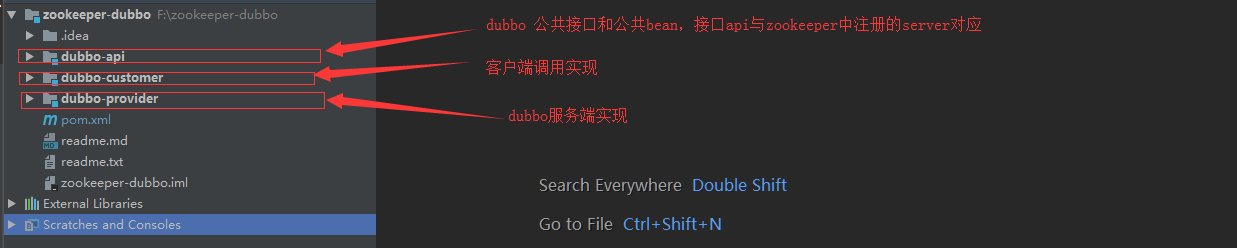
一、项目搭建：

1、创建工程zookeeper-dubbo 删除src和target文件夹，作为多模块的总工程

其pom.xml如下，其他的都删除，其中以dubbo-spi、dubbo-provider和dubbo-customer举例

<?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.insigma.zookeeper.dubbo</groupId>  
 <artifactId>zookeeper-dubbo</artifactId>  
 <packaging>pom</packaging>  
 <version>1.0-SNAPSHOT</version>  
  
 <modules>  
 <module>dubbo-api</module>  
 <module>dubbo-provider</module>  
 <module>dubbo-customer</module>  
 </modules>  
</project>

项目结构

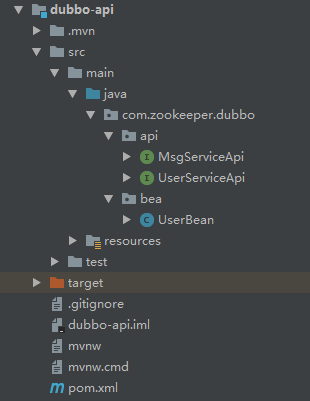


2、创建子模块dubbo-spi作为 dubbo 的公共接口，与zookeeper中注册的server相对应

其pom.xml 主要添加，bean中引入对应注解用，比如@Data

<dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.2</version>  
 <scope>compile</scope>  
</dependency>

模块结构如下： 注意bean 需要序列化 impliments java.io.Serializible



3、服务端模块创建 dubbo-provider

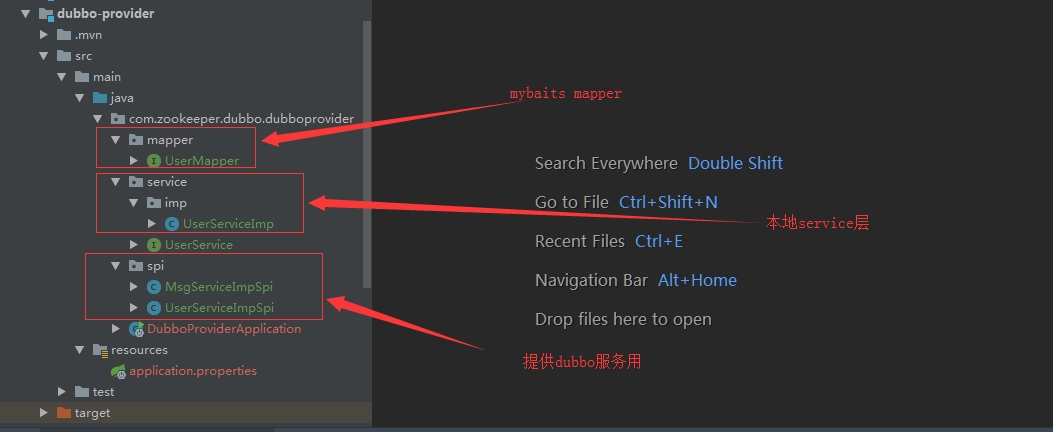
其pom.xml 中添加

<!--引入dubbo-spi模块-->  
<dependency>  
 <groupId>com.springboot.duboo</groupId>  
 <artifactId>dubbo-api</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
</dependency>  
<dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>1.3.2</version>  
</dependency>  
<dependency>  
 <groupId>com.oracle</groupId>  
 <artifactId>ojdbc6</artifactId>  
 <version>11.2.0.3</version>  
</dependency>  
  
<dependency>  
 <groupId>com.alibaba.boot</groupId>  
 <artifactId>dubbo-spring-boot-starter</artifactId>  
 <version>0.2.0</version>  
</dependency>

模块配置文件：application.properties

spring.application.name=dubbo-provider  
server.port=8090  
  
#dubbo  
dubbo.application.name=dubbo-provider  
dubbo.protocol.name=dubbo  
dubbo.protocol.port=20800  
dubbo.registry.address=zookeeper://127.0.0.1:2181  
#dubbo.scan.base-packages=com.zookeeper.dubbo.dubboprovider.spi 用这个没有调用到zookeeper  
dubbo.scan.basePackages=com.zookeeper.dubbo.dubboprovider.spi  
mydubbo.demo.service.version=1.0.0  
  
#datasource  
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver  
spring.datasource.url=jdbc:oracle:thin:@127.0.0.1:1521:orcl  
spring.datasource.username=my\_test  
spring.datasource.password=my\_test

模块结构



UserMapper.java主要代码

@Repository //没有这个注解，idea中会提示红色下划线，但是编译和运行正常，只是看着不爽。  
@Mapper //有这个注解，就不需要在启动类中添加扫描  
public interface UserMapper {  
 @Select("select \* from my\_users where user\_id=#{id}")  
 @Results({//如果没有results结果集配置，查询正常，但是没有内容  
 @Result(id=true,column = "user\_id",property = "userId"),  
 @Result(column = "user\_name",property = "userName"),  
 @Result(column = "user\_pwd",property = "userPwd")  
 })  
 UserBean getUser(Long id);  
  
 @Select("select \* from my\_users")  
 @Results({//如果没有results结果集配置，查询正常，但是没有内容  
 @Result(id=true,column = "user\_id",property = "userId"),  
 @Result(column = "user\_name",property = "userName"),  
 @Result(column = "user\_pwd",property = "userPwd")  
 })  
 List<UserBean>getAll();  
}

UserServiceImp.java 主要代码

*/\*\*  
 \** ***@Service*** *org.springframework.stereotype.Service 是spring 业务层的@Service注解，不是dubbo的，要与dubbo服务的注解区别  
 \* 此处单独有非dubbo服务的service层，所以dubbo服务的都放在了另外的spi包中，且类后都添加Spi,以示区别  
 \*/*@Service  
public class UserServiceImp implements UserService {  
 @Autowired  
 private UserMapper userMapper; //注入mybatis 的借口（UserMapper);  
  
 @Override  
 public UserBean getUser(Long id) {  
 return userMapper.getUser(id);  
 }  
  
 @Override  
 public List<UserBean> getAll() {  
 return userMapper.getAll();  
 }  
}

UserServiceImpSpi.java 主要代码

*/\*\*  
 \** ***@Service*** *com.alibaba.dubbo.config.annotation.Service 是dubbo的@Service注解，而不是spring的业务@Service 注解  
 \* mydubbo.demo.service.version 版本号，在配置文件中配置，在客服端用于识别调用  
 \*/*@Service(version = "${mydubbo.demo.service.version}")  
public class UserServiceImpSpi implements UserServiceApi {  
  
 @Autowired  
 private UserService userService; //注入业务层service（非提供dubbo服务用的）  
  
 @Override  
 public UserBean getUser(Long id) {  
 return userService.getUser(id);  
 }  
  
 @Override  
 public List<UserBean> getAllUser() {  
 return userService.getAll();  
 }  
}

4、客户端模块dubbo-customer

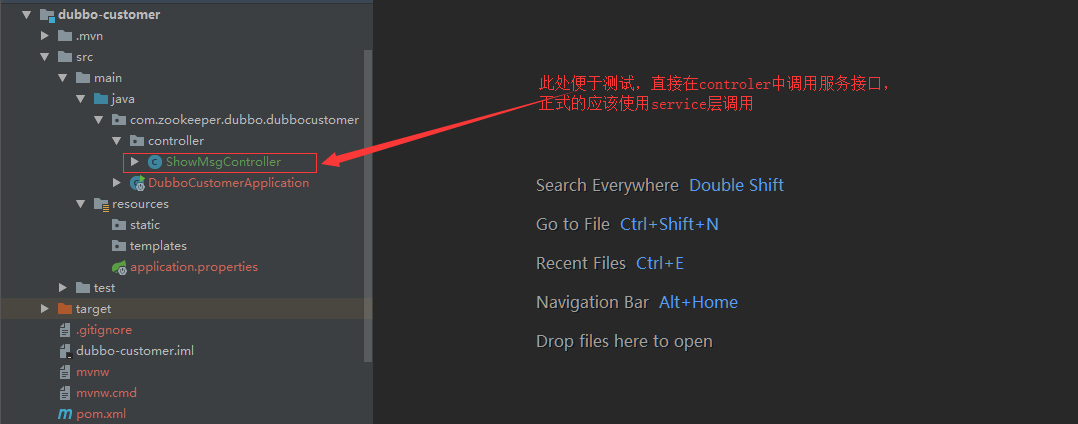
pom.xml中添加

<dependency>  
 <groupId>com.springboot.duboo</groupId>  
 <artifactId>dubbo-api</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
</dependency>  
<dependency>  
 <groupId>com.alibaba.boot</groupId>  
 <artifactId>dubbo-spring-boot-starter</artifactId>  
 <version>0.2.0</version>  
</dependency>

模块配置文件：application.properties

spring.application.name=dubbo-customer  
## 避免和 server 工程端口冲突  
server.port=8099  
  
## Dubbo 服务消费者配置  
dubbo.application.name=dubbo-client  
dubbo.registry.address=zookeeper://127.0.0.1:2181  
dubbo.scan.basePackages=com.zookeeper.dubboclient.controller  
mydubbo.demo.service.version=1.0.0

模块结构



ShowMsgController.java 主要代码

@RestController  
@RequestMapping(value="/msg")  
public class ShowMsgController {  
  
 */\*\*  
 \** ***@Reference*** *com.alibaba.dubbo.config.annotation.Reference 时duboo 的@Reference注解  
 \* 版本是与服务端对应的版本，都是在配置文件中mydubbo.demo.service.version  
 \*/* @Reference(version = "${mydubbo.demo.service.version}")  
 private MsgServiceApi msgServiceApi;  
  
 */\*\*  
 \** ***@Reference*** *com.alibaba.dubbo.config.annotation.Reference 时duboo 的@Reference注解  
 \* 版本是与服务端对应的版本，都是在配置文件中mydubbo.demo.service.version  
 \*/* @Reference(version = "${mydubbo.demo.service.version}")  
 private UserServiceApi userServiceApi;  
  
 @GetMapping(value="/showMsg/{id}/{name}")  
 public String showMsg(@PathVariable("id")Long id,@PathVariable("name")String name){  
 return msgServiceApi.getMsg(name)+">>>>>>>>>>>>>>>>>这是你需要的信息："+userServiceApi.getUser(id);  
 }  
  
  
 @GetMapping(value="/getAll")  
 public List<UserBean> getAllUser(){  
 return userServiceApi.getAllUser();  
 }  
}

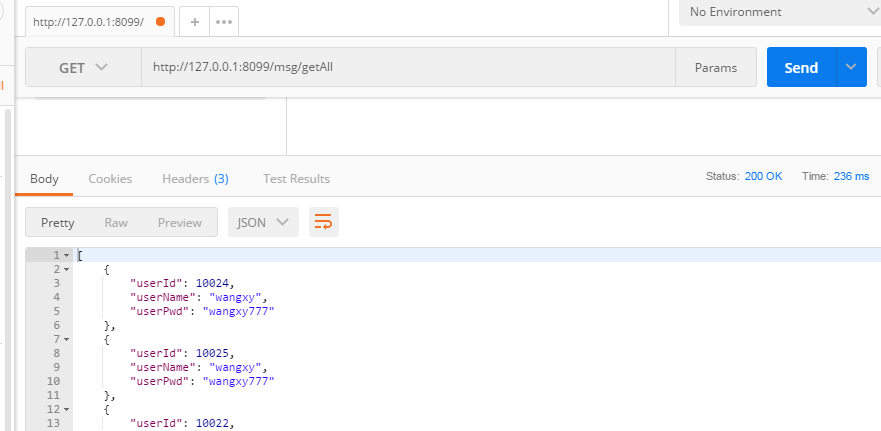
二、测试

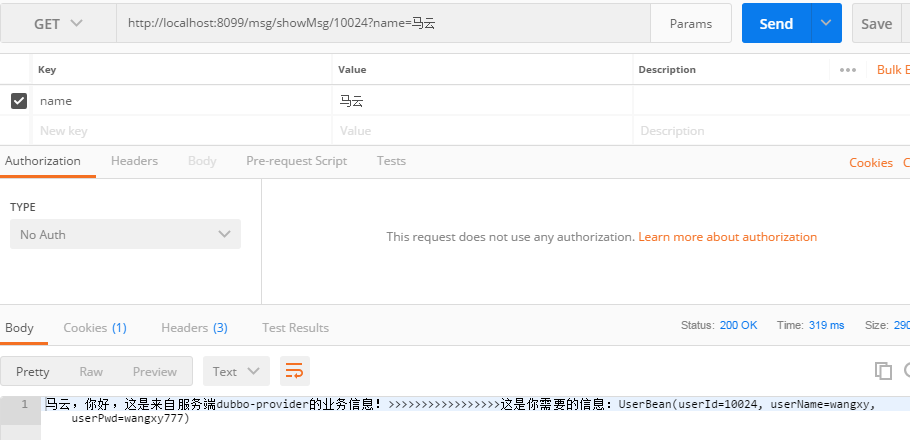
1、启动zookeeper （安装和配置此处不详述，网上很多，很简单）

2、启动dubbo-provider

3、启动dubbo-customer

4、Postman 中访问





5、也可以在dubbo-admin中查看dubbo service

tomcat8中加入dubbo-admin的war包，启动，与访问其他项目一样 （war包包含在项目中，具体的用户名和密码自己可以配置下，此处采用默认的root/root）

具体

**进入webapps目录下，进入webapps\dubbo-admin-2.6.0\WEB-INF目录下，找到dubbo.properties，修改如下：**

dubbo.registry.address=zookeeper://127.0.0.1:2181  
dubbo.admin.root.password=root  
dubbo.admin.guest.password=guest

<http://localhost:8080/dubbo-admin-2.6.0/> 输入用户名和密码

