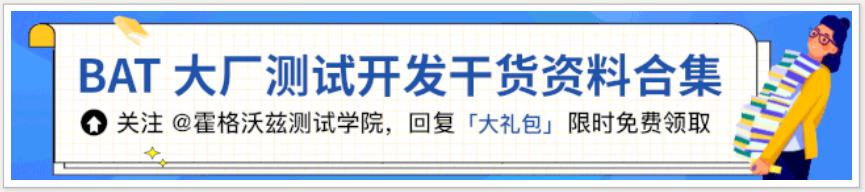
干货 | Dubbo 接口测试原理及多种方法实践总结

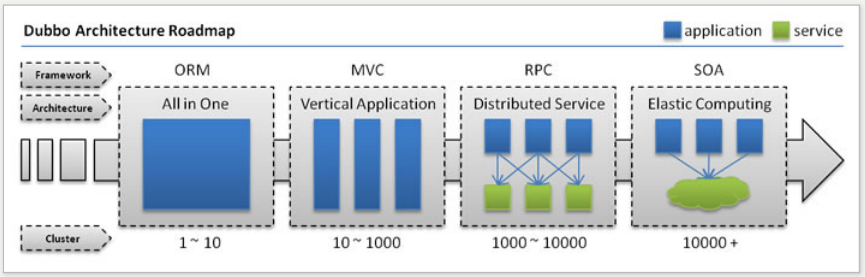


> 本文为霍格沃兹测试学院优秀学员学习笔记，**Java 中高级测试开发名企定向**进阶学习文末加群。

1、什么是 Dubbo?

Dubbo 最开始是应用于淘宝网，由阿里巴巴开源的一款优秀的高性能服务框架，由 Java 开发，后来贡献给了 Apache 开源基金会组织。

下面以官网的一个说明来了解一下架构的演变过程，从而了解 Dubbo 的诞生原因：



- **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。

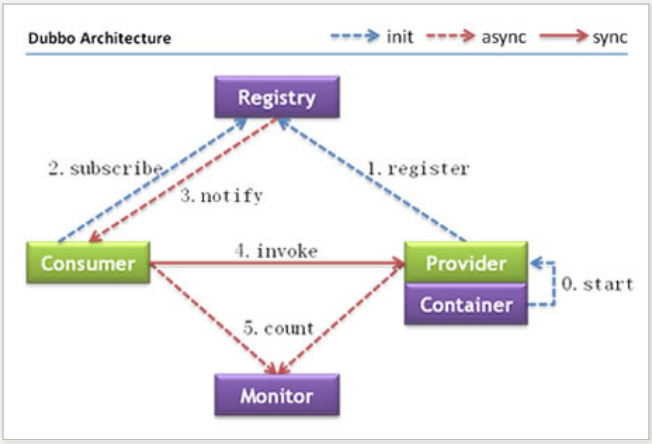
- **分布式服务架构**

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。

2、Dubbo 架构简介



节点	角色说明	实例对应
Provider	暴露服务的服务提供方	开发的代码
Consumer	调用远程服务的服务消费方	调用方代码、Telnet、Jmeter
Registry	服务注册与发现的注册中心	zookeeper
Monitor	统计服务的调用次数和调用时间的监控中心	dubbo-monitor
Container	服务运行容器	dubbo内置容器、Jetty

> Dubbo 比较有特点的地方就是这个注册中心，平常我们测试较多的 HTTP 接口，直接请求接口，调用后端服务即可；而 Dubbo 是要先走注册中心获取服务的位置，下面来举个现实生活中的例子来说明。

现实举例

好比大家平常约朋友一起出去吃饭，听说川菜馆 “赠李白” 不错，然后需要找这家饭店在哪（用小蓝或小黄 App），知道了具体的地址才出发，至于是走路，打车还是骑车，就随意了。

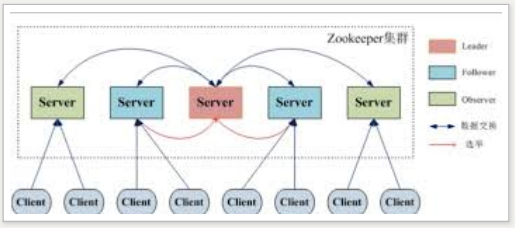
这里 App 就相当于注册中心（`Registry`），我们这群吃货就是消费者（`Consumer`），商家属于生产者（`Provider`）。商家把自己的信息注册在 App 上，消费者根据 App 查询到商家的信息，再根据信息找到商家进行消费。

2.1、Zookeeper 简介

之前经常有小伙伴问我 zk 干啥的？怎么用？下面就来简单了解一哈：

ZK，全称就是 `zookeeper` ，是 Apache 软件基金会的一个软件项目，它为大型分布式计算提供开源的分布式配置服务、同步服务和命名注册。

下面的图示也可以清晰的说明 zk 的部署和工作的一些方式(具体的技术细节需要的话可以针对 zk 专门搜索学习)：



- **Leader**：集群工作的核心，事务请求的唯一调度和处理者，保证事务处理的顺序性。对于有写操作的请求，需统一转发给 Leader 处理。Leader 需决定编号执行操作。
- **Follower**：处理客户端非事务请求，转发事务请求转发给 Leader，参与 Leader 选举。
- **Observer 观察者**：进行非事务请求的独立处理, 对于事务请求, 则转发给 Leader 服务器进行处理. 不参与投票。

3、什么是 Dubbo 接口?

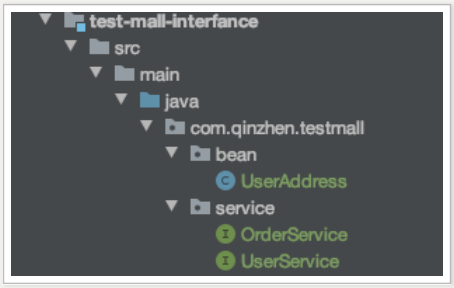
所谓的 Dubbo 接口，其实就是一个个 Dubbo 服务中的方法，而测试 Dubbo 接口就相当于我们测试人员充当消费者或者创造消费者去 "消费" 这个方法。

具体的方式有很多，代码、工具、命令皆可，在接下来的内容中来一一演示。

4、Dubbo 接口测试 (创造消费者)

> 以下我将以本地的一个简单的 Dubbo 服务 demo 为例，演示 Dubbo 测试的各种方法。

`interface` 只有两个，分别是 `OrderService` 和 `UserService`



`OrderService`：

```
package com.qinzhen.testmall.service;

import com.qinzhen.testmall.bean.UserAddress;
import java.util.List;

public interface OrderService {

    /**
     * 初始化订单
     * @param userID
     */
    public List<UserAddress> initOrder(String userID);
}
```

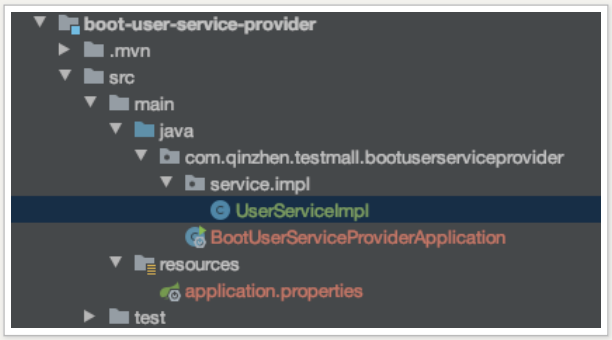
`UserService`：

```
package com.qinzhen.testmall.service;        import com.qinzhen.testm
```

JavaBean 对象 `UserAddress` 如下：

```
package com.qinzhen.testmall.bean;        import lombok.AllArgsConstr
```

创建一个 `provider` 来实现 `UserService` 的 `Interface`：



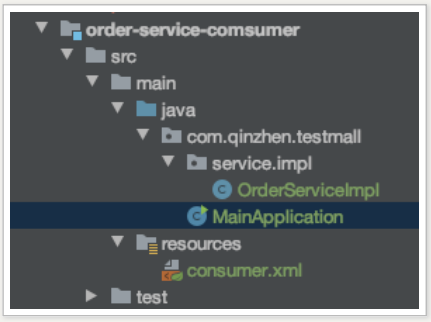
实现方法中，根据 id 返回对应的用户地址信息即可：

```
package com.qinzhen.testmall.bootuserserviceprovider.service.impl;
```

4.1 Java consumer 代码

> 下面我们编写 `consumer` 代码，让服务消费者去注册中心订阅服务提供者的服务地址，以 `RPC` 方式，获取远程服务代理，从而执行远程方法，代码也很简单，如下：

- **代码结构：**



实现场景就是实现 `OrderService` 中的 `initOrder()` 方法，初始化订单，初始化中直接调用 `userService` 的 `getUserAddressLis(java.lang.String)` 方法，具体代码如下：

```
package com.qinzhen.testmall.service.impl;        import com.qinzhen.
```

- **consumer MainApplication**

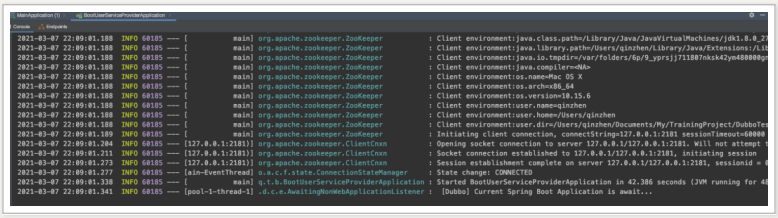
```
package com.qinzhen.testmall;        import com.qinzhen.testmall.bean
```

- **consumer.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>    <beans xmlns="http://www.sp
```

- **实例演示**:

首先确保 `provider` 已启动:



运行 `consumer`，可以看到成功调用到 dubbo 方法，获取地址列表信息:

4.2 telnet+invoke

> 我们使用 telnet 命令可以直接访问对应的服务，但是前提是你需要知道服务对应的 ip + 端口。

如下配置文件，我们可以知道服务暴露在本地的 20880 端口

```
    dubbo.application.name=boot-user-service-provider
    dubbo.registry.address=127.0.0.1:2181
    dubbo.registry.protocol=zookeeper
    dubbo.protocol.name=dubbo
    dubbo.protocol.port=20880
```

使用 telnet 命令进行访问，如下出现 Dubbo 字样时说明连接成功:

```
% telnet localhost 20880    Trying ::1...    Connected to localhost.
```

Dubbo 内建的 telnet 命令的说明和用法如下

- **ls**
  - `ls`：显示服务列表
  - `ls -l`：显示服务详细信息列表
  - `ls XxxService`：显示服务的方法列表
  - `ls -l XxxService`：显示服务的方法详细信息列表

```
dubbo>ls    com.qinzhen.testmall.service.UserService        dubbo>ls
```

- **invoke**
  - `invoke XxxService.xxxMethod(1234, "abcd", {"prop" : "value"})`：调用服务的方法
  - `invoke com.xxx.XxxService.XxxService.xxxMethod(1234, "abcd", {"prop" : "value"})`：调用全路径服务的方法
  - `invoke xxxMethod(1234, "abcd", {"prop" : "value"})`：调用服务的方法 (自动查找包含此方法的服务)
  - `invoke xxxMethod({"name":"zhangsan","age":12,"class":"org.apache.dubbo.qos.legacy.service.Person"})`：当有参数重载，或者类型转换失败的时候，可以通过增加 class 属性指定需要转换类
  - 当参数为 `Map<Integer,T>`， `key` 的类型为 `Integer` 时，建议指定类型。例如 `invoke com.xxx.xxxApiService({"3":0.123,"class":"java.util.HashMap"})`

然后我们使用 `invoke` 命令对 dubbo 方法 `getUserAddressList()` 进行调用，如下：

```
dubbo>invoke getUserAddressList()    [{"consignee":"qz","id":1,"isDef
```

学习链接：

其他 Telnet 命令相关操作，需要可参考 Dubbo 官网：

- https://dubbo.apache.org/zh/docs/v2.7/user/references/telnet

4.3 JMeter

对于 JMeter 测试 Dubbo 接口的方法，可参考往期文章：

《基于 Jmeter 完成 Dubbo 接口的测试》

4.4 Dubbo-admin

对于 Dubbo-admin 的安装调试，可参考文章：

《dubbo-admin+zookeeper 的环境搭建实操与 Could not extract archive 报错踩坑》

- https://blog.csdn.net/weixin_43291944/article/details/103998883

4.5 泛化调用

> 测试 Dubbo 服务的时候，我们需要服务端的同学给我们提供 API，没有这个 API 我们是测不了的，而为了解决这个问题，Dubbo 官方又给我们提供了另外一个方法，就是泛化调用，来看看官方的解释：

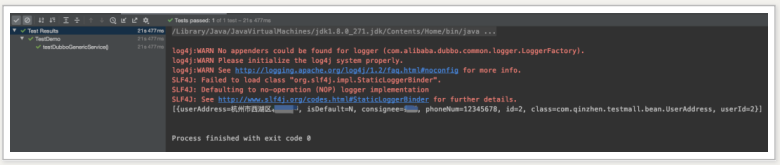
- **泛化调用的使用**

Dubbo 给我们提供了一个接口 `GenericService`，这个接口只有一个方法，就是 `$invoke`，它接受三个参数，分别为 `方法名`、`方法参数类型数组` 和 `参数值数组`；

下面我们直接上代码演示：

```
import com.alibaba.dubbo.config.ApplicationConfig;    import com.alib
```

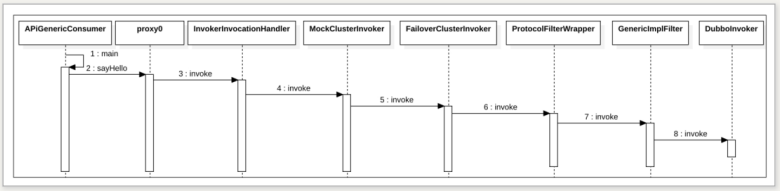
运行后我们来看看结果，咦~ 也成功访问了：



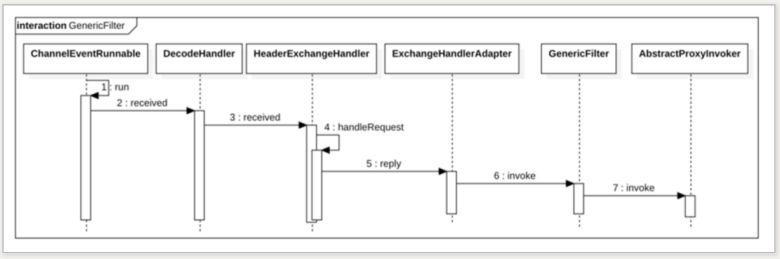
- **泛化调用的原理**

我们通过 debug 跟入 Dubbo 的源码中，可以得到如下的调用链：

服务消费端：

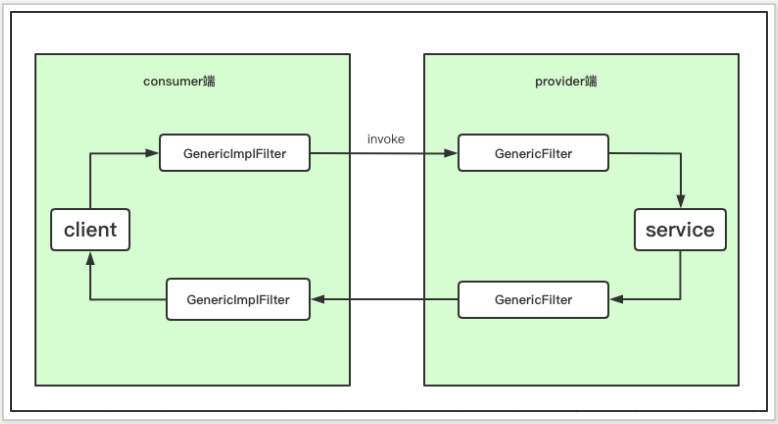


服务提供端：



> 从上面的调用链可以知道完成一次泛化调用，Dubbo 框架经历了很多过滤器，我们分别选取两端链路中的最后一步的 Filter 来简单了解一下泛化调用做了哪些事.

简化后的调用关系就如下：



先来看 `consumer` 端的 `GenericImplFilter` ，大概看下核心的处理步骤：

```
// 判断是否为泛化调用     if (invocation.getMethodName().equals(Constan
```

再来看 `provider` 端的 `GenericFilter` ，大概的核心处理步骤如下：

```
package com.alibaba.dubbo.rpc.filter;        import ...        /**
```

上面的代码很多，着重来提取一小段看一下：

```
Method method = ReflectUtils.findMethodByMethodSignature(invoker.getI
    Class<?>[] params = method.getParameterTypes();
```

从上面的代码中我们便可以得知原来泛化调用中使用了 Java 的反射技术来获取对应的方法信息完成调用的

4.6 用 Python 来测试 Dubbo

> 我们知道 Dubbo 是个 Java 项目，测试 Dubbo 就是模拟消费者去调用 Dubbo 的 Java 方法，那显而易见，用 Python 是肯定没法去直接调用 Java 的，但是在日常的工作中，很多小伙伴可能是 Pyhton 技术栈的，或者因为一些测试条件限制亦或历史原因，必须要将 Dubbo 测试用 Python 实现以满足各种接口测试的一个组合。

- **1. python-hessian 库**

`Dubbo` 是支持 `hessian+http` 协议调用的， `hessian` 是一种二进制序列化的方式。

了解到可以通过这种方式实现，具体没有尝试过，还需要开发在项目中将序列化的方式改为 `hessian` ，并且需要知道 URL，有兴趣的小伙伴可以去了解一下。

- **2. telnetlib 库**

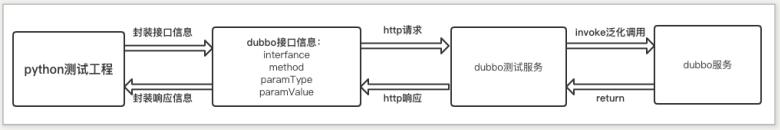
`telnetlib` 是 Python3 自带的一个库，可以调用 `telnet` 命令，其实也就相当于上面说到的使用 `telnet` 方式访问 `dubbo` 的方法

- **3. 开发 dubbo 测试服务**

我们可以使用 Java 来开发一个 Dubbo 测试的 Web 服务，实现上就可以使用 Dubbo 的泛化调用，然后我们再用 HTTP 访问的形式去访问这个服务，将我们的测试参数信息传过去，剩下的就交给 Java 去处理就好了。

这样经过封装设计后，可以实现 Python 端的使用者在访问 Dubbo 时就像在测试 `HTTP` 接口一样 (例如 Python 的 `request` 库)；另外服务的 IP、端口、注册中心等信息都不用出现在测试的工程中，只需要用环境标签做区分，在服务端进行请求转发即可，也保证了一定安全性。

大体上的思路流程如下：



以上，期待与大家多交流学习。

推荐学习

Java 中高级测试开发名企定向培养计划（30+ 企业级项目实战，对标阿里 P6-P7，年薪 50W+）！内推 BAT 大厂测试经理，成功入职返学费！

点击“阅读原文”，提升测试核心竞争力！

全文完