[最近项目用到Dubbo框架，临时抱佛脚分享一下共探讨。](http://www.cnblogs.com/Javame/p/3632473.html)

**1. Dubbo是什么？**

Dubbo是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。简单的说，dubbo就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有dubbo这样的分布式服务框架的需求，并且本质上是个服务调用的东东，**说白了就是个远程服务调用的分布式框架（告别Web Service模式中的WSdl，以服务者与消费者的方式在dubbo上注册）**  
其核心部分包含:  
1. 远程通讯: 提供对多种基于长连接的NIO框架抽象封装，包括多种线程模型，序列化，以及“请求-响应”模式的信息交换方式。  
2. 集群容错: 提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持。  
3. 自动发现: 基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。

[基于SOA架构的TDD测试驱动开发模式](http://www.cnblogs.com/Javame/p/5833885.html)

[服务治理要先于SOA](http://www.cnblogs.com/Javame/p/5817905.html)

[简述我的SOA服务治理](http://www.cnblogs.com/Javame/p/5811043.html)

[从页面走向单元实现真正的业务驱动](http://www.cnblogs.com/Javame/p/5868170.html)

**2. Dubbo能做什么？**

1.透明化的远程方法调用，就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。        
2.软负载均衡及容错机制，可在内网替代F5等硬件负载均衡器，降低成本，减少单点。  
3. 服务自动注册与发现，不再需要写死服务提供方地址，注册中心基于接口名查询服务提供者的IP地址，并且能够平滑添加或删除服务提供者。  
  
Dubbo采用全Spring配置方式，透明化接入应用，对应用没有任何API侵入，只需用Spring加载Dubbo的配置即可，Dubbo基于Spring的Schema扩展进行加载。

**之前使用Web Service，我想测试接口可以通过模拟消息的方式通过soapui或LR进行功能测试或性能测试。但现在使用Dubbo，接口之间不能直接交互，我尝试通过模拟消费者地址测试，结果不堪入目，再而使用jmeter通过junit进行测试，但还是需要往dubbo上去注册，如果再不给提供源代码的前提下，这个测试用例不好写啊....**

**3. dubbo的架构**

dubbo架构图如下所示：



**节点角色说明：**

       Provider: 暴露服务的服务提供方。

       Consumer: 调用远程服务的服务消费方。

       Registry: 服务注册与发现的注册中心。

       Monitor: 统计服务的调用次调和调用时间的监控中心。

       Container: 服务运行容器。

**这点我觉得非常好，角色分明，可以根据每个节点角色的状态来确定该服务是否正常。**

**调用关系说明：**

0 服务容器负责启动，加载，运行服务提供者。

**1. 服务提供者在启动时，向注册中心注册自己提供的服务。**

**2. 服务消费者在启动时，向注册中心订阅自己所需的服务。**

**3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。**

**4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。**

**5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。**

**dubbo的容错性显而易见，性能方面还没有还得及测，我们系统某页面需要掉5次接口，本来想建议做个缓存，但业务关系不能采纳，还需要研究下dubbo的性能调优问题...**

**4. dubbo使用方法。**

Dubbo采用全Spring配置方式，透明化接入应用，对应用没有任何API侵入，只需用Spring加载Dubbo的配置即可，Dubbo基于Spring的Schema扩展进行加载。如果不想使用Spring配置，而希望通过API的方式进行调用（不推荐）

下面我们就来看看spring配置方式的写法:

**服务提供者：**

1. 下载zookeeper注册中心，下载地址：[http://www.apache.org/dyn/closer.cgi/zookeeper/](http://www.apache.org/dyn/closer.cgi/zookeeper/" \t "_blank)  下载后解压即可，进入D:\apach-zookeeper-3.4.5\bin，

双击zkServer.cmd启动注册中心服务。

**2. 定义服务接口: (该接口需单独打包，在服务提供方和消费方共享)**

**下面这个例子不错，写的很详细可以做个model.**

1. package com.unj.dubbotest.provider;
3. import java.util.List;
5. public interface DemoService {
7. String sayHello(String name);
9. public List getUsers();
11. }

在服务提供方实现接口：(对服务消费方隐藏实现)

1. package com.unj.dubbotest.provider;
3. import java.util.ArrayList;
4. import java.util.LinkedList;
5. import java.util.List;

8. public class DemoServiceImpl implements DemoService{
10. public String sayHello(String name) {
11. return "Hello " + name;
12. }
13. public List getUsers() {
14. List list = new ArrayList();
15. User u1 = new User();
16. u1.setName("jack");
17. u1.setAge(20);
18. u1.setSex("男");
20. User u2 = new User();
21. u2.setName("tom");
22. u2.setAge(21);
23. u2.setSex("女");
25. User u3 = new User();
26. u3.setName("rose");
27. u3.setAge(19);
28. u3.setSex("女");
30. list.add(u1);
31. list.add(u2);
32. list.add(u3);
33. return list;
34. }
35. }

用Spring配置声明暴露服务：

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
5. xsi:schemaLocation="http://www.springframework.org/schema/beans
6. http://www.springframework.org/schema/beans/spring-beans.xsd
7. http://code.alibabatech.com/schema/dubbo
8. http://code.alibabatech.com/schema/dubbo/dubbo.xsd
9. ">
11. <!-- 具体的实现bean -->
12. <bean id="demoService" class="com.unj.dubbotest.provider.DemoServiceImpl" />
14. <!-- 提供方应用信息，用于计算依赖关系 -->
15. <dubbo:application name="xixi\_provider"  />
17. <!-- 使用multicast广播注册中心暴露服务地址
18. <dubbo:registry address="multicast://224.5.6.7:1234" />-->
20. <!-- 使用zookeeper注册中心暴露服务地址 -->
21. <dubbo:registry address="zookeeper://127.0.0.1:2181" />
23. <!-- 用dubbo协议在20880端口暴露服务 -->
24. <dubbo:protocol name="dubbo" port="20880" />
26. <!-- 声明需要暴露的服务接口 -->
27. <dubbo:service interface="com.unj.dubbotest.provider.DemoService" ref="demoService" />
29. </beans>

加载Spring配置，启动服务：

1. package com.unj.dubbotest.provider;
3. import org.springframework.context.support.ClassPathXmlApplicationContext;
5. public class Provider {
7. public static void main(String[] args) throws Exception {
8. ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(new String[] {"applicationContext.xml"});
9. context.start();
11. System.in.read(); // 为保证服务一直开着，利用输入流的阻塞来模拟
12. }
14. }

**服务消费者：**

**applicationContext-dubbo.xml 中注册自己需要调用的接口，我刚开始测试的时候需要的接口很多，所以把这个文件写的满满的，后来熟悉了把接口按业务类型分开，写了N多个 applicationContext-dubbo-\*\*\*.xml 简练多了 》。**

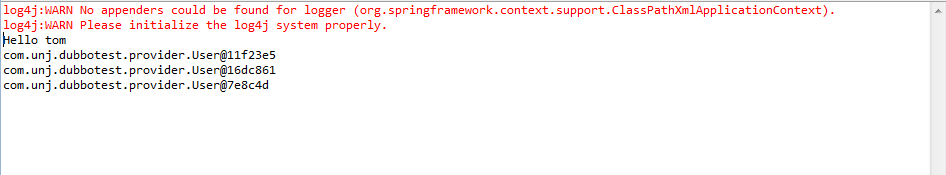
1.通过Spring配置引用远程服务：

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans
5. http://www.springframework.org/schema/beans/spring-beans.xsd
6. http://code.alibabatech.com/schema/dubbo
7. http://code.alibabatech.com/schema/dubbo/dubbo.xsd
8. ">
10. <!-- 消费方应用名，用于计算依赖关系，不是匹配条件，不要与提供方一样 -->
11. <dubbo:application name="hehe\_consumer" />
13. <!-- 使用zookeeper注册中心暴露服务地址 -->
14. <!-- <dubbo:registry address="multicast://224.5.6.7:1234" /> -->
15. <dubbo:registry address="zookeeper://127.0.0.1:2181" />
17. <!-- 生成远程服务代理，可以像使用本地bean一样使用demoService -->
18. <dubbo:reference id="demoService"
19. interface="com.unj.dubbotest.provider.DemoService" />
21. </beans>

2.加载Spring配置，并调用远程服务：

1. package com.alibaba.dubbo.demo.pp;
3. import java.util.List;
5. import org.springframework.context.support.ClassPathXmlApplicationContext;
7. import com.unj.dubbotest.provider.DemoService;
9. public class Consumer {
11. public static void main(String[] args) throws Exception {
12. ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(
13. new String[] { "applicationContext.xml" });
14. context.start();
16. DemoService demoService = (DemoService) context.getBean("demoService"); //
17. String hello = demoService.sayHello("tom"); // ִ
18. System.out.println(hello); //
20. //
21. List list = demoService.getUsers();
22. if (list != null && list.size() > 0) {
23. for (int i = 0; i < list.size(); i++) {
24. System.out.println(list.get(i));
25. }
26. }
27. // System.out.println(demoService.hehe());
28. System.in.read();
29. }
31. }

调用结果为：

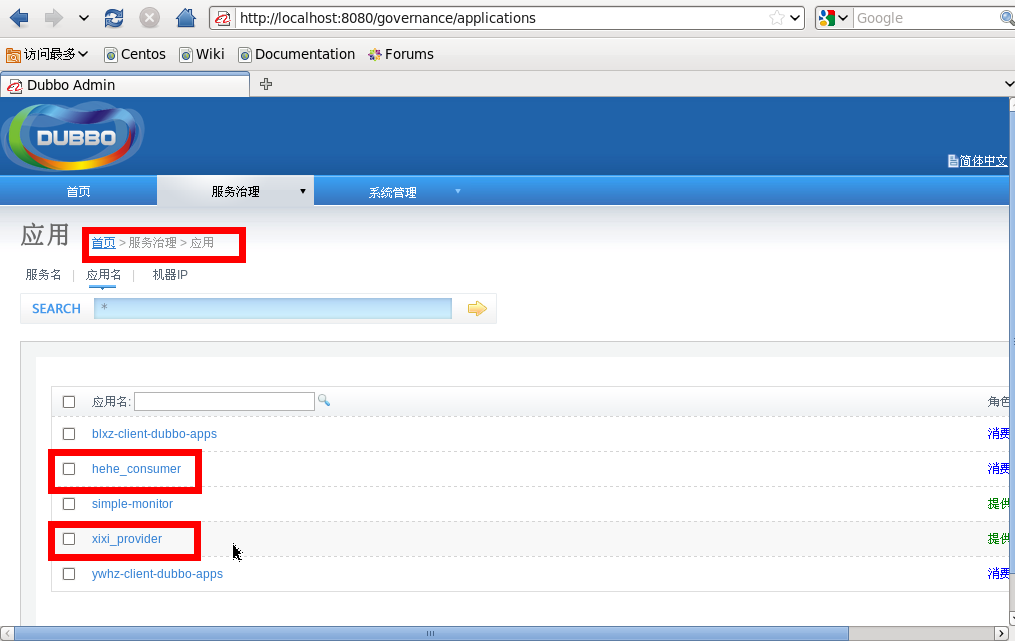


dubbo管理页面：

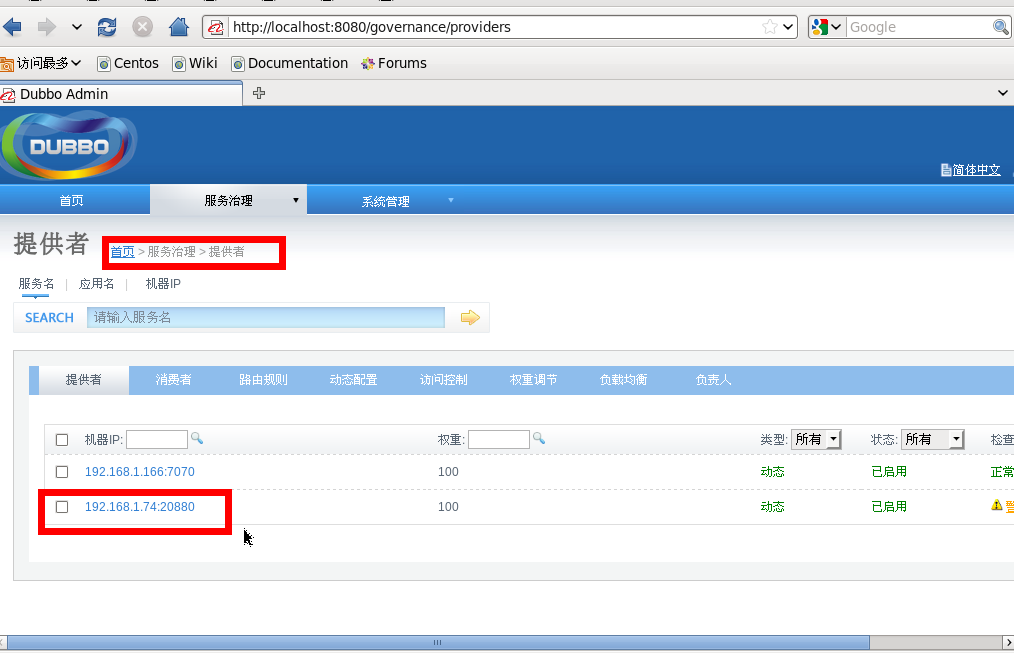
**这个管理页面还需要部署一个环境的，一开始我还以为是dubbo自带的，找了半天没有找到....**



应用页面：



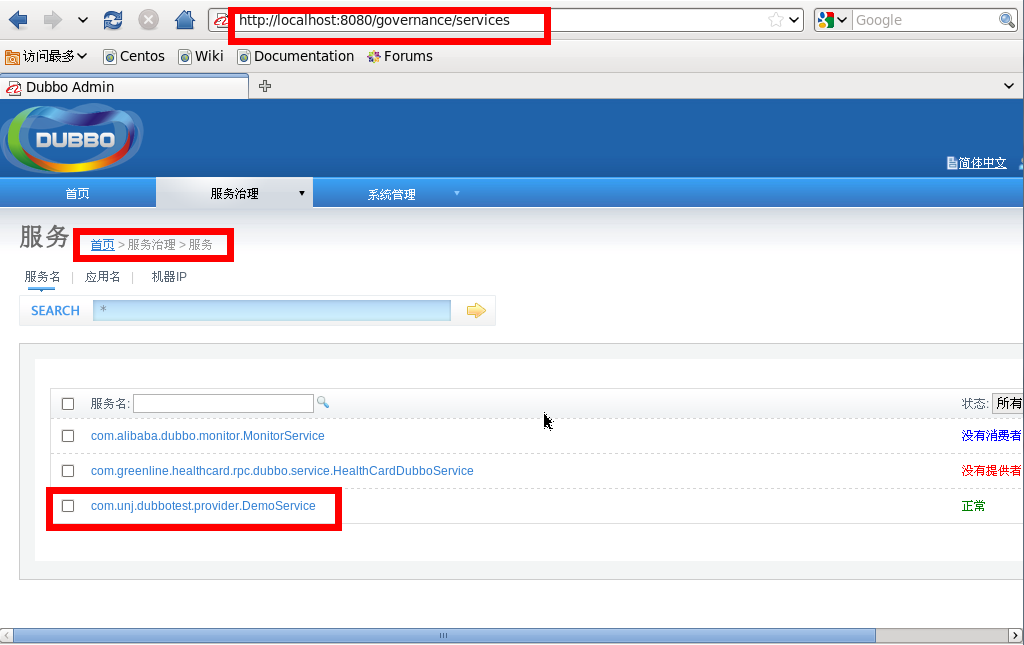
提供者页面:



消费者页面：



服务页面：



测试是否成功，我觉得只要看看状态是否正常，就ok了 ....

**感兴趣的M我微信：wonter**

案例代码下载：<http://download.csdn.net/detail/yiyu1/7116319>

很多朋友都在问这个管理界面哪来的？他其实是另外一个项目工程，需要下载后部署到web服务器。

dubbo管理台下载：<http://download.csdn.net/detail/yiyu1/9650594>