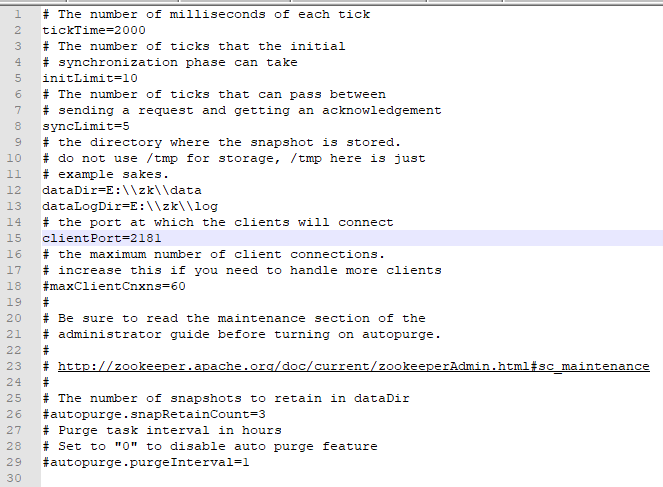
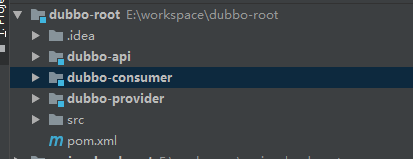
1. **先安装zk**

在官网进行下载：[https://zookeeper.apache.org/releases.html](https://zookeeper.apache.org/releases.html" \t "https://blog.csdn.net/ring300/article/details/_blank)，

安装完之后，解压，将conf目录下的zoo\_sample.cfg文件，复制一份，重命名为zoo.cfg。



1. **直接上代码**
2. 、先看项目框架：



dubbo-api:服务暴露接口

dubbo-consumer:消费端

dubbo-provider:服务实现端

dubbo-root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>dubbo-root</groupId>

<artifactId>dubbo-root</artifactId>

<packaging>pom</packaging>

<version>1.0-SNAPSHOT</version>

<modules>

<module>dubbo-provider</module>

<module>dubbo-consumer</module>

<module>dubbo-api</module>

</modules>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.2.5.RELEASE</version>

</parent>

<properties>

<dubbo.version>2.6.1</dubbo.version>

<fastjson.version>1.2.29</fastjson.version>

</properties>

<dependencies>

<!-- 引入springboot jar，包含了spring的jar -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>2.2.5.RELEASE</version>

</dependency>

<!--引入dubbo所需jar-->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>${dubbo.version}</version>

<exclusions>

<exclusion>

<artifactId>spring</artifactId>

<groupId>org.springframework</groupId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.apache.curator</groupId>

<artifactId>curator-client</artifactId>

<version>4.0.1</version>

</dependency>

<dependency>

<groupId>org.apache.curator</groupId>

<artifactId>curator-framework</artifactId>

<version>4.0.1</version>

</dependency>

<!--json处理的jar-->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>${fastjson.version}</version>

</dependency>

<!--zk所需的jar-->

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

<exclusions>

<exclusion>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</exclusion>

</exclusions>

<version>3.4.8</version>

</dependency>

<dependency>

<groupId>com.101tec</groupId>

<artifactId>zkclient</artifactId>

<exclusions>

<exclusion>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</exclusion>

</exclusions>

<version>0.10</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 编译插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<configuration>

<skip>true</skip>

</configuration>

</plugin>

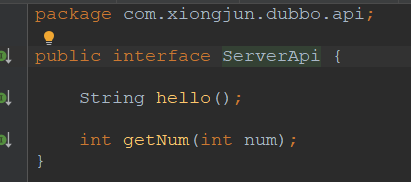
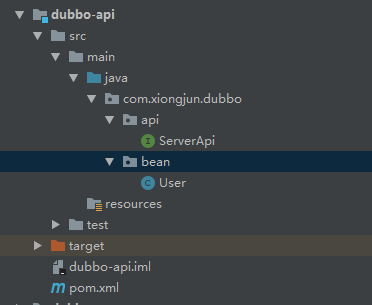
</plugins>

</build>

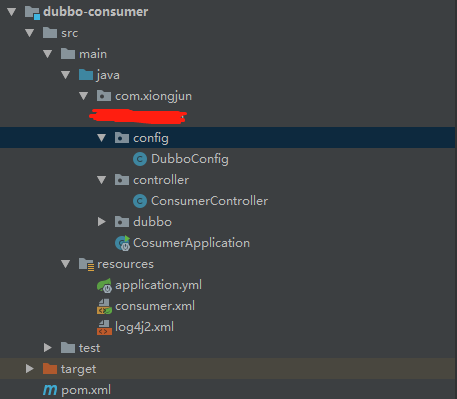
</project>

1. 、查看各个模块：

接口端：

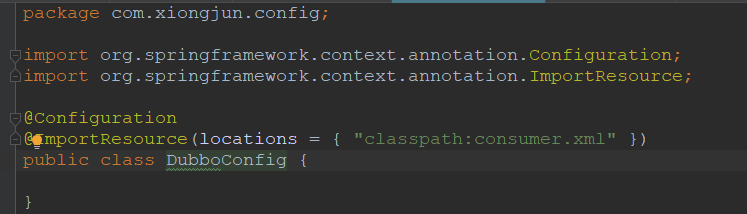


消费端：

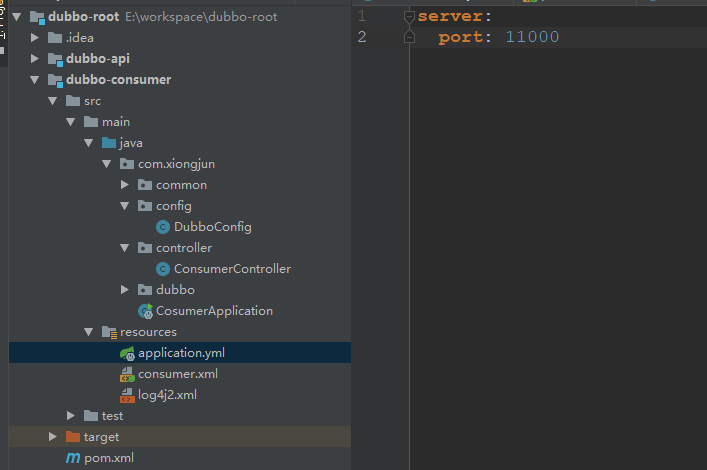


CosumerApplication.java为消费端服务启动类（springboot方式启动）。

配置文件DubboConfig.java，该配置文件是为了启动时加载consumer.xml文件，也可以不配置文件，在CosumerApplication.java启动类里面像服务端启动类一样增加如下注解：@ImportResource("classpath:provider.xml")，详细请看ProviderApplication.java。



消费端和服务端模块的application.yml只是定义了端口



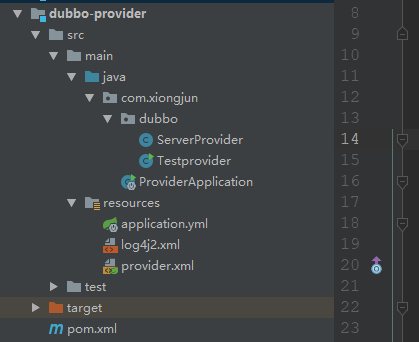
consumer.xml:



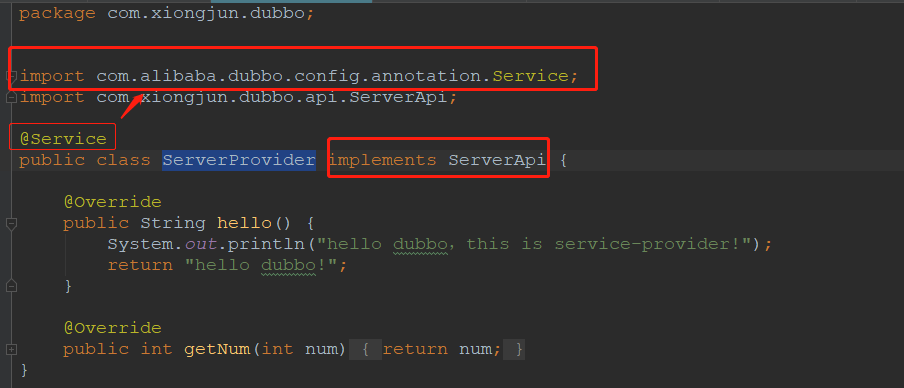
消费端和服务端都要引入api的jar：

<dependency>  
 <groupId>dubbo-root</groupId>  
 <artifactId>dubbo-api</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <scope>compile</scope>  
</dependency>

服务端：

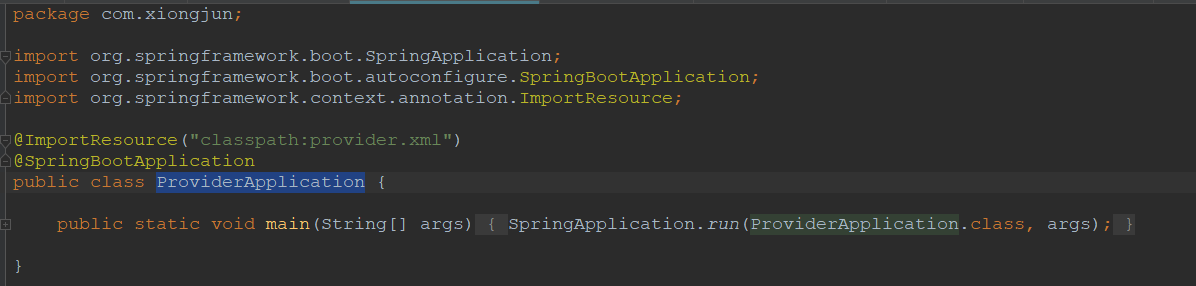


服务实现类ServerProvider.java：



这里引入的@Service必须是dubbo的，否则，消费端调用服务的时候报没有提供服务的错。

服务启动类ProviderApplication.java:



Provider.xml:



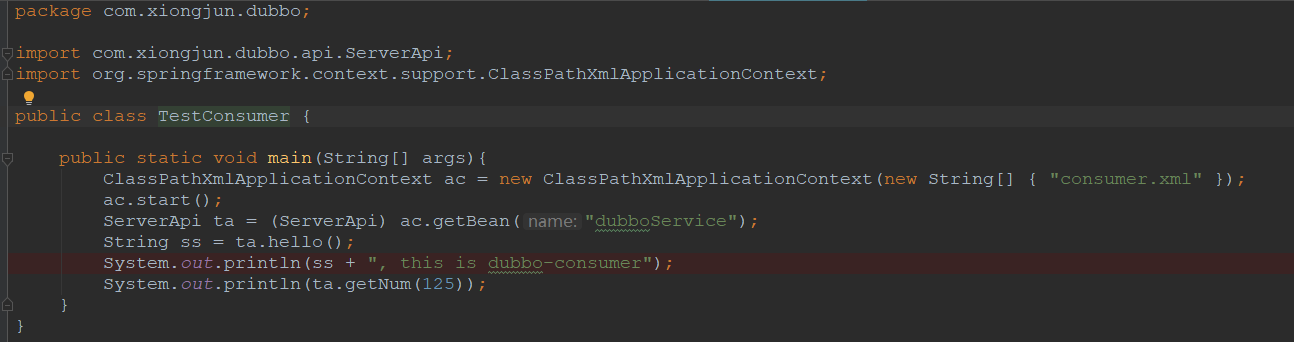
在该项目中有两种方式去测试服务，第一种是在main方法中启动和调用服务，第二种是用springboot的方式启动项目，dubbo服务会自动开启，然后通过在controller里调用服务。

第一种方式：

在服务端开启服务：



在消费端调用服务：



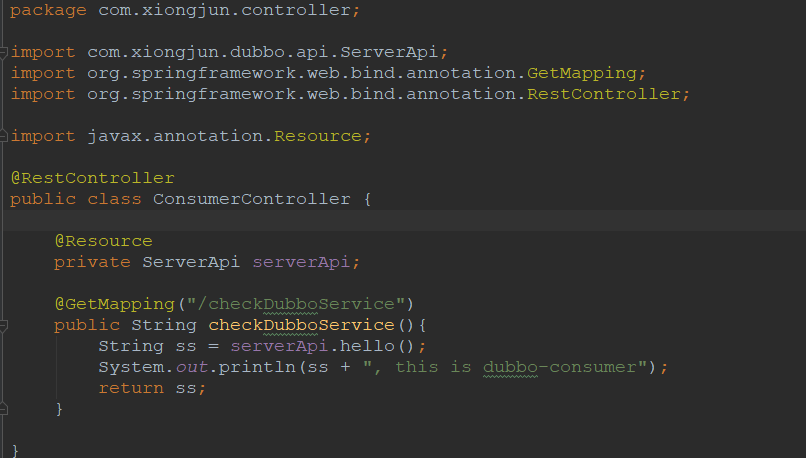
第一种方式不需要DubboConfig.java去加载xml配置。

第二种方式：

需要DubboConfig.java或者在springboot启动类里面加上注解

@ImportResource("classpath:provider.xml")。

在controller中需要用@Resource注解去获取服务接口：



此处，代码就完了。

1. 启动服务：

服务端和消费端都有两种方式启动，

服务端第一种是启动Testprovider.java,第二种是启动ProviderApplication.java

消费端第一种是启动Testconsumer.java,第二种是启动ConsumerApplication.java

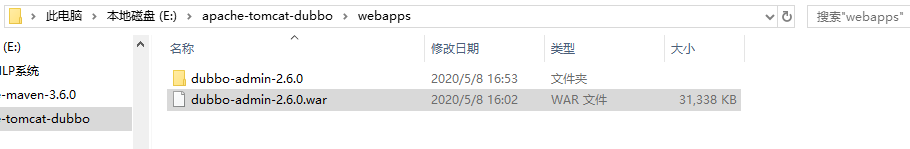
查看是否能够打印结果即可。如果消费端按照第二种方式启动，则可以在浏览器访问：

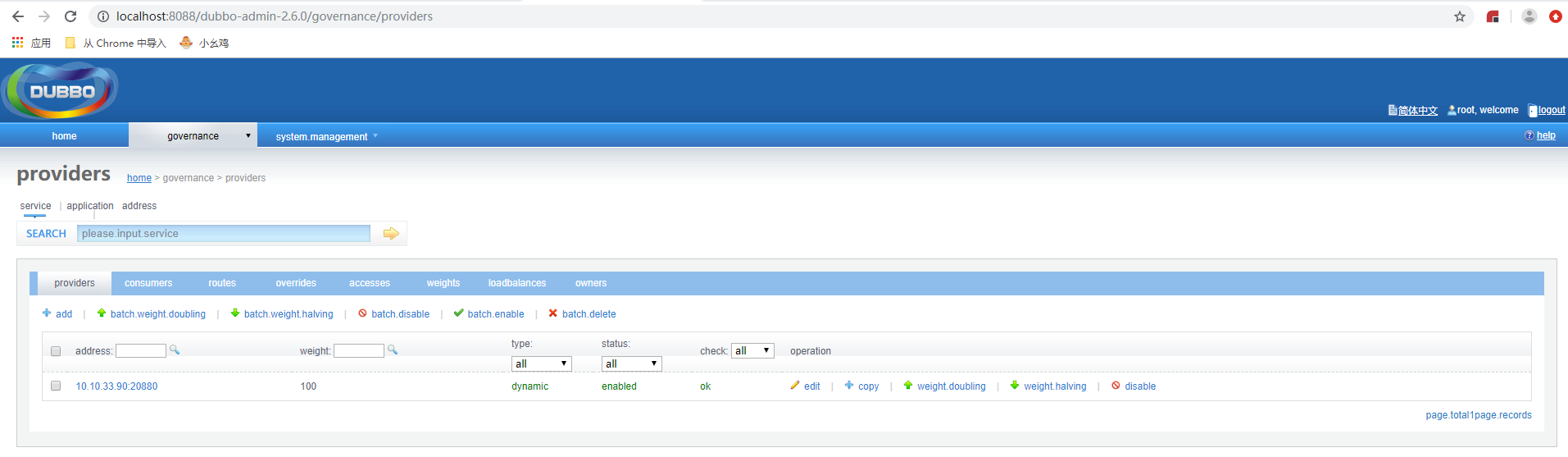
<http://localhost:11000/checkDubboService>，查看是否正常访问。



1. 查看dubbo管理平台：

首先下载dubbo-admin.jar,然后把该jar放到tomcat目录下的webapps文件夹下，然后启动tomcat服务，在浏览器访问<http://localhost:8088/dubbo-admin-2.6.0>即可。

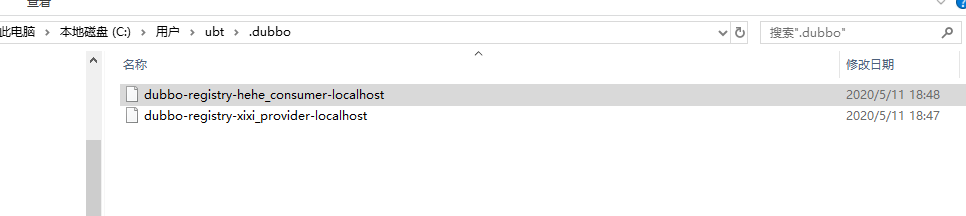




在学习dubbo的过程中，也遇到了一些问题：

Dubbo服务必须要在zk启动成功之后才能注册上去，dubbo服务和zk的启动可以没有先后顺序。

当dubbo服务成功注册到zk上之后，如果zk挂掉了，此时如果消费端第一次调用服务，是不可以调用的，如果消费端之前有调用过dubbo服务，则可以继续调用dubbo服务的。因为，当dubbo成功注册到zk之后，会在本地磁盘生成一个文件，该文件记录了dubbo服务的名称以及ip等信息，当消费者第一次注册到zk成功之后也会在本地磁盘生成一个文件，该文件记录了dubbo服务的名称以及dubbo服务的ip等信息，此时zk'挂掉之后，消费者调用dubbo服务的时候是从本地文件中取获取dubbo的信息的，所以此时可以继续调用dubbo服务，而如果zk挂掉之后，消费端没有注册到zk，是不会在本地磁盘生成文件的，所以就找不到dubbo服务的信息，继而访问不到dubbo服务。



dubbo可以不通过注册中心访问，可以用dubbo直连的方式直接放在，用@Reference(url="")属性进行连接dubbo服务。

如果服务端全部宕机，消费者将无法访问到服务，会无限次重连，等待服务端提供服务。

