**Dubbo 企业级应用进阶**

## **课程概要：**

1. 分布式项目开发与联调
2. Dubbo控制管理后台使用
3. Dubbo注册中心选用

## **分布式项目开发与联调**

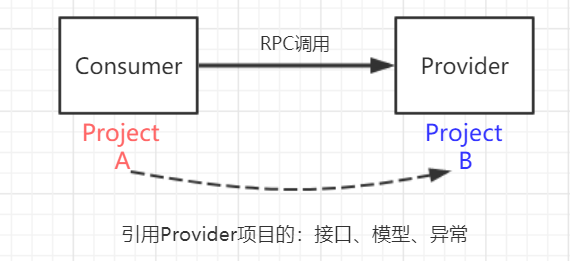
### **接口暴露与引用**

在一个RPC场景中 ，调用方是通过接口来调用服务端，传入参数并获得返回结果。这样服务端的接口和模型必须暴露给调用方项目。服务端如何暴露呢？客户端如何引用呢？

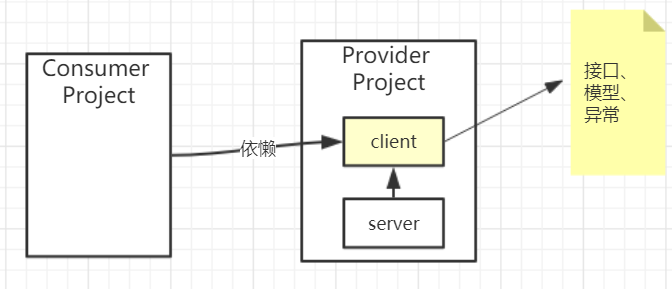
**接口信息**

**、模型信息**

**、异常**

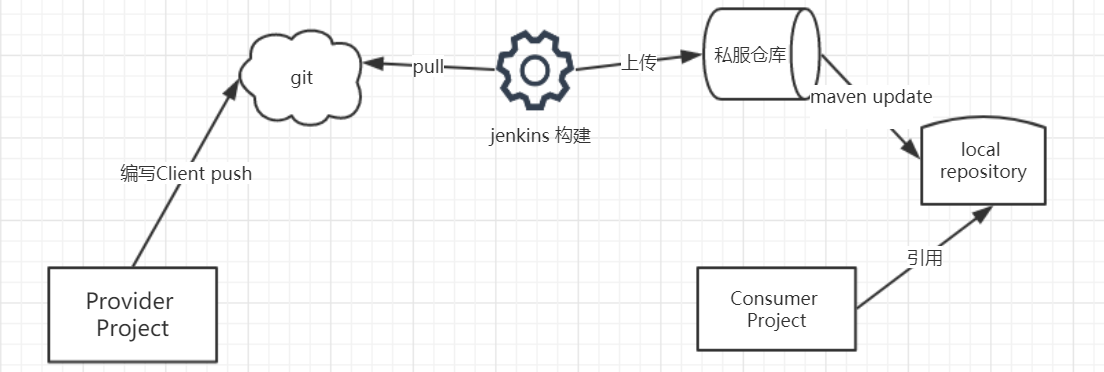


暴露接口的通常做法是 接口与实现分离，服务端将 接口、模型、异常 等统一放置于一个模块，实现置于另一个模块。调用方通过Maven进行引用。



**自动化构建与协作**

当项目越来越多，服务依懒关系越发复杂的时候，为了提高协作效率，必须采用自动化工具 完成 接口从编写到构建成JAR包，最后到引用的整个过程。

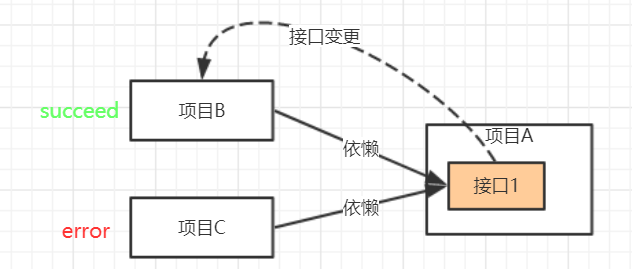


流程描述：

1. 服务提供者项目发人员编写Client 接口
2. push 至远程仓库
3. jenkins 构建指定版本
4. jenkins Deploye 至私服仓库 nexus
5. 服务消费者项目开发人员基于maven 从私服务仓库下载

**接口平滑升级：**

在项目迭代过程当中， 经常会有多个项目依懒同一个接口，如下图 项目B、C都依懒了项目A当中的接口1，此时项目B业务需要，需要接口1多增加一个参数，升级完成后。项目B能正确构建上线，项目C却不行。



解决办法与原则：

1. 接口要做到向下兼容：接口参数尽量以对象形式进行封装。Model属性只增不删，如果需要作废，可以添加@Deprecated 标识。
2. 如果出现了不可兼容的变更，则必须通知调用方整改，并制定上线计划。

**开发联调：**

在项目开发过程当中，一个开发或测试环境的注册中心很有可能会同时承载着多个服务，如果两组服务正在联调，如何保证调用的是目标服务呢？

**1、基于临时分组联调**

group 分组

在reference 和server 当中采用相同的临时组 ,通过group 进行设置

**2、直连提供者：**

在reference 中指定提供者的url即可做到直连

<**dubbo:reference url="dubbo://127.0.0.1:20880" id="demoService"**

**timeout="2000"**

**interface="com.xxx.service.DemoService" check="false"**/>

**3、只注册：**

一个项目有可能同是为即是服务提供者又消费者，在测试时需要调用某一服务同时又不希望正在开发的服务影响到其它订阅者如何实现？

通过修改 register=false 即可实现

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

## **Dubbo控制管理后台的使用**

**Dubbo 控制后台的安装：**

#从github 中下载dubbo 项目

git clone https://github.com/alibaba/dubbo.git

#更新项目

git fetch

#临时切换至 dubbo-2.5.8 版本

git checkout dubbo-2.5.8

#进入 dubbo-admin 目录

cd dubbo-admin

#mvn 构建admin war 包

mvn clean pakcage -DskipTests

#得到 dubbo-admin-2.5.8.war 即可直接部署至Tomcat

#**修改 dubbo.properties 配置文件**

dubbo.registry.address=zookeeper://127.0.0.1:2181

**控制后台基本功能介绍 ：**

服务查找：

服务关系查看:

服务权重调配：

服务路由：

## **Dubbo注册中心选用**

Dubbo所支持的注册中心：

**Redis：**

采用K/V 形式进行存储,基结构如下：

/dubbbo/**com.xxx.service.DemoService/providers**

dubbo://192.168.246.1:20880/com.xxx.service.UserService

dubbo://192.168.246.1:20880/com.xxx.service.UserService

/dubbbo/com.xxx.service.DemoService/consumers

dubbo://192.168.246.1:20880/com.xxx.service.DemoService

/dubbbo/com.xxx.service.DemoService/routers

/dubbbo/com.xxx.service.DemoService/configurators

并基于 Redis 的 Publish/Subscribe 事件通知数据变更，其模型如下：

订阅：



推送变更：

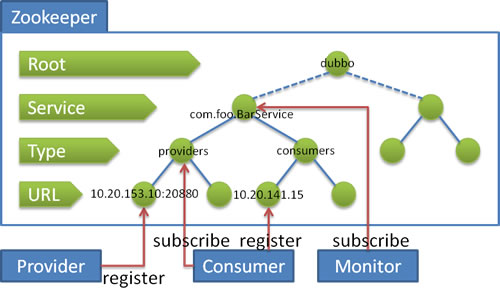


调用过程：

1. 服务提供方启动时，向 Key:/dubbo/XXXServer/providers 下，添加当前提供者的地址
2. 并向 Channel:/dubbo/com.foo.BarService/providers 发送 register 事件
3. 服务消费方启动时，从 Channel:/dubbo/com.foo.BarService/providers 订阅 register 和 unregister 事件
4. 并向 Key:/dubbo/com.foo.BarService/providers 下，添加当前消费者的地址
5. 服务消费方收到 register 和 unregister 事件后，从 Key:/dubbo/com.foo.BarService/providers 下获取提供者地址列表

**Zookeper：**

是一个树型的目录服务，支持变更推送，适合作为 Dubbo 服务的注册中心，工业强度较高，可用于生产环境，并推荐使用



流程说明:

* 服务提供者启动时: 向 /dubbo/com.foo.BarService/providers 目录下写入自己的 URL 地址
* 服务消费者启动时: 订阅 /dubbo/com.foo.BarService/providers 目录下的提供者 URL 地址。并向 /dubbo/com.foo.BarService/consumers 目录下写入自己的 URL 地址
* 监控中心启动时: 订阅 /dubbo/com.foo.BarService 目录下的所有提供者和消费者 URL 地址

## **下周安排**

Dubbo性能调优：集群容错、负载算法、线程模型、并发控制、连接控制

Dubbo协议：Dubbo、RMI、Http

Dubbo 非典型使用场景：多协议、泛化调用、泛化引用、隐示传参、异步调用、令版验证、路由控制。

Dubbo 扩展点介绍与使用：

Dubbo核心模块的底层原理：

自定义设计一个RPC框架：