使用框架:

Spring+Dubbo

分为 提供者/消费者

provider:

配置:新建文件 : beans-dubbo.xml,并写入, 可以把具体值写在properties中

```
<!-- 配置系统应用名称 -->
   <dubbo:application name="${APP_NAME}" />
   <!-- 通过注册中心发现监控中心服务 -->
   <dubbo:monitor protocol="registry" />
   <!-- 配置注册中心地址 -->
   <dubbo:registry protocol="zookeeper" address="${ZOOKEEPER_ADDRESS}" />

   <!-- 配置服务发布方式（Dubbo 支持多种协议发布，Hessian只是其中的一种,这里用了原生方式） -->
   <dubbo:protocol name="dubbo" port="${APP_PORT}" />
```

application.xml: 写入 以下配置, 作用: 读取配置文件

```
   <bean id="propertyConfigurer"
      class="com.jieshun.jht.framework.config.properties.PropertyPlaceholderConfigurer">

      <property name="locations">
         <list>
            <value>classpath:${profile.properties}/*.properties</value>
         </list>
      </property>
   </bean>
```

作用:使beans-dubbo.xml 文件生效(在beans路径下,这应该看得懂吧,傻子)

```
<!-- hessian调用配置 -->
   <import resource="classpath:beans/beans-*.xml" />
```

文件:

1.写一个接口,写个方法

2.写个实现类,实现这个接口,这个就是你对外提供的方法,实现类上写个注解

@Service("userIntegralManageService"),名字与消费者中配置文件中的订阅服务的id保持一致

别人就能通过接口来调用这个方法,就达到了远程调用

consume:

```
<!-- 配置系统应用名称 -->
    <dubbo:application name="${APP_NAME}" />
    <!-- 通过注册中心发现监控中心服务 -->
    <dubbo:monitor protocol="registry" />
    <!-- 配置注册中心地址 -->
    <dubbo:registry protocol="zookeeper" address="${ZOOKEEPER_ADDRESS}" />

    <!-- 订阅服务 ,这个服务具体指的是提供的一个接口类,这id就是你能用的对象, -->
    <dubbo:reference id="userIntegralManageService"
interface="com.jieshun.jht.account.service.IUserIntegralManageService" />
```

基本就是这样,如果未成功,详情看 ″会员积分系统″ 项目

如果要使用一个公共工程来放在方法,基本思路一样,就是把接口放在公共工程中,在提供者中去实现就行(此时需要将公共工程当成jar包导入,使用pom文件嘛)

使用框架:

springBoot + Dubbo:

consume:

配置:新建文件 : dubbo-consume.xml,并写入, 可以把具体值写在properties中

```
<!-- 提供方应用信息，用于计算依赖关系 -->
    <dubbo:application name="con" />

    <!-- 注册中心暴露服务地址 -->
   <dubbo:registry protocol="zookeeper" address="127.0.0.1:2181" />

    <dubbo:reference id="exampleService"
interface="com.xkj.springboot.service.ExampleService" />
```

文件:

随便写个类,在类上加上注释: (也就是说,写哪个类上都可以,只要写了就行), (这是用了xml的方式来配置,SpringBoot肯定有直接写properties的方式来配置)

```
@Configuration
@PropertySource("classpath:dubbo/dubbo.properties")
@ImportResource({ "classpath:dubbo/*.xml" })
```

可以写个controller,方便验证.

provider:

配置:

```
<!-- 提供方应用信息，用于计算依赖关系 -->
    <dubbo:application name="pro"  />

    <!-- 注册中心暴露服务地址 -->
    <!-- <dubbo:registry address="multicast://224.5.6.7:1234" /> -->

   <!-- <dubbo:registry protocol="zookeeper"
address="10.170.219.98:2181,10.173.55.173:2181" /> -->
   <dubbo:registry protocol="${dubbo.registry.protocol}"
address="${dubbo.registry.address}" />

    <!-- 暴露服务 -->
    <dubbo:protocol name="${dubbo.protocol.name}" port="${dubbo.protocol.port}" />

    <bean id="exampleServiceImpl"
class="com.xkj.springboot.service.impl.ExampleServiceImpl"></bean>

    <dubbo:service interface="com.xkj.springboot.service.ExampleService"
ref="exampleServiceImpl" retries="0" timeout="6000" />
```

文件:

1. 写一个接口, 写个方法

2. 写个实现类, 实现这个接口, 这个就是你对外提供的方法, 实现类上写个注解

@Service("exampleServiceImpl"), 名字与消费者中配置文件中的订阅服务的id保持一致

别人就能通过接口来调用这个方法, 就达到了远程调用

pom.xml:

```
<!-- dubbo -->
      <dependency>
         <groupId>com.alibaba</groupId>
         <artifactId>dubbo</artifactId>
         <exclusions>
            <exclusion>
               <groupId>org.springframework</groupId>
               <artifactId>spring</artifactId>
            </exclusion>
         </exclusions>
         <version>2.5.3</version>
      </dependency>
      <dependency>
         <groupId>org.apache.zookeeper</groupId>
```

```
    <artifactId>zookeeper</artifactId>
    <version>3.4.6</version>
</dependency>
<dependency>
    <groupId>com.github.sgroschupf</groupId>
    <artifactId>zkclient</artifactId>
    <version>0.1</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-configuration-processor</artifactId>
    <optional>true</optional>
</dependency>
```