**品优购电商系统开发**

**第1章**

**分布式服务框架-Dubbo**

传智播客.黑马程序员

# 课程目标

目标1：了解电商基本知识

目标2：了解品优购的需求与设计

目标3：掌握Dubbo框架的使用

目标4：掌握Dubbo框架与SSM框架的搭建

目标5：完成品优购品牌列表后端代码的开发

# 1.走进电商【电商模式】

## 1.1电商行业分析

近年来，中国的电子商务快速发展，交易额连创新高，电子商务在各领域的应用不断拓展和深化、相关服务业蓬勃发展、支撑体系不断健全完善、创新的动力和能力 不断增强。电子商务正在与实体经济深度融合，进入规模性发展阶段，对经济社会生活的影响不断增大，正成为我国经济发展的新引擎。

中国电子商务研究中心数据显示，截止到 2015 年底，中国电子商务市场交易规模达 18.2万亿人民币，同比增长 34.81%。



2016 双 11 开场 30 分钟，创造每秒交易峰值 17.5 万笔，每秒支付峰值 12 万笔的新纪录。菜鸟单日物流订单量超过 4.67亿，创历史新高。



## 1.2电商行业技术特点

* 技术新
* 技术范围广
* 分布式
* 高并发、集群、负载均衡、高可用
* 海量数据
* 业务复杂
* 系统安全

## 1.3主要电商模式

### 1.3.1 **B2B--企业对企业**

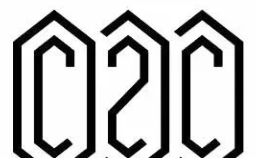
B2B （ Business to Business）是指进行[电子商务](http://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E5%95%86%E5%8A%A1" \t "http://baike.baidu.com/_blank)交易的供需双方都是商家（或企业、公司），她（他）们使用了[互联网](http://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91/199186" \t "http://baike.baidu.com/_blank)的技术或各种商务网络平台，完成商务交易的过程。电子商务是现代 B2B marketing的一种具体主要的表现形式。



案例：阿里巴巴、慧聪网

### 1.3.2 **C2C--个人对个人**

C2C即 Customer（Consumer） to Customer（Consumer），意思就是消费者个人间的电子商务行为。比如一个消费者有一台电脑，通过网络进行[交易](http://baike.baidu.com/item/%E4%BA%A4%E6%98%93/32757" \t "http://baike.baidu.com/_blank)，把它出售给另外一个消费者，此种交易类型就称为C2C电子商务。



案例：淘宝、易趣、瓜子二手车

### 1.3.3 **B2C--企业对个人**

B2C是[Business-to-Customer](http://baike.baidu.com/item/Business-to-Customer" \t "http://baike.baidu.com/_blank)的缩写，而其中文简称为“商对客”。“商对客”是[电子商务](http://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E5%95%86%E5%8A%A1" \t "http://baike.baidu.com/_blank)的一种模式，也就是通常说的直接面向[消费者](http://baike.baidu.com/item/%E6%B6%88%E8%B4%B9%E8%80%85" \t "http://baike.baidu.com/_blank)销售产品和服务商业[零售](http://baike.baidu.com/item/%E9%9B%B6%E5%94%AE" \t "http://baike.baidu.com/_blank)模式。这种形式的电子商务一般以网络零售业为主，主要借助于互联网开展在线销售活动。B2C即[企业](http://baike.baidu.com/item/%E4%BC%81%E4%B8%9A" \t "http://baike.baidu.com/_blank)通过互联网为消费者提供一个新型的购物环境——[网上商店](http://baike.baidu.com/item/%E7%BD%91%E4%B8%8A%E5%95%86%E5%BA%97" \t "http://baike.baidu.com/_blank)，消费者通过网络在[网上购物](http://baike.baidu.com/item/%E7%BD%91%E4%B8%8A%E8%B4%AD%E7%89%A9" \t "http://baike.baidu.com/_blank)、[网上支付](http://baike.baidu.com/item/%E7%BD%91%E4%B8%8A%E6%94%AF%E4%BB%98" \t "http://baike.baidu.com/_blank)等消费行为。



案例：唯品会、乐蜂网

### 1.3.4 O2O**--线上到线下**

O2O即Online To Offline（在线离线/[线上到线下](http://baike.baidu.com/item/%E7%BA%BF%E4%B8%8A%E5%88%B0%E7%BA%BF%E4%B8%8B" \t "http://baike.baidu.com/_blank)），是指将线下的商务机会与互联网结合，让互联网成为线下交易的平台，这个概念最早来源于[美国](http://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD/125486" \t "http://baike.baidu.com/_blank)。O2O的概念非常广泛，既可涉及到线上，又可涉及到线下,可以通称为O2O。主流商业管理课程均对O2O这种新型的商业模式有所介绍及关注。



案例：美团、饿了吗

### 1.3.5 B2B2C -企业-企业-个人

[B2B2C](https://www.baidu.com/s?wd=B2B2C&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWuynvmWw-n10znjIhmWPB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10zPW6dnHf4" \t "https://zhidao.baidu.com/question/_blank)是一种[电子商务类型](https://www.baidu.com/s?wd=%E7%94%B5%E5%AD%90%E5%95%86%E5%8A%A1%E7%B1%BB%E5%9E%8B&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWuynvmWw-n10znjIhmWPB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10zPW6dnHf4" \t "https://zhidao.baidu.com/question/_blank)的网络购物商业模式，B是BUSINESS的简称，C是CUSTOMER的简称，第一个B指的是商品或服务的供应商，第二个B指的是从事电子商务的企业，C则是表示消费者。  
　　第一个BUSINESS，并不仅仅局限于品牌供应商、影视制作公司和图书出版商，任何的商品供应商或服务供应商都能可以成为第一个BUSINESS；第二B是[B2B2C](https://www.baidu.com/s?wd=B2B2C&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWuynvmWw-n10znjIhmWPB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10zPW6dnHf4" \t "https://zhidao.baidu.com/question/_blank)模式的电子商务企业，通过统一的经营管理对商品和服务、消费者终端同时进行整合，是广大供应商和消费者之间的桥梁，为供应商和消费者提供优质的服务，是[互联网电子商务](https://www.baidu.com/s?wd=%E4%BA%92%E8%81%94%E7%BD%91%E7%94%B5%E5%AD%90%E5%95%86%E5%8A%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWuynvmWw-n10znjIhmWPB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10zPW6dnHf4" \t "https://zhidao.baidu.com/question/_blank)服务供应商。C表示消费者，在第二个B构建的统一电子商务平台购物的消费者；  
　　[B2B2C](https://www.baidu.com/s?wd=B2B2C&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWuynvmWw-n10znjIhmWPB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10zPW6dnHf4" \t "https://zhidao.baidu.com/question/_blank)的来源于目前的B2B、B2C模式的演变和完善，把B2C和C2C完美地结合起来，通过B2B2C模式的电子商务企业构建自己的物流供应链系统，提供统一的服务。

案例：京东商城、天猫商城

# 2.品优购【项目简介&系统架构】

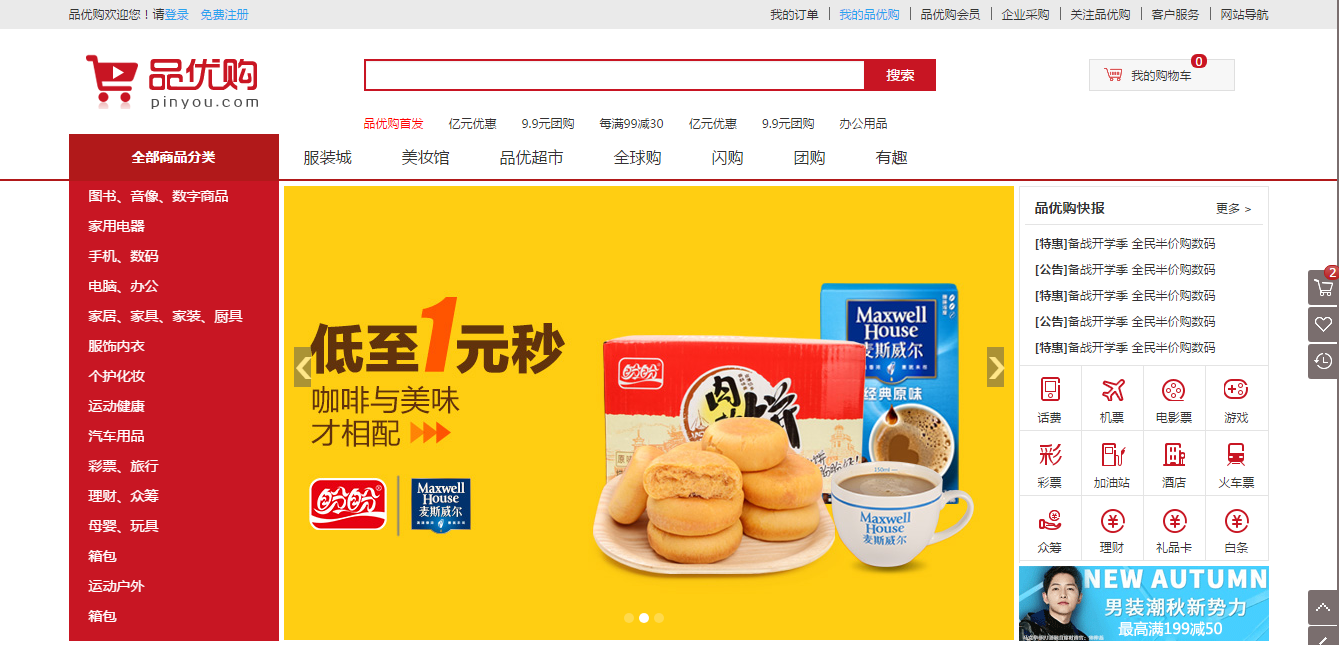
## 2.1品优购简介

品优购网上商城是一个综合性的 B2B2C 平台，类似京东商城、天猫商城。网站采用商家入驻的模式，商家入驻平台提交申请，有平台进行资质审核，审核通过后，商家拥有独立的管理后台录入商品信息。商品经过平台审核后即可发布。

品优购网上商城主要分为网站前台、运营商后台、商家管理后台三类子系统

### 2.1.1 网站前台

主要包括网站首页、商家首页、商品详细页、购物车页、搜索页、会员中心、订单与支付相关页面、秒杀频道等。



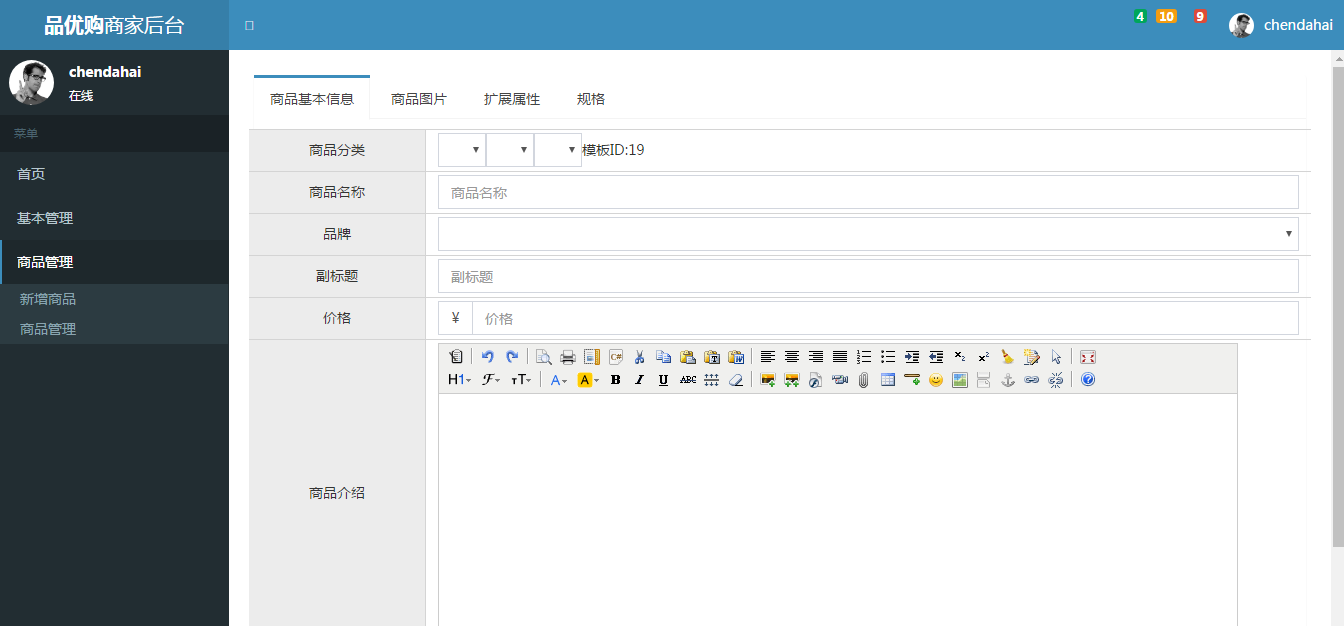
### 2.1.2 运营商后台

是运营商的运营人员的管理后台。主要包括商家审核、品牌管理、规格管理、模板管理、商品分类管理、商品审核、广告类型管理、广告管理、订单查询、商家结算等。



### 2.1.3 商家管理后台

入驻的商家进行管理的后台，主要功能是对商品的管理以及订单查询统计、资金结算等功能。



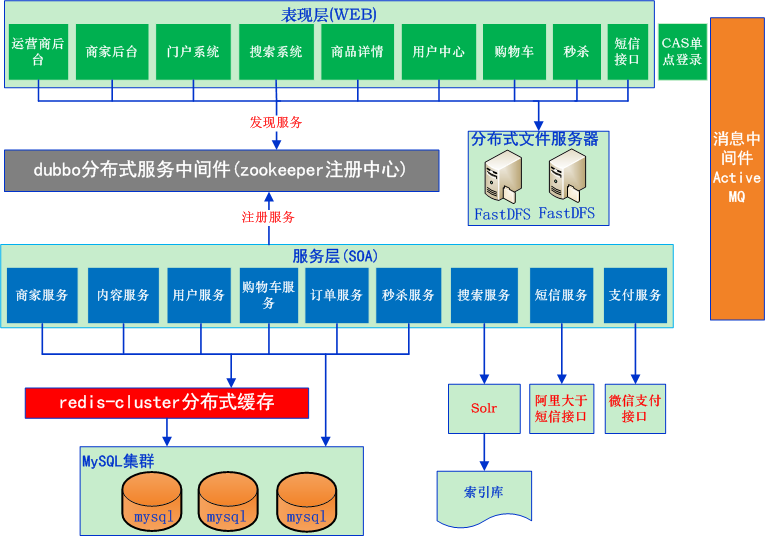
## 2.2系统架构特点

### 2.2.1 什么是SOA架构

SOA是Service-Oriented Architecture的首字母简称，它是一种支持面向服务的架构样式。从服务、基于服务开发和服务的结果来看，面向服务是一种思考方式。其实SOA架构更多应用于互联网项目开发。

为什么互联网项目会采用SOA架构呢？随着互联网的发展，网站应用的规模不断扩大，常规的垂直应用架构已无法应对，分布式服务架构以及流动计算架构势在必行，迫切需一个治理系统确保架构有条不紊的演进。

### 2.2.2 品优购架构图



### 2.2.3 框架组合

品优购采用当前流行的**前后端分离**编程架构。

前端采用：**AngularJS + Bootstrap**

后端框架采用：**Spring + SpringMVC + MyBatis + Dubbo**

## 2.3数据库表结构

|  |  |
| --- | --- |
| **表名称** | **含义** |
| **tb\_brand** | **品牌** |
| **tb\_specification** | **规格** |
| **tb\_specification\_option** | **规格选项** |
| **tb\_type\_template** | **类型模板：用于关联品牌和规格** |
| **tb\_item\_cat** | **商品分类** |
| **tb\_seller** | **商家** |
| **tb\_goods** | **商品** |
| **tb\_goods\_desc** | **商品详情** |
| **tb\_item** | **商品明细** |
| **tb\_content** | **内容（广告）** |
| **tb\_content\_category** | **内容（广告）类型** |
| **tb\_user** | **用户** |
| **tb\_order** | **订单** |
| **tb\_order\_item** | **订单明细** |
| **tb\_pay\_log** | **支付日志** |
| **tb\_seckill\_goods** | **秒杀商品** |
| **tb\_seckill\_order** | **秒杀订单** |

# Dubbo【分布式服务框架】

## 3.1架构分析

### 3.1.1单一应用架构

使用单一应用架构：有耦合性高，无法水平扩展（集群）的问题



### 3.1.2垂直应用架构

按系统业务功能进行水平（后台、门户、搜索...）拆分。



垂直应用架构能解决单一应用架构存在的问题，但是如果上述拆分，会重复编写同样功能方法的问题。如上图，两个系统都需要实现根据商品id查询商品的功能。

### 3.1.3分布式服务架构

把页面控制层Controller单独拆出来作为表现层；业务层和持久层拆出来作为服务层。这样不会存在编写通用功能的方法。



如果业务复杂，服务多了以后则将出现如下图示：



存在的问题：

a、当服务越来越多，调用服务的URL就越来越难以管理。

b、当调用服务量越来越大，很难确定服务需要多少机器支撑，什么时候增加机器。而dubbo分布式服务框架，可用来解决这些问题。

## 3.2 Dubbo简介

Dubbo是一个分布式服务框架，是阿里巴巴开源项目Dubbo ，被国内电商及互联网项目中使用。

Dubbo 致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。简单的说，dubbo就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有dubbo这样的分布式服务框架的需求，并且本质上是个服务调用的东东，说白了就是个远程服务调用的分布式框架。

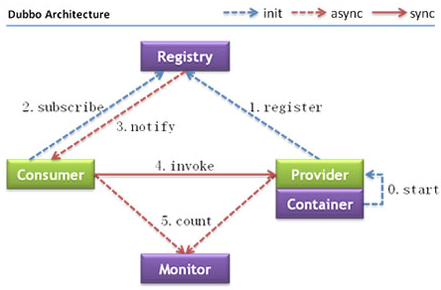
Dubbo |ˈdʌbəʊ| 是一款高性能、轻量级的开源Java RPC框架，它提供了三大核心能力：

* 面向接口的远程方法调用。
* 智能容错和负载均衡。
* 以及服务自动注册和发现。

官网地址：

<http://dubbo.io/> (原来)

<http://dubbo.apache.org(现在>)



**节点角色说明：**

 Provider: 暴露服务的服务提供方。

 Consumer: 调用远程服务的服务消费方。

 Registry: 服务注册与发现的注册中心。

 Monitor: 统计服务的调用次数和调用时间的监控中心。

 Container: 服务运行容器。

**调用关系说明：**

 0. 服务容器负责启动，加载，运行服务提供者。

 1. 服务提供者在启动时，向注册中心注册自己提供的服务。

 2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

 3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。

 4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，

如果调用失败，再选另一台调用。

 5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

## 3.3注册中心Zookeeper

### 3.3.1 Zookeeper介绍

官方推荐使用 zookeeper注册中心。注册中心负责服务地址的注册与查找，相当于目录服务，服务提供者和消费者只在启动时与注册中心交互，注册中心不转发请求，压力较小。分布式协调服务服务器。

Zookeeper 是 Apacahe Hadoop 的子项目，是一个树型的目录服务，支持变更推送，适合作为Dubbo服务的注册中心，工业强度较高，可用于生产环境。

### 3.3.2 Linux系统安装Zookeeper

**安装参考：【资料\zookeeper\Linux安装zookeeper.docx】**

## 3.4 Dubbo监控中心

我们在开发时，需要知道注册中心都注册了哪些服务，以便我们开发和测试。我们可以通过部署一个监控中心来实现。其实监控中心就是一个web应用，部署到tomcat即可。

说明：**监控中心它监听的是注册中心**。

### 3.4.1监控中心安装

**部署监控中心参考：【资料\dubbo\安装部署dubbo的监控中心monitor.docx】**

### 3.4.2登录监控中心

（1）打开浏览器，输入http://192.168.12.131:8080 ,登录用户名和密码均为root 进入首页。 (192.168.12.131)是我部署的linux主机地址。



# 4.品优购【工程构建】

## 4.1工程结构分析

开发工具：**IntelliJ IDEA**

版本控制：**Git**

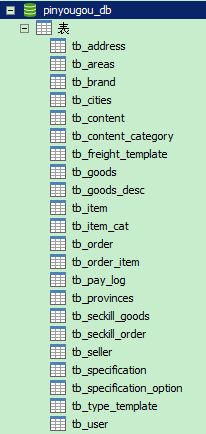
Maven构建品优购分布式项目结构图：

Maven构建品优购分布式项目依赖图：



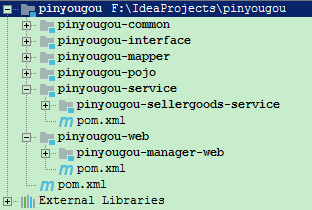
## 4.2创建数据库表

执行资源文件夹中：【资料\建库语句\pinyougou\_db.sql】

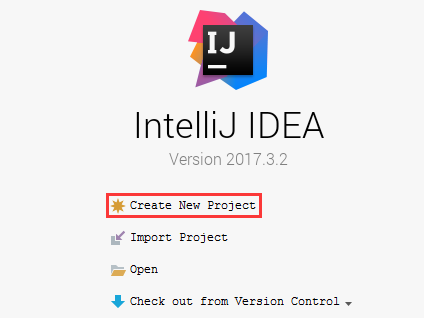


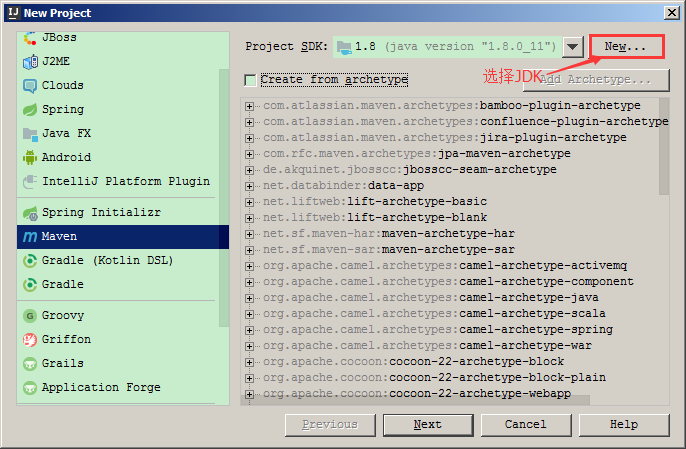
## 4.3工程构建

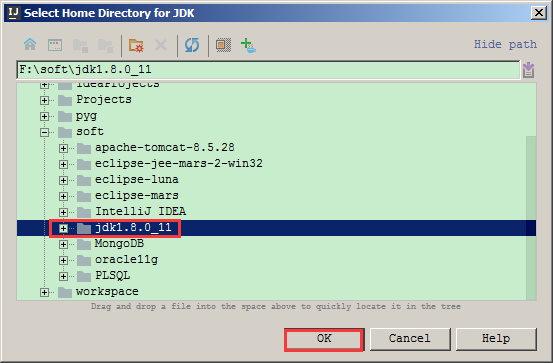
构建效果如下：

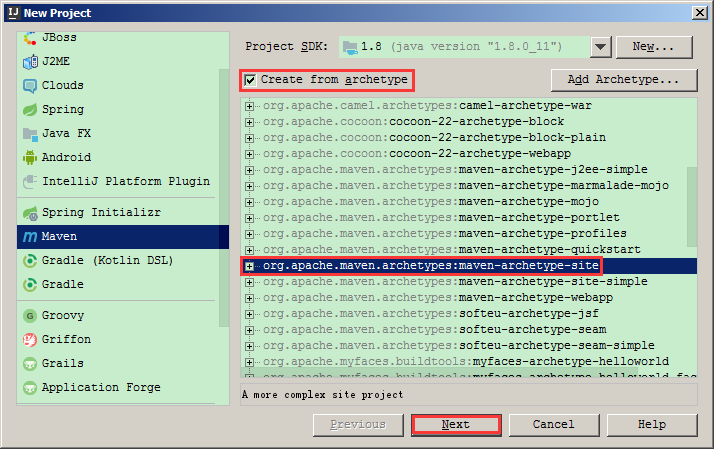


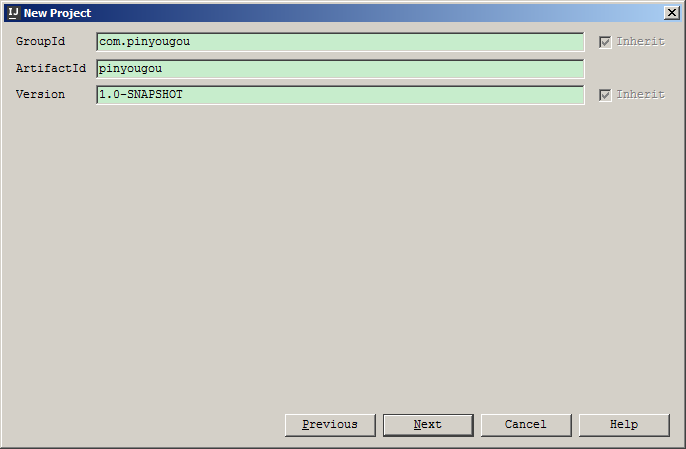
### 4.3.1 pinyougou聚合父项目(pom)

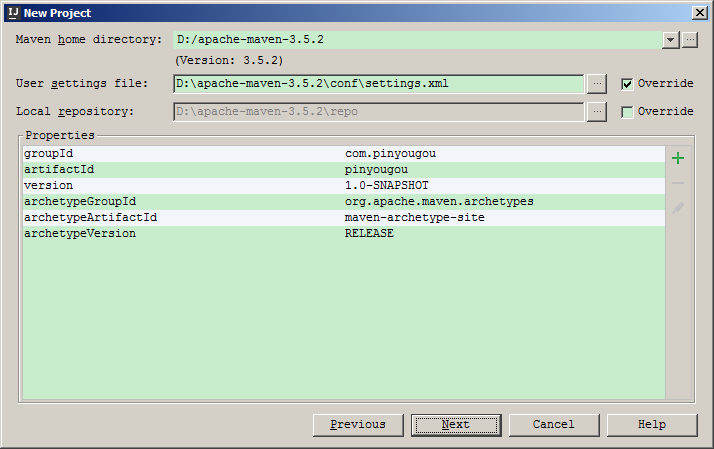


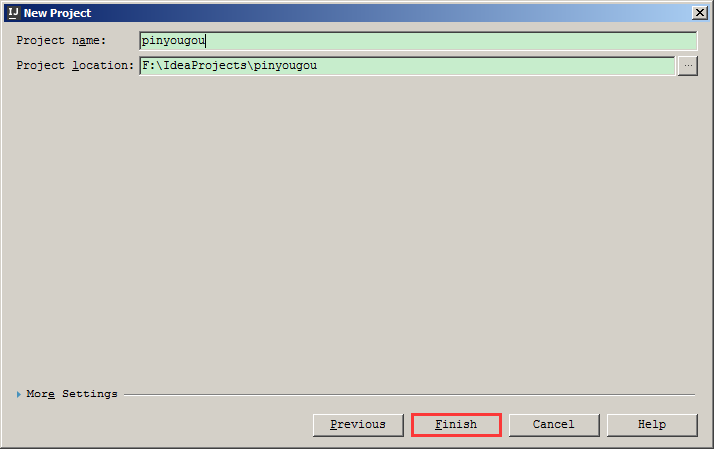




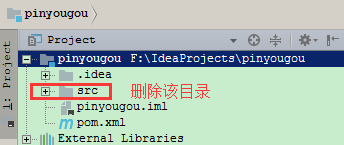




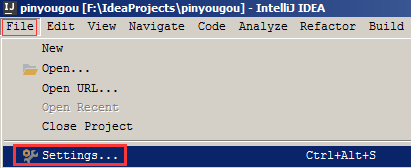


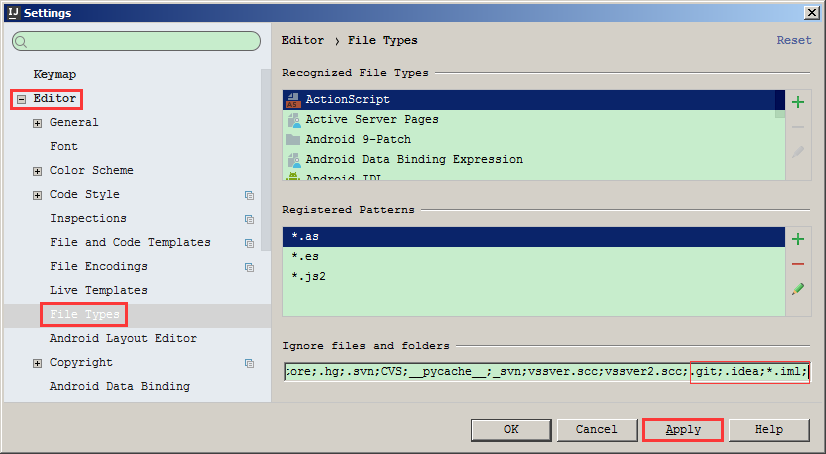


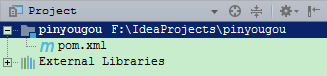




配置隐藏.idea、pinyougou.iml、.git文件





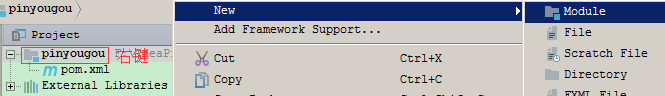


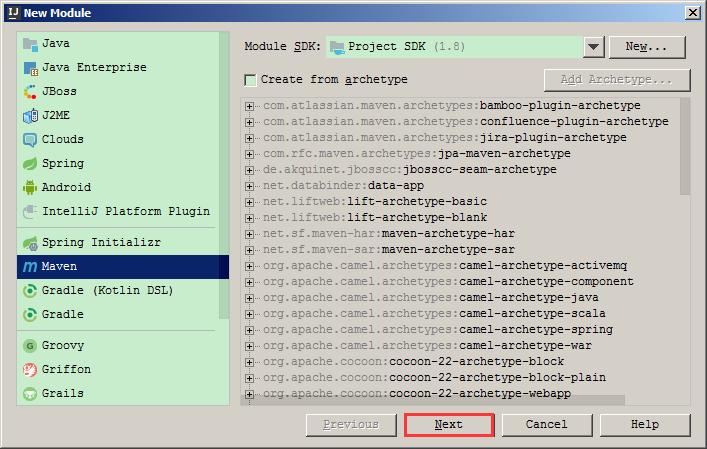
**修改pinyougou/pom.xml文件：【资料\pinyougou-pom.xml】**

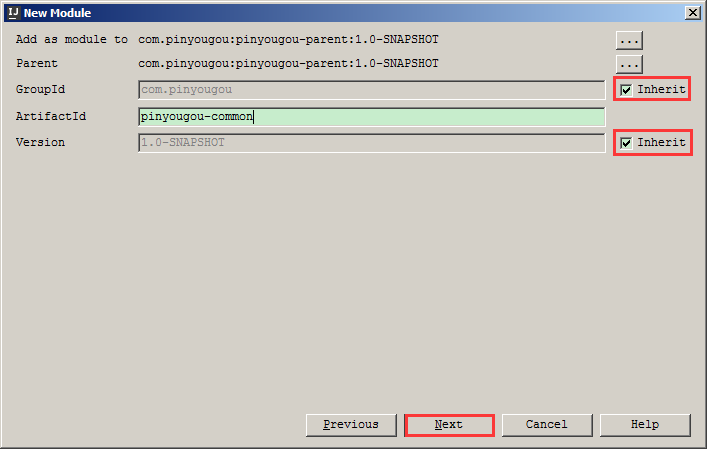
|  |
| --- |
| <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/xsd/maven-4.0.0.xsd"**>   <**modelVersion**>4.0.0</**modelVersion**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-parent</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  <**packaging**>pom</**packaging**>  <**name**>${project.artifactId}</**name**>   *<!-- 定义全局的属性(定义依赖jar包的版本号) -->* <**properties**>  <**java.version**>1.8</**java.version**>  <**junit.version**>4.12</**junit.version**>  <**tomcat.version**>8.5.28</**tomcat.version**>  <**spring.version**>4.3.10.RELEASE</**spring.version**>  <**spring.security.version**>4.2.3.RELEASE</**spring.security.version**>  <**mybatis.version**>3.4.1</**mybatis.version**>  <**mybatis.spring.version**>1.3.1</**mybatis.spring.version**>  <**pagehelper.version**>5.0.0</**pagehelper.version**>  <**mapper.version**>3.4.0</**mapper.version**>  <**mybatis.paginator.version**>1.2.15</**mybatis.paginator.version**>  <**mysql.version**>5.1.38</**mysql.version**>  <**druid.version**>1.0.9</**druid.version**>  <**dubbo.version**>2.5.7</**dubbo.version**>  <**zookeeper.version**>3.4.11</**zookeeper.version**>  <**zkclient.version**>0.1</**zkclient.version**>  <**commons-lang3.version**>3.3.2</**commons-lang3.version**>  <**commons-io.version**>1.3.2</**commons-io.version**>  <**commons-net.version**>3.3</**commons-net.version**>  <**commons-logging.version**>1.2</**commons-logging.version**>  <**commons-fileupload.version**>1.3.1</**commons-fileupload.version**>  <**commons-codec.version**>1.9</**commons-codec.version**>  <**jedis.version**>2.9.0</**jedis.version**>  <**activemq.version**>5.14.5</**activemq.version**>  <**freemarker.version**>2.3.26-incubating</**freemarker.version**>  <**joda-time.version**>2.5</**joda-time.version**>  <**spring.data.redis.version**>1.8.6.RELEASE</**spring.data.redis.version**>  <**cas.client.version**>3.4.1</**cas.client.version**>  <**solrj.version**>4.10.3</**solrj.version**>  <**ik.version**>2012\_u6</**ik.version**>  <**slf4j.version**>1.7.5</**slf4j.version**>  <**fastjson.version**>1.2.29</**fastjson.version**>  <**fastdfs.version**>1.25-RELEASE</**fastdfs.version**>  <**httpclient.version**>4.5.3</**httpclient.version**>  <**kaptcha.version**>2.3.2</**kaptcha.version**>  </**properties**>   *<!-- 依赖管理配置(用于锁定指定版本) -->* <**dependencyManagement**>  <**dependencies**>  *<!-- junit -->* <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  <**version**>${junit.version}</**version**>  <**scope**>test</**scope**>  </**dependency**>  *<!-- servlet-api -->* <**dependency**>  <**groupId**>org.apache.tomcat.embed</**groupId**>  <**artifactId**>tomcat-embed-core</**artifactId**>  <**version**>${tomcat.version}</**version**>  <**scope**>provided</**scope**>  </**dependency**>  *<!-- spring4 -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jdbc</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-web</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context-support</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-test</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  *<!-- mybatis -->* <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis</**artifactId**>  <**version**>${mybatis.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis-spring</**artifactId**>  <**version**>${mybatis.spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**>  <**version**>${pagehelper.version}</**version**>  </**dependency**>  *<!-- 通用Mapper -->* <**dependency**>  <**groupId**>tk.mybatis</**groupId**>  <**artifactId**>mapper</**artifactId**>  <**version**>${mapper.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.miemiedev</**groupId**>  <**artifactId**>mybatis-paginator</**artifactId**>  <**version**>${mybatis.paginator.version}</**version**>  </**dependency**>  *<!-- mysql -->* <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  <**version**>${mysql.version}</**version**>  </**dependency**>  *<!-- druid连接池 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>druid</**artifactId**>  <**version**>${druid.version}</**version**>  </**dependency**>  *<!-- dubbo相关 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>${dubbo.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.zookeeper</**groupId**>  <**artifactId**>zookeeper</**artifactId**>  <**version**>${zookeeper.version}</**version**>  <**exclusions**>  <**exclusion**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-api</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.sgroschupf</**groupId**>  <**artifactId**>zkclient</**artifactId**>  <**version**>${zkclient.version}</**version**>  <**exclusions**>  <**exclusion**>  <**groupId**>org.apache.zookeeper</**groupId**>  <**artifactId**>zookeeper</**artifactId**>  </**exclusion**>  <**exclusion**>  <**groupId**>log4j</**groupId**>  <**artifactId**>log4j</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  *<!-- 日志 -->* <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-log4j12</**artifactId**>  <**version**>${slf4j.version}</**version**>  </**dependency**>  *<!-- apache工具组件 -->* <**dependency**>  <**groupId**>org.apache.commons</**groupId**>  <**artifactId**>commons-lang3</**artifactId**>  <**version**>${commons-lang3.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.commons</**groupId**>  <**artifactId**>commons-io</**artifactId**>  <**version**>${commons-io.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>commons-codec</**groupId**>  <**artifactId**>commons-codec</**artifactId**>  <**version**>${commons-codec.version}</**version**>  </**dependency**>  *<!-- 文件上传组件 -->* <**dependency**>  <**groupId**>commons-fileupload</**groupId**>  <**artifactId**>commons-fileupload</**artifactId**>  <**version**>${commons-fileupload.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.csource</**groupId**>  <**artifactId**>fastdfs-client</**artifactId**>  <**version**>${fastdfs.version}</**version**>  </**dependency**>  *<!-- JSON处理工具包 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  <**version**>${fastjson.version}</**version**>  </**dependency**>  *<!-- redis缓存 -->* <**dependency**>  <**groupId**>redis.clients</**groupId**>  <**artifactId**>jedis</**artifactId**>  <**version**>${jedis.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.data</**groupId**>  <**artifactId**>spring-data-redis</**artifactId**>  <**version**>${spring.data.redis.version}</**version**>  </**dependency**>  *<!-- activeMQ依赖 -->* <**dependency**>  <**groupId**>org.apache.activemq</**groupId**>  <**artifactId**>activemq-client</**artifactId**>  <**version**>${activemq.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jms</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  *<!-- freemarker -->* <**dependency**>  <**groupId**>org.freemarker</**groupId**>  <**artifactId**>freemarker</**artifactId**>  <**version**>${freemarker.version}</**version**>  </**dependency**>  *<!-- spring-security -->* <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-web</**artifactId**>  <**version**>${spring.security.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-config</**artifactId**>  <**version**>${spring.security.version}</**version**>  </**dependency**>  *<!-- 验证码 -->* <**dependency**>  <**groupId**>com.github.penggle</**groupId**>  <**artifactId**>kaptcha</**artifactId**>  <**version**>${kaptcha.version}</**version**>  <**exclusions**>  <**exclusion**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>javax.servlet-api</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-cas</**artifactId**>  <**version**>${spring.security.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.jasig.cas.client</**groupId**>  <**artifactId**>cas-client-core</**artifactId**>  <**version**>${cas.client.version}</**version**>  *<!-- 排除slf4j-api包冲突 -->* <**exclusions**>  <**exclusion**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-api</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  *<!-- solrj -->* <**dependency**>  <**groupId**>org.apache.solr</**groupId**>  <**artifactId**>solr-solrj</**artifactId**>  <**version**>${solrj.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.janeluo</**groupId**>  <**artifactId**>ikanalyzer</**artifactId**>  <**version**>${ik.version}</**version**>  </**dependency**>  *<!-- httpclient -->* <**dependency**>  <**groupId**>org.apache.httpcomponents</**groupId**>  <**artifactId**>httpclient</**artifactId**>  <**version**>${httpclient.version}</**version**>  </**dependency**>  </**dependencies**>  </**dependencyManagement**>  *<!-- 项目构建部分 -->* <**build**>  *<!-- 插件管理(用于锁定插件版本) -->* <**pluginManagement**>  <**plugins**>  *<!-- 配置编译插件 -->* <**plugin**>  <**groupId**>org.apache.maven.plugins</**groupId**>  <**artifactId**>maven-compiler-plugin</**artifactId**>  <**version**>3.3</**version**>  <**configuration**>  <**source**>${java.version}</**source**>  <**target**>${java.version}</**target**>  <**encoding**>UTF-8</**encoding**>  </**configuration**>  </**plugin**>  *<!-- 配置tomcat插件 -->* <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.2</**version**>  </**plugin**>  </**plugins**>  </**pluginManagement**>  <**plugins**>  *<!-- 配置编译插件 -->* <**plugin**>  <**groupId**>org.apache.maven.plugins</**groupId**>  <**artifactId**>maven-compiler-plugin</**artifactId**>  <**version**>3.3</**version**>  </**plugin**>  </**plugins**>  </**build**> </**project**> |

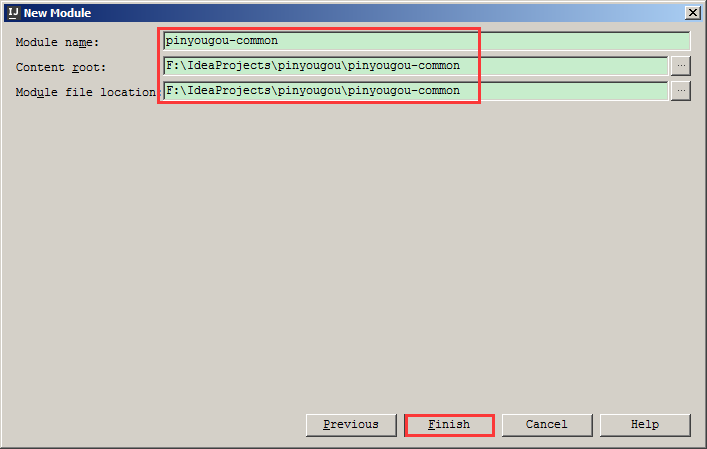
**说明：格式化代码快捷键【Ctrl + Alt + L】**

### 4.3.2 pinyougou-common**工具类(jar)**

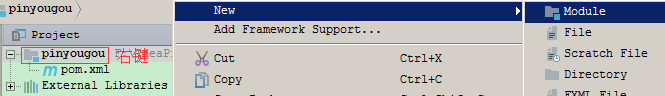


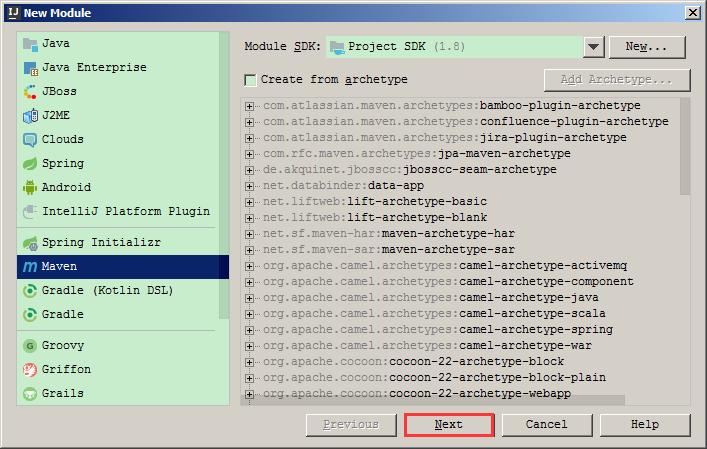


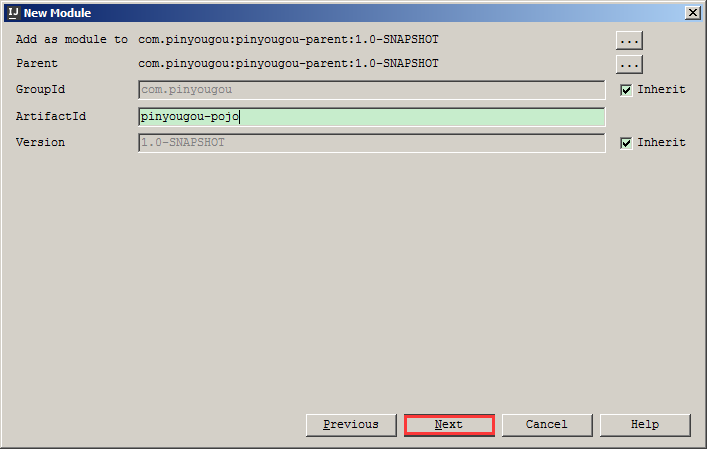


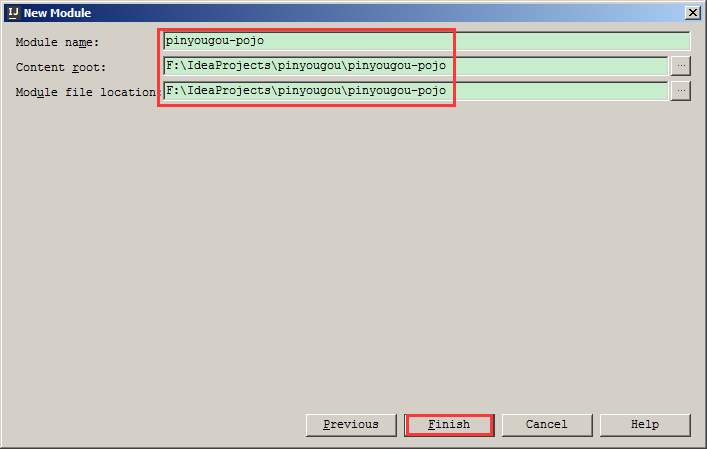


### 4.3.3 pinyougou-pojo**实体类(jar)**

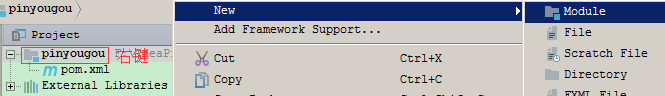


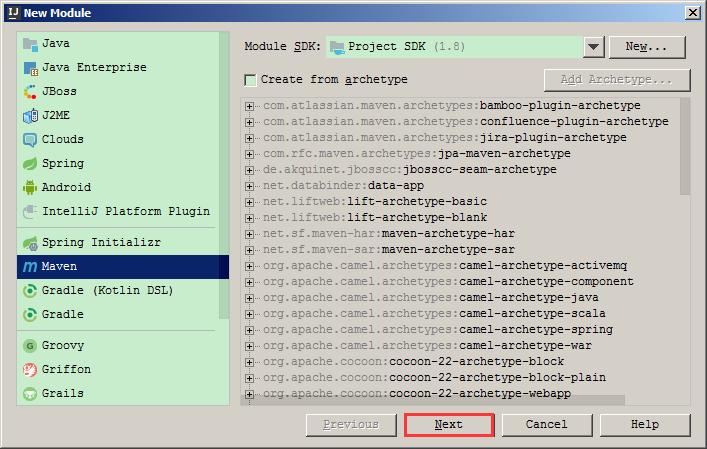


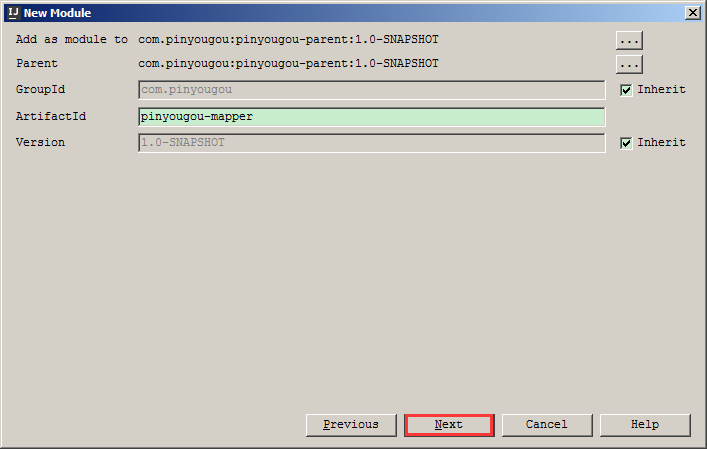


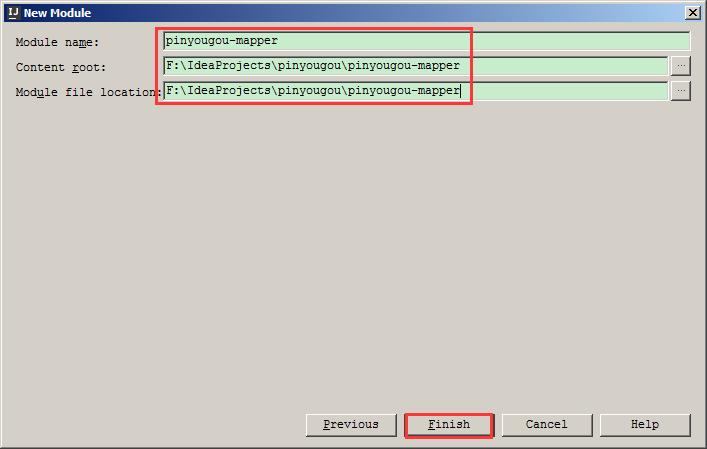


### 4.3.4 pinyougou-mapper**数据访问(jar)**









pinyougou-mapper/pom.xml文件，添加依赖：

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>pinyougou-parent</**artifactId**>  <**groupId**>com.pinyougou</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>  <**artifactId**>pinyougou-mapper</**artifactId**>  *<!-- 配置依赖关系 -->* <**dependencies**>  *<!-- spring-jdbc -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jdbc</**artifactId**>  </**dependency**>  *<!-- mybatis -->* <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis</**artifactId**>  </**dependency**>  *<!-- mybatis-spring -->* <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis-spring</**artifactId**>  </**dependency**>  *<!-- mysql -->* <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  </**dependency**>  *<!-- druid连接池 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>druid</**artifactId**>  </**dependency**>  *<!-- pinyougou-pojo -->* <**dependency**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-pojo</**artifactId**>  <**version**>${project.version}</**version**>  </**dependency**>  </**dependencies**> </**project**> |

在pinyougou-mapper/src/main/resources/目录下提供Spring整合MyBatis配置文件applicationContext-mapper.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://www.springframework.org/schema/beans"  xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx.xsd"**>   *<!-- 配置数据源 -->* <**bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"  destroy-method="close"**>  <**property name="url" value="${jdbc.url}"** />  <**property name="username" value="${jdbc.username}"**/>  <**property name="password" value="${jdbc.password}"**/>  <**property name="driverClassName" value="${jdbc.driver}"**/>  <**property name="maxActive" value="${jdbc.maxActive}"**/>  <**property name="minIdle" value="${jdbc.minIdle}"**/>  </**bean**>   *<!-- 配置SqlSessionFactory -->* <**bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>  *<!-- 设置数据源 -->* <**property name="dataSource" ref="dataSource"**/>  *<!-- 设置类型别名 (采用包扫描) -->* <**property name="typeAliasesPackage" value="com.pinyougou.pojo"**/>  *<!-- 设置mybatis-config.xml -->* <**property name="configLocation" value="classpath:mybatis-config.xml"**/>  *<!-- 设置SQL映射文件 -->* <**property name="mapperLocations" value="classpath:mappers/\*\*/\*.xml"**/>  </**bean**>   *<!-- 配置数据访问接口的代理对象 (批量配置)  到基础包下扫描所有的数据访问接口，再创建它们的代理对象，然后交给Spring容器  bean的id: 默认为接口的类名，首字母小写  -->* <**bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>  <**property name="basePackage" value="com.pinyougou.mapper"**/>  </**bean**>   *<!-- 配置数据源事务管理器(DataSourceTransactionManager) -->* <**bean id="transactionManager"**  **class="org.springframework.jdbc.datasource.DataSourceTransactionManager"**>  <**property name="dataSource" ref="dataSource"**/>  </**bean**>   *<!-- 配置开启事务注解驱动，由于dubbo用@service注解暴露服务，业务层需要加事务会产生代理对象，需设置proxy-target-class为true采用cglib产生代理 -->* <**tx:annotation-driven proxy-target-class="true"**/> </**beans**> |

提供数据源属性配置文件

pinyougou-mapper/src/main/resources/props/db.properties

|  |
| --- |
| **jdbc.driver**=**com.mysql.jdbc.Driver jdbc.url**=**jdbc:mysql://localhost:3306/pinyougou\_db?characterEncoding=utf-8 jdbc.username**=**root jdbc.password**=**root jdbc.maxActive**=**10 jdbc.minIdle**=**5** |

pinyougou-mapper/src/main/resources/mybatis-config.xml(MyBatis全局配置文件)

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>*** <**configuration**>  *<!-- 全局的设置信息-->* <**settings**>  *<!-- 开启驼峰命名(方便自动映射) dept\_id deptId -->* <**setting name="mapUnderscoreToCamelCase" value="true"**/>  </**settings**> </**configuration**> |

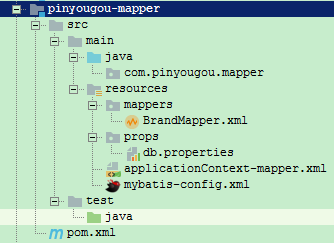
pinyougou-mapper/src/main/resources/mappers/BrandMapper.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"** *?>* **<!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>*** <**mapper namespace="test"**>  </**mapper**> |

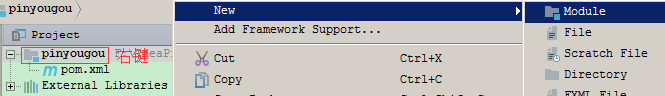
在pinyougou-pojo/src/main/java下面创建**com.pinyougou.pojo**包名

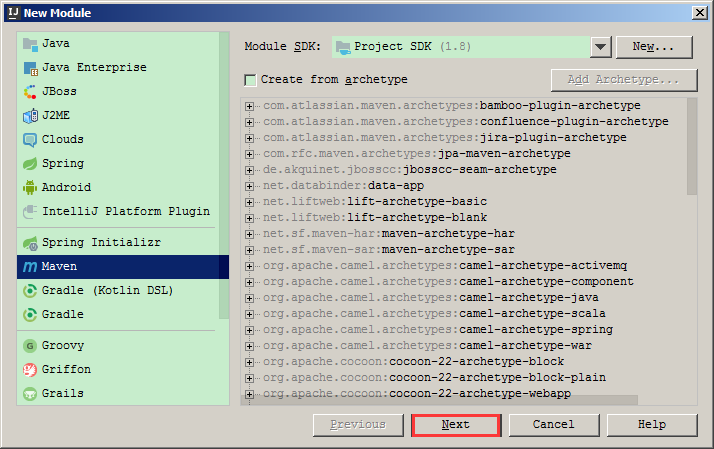
在pinyougou-mapper/src/main/java下面创建**com.pinyougou.mapper**包名

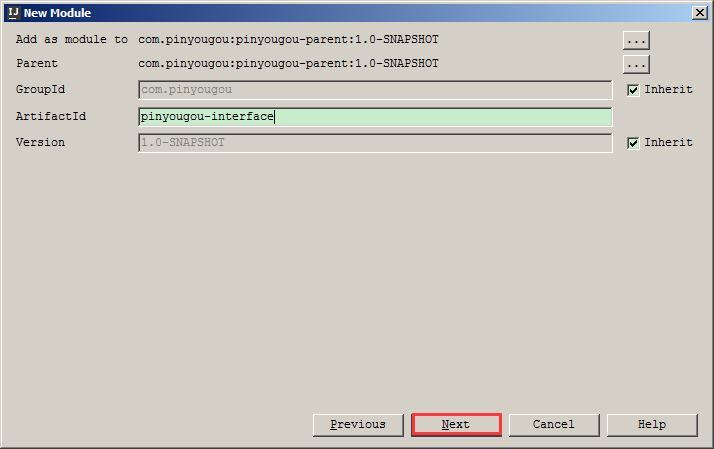
效果如下：

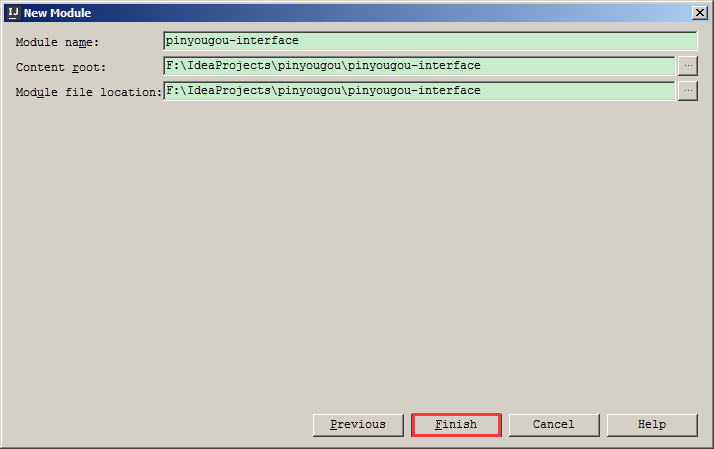


### 4.3.5 pinyougou-interface服务接口(jar)





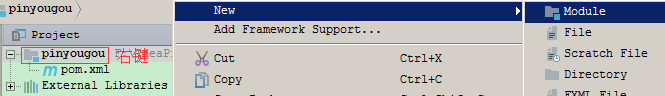


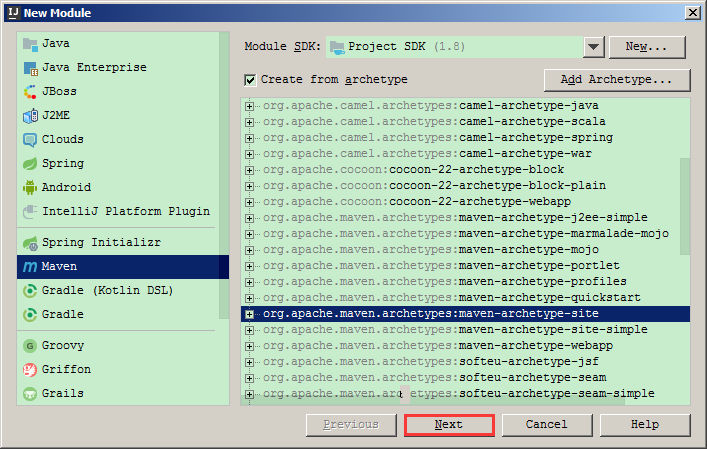


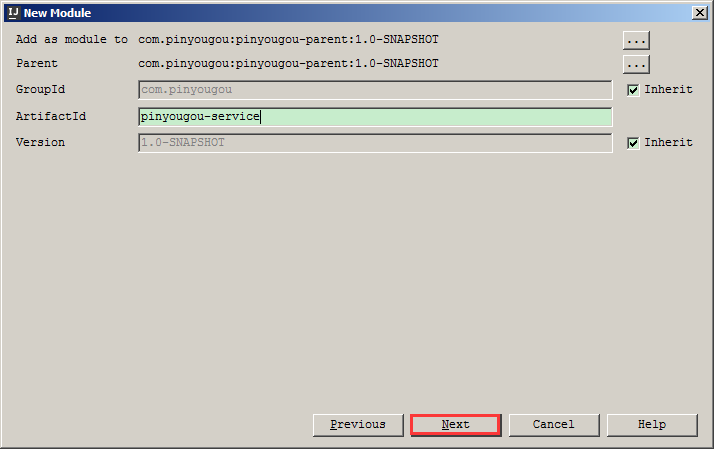
pinyougou-interface/pom.xml：

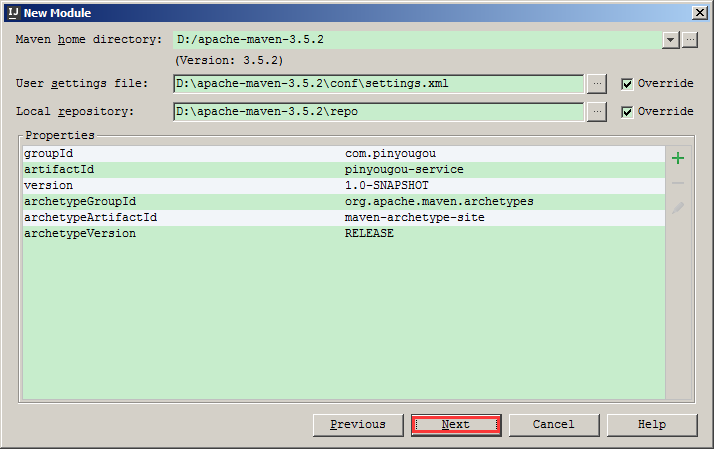
|  |
| --- |
| <**dependencies**>  *<!-- pinyougou-pojo -->* <**dependency**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-pojo</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> |

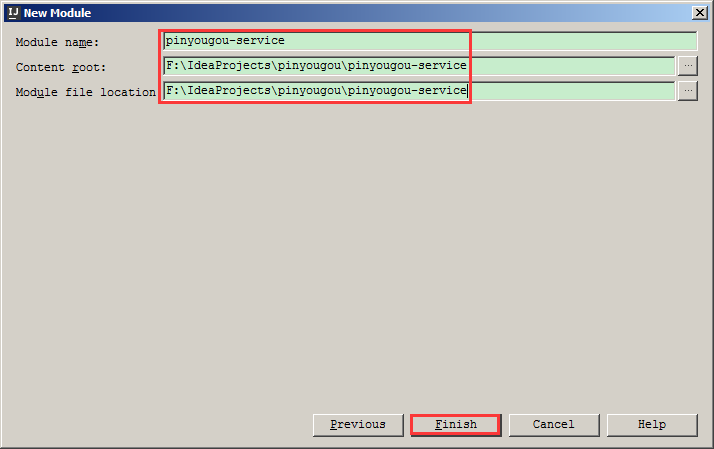
### **4.3.6 pinyougou-service服务提供者(pom)**

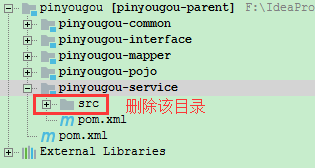








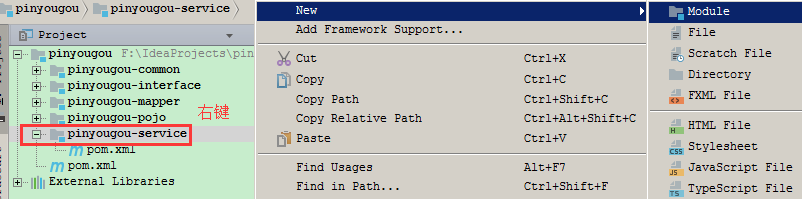


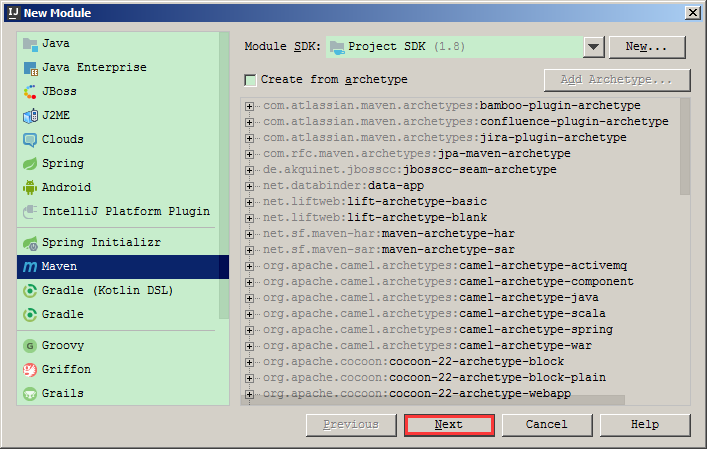


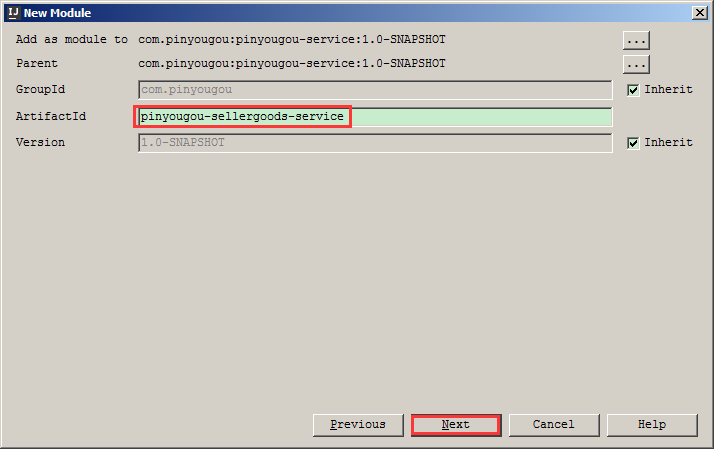
pinyougou-service/pom.xml，添加依赖：

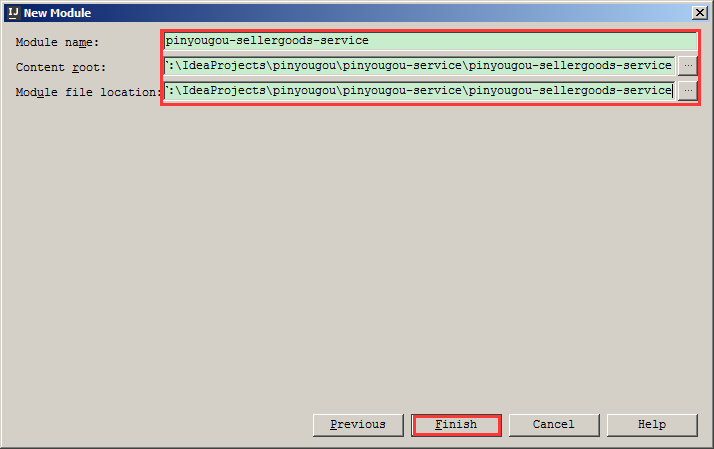
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>pinyougou-parent</**artifactId**>  <**groupId**>com.pinyougou</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>  <**artifactId**>pinyougou-service</**artifactId**>  <**packaging**>pom</**packaging**>  <**name**>pinyougou-service</**name**>  <**properties**>  *<!-- tomcat端口号 -->* <**port**>**9000**</**port**>  </**properties**>   <**dependencies**>  *<!-- 日志 -->* <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-log4j12</**artifactId**>  </**dependency**>  *<!-- JSON处理工具包 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  </**dependency**>  *<!-- spring -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-web</**artifactId**>  </**dependency**>  *<!-- pinyougou-interface -->* <**dependency**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  *<!-- pinyougou-mapper -->* <**dependency**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-mapper</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  </**dependencies**>  <**build**>  <**plugins**>  *<!-- 配置tomcat插件 -->* <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**configuration**>  <**path**>/</**path**>   **<port>${port}</port>**  </**configuration**>  </**plugin**>  </**plugins**>  </**build**> </**project**> |

#### 4.3.6.1 pinyougou-sellergoods-service商家服务(war)

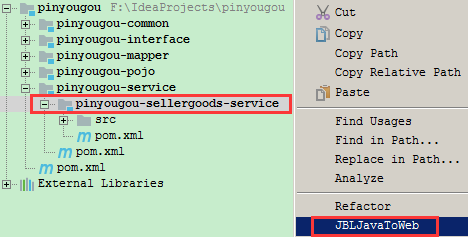


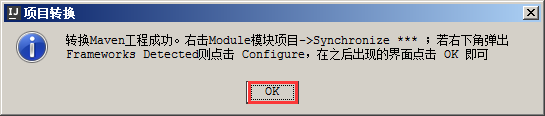






把java项目转化成Web项目





pinyougou-sellergoods-service/pom.xml添加依赖：

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>pinyougou-service</**artifactId**>  <**groupId**>com.pinyougou</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>  <**artifactId**>pinyougou-sellergoods-service</**artifactId**>  <**packaging**>war</**packaging**>   <**properties**>  *<!-- 覆盖父工程中的port属性 -->* <**port**>9001</**port**>  </**properties**> </**project**> |

修改pinyougou-sellergoods-service/src/main/webapp/WEB-INF/web.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" version="3.0"**>  *<!-- 配置Spring加载文件 -->* <**context-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:applicationContext\*.xml</**param-value**>  </**context-param**>  *<!-- 配置Spring的核心监听器 -->* <**listener**> <**listener-class**>org.springframework.web.context.ContextLoaderListener</**listener-class**>  </**listener**> </**web-app**> |

pinyougou-sellergoods-service/src/main/resources/log4j.properties

|  |
| --- |
| **log4j.rootLogger**=**DEBUG,stdout log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern**=**%-d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c]-[%p] %m%n** |

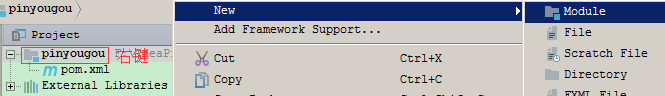
pinyougou-sellergoods-service/src/main/java下面创建包

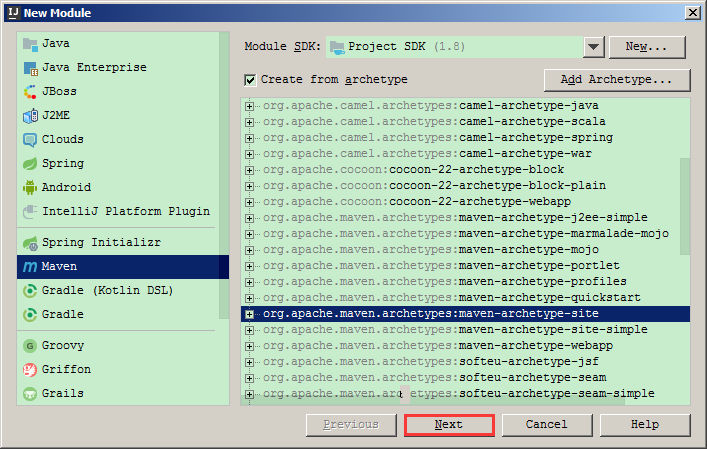
com.pinyougou.sellergoods.service.impl

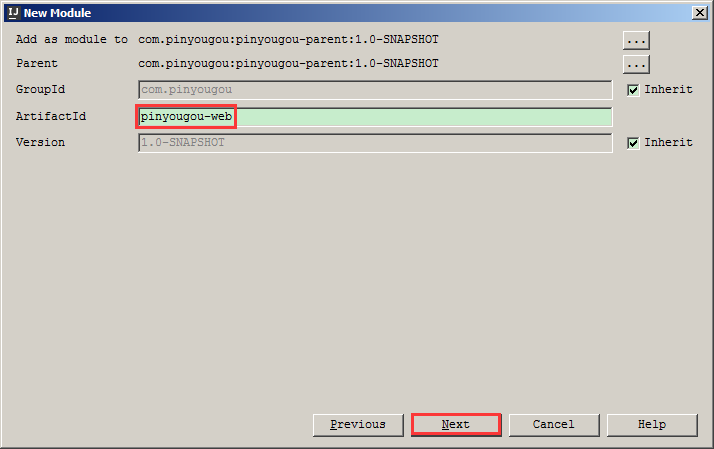
pinyougou-sellergoods-service/src/main/resources下创建Spring配置文件applicationContext-service.xml

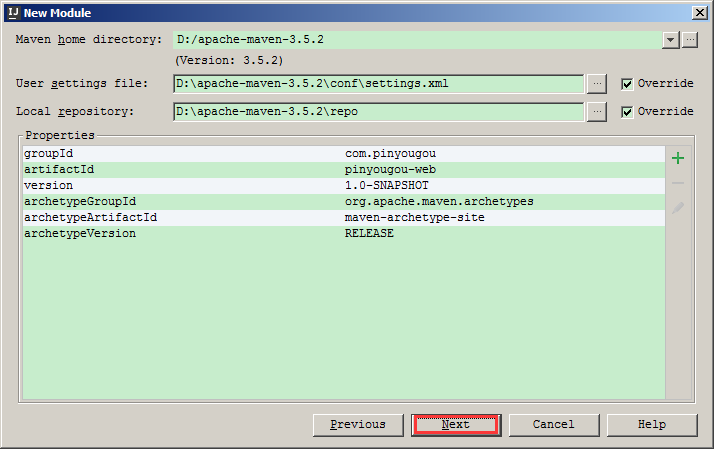
|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  *<!-- 配置加载属性文件 -->* **<context:property-placeholder  location="classpath:props/\*.properties"/>**  *<!-- 导入数据访问配置文件 -->* <**import resource="classpath:applicationContext-mapper.xml"**/> </**beans**> |

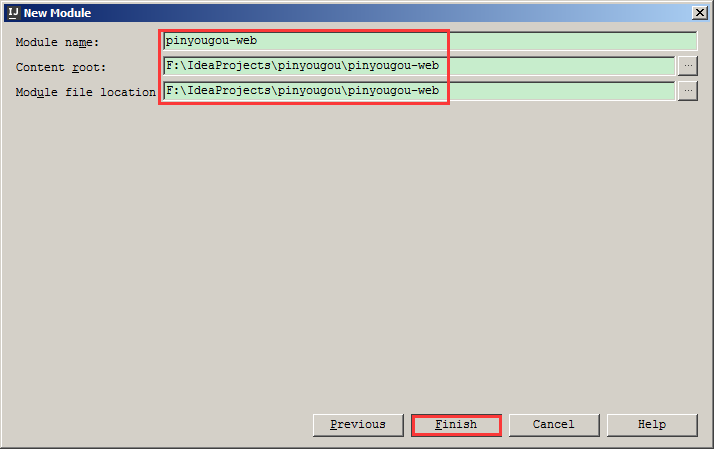
### 4.3.7 pinyougou-web服务消费者(pom)









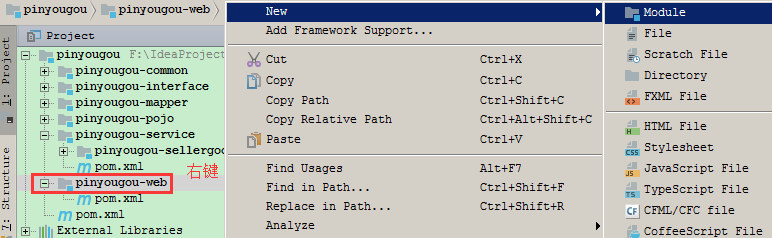


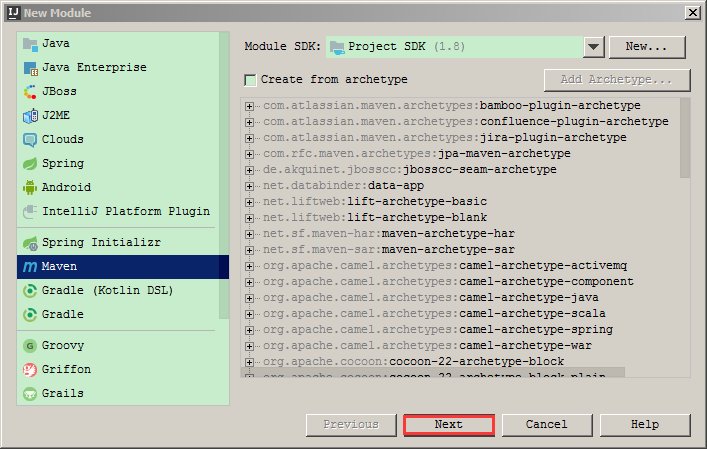


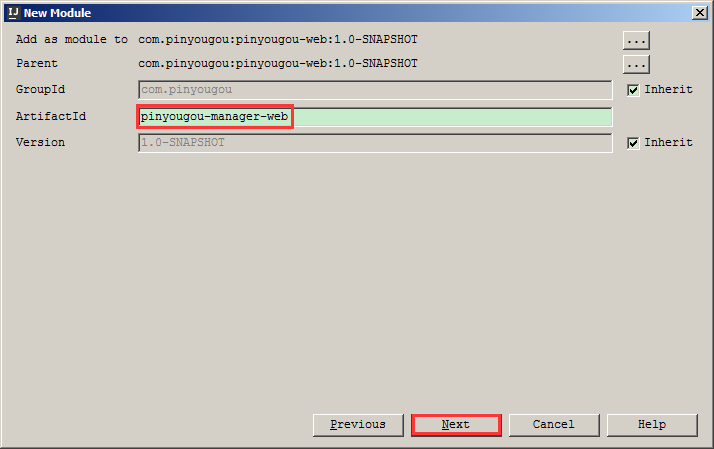
pinyougou-web/pom.xml添加依赖：

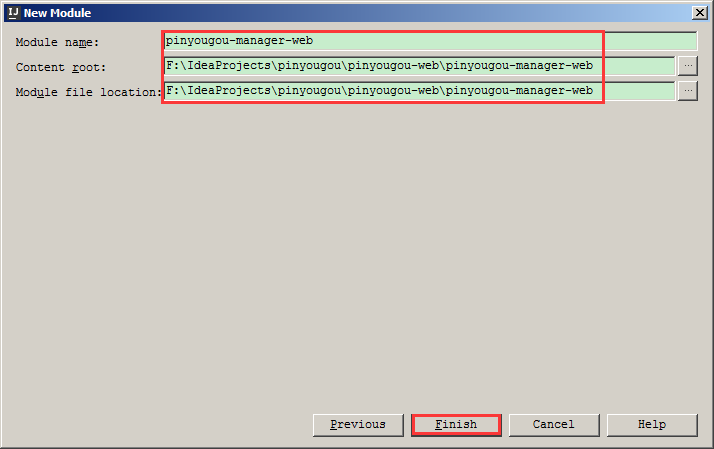
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>pinyougou-parent</**artifactId**>  <**groupId**>com.pinyougou</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>  <**artifactId**>pinyougou-web</**artifactId**>  **<packaging>pom</packaging>**  <**name**>pinyougou-web</**name**>   <**properties**>  *<!-- tomcat端口号 -->* **<port>9100</port>**  </**properties**>   *<!-- 配置依赖关系 -->* <**dependencies**>  *<!-- 日志 -->* <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-log4j12</**artifactId**>  </**dependency**>  *<!-- servlet-api -->* <**dependency**>  <**groupId**>org.apache.tomcat.embed</**groupId**>  <**artifactId**>tomcat-embed-core</**artifactId**>  <**scope**>provided</**scope**>  </**dependency**>  *<!-- JSON处理工具包 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  </**dependency**>  *<!-- apache工具组件 -->* <**dependency**>  <**groupId**>org.apache.commons</**groupId**>  <**artifactId**>commons-lang3</**artifactId**>  </**dependency**>  *<!-- spring4 -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  </**dependency**>  *<!-- 加密解密 -->* <**dependency**>  <**groupId**>commons-codec</**groupId**>  <**artifactId**>commons-codec</**artifactId**>  </**dependency**>  *<!-- pinyougou-interface -->* <**dependency**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  </**dependencies**>   *<!-- 构建部分 -->* <**build**>  <**plugins**>  *<!-- 配置tomcat插件 -->* <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**configuration**>   **<port>${port}</port>**  <**path**>/</**path**>  </**configuration**>  </**plugin**>  </**plugins**>  </**build**> </**project**> |

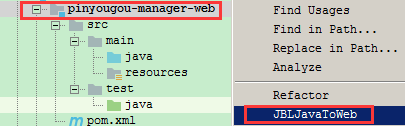
#### 4.3.7.1 pinyougou-manager-web运营商管理后台(war)











pinyougou-manager-web/pom.xml添加依赖

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>pinyougou-web</**artifactId**>  <**groupId**>com.pinyougou</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>  <**artifactId**>pinyougou-manager-web</**artifactId**>  <**packaging**>war</**packaging**>   <**properties**>  *<!-- 覆盖父工程中的port属性 -->* **<port>9101</port>**  </**properties**> </**project**> |

修改pinyougou-manager-web/src/main/webapps/WEB-INF/web.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  version="3.0"**>   *<!-- 配置Spring MVC前端控制器(核心控制器) -->* <**servlet**>  <**servlet-name**>pinyougou</**servlet-name**> <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  <**init-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:springmvc.xml</**param-value**>  </**init-param**>  <**load-on-startup**>1</**load-on-startup**>  </**servlet**>  <**servlet-mapping**>  <**servlet-name**>pinyougou</**servlet-name**>  <**url-pattern**>/</**url-pattern**>  </**servlet-mapping**>   *<!-- 配置全局编码过滤器 -->* <**filter**>  <**filter-name**>characterEncodingFilter</**filter-name**> <**filter-class**>org.springframework.web.filter.CharacterEncodingFilter</**filter-class**>  <**init-param**>  <**param-name**>encoding</**param-name**>  <**param-value**>UTF-8</**param-value**>  </**init-param**>  </**filter**>  <**filter-mapping**>  <**filter-name**>characterEncodingFilter</**filter-name**>  <**url-pattern**>/\*</**url-pattern**>  </**filter-mapping**>   *<!-- 欢迎列表 -->* <**welcome-file-list**>  <**welcome-file**>index.html</**welcome-file**>  </**welcome-file-list**> </**web-app**> |

pinyougou-manager-web/src/main/resources/目录下创建springmvc.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  *<!-- 开启组件扫描 -->* <**context:component-scan  base-package="com.pinyougou.manager.controller"**/>  *<!-- 开启MVC注解驱动 -->* <**mvc:annotation-driven**>  *<!-- 配置消息转换 -->* <**mvc:message-converters**>  *<!-- 配置用fastjson作为json处理框架 -->* <**bean class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter"**>  *<!-- 配置支持的类型 -->* <**property name="supportedMediaTypes"  value="application/json;charset=UTF-8"**/>  </**bean**>  </**mvc:message-converters**>  </**mvc:annotation-driven**>  *<!-- 配置静态资源用WEB容器默认的servlet来处理 -->* <**mvc:default-servlet-handler**/> </**beans**> |

pinyougou-manager-web/src/main/java目录下创建包名com.pinyougou.manager.controller

pinyougou-manager-web/src/main/resources/log4j.properties

|  |
| --- |
| **log4j.rootLogger**=**DEBUG,stdout log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern**=**%-d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c]-[%p] %m%n** |

# 5.品优购【查询品牌列表】

## 5.1需求分析

完成品牌管理的后端代码，在浏览器可查询品牌的数据（json格式）

## 5.2数据库表

**tb\_brand 品牌表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 含义 |
| id | bigint |  | 主键 |
| name | varchar | 255 | 品牌名称 |
| first\_char | varchar | 1 | 品牌首字母 |

## 5.3后端代码

### 5.3.1实体类

pinyougou-pojo/src/main/java/com.pinyougou.pojo/包下创建Brand.java实体类。

### 5.3.2数据访问

1. 在pinyougou-mapper/src/main/java/com.pinyougou.mapper包下创建BrandMapper.java接口类。

|  |
| --- |
| **public interface** BrandMapper {  */\*\* 查询全部品牌 \*/* @Select(**"select \* from tb\_brand order by id asc"**)  List<Brand> findAll(); } |

1. 修改pinyougou-mapper/src/main/resources/mappers/BrandMapper.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"** *?>* **<!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>*** <**mapper namespace="com.pinyougou.mapper.BrandMapper"** </**mapper**> |

### 5.3.3服务接口

在pinyougou-interface工程的src/main/java/com.pinyougou.service包下创建BrandService.java服务接口：

|  |
| --- |
| **public interface** BrandService {  */\*\* 查询所有品牌 \*/* List<Brand> findAll(); } |

### 5.3.4服务实现

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl包下创建BrandServiceImpl.java

|  |
| --- |
| **package** com.pinyougou.sellergoods.service.impl;  **import** com.pinyougou.mapper.BrandMapper; **import** com.pinyougou.pojo.Brand; **import** com.pinyougou.service.BrandService; **import** org.springframework.beans.factory.annotation.Autowired; **import** java.util.List;  **public class** BrandServiceImpl **implements** BrandService {  */\*\* 注入数据访问接口代理对象 \*/* @Autowired  **private** BrandMapper **brandMapper**;  @Override  **public** List<Brand> findAll() {  **return brandMapper**.findAll();  } } |

### 5.3.5控制器

在pinyougou-manager-web工程src/main/java的

com.pinyougou.manager.controller包下创建BrandController.java

|  |
| --- |
| **package** com.pinyougou.manager.controller; **import** com.pinyougou.pojo.Brand; **import** com.pinyougou.service.BrandService; **import** org.springframework.web.bind.annotation.GetMapping; **import** org.springframework.web.bind.annotation.RestController; **import** java.util.List;  @RestController **public class** BrandController {  @Autowired(required=false) **private** BrandService **brandService**;  */\*\* 查询全部品牌 \*/* @GetMapping(**"/brand/findAll"**)  **public** List<Brand> findAll(){  **return brandService**.findAll();  } } |

## 5.4 Dubbo服务提供者【商家服务】

### 5.4.1配置依赖

pinyougou-service/pom.xml添加依赖

|  |
| --- |
| *<!-- dubbo -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**> </**dependency**> *<!-- zookeeper分布式协调服务 -->* <**dependency**>  <**groupId**>org.apache.zookeeper</**groupId**>  <**artifactId**>zookeeper</**artifactId**> </**dependency**> *<!-- zkclient(zookeeper客户端) -->* <**dependency**>  <**groupId**>com.github.sgroschupf</**groupId**>  <**artifactId**>zkclient</**artifactId**> </**dependency**> |

### 5.4.2配置文件

pinyougou-sellergoods-service/src/main/resources/applicationContext-service.xml 配置整合Dubbo

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://code.alibabatech.com/schema/dubbo  http://code.alibabatech.com/schema/dubbo/dubbo.xsd"**>  *<!-- 配置加载属性文件 -->* <**context:property-placeholder  location="classpath:props/db.properties"**/>   *<!-- 导入数据访问配置文件 -->* <**import resource="classpath:applicationContext-mapper.xml"**/>   *<!-- ############## 配置dubbo服务提供者 ############## -->  <!-- 配置当前应用的名称 -->* <**dubbo:application name="pinyougou-sellergoods-service"**/>  *<!-- 配置zookeeper作为注册中心，注册服务地址 -->* <**dubbo:registry protocol="zookeeper" address="192.168.12.131:2181"**/>  *<!-- 用dubbo协议在20880端口暴露服务 -->* <**dubbo:protocol name="dubbo" port="20880"**/>  *<!-- 配置采用包扫描来暴露服务 -->* **<dubbo:annotation package="com.pinyougou.sellergoods.service.impl"/>** </**beans**> |

### 5.4.3服务实现类

说明：服务接口实现类上面需加**Dubbo的注解**：@Service

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl包下的BrandServiceImpl.java

|  |
| --- |
| **package** com.pinyougou.sellergoods.service.impl;  **import com.alibaba.dubbo.config.annotation.Service;** **import** com.pinyougou.mapper.BrandMapper; **import** com.pinyougou.pojo.Brand; **import** com.pinyougou.service.BrandService; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.transaction.annotation.Transactional; **import** java.util.List; @Service(interfaceName=**"com.pinyougou.service.BrandService"**)@Transactional **public class** BrandServiceImpl **implements** BrandService {  */\*\* 注入数据访问接口代理对象 \*/* @Autowired  **private** BrandMapper **brandMapper**;  @Override  **public** List<Brand> findAll() {  **return brandMapper**.findAll();  } } |

**@Service注解包名为：com.alibaba.dubbo.config.annotation.Service**

## 5.5 Dubbo服务消费者【运营商后台】

### 5.5.1配置依赖

pinyougou-web/pom.xml添加依赖

|  |
| --- |
| *<!-- dubbo -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**> </**dependency**> *<!-- zookeeper分布式协调服务 -->* <**dependency**>  <**groupId**>org.apache.zookeeper</**groupId**>  <**artifactId**>zookeeper</**artifactId**> </**dependency**> *<!-- zkclient(zookeeper客户端) -->* <**dependency**>  <**groupId**>com.github.sgroschupf</**groupId**>  <**artifactId**>zkclient</**artifactId**> </**dependency**> |

### 5.5.2配置文件

pinyougou-manager-web/src/main/resources/springmvc.xml整合Dubbo

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://code.alibabatech.com/schema/dubbo  http://code.alibabatech.com/schema/dubbo/dubbo.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  *~~<!-- 开启组件扫描 -->~~* ~~<~~**~~context:component-scan  base-package="com.pinyougou.manager.controller"~~**~~/>~~  *<!-- 开启MVC注解驱动 -->* <**mvc:annotation-driven**>  *<!-- 配置消息转换 -->* <**mvc:message-converters**>  *<!-- 配置用fastjson作为json处理框架 -->* <**bean class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter"**>  *<!-- 配置支持的类型 -->* <**property name="supportedMediaTypes"  value="application/json;charset=UTF-8"**/>  </**bean**>  </**mvc:message-converters**>  </**mvc:annotation-driven**>  *<!-- 配置静态资源用WEB容器默认的servlet来处理 -->* <**mvc:default-servlet-handler**/>  *<!-- ############## 配置dubbo服务消费者 ############## -->  <!-- 配置当前应用的名称 -->* <**dubbo:application name="pinyougou-manager-web"**/>  *<!-- 配置zookeeper作为注册中心，注册服务地址 -->* <**dubbo:registry protocol="zookeeper" address="192.168.12.131:2181"**/>  ***<!-- 配置采用包扫描来引用服务，产生服务接口的代理对象 -->* <dubbo:annotation package="com.pinyougou.manager.controller"/>** </**beans**> |

### 5.5.3控制器类

说明：在控制器类中的服务接口上面加**Dubbo的注解**：@Reference

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller包下的BrandController.java

|  |
| --- |
| **package** com.pinyougou.manager.controller; **import com.alibaba.dubbo.config.annotation.Reference;** **import** com.pinyougou.pojo.Brand; **import** com.pinyougou.service.BrandService; **import** org.springframework.web.bind.annotation.GetMapping; **import** org.springframework.web.bind.annotation.RestController; **import** java.util.List; @RestController **public class** BrandController {  */\*\*  \* 引用服务接口代理对象  \* timeout: 调用服务接口方法超时时间毫秒数  \*/* **@Reference(timeout = 10000)**  **private** BrandService **brandService**;  */\*\* 查询全部品牌 \*/* @GetMapping(**"/brand/findAll"**)  **public** List<Brand> findAll(){  **return brandService**.findAll();  } } |

**@Reference注解包名为：com.alibaba.dubbo.config.annotation.Reference**

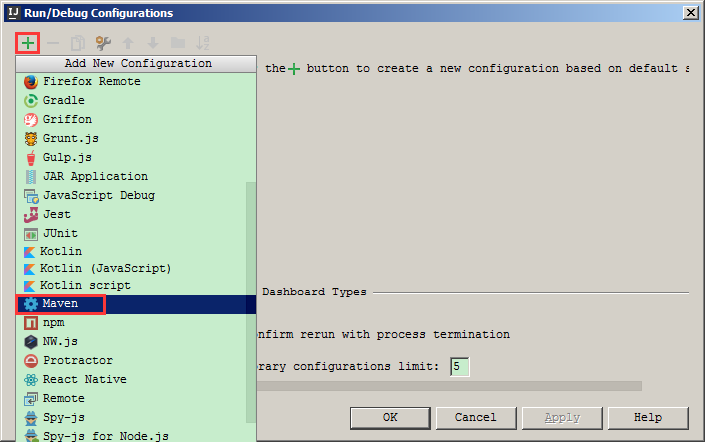
**具体调用流程参考：【资料/Dubbo调用流程图.png】**

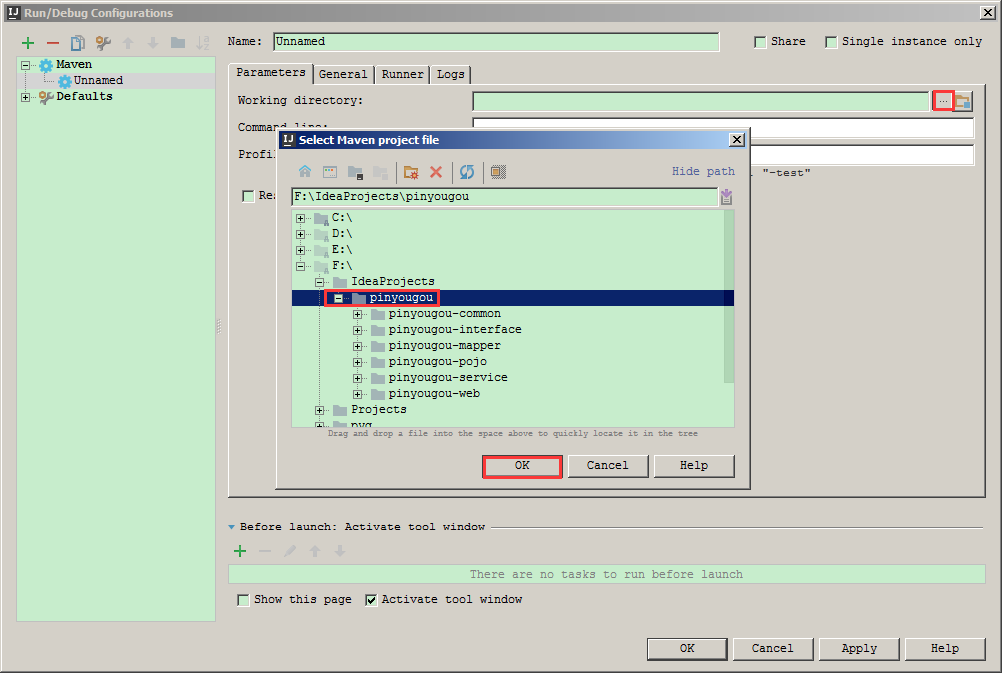
## 5.6测试

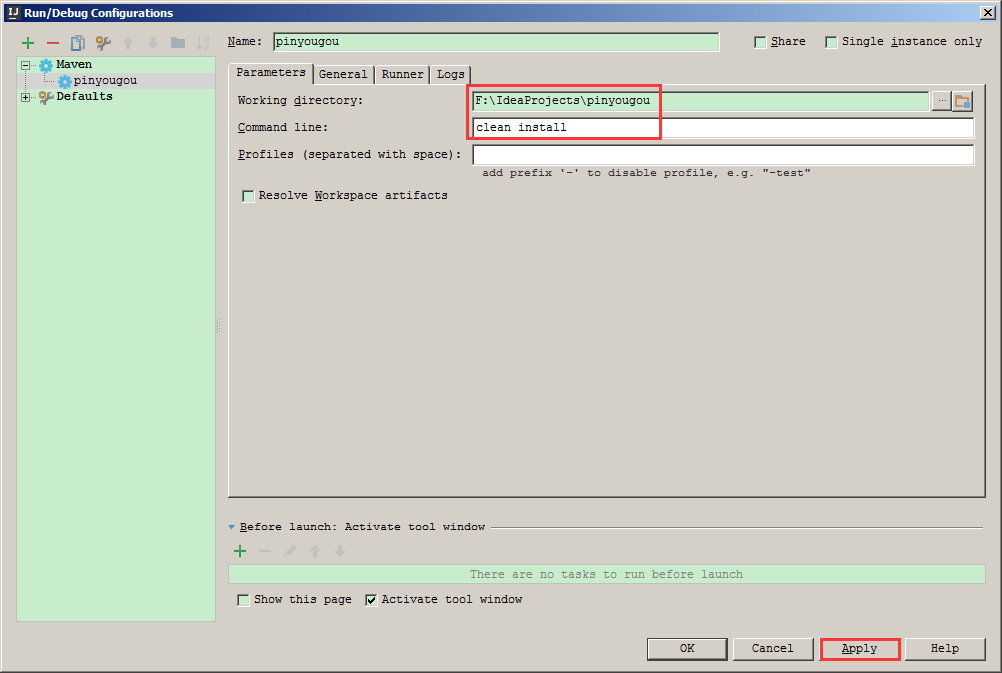
安装品优购所有模块：

**pinyougou: clean install**





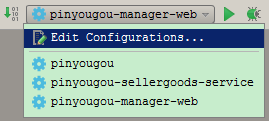




按上面方式配置下面两个Web工程：

pinyougou-sellergoods-service tomcat7:run

pinyougou-manager-web tomcat7:run



清理安装：

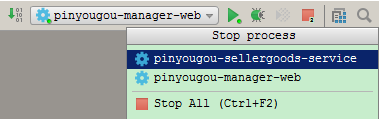
pinyougou



启动：

pinyougou-sellergoods-service

pinyougou-manager-web



在浏览器地址栏输入：http://localhost:9101/brand/findAll

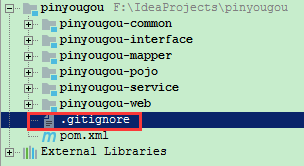


# 6.Git版本控制

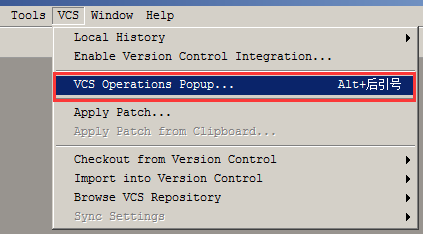
## 6.1提交代码到本地仓库

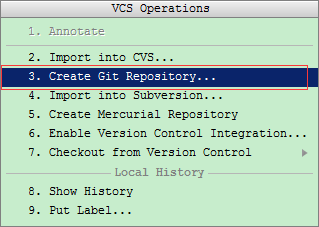
第一步：添加忽略文件

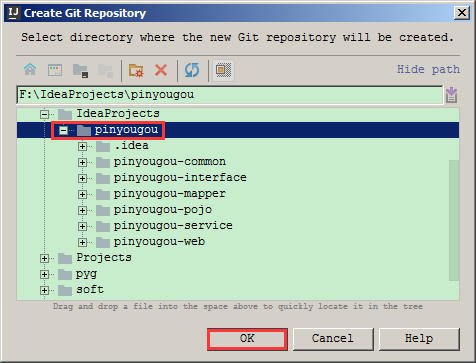
拷贝【资料\.gitignore】到 pinyougou项目下面



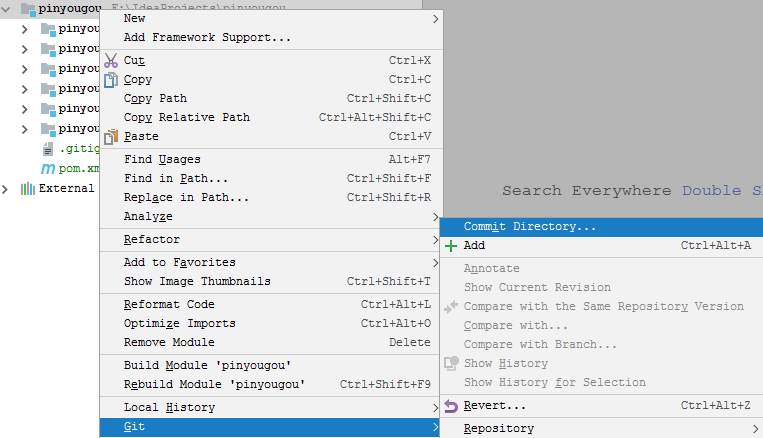
第二步：选择版本控制操作菜单

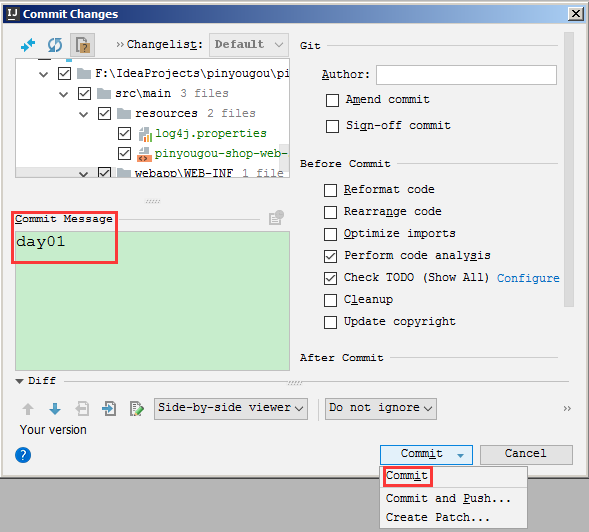


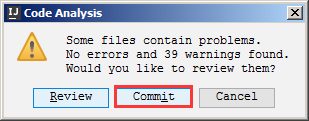




第三步：提交到本地仓库







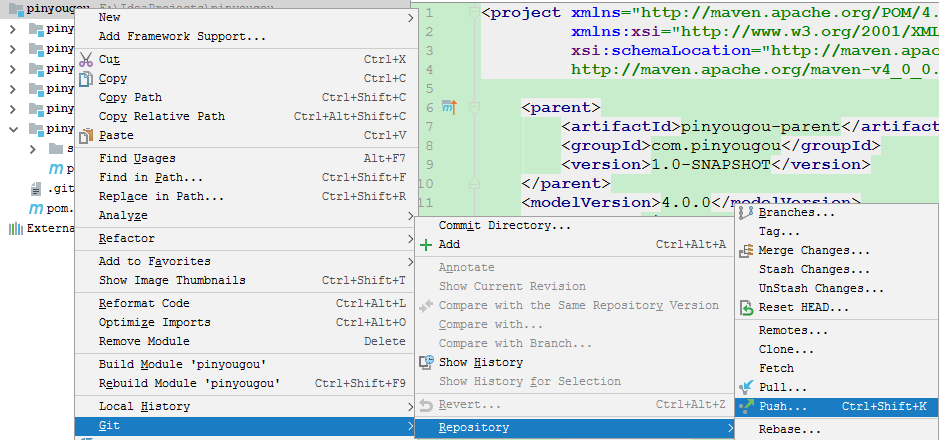
## 6.2同步代码到远程仓库

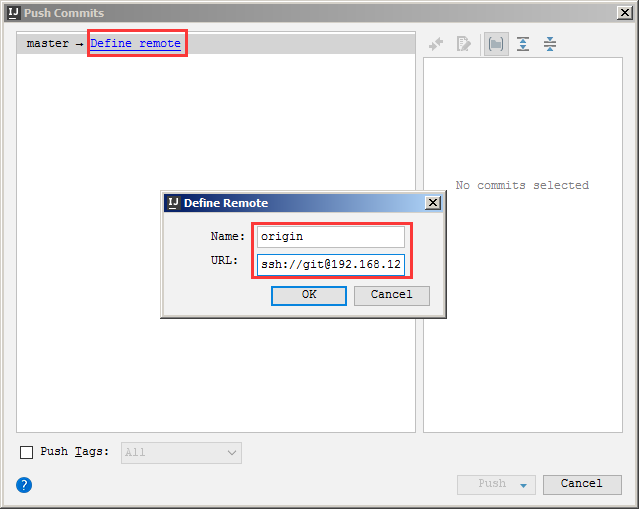
第一步：在linux系统中创建git远程仓库

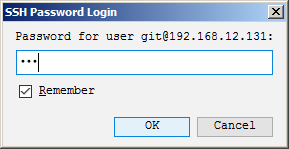
|  |
| --- |
| rm -rf /home/git/pinyougou.git  git init --bare /home/git/pinyougou.git  # 授予git用户为 /home/git目录及所有子目录的拥有者  **chown -R git:git /home/git** |

第二步：设置远程仓库，并推送本地仓库到远程库

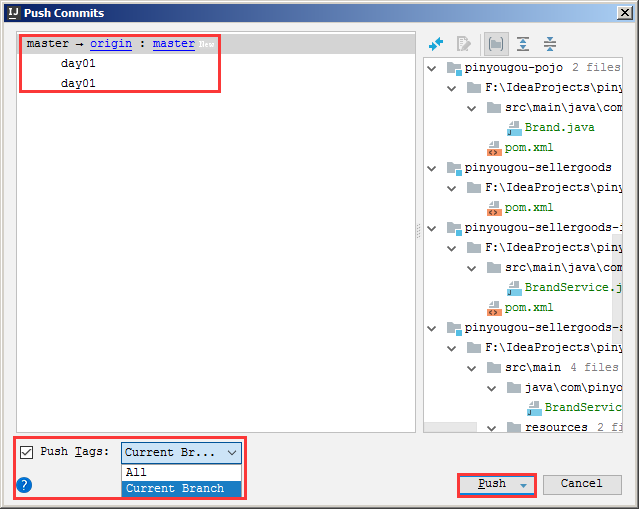
**连接git服务器地址：ssh://git@192.168.12.131/home/git/pinyougou.git**





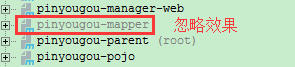


密码：git

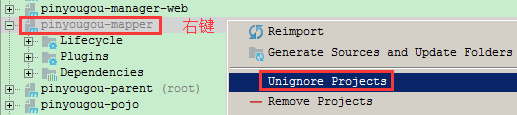




# 7.Intellij IDEA忽略项目



注意：如果创建出来的模块是“灰色”代表此模块已怱略，该模块不会编译、不能导入jar包。如果想不想怱略该模块，设置如下图：



# 附录：常见错误

## 1.在注册中心找不到对应的服务

|  |
| --- |
| java.lang.IllegalStateException: Failed to check the status of the service com.pinyougou.sellergoods.service.BrandService. No provider available for the service com.pinyougou.sellergoods.service.BrandService from the url zookeeper://192.168.12.128:2181/com.alibaba.dubbo.registry.RegistryService?application=pinyougou-manager-web&dubbo=2.5.7&interface=com.pinyougou.sellergoods.service.BrandService&methods=update,get,delete,selectOptionList,add,getListByPage&pid=3980&revision=0.0.1-SNAPSHOT&side=consumer&timestamp=1501146823396 to the consumer 172.16.17.14 use dubbo version 2.5.7 |

这种错误是服务层代码没有成功注册到注册中心导致，请检查一下你的服务层代码是否添加了@Service注解，

并且该注解的包一定是com.alibaba.dubbo.config.annotation包，不是org.springframework.stereotype.Service，这个地方极容易出错。另外还有一个原因就是你的服务层工程由于某些原因没有正常启动，也无法注册到注册中心里。

**@Service注解：com.alibaba.dubbo.config.annotation.Service包**

**@Reference注解：com.alibaba.dubbo.config.annotation.Reference包**

*/\*\* 引用服务，调用服务超时的毫秒数 \*/*@Reference(timeout = 10000)

## 2.无法连接到注册中心

|  |
| --- |
| org.I0Itec.zkclient.exception.ZkTimeoutException: Unable to connect to zookeeper server within timeout: 5000 org.I0Itec.zkclient.ZkClient.connect(ZkClient.java:876) org.I0Itec.zkclient.ZkClient.<init>(ZkClient.java:98) org.I0Itec.zkclient.ZkClient.<init>(ZkClient.java:92) org.I0Itec.zkclient.ZkClient.<init>(ZkClient.java:80)  com.alibaba.dubbo.remoting.zookeeper.zkclient.ZkclientZookeeperClient.<init>(ZkclientZookeeperClient.java:26) |

请检查**IP与端口**是否填写正确，**检查注册中心是否正常启动**