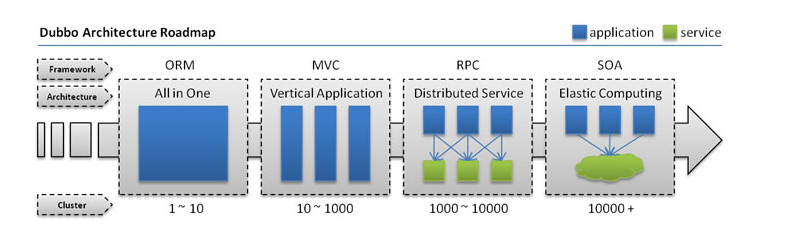
谷粒商城

版本：V 1.0

# 一、分布式架构

## 1 分布式架构的演进



### 1.1 单一应用架构



适用于小型网站，小型管理系统，将所有功能都部署到一个功能里，简单易用。

缺点： 1、性能扩展比较难

2、协同开发问题

3、不利于升级维护

### 1.2 垂直应用架构

通过切分业务来实现各个模块独立部署，降低了维护和部署的难度，团队各司其职更易管理，性能扩展也更方便，更有针对性。

缺点： 公用模块无法重复利用，开发性的浪费



### 1.3 分布式应用架构

将各个应用通过分层独立出来，可以利用**rpc**实现web与service、service与service的互相调用，提高了代码的复用性。

缺点： 每个调用的模块要存储一份完整的被调用模块的位置和状态，一旦位置和状态发生变化，就要更新所有涉及的配置。

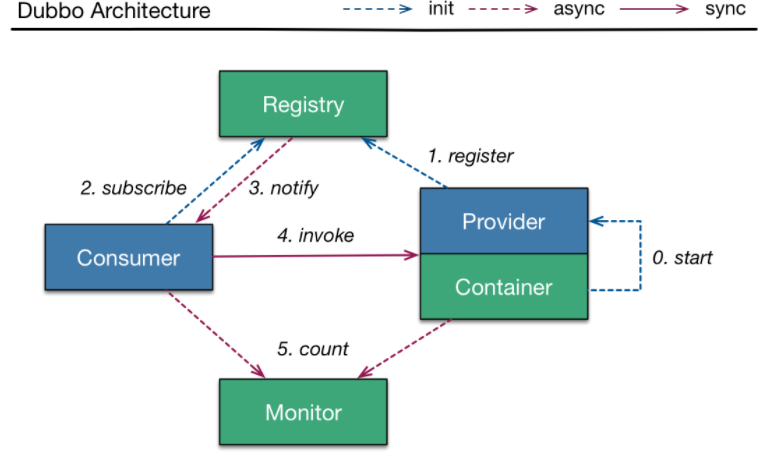


### 1.4 面向服务的分布式架构

随着架构不断增大，服务节点也越来越多，服务之间的调用和依赖关系也越来越复杂，需要有一个统一的中心来调度、路由、管理所有的服务，基于这个中心构建的这个星型架构就是现在目前最主流的SOA分布式架构。



### 1.5 dubbo架构图



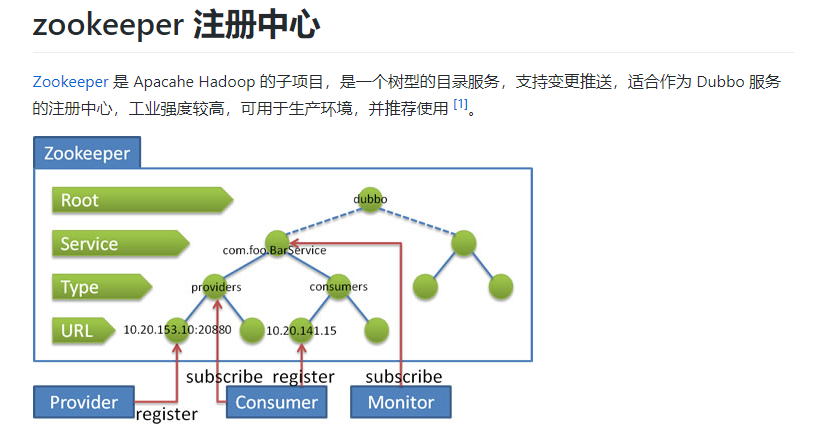
Dubbo ：注册中心采用zookeeper！

Dubbo : 订阅发布服务功能！

### 1.6 dubbo注册中心



官网：推荐使用 [Zookeeper 注册中心](http://dubbo.apache.org/zh-cn/docs/user/references/registry/zookeeper.html)



使用：

1. 依赖jar 包
2. 配置
   1. 配置单机版 <dubbo:registry address="zookeeper://10.20.153.10:2181" />
   2. 配置集群版

<dubbo:registry address="zookeeper://10.20.153.10:2181?backup=10.20.153.11:2181,10.20.153.12:2181" />

### 1.7 使用dubbo

<http://dubbo.apache.org/zh-cn/docs/user/configuration/annotation.html>

### 1.8 最佳实战

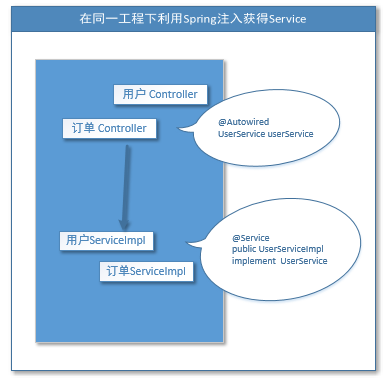
1. 配置版本
2. 序列化

### 1.9 推荐用法



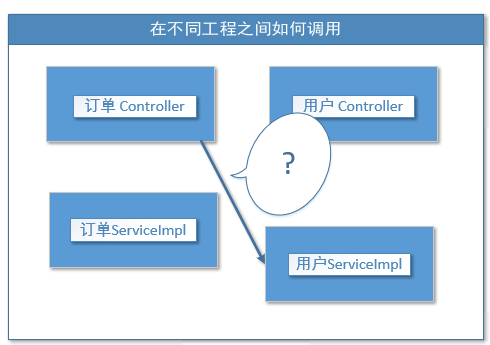
## 2 如何实现这种SOA架构

原来所有的controller、service接口、service实现都在一个工程，通过Spring的ioc就可以实现互相调用。

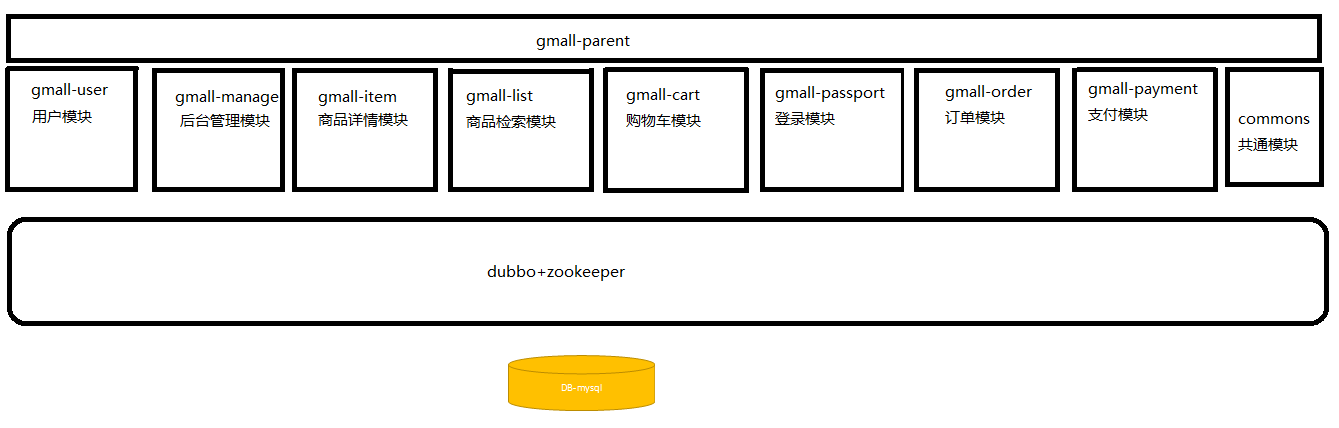


那么假如controller和service实现隶属于不同的应用如何实现调用呢？

答案：rpc，dubbo, spring cloud



## 3 项目构成



1. Commons 共通模块内容
   1. gmall-parent[定义jar包的版本]，springBoot，dubbo，zookeeper，apache工具类
2. 共通中又分为：web-util，service-util【redis】
   1. Web-util 内容包括
      1. Commons，thymeleaf【前端显示模板】
      2. **springBoot可以要jsp作为显示的模板，需要自己手动配置**

|  |
| --- |
| spring.mvc.view.suffix=.jsp spring.mvc.view.prefix=/WEB-INF/jsp/ |

* 1. Service-util 内容包括
     1. Commons，mysql，redis，activemq【消息队列-分布式事务】

# 二、分布式工程的模块搭建

在新建项目的时候，注意修改maven的本地仓库

配置idea 的信息

调整idea占用磁盘空间

在安装目录：

idea.properties

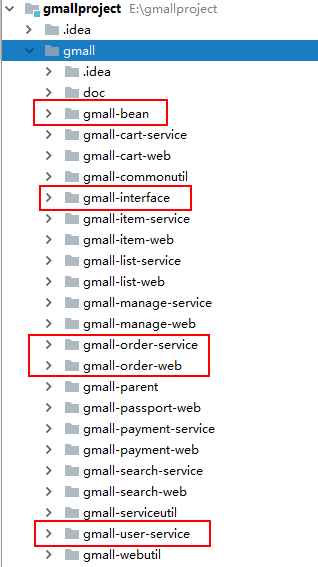
idea.config.path=E:/idea/config

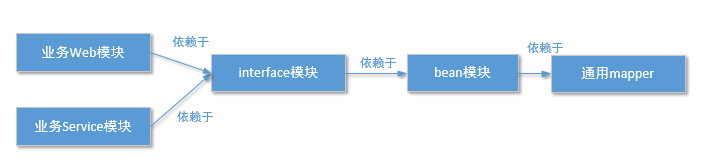
idea.system.path=E:/idea/system

默认配置如下：

# idea.config.path=${user.home}/.IntelliJIdea/config

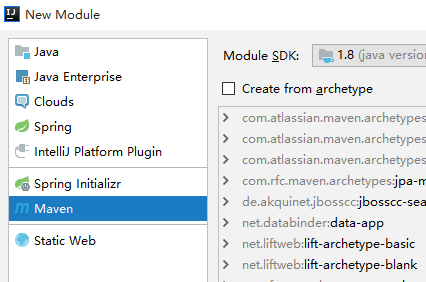
# idea.system.path=${user.home}/.IntelliJIdea/system

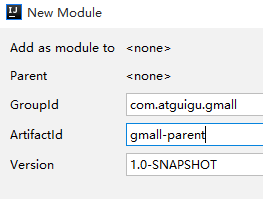


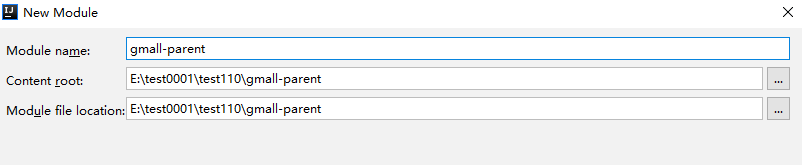


依赖关系使用Maven 构建：依赖，继承，聚合【pom.xml packaging pom】

### 1 搭建parent模块



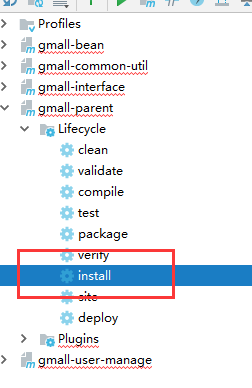




parent模块的 pom.xml

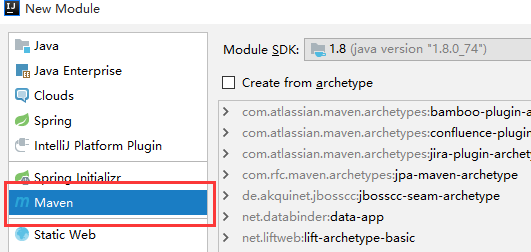
|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-parent</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>pom</packaging>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>   <fastjson.version>1.2.46</fastjson.version>  <dubbo-starter.version>1.0.10</dubbo-starter.version>  <dubbo.version>2.6.0</dubbo.version>  <zkclient.version>0.10</zkclient.version>  <mybatis.version>1.3.1</mybatis.version>  <nekohtml.version>1.9.20</nekohtml.version>  <xml-apis.version>1.4.01</xml-apis.version>  <batik-ext.version>1.9.1</batik-ext.version>  <jsoup.version>1.11.2</jsoup.version>  <httpclient.version>4.5.5</httpclient.version>  <commons-lang3.version>3.7</commons-lang3.version>  <mapper-starter.version>1.2.3</mapper-starter.version>  <jedis.version>2.9.0</jedis.version>  <jest.version>5.3.3</jest.version>  <jna.version>4.5.1</jna.version>  <beanUtils.version>1.9.3</beanUtils.version>  </properties>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.10.RELEASE</version>  <relativePath/> *<!-- lookup parent from repository -->* </parent>   <dependencyManagement>  <dependencies>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>${fastjson.version}</version>  </dependency>   <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <version>${dubbo.version}</version>  </dependency>   <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>${zkclient.version}</version>  </dependency>   <dependency>  <groupId>com.gitee.reger</groupId>  <artifactId>spring-boot-starter-dubbo</artifactId>  <version>${dubbo-starter.version}</version>  </dependency>   <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>${mybatis.version}</version>  </dependency>    <dependency>  <groupId>net.sourceforge.nekohtml</groupId>  <artifactId>nekohtml</artifactId>  <version>${nekohtml.version}</version>  </dependency>   <dependency>  <groupId>xml-apis</groupId>  <artifactId>xml-apis</artifactId>  <version>${xml-apis.version}</version>  </dependency>   <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-ext</artifactId>  <version>${batik-ext.version}</version>  </dependency>   *<!-- https://mvnrepository.com/artifact/org.jsoup/jsoup -->* <dependency>  <groupId>org.jsoup</groupId>  <artifactId>jsoup</artifactId>  <version>${jsoup.version}</version>  </dependency>   *<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->* <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  <version>${httpclient.version}</version>  </dependency>   <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-lang3</artifactId>  <version>${commons-lang3.version}</version>  </dependency>    <dependency>  <groupId>tk.mybatis</groupId>  <artifactId>mapper-spring-boot-starter</artifactId>  <version>${mapper-starter.version}</version>  </dependency>   <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>${jedis.version}</version>  </dependency>   *<!-- https://mvnrepository.com/artifact/io.searchbox/jest -->* <dependency>  <groupId>io.searchbox</groupId>  <artifactId>jest</artifactId>  <version>${jest.version}</version>  </dependency>   *<!-- https://mvnrepository.com/artifact/net.java.dev.jna/jna -->* <dependency>  <groupId>net.java.dev.jna</groupId>  <artifactId>jna</artifactId>  <version>${jna.version}</version>  </dependency>   <dependency>  <groupId>commons-beanutils</groupId>  <artifactId>commons-beanutils</artifactId>  <version>${beanUtils.version}</version>  </dependency>    </dependencies>  </dependencyManagement> </project> |

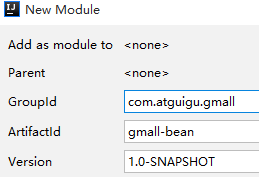
然后在idea右边菜单执行安装

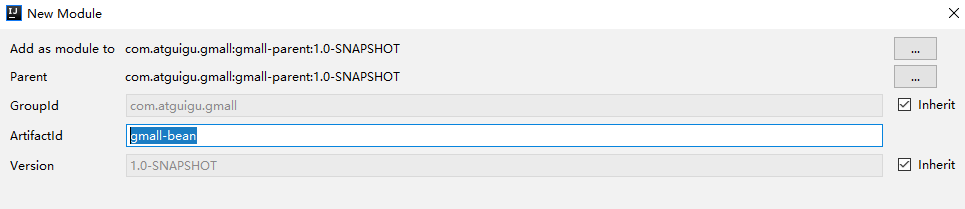


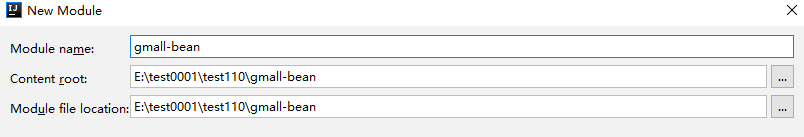
Install 之后，将项目安装到本地仓库！

### 2 创建bean模块



or





实体类：bean，pojo，entity，domain,vo ,dto

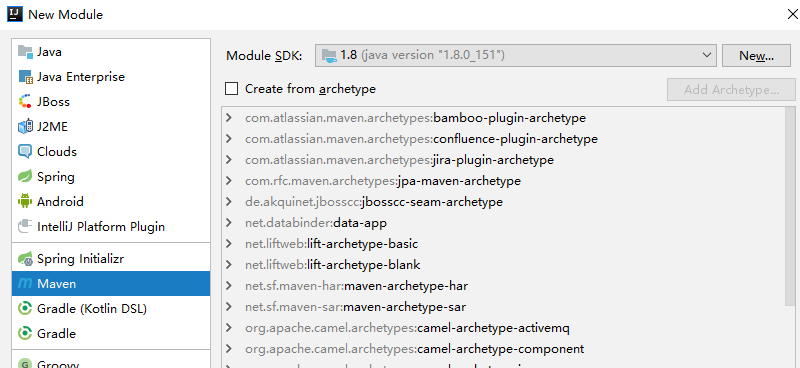
将所有的bean 类都放到该项目中 gmall-bean 中

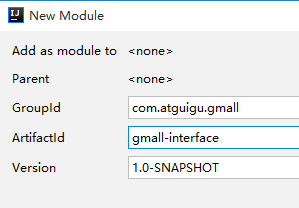
例如：UserInfo类

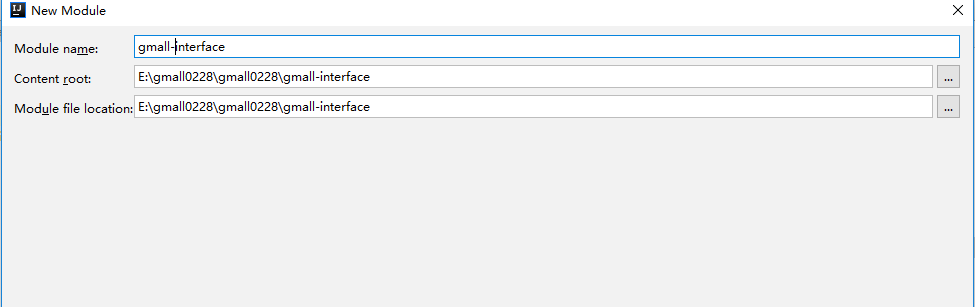
bean模块的pom.xml

|  |
| --- |
| *<?*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">  <modelVersion>4.0.0</modelVersion>   <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-bean</artifactId>  <version>1.0-SNAPSHOT</version>   <parent>  <artifactId>gmall-parent</artifactId>  <groupId>com.atguigu.gmall</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <dependencies>  <dependency>  <groupId>tk.mybatis</groupId>  <artifactId>mapper-spring-boot-starter</artifactId>  <exclusions>  <exclusion>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId>  </exclusion>  </exclusions>  </dependency>  <**dependency**>  <**groupId**>org.projectlombok</**groupId**>  <**artifactId**>lombok</**artifactId**> </**dependency**>  </dependencies> </project> |

### 3 搭建interface模块







interface的pom.xml

|  |
| --- |
| *<?*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">   <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-interface</artifactId>  <version>1.0-SNAPSHOT</version>  <modelVersion>4.0.0</modelVersion>  <packaging>jar</packaging>  <**parent**>  <**groupId**>com.atguigu.gmall</**groupId**>  <**artifactId**>gmall-parent</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**> </**parent**>  <dependencies>  <dependency>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-bean</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

将原来的UserInfoService 模块放到该模块中

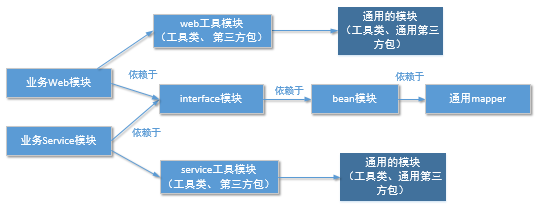
如果写mapper 不利于后期维护，mapper 应该放在具体的模块项目中-service。

### 4 搭建 util模块

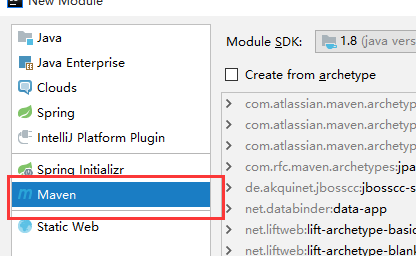
首先我们可以把所有的第三方依赖包分为三种

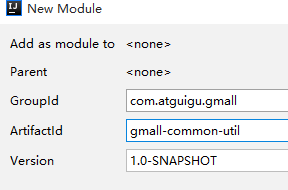
|  |
| --- |
| 1. web业务模块用到的第三方包,比如文件上传客户端、页面渲染工具、操作cookie的工具类等等。 2. service业务模块用到的第三方包，比如jdbc、mybatis、jedis、activemq工具包等等。 3. 通用型的第三方包，比如fastjson、httpclient、apache工具包等等。 4. 只有本模块用到的 es |

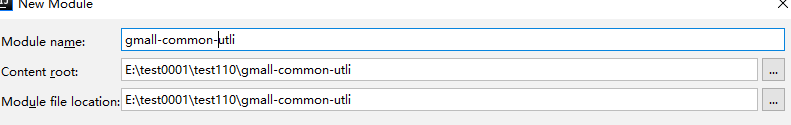
基于这三种情况我们可以搭建如下的依赖结构：



#### 创建common-util的模块







pom.xml文件

首先先分析具体哪些包是通用的

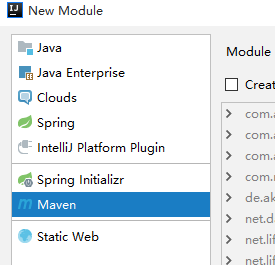
gmall-common-util

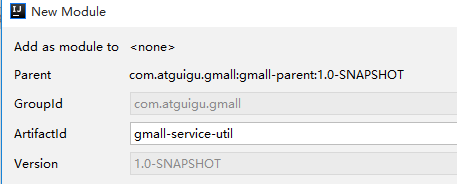
|  |  |
| --- | --- |
| spring-boot-starter-test | 测试(springboot有默认版本号) |
| spring-boot-starter-web | 内含tomcat容器、HttpSevrletRequest等  (springboot有默认版本号) |
| fastjson | json工具 |
| commons-lang3 | 方便好用的apache工具库 |
| commons-beanutils | 方便好用的apache处理实体bean工具库 |
| commons-codec | 方便好用的apache解码工具库sso用到 |
| httpclient | restful调用客户端 |

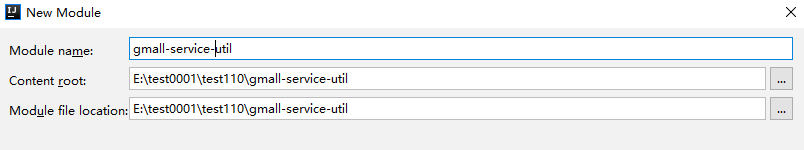
gmall-common-util的pom.xml

|  |
| --- |
| *<?*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">  <modelVersion>4.0.0</modelVersion>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-common-util</artifactId>  <version>1.0-SNAPSHOT</version>  <parent>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-parent</artifactId>  <version>1.0-SNAPSHOT</version>  </parent>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  </dependency>   <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>   <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  </dependency> <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-lang3</artifactId>  </dependency>   <dependency>  <groupId>commons-beanutils</groupId>  <artifactId>commons-beanutils</artifactId>  </dependency>   <dependency>  <groupId>commons-codec</groupId>  <artifactId>commons-codec</artifactId>  </dependency>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  </dependency>    </dependencies> </project> |

#### 创建service-util模块







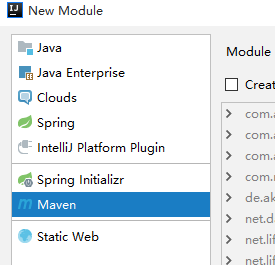
gmall-service-util

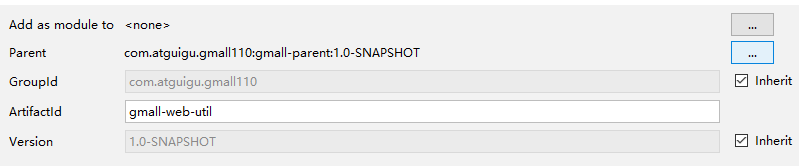
|  |  |
| --- | --- |
| spring-boot-starter-jdbc | 数据库驱动(springboot有默认版本号) |
| mysql-connector-java | 数据库连接器(springboot有默认版本号) |
| mybatis-spring-boot-starter | mybatis |

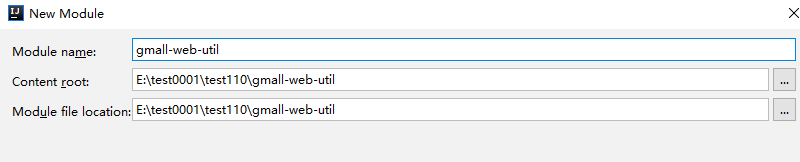
gmall-service-util的pom.xml

|  |
| --- |
| *<?*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>gmall-parent</artifactId>  <groupId>com.atguigu.gmall</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>gmall-service-util</artifactId>  <dependencies>  <dependency>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-common-util</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  </dependency>   <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <scope>runtime</scope>  </dependency>   <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  </dependency>    </dependencies> </project> |

#### 创建web-util模块







gmall-web-util

|  |  |
| --- | --- |
| thymeleaf | springboot自带页面渲染工具(springboot有默认版本号) |

gmall-web-util 项目

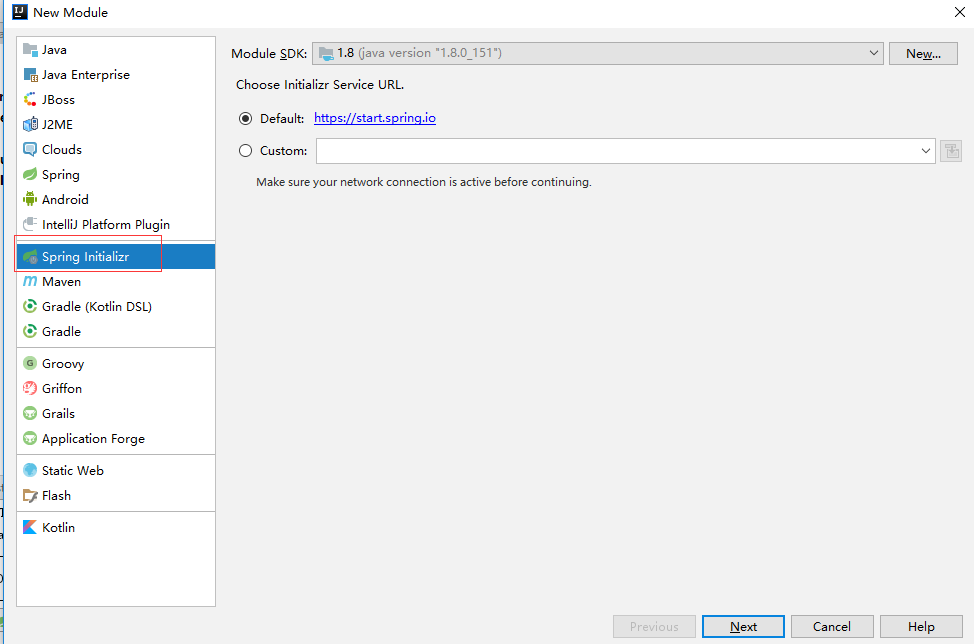
gmall-web-util的pom.xml

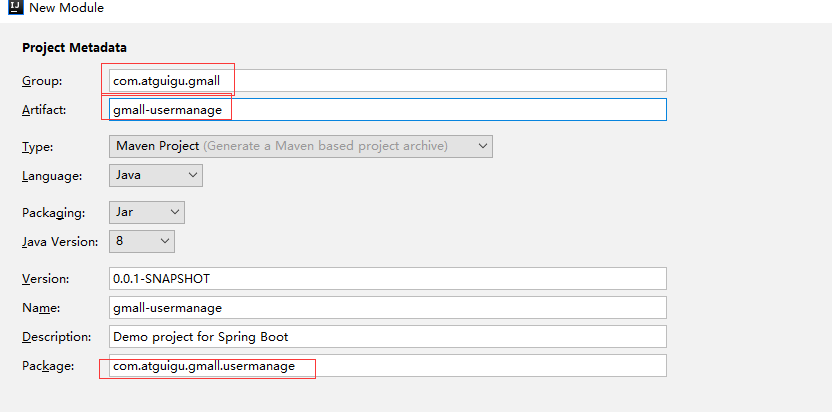
|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <parent>  <artifactId>gmall-parent</artifactId>  <groupId>com.atguigu.gmall</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>   <artifactId>gmall-web-util</artifactId>  <groupId>com.atguigu.gmall</groupId>  <version>1.0-SNAPSHOT</version>  <dependencies>  <dependency>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-common-util</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>   <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency>   </dependencies>  </project> |

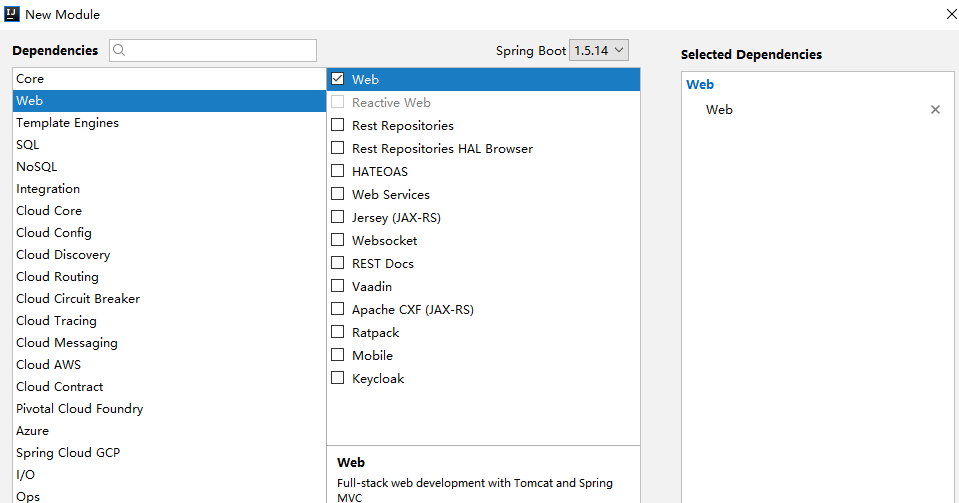
注意：到此处要休息片刻！install gmall-parent! 如果没有问题，则继续下面步骤！

### 5 重构gmall-usermanage 模块

有页面等参与，相当于服务。Springboot项目







#gmall-usermanage 项目中 pom.xml

|  |
| --- |
| <**parent**>  <**artifactId**>gmall-parent</**artifactId**>  <**groupId**>com.atguigu.gmall0416</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  <**relativePath**>../gmall-parent/pom.xml</**relativePath**> </**parent**>  <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu.gmall0416</**groupId**>  <**artifactId**>gmall-interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>   <**dependency**>  <**groupId**>com.atguigu.gmall0416</**groupId**>  <**artifactId**>gmall-service-util</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> |

### 代码开发

### 5.1 bean

|  |
| --- |
| **public class** UserInfo **implements** Serializable{  @Id  @Column  **private** String **id**;  @Column  **private** String **loginName**;  @Column  **private** String **nickName**;  @Column  **private** String **passwd**;  @Column  **private** String **name**;  @Column  **private** String **phoneNum**;  @Column  **private** String **email**;  @Column  **private** String **headImg**;  @Column  **private** String **userLevel**;  } |

注意：@Column 和@Id 都是javax.persistence包中的

**技巧 idea快捷键：alt+insert可以快速插入getter 和setter**

### 5.2 Mapper

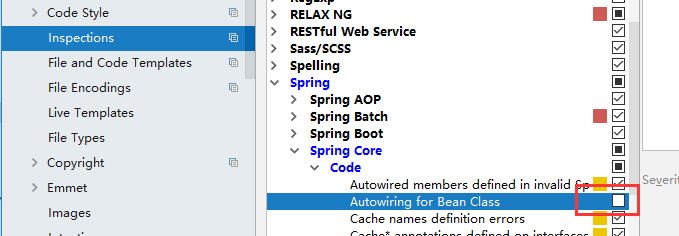
通常跟实体类+Mapper

|  |
| --- |
| **public interface** UserInfoMapper **extends** Mapper<UserInfo> { } |

注意： Mapper也是引用tk.mybatis.mapper.common.Mapper包中的

Idea 有的时候校验@Autowired不准 可以把校验关闭

settings -> Inspections -> spring->spring core -> code-> Autowiring for Bean class



### 5.3 service

Service接口—将其放到gmall-interface  
package com.atguigu.gmall.service

|  |
| --- |
| **public interface** UserInfoService {  List<UserInfo> getUserInfoList(); } |

### 5.4 ServiceImpl

在gmall-usermanage中添加service.impl

|  |
| --- |
| @Service **public class** UserInfoServiceImpl **implements** UserInfoService {   @Autowired  **private** UserInfoMapper **userInfoMapper**;  @Override  **public** List<UserInfo> getUserInfoList() {  **return userInfoMapper**.selectAll();  } } |

### 5.5 Controller

|  |
| --- |
| @Controller **public class** UserInfoController {   @Autowired  **private** UserInfoService **userInfoService**;   @RequestMapping(**"findAll"**)  @ResponseBody  **public** List<UserInfo> findAll(){  List<UserInfo> userInfoList = **userInfoService**.getUserInfoList();  **return** userInfoList;  } } |

### 5.6 测试

|  |
| --- |
| **import tk.mybatis.spring.annotation**.MapperScan;  @SpringBootApplication @MapperScan(basePackages = **"com.atguigu.gmall.usermanage.mapper"**) **public class** GmallOrderServiceApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(GmallOrderServiceApplication.**class**, args);  } } |

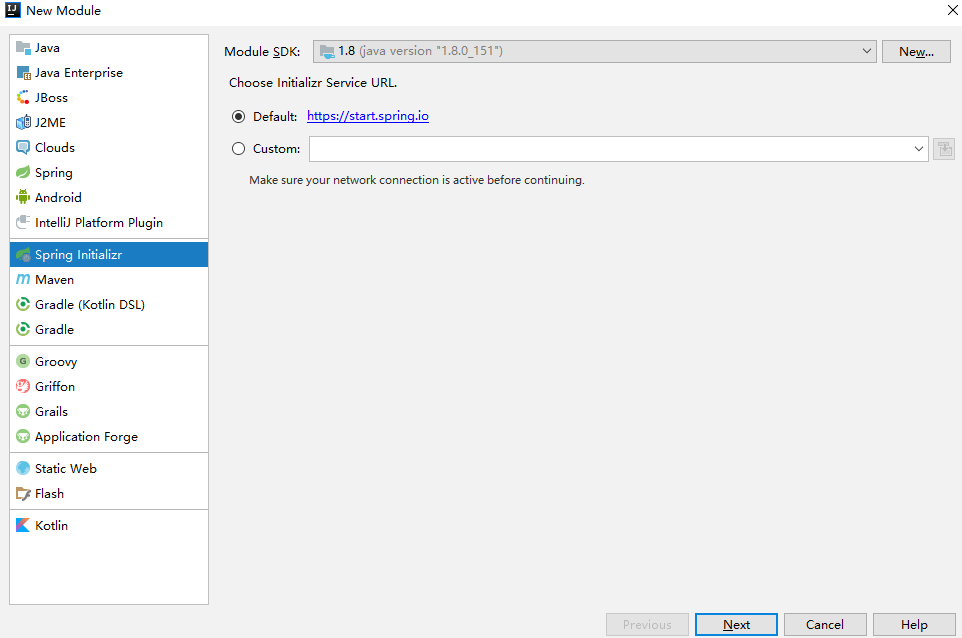
5.7 application.properties

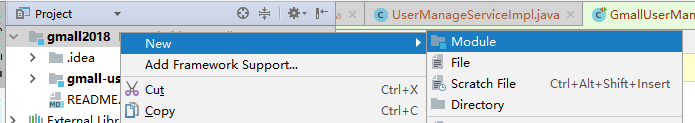
|  |
| --- |
| **server.port**=**8080 spring.datasource.url**=**jdbc:mysql://localhost:3306/gmall?characterEncoding=UTF-8 spring.datasource.username**=**root spring.datasource.password**=**root** |

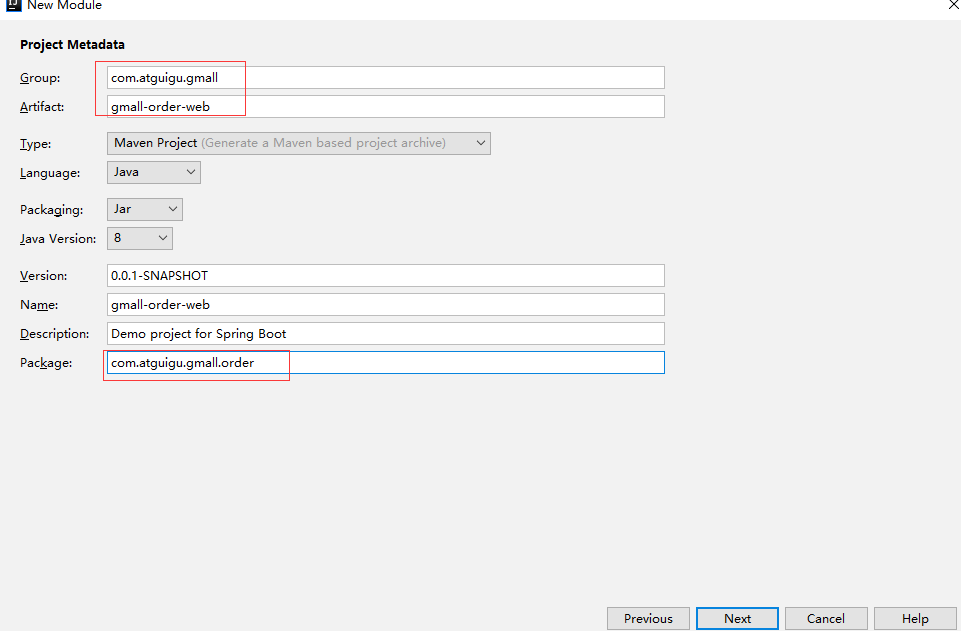
### 6 **实现订单的Web应用（Controller）调用用户的Service应用的用户地址信息功能**

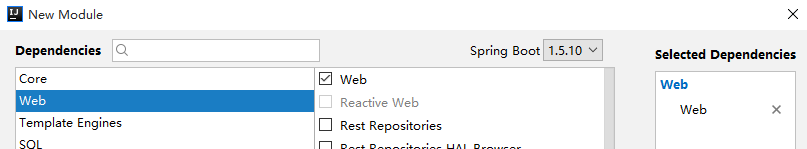
用户与订单有关系么？通过用户id查找订单信息[用户地址]

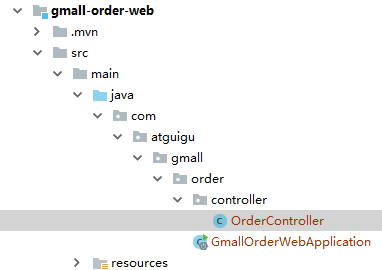
#### 6.1 搭建订单的gmall-order-web模块工程











# application.properties

|  |
| --- |
| **server.port**=**8081** |

需要开发的类

|  |  |  |
| --- | --- | --- |
| 包 | 类 | 说明 |
| controller | OrderController | web controller |

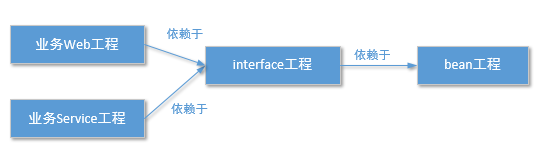
由于需要让订单的web应用可以调用用户的Service接口，那么必须在订单的工程中也要包含一份Service接口。

如果拷贝一个接口到订单工程中，那么如果以后有更多的模块都调用这个接口呢？每个都拷贝一份接口类么？

这种情况我们就可以利用maven的依赖把这些接口作为公共的包管理起来。

同时接口类种的方法也引用了很多的实体bean, 那么同样的实体bean的类我们也统一管理起来。

这样我们就有了如下的依赖关系：



Pom.xml 依赖关系

|  |
| --- |
| <**parent**>  <**groupId**>com.atguigu.gmall</**groupId**>  <**artifactId**>gmall-parent</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**> </**parent**>  <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu.gmall</**groupId**>  <**artifactId**>gmall-interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>   <**dependency**>  <**groupId**>com.atguigu.gmall</**groupId**>  <**artifactId**>gmall-web-util</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> |

// 用户跟订单，订单中是不是要有用户地址。根据用户id 查找用户的地址！

Select \* from user\_address where user\_id = ?

制作控制器

|  |
| --- |
| @RequestMapping(**"trade"**) **public** String trade(){  *// 返回一个视图名称叫index.html* **return "index"**; } |
| 页面需要进行松校验  在web-util 中添加  *<!--松校验-->* <**dependency**>  <**groupId**>net.sourceforge.nekohtml</**groupId**>  <**artifactId**>nekohtml</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>xml-apis</**groupId**>  <**artifactId**>xml-apis</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>org.apache.xmlgraphics</**groupId**>  <**artifactId**>batik-ext</**artifactId**> </**dependency**> |
| 在application.properties中做热部署  **spring.thymeleaf.cache**=**false spring.thymeleaf.mode**=**LEGACYHTML5** |

#### 6.2 用户地址信息查询

需要开发的类

bean

|  |
| --- |
| **public class** UserAddress **implements** Serializable{  @Column  @Id  **private** String **id**;  @Column  **private** String **userAddress**;  @Column  **private** String **userId**;  @Column  **private** String **consignee**;  @Column  **private** String **phoneNum**;  @Column  **private** String **isDefault**;  } |

**在gmall-usermanage 项目中添加UserAddressMapper**

|  |
| --- |
| **public interface** UserAddressMapper **extends** Mapper<UserAddress> { } |

**Gmall-interface**

|  |
| --- |
| **public** List<UserAddress> getUserAddressList(String userId); |

UserInfoServiceImpl中增加方法

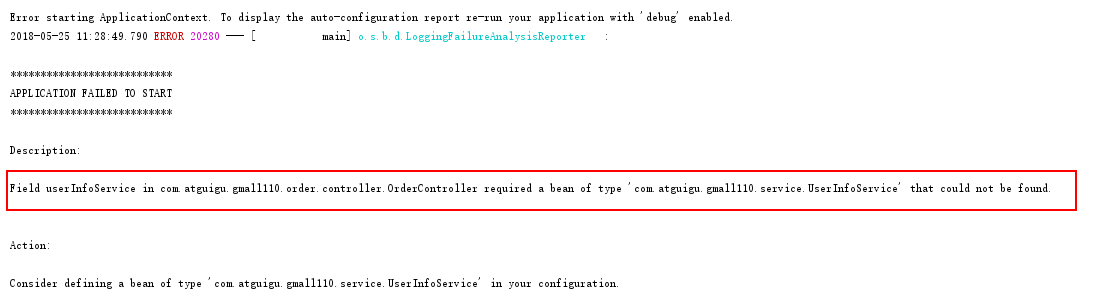
|  |
| --- |
|  |
| @Override **public** List<UserAddress> getUserAddressList(String userId) {  *// 调用mapper  // select \* from userAddress where userId=?* UserAddress userAddress = **new** UserAddress();  userAddress.setUserId(userId);  **return userAddressMapper**.select(userAddress); } |

控制器 OrderController

|  |
| --- |
| @RestController **public class** OrderController {   @Autowired  **private** UserInfoService **userInfoService**;  @RequestMapping(**"trade"**)  @ResponseBody *// 第一个返回json 字符串，fastJson.jar 第二直接将数据显示到页面！*  **public** List<UserAddress> trade(String userId){  *// 返回一个视图名称叫index.html* **return userService**.getUserAddressList(userId);  }  } |

启动gmall-order-web 项目，需要改一下端口号，在配置文件中 8089

结果图：



|  |
| --- |
| Field userService in com.atguigu.gmall0218.order.controller.OrderController required a bean of type 'com.atguigu.gmall0218.service.UserService' that could not be found.  因为：userService 在8080 服务器中 |

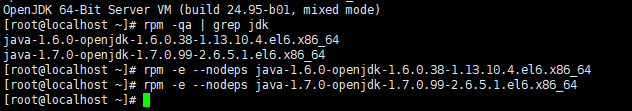
**重申：先搭建一个空的！git。搭建完前四步，停一会！回味一下。看看下一步是否还跟前四步一样！如果搭建错了！删除。**

# 三、Dubbo和zookeeper

Dubbo 正常运行必须依赖zookeeper！zookeeper 什么语言[java]

配置jdk 步骤：注意：jdk-1.8

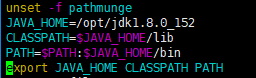
rpm -qa | grep jdk



[root@localhost opt]# tar -zxvf jdk-8u152-linux-x64.tar.gz

# vim /etc/profile

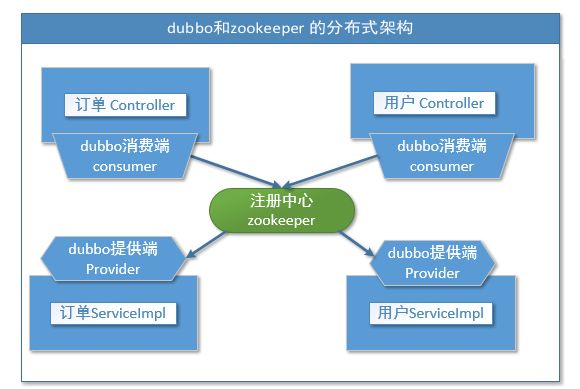
在最后添加如下配置



# source /etc/profile

使配置文件生效。永久生效则必须要reboot！

# java -version



那dubbo和zookeeper如何引入？

dubbo其实是一组jar包，通过maven引入就可以。

zookeeper是一个开源的服务软件，需要安装到linux中。

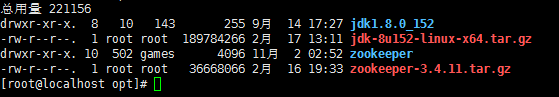
### 1 安装zookeeper

#### 1.1安装环境：

linux版本: CentOS 7.0

zookeeper版本 zookeeper-3.4.11.tar.gz

拷贝zookeeper-3.4.11.tar.gz到/opt下，并解压缩



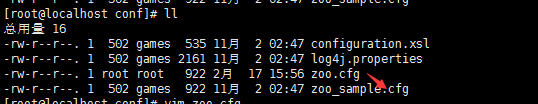
改名叫zookeeper



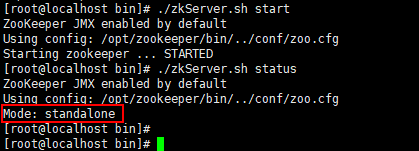
#### 1.2初始化zookeeper配置文件

拷贝/opt/zookeeper/conf/zoo\_sample.cfg

到同一个目录下改个名字叫zoo.cfg



然后咱们启动zookeeper



以上状态即为安装成功。

默认的端口号是2181!

### 2 dubbo的监控中心使用

dubbo本身并不是一个服务软件。它其实就是一个jar包能够帮你的java程序连接到zookeeper，并利用zookeeper消费、提供服务。所以你不用在Linux上启动什么dubbo服务。

但是为了让用户更好的管理监控众多的dubbo服务，官方提供了一个可视化的监控程序，不过这个监控即使不装也不影响使用。

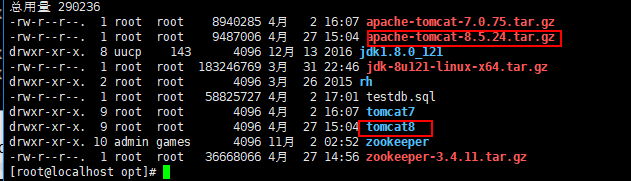
#### 2.1安装监控软件：

配置： tomcat8 + dubbo-admin.war + jdk1.8

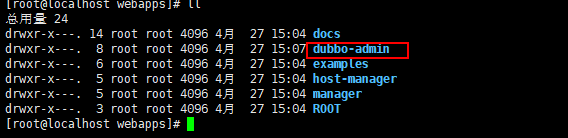
扩展：**tomcat7+dubbo-admin.2.5.3.war+jdk1.7**

dubbo-admin.war 是一个动态web项目

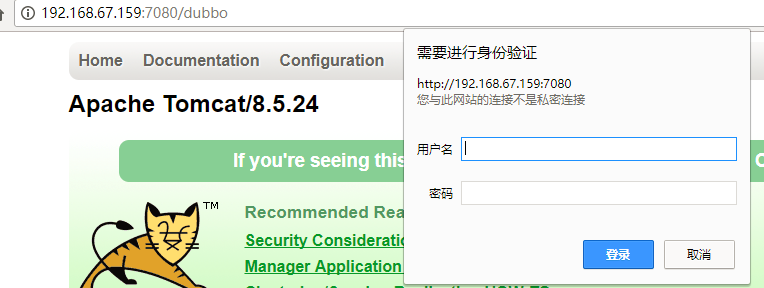
拷贝tomcat8和dubbo-admin.war到/opt目录下



然后把dubbo-admin-2.6.0.war拷贝到tomcat的webapps目录下



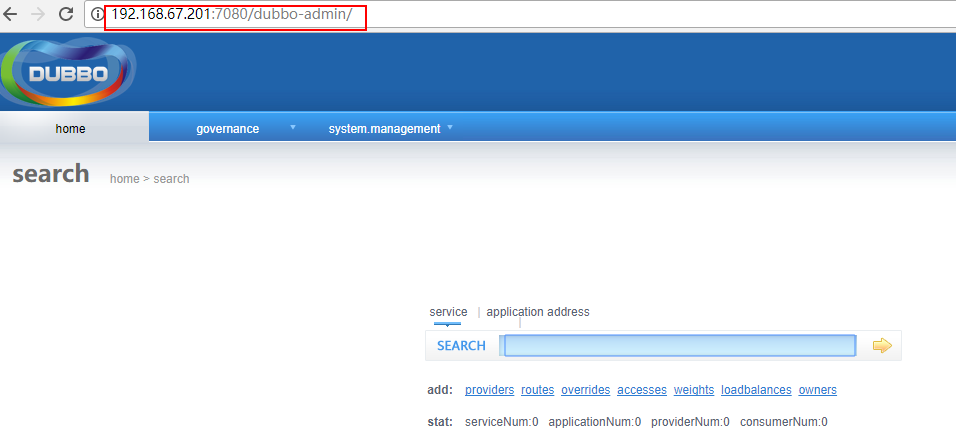
#### 2.2启动后用浏览器访问



可以看到要提示用户名密码，默认是root/root

Dubbo.properties.配置文件中写明了，密码是什么以及注册中心的地址！

(修改的话，可以去)



打开这个界面就说明，dubbo的监控服务已经启动。但是现在咱们还没有搭建dubbo的提供端和消费端。

### 3 开发功能

通过用户的id 查找用户的地址信息

#### 3.1 引入dubbo的依赖

*spring-boot-starter-dubbo*

*dubbo*

*zkclient*

这个依赖首先要放到gmall-parent工程中，用来定义要引入的三个包是什么版本。

|  |
| --- |
| *<?*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">  <modelVersion>4.0.0</modelVersion>   <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-parent</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>pom</packaging>  <properties>  <fastjson.version>1.2.46</fastjson.version>  <dubbo-starter.version>1.0.10</dubbo-starter.version>  <dubbo.version>2.6.0</dubbo.version>  <zkclient.version>0.10</zkclient.version>  </properties>   <dependencyManagement>  <dependencies>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>${fastjson.version}</version>  </dependency>  <dependency>  <groupId>tk.mybatis</groupId>  <artifactId>mapper</artifactId>  <version>${mapper.version}</version>  </dependency>  *<dependency>*  *<groupId>com.alibaba</groupId>*  *<artifactId>dubbo</artifactId>*  *<version>${dubbo.version}</version>*  *</dependency>*  *<dependency>*  *<groupId>com.101tec</groupId>*  *<artifactId>zkclient</artifactId>*  *<version>${zkclient.version}</version>*  *</dependency>*  *<dependency>*  *<groupId>com.gitee.reger</groupId>*  *<artifactId>spring-boot-starter-dubbo</artifactId>*  *<version>${dubbo-starter.version}</version>*  *</dependency>*   </dependencies>  </dependencyManagement> </project> |

然后加入到gmall-common-util模块中

|  |
| --- |
| <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**> </**dependency**>  <**dependency**>  <**groupId**>com.101tec</**groupId**>  <**artifactId**>zkclient</**artifactId**>  <**exclusions**>  <**exclusion**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-log4j12</**artifactId**>  </**exclusion**>  </**exclusions**> </**dependency**>  <**dependency**>  <**groupId**>com.gitee.reger</**groupId**>  <**artifactId**>spring-boot-starter-dubbo</**artifactId**> </**dependency**> |

这样在所有的业务模块中都可以使用dubbo了。

#### 3.2 如何使用：

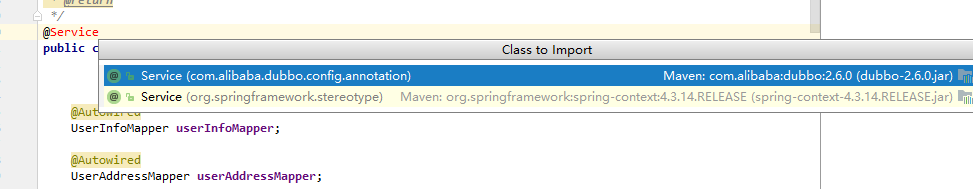
dubbo的使用分为**提供端**和**消费端**。使用起来非常方便只要记住两个注解@Reference和@Service，加上application.properties的一段配置就可以了。

#### 3.3 提供端

顾名思义就是提供服务供别人调用的，相当于spring中的Service的实现类。

使用也很简单，就是一个注解加一份配置

提供端在实现类上增加注解 @Service，和spring的是一样的但是引的包是不一样的。如下



在UserInfoServiceImpl实现类上重新引包，这次引入com.alibaba.dubbo.config.annotation这个包。

**在gmall-user-manage项目中为配置文件application.properties中增加**

|  |
| --- |
| **spring.dubbo.application.name**=**usermanage spring.dubbo.registry.protocol**=**zookeeper spring.dubbo.registry.address**=**192.168.67.201:2181 spring.dubbo.base-package**=**com.atguigu.gmall spring.dubbo.protocol.name**=**dubbo** |

其中：

application.name就是服务名，不能跟别的dubbo提供端重复

registry.protocol 是指定注册中心协议

registry.address 是注册中心的地址加端口号

base-package 注解方式要扫描的包

protocol.name 协议名称dubbo

注意：防火墙要关闭，或者是放行2181端口号

#### 3.4 消费端

gmall-order-web模块 **application.properties配置**

|  |
| --- |
| **spring.dubbo.application.name=order-web spring.dubbo.registry.protocol**=**zookeeper spring.dubbo.registry.address**=**192.168.67.201:2181 spring.dubbo.base-package**=**com.atguigu.gmall spring.dubbo.protocol.name**=**dubbo spring.dubbo.consumer.timeout**=**10000 spring.dubbo.consumer.check**=**false** |

consumer.timeout 是访问提供端服务的超时时间，默认是1000毫秒

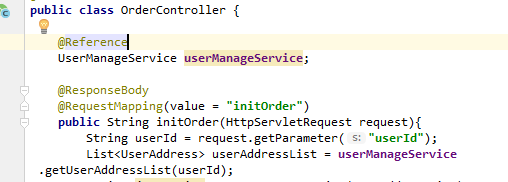
consumer.check 是启动消费端时，是否检查服务端能否正常访问。如果选择true，那启动消费端时，必须保证提供端服务正常，否则接口无法注入。

**消费端代码**

使用起来也比较简单，只要把原来@Autowired改成@Reference就可以 注意引用的包是

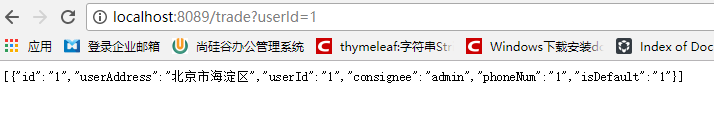
|  |
| --- |
| com.alibaba.dubbo.config.annotation.Reference |

不要引用错了



#### 3.5 启动测试

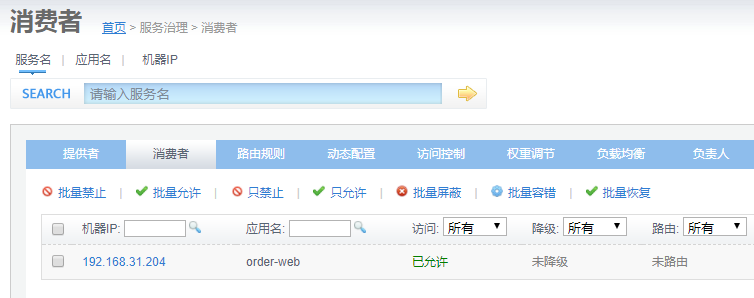
然后访问



说明controller可以通过dubbo调用不同模块的service

我们也可以通过dubbo-admin进行观察：

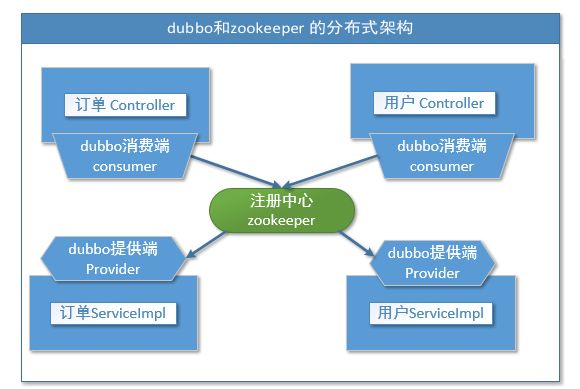
消费端



提供端



那么我们的分布式就可以基于这种方式实现不同模块间的调用。每一个实现服务的消费端和提供端分离。



#### 3.6 设置zookeeper开机自启动 【选做】

1. 进入目录
   1. cd /etc/rc.d/init.d
2. 创建zookeeper文件
   1. vim zookeeper
3. 编辑zookeeper 文件

|  |
| --- |
| #!/bin/bash  #chkconfig: 2345 10 90  #description: service zookeeper  export JAVA\_HOME=/opt/jdk1.8.0\_152  export ZOO\_LOG\_DIR=/opt/zookeeper/log  ZOOKEEPER\_HOME=/opt/zookeeper  su root ${ZOOKEEPER\_HOME}/bin/zkServer.sh "$1" |

1. 授予可执行权限
   1. chmod +x /etc/rc.d/init.d/zookeeper
2. 添加到服务中,并查看是否添加成功
   1. [root@localhost init.d]# chkconfig --add zookeeper
   2. chkconfig --list
3. 测试重启虚拟机
   1. #reboot

# 四、搭建后台页面

1. 拷贝资料中的前段项目页面，放入一个没有中文目录的文件下
2. 在gmall-admin当前目录下cmd 回车
   1. npm install
   2. npm run dev
   3. 直接访问浏览器
3. 讲两个配置文件
   1. dev.env.js 【<http://192.168.1.73:8082> 】 改为<http://localhost:8082>
   2. index.js host: '192.168.1.73' 改为 host: 'localhost'