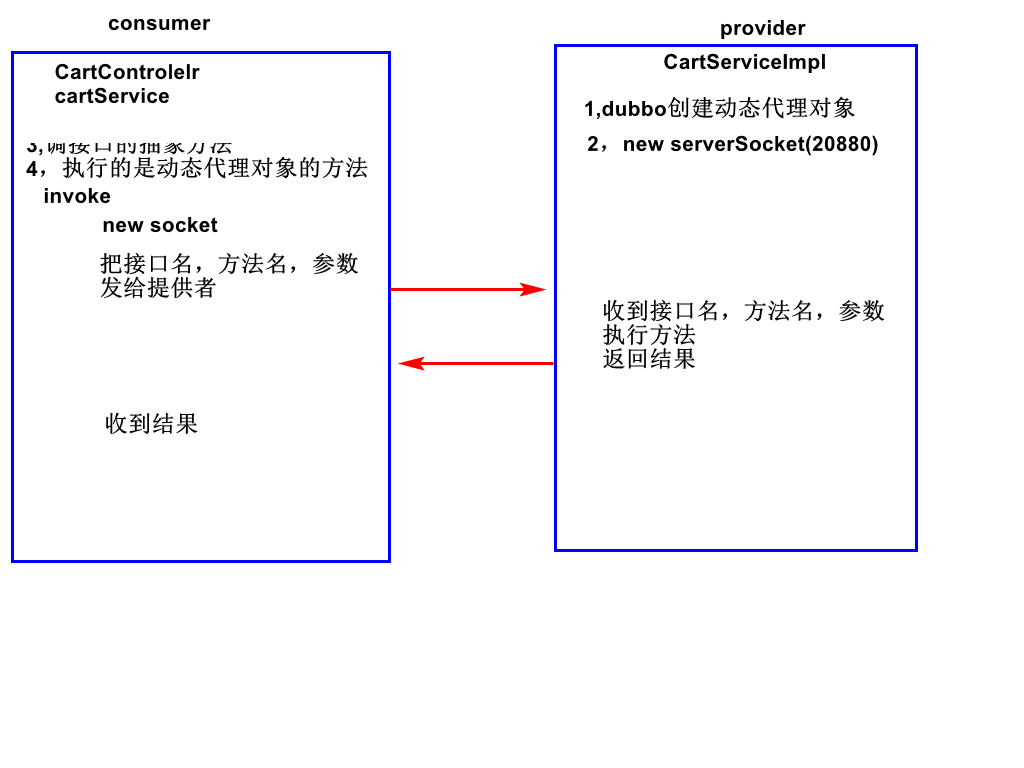
# 复习

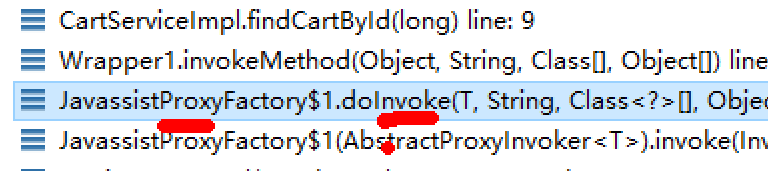
Dubbo工作原理



消费者动态代理



提供者动态代理



Springcloud

Provider中要写controller,通过controller调用service

Dubbo不写controller

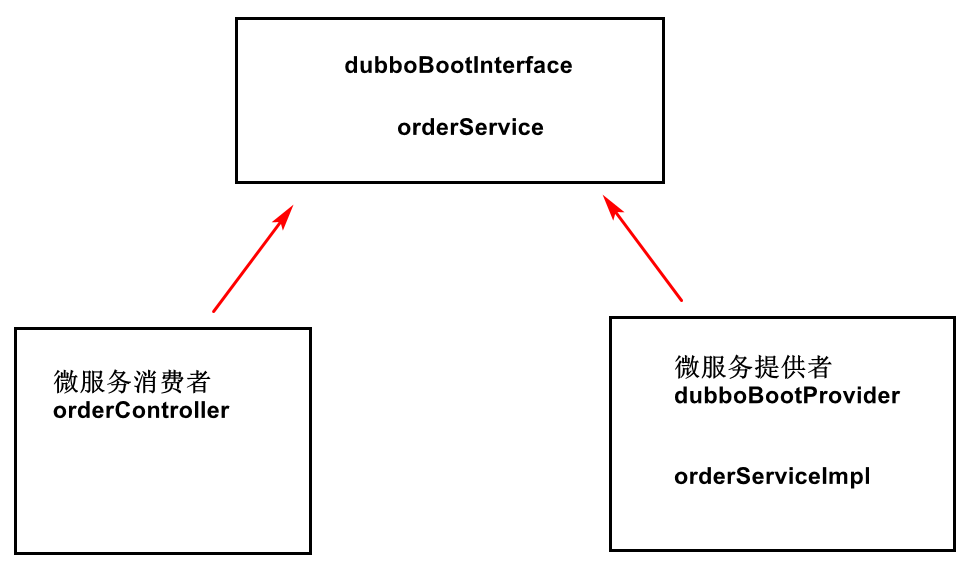
Springcloud中消费者连接提供者用的http

Dubbo用的rpc 远程进程调用

Rpc比http快。

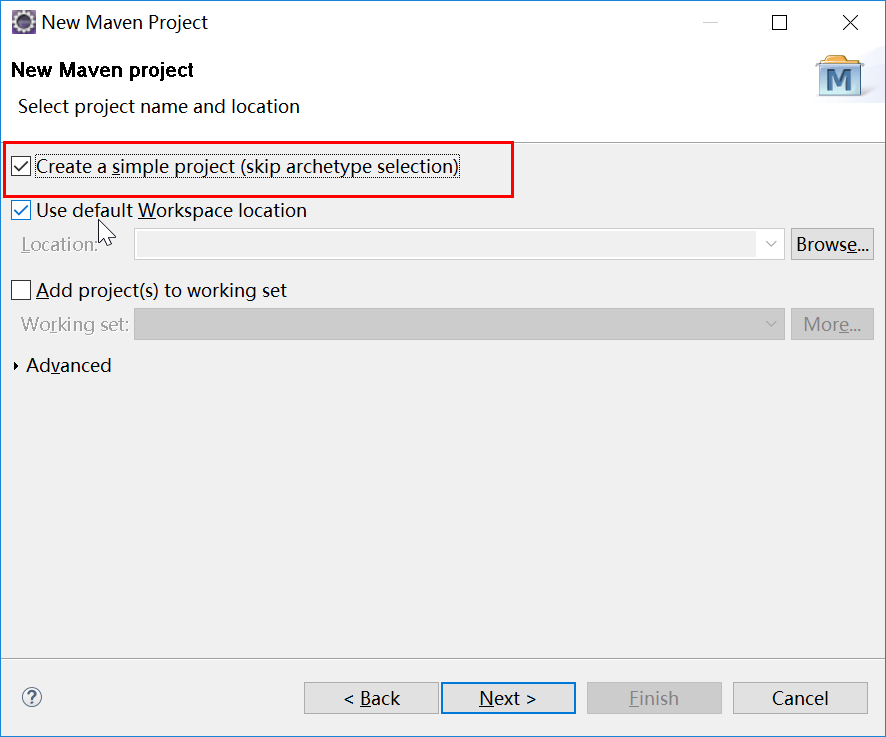
## Springboot整合Dubbo

### 项目分析



### 创建dubboBootInterface项目

创建项目时，选择“create a simple project”

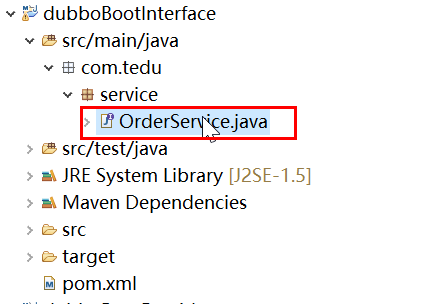


选择“create a simple project”创建出的项目有resources文件夹，quick start没有。

groupId是com.tedu

artifactId是dubboBootInterface

1. 创建包com.tedu.service包



1. 创建接口

**package** com.tedu.service;

**public** **interface** OrderService {

**public** String findOrderById(Long userId);

}

1. 发布jar

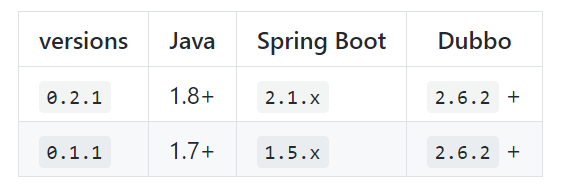
Run as 🡪maven install

### 创建第一个提供者 dubboBootProvider

Dubbo与springboot整合框架官网

<https://github.com/apache/incubator-dubbo-spring-boot-project>

springboot与dubbo版本兼容性如下：



1. 创建maven项目，选“create a simple project”，项目名为dubboBootProvider
2. 拷贝下面的parent和依赖

<!-- Spring Boot 启动父依赖 -->

<parent>

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

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

<version>1.5.4.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<dependencies>

<!-- Spring Boot Dubbo 依赖 -->

<dependency>

<groupId>com.alibaba.boot</groupId>

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

<version>0.1.1</version>

</dependency>

<!-- Spring Boot Web 依赖 -->

<dependency>

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

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

</dependency>

</dependencies>

添加dubbo依赖，因为dubbo依赖spring等jar,这些jar会自动添加进来，如图-25 所示

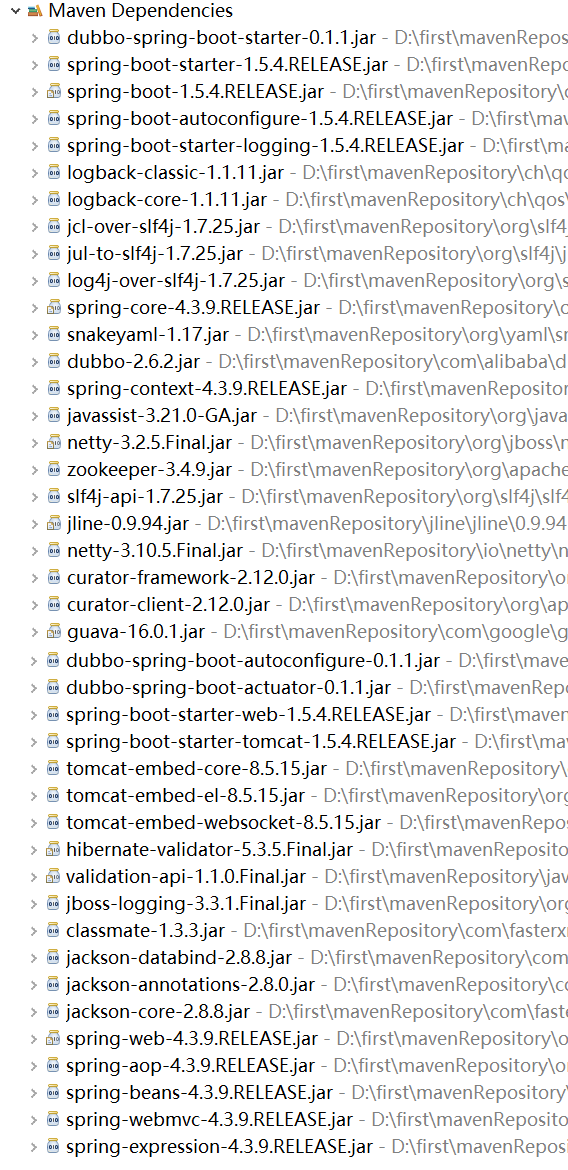


图- 25

1. 依赖dubboBootInterface项目。
2. 拷贝“\亿发,移动端课前资料\dubbo\springboot整合dubbo\provider\”中的application.properties和application.yml到/src/main/resources/。文件中的内容如下

application.yml内容如下：

server:

port: 8001

application.properties内容如下：

# Spring boot application

spring.application.name = order

# Base packages to scan Dubbo Components (e.g., @Service, @Reference)

dubbo.scan.basePackages = com.tedu

# Dubbo Config properties

## ApplicationConfig Bean

dubbo.application.id = order

dubbo.application.name = order

## ProtocolConfig Bean

dubbo.protocol.id = dubbo

dubbo.protocol.name = dubbo

dubbo.protocol.port = 20991

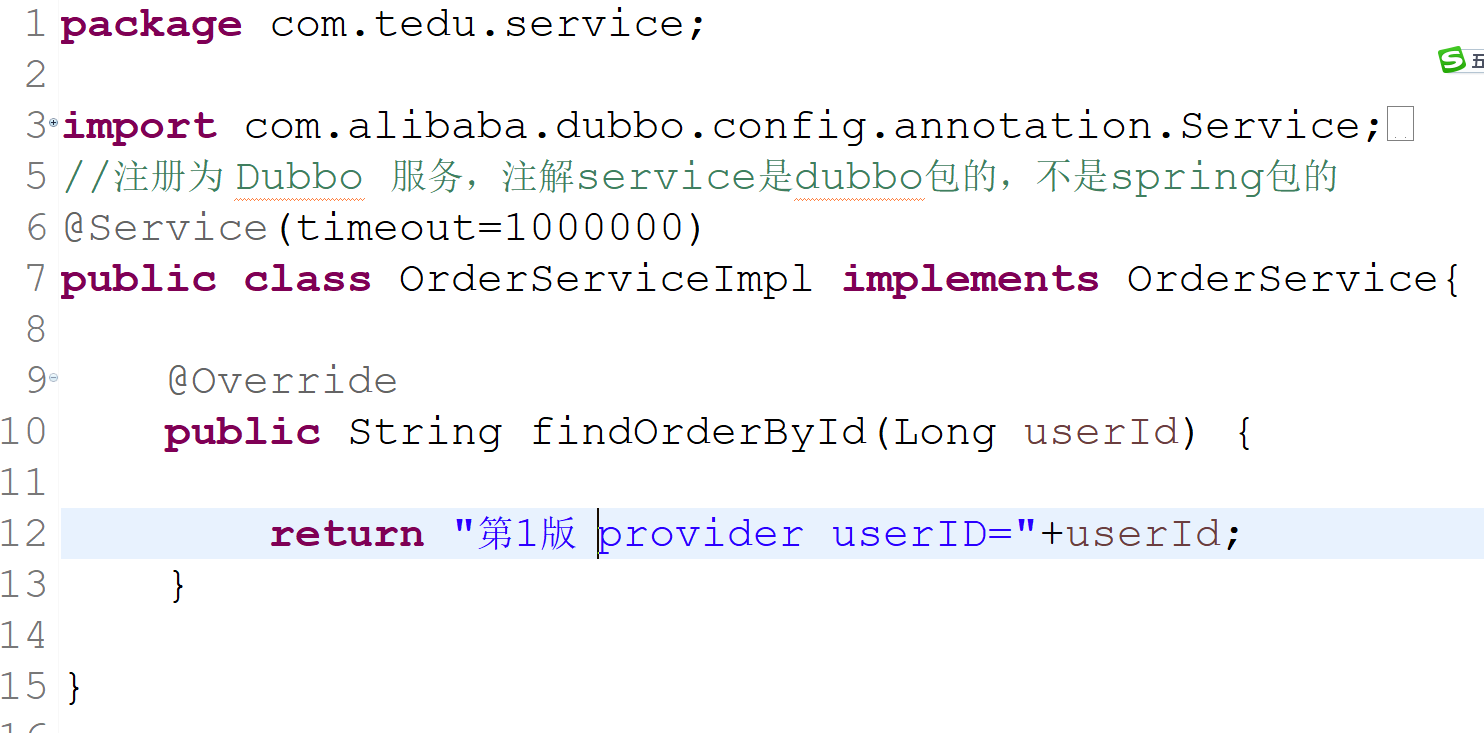
## RegistryConfig Bean

dubbo.registry.id = my-registry

dubbo.registry.address = zookeeper://192.168.216.230:2181

1. 添加服务实现类

创建包com.tedu.service



1. 创建启动类

@SpringBootApplication

**public** **class** RunProviderAPP {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(RunProviderAPP.**class**, args);

}

}

### 连接失败解决办法

报错信息为KeeperErrorCode = ConnectionLoss

1. 在linux上重启zookeeper服务，关闭防火墙
2. 关闭客户端，重启eclipse

### zkCli查看服务

cd /usr/local/src/zookeeper-3.4.8/bin

sh zkCli.sh

[zk: localhost:2181(CONNECTED) 1] ls /

[dubbo, zookeeper]

//查看 dubbo服务

[zk: localhost:2181(CONNECTED) 2] ls /dubbo

[com.tedu.service.OrderService]

//查看cartService有多少个提供者

[zk:localhost:2181(CONNECTED)3] ls /dubbo/com.tedu.service.OrderService/providers

//退出zkCli

[zk:localhost:2181(CONNECTED)3] quit

1. dubbo监控中能查到服务,选中服务查看详细信息

访问localhost:8090，如图- 31所示



图- 31

### 创建消费者dubboBootConsumer

1. 拷贝“\亿发,移动端课前资料\dubbo\springboot整合dubbo\consumer\初始版本“文件夹下的dubboBootConsumer项目到工作区，导入到工作区中。
2. 测试controller能不能访问

http://localhost:9001/order/find

1. 拷贝dubbo依赖

<!-- Spring Boot Dubbo 依赖 -->

<dependency>

<groupId>com.alibaba.boot</groupId>

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

<version>0.1.1</version>

</dependency>

1. 添加dubboBootInterface依赖
2. 从“\亿发,移动端课前资料\dubbo\springboot整合dubbo\consumer”拷贝application.propeties，消费者的配置中不用配置端口号。文件内容如下

# Spring boot application

spring.application.name = order

# Base packages to scan Dubbo Components (e.g., @Service, @Reference)

dubbo.scan.basePackages = com.tedu

# Dubbo Config properties

## ApplicationConfig Bean

dubbo.application.id = order

dubbo.application.name = order

## ProtocolConfig Bean

##dubbo.protocol.id = dubbo

##dubbo.protocol.name = dubbo

##ubbo.protocol.port = 20001

## RegistryConfig Bean

dubbo.registry.id = my-registry

dubbo.registry.address = zookeeper://192.168.216.230:2181

1. 修改OrderController

@RestController

@RequestMapping("/order")

**public** **class** OrderController {

@Reference(timeout=100000,check=**false**)

OrderService orderService;

@RequestMapping("/find")

**public** String find() {

String result=orderService.findOrderById(18L);

**return** result;

}

}

1. 用断点方式启动RunConsumerAPP
2. 通过zkCli查看信息

cd /usr/local/src/zookeeper-3.4.8/bin

sh zkCli.sh

[zk: localhost:2181(CONNECTED) 1] ls /

[dubbo, zookeeper]

//查看 dubbo服务

[zk: localhost:2181(CONNECTED) 2] ls /dubbo

[com.tedu.service.CartService]

//查看cartService有多少个提供者

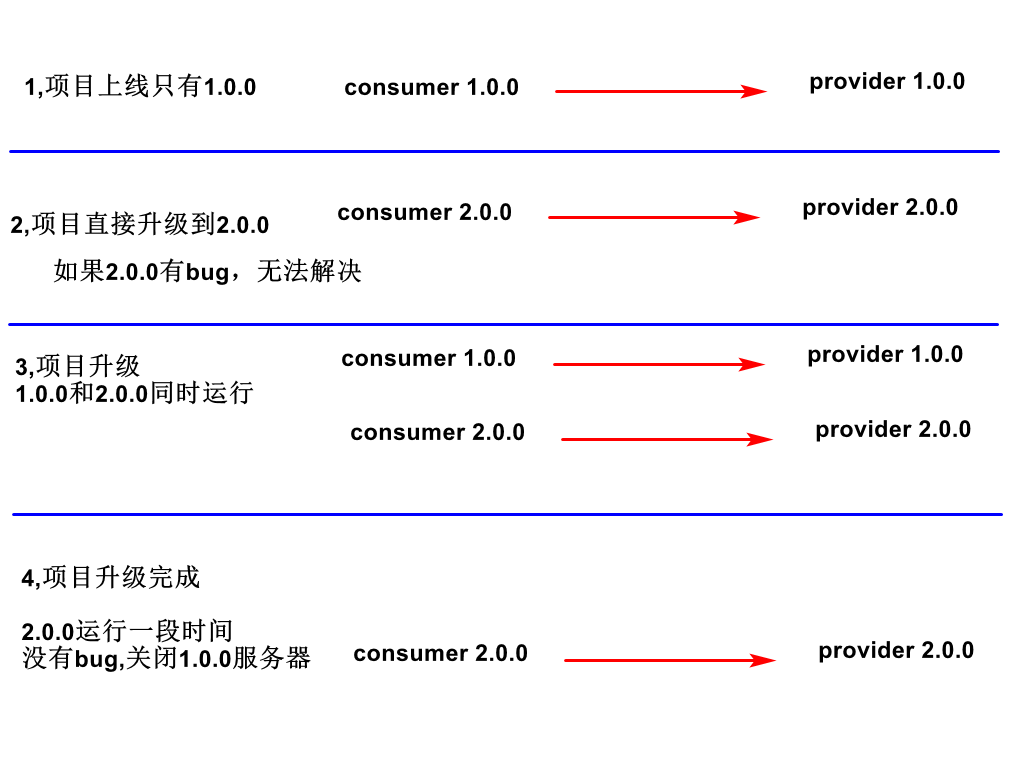
[zk:localhost:2181(CONNECTED)3] ls /dubbo/com.tedu.service.OrderService/consumers

//退出zkCli

[zk:localhost:2181(CONNECTED)3] quit

## 微服务升级

### 分析



灰度发布

### 加version

OrderServiceImpl加version

@Service(version="1.0.0",timeout=1000000)

**public** **class** OrderServiceImpl **implements** OrderService

OrderController加version

@Reference(version="1.0.0",timeout=100000,check=**false**)

OrderService orderService;

### 拷贝dubboBootProvider产生第二版

* + - 1. 拷贝dubboBootProvider改名成dubboBootProvider2。
      2. 修改pom.xml中的artifactid。
      3. 修改application.yml中的tomcat端口号和dubbo端口号。
      4. 把RunProviderAPP改名成RunProvider2APP
      5. 修改orderServiceImpl中的版本号和字符串

@Service(version="2.0.0",timeout=1000000)

**public** **class** OrderServiceImpl **implements** OrderService{

@Override

**public** String findOrderById(Long userId) {

**return** "第2版 provider userID="+userId;

}

}

### 拷贝dubboBootConsumer产生第二版

* + - 1. 拷贝dubboBootConsumer改名成dubboBootConsumer2，
      2. 修改pom.xml中的artifactid,
      3. 修改application.yml中的tomcat端口号。
      4. 把RunConsumerAPP改名成RunConsumer2APP
      5. 修改orderController中的版本号为2.0.0

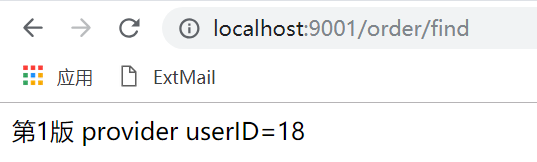
@Reference(version="2.0.0",timeout=100000,check=**false**)

OrderService orderService;

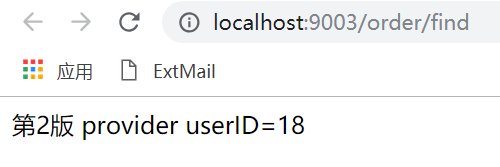
### 测试

两个消费者，两个提供者都重启。

访问第一个消费者



访问第二个消费者



Version 2.0.0版本运行一段时间没有bug后，关闭1.0.0的微服务。

## 概念介绍

### RPC

RPC（Remote Procedure Call）—远程过程调用

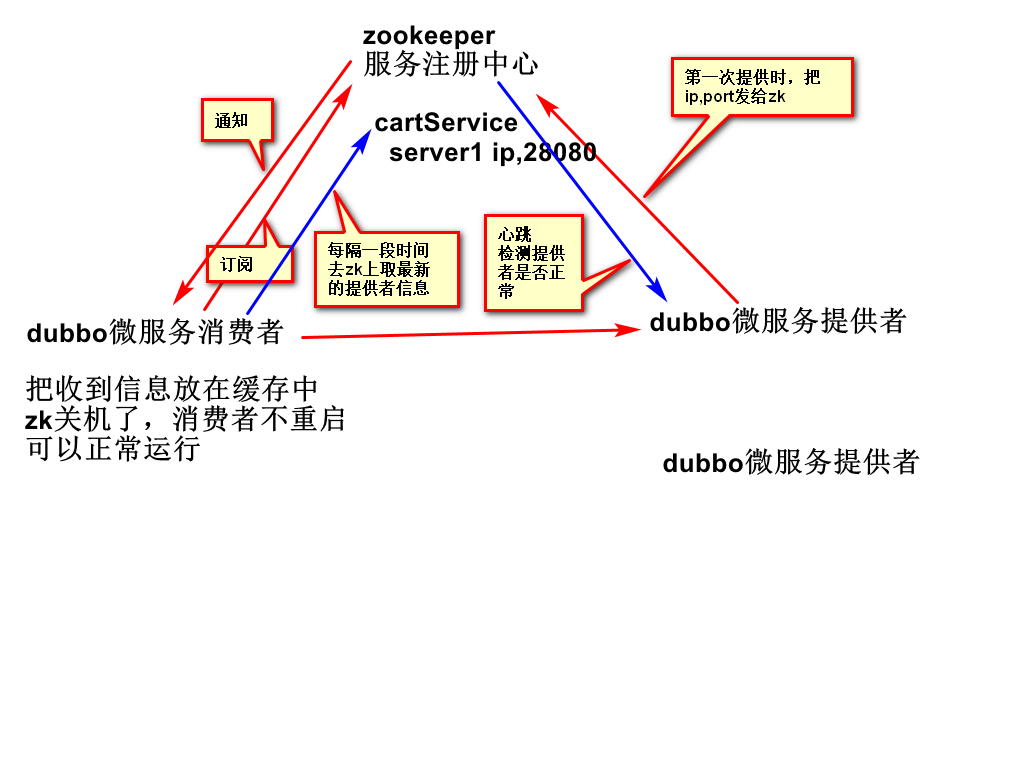
Dubbo微服务框架底层用的是RPC。

### SOA介绍

SOA面向服务的架构（Service-oriented architecture）是一个组件模型，它将应用程序的不同功能单元（称为服务）通过接口联系起来。各种各样的服务可以通过接口进行交互。

## Dubbo，zookeeper总结

### Zookeeper总结



Zookeeper还可以管理redis,mysql,tomcat等服务

### Dubbo总结

Dubbo是远程调用一个方法。是跨进程调用

### Dubbo优点

1. 远程通信，向本地调用一样调用远程方法。
2. 服务自动发现和注册，可平滑添加或者删除服务提供者。
3. 相比httpClient少写代码
4. 易定位bug
5. 易扩展
6. 易发布
7. 支持集群
8. 支持故障转移

### Dubbo缺点

1、性能低，分布式系统是跨进程，跨网络的，性能很受网络延迟和带宽的影响。

2、可靠性差：由于高度依赖网络状况，任何一次远程调用都可能失败。

3、运维成本：一个系统拆成了多个服务，每个服务都得配置，部署，监控，日志处理

# 拼多检索重构

## 项目分析

检索是个独立功能，应该做成微服务。

### Dubbo实现

interface项目的作用（service,pojo），与消费者，提供者的关系，如图- 38所示。

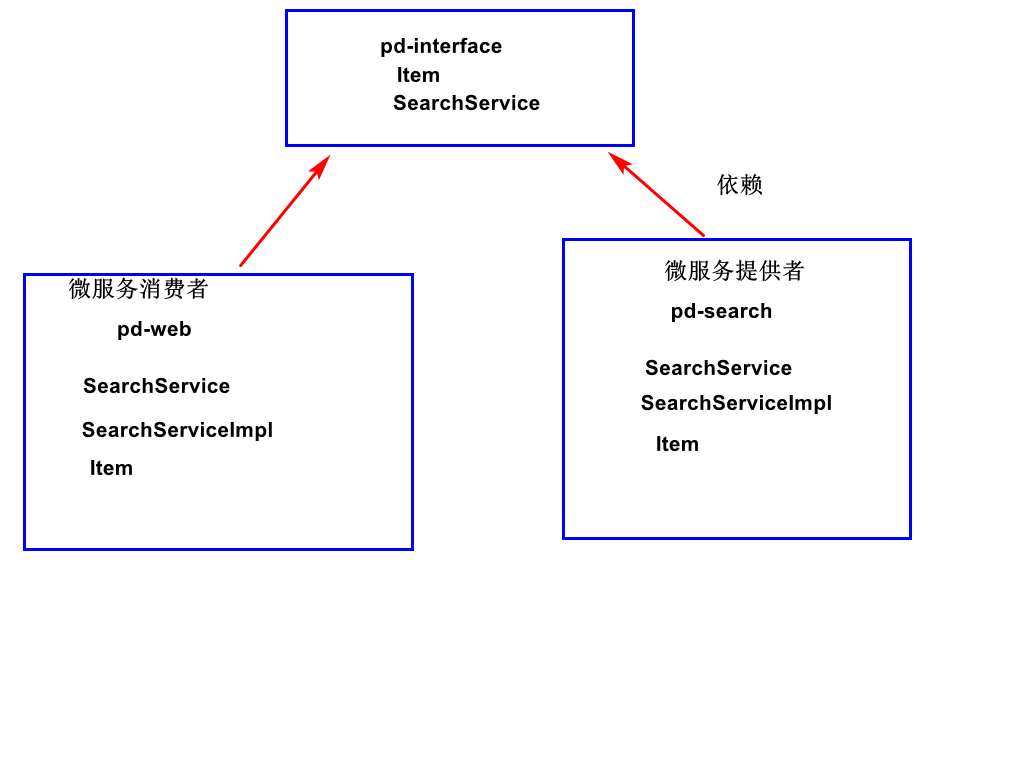


图- 38

## 创建pd-interface

### 创建项目

创建项目时，选择“create a simple project”

，如图-39 所示

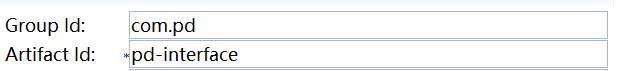


图- 39

### 添加依赖

Item实体类中要用到solrJ框架的Field注解。需要添加solrJ依赖。

<dependencies>

<dependency>

<groupId>org.apache.solr</groupId>

<artifactId>solr-solrj</artifactId>

<version>5.2.1</version>

</dependency>

</dependencies>

### 拷贝pojo,service

1.从pd-web拷贝pojo,service，如图-40 所示

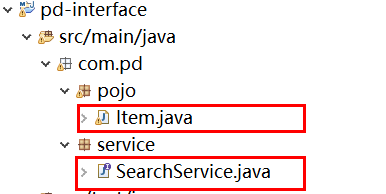


图- 40

创建com.pd.pojo包

从pd-web中拷贝Item.java到pd-interface/com.pd.pojo包,

创建com.pd.service包，

从pd-web中拷贝SearchService.java到pd-interface/com.pd.service

## 修改pd-search

### 拷贝pd-web项目改名成pd-search

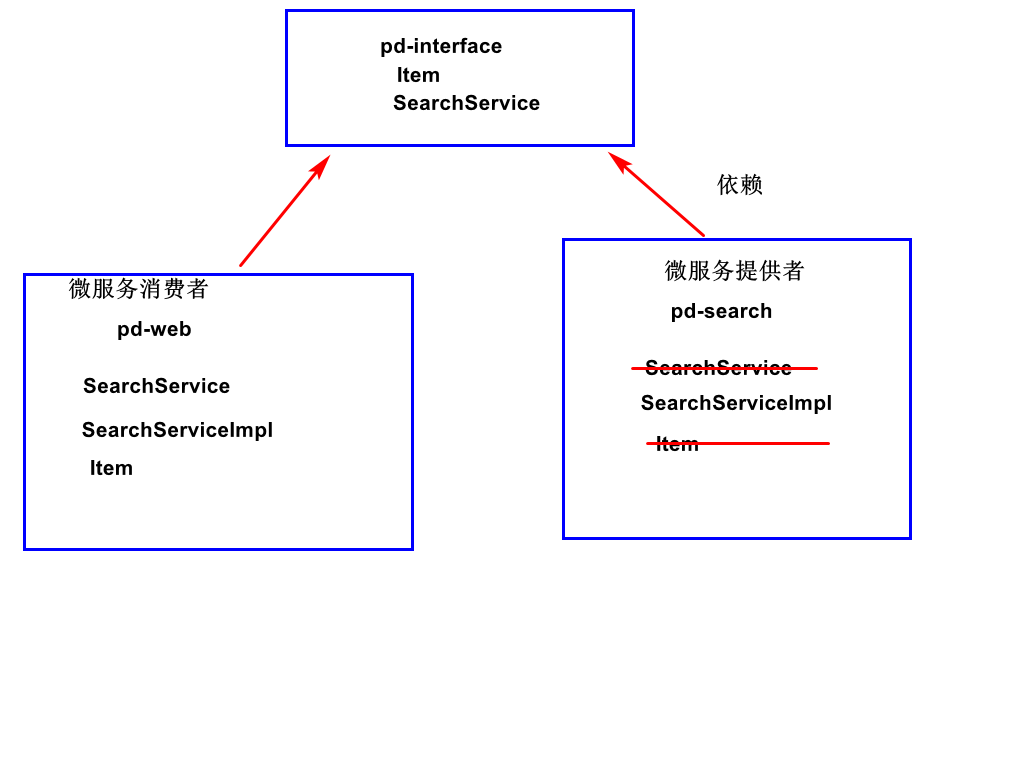
修改pom.xml中的artifactId

修改application.yml中的tomcat端口号

把RunPdApp改名成RunSearchApp

### 依赖pd-interface

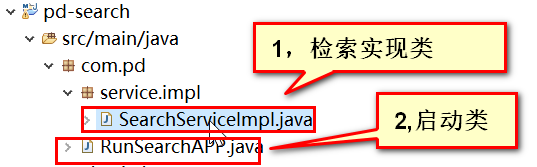
### 删除pojo和service，如图-41 所示



Pd-search要删除item和searchService,上图中的SearchServiceImpl不要删除

图- 41

只保留SearchServiceImp和RunSearchAPP



### 添加dubbo依赖

<!-- Spring Boot Dubbo 依赖 -->

<dependency>

<groupId>com.alibaba.boot</groupId>

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

<version>0.1.1</version>

</dependency>

### 修改application.properties，把原先的内容全删除，拷贝下面的内容，修改zk地址为自己的。

# Spring boot application

spring.application.name = pd

# Base packages to scan Dubbo Components (e.g., @Service, @Reference)

dubbo.scan.basePackages = com.pd

# Dubbo Config properties

## ApplicationConfig Bean

dubbo.application.id = pd

dubbo.application.name = pd

## ProtocolConfig Bean

dubbo.protocol.id = dubbo

dubbo.protocol.name = dubbo

dubbo.protocol.port = 30001

## RegistryConfig Bean

dubbo.registry.id = my-registry

dubbo.registry.address = zookeeper://192.168.216.230:2181

### 修改SearchServiceImpl中的@Service为dubbo包的。

//由dubbo框架来调实现类，要把service注解改成dubbo的

@Service(version="1.0.0")

**public** **class** SearchServiceImpl **implements** SearchService

### 启动测试

1. Zk客户端上查看服务

sh /zkCli.sh

ls /

ls /dubbo

ls /dubbo/com.pd.service.SearchService/providers

quit

## 修改pd-web中检索功能

### 添加pd-interface依赖

### 删除pojo和service，如图-44 所示.

删除item实体类，使用pd-interface项目中的Item。

删除SearchService接口, 使用pd-interface项目中的SearchService。

删除SearchServiceImpl实现类,实现类由pd-search项目实现。

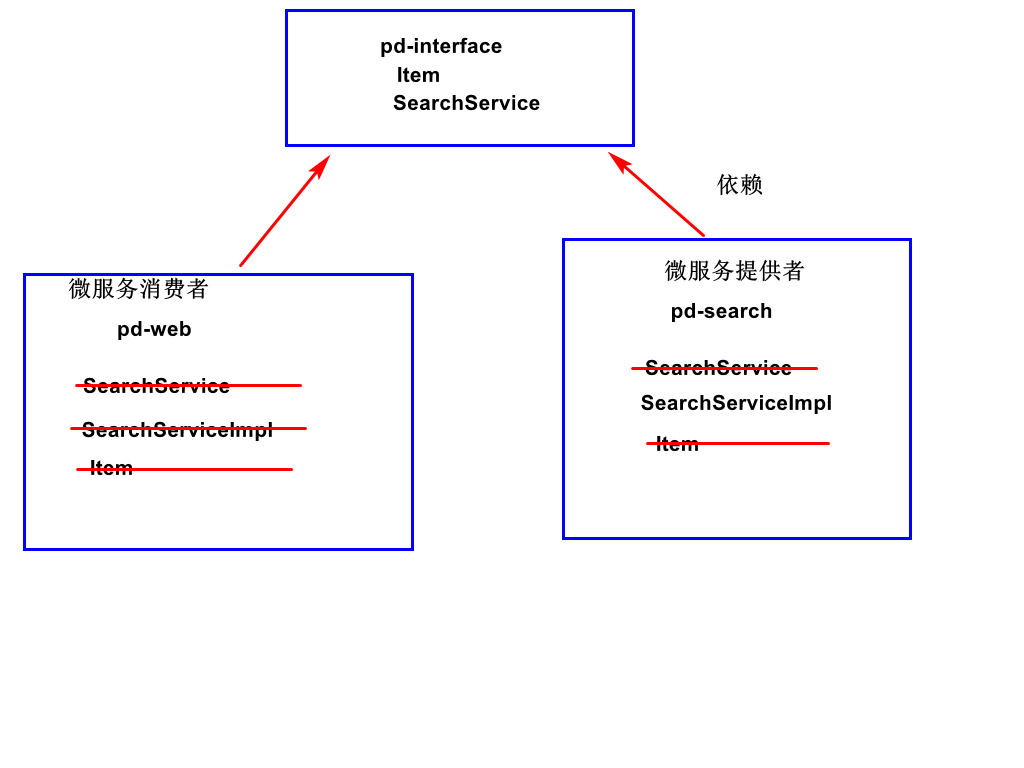


图- 44

### 添加dubbo依赖

<!-- Spring Boot Dubbo 依赖 -->

<dependency>

<groupId>com.alibaba.boot</groupId>

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

<version>0.1.1</version>

</dependency>

### 拷贝配置信息

拷贝下面内容到application.properties中，修改zk的ip。

# Spring boot application

spring.application.name = pd

# Base packages to scan Dubbo Components (e.g., @Service, @Reference)

dubbo.scan.basePackages = com.pd

# Dubbo Config properties

## ApplicationConfig Bean

dubbo.application.id = pd

dubbo.application.name = pd

## ProtocolConfig Bean

##dubbo.protocol.id = dubbo

##dubbo.protocol.name = dubbo

##dubbo.protocol.port = 30001

## RegistryConfig Bean

dubbo.registry.id = my-registry

dubbo.registry.address = zookeeper://192.168.216.230:2181

### 修改searchController

@Controller

**public** **class** SearchController {

//@Autowired

@Reference(version="1.0.0",check=**false**,timeout=90000)

SearchService searchService;

@RequestMapping("/search/toSearch.html")

**public** String search(String key,Model model)

**throws** Exception

{

List<Item> itemList=

searchService.findItemByKey(key);

model.addAttribute("list", itemList);

//转发到search.jsp

**return** "/search.jsp";

}

}

### 启动solr和rabbitmq

cd /usr/sbin

service rabbitmq-server start

cd /usr/local/src/solr-5.2.1/bin

sh solr start

### 断点启动pd-web项目

### 查看消费者是否注册成功

1. Zk客户端上查看服务

cd /usr/local/src/zookeeper-3.4.8/bin

sh /zkCli.sh

ls /

ls /dubbo

ls /dubbo/com.pd.service.SearchService/consumers

quit

1. dubbo监控上查看服务

访问http://localhost:8090,

### 测试

直接输入网址<http://www.pd.com/search/toSearch.html?key>=电脑,断点跟踪如如图- 46所示。

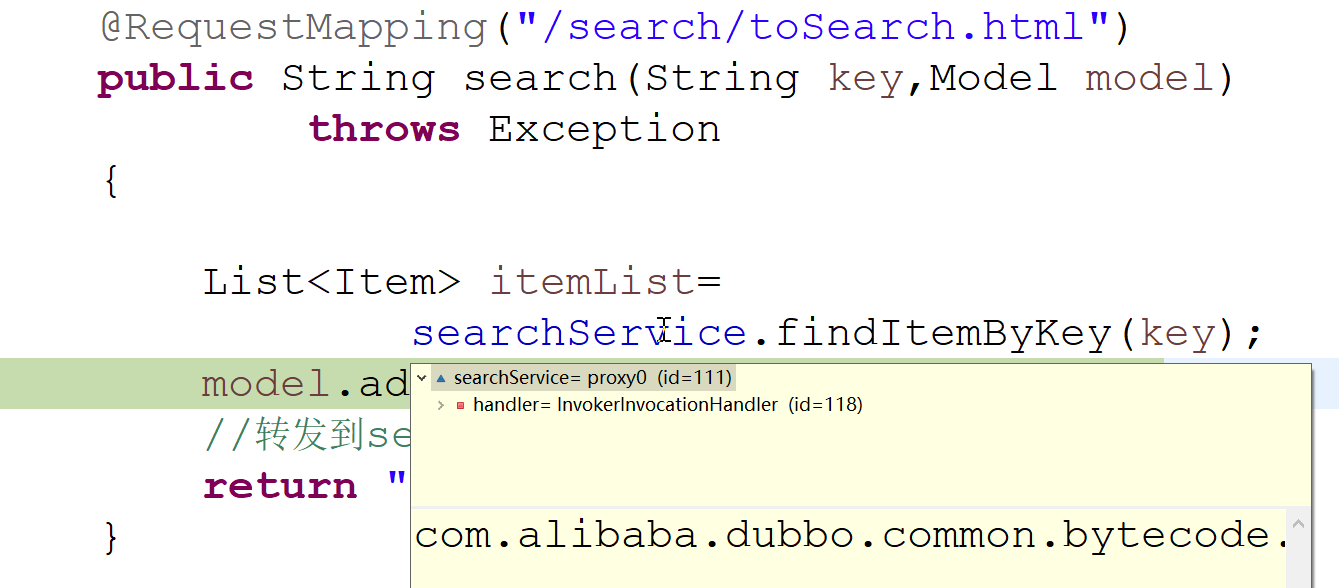


图- 46

### 页面效果，如图-47 所示



图- 47

474530502@qq.com

# Uml

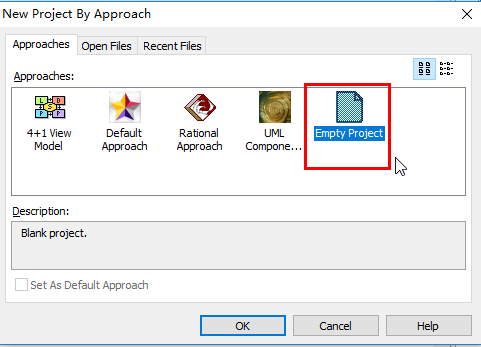
设计软件的图纸

设计软件的语言

用starUml对检索系统做设计，

生成图，用图来描述软件如何实现

设计完后能生成代码



Src

Com.pd.pojo

Item{

String title

String sellPoint

}

Com.pd.search

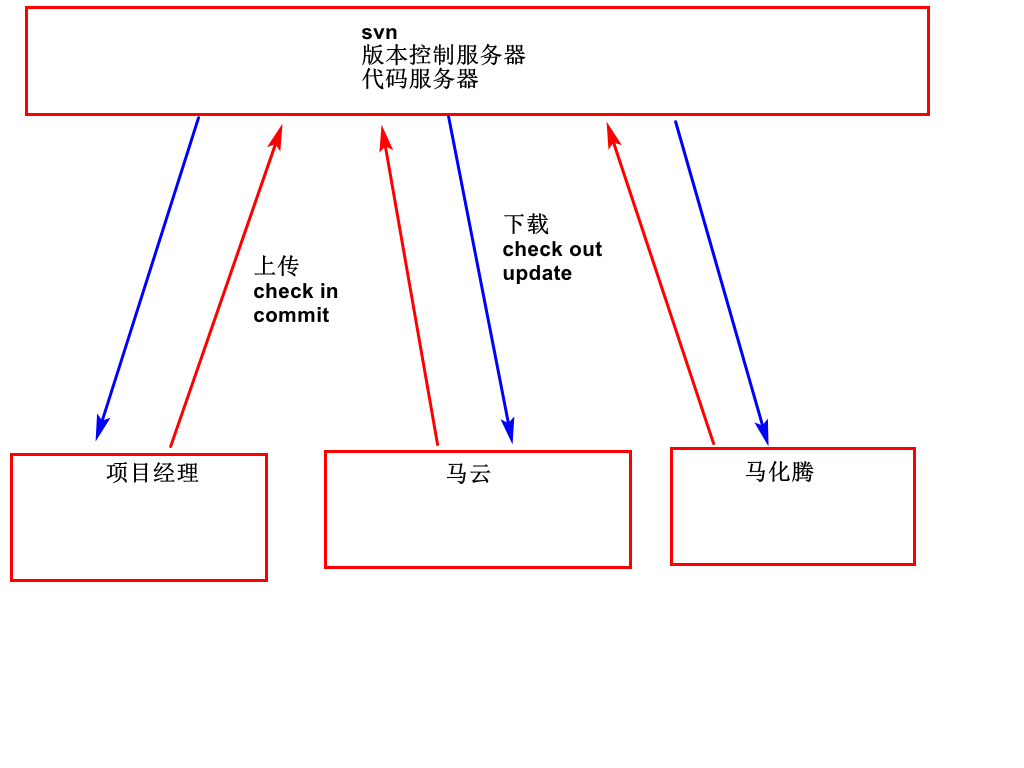
searchController

String search(String key)

searchService

List<Item> search(String key)

# Svn



\亿发,移动端课前资料\software\VisualSVN\ VisualSVN-Server-2.5.4.msi

https://www.cnblogs.com/ruiati/p/3584120.html 

# 简历

重点突出最难的项目和最难的技术点

## 过HR

### 项目经验丰富

最少3个，项目技术描述多

### 技术会的多

技能，把最难的写在最前面

相关专业，本科

## 过技术

有很多难的技术 ，新技术，会技术多。