# Dubbo产生的背景

## 什么是Dubbo

一款分布式服务框架

高性能和透明化的RPC远程服务调用方案

SOA服务治理方案

每天为2千多个服务提供大于30亿次访问量支持，并被广泛应用于阿里巴巴集团的各成员站点以及别的公司的业务中。

## Dubbo注册中心

对于服务提供方，它需要发布服务，而且由于应用系统的复杂性，服务的数量、类型也不断膨胀；

对于服务消费方，它最关心如何获取到它所需要的服务，而面对复杂的应用系统，需要管理大量的服务调用。

而且，对于服务提供方和服务消费方来说，他们还有可能兼具这两种角色，即既需要提供服务，有需要消费服务。

通过将服务统一管理起来，可以有效地优化内部应用对服务发布/使用的流程和管理。服务注册中心可以通过特定协议来完成服务对外的统一。

Dubbo提供的注册中心有如下几种类型可供选择：

Multicast注册中心

Zookeeper注册中心

Redis注册中心

Simple注册中心

## Dubbo的优缺点

优点：

透明化的远程方法调用

- 像调用本地方法一样调用远程方法；只需简单配置，没有任何API侵入。

软负载均衡及容错机制

可在内网替代nginx lvs等硬件负载均衡器。

服务注册中心自动注册 & 配置管理

-不需要写死服务提供者地址，注册中心基于接口名自动查询提供者ip。

使用类似zookeeper等分布式协调服务作为服务注册中心，可以将绝大部分项目配置移入zookeeper集群。

服务接口监控与治理

-Dubbo-admin与Dubbo-monitor提供了完善的服务接口管理与监控功能，针对不同应用的不同接口，可以进行 多版本，多协议，多注册中心管理。

缺点：

只支持JAVA语言

# Dubbo快速入门

环境步骤：

1. 安装Zookeepr启动
2. 创建Maven项目搭建生产者和消费者
3. 安装DubboAdmin平台，实现监控

## Zookeeper介绍与安装

Dubbo注册中心采用的是Zookeeper。为什么采用Zookeeper呢？

Zookeeper是一个分布式的服务框架，是树型的目录服务的数据存储，能做到集群管理数据 ，这里能很好的作为Dubbo服务的注册中心。

Dubbo能与Zookeeper做到集群部署，当提供者出现断电等异常停机时，Zookeeper注册中心能自动删除提供者信息，当提供者重启时，能自动恢复注册数据，以及订阅请求

## 创建Maven工程

项目结构：

主要分三大模块：

Itmayiedu-dubbo-api : 存放公共接口；

Itmayiedu-dubbo-consumer :　调用远程服务；

Itmayiedu-dubbo-provider : 提供远程服务。

## Itmayiedu-dubbo-api 新建接口

|  |
| --- |
| **public** **interface** DemoApiService {  **public** String getUser(Long userId);  } |

## 生产者提供服务接口

### itmayiedu-dubbo-provider 实现接口业务逻辑

|  |
| --- |
| **public** **class** DemoApiServiceImpl **implements** DemoApiService {  **public** String getUser(Long userId) {  System.***out***.println("生产者调用消费者服务接口userId:" + userId);  **return** "yushengjun";  }  } |

### Maven依赖信息

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.itmayiedu</groupId>  <artifactId>itmayiedu-dubbo-api</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>0.10</version>  </dependency>  <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>org.jboss.netty</groupId>  <artifactId>netty</artifactId>  <version>3.2.5.Final</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.3.9.RELEASE</version>  </dependency>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <version>2.5.3</version>  <exclusions>  <exclusion>  <groupId>org.springframework</groupId>  <artifactId>spring</artifactId>  </exclusion>  <exclusion>  <groupId>org.jboss.netty</groupId>  <artifactId>netty</artifactId>  </exclusion>  </exclusions>  </dependency>  </dependencies> |

### 新增 dubbo-provider.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://code.alibabatech.com/schema/dubbo"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://code.alibabatech.com/schema/dubbo*  *http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>  <!--定义了提供方应用信息，用于计算依赖关系；在 dubbo-admin 或 dubbo-monitor 会显示这个名字，方便辨识 -->  <dubbo:application name=*"demotest-provider"* />  <!--使用 zookeeper 注册中心暴露服务，注意要先开启 zookeeper -->  <dubbo:registry address=*"zookeeper://localhost:2181"* />  <!-- 用dubbo协议在20880端口暴露服务 -->  <dubbo:protocol name=*"dubbo"* port=*"20880"* />  <!--使用 dubbo 协议实现定义好的 api.PermissionService 接口 -->  <dubbo:service interface=*"com.itmayiedu.demo.api.service.DemoApiService"*  ref=*"demoService"* protocol=*"dubbo"* />  <!--具体实现该接口的 bean -->  <bean id=*"demoService"* class=*"com.itmayiedu.demo.api.service.impl.DemoApiServiceImpl"* />  </beans> |

### 启动发布服务

|  |
| --- |
| ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("provider.xml");  System.***out***.println(context.getDisplayName() + ": here");  context.start();  System.***out***.println("服务已经启动...");  System.***in***.read(); |

## 消费者调用服务接口

### Maven依赖信息

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.itmayiedu</groupId>  <artifactId>itmayiedu-dubbo-api</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>0.10</version>  </dependency>  <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>org.jboss.netty</groupId>  <artifactId>netty</artifactId>  <version>3.2.5.Final</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.3.9.RELEASE</version>  </dependency>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <version>2.5.3</version>  <exclusions>  <exclusion>  <groupId>org.springframework</groupId>  <artifactId>spring</artifactId>  </exclusion>  <exclusion>  <groupId>org.jboss.netty</groupId>  <artifactId>netty</artifactId>  </exclusion>  </exclusions>  </dependency>  </dependencies> |

### 调用服务接口

|  |
| --- |
| ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("consumer.xml");  context.start();  System.***out***.println("consumer start");  DemoApiService demoApiService = context.getBean(DemoApiService.**class**);  String result = demoApiService.getUser(1l);  System.***out***.println("result:" + result); |

### 消费者配置文件

# DubboAdmin部署

将dubbo-admin.zip放入到TomcatWebapps目录下，修改dubbo.properties中的Zookeeper连接地址即可。

# Dubbo支持协议

Dubbo支持dubbo、rmi、hessian、http、webservice、thrift、redis等多种协议，但是Dubbo官网是推荐我们使用Dubbo协议的。

# SpringBoot整合Dubbo

itmayiedu-dubbo-parent

--------------itmayiedu-dubbo-api

------------------------itmayiedu-dubbo-api-member

--------------itmayiedu-dubbo-api-member-impl

--------------itmayiedu-dubbo-api-order-impl

## itmayiedu-dubbo-parent

### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  <curator-framework.version>4.0.1</curator-framework.version>  <zookeeper.version>3.4.13</zookeeper.version>  <dubbo.starter.version>0.2.0</dubbo.starter.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>com.alibaba.boot</groupId>  <artifactId>dubbo-spring-boot-starter</artifactId>  <version>${dubbo.starter.version}</version>  </dependency>  <dependency>  <groupId>org.apache.curator</groupId>  <artifactId>curator-framework</artifactId>  <version>${curator-framework.version}</version>  </dependency>  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  <version>${zookeeper.version}</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build> |

## 生产者

### itmayiedu-dubbo-api-member-impl

#### Maven依赖信息

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.itmayiedu</groupId>  <artifactId>itmayiedu-dubbo-api-member</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies> |

#### application.yml

|  |
| --- |
| server:  port: 8080  dubbo:  application:  name: member  protocol:  name: dubbo  port: 20880  registry:  address: zookeeper://127.0.0.1:2181  scan:  base-packages: com.itmayiedu.api.member.impl |

#### 发布服务

|  |
| --- |
| @Service  **public** **class** MemberServiceImpl **implements** MemberService {  @Override  **public** String getUser() {  **return** "订单服务调用会员服务接口";  }  } |

注意:@Service 注解引入为阿里巴巴注解

#### 启动服务

|  |
| --- |
| @EnableDubbo  @SpringBootApplication  **public** **class** AppMember {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(AppMember.**class**, args);  }  } |

@EnableDubbo 开启Dubbo服务功能

## 消费者

### itmayiedu-dubbo-api-order-impl

#### Maven依赖信息

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.itmayiedu</groupId>  <artifactId>itmayiedu-dubbo-api-member</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies> |

#### application.yml

|  |
| --- |
| server:  port: 8081  ###dubbo 注册服务名称  dubbo:  application:  name: order  ###dubbo服务地址  protocol:  name: dubbo  port: 20880  registry:  address: zookeeper://localhost:2181  consumer:  timeout: 5000 |

#### 调用服务

|  |
| --- |
| @Reference  **private** MemberService memberService;  @RequestMapping("/orderToMember")  **public** String orderToMember() {  **return** memberService.getUser();  } |

使用该 @Reference调用dubbo接口

#### 启动项目

|  |
| --- |
| @EnableDubbo  @SpringBootApplication  **public** **class** AppOrder {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(AppOrder.**class**, args);  }  } |

# SpringCloud与Dubbo区别

为什么放弃Dubbo 使用SpringCloud？

相同点：SpringCloud 和Dubbo可以实现RPC远程调用框架，可以实现服务治理。

不同点:

SpringCloud是一套目前比较网站微服务框架了，整合了分布式常用解决方案遇到了问题注册中心Eureka、负载均衡器Ribbon ，客户端调用工具Rest和Feign，分布式配置中心Config，服务保护Hystrix，网关Zuul Gateway ，服务链路Zipkin，消息总线Bus等。

从架构上分析

Dubbo内部实现功能没有SpringCloud强大，只是实现服务治理，缺少分布式配置中心、网关、链路、总线等，如果需要用到这些组件，需要整合其他框架。

从更新迭代速度分析

Dubbo目前更新速度没有SpringCloud快，到SpringCloud2.0后SpringCloud会越来完善和稳定。

从开发背景角度分析

Dubbo的开发背景是阿里巴巴， 在中国也推出了非常多的优秀的开源框架

但是在SpringCloud的背景是Spring家族，Spring是专注于企业级开源框架开发，在中国，或者在整个世界上Spring框架都应用的非常广泛。所有相对来说SpringCloud的背景比Dubbo更加强大。

最后总结下：如果学习Dubbo的话，学习其他的分布式解决方案需要自己组装，反而如果学习SpringCloud，它已经把整个常用分布式解决都整合好了。