Toc76067406)

[三、创建dubbo-nacos-mybatisplus-api子模块 8](#_Toc76067407)

[3.1 创建 8](#_Toc76067408)

[3.2 添加java代码 9](#_Toc76067409)

[四、创建dubbo-nacos-mybatisplus-consumer子模块 11](#_Toc76067410)

[4.1 创建 11](#_Toc76067411)

[4.2 添加依赖 13](#_Toc76067412)

[4.3 application.yml配置 13](#_Toc76067413)

[4.4 添加java代码 14](#_Toc76067414)

[五、模块之间的依赖配置 15](#_Toc76067415)

[5.1 模块打包 15](#_Toc76067416)

[5.2 添加依赖 16](#_Toc76067417)

[六、添加代码 17](#_Toc76067418)

[6.1 dubbo-nacos-mybatisplusr-api接口声明函数 17](#_Toc76067419)

[6.2 dubbo-nacos-mybatisplusr-provider函数实现 17](#_Toc76067420)

[6.2.1 PersonDao函数声明以及PersonDao.xml的sql 17](#_Toc76067421)

[6.2.2 PersonServiceImpl接口函数实现 18](#_Toc76067422)

[6.3 dubbo-nacos-mybatisplusr-consumer调用 18](#_Toc76067423)

[项目结构 19](#_Toc76067424)

[项目运行 20](#_Toc76067425)

[1. 启动nacos 20](#_Toc76067426)

[2. 运行启动类 20](#_Toc76067427)

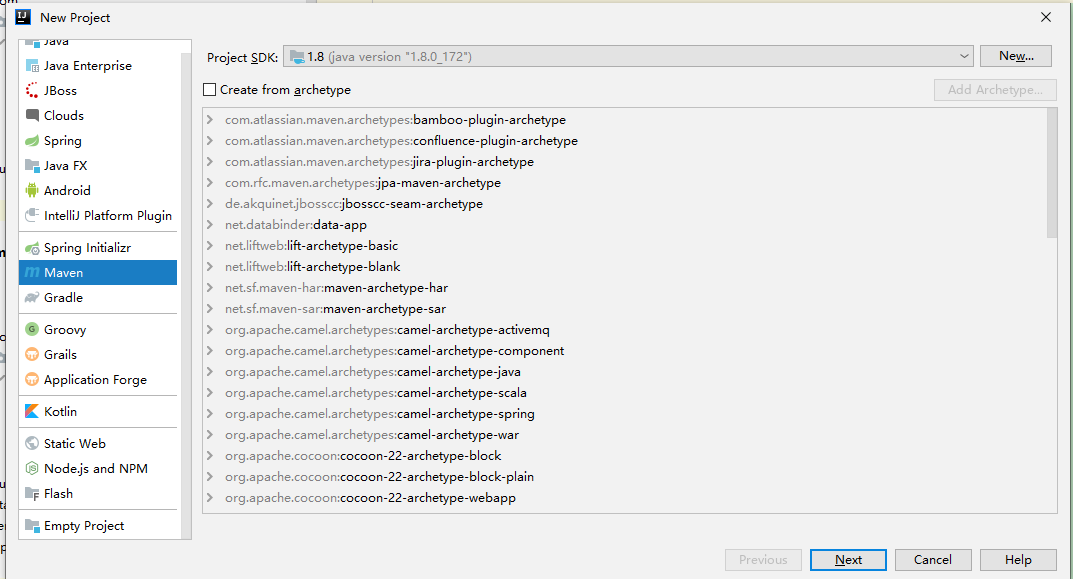
[3. 测试 20](#_Toc76067428)

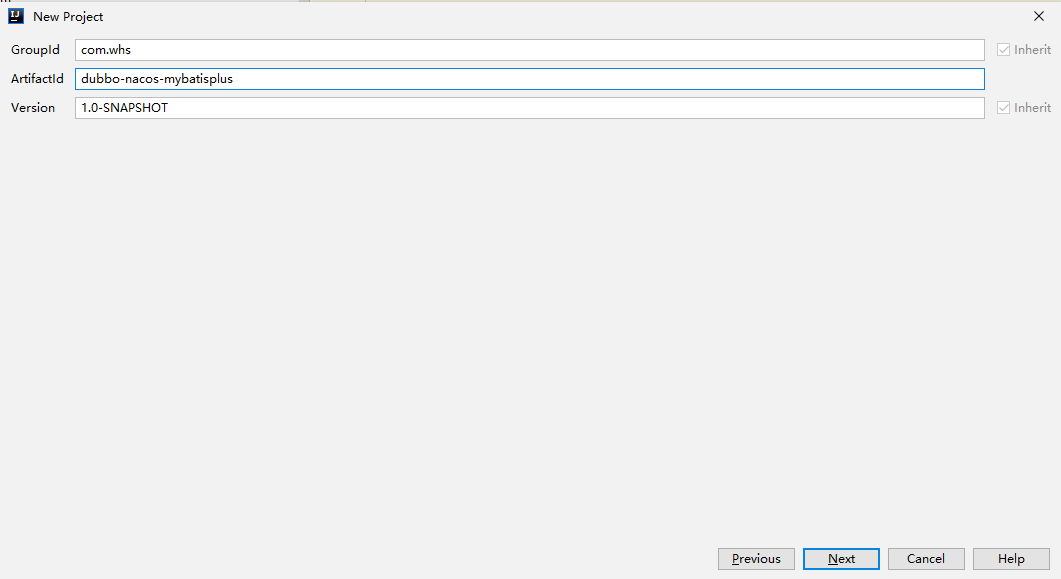
[附件 22](#_Toc76067429)

[附件1 dubbo-nacos-mybatisplusr-provider子模块的依赖 22](#_Toc76067430)

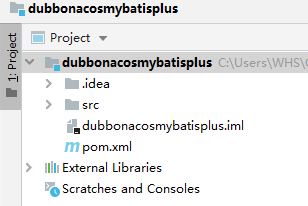
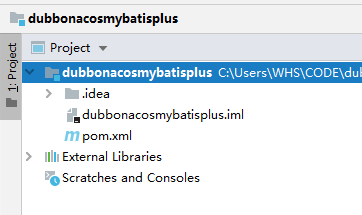
[附件2 CodeGeneration代码 23](#_Toc76067431)

# 一、创建maven工程

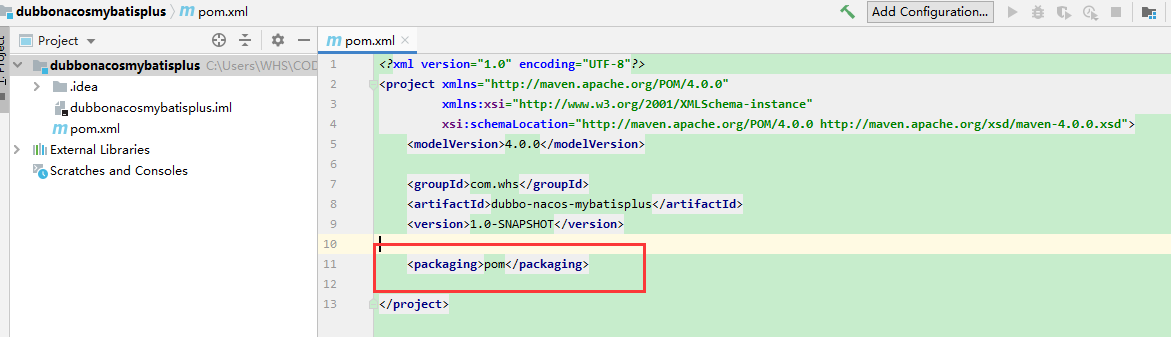




项目创建成功后，项目结构如下左图。删除src包，项目结构如下右图。

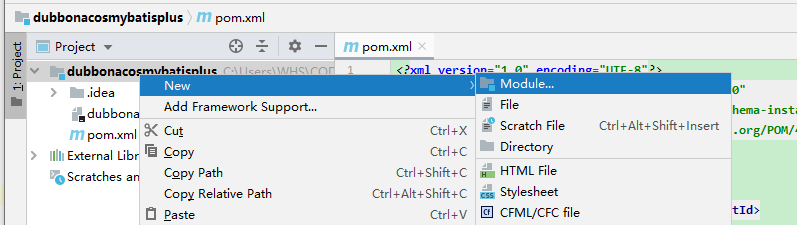
设置工程pom文件的打包方式为pom，如下：

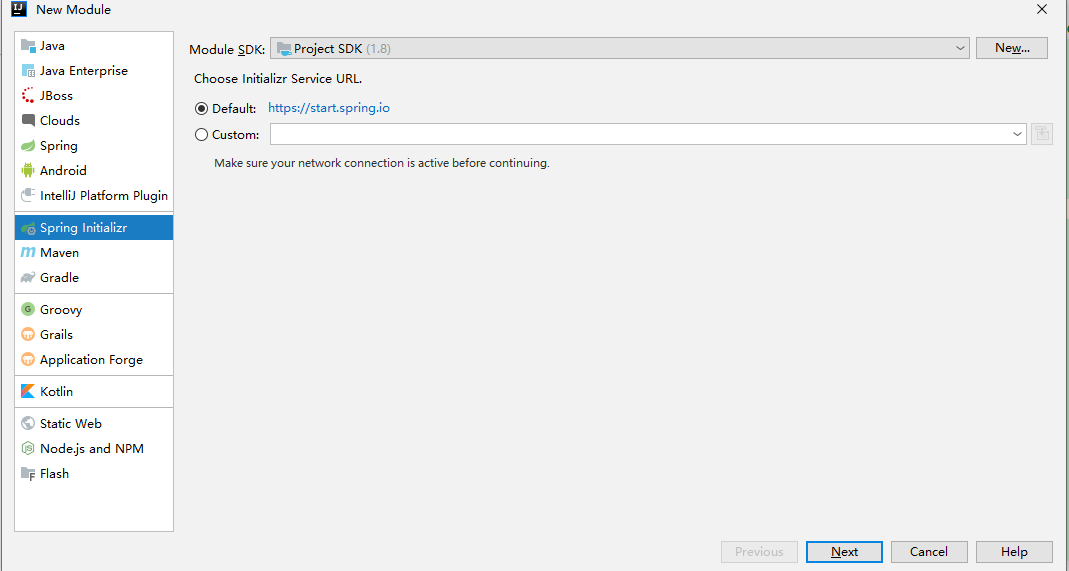


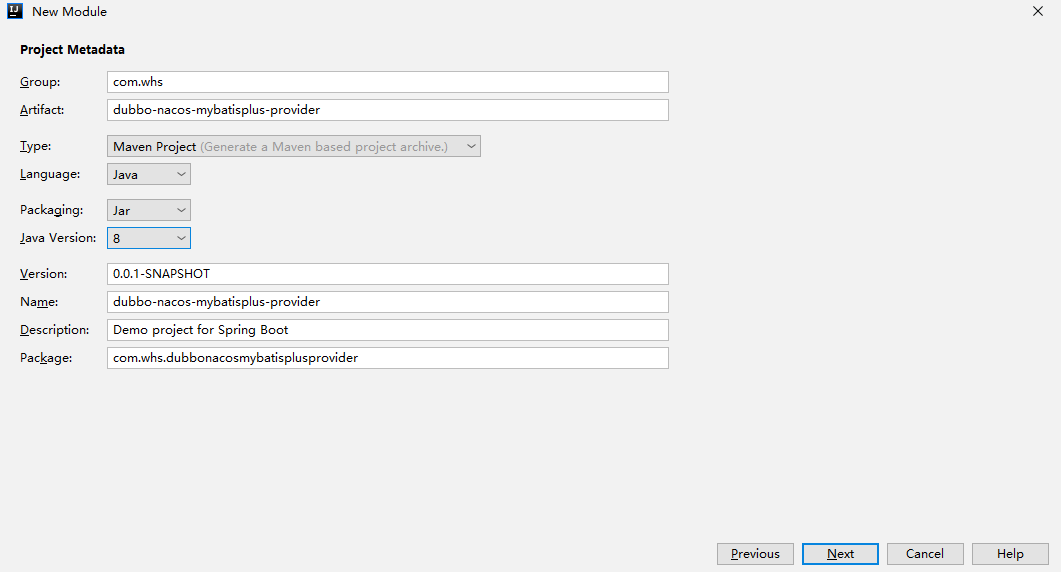
# 二、创建dubbo-nacos-mybatisplus-provider子模块

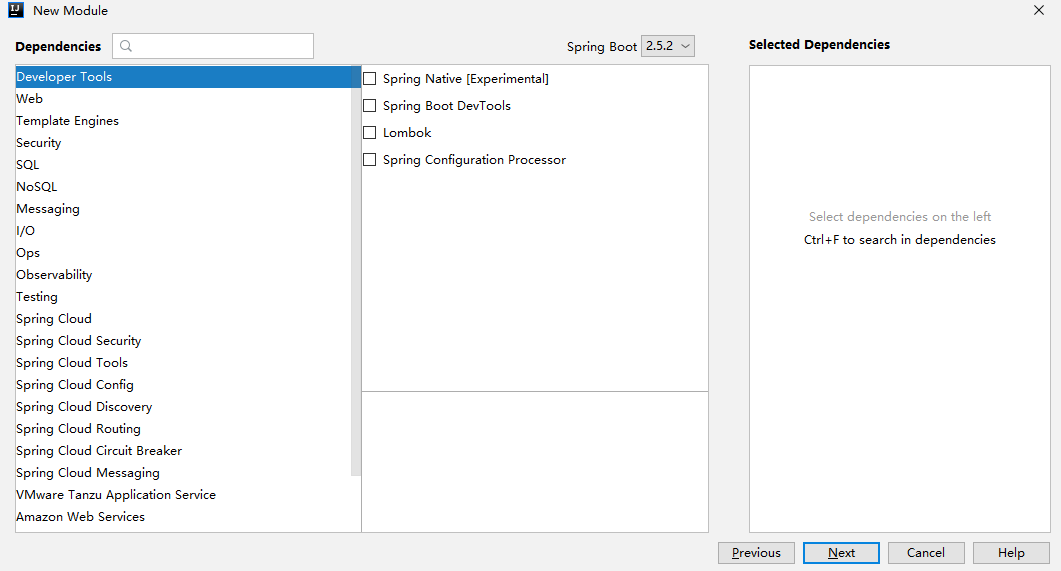
## 2.1 创建

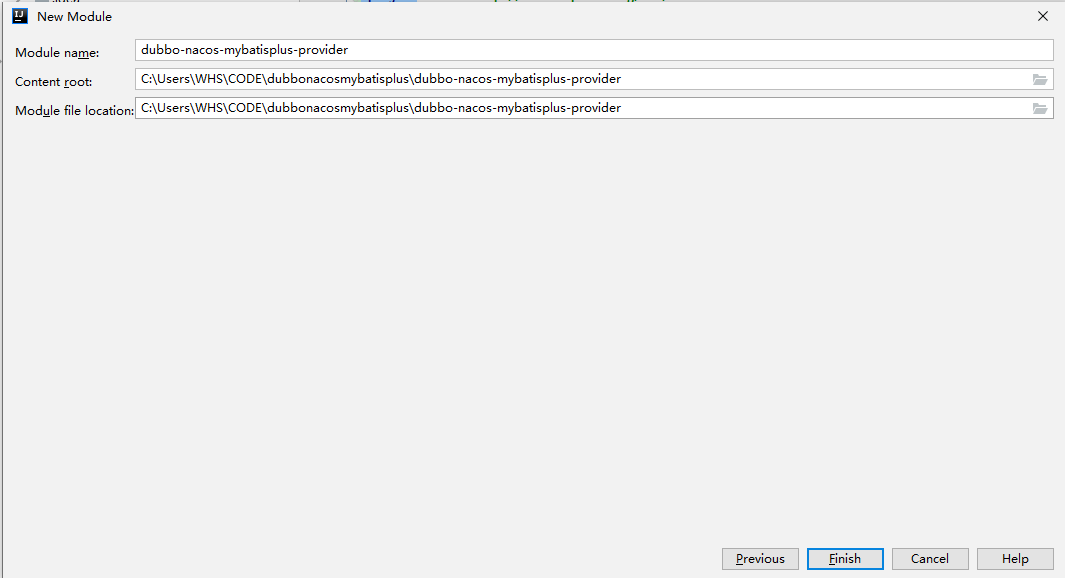
右击本工程，new—>Module，如下所示：（创建springboot项目）





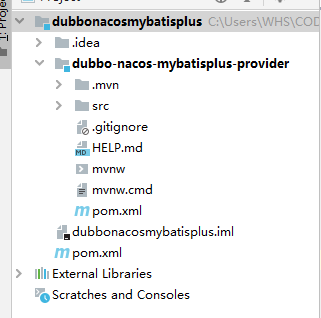
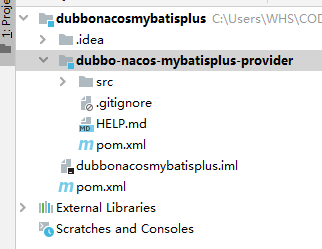






最后一步，点击finish，完成该模块的创建。

创建该模块后的项目结构如下左图，删除“.mvn”、“mvnw”和“mvnw.cmd”文件，如下右图。

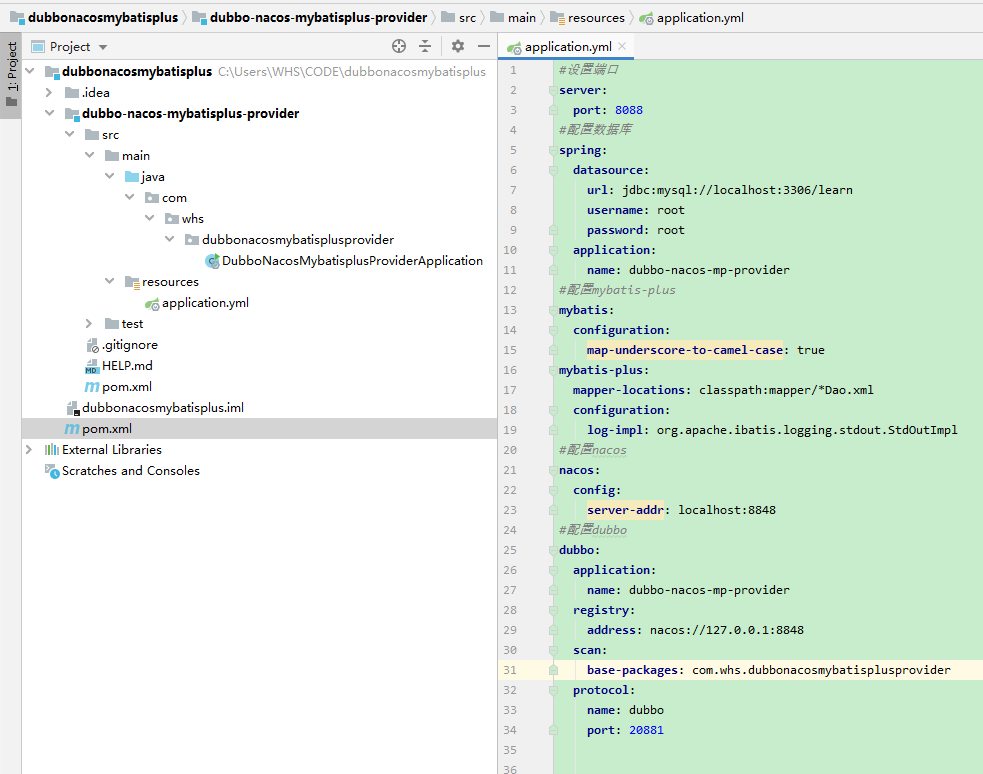
 

## 2.2 添加依赖

在dubbo-nacos-mybatisplusr-provider子模块的pom文件中，添加依赖，如[附件1](#_附件1_dubbo-nacos-mybatisplusr-provid)所示。其中，添加的依赖包括：mysql、mybatis-plus、nacos、dubbo等依赖包。

## 2.3 application.yml配置

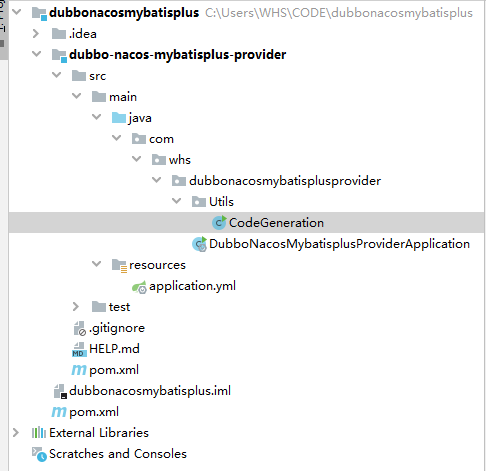
dubbo-nacos-mybatisplusr-provider模块的application.yml的配置如下：



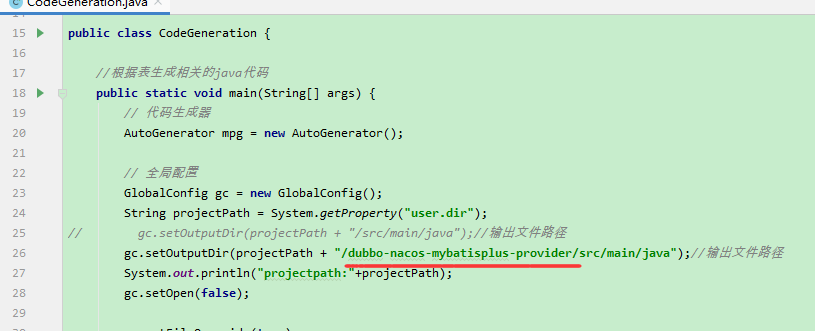
## 2.4 mybatisplus逆向工程

逆向工程的代码CodeGeneration见[附件2](#_附件2_CodeGeneration代码)。

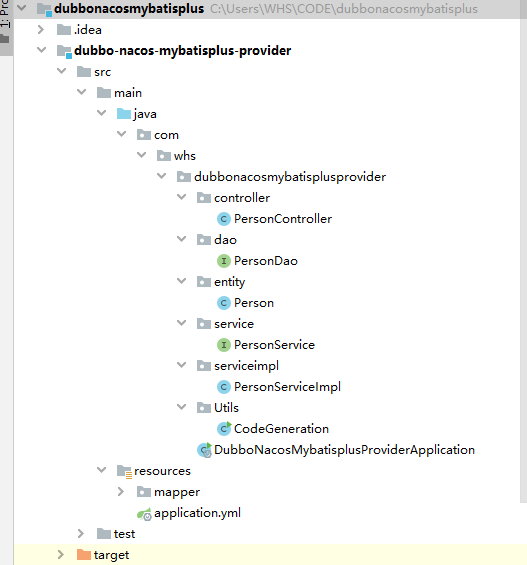
添加逆向生成代码后的模块结构如下：



其中，要注意的是，在生成的java的路径中，要加上模块的名称，如下：



在CodeGeneration中配置好信息以后，直接运行该类的主函数，最后得到的项目结构如下：

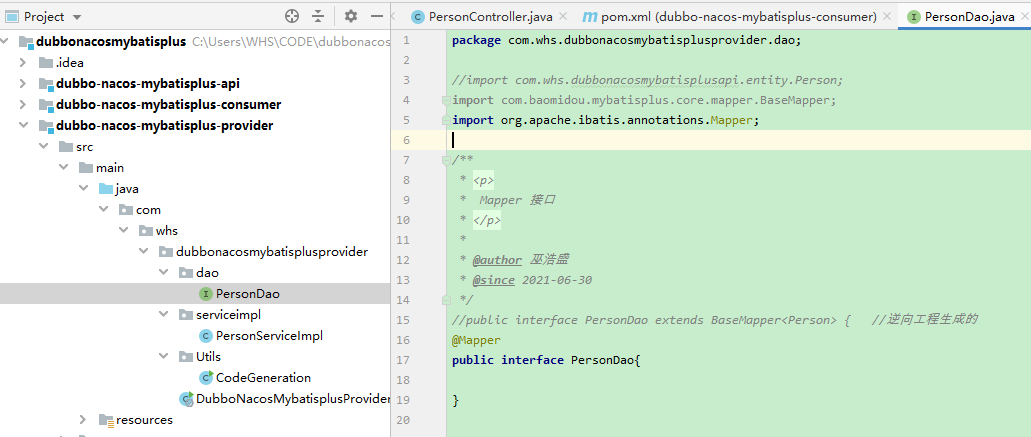


注：接下来会新建api模块和consumer模块，生成的entity和service会移到api模块，controller会移到consumer模块。

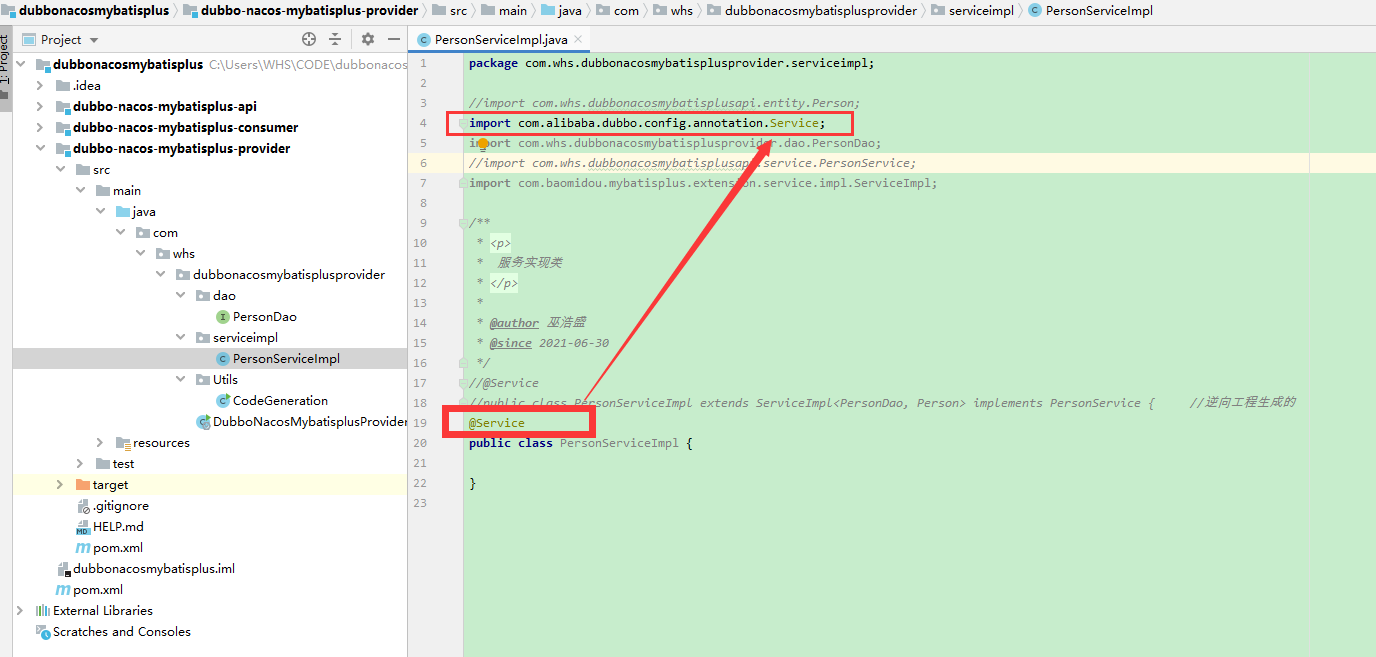
## 2.5 修改代码

对PersonDao和PersonServiceImpl进行了改动。具体如下：

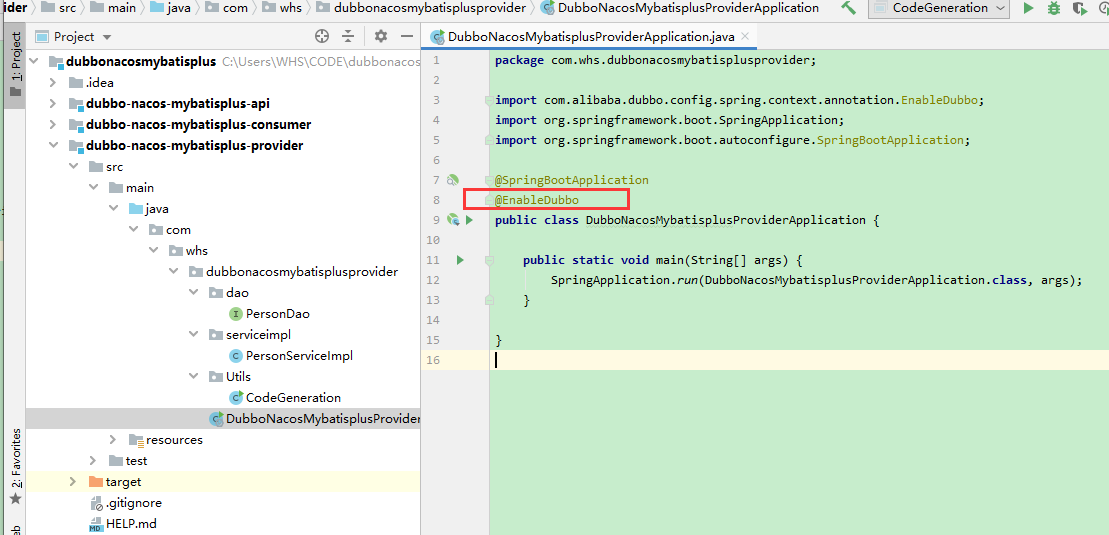
PersonDao的修改如下，并加上@Mapper注解（不加会导致启动项目无法扫描到）



PersonServiceImpl修改如下，并将@Service换成dubbo的@Service注解。



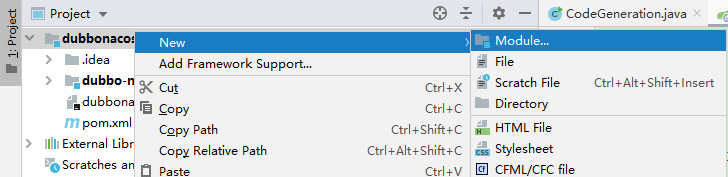
在主类中，加上@EnableDubbo注解，如下所示：

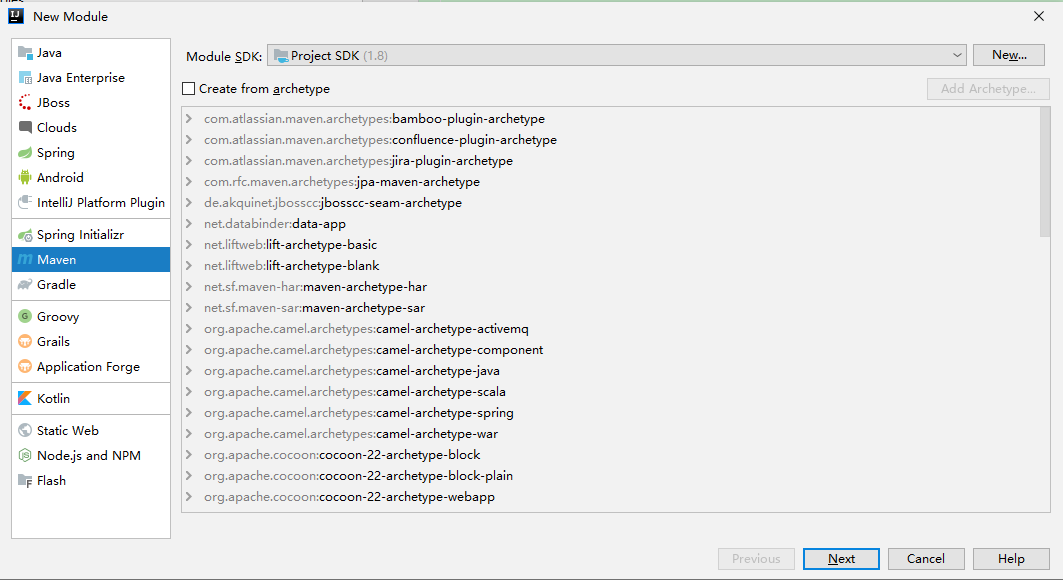


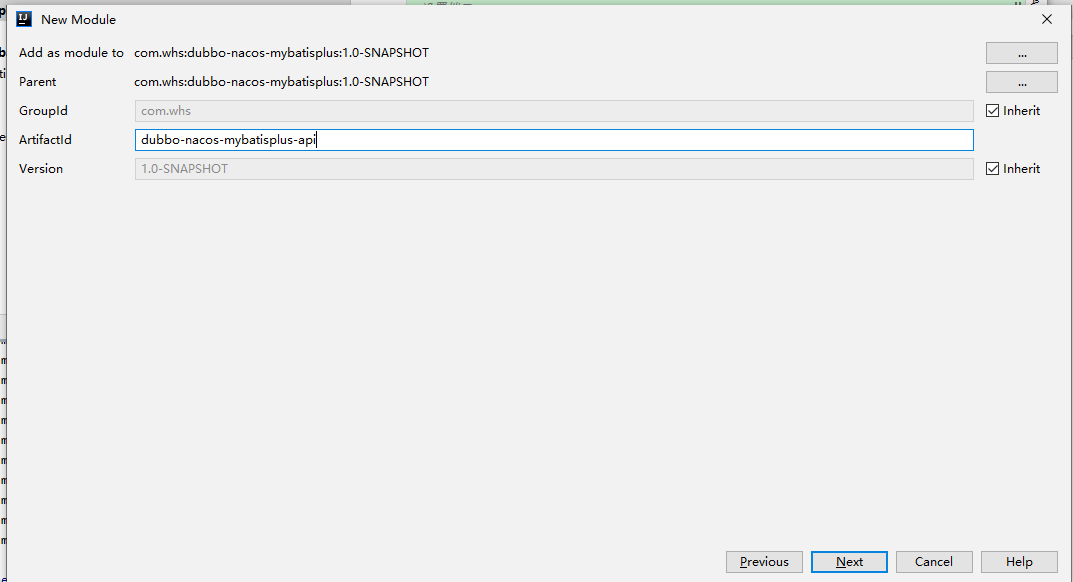
# 三、创建dubbo-nacos-mybatisplus-api子模块

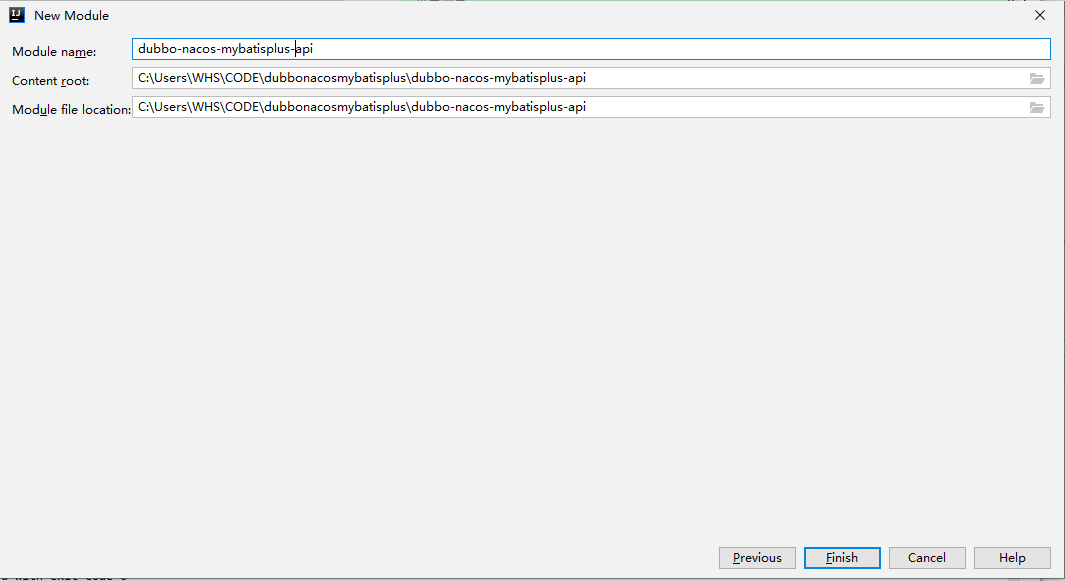
## 3.1 创建

右击本工程，new—>Module，如下所示：（创建maven项目）





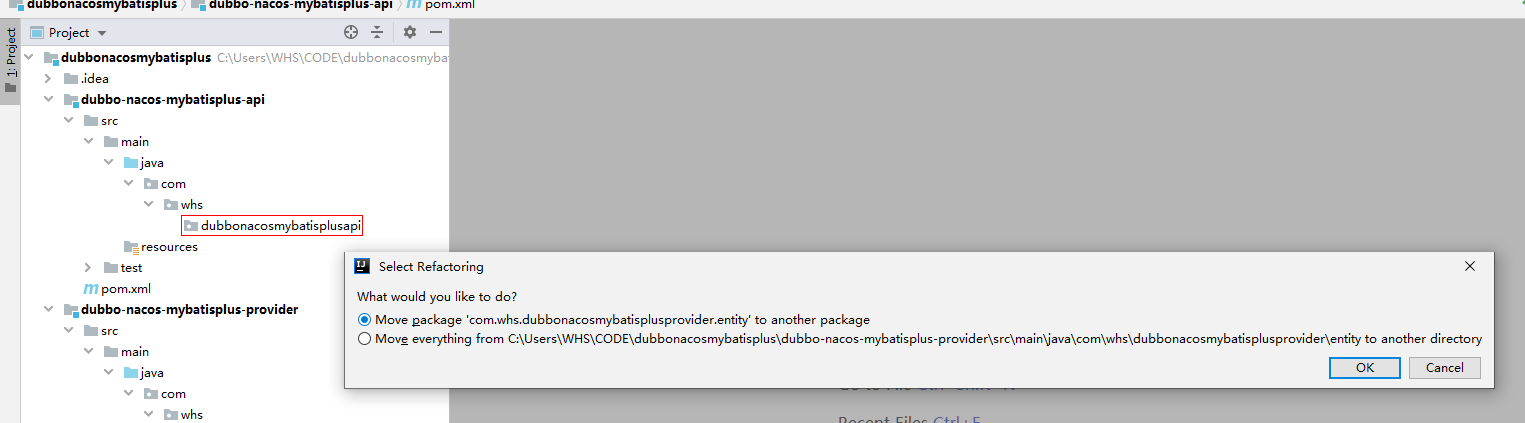




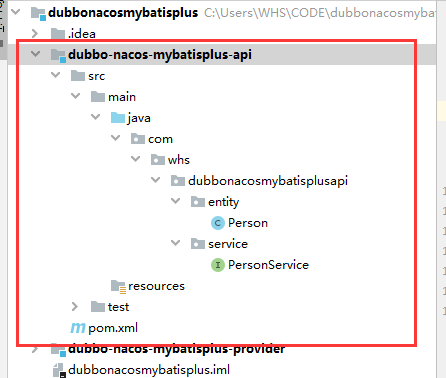
点击finish完成dubbo-nacos-mybatisplus-api子模块的创建。

## 3.2 添加java代码

将dubbo-nacos-mybatisplus-provider模块逆向工程生成的entity和service代码移动到dubbo-nacos-mybatisplus-api模块下，如下所示。

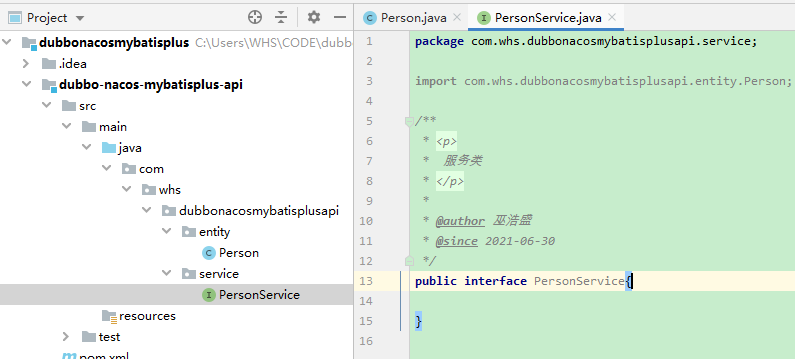
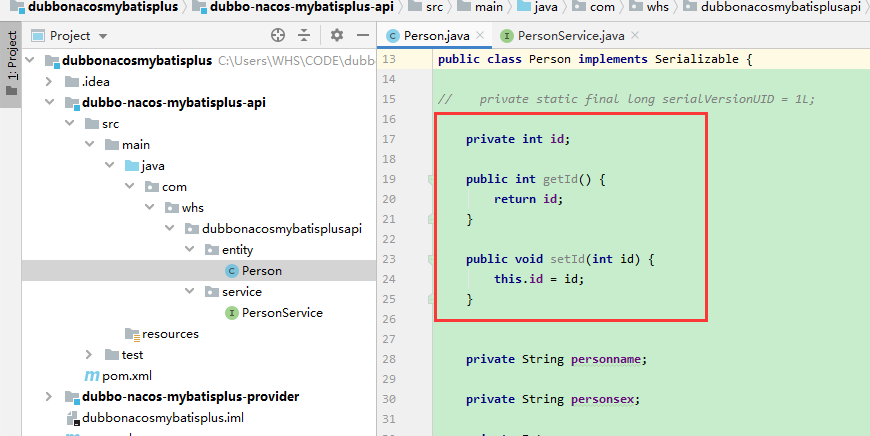


该模块结构如下图所示：



此处改动了Person实体和PersonService的代码，分别下图。

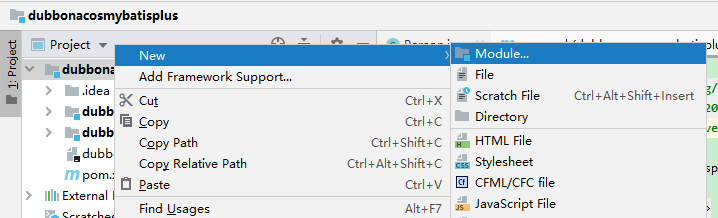
Person这个类注释掉了serialVersionUID，自己加入了id以及set和get方法。

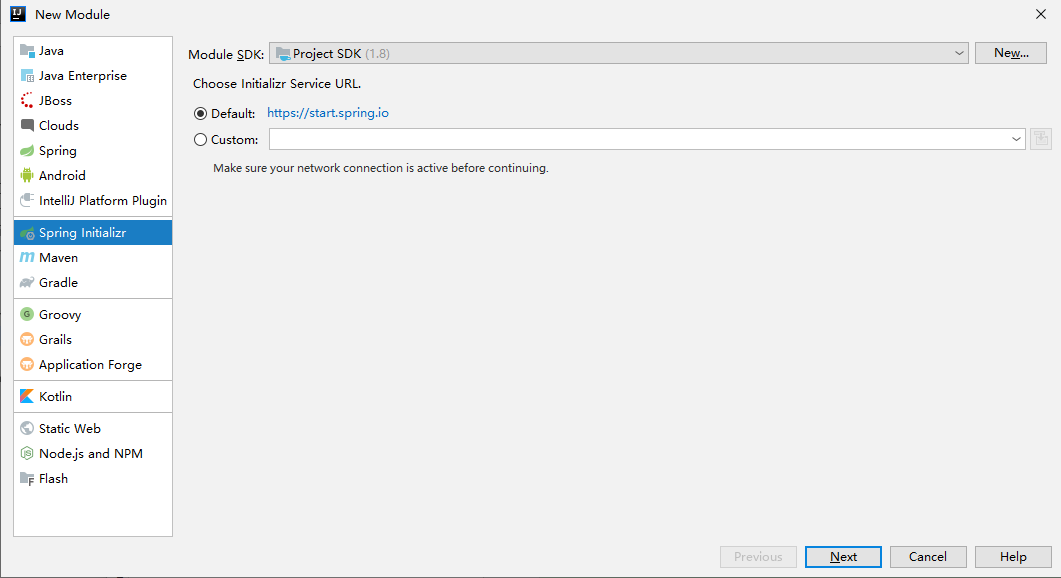


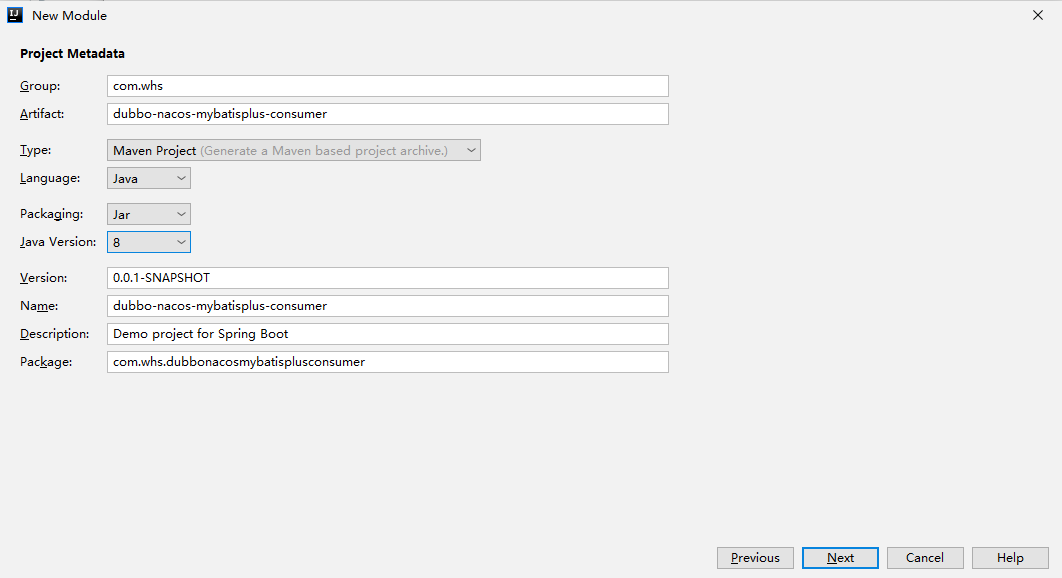
# 四、创建dubbo-nacos-mybatisplus-consumer子模块

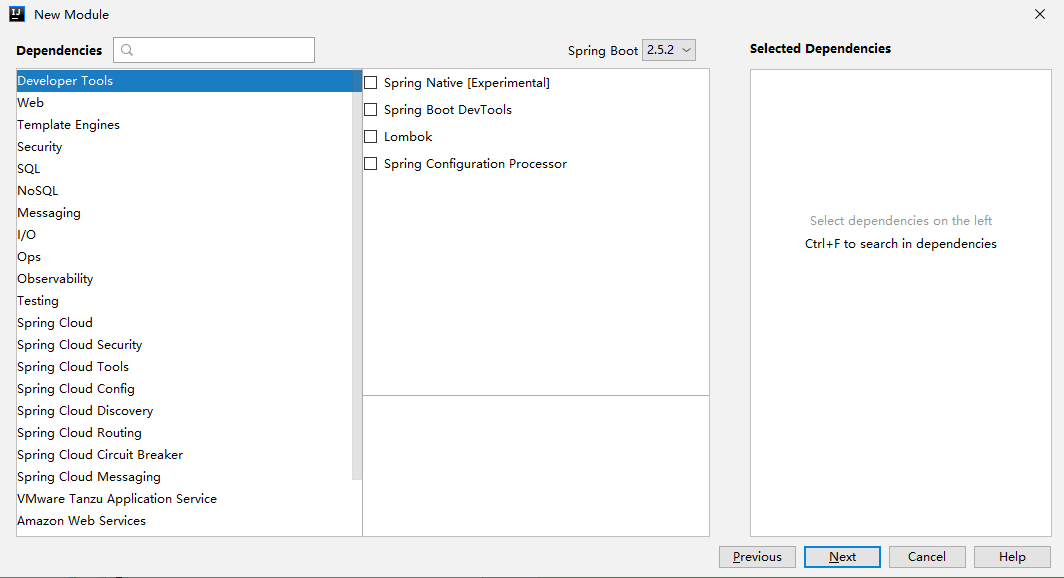
## 4.1 创建

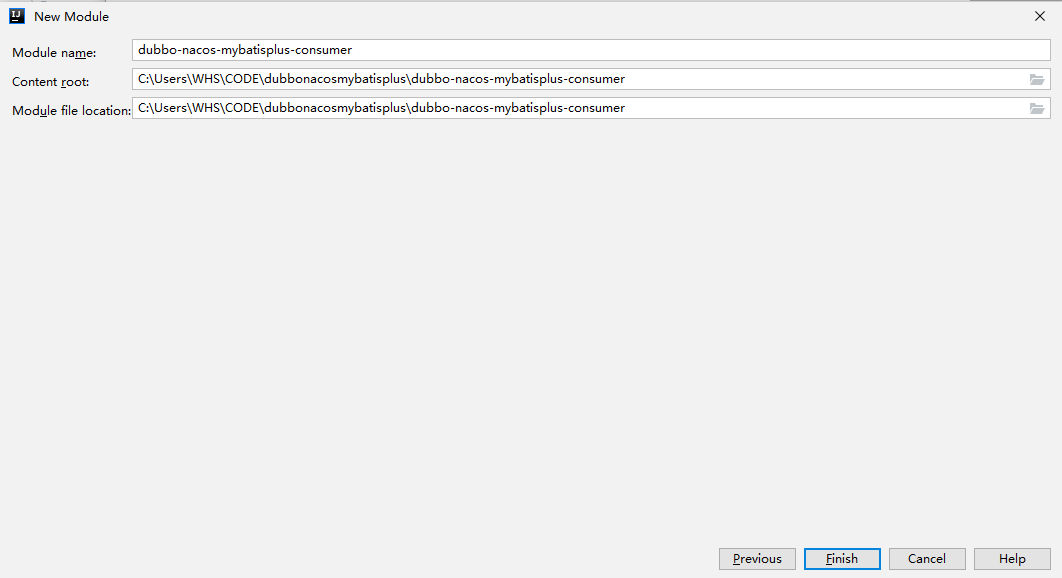
右击本工程，new—>Module，如下所示：（创建springboot项目）



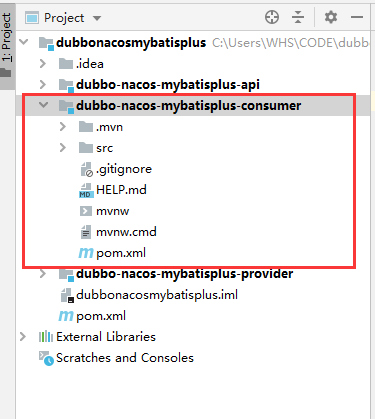
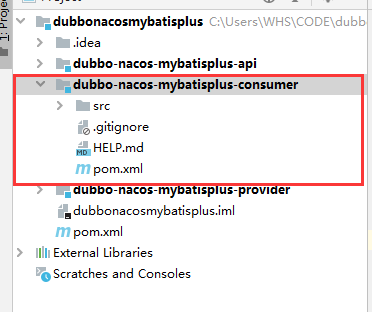








创建该模块后的项目结构如下左图，删除“.mvn”、“mvnw”和“mvnw.cmd”文件，如下右图。

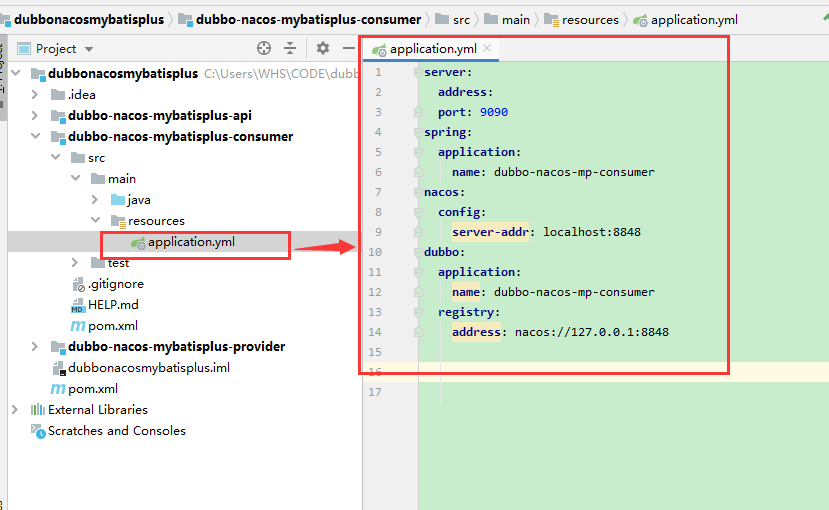
## 4.2 添加依赖

在dubbo-nacos-mybatisplusr-consumer子模块的pom文件中，添加依赖，所有依赖如下。

<**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**> *<!-- Dubbo Nacos registry dependency -->* <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>dubbo-registry-nacos</**artifactId**>  
 <**version**>0.0.1</**version**>  
 </**dependency**>  
 *<!-- Dubbo dependency -->* <**dependency**>  
 <**groupId**>com.alibaba.boot</**groupId**>  
 <**artifactId**>dubbo-spring-boot-starter</**artifactId**>  
 <**version**>0.2.0</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.apache.curator</**groupId**>  
 <**artifactId**>curator-framework</**artifactId**>  
 <**version**>2.12.0</**version**>  
 </**dependency**>  
 </**dependencies**>

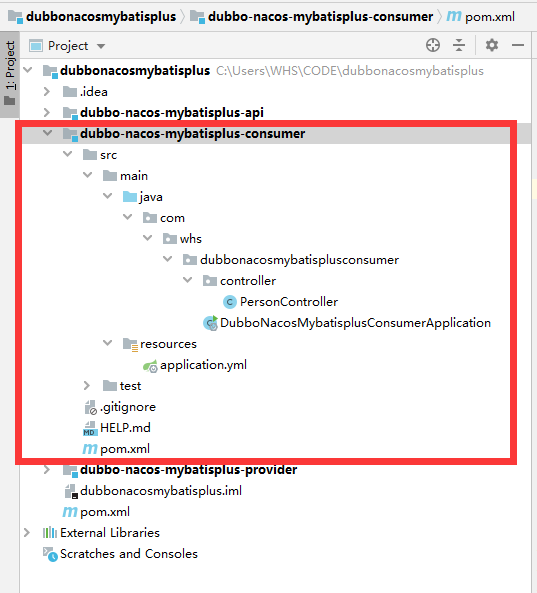
## 4.3 application.yml配置

dubbo-nacos-mybatisplusr-consumer模块的application.yml的配置如下：



## 4.4 添加java代码

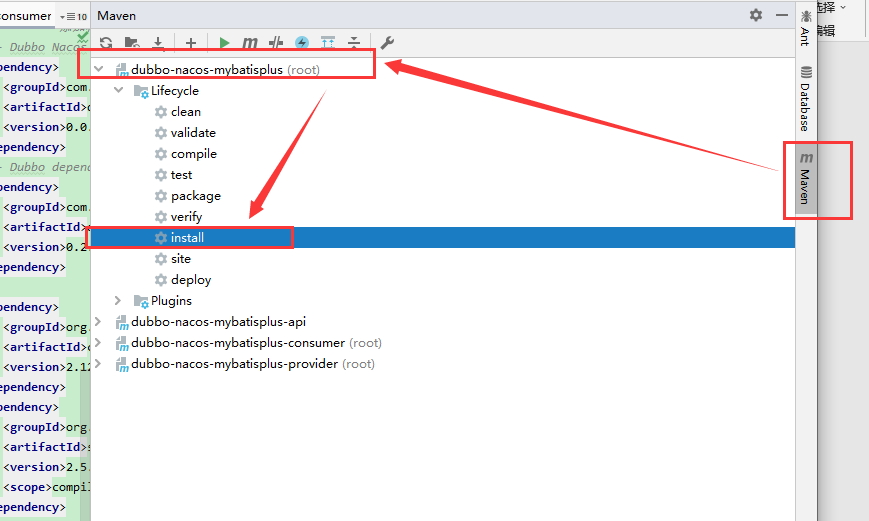
将dubbo-nacos-mybatisplus-provider模块逆向工程生成的controller代码移动到dubbo-nacos-mybatisplus-consumer模块下，操作与dubbo-nacos-mybatisplus-api模块添加java代码的操作一样，最后的结构如下所示。



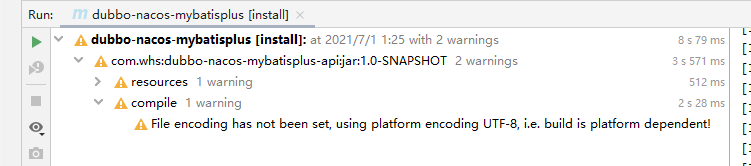
# 五、模块之间的依赖配置

## 5.1 模块打包

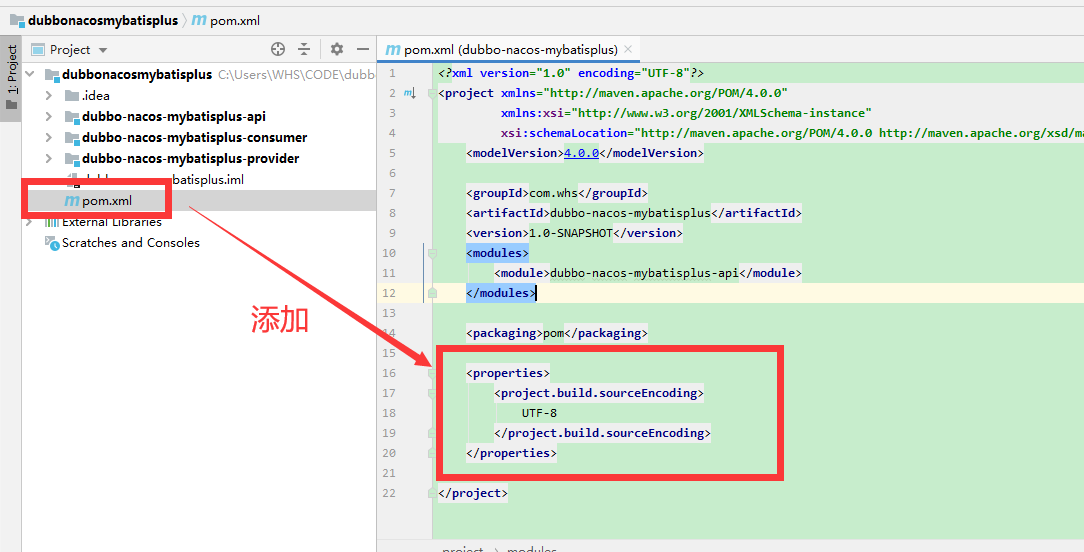
如下所示，直接对整个工程打包。最后双击install等待打包完成。



其中，可能会遇到以下警告（如果没遇到一下问题则不用理会，忽略以下步骤即可）。

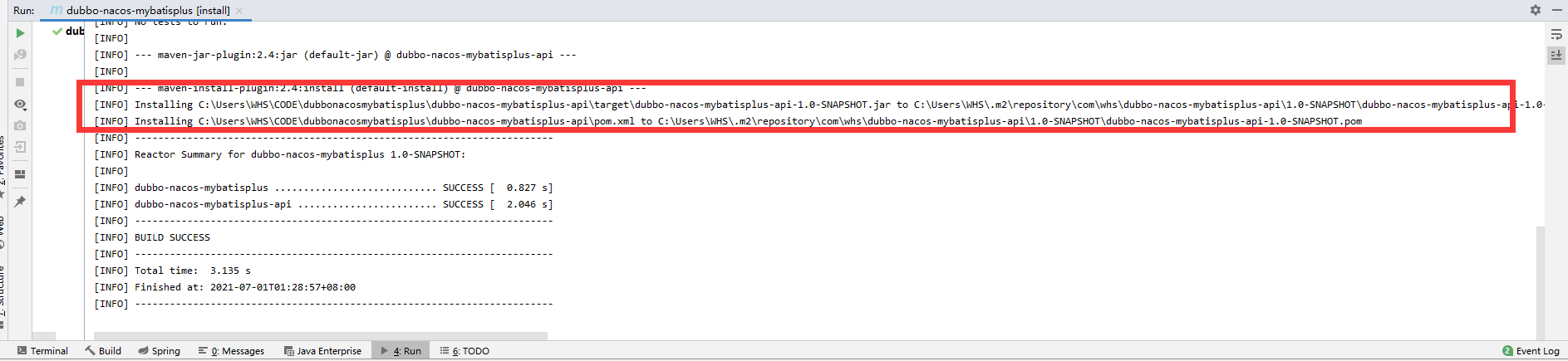


直接在本工程的pom文件添加以下即可，如下：



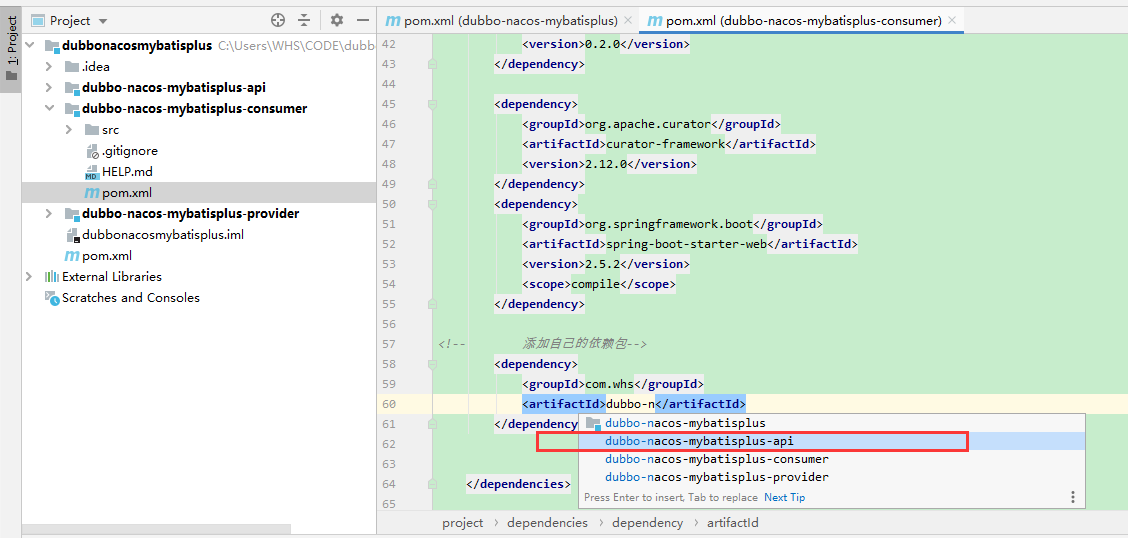
重新install即可。

如下所示，可以看到打包的信息，以及打包之后，文件所在的位置。

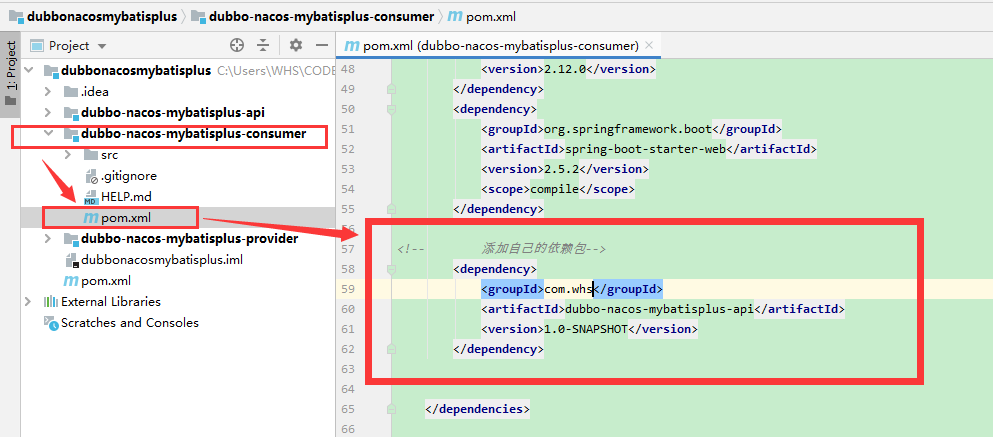


## 5.2 添加依赖

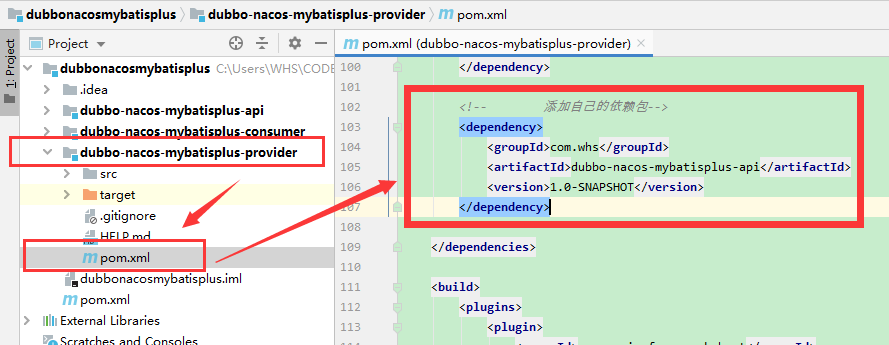
这一步是为我们自己的写的模块添加自己写的接口的依赖包。



可以看到给出了提示，表明自己写的东西已经被打包好，添加即可。如下所示，是在dubbo-nacos-mybatisplusr-consumer模块中的pom文件添加依赖。

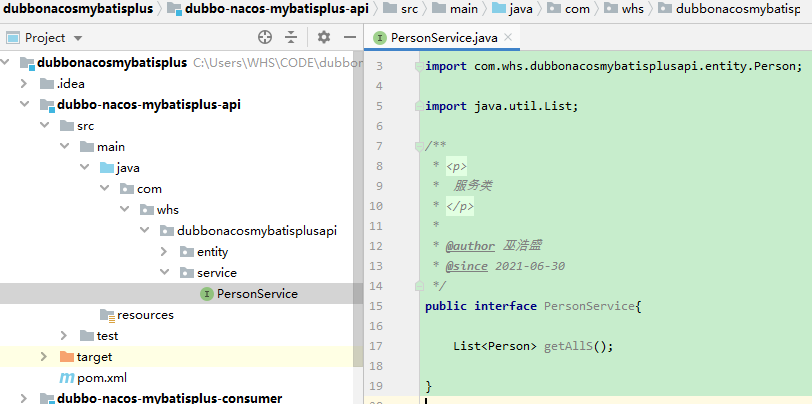


对dubbo-nacos-mybatisplusr-provider模块的pom文件添加同样的依赖包。如下所示：



# 六、添加代码

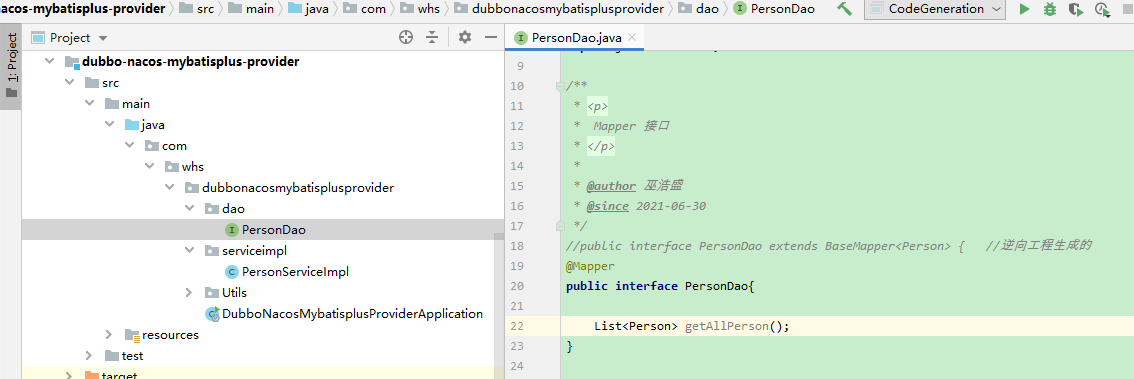
## 6.1 dubbo-nacos-mybatisplusr-api接口声明函数



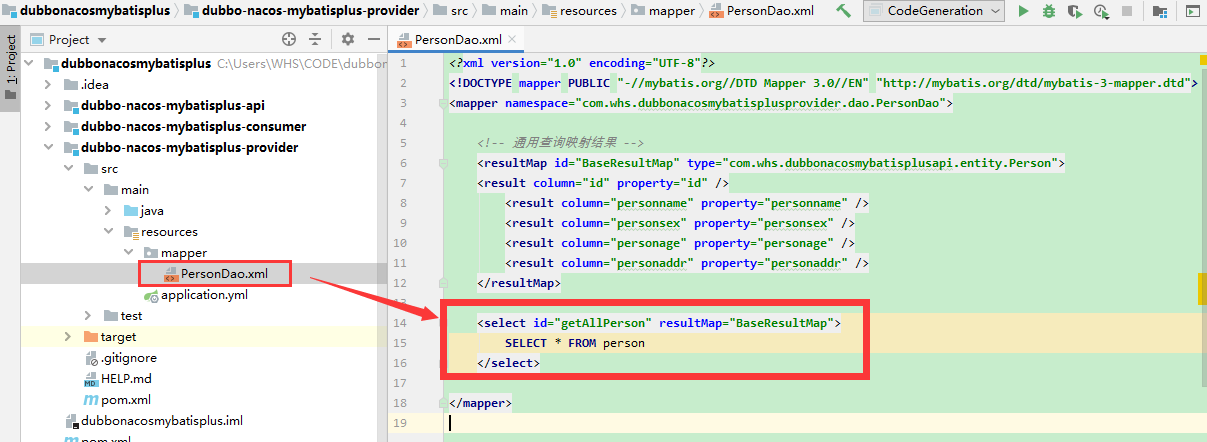
## 6.2 dubbo-nacos-mybatisplusr-provider函数实现

### 6.2.1 PersonDao函数声明以及PersonDao.xml的sql

PersonDao函数声明如下：

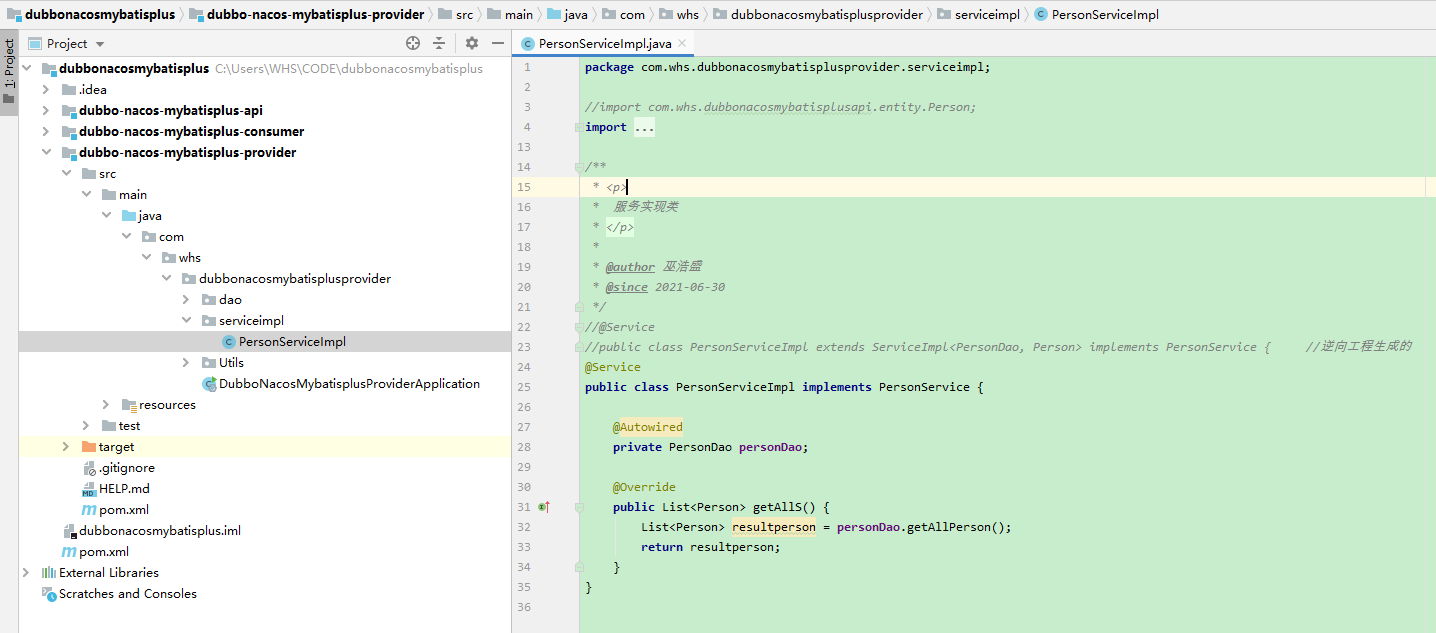


PersonDao.xml对应sql如下：



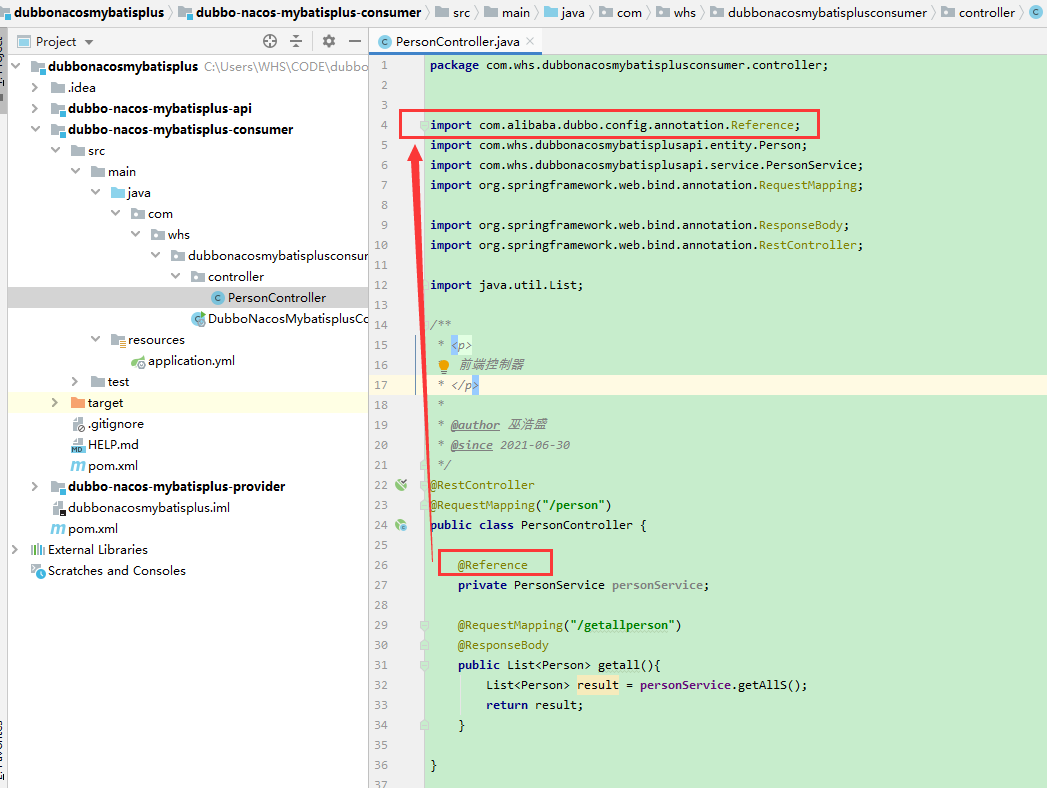
### 6.2.2 PersonServiceImpl接口函数实现

PersonServiceImpl的实现如下：



## 6.3 dubbo-nacos-mybatisplusr-consumer调用

PersonController的实现如下：

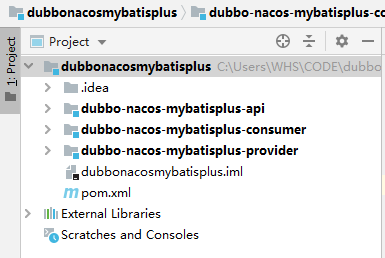


注意：此处的@Reference注解需要使用dubbo提供的。

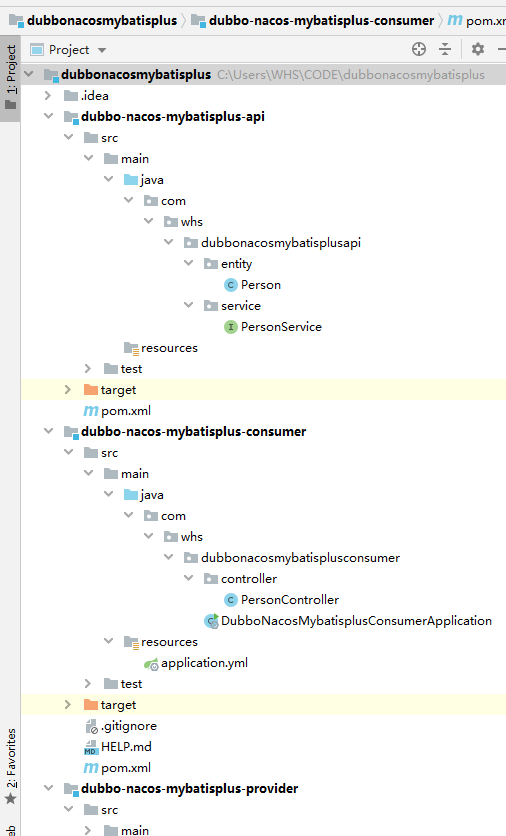
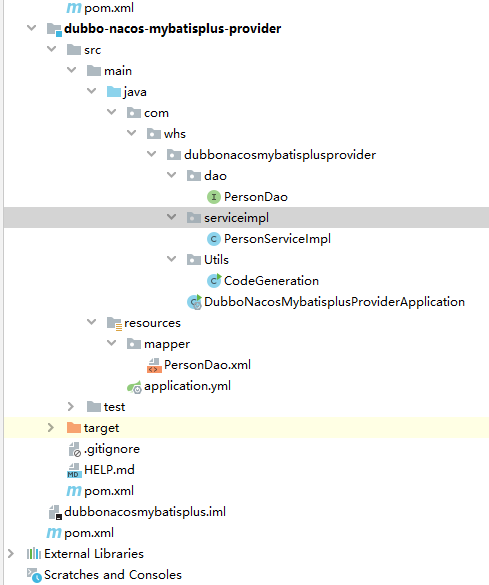
# 项目结构

至此，项目就搭建完成，最终的项目结构如下所示：

项目总结构：



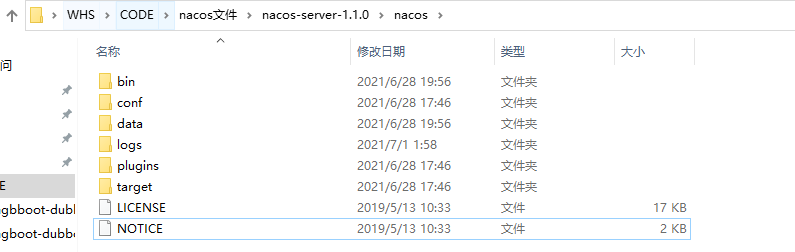
项目详细结构：

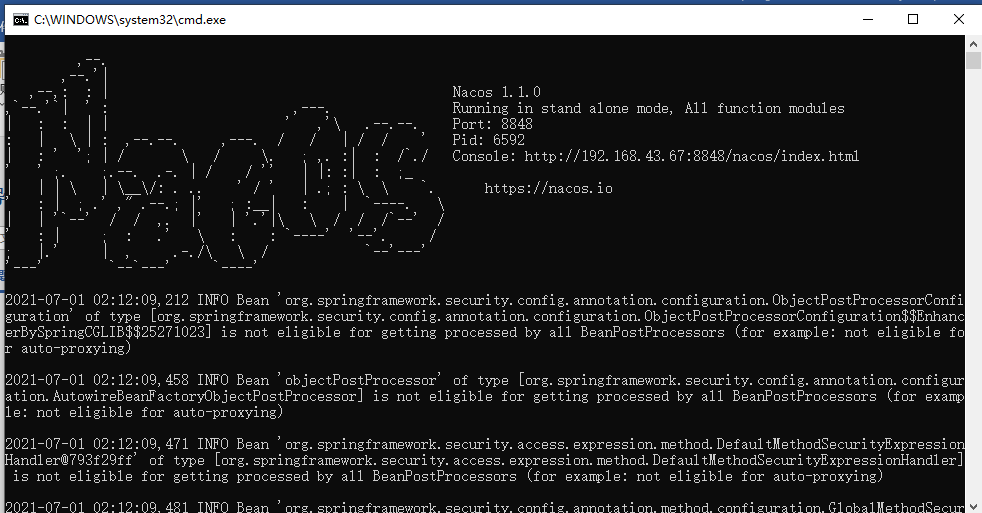
# 项目运行

## 1. 启动nacos

下载nacos的压缩包，解压得到以下文件：



进入bin目录下，双击startup.cmd即可启动nacos，如下所示：

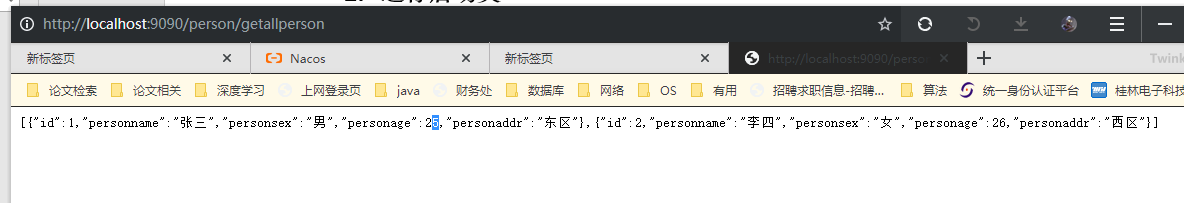


## 2. 运行启动类

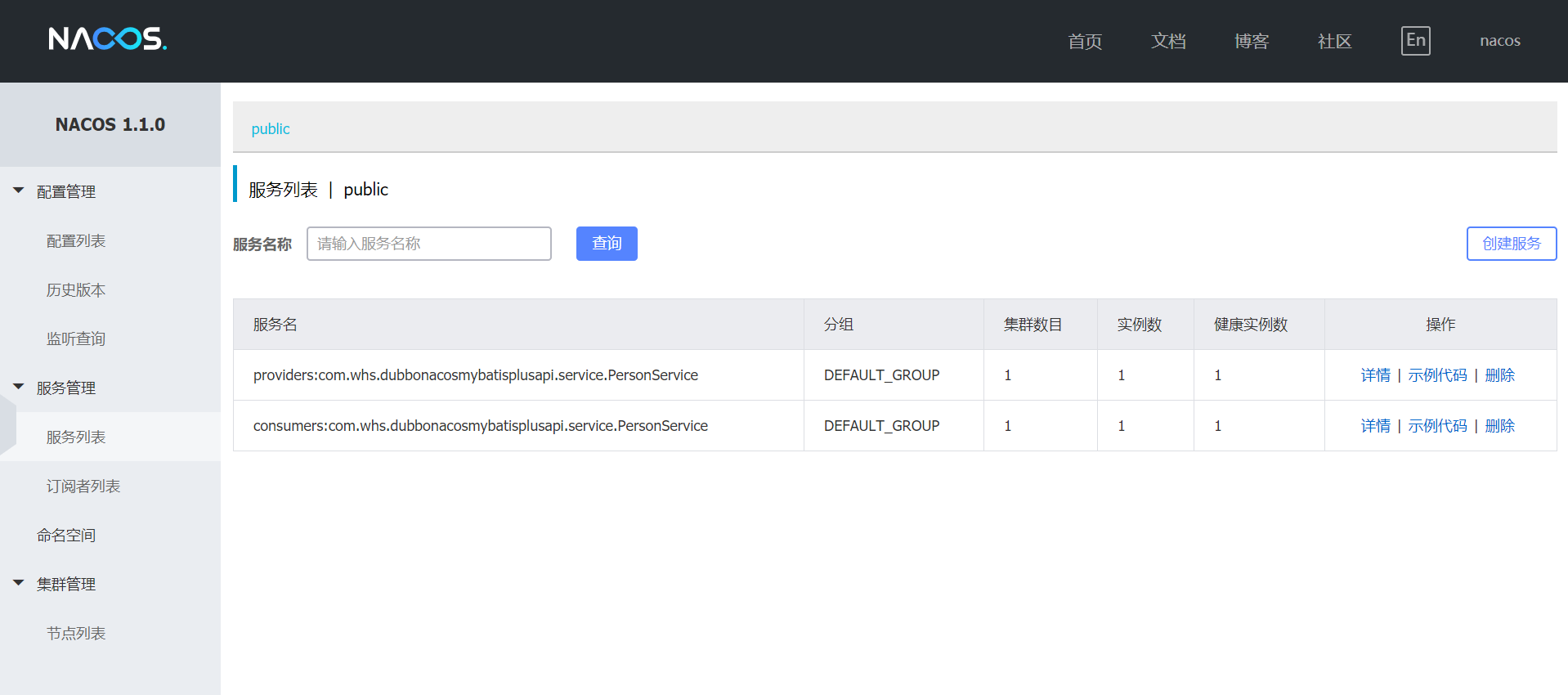
依次运行dubbo-nacos-mybatisplus-provider和dubbo-nacos-mybatisplus-consumer模块的主类。

## 3. 测试

在浏览器中输入如下：<http://localhost:9090/person/getallperson>，得到如下：



其中，nacos后台管理的服务列表如下：（在浏览器中输入<http://localhost:8848/nacos>即可，然后输入用户名和密码，均为nacos）



# 附件

## 附件1 dubbo-nacos-mybatisplusr-provider子模块的依赖

dubbo-nacos-mybatisplusr-provider子模块的pom文件中需要添加的依赖。

*<!-- mysql-->*<**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**scope**>runtime</**scope**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
</**dependency**>  
*<!-- 这是lombok的依赖 -->*<**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 <**optional**>true</**optional**>  
</**dependency**>  
*<!-- 这是mybatis-plus依赖 -->*<**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**version**>3.1.1</**version**>  
</**dependency**>  
*<!-- 这是mybatis-plus的代码自动生成器 -->*<**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-generator</**artifactId**>  
 <**version**>3.1.1</**version**>  
</**dependency**>  
*<!-- 这是模板引擎依赖 -->*<**dependency**>  
 <**groupId**>org.freemarker</**groupId**>  
 <**artifactId**>freemarker</**artifactId**>  
 <**version**>2.3.28</**version**>  
</**dependency**>  
*<!-- 添加相应依赖-->*<**dependency**>  
 <**groupId**>com.alibaba.nacos</**groupId**>  
 <**artifactId**>nacos-client</**artifactId**>  
 <**version**>1.0.1</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>com.alibaba.spring</**groupId**>  
 <**artifactId**>spring-context-support</**artifactId**>  
 <**version**>1.0.2</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>io.netty</**groupId**>  
 <**artifactId**>netty-all</**artifactId**>  
 <**version**>4.1.32.Final</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>dubbo-registry-nacos</**artifactId**>  
 <**version**>2.6.7</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>com.alibaba.boot</**groupId**>  
 <**artifactId**>dubbo-spring-boot-starter</**artifactId**>  
 <**version**>0.2.0</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.apache.curator</**groupId**>  
 <**artifactId**>curator-framework</**artifactId**>  
 <**version**>2.12.0</**version**>  
</**dependency**>

## 附件2 CodeGeneration代码

CodeGeneration的代码如下：

public class CodeGeneration {  
 //根据表生成相关的java代码  
 public static void main(String[] args) {  
 AutoGenerator mpg = new AutoGenerator();  
 // 全局配置  
 GlobalConfig gc = new GlobalConfig();  
 String projectPath = System.getProperty("user.dir");  
 gc.setOutputDir(projectPath + "/dubbo-nacos-mybatisplus-provider/src/main/java");//输出文件路径  
 System.out.println("projectpath:"+projectPath);  
 gc.setOpen(false);  
 gc.setFileOverride(true);  
 gc.setActiveRecord(false); // 不需要ActiveRecord特性的请改为false  
 gc.setEnableCache(false); // XML 二级缓存  
 gc.setBaseResultMap(true); // XML ResultMap  
 gc.setBaseColumnList(false); // XML columList  
 gc.setAuthor("巫浩盛"); //设置作者  
 gc.setDateType(DateType.ONLY\_DATE);//设置生成时间  
 // 自定义文件命名，注意 %s 会自动填充表实体属性！  
 gc.setServiceName("%sService");  
 gc.setMapperName("%sDao");  
 mpg.setGlobalConfig(gc);  
 // 数据源配置  
 DataSourceConfig dsc = new DataSourceConfig();  
dsc.setUrl("jdbc:mysql://localhost:3306/learn?useUnicode=true&useSSL=false&characterEncoding=utf8&serverTimezone=UTC");  
 dsc.setDriverName("com.mysql.cj.jdbc.Driver");  
 dsc.setUsername("root");  
 dsc.setPassword("root");  
 mpg.setDataSource(dsc);  
 PackageConfig pc = new PackageConfig();  
 //设置文件的生成的包的位置：修改为自己的项目路径  
 pc.setParent("com.whs.dubbonacosmybatisplusprovider");  
 pc.setController("controller");  
 pc.setService("service");  
 pc.setServiceImpl("serviceimpl");  
 pc.setMapper("dao");  
 pc.setEntity("entity");  
 mpg.setPackageInfo(pc);  
 // 自定义配置  
 InjectionConfig cfg = new InjectionConfig() {  
 @Override  
 public void initMap() {  
 // to do nothing  
 }  
 };  
 List<FileOutConfig> focList = new ArrayList<>();  
 //下面的代码需要用到freemarker的相关依赖  
 focList.add(new FileOutConfig("/templates/mapper.xml.ftl") {  
 @Override  
 public String outputFile(TableInfo tableInfo) {  
 return projectPath + "//dubbo-nacos-mybatisplus-provider/src/main/resources/mapper/" + "/" + tableInfo.getEntityName() + "Dao" + StringPool.DOT\_XML;//mapper.xml文件路径  
 }  
 });  
 cfg.setFileOutConfigList(focList);  
 mpg.setCfg(cfg);  
 mpg.setTemplate(new TemplateConfig().setXml(null));  
 // 策略配置  
 StrategyConfig strategy = new StrategyConfig();  
 strategy.setNaming(NamingStrategy.underline\_to\_camel);  
 strategy.setColumnNaming(NamingStrategy.underline\_to\_camel);  
 //strategy.setEntityLombokModel(true);//是否系列化类  
 strategy.setRestControllerStyle(true);  
 strategy.setInclude(new String[] { "person" });//设置要生成的代码的表名  
 strategy.setSuperEntityColumns("id");  
 strategy.setControllerMappingHyphenStyle(true);  
 strategy.setTablePrefix(pc.getModuleName() + "\_");  
 mpg.setStrategy(strategy);  
 mpg.setTemplateEngine(new FreemarkerTemplateEngine());  
 mpg.execute();  
 }  
}