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。

## 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。

**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连接方式

使用Dubbo进行远程调用实现服务交互，它支持多种协议，如Hessian、HTTP、RMI、Memcached、Redis等等。由于Dubbo将这些协议的实现进行了封装了，无论是服务端（开发服务）还是客户端（调用服务），都不需要关心协议的细节，只需要在配置中指定使用的协议即可，从而保证了服务提供方与服务消费方之间的透明。

Dubbo的客户端和服务端有三种连接方式，分别是：广播，直连和使用zookeeper注册中心

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>**

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>**

**springboot整合dubbo**

#### 一、服务端开发

###### 1、添加依赖

<dependency>

<groupId>com.alibaba.spring.boot</groupId>

<artifactId>dubbo-spring-boot-starter</artifactId>

<version>2.0.0</version>

</dependency>

###### 2、在application.properties添加dubbo的相关配置信息

# Spring boot application

spring.application.name = dubbo-provider-demo

server.port = 9090

management.port = 9091

# Base packages to scan Dubbo Components (e.g., @Service, @Reference)

dubbo.scan.basePackages = com.alibaba.boot.dubbo.demo.provider.service

# Dubbo Config properties

## ApplicationConfig Bean

dubbo.application.id = dubbo-provider-demo

dubbo.application.name = dubbo-provider-demo

## ProtocolConfig Bean

dubbo.protocol.id = dubbo

dubbo.protocol.name = dubbo

dubbo.protocol.port = 12345

## RegistryConfig Bean

dubbo.registry.id = my-registry

dubbo.registry.address = N/A

###### 3、接下来在Spring Boot Application的上添加@EnableDubboConfiguration，表示要开启dubbo功能. (dubbo provider服务可以使用或者不使用web容器)

@SpringBootApplication

@EnableDubboConfiguration

public class DubboProviderLauncher {

//...

}

###### 4、编写你的dubbo服务，只需要添加要发布的服务实现上添加@Service（import com.alibaba.dubbo.config.annotation.Service）注解，其中interfaceClass是要发布服务的接口.

@Service(interfaceClass = IHelloService.class)

@Component

public class HelloServiceImpl implements IHelloService {

//...

}

==注意：实体类必须实现Serializable接口==

#### 二、消费端的消费服务

###### 1、添加依赖

同上

###### 2、配置文件

# Spring boot application

spring.application.name = dubbo-provider-demo

# Dubbo Config properties

## ApplicationConfig Bean

dubbo.application.id = dubbo-consumer-demo

dubbo.application.name = dubbo-consumer-demo

## ProtocolConfig Bean

dubbo.protocol.id = dubbo

dubbo.protocol.name = dubbo

dubbo.protocol.port = 12345

###### 3、add @EnableDubboConfiguration on Spring Boot Application

@SpringBootApplication

@EnableDubboConfiguration

public class DubboConsumerLauncher {

//...

}

###### 4、使用

@RestController

public class DubboConsumer {

@Reference(url = "dubbo://127.0.0.1:20880")

private HseCustomerService customerService;

@GetMapping("list2")

public Object getList(@RequestParam(defaultValue = "0")Integer fromId,

@RequestParam(defaultValue = "2")Integer limit ){

//初始化page插件，传入分页参数

PageHelper.startPage(fromId,limit);

List<HseCustomer> list = customerService.getList();

//包装想要返回的结果，包含多种信息

PageInfo pageInfo = new PageInfo(list);

return pageInfo;

}

}