**概要：**

1. Dubbo 快速入门
2. Dubbo 常规配置说明

一、Dubbo 快速入门

**Dubbo核心功能解释**

dubbo 阿里开源的一个SOA服务治理框架，从目前来看把它称作是一个RPC远程调用框架更为贴切。单从RPC框架来说，功能较完善，支持多种传输和序列化方案。所以想必大家已经知道他的核心功能了：就是远程调用。



Dubbo服务端和消费端之间调用：

1. 通过注册中心。
2. 服务端和消费端直连。

|  |
| --- |
| <dubbo:reference interface="edu.facade.user.service.PmsUserFacade" id="pmsUserFacade" timeout="2000" check="false" url="dubbo://localhost:20880"/> |

1. 消费端直接调用提供方的服务。

如演示：

**快速演示Dubbo的远程调用**

实现步骤

* 创建服务端项目
  + 引入dubbo 依赖
  + 编写服务端代码
* 创建客户端项目
  + 引入dubbo 依赖
  + 编写客户端调用代码

dubbo 引入：

<dependency>

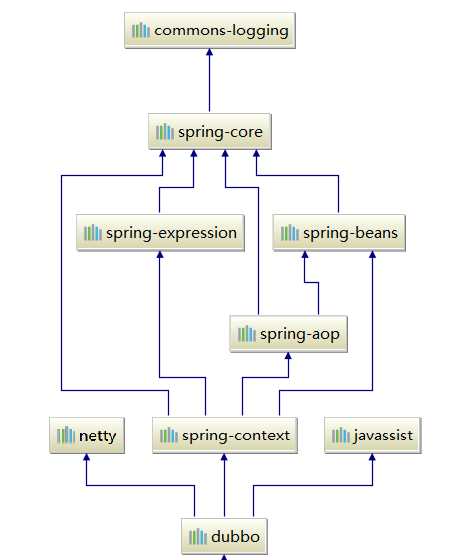
<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>2.6.2</version>

</dependency>

dubbo 默认依懒：



引用dubbo，默认还引用了netty（网络通讯框架）、spring-context、javassist（代理）

客户端代码：

|  |
| --- |
| 直接调用服务地址。 |

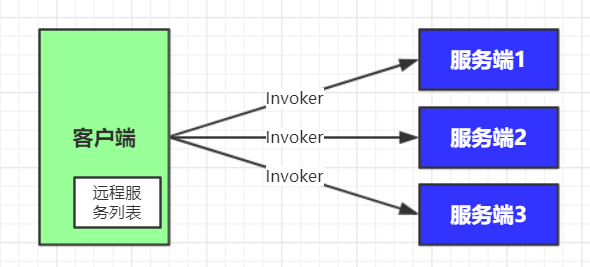
服务端代码：

|  |
| --- |
| 一个接口服务对应一个serviceConfig |

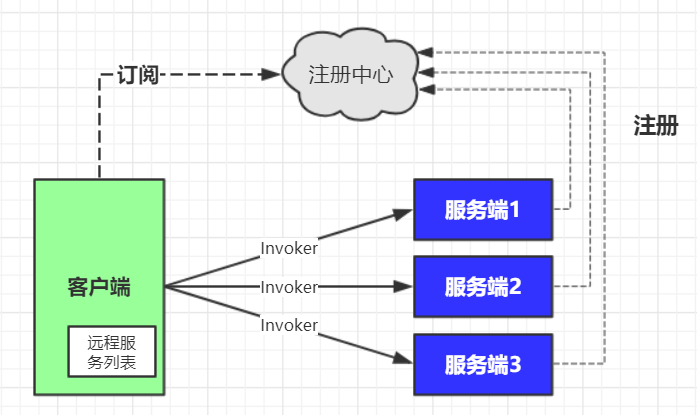
**基于Dubbo实现服务集群：**

在上一个例子中如多个服务的集群？即当有多个服务同时提供的时候，客户端该调用哪个？以什么方式进行调用以实现负载均衡？

一个简单的办法是将多个服务的URL同时设置到客户端并初始化对应的服务实例，然后以轮询的方式进行调用。



但如果访问增大，需要扩容服务器数量，那么就必须增加配置重启客户端实例，重写代码。显然这不是我们愿意看到的。Dubbo引入了服务注册中的概念，可以解决动态扩容的问题。



演示基于注册中心实现服集群：

* 修改服务端代码，添加multicast 注册中心。
* 修改客户端代码，添加multicast 注册中心。
* 观察 多个服务时，客户端如何调用。
* 观察 动态增减服务，客户端的调用。

# 服务端连接注册中心

serviceConfig.setRegistry(new RegistryConfig("multicast://224.1.1.1:2222"));

# 客户端连接注册中心

referenceConfig.setRegistry(new RegistryConfig("multicast://224.1.1.1:2222"));

#查看 基于UDP 占用的2222 端口

netstat -ano|findstr 2222

**基于spring IOC维护Dubbo 实例**

在前面两个例子中 出现了,ApplicationConfig、ReferenceConfig、RegistryConfig、com.alibaba.dubbo.config.ServiceConfig等实例 ，很显然不需要每次调用的时候都去创建该实例那就需要一个IOC 容器去管理这些实例，spring 是一个很好的选择。

**提供者配置----------------------------------**

<?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"

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">

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name="simple-app" />

<!-- 使用multicast广播注册中心暴露服务地址 -->

<dubbo:registry address="multicast://224.5.6.7:1234" />

<!-- 用dubbo协议在20880端口暴露服务 -->

<dubbo:protocol name="dubbo" port="20880" />

<!-- 声明需要暴露的服务接口 -->

<dubbo:service interface="tuling.dubbo.server.UserService" ref="userService" />

<!-- 和本地bean一样实现服务 -->

<bean id="userService" class="tuling.dubbo.server.impl.UserServiceImpl" />

</beans>

提供者服务暴露代码：

ApplicationContext context = new ClassPathXmlApplicationContext("/spring-provide.xml");

((ClassPathXmlApplicationContext) context).start();

System.in.read();

**消费者配置---------------------------------------**

<?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"

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">

<dubbo:application name="young-app"/>

<dubbo:registry address="multicast://224.5.6.7:1234"/>

<dubbo:reference id="userService" interface="tuling.dubbo.server.UserService"/>

</beans>

消费者调用代码：

ApplicationContext context = new ClassPathXmlApplicationContext("/spring-consumer.xml");

UserService userService = context.getBean(UserService.class);

UserVo u = userService.getUser(1111);

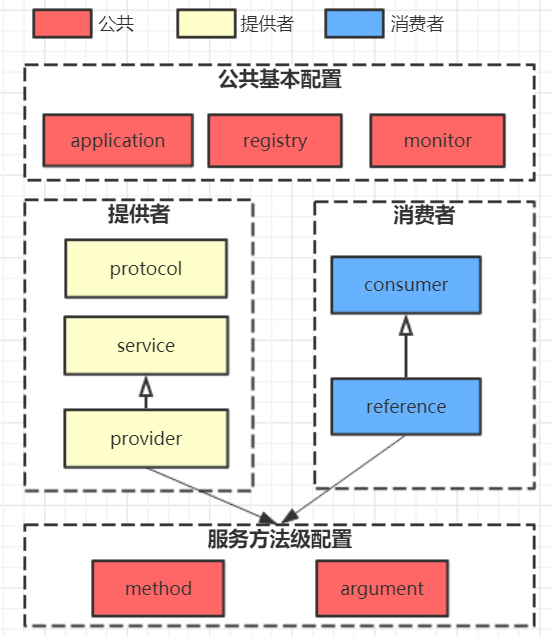
System.out.println(u);

二、Dubbo常规配置说明

**Dubbo配置的整体说明：**

|  |  |  |
| --- | --- | --- |
| **标签** | **用途** | **解释** |
| <dubbo:application/> | 公共 | 用于配置当前应用信息，不管该应用是提供者还是消费者 |
| <dubbo:registry/> | 公共 | 用于配置连接注册中心相关信息 |
| <dubbo:protocol/> | 服务 | 用于配置提供服务的协议信息，协议由提供方指定，消费方被动接受 |
| <dubbo:service/> | 服务 | 用于暴露一个服务，定义服务的元信息，一个服务可以用多个协议暴露，一个服务也可以注册到多个注册中心 |
| <dubbo:provider/> | 服务 | 当 ProtocolConfig 和 ServiceConfig 某属性没有配置时，采用此缺省值，可选 |
| <dubbo:consumer/> | 引用 | 当 ReferenceConfig 某属性没有配置时，采用此缺省值，可选 |
| <dubbo:reference/> | 引用 | 用于创建一个远程服务代理，一个引用可以指向多个注册中心 |
| <dubbo:method/> | 公共 | 用于 ServiceConfig 和 ReferenceConfig 指定方法级的配置信息 |
| <dubbo:argument/> | 公共 | 用于指定方法参数配置 |

配置关系图：



**配置分类**

所有配置项分为三大类。

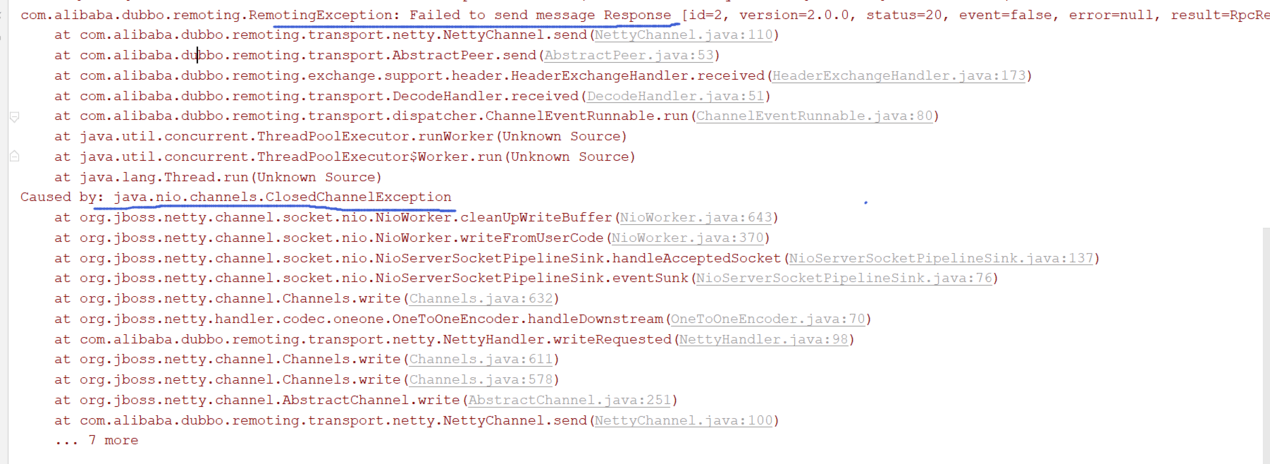
1. 服务发现：表示该配置项用于服务的注册与发现，目的是让消费方找到提供方。
2. 服务治理：表示该配置项用于治理服务间的关系，或为开发测试提供便利条件。
3. 性能调优：表示该配置项用于调优性能，不同的选项对性能会产生影响。

**dubbo 配置的一些套路:**

先来看一个简单配置

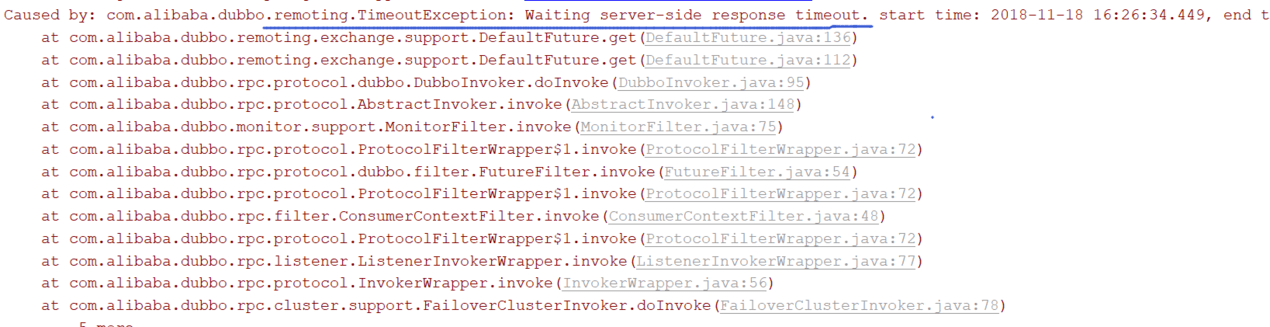
<dubbo:service interface="tuling.dubbo.server.UserService"  **timeout**="2000">

通过字面了解 timeout即服务的执行超时时间。但当服务执行真正超时的时候 报的错跟timeout并没有半毛钱的关系，其异常堆栈如下：



可以看到错误表达的意思是 因为Channel 关闭导致 无法返回 Response 消息。

出现这情况的原因在于 虽然timeout 配置在服务端去是用在客户端，其表示的是客户端调用超时间，而非服务端方法的执行超时。当我们去看客户端的日志时候就能看到timeout异常了



类似这种配在服务端用在客户端的配置还有很多，如retries/riː'traɪ/(重试次数)、async/əˈsɪŋk/（是否异步）、loadbalance(负载均衡)。。。等。

**套路一：***服务端配置客户端来使用*。

注：其参数传递机制是 服务端所有配置都会封装到URL参数，在通过注册中心传递到客户端

如果需要暴露多个服务的时候，每个服务都要设置其超时时间，貌似有点繁琐。Dubbo中可以通过 <dubbo:provider> 来实现服务端缺省配置。它可以同时为 <dubbo:service> 和 <dubbo:protocol> 两个标签提供缺省配置。如：

#相当于每个服务提供者设置了超时时间 和重试次数

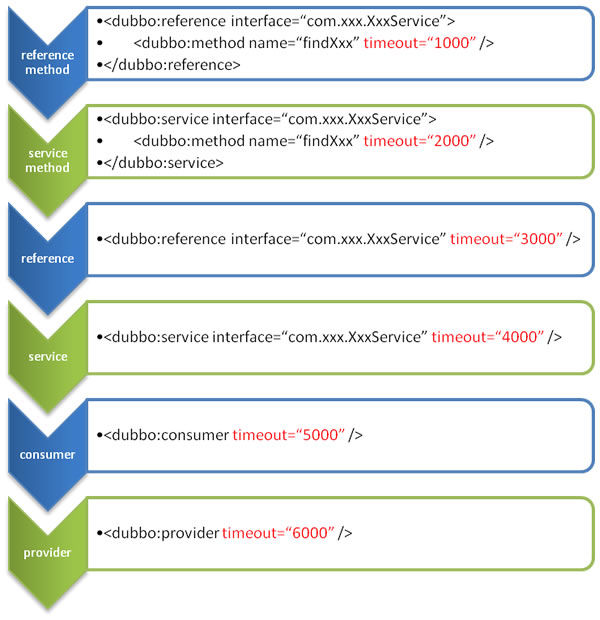
<dubbo:provider timeout="2000" retries="2"></dubbo:provider>

同样客户端也有缺省配置标签：<dubbo:consumer>，这些缺省设置可以配置多个 通过 <dubbo:service provider="providerId"> ,如果没指定就用第一个。

、

**套路二**：<dubbo:provider>与<dubbo:service> ，<dubbo:consumer>与<dubbo:reference>傻傻分不清楚

在服务端配置timeout 之后 所有客户端都会采用该方超时时间，其客户端可以自定义超时时间吗？通过  <dubbo:reference timeout="2000"> 可以设定或者在<dubbo:consumer timeout="2000"> 也可以设定 甚至可以设定到方法级别 <dubbo:method name="getUser" timeout="2000"/>。加上服务端的配置，超时总共有6处可以配置。如果6处都配置了不同的值，最后肯定只会有一个超时值生效，其优先级如下：



小提示：通过DefaultFuture的get 方法就可观测到实际的超时设置。

com.alibaba.dubbo.remoting.exchange.support.DefaultFuture

**套路三：**同一属性到处配置，优先级要小心。

**一般建议配置示例：**

提供端：---------------------------

 <dubbo:application name="demo-provider"/>

<dubbo:registry protocol="redis" address="192.168.0.147:6379" check="true"/>

<dubbo:protocol name="dubbo" port="20880"/>

<dubbo:provider group="tuling.dubbo.demo"

                threadpool="fixed"

                threads="500"

                timeout="5000"

                retries="2"