Day13

互联网项目架构的特点

- 特点
- 用户体验
- 网站的速度

互联网项目架构目标

- 衡量网站的性能指标
- 专业术语

大型互联网项目架构目标

- 什么是集群和分布式

架构演进

- 单体架构
- 垂直架构
- 单体架构存在的问题
- 垂直架构存在的问题
- 分布式架构
- 分布式架构存在的问题
- SOA（Service- Oriented Architecture）
- ESB（Enterparise Servce Bus）
- 微服务架构
  - 特点

网站架构演进图

Dubbo

Dubbo 概念

Dubbo 工作原理

- 工作原理图
- 节点角色说明
- 调用关系说明

Dubbo 入门案例

ZooKeeper 安装

安装步骤：

第一步

第二步

第三步

第四步

第五步

第六步

第七步

使用 Dubbo

互联网项目架构的特点

特点

- 用户多
- 流量大，并发高
- 海量数据
- 易受攻击
- 功能繁琐
- 变更快

用户体验

- 美观
- 功能
- 速度
- 稳定性

网站的速度

- 打开一个新页面一瞬间完成
- 页面内跳转，刹那间完成。

互联网项目架构目标

衡量网站的性能指标

- 响应时间
  - 指执行一个请求从开始到最后收到响应数据所花费的总体时间。
- 并发数
  - 指系统同时能处理的请求数量。
- 并发连接数
  - 指的是客户端向服务器发起请求，并建立了TCP连接。每秒钟服务器连接的总TCP数量。
- 请求数
  - 也称为QPS(Query Per Second)指每秒多少请求。
- 并发用户数
  - 单位时间内有多少用户。
- 吞吐量
  - 指单位时间内系统能处理的请求数量。

专业术语

- QPS: Query Per Second每秒查询数。
- TPS: Transactions Per Second每秒事务数。
- 一个事务是指一个客户端向服务器发送请求然后服务器做出反应的过程。客户机在发送请求时开始计时，收到服务器响应后结束计时，以此来计算使用的时间和完成的事务个数。
- 一个页面的一次访问，只会形成一 个TPS; 但一次页面请求，可能产生多次对服务器的请求，就会有多个QPS。

QPS >= 并发连接数 >= TPS

大型互联网项目架构目标

- 高性能：提供快速的访问体验。
- 高可用：网站服务永远可以正常访问。

什么是集群和分布式

集群和分布式都是多台服务器一起做事情，一起做的事情的方式有所区别。

- 集群
  - 多台服务器一起，干一样的事。

    - 一个业务模块，部署在多台服务器上。
- 分布式
    - 多台服务器一起，干不一样的事。
    - 这些不一样的事， 合起来是一件大事。

架构演进

单体架构

- 优点：简单
    - 开发部署都很方便，小型项目首选
- 缺点
    - 可靠性差

垂直架构

垂直架构是指将单体架构中的多个模块拆分为多个独立的项目。形成多个独立的单体架构。

单体架构存在的问题

- 项目启动慢
- 可靠性差
- 可伸缩性差
- 扩展性和可维护性差
- 性能低

垂直架构存在的问题

重复功能太多

分布式架构

分布式架构是指在垂直架构的基础上，将公共业务模块抽取出来,作为独立的服务**提供**其他调用者**消费**，以实现服务的共享和重用。

底层通过 RPC（RPC 即 Remote Procedure Call 远程过程调用）。

有非常多的协议和技术来都实现了RPC的过程。

比如: HTTP REST风格，Java RMI 规范、WebService SOAP 协议 Hession 等等。

分布式架构存在的问题

- 服务提供方一旦产生变更，所有消费方都需要变更。

SOA（Service– Oriented Architecture）

SOA 即面向服务的架构。

SOA 是一个组件模型，它将应用程序的不同功能单元(称为服务)进行拆分。

拆分之后，这些服务之间通过定义好的接口和契约联系起来。

ESB（Enterparise Servce Bus）

ESB 即企业服务总线，可以看做是一个服务中介。

主要是提供了一个服务与服务之间的交互方式。

ESB 通常包含的功能：

- 负载均衡
- 流量控制
- 加密处理
- 服务的监控
- 异常处理
- 监控告急
- 。。。

微服务架构

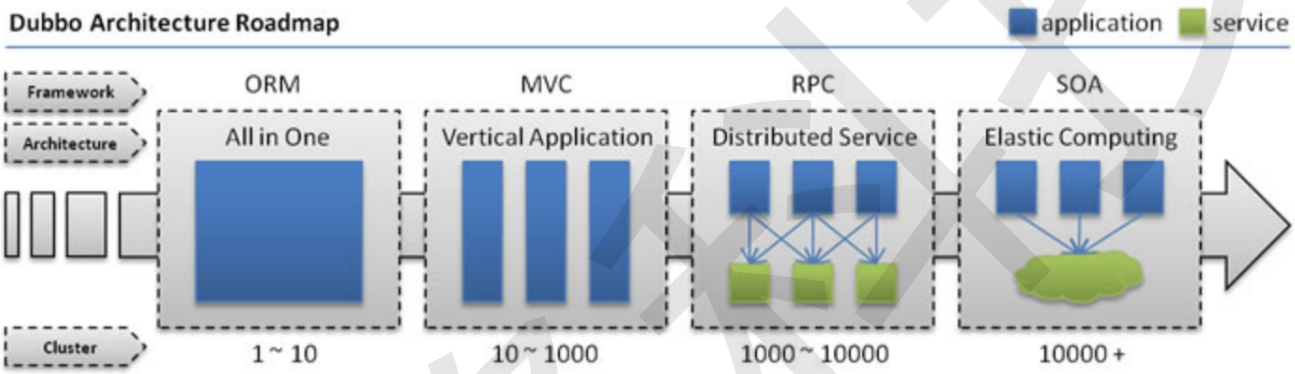
- 微服务架构是在SOA上做的升华，微服务架构强调的一个重点是“业务需要彻底的组件化和服务化”，原有的单个业务系统会拆分为多个可以独立开发、设计、运行的小应用。
- 这些小应用之间通过服务完成交互和集成。

微服务架构 = 80%的SOA服务架构思想 + 100%的组件化架构思想 + 80%的领域建模思想

特点

- 服务实现组件化:开发者可以自由选择开发技术。也不需要协调其他团队
- 服务之间交互一 般使用REST API
- 去中心化:每个微服务有自己私有的数据库持久化业务数据
- 自动化部署:把应用拆分成为一 个–个独立的单个服务,方便自动化部署、测试、运维

网站架构演进图



- 单一应用架构
  - 当网站流量很小时，只需一个应用，将所有功能都部署在一起，以减少部署节点和成本。此时，用于简化增删改查工作量的数据访问框架(ORM)是关键。
- 垂直应用架构
  - 当访问量逐渐增大，单一应用增加机器带来的加速度越来越小，提升效率的方法之一是将应用拆成互不相干的几个应用，以提升效率。此时，用于加速前端页面开发的Web框架(MVC)是关键。
- 分布式服务架构
  - 当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，使前端应用能更快速的响应多变的市场需求。此时，用于提高业务复用及整合的分布式服务框架(RPC)是关键。
- 流动计算架构
  - 当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率。此时，用于提高机器利用率的资源调度和治理中心(SOA)是关键。

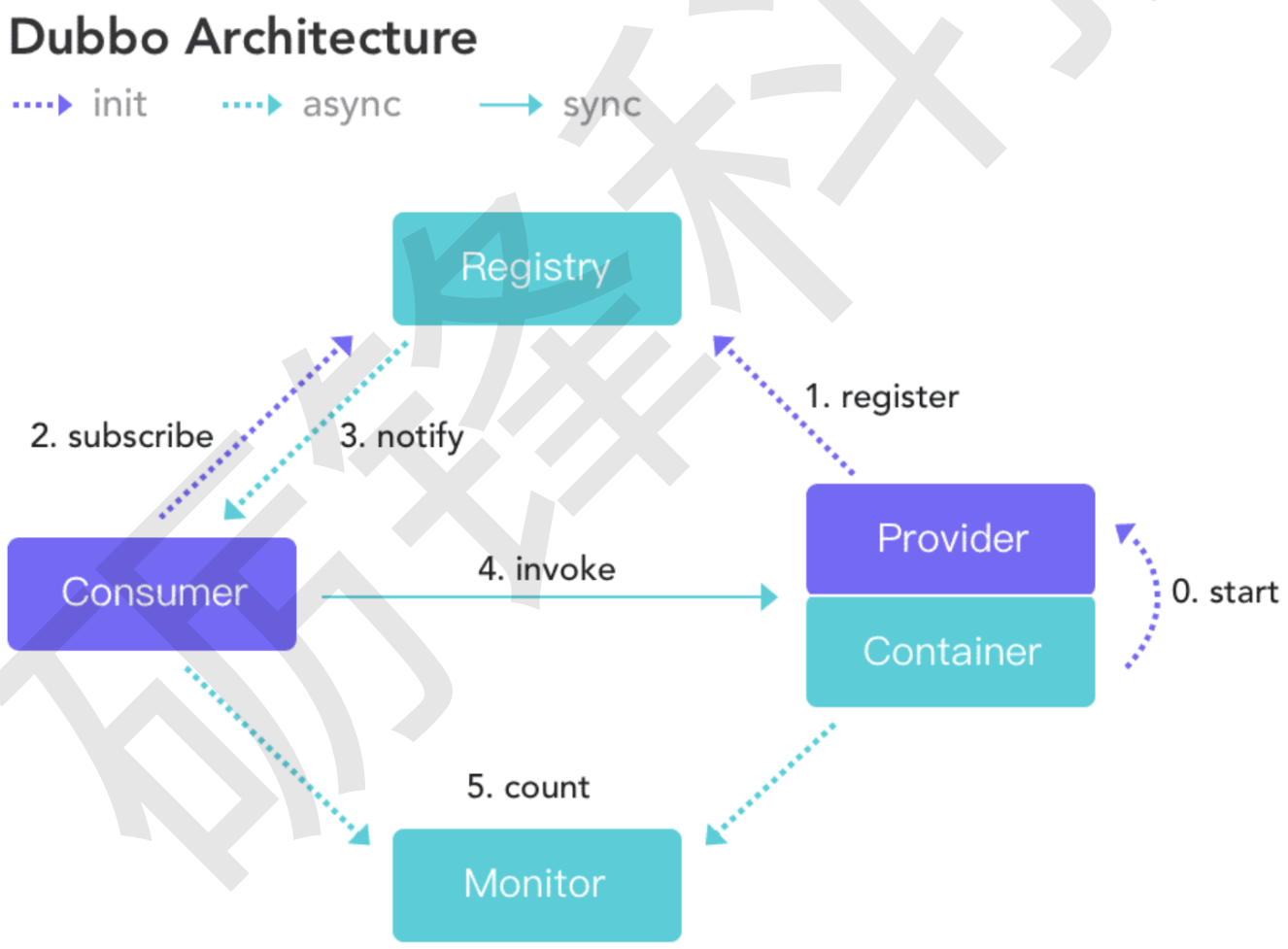
Dubbo

Dubbo 概念

- Dubbo是阿里巴巴公司开源的一个高性能、轻量级的Java RPC框架。
- 致力于提供高性能和透明化的RPC远程服务调用方案,以及 SOA 服务治理方案。
- 官网: https://dubbo.apache.org/

Dubbo 工作原理

工作原理图



节点角色说明

节点	角色说明
Provider	暴露服务的服务提供方
Consumer	调用远程服务的服务消费方
Registry	服务注册与发现的注册中心
Monitor	统计服务的调用次数和调用时间的监控中心
Container	服务运行容器

调用关系说明

1. 服务容器负责启动，加载，运行服务提供者。
2. 服务提供者在启动时，向注册中心注册自己提供的服务。
3. 服务消费者在启动时，向注册中心订阅自己所需的服务。
4. 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。
5. 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。
6. 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。

Dubbo 入门案例

ZooKeeper 安装

安装步骤：

第一步

安装 jdk

第二步

把 zookeeper 的压缩包（zookeeper-3.x.x.tar.gz）上传到 linux 系统

第三步

解压缩压缩包

```shell
tar -zxvf  apacher-zookeeper-3.7.0.tar.gz
mv apacher-zookeeper-3.7.0-bin zookeeper-3.7.0
```

第四步

进入 zookeeper-3.x.x 的目录，创建data目录

```shell
mkdir data
```

```
[parallels@localhost zookeeper-3.7.0]$ mkdir data
mkdir: cannot create directory ‘data’: File exists
[parallels@localhost zookeeper-3.7.0]$ cd data
[parallels@localhost data]$ pwd
/home/parallels/Documents/zookeeper-3.7.0/data
[parallels@localhost data]$
```

第五步

进入conf目录 ， 把 zoo_sample.cfg 改名为zoo.cfg

```shell
cd conf
mv zoo_sample.cfg zoo.cfg
```

第六步

打开zoo.cfg文件，修改data属性：

```shell
dataDir=/root/zookeeper-3.x.x/data
```

```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/home/parallels/Documents/zookeeper-3.7.0/data
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#
#
```

第七步

进入 Zookeeper 的 bin 目录，启动服务命令

```
[parallels@localhost Documents]$ cd zookeeper-3.7.0/
[parallels@localhost zookeeper-3.7.0]$ ls
bin   data  lib          logs        README.md
conf  docs  LICENSE.txt  NOTICE.txt  README_packaging.md
[parallels@localhost zookeeper-3.7.0]$ cd bin
[parallels@localhost bin]$ ls
README.txt    zkEnv.sh                zkSnapshotComparer.sh
zkCleanup.sh  zkServer.cmd            zkSnapShotToolkit.cmd
zkCli.cmd     zkServer-initialize.sh  zkSnapShotToolkit.sh
zkCli.sh      zkServer.sh             zkTxnLogToolkit.cmd
zkEnv.cmd     zkSnapshotComparer.cmd  zkTxnLogToolkit.sh
[parallels@localhost bin]$ ./zkServer.
zkServer.cmd  zkServer.sh
[parallels@localhost bin]$ ./zkServer.
zkServer.cmd  zkServer.sh
[parallels@localhost bin]$ ./zkServer.sh
```

Shell | 复制代码

```
1  cd bin
2  ./zkServer.sh start
```

```
[parallels@localhost zookeeper-3.7.0]$ cd bin
[parallels@localhost bin]$ ./zkServer.sh start
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /home/parallels/Documents/zookeeper-3.7.0/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[parallels@localhost bin]$
```

停止服务命令

Shell 复制代码

```
./zkServer.sh stop
```

```
[parallels@localhost bin]$ ./zkServer.sh stop
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /home/parallels/Documents/zookeeper-3.7.0/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
```

查看服务状态：standalone 单节点

Shell 复制代码

```
./zkServer.sh status
```

```
[parallels@localhost bin]$ ./zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /home/parallels/Documents/zookeeper-3.7.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: standalone
```

```
[parallels@localhost bin]$ ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /home/parallels/Documents/zookeeper-3.7.0/bin/../conf/zoo.cfg
Starting zookeeper ... already running as process 50965.
```

192.168.122.1

使用 Dubbo

实施步骤：

1. 创建服务提供者 Provider
2. 创建服务消费者 Consumer

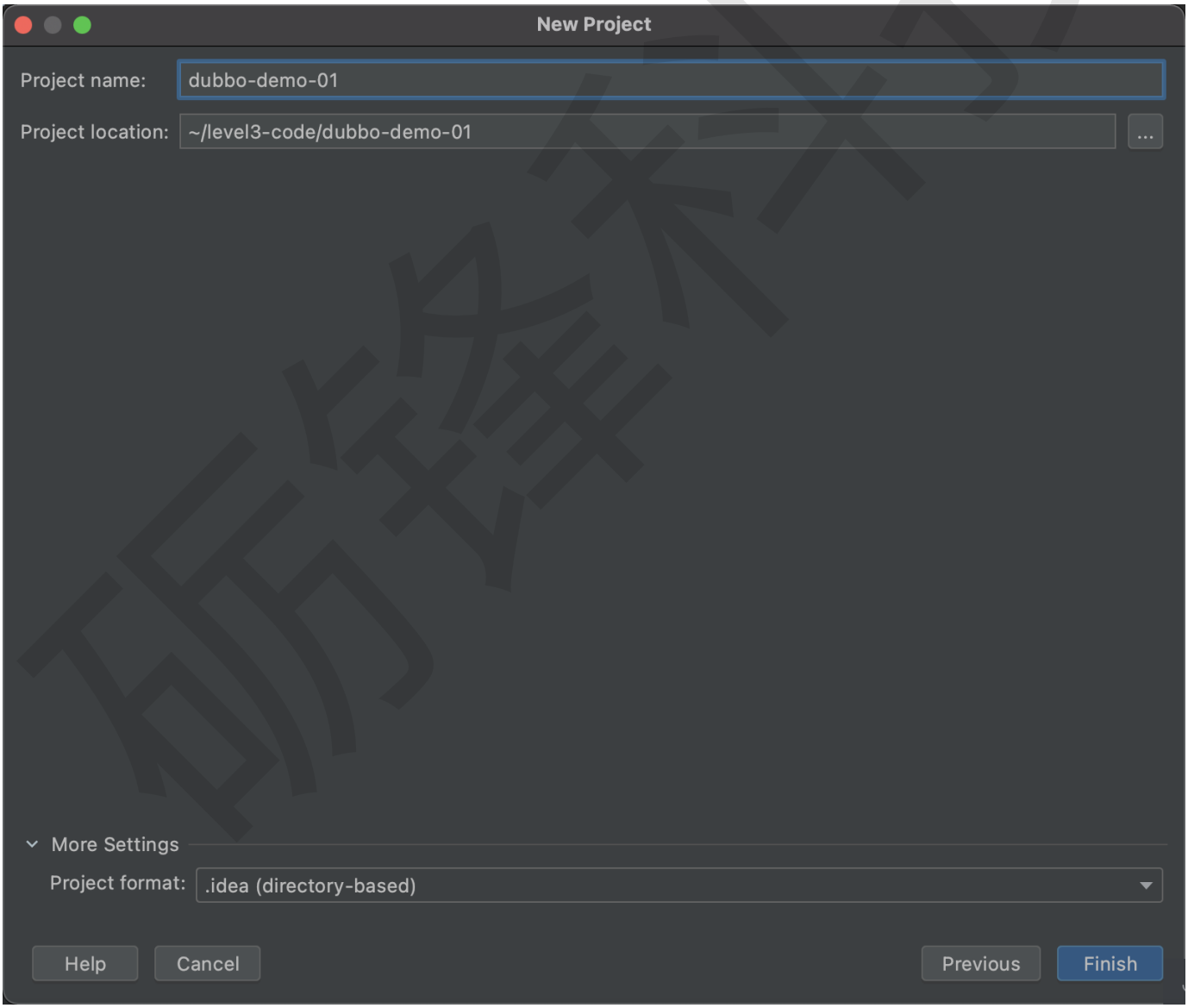
3. 在服务提供者模块编写 UserServiceImpl 提供服务
4. 在服务消费者中的 UserController 远程调用
5. UserServiceImpl 提供的服务
6. 分别启动两个服务，测试

Dubbo作为一个RPC框架，其最核心的功能就是要实现跨网络的远程调用。

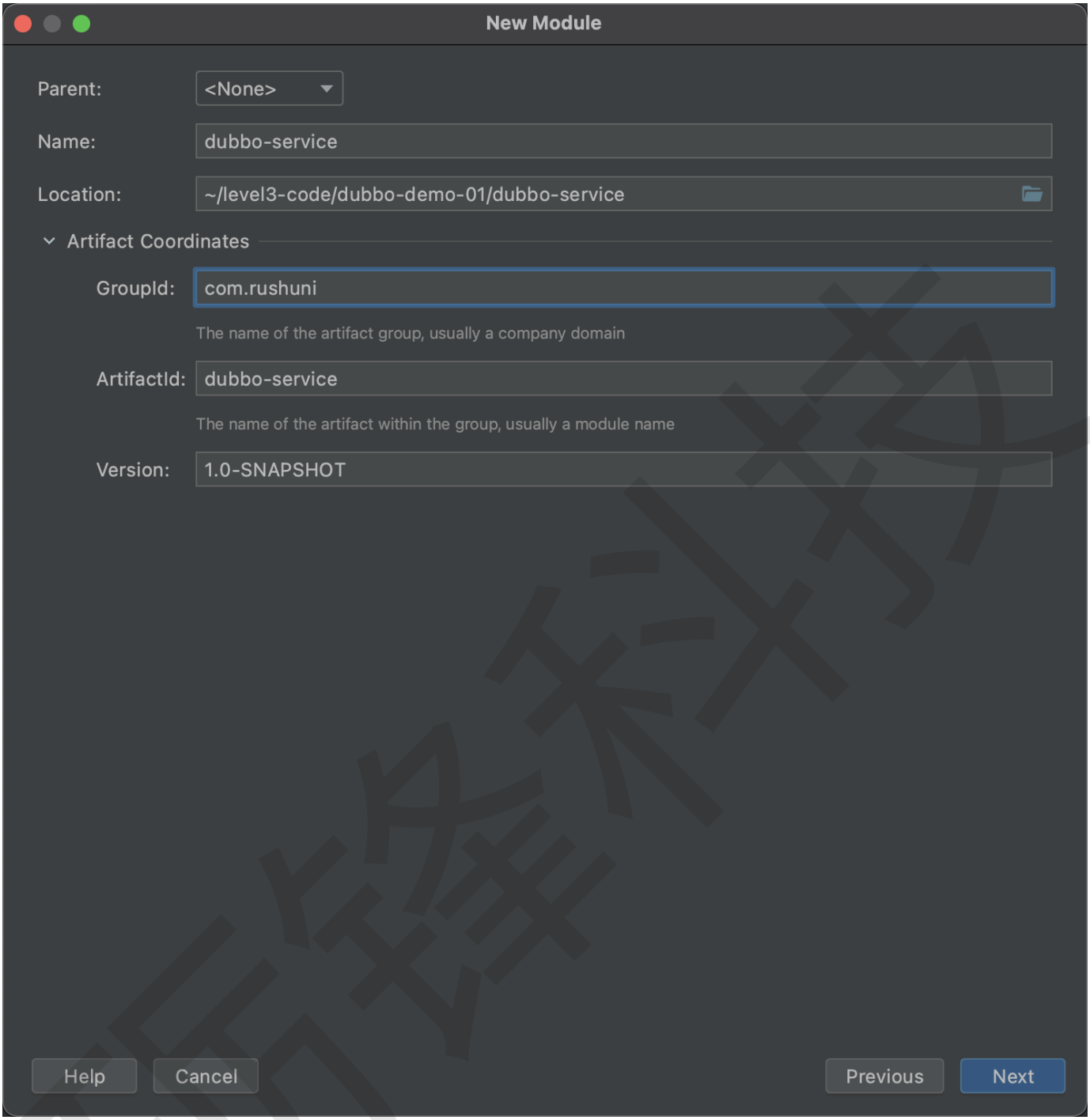
现在，我们创建两个应用，一个作为服务的提供方，一个作为服务的消费方。

通过 Dubbo 来实现服务消费方远程调用服务提供方的方法。

创建空工程，dubbo-demo-01



新建两个 module



添加依赖

dubbo-interface: pom.xml

XML 复制代码

```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.rushuni</groupId>
    <artifactId>dubbo-interface</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>

</project>
```

dubbo-service: pom.xml

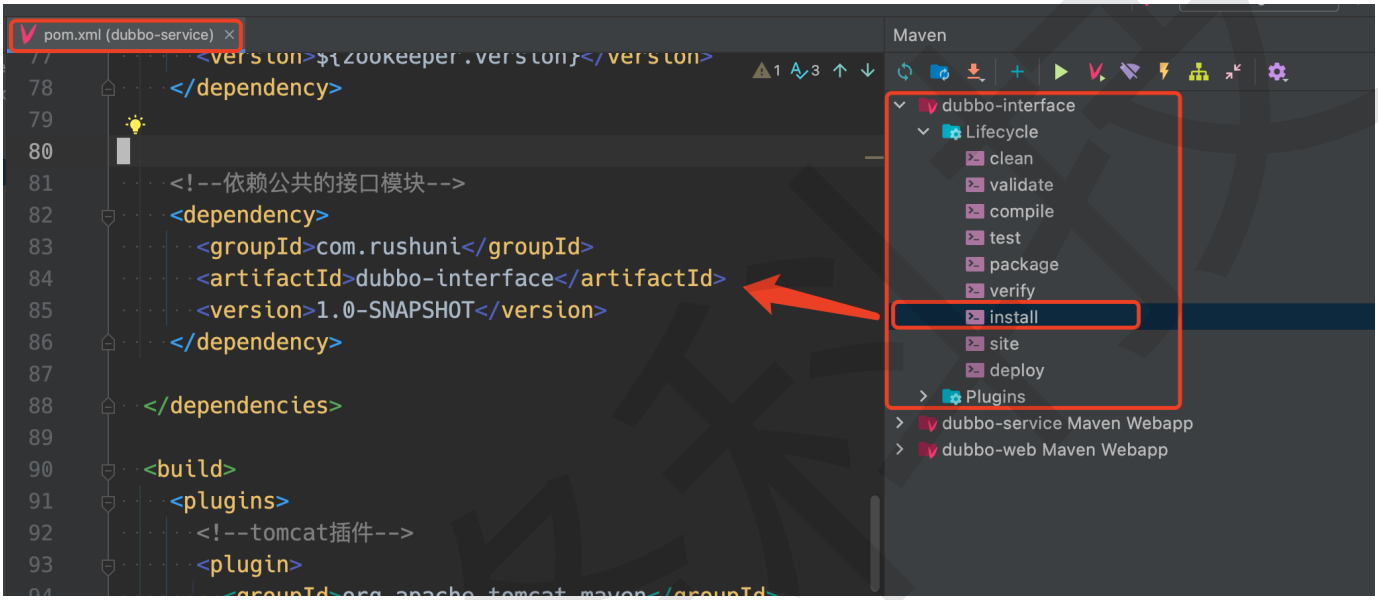
XML 复制代码

```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.rushuni</groupId>
  <artifactId>dubbo-service</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>dubbo-service Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <spring.version>5.3.9</spring.version>
    <dubbo.version>2.7.4.1</dubbo.version>
    <zookeeper.version>4.0.0</zookeeper.version>

  </properties>

  <dependencies>
    <!-- servlet3.0规范的坐标 -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
      <scope>provided</scope>
    </dependency>
    <!--spring的坐标-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!--springmvc的坐标-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>

```

```
    <!--日志-->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>1.7.21</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>1.7.21</version>
    </dependency>



    <!--Dubbo的起步依赖，版本2.7之后统一为org.apache.dubb0 -->
    <dependency>
      <groupId>org.apache.dubbo</groupId>
      <artifactId>dubbo</artifactId>
      <version>${dubbo.version}</version>
    </dependency>
    <!--ZooKeeper客户端实现 -->
    <dependency>
      <groupId>org.apache.curator</groupId>
      <artifactId>curator-framework</artifactId>
      <version>${zookeeper.version}</version>
    </dependency>
    <!--ZooKeeper客户端实现 -->
    <dependency>
      <groupId>org.apache.curator</groupId>
      <artifactId>curator-recipes</artifactId>
      <version>${zookeeper.version}</version>
    </dependency>


    <!--依赖公共的接口模块-->
    <dependency>
      <groupId>com.rushuni</groupId>
      <artifactId>dubbo-interface</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>

  </dependencies>

  <build>
    <plugins>
      <!--tomcat插件-->
      <plugin>
        <groupId>org.apache.tomcat.maven</groupId>
        <artifactId>tomcat7-maven-plugin</artifactId>
        <version>2.1</version>
```

```
        <configuration>
          <port>9000</port>
          <path>/</path>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

引用到 interface 模块，所以要对 interface 模块执行install



dubbo-web：pom.xml

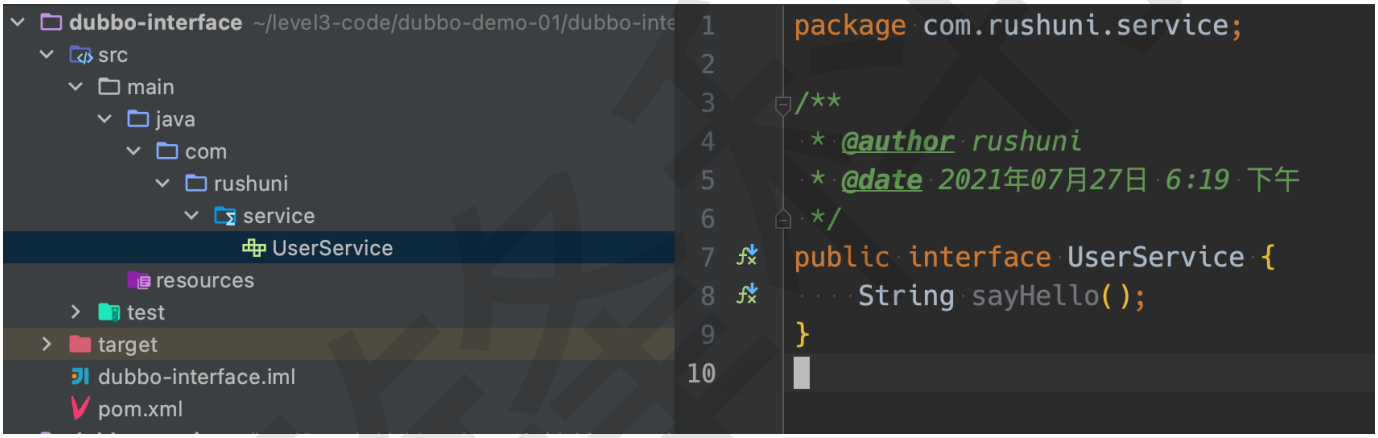
XML 复制代码

```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.rushuni</groupId>
  <artifactId>dubbo-web</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>dubbo-web Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <spring.version>5.1.9.RELEASE</spring.version>
    <dubbo.version>2.7.4.1</dubbo.version>
    <zookeeper.version>4.0.0</zookeeper.version>

  </properties>

  <dependencies>
    <!-- servlet3.0规范的坐标 -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
      <scope>provided</scope>
    </dependency>
    <!--spring的坐标-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!--springmvc的坐标-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>

```

```
    <!--日志-->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>1.7.21</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>1.7.21</version>
    </dependency>

    <!--Dubbo的起步依赖，版本2.7之后统一为org.apache.dubbo -->
    <dependency>
      <groupId>org.apache.dubbo</groupId>
      <artifactId>dubbo</artifactId>
      <version>${dubbo.version}</version>
    </dependency>
    <!--ZooKeeper客户端实现 -->
    <dependency>
      <groupId>org.apache.curator</groupId>
      <artifactId>curator-framework</artifactId>
      <version>${zookeeper.version}</version>
    </dependency>
    <!--ZooKeeper客户端实现 -->
    <dependency>
      <groupId>org.apache.curator</groupId>
      <artifactId>curator-recipes</artifactId>
      <version>${zookeeper.version}</version>
    </dependency>

    <!--依赖公共的接口模块-->
    <dependency>
      <groupId>com.rushuni</groupId>
      <artifactId>dubbo-interface</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>


    <!--依赖service模块-->
    <!-- <dependency>
         <groupId>com.rushuni</groupId>
         <artifactId>dubbo-service</artifactId>
         <version>1.0-SNAPSHOT</version>
     </dependency>-->

  </dependencies>


  <build>
```

```
    <plugins>
      <!--tomcat插件-->
      <plugin>
        <groupId>org.apache.tomcat.maven</groupId>
        <artifactId>tomcat7-maven-plugin</artifactId>
        <version>2.1</version>
        <configuration>
          <port>8000</port>
          <path>/</path>
        </configuration>
      </plugin>
    </plugins>
  </build>


</project>
```

完成dubbo-interface的接口编写：



Java 复制代码

```java
package com.rushuni.service;

/**
 * @author rushuni
 * @date 2021年07月27日 6:19 下午
 */
public interface UserService {
    String sayHello();
}
```

完成 provider，也即是 dubbo-service 模块

配置 web.xml，主要配置 spring 容器 applicationContext.xml 配置文件的位置:

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5">


  <!-- spring -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath*:spring/applicationContext*.xml</param-value>
  </context-param>
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>


</web-app>
```

applicationContext*.xml:

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
           http://dubbo.apache.org/schema/dubbo
           http://dubbo.apache.org/schema/dubbo/dubbo.xsd
           http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd">


    <!--<context:component-scan base-package="com.rushuni.service" />-->

    <!--dubbo的配置-->
    <!--1.配置项目的名称,唯一-->
    <dubbo:application name="dubbo-service"/>
    <!--2.配置注册中心的地址-->
    <dubbo:registry address="zookeeper://10.211.55.5:2181"/>
    <!--3.配置dubbo包扫描-->
    <dubbo:annotation package="com.rushuni.service.impl" />

</beans>
```

完成实现：

Java 复制代码

```java
package com.rushuni.service.impl;

import com.rushuni.service.UserService;
import org.apache.dubbo.config.annotation.Service;


// @Service 将该类的对象创建出来，放到Spring的IOC容器中  bean定义

/**
 * 将这个类提供的方法（服务）对外发布。将访问的地址 ip，端口，路径注册到注册中心中
 */
@Service
public class UserServiceImpl implements UserService {

    @Override
    public String sayHello() {
        return "hello, dubbo - provider!~";
    }
}
```

完成消费者，也就是 dubbo-web，它将调用 service。

web.xml

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5">



  <!-- Springmvc -->
  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 指定加载的配置文件 ，通过参数contextConfigLocation加载-->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:spring/springmvc.xml</param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>


</web-app>
```

配置springmvc.xml

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
         http://dubbo.apache.org/schema/dubbo
http://dubbo.apache.org/schema/dubbo/dubbo.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd">



    <mvc:annotation-driven/>
    <context:component-scan base-package="com.rushuni.controller"/>


    <!--dubbo的配置-->
    <!--1.配置项目的名称,唯一-->
    <dubbo:application name="dubbo-web" >
        <dubbo:parameter key="qos.port" value="33333"/>
    </dubbo:application>
    <!--2.配置注册中心的地址-->
    <dubbo:registry address="zookeeper://10.211.55.5:2181"/>
    <!--3.配置dubbo包扫描-->
    <dubbo:annotation package="com.rushuni.controller" />


</beans>
```

完成接口

Java 复制代码

```java
package com.rushuni.controller;

import com.rushuni.service.UserService;
import org.apache.dubbo.config.annotation.Reference;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/user")
public class UserController {

    //注入Service
    //@Autowired//本地注入
    /*
        1. 从zookeeper注册中心获取userService的访问url
        2. 进行远程调用RPC
        3. 将结果封装为一个代理对象。给变量赋值
     */

    /**
     * 远程注入
      */
    @Reference
    private UserService userService;

    @RequestMapping("/sayHello")
    public String sayHello(){
        return userService.sayHello();
    }

}

```