谷粒商城

版本：V 2.0

# 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。



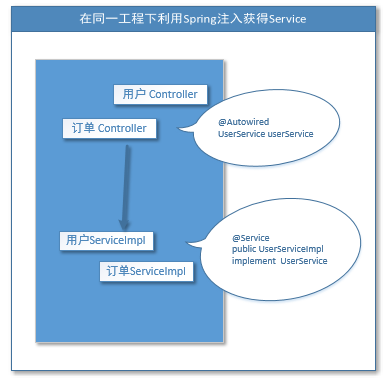
### 1.4 面向服务的分布式架构

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。

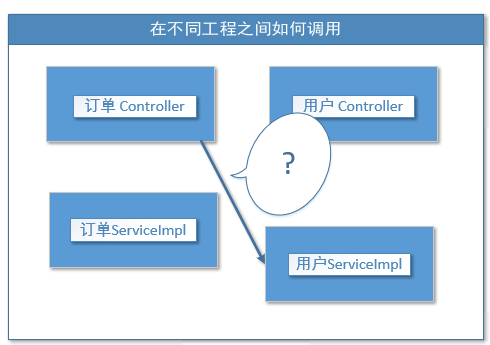


## 2 如何实现这种SOA架构

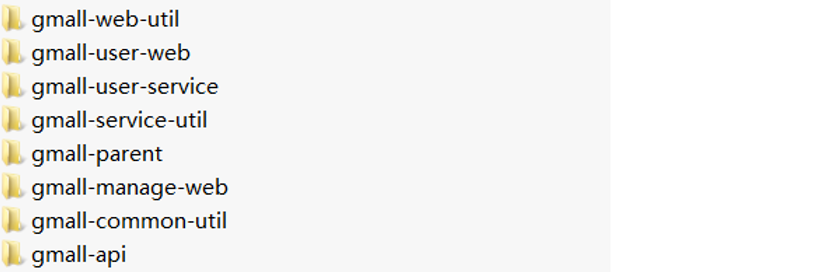
原来所有的controller、service接口、service实现都在一个工程，通过Spring的ioc就可以实现互相调用。



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



# 二、分布式工程的模块搭建



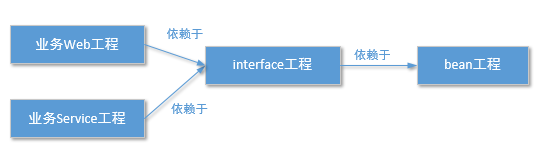
由于需要让订单的web应用可以调用用户的Service接口，那么必须在订单的工程中也要包含一份Service接口。

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？

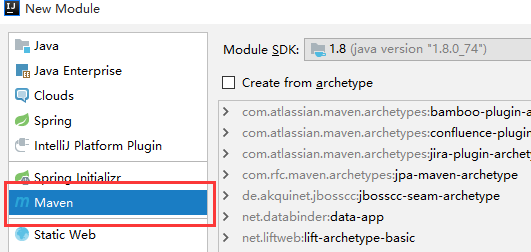
这种情况我们就可以利用maven的依赖把这些接口作为公共的包管理起来。

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。

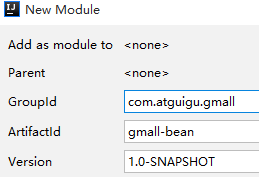
这样我们就有了如下的依赖关系：

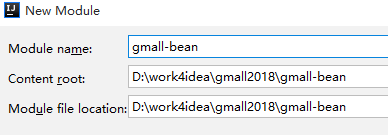


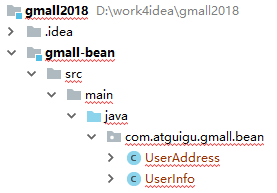
### 1 创建bean模块(与interface一起整合为api模块)



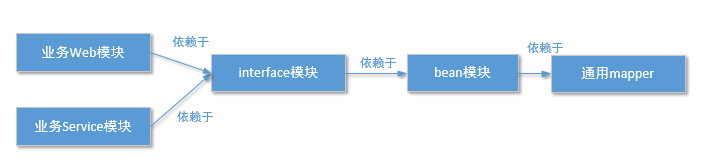
同时我们把UserManage中的bean剪切到bean模块中







bean模块报错是因为其中引用了通用mapper，所以我们把通用mapper的依赖提取出来放到bean模块后面，变成如下结构。



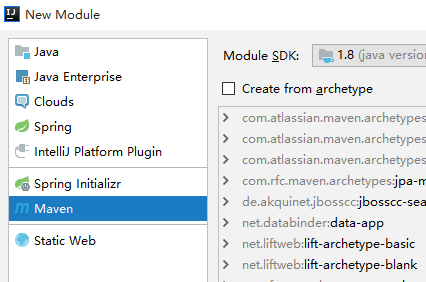
bean模块的pom.xml

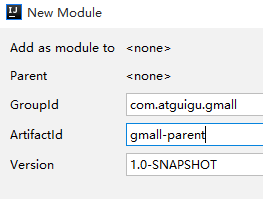
|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <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>  <version>1.2.3</version>  <exclusions>  <exclusion>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId>  </exclusion>  </exclusions>  </dependency>  </dependencies> </project> |

由于依赖包分布于多个模块中，最好有一个地方能够把所有依赖的版本通用管理起来。

这就用到了maven的<parent>概念。可以让所有的模块都继承这个parent模块，由这个parent模块来管理版本。

### 2 搭建parent模块

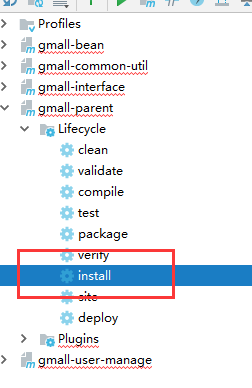




parent模块的 pom.xml

|  |
| --- |
| *<?*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



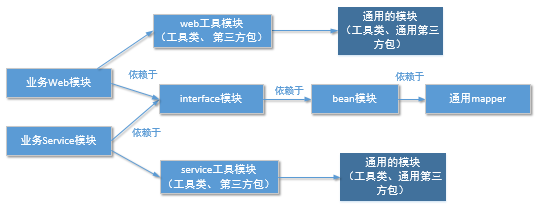
那么除了通用mapper以外其他的第三方依赖我们如何放置

### 3 搭建util模块

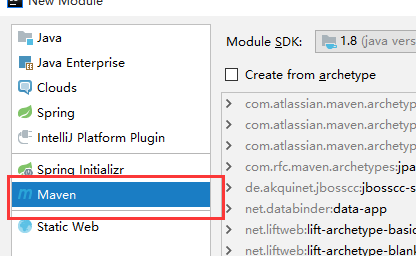
首先我们可以把所有的第三方依赖包分为三种

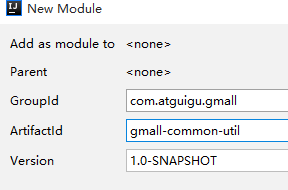
|  |
| --- |
| 1. web业务模块用到的第三方包,比如文件上传客户端、页面渲染工具、操作cookie的工具类等等。 2. service业务模块用到的第三方包，比如jdbc、mybatis、jedis、activemq工具包等等。 3. 通用型的第三方包，比如fastjson、httpclient、apache工具包等等。 4. 只有本模块用到的 es |

基于这三种情况我们可以搭建如下的依赖结构：



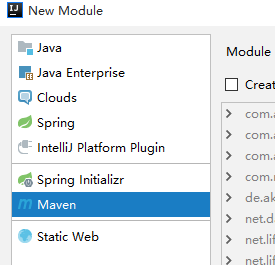
创建common-util的模块

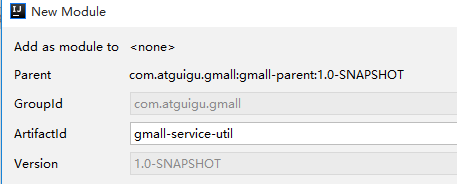




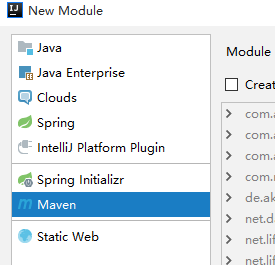
创建gmall-web-util和gmall-service-util

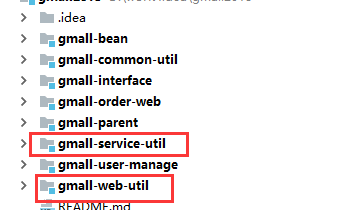
创建service-util模块





创建web-util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 |
| httpclient | restful调用客户端 |

gmall-web-util

|  |  |
| --- | --- |
| Thymeleaf/jsp/freemarker | springboot自带页面渲染工具(springboot有默认版本号) |

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-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>    </dependencies> </project> |

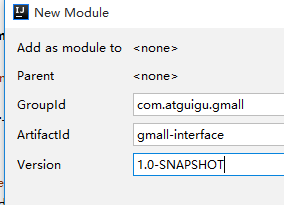
gmall-web-util的pom.xml

|  |
| --- |
| *<?*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> |

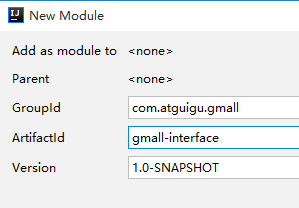
gmall-service-util的pom.xml

|  |
| --- |
| *<?*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> |

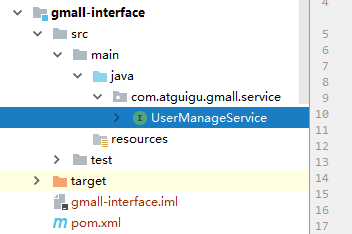
创建interface模块



### 4 搭建interface模块(与bean一起整合为api模块)



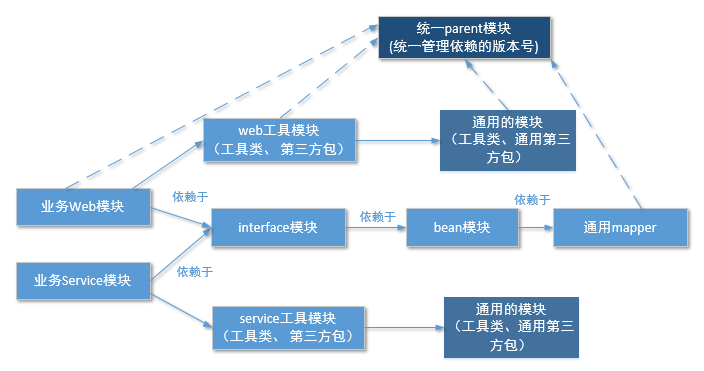
把UserManageService接口移动到该模块下



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>   <dependencies>  <dependency>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-bean</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

最终的结构图



### 5 gmall-User-manage模块

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-user-manage</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>   <name>gmall-user-manage</name>  <description>Demo project for Spring Boot</description>   <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-service-util</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build> </project> |

其他的类，要重新引一下包

|  |
| --- |
| 1 同时要修改bean的引入  2 同时要修改@Service和@Autowrited注解  3 将接口和bean转移到bean和interface项目中  4 原来user-manage中的mapper，service接口，和实现类中的引用bean类的位置需要修改  import com.atguigu.gmall.usermanage.bean.UserInfo;  import com.atguigu.gmall.bean.UserInfo;  5 重新安装maven |

### 6 继续开发gmall-XXX-web模块

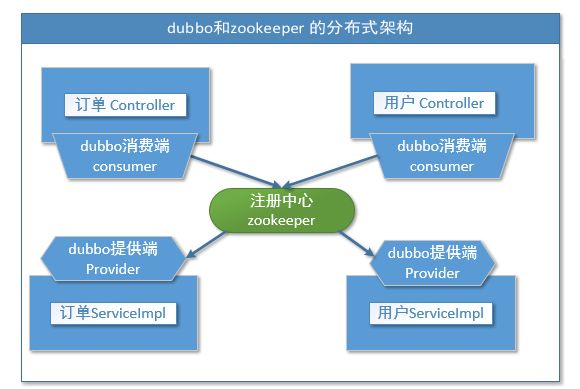
xxx-web的pom.xml

|  |
| --- |
| *<?*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-order-web</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>   <name>gmall-order-web</name>  <description>Demo project for Spring Boot</description>   <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-api</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>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>   </project> |

# 三、Dubbo和zookeeper

dubbo和springcloud的区别

|  |
| --- |
| Dubbo出生于阿里系，是阿里巴巴服务化治理的核心框架，并被广泛应用于中国各互联网公司；只需要通过spring配置的方式即可完成服务化，对于应用无入侵。但框架本身的成熟度以及文档的完善程度，完全能满足各大互联网公司的业务需求。如果我们需要使用配置中心、分布式跟踪这些内容都需要自己去集成，这样无形中增加了使用 Dubbo 的难度。  Spring Cloud 是大名鼎鼎的 Spring 家族的产品， 专注于企业级开源框架的研发。 Spring Cloud 自从发展到现在，仍然在不断的高速发展，几乎考虑了服务治理的方方面面，开发起来非常的便利和简单。    Spring Cloud更新的非常快，企业需要根据自身的研发水平和所处阶段选择合适的架构来解决业务问题，不管是Dubbo还是Spring Cloud都是实现微服务有效的工具。 |



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 dubbo的基本概念

调用关系：

**服务提供者（Provider）**：暴露服务的服务提供方，服务提供者在启动时，向注册中心注册自己提供的服务。

**服务消费者（Consumer）**: 调用远程服务的服务消费方，服务消费者在启动时，向注册中心订阅自己所需的服务，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。

**注册中心（Registry）**：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

**监控中心（Monitor）**：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

* 调用关系说明
* 服务容器负责启动，加载，运行服务提供者。
* 服务提供者在启动时，向注册中心注册自己提供的服务。
* 服务消费者在启动时，向注册中心订阅自己所需的服务。
* 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。
* 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。
* 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。

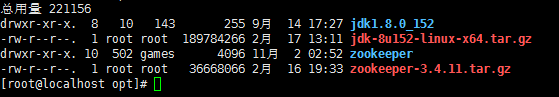
### 2 安装zookeeper

#### 2.1安装环境：

linux版本: CentOS 6.8

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

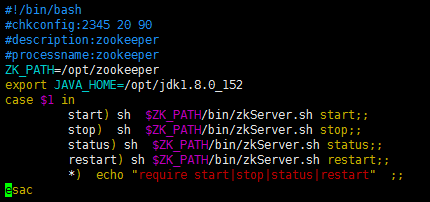


#### 2.2制作开机启动的脚本

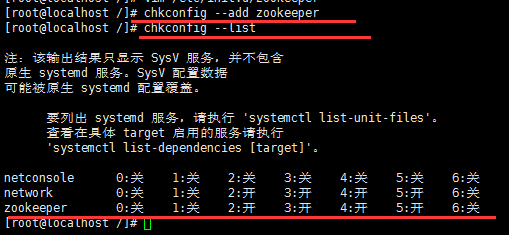


把如下脚本复制进去

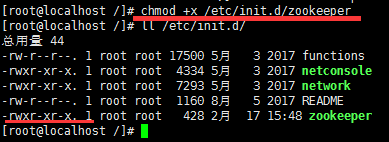
|  |
| --- |
| #!/bin/bash  #chkconfig:2345 20 90  #description:zookeeper  #processname:zookeeper  ZK\_PATH=/opt/zookeeper  export JAVA\_HOME=/opt/jdk1.8.0\_152  case $1 in  start) sh $ZK\_PATH/bin/zkServer.sh start;;  stop) sh $ZK\_PATH/bin/zkServer.sh stop;;  status) sh $ZK\_PATH/bin/zkServer.sh status;;  restart) sh $ZK\_PATH/bin/zkServer.sh restart;;  \*) echo "require start|stop|status|restart" ;;  esac |



然后把脚本注册为Service



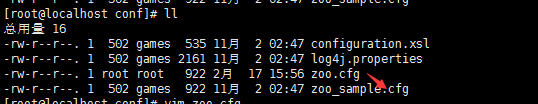
增加权限



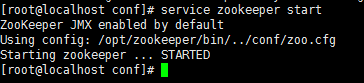
#### 2.3初始化zookeeper配置文件

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。

### 3 dubbo的使用

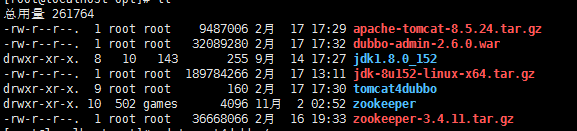
dubbo本身并不是一个服务软件。它其实就是一个jar包能够帮你的java程序连接到zookeeper，并利用zookeeper消费、提供服务。所以你不用在Linux上启动什么dubbo服务。

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。

#### 3.1安装监控软件：

材料： tomcat8 + dubbo-admin

拷贝tomcat8和dubbo-admin到/opt目录下



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



#### 3.2设置开机启动tomcat

|  |
| --- |
| [root@localhost tomcat4dubbo]# vim /etc/init.d/dubbo-admin |

复制如下脚本

|  |
| --- |
| #!/bin/bash  #chkconfig:2345 20 90  #description:dubbo-admin  #processname:dubbo-admin  CATALANA\_HOME=/opt/tomcat4dubbo  export JAVA\_HOME=/opt/jdk1.8.0\_152  case $1 in  start)  echo "Starting Tomcat..."  $CATALANA\_HOME/bin/startup.sh  ;;    stop)  echo "Stopping Tomcat..."  $CATALANA\_HOME/bin/shutdown.sh  ;;    restart)  echo "Stopping Tomcat..."  $CATALANA\_HOME/bin/shutdown.sh  sleep 2  echo  echo "Starting Tomcat..."  $CATALANA\_HOME/bin/startup.sh  ;;  \*)  echo "Usage: tomcat {start|stop|restart}"  ;; esac |

然后同样的注册进入到服务中

|  |
| --- |
| [root@localhost tomcat4dubbo]# chkconfig --add dubbo-admin |

加入权限

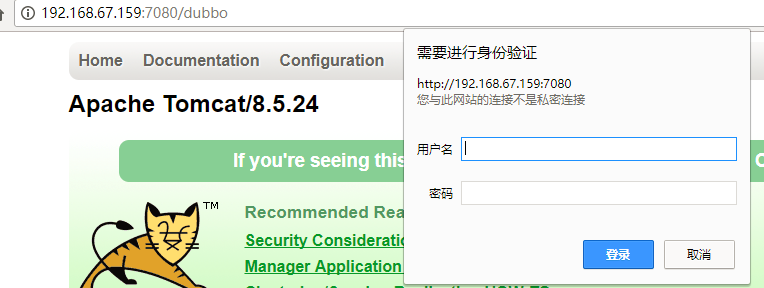
|  |
| --- |
| [root@localhost tomcat4dubbo]# chmod +x dubbo-admin |

如果想改变端口号去tomcat中的server.conf中修改，课件中已改为7080,然后就可以启动服务了。

#### 3.3 启动服务

|  |
| --- |
| [root@localhost tomcat4dubbo]# service dubbo-admin start |

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

(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。

### 4 将dubbo整合到项目中

#### 4.1 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>  <mapper.version>3.4.6</mapper.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。

#### 4.2 如何使用：

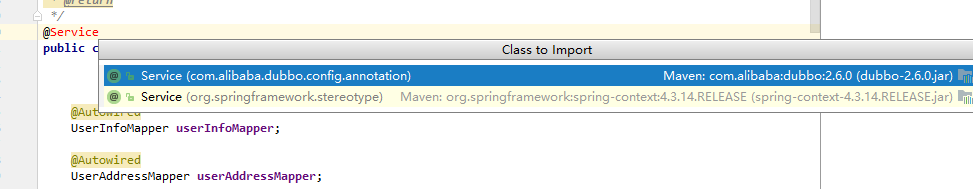
dubbo的使用分为**提供端**和**消费端**。使用起来非常方便只要记住两个注解@Reference和@Service，加上application.properties的一段配置就可以了。

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

**在application.properties中增加**

|  |
| --- |
| **spring.dubbo.application.name**=**usermanage spring.dubbo.registry.protocol**=**zookeeper spring.dubbo.registry.address**=**192.168.222.20:2181 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

protocol.name 是分布式固定是dubbo,不要改。

base-package 注解方式要扫描的包

port 是服务提供端为zookeeper暴露的端口，不能跟别的dubbo提供端重复。

#### 4.4 消费端

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

|  |
| --- |
| **spring.dubbo.application.name=order-web spring.dubbo.registry.protocol**=**zookeeper spring.dubbo.registry.address**=**192.168.222.20:2181 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

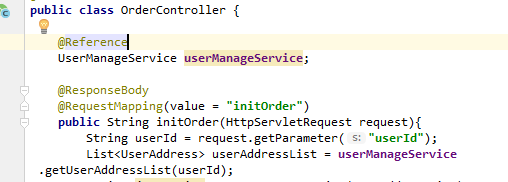
consumer.check 是启动消费端时，是否检查服务端能否正常访问。如果选择true，那启动消费端时，必须保证提供端服务正常，否则接口无法注入。

**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

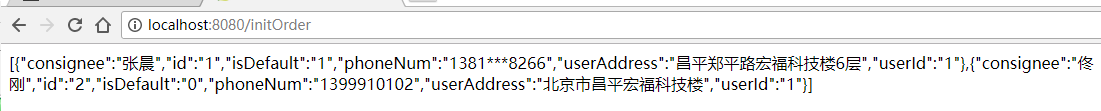


#### 4.5 启动测试

那么这时候就可以测试消费端和服务端了

分别启动order-web模块和usermanage模块

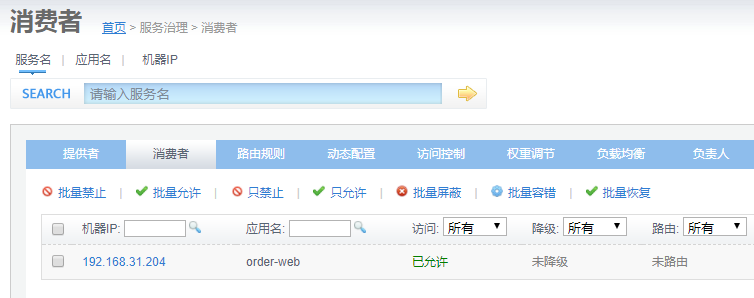
然后访问



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。

