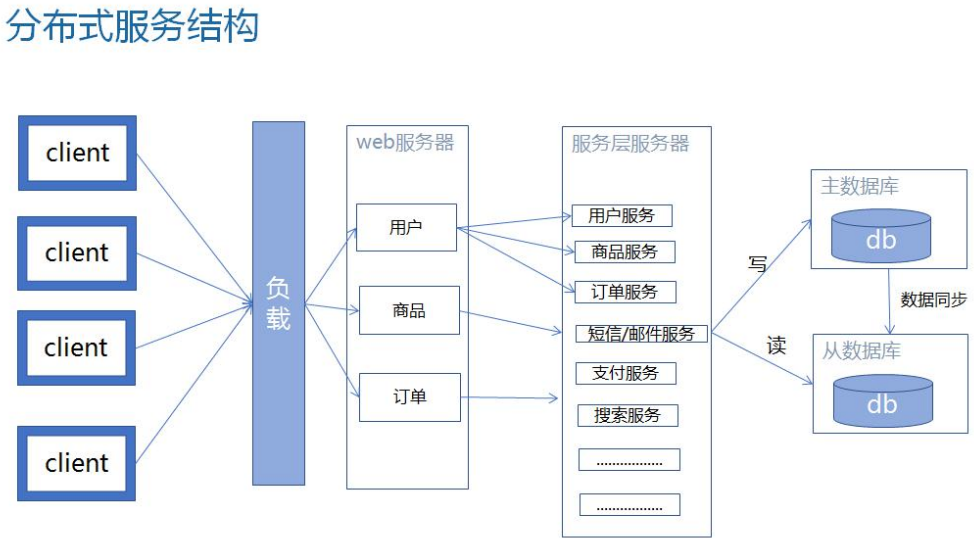
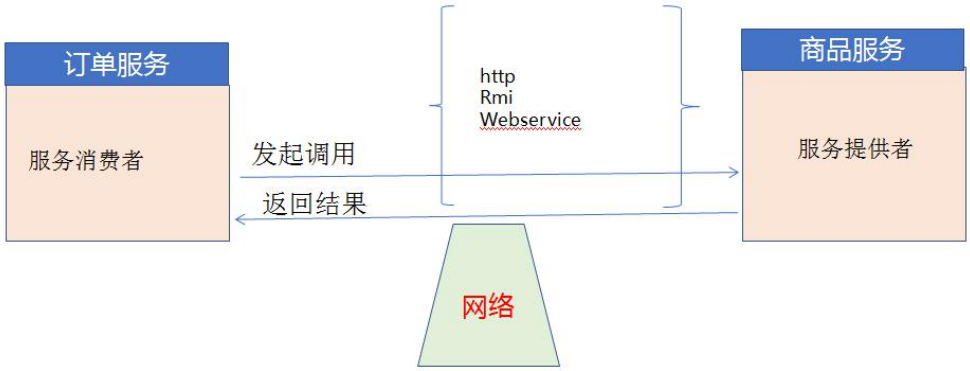
1 RPC 场景和过程

1.1RPC 场景

在微服务环境下，存在大量的跨 JVM 进行方法调用的场景，如下图：



具体到某一个调用来说，希望 A 机器能通过网络，调用 B 机器内的某个服务方法，并得到返回值：



1.2RPC 的实现切入口

从本质上来讲，某个 jvm 内的对象方法，是无法在 jvm 外部被调用的，如下图的代码：

```
OrderService orderService = (OrderService) ctx.getBean("orderService");
OrderEntiry entiry = orderService.getDetail("1");
```

orderService.getDetail("1")的这一句调用，是无法脱离本地 jvm 环境被调用的。

但是，好在 java 中的对象方法的调用，还有反射模式的调用：

```
Method method = target.getClass().getMethod(methodName, argTypes);
return method.invoke(target, args);
```

只要我们传入反射需要的目标对象 **orderService,**方法名称 **getDetail**

和参数值**"1",**反射就能将 **orderService.getDetail** 调用起来

于是，我们可以把调用某个方法的过程，改造成这样

```
Map<String,String> info = new HashMap();
info.put("target","orderService");
info.put("methodName","getDetail");
info.put("arg","1");
//反射调用
Object result = InvokeUtils.call(info,ctx);
```

现在，只要谁告诉我了反射需要信息，target/method/arg，我就能调用本地的任何对象方法了

1.3网络通信传递反射信息

在上一节的步骤中，本地方法已经宣称，只要传递给他 target/method/arg，它就能帮助我们执行想要的目标服务方法，那么现在，我们只需要解决 target/method/arg 三种信息的网络传输问题。

网络通信的方法很多，如 http/rmi/webservice 等等，我们只需要选用任意一种即可，为简便起见，我们选用 jdk 的 rmi 方式，其使用方式如下：

1.3.1 定义一个继承自 remote 接口的类

```
public interface InfoService extends Remote {//继承 remote 接口
    String RMI_URL = "rmi://127.0.0.1:9080/InfoService";
    int port = 9080;
    Object passInfo(Map<String,String> info) throws RemoteException;
}
```

创建一个实现类（为简化实现，继承 `UnicastRemoteObject` 类）

```
public class InfoServiceImpl extends UnicastRemoteObject implements InfoService {
    public InfoServiceImpl() throws RemoteException {
        super();
    }
    @Override
```

```
    public Object passInfo(Map<String, String> info) {
        System.out.println("恭喜你，调通了，参数："+JSON.toJSONString(info));
        info.put("msg","你好，调通了！");
        return info;
    }
}
```

1.3.2 RMI 开放服务到指定 URL

只需要将实例绑定注册到指定的 URL 和 port 上,远程即可调用此实例

```
InfoService infoService = new InfoServiceImpl();
//注册通讯端口
LocateRegistry.createRegistry(9080);
//注册通讯路径
Naming.bind("rmi://127.0.0.1:9080/InfoService
", infoService);
```

1.3.3 RMI 远程通过 URL 连接并调用

```
//取远程服务实现
infoService = (InfoService) Naming.lookup(InfoService.RMI_URL);
//呼叫远程反射方法
Map<String,String> info = new HashMap();
info.put("target","orderService");
info.put("methodName","getDetail");
info.put("arg","1");
Object result = infoService.passInfo(info);
```

至此，已经实现了通过 RMI 跨机器传递 target/method/arg

1.4远程调用的融合

可以看到，前两步，已经实现了跨机器的反射信息收发和反射动作调用。我们现在只需要在 `InfoService` 的实现上，对传递过来的 `info` 信息，直接发起反射调用即可

```
InfoService infoService = new InfoServiceImpl(){
    public Object passInfo(Map<String, String> info) {//对象，方法，参数

        super.passInfo(info);//info 内包含的信息，是反射需要的信息
        Object result = InvokeUtils.call(info,ctx);
        System.out.println("测试 InvokeUtils.call 调用功能，调用结果：" + JSON.toJSONString(result));
        return result;
```

```
    }
};
```

这样，远程机器只要通过 infoService 传递信息过来，就自动将目标服务反射调用，并返回结果值回去，整个 RPC 过程完成

1.5对客户端友好的透明化封装

虽然在上一节中，RPC 的整个调用链条已经拉通，但是我们发现还有一个易出错的地方，就是客户端封装反射信息的地方，功能不够内聚，容易出现手误，代码易读性也很差

```
//呼叫远程反射方法
Map<String,String> info = new HashMap();
info.put("target","orderService");
info.put("methodName","getDetail");
info.put("arg","1");
Object result = infoService.passInfo(info);
```

有什么改善的办法呢？
仔细核对，其实 target/method/arg 这三个信息，全部都可以从接口方法调用 `OrderService.getDetail("1")` 中得到，于是，我们可以为此接口做一个代理对象，在代理对象内部完成反射信息的包装：

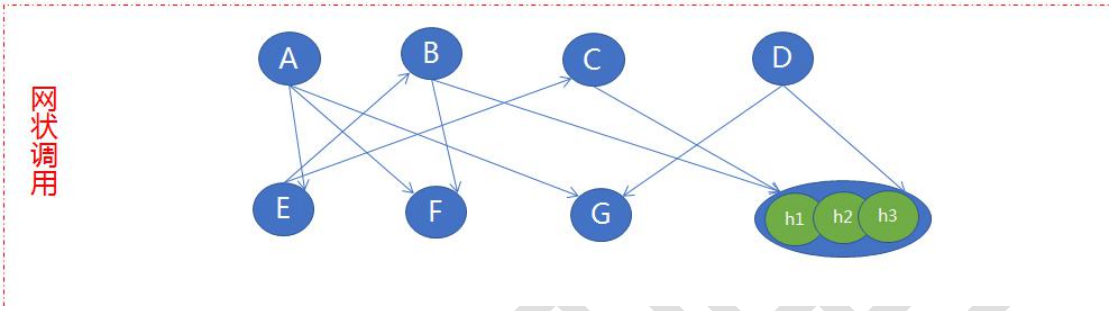
```
OrderService service = new OrderService(){
    @Override
    public OrderEntiry getDetail(String id) {
        Map<String,String> info = new HashMap();
        //写死了反射的目标，静态代理
        info.put("target","orderService");//对象
        info.put("methodName","getDetail");//方法
        info.put("arg",id);//参数

        OrderEntiry result = null;
        try {
            result = (OrderEntiry)infoService.passInfo(info);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        return result;
    }
};
```

大功告成，从此，客户端远程传递反射信息的过程，直接变成调用接口的代理对象即可。调用者，甚至不再需要区分，此接口代理对象到底是谁，像调

用正常的本地服务一起使用就 ok

1.6Dubbo 使命

在上面的章节，我们重点详述了，一个具体的 RPC 调用的全过程。那么在现实工作中，服务节点间的 RPC 调用是非常普遍并且错踪复杂的



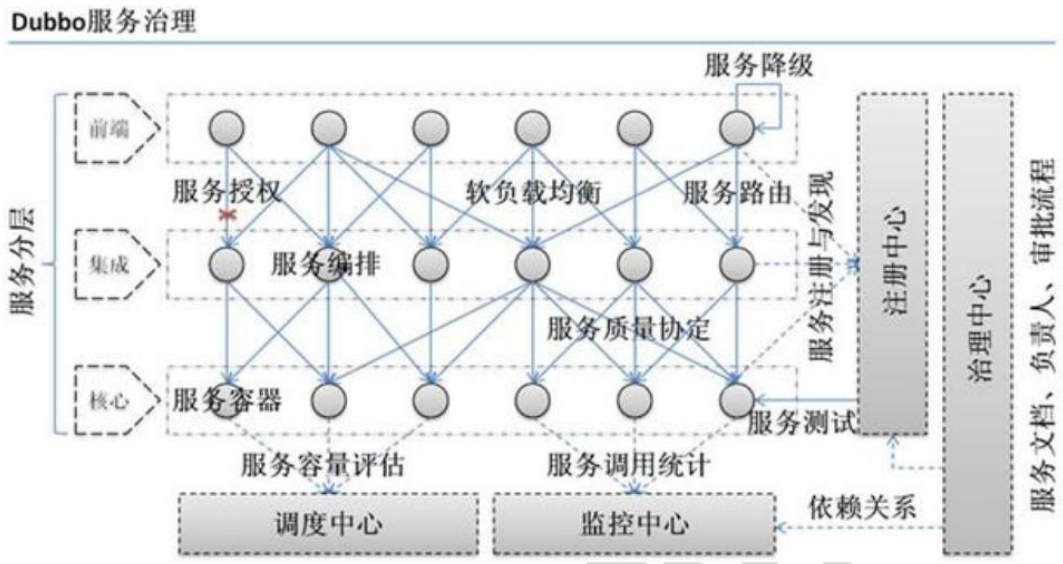
我们除了要关心 RPC 的过程实现，还需要考虑：
1.服务方是集群时，如何挑选一台机器来响应客户端？
2.因网络抖动引起的调用失败，如何重试来弥补？
3.服务方机器的动态增减，如何能够让客户端及时了解到并做出调整？
.....

Dubbo 的使命，即是解决上述围绕 RPC 过程的一览子问题

2 Dubbo 简介

在分布式服务架构下，各个服务间的相互 rpc 调用会越来越复杂。最终形成网状结构，此时服务的治理极为关键。

Dubbo 是一个带有服务治理功能的 RPC 框架，提供了一套较为完整的服务治理方案，其底层直接实现了 rpc 调用的全过程，并尽力做事 rpc 远程对使用者透明。下图展示了 Dubbo 服务治理的功能。

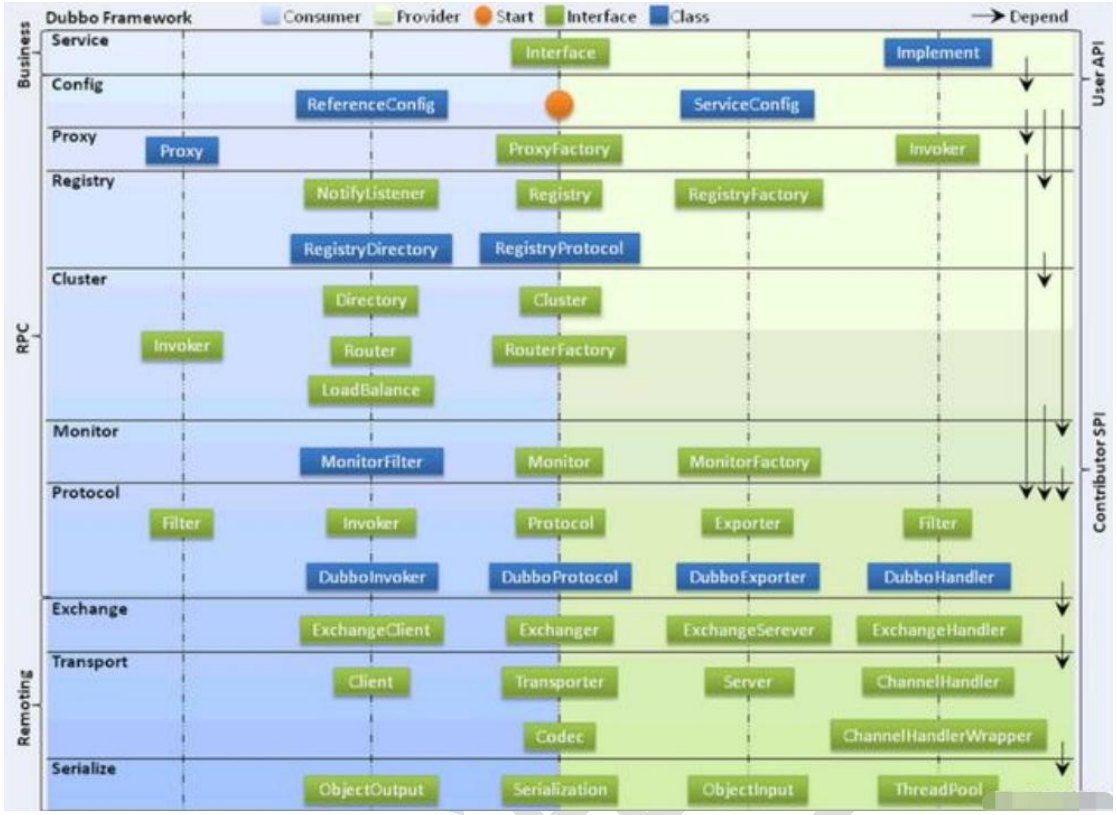


简单的说，Dubbo 就是个服务调用的框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有使用 Dubbo 这样的分布式服务框架的需求，并且本质上是个服务调用的东东。
其核心部分包含:

- 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。
- 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。
- 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×××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。

2.1 dubbo 的架构及特点

下图展示了 dubbo 的整体结构



Dubbo 总体架构设计一共划分了 10 层，而最上面的 Service 层是留给实际想要使用 Dubbo 开发分布式服务的开发者实现业务逻辑的接口层。图中左边淡蓝背景的为服务消费方使用的接口，右边淡绿色背景的为服务提供方使用的接口，位于中轴线上的为双方都用到的接口。

- 服务接口层(Service)：该层是与实际业务逻辑相关的，根据服务提供方和服务消费方的业务设计对应的接口和实现。
- 配置层(Config)：对外配置接口，以 ServiceConfig 和 ReferenceConfig 为中心，可以直接 new 配置类，也可以通过 Spring 解析配置生成配置类。
- 服务代理层(Proxy)：服务接口透明代理，生成服务的客户端 Stub 和服务器端 Skeleton，以 ServiceProxy 为中心，扩展接口为 ProxyFactory。
- 服务注册层(Registry)：封装服务地址的注册与发现，以服务 URL 为中心，扩展接口为 RegistryFactory、Registry 和 RegistryService。可能没有服务注册中心，此时服务提供方直接暴露服务。
- 集群层(Cluster)：封装多个提供者的路由及负载均衡，并桥接注册中心，以 Invoker 为中心，扩展接口为 Cluster、Directory、Router 和 LoadBalance。将多个服务提供方组合为一个服务提供方，实现对服务消费方透明，只需要与一个服务提供方进行交互。
- 监控层(Monitor)：RPC 调用次数和调用时间监控，以 Statistics 为中心，扩展接口为 MonitorFactory、Monitor 和 MonitorService。
- 远程调用层(Protocol)：封将 RPC 调用，以 Invocation 和 Result 为中心，扩展接口为 Protocol、Invoker 和 Exporter。Protocol 是服务域，它是 Invoker 暴露和引用的主功能入口，它负责 Invoker 的生命周期管理。Invoker 是实体域，它是 Dubbo 的核心模型，其他模型都向它靠扰，或转换成它，

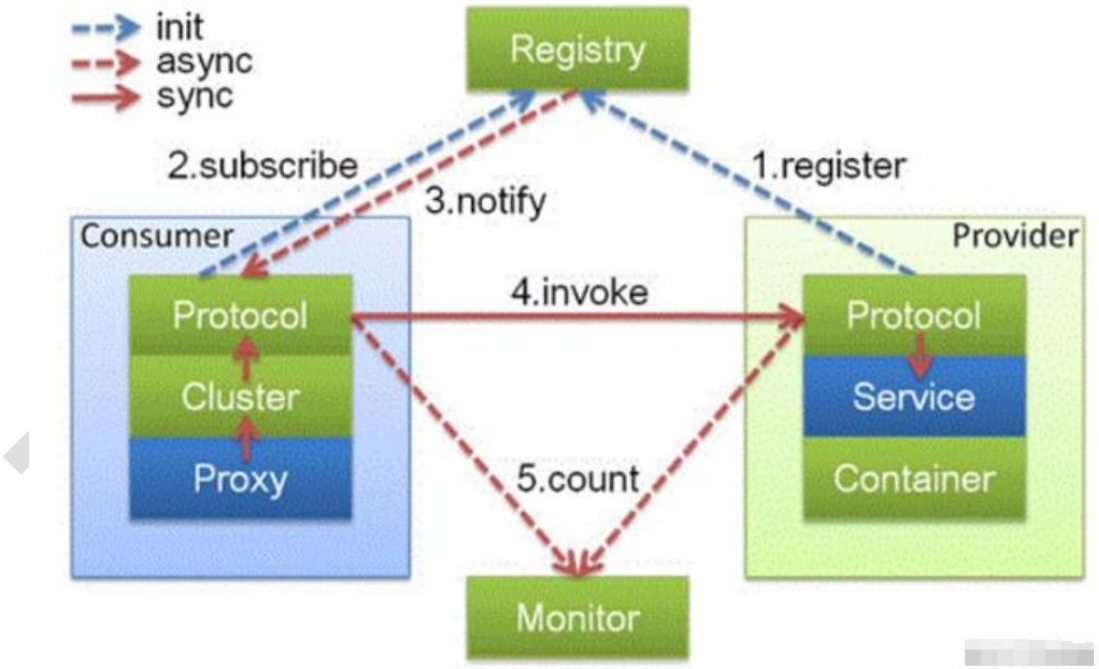
它代表一个可执行体，可向它发起 invoke 调用。它有可能是一个本地的实现，也可能是一个远程的实现，也可能是一个集群实现。

- 信息交换层(Exchange)：封装请求响应模式，同步转异步，以 Request 和 Response 为中心，扩展接口为 Exchanger、ExchangeChannel、ExchangeClient 和 ExchangeServer。
- 网络传输层(Transport)：抽象 mina 和 netty 为统一接口，以 Message 为中心，扩展接口为 Channel、Transporter、Client、Server 和 Codec。
- 数据序列化层(Serialize)：可复用的一些工具，扩展接口为 Serialization、ObjectInput、ObjectOutput 和 ThreadPool。

从上图可以看出，Dubbo 对于服务提供方和服务消费方，从框架的 10 层中分别提供了各自需要关心和扩展的接口，构建整个服务生态系统(服务提供方和服务消费方本身就是一个以服务为中心的)。

2.2Dubbo 服务的角色关系

服务提供方和服务消费方之间的调用关系，如图所示：



节点角色说明：

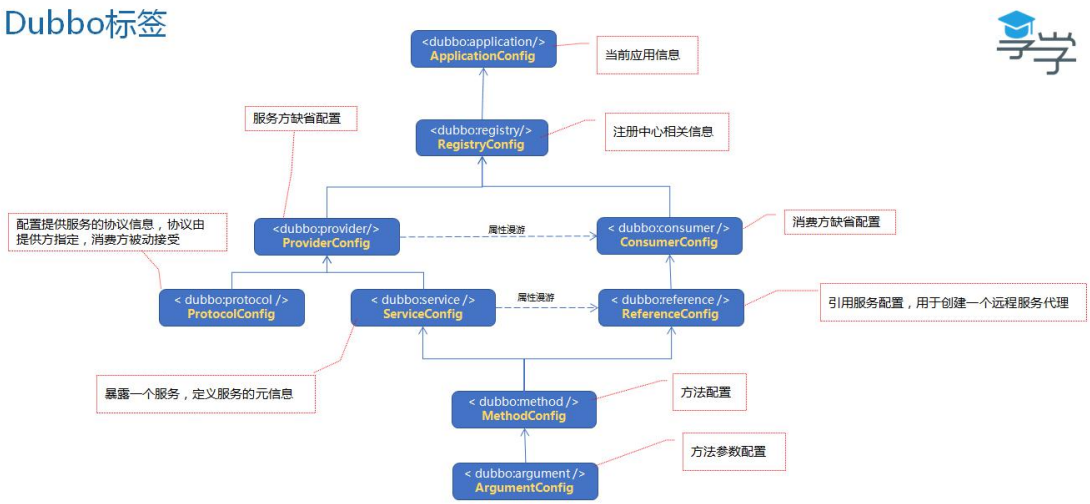
节点	角色说明
Provider	暴露服务的服务提供方
Consumer	调用远程服务的服务消费方
Registry	服务注册与发现的注册中心
Monitor	统计服务的调用次数和调用时间的监控中心
Container	服务运行容器

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。

3 Dubbo 的基础配置使用

3.1 xml 配置方式

3.1.1 dubbo 功能标签集



1、标签属性有继承关系，即：下层有设置则使用，未配置则沿用上一级的设置
2、 timeout/retries/loadbalance消费方未设置，则沿用服务方的设置。

<dubbo:service/> 服务配置，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:reference/> 引用配置，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:protocol/> 协议配置，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:application/> 应用配置，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/> 注册中心配置，用于配置连接注册中心相关信息。

<dubbo:module/> 模块配置，用于配置当前模块信息，可选。

<dubbo:monitor/> 监控中心配置，用于配置连接监控中心相关信息，可选。

<dubbo:provider/> 提供方的缺省值，当 ProtocolConfig 和 ServiceConfig 某属性没有配置时，采用此缺省值，可选。

<dubbo:consumer/> 消费方缺省配置，当 ReferenceConfig 某属性没有配置时，采用此缺省值，可选。

<dubbo:method/> 方法配置，用于 ServiceConfig 和 ReferenceConfig 指定方法级的配置信息。

<dubbo:argument/> 用于指定方法参数配置。

dubbo 标签的使用以及标签属性详解

其实

3.1.2 标签详解

所有配置项分为三大类，参见下表中的"作用"一列。

- 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。
- 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。

◆ 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。

所有配置最终都将转换为 URL 表示，并由服务提供方生成，经注册中心传递给消费方，各属性对应 URL 的参数，参见配置项一览表中的"对应 URL 参数"列。

注意：只有 group，interface，version 是服务的匹配条件，三者决定是不是同一个服务，其它配置项均为调优和治理参数。

<dubbo:service/>

服务提供者暴露服务配置：

配置类：com.alibaba.dubbo.config.ServiceConfig

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:service>	interface		class	必填		服务发现	服务接口名	1.0.0 以上版本
<dubbo:service>	ref		object	必填		服务发现	服务对象实现引用	1.0.0 以上版本
<dubbo:service>	version	version	string	可选	0.0.0	服务发现	服务版本，建议使用两位数字版本，如：1.0，通常在接口不兼容时版本号才需要升级	1.0.0 以上版本
<dubbo:service>	group	group	string	可选		服务发现	服务分组，当一个接口有多个实现，可以用分组区分	1.0.7 以上版本
<dubbo:service>	path	<path>	string	可选	缺省为接口名	服务发现	服务路径 (注意：1.0 不支持自定义路径，总是使用接口名，如果有 1.0 调 2.0，配置服务路径可能不兼容)	1.0.12 以上版本
<dubbo:service>	delay	delay	int	可选	0	性能调优	延迟注册服务时间(毫秒) ，设为-1 时，表示延迟到 Spring 容器初始化完成时暴露服务	1.0.14 以上版本
<dubbo:service>	timeout	timeout	int	可选	1000	性能调优	远程服务调用超时时间(毫秒)	2.0.0 以上版本
<dubbo:service>	retries	retries	int	可选	2	性能调优	远程服务调用重试次数，不包括第一次调用，不需要重试请设为 0	2.0.0 以上版本
<dubbo:service>	connections	connections	int	可选	100	性能调优	对每个提供者的最大连接数，rmi、http、hessian 等短连接协议表示限制连接数，dubbo 等长连接协表示建立的长连接个数	2.0.0 以上版本
<dubbo:service>	loadbalance	loadbalance	string	可选	random	性能调优	负载均衡策略，可选值：random,roundrobin,leastactive，分别表示：随机，轮循，最少活跃调用	2.0.0 以上版本
<dubbo:service>	async	async	boolean	可选	false	性能调优	是否缺省异步执行，不可靠异步，只是忽略返回值，不阻塞执行线程	2.0.0 以上版本
<dubbo:service>	stub	stub	class/boolean	可选	false	服务治理	设为 true，表示使用缺省代理类名，即：接口名 + Local 后缀，服务接口客户端本地代理类名，用于在客户端执行本地逻辑，如本地缓存等，该本地代理类的构造函数必须允许传入远程代理	2.0.0 以上版本

							对象，构造函数如：public XxxServiceLocal(XxxService xxxService)	
<dubbo:service>	mock	mock	class/boolean	可选	false	服务治理	设为 true，表示使用缺省 Mock 类名，即：接口名 + Mock 后缀，服务接口调用失败 Mock 实现类，该 Mock 类必须有一个无参构造函数，与 Local 的区别在于，Local 总是被执行，而 Mock 只在出现非业务异常(比如超时，网络异常等)时执行，Local 在远程调用之前执行，Mock 在远程调用后执行。	2.0.0 以上版本
<dubbo:service>	token	token	string/boolean	可选	false	服务治理	令牌验证，为空表示不开启，如果为 true，表示随机生成动态令牌，否则使用静态令牌，令牌的作用是防止消费者绕过注册中心直接访问，保证注册中心的授权功能有效，如果使用点对点调用，需关闭令牌功能	2.0.0 以上版本
<dubbo:service>	registry		string	可选	缺省向所有 registry 注册	配置关联	向指定注册中心注册，在多个注册中心时使用，值为<dubbo:registry>的 id 属性，多个注册中心 ID 用逗号分隔，如果不想将该服务注册到任何 registry，可将值设为 N/A	2.0.0 以上版本
<dubbo:service>	provider		string	可选	缺使用第一个 provider 配置	配置关联	指定 provider，值为<dubbo:provider>的 id 属性	2.0.0 以上版本
<dubbo:service>	deprecated	deprecated	boolean	可选	false	服务治理	服务是否过时，如果设为 true，消费方引用时将打印服务过时警告 error 日志	2.0.5 以上版本
<dubbo:service>	dynamic	dynamic	boolean	可选	true	服务治理	服务是否动态注册，如果设为 false，注册后将显示后 disable 状态，需人工启用，并且服务提供者停止时，也不会自动取消册，需人工禁用。	2.0.5 以上版本
<dubbo:service>	accesslog	accesslog	string/boolean	可选	false	服务治理	设为 true，将向 logger 中输出访问日志，也可填写访问日志文件路径，直接把访问日志输出到指定文件	2.0.5 以上版本
<dubbo:service>	owner	owner	string	可选		服务治理	服务负责人，用于服务治理，请填写负责人公司邮箱前缀	2.0.5 以上版本
<dubbo:service>	document	document	string	可选		服务治理	服务文档 URL	2.0.5 以上版本
<dubbo:service>	weight	weight	int	可选		性能调优	服务权重	2.0.5 以上版本
<dubbo:service>	executes	executes	int	可选	0	性能调优	服务提供者每服务每方法最大可并行执行请求数	2.0.5 以上版本
<dubbo:service>	actives	actives	int	可选	0	性能调优	每服务消费者每服务每方法最大并发调用数	2.0.5 以上版本
<dubbo:service>	proxy	proxy	string	可选	javassist	性能调优	生成动态代理方式，可选：jdk/javassist	2.0.5 以上版本
<dubbo:service>	cluster	cluster	string	可选	failover	性能调优	集群方式，可选：failover/failfast/failsafe/failback/forking	2.0.5 以上版本
<dubbo:service>	filter	service.filter	string	可选	default	性能调优	服务提供方远程调用过程拦截器名称，多个名称用逗号分隔	2.0.5 以上版本
<dubbo:service>	listener	exporter.listener	string	可选	default	性能调优	服务提供方导出服务监听器名称，多个名称用逗号分隔	

<dubbo:service>	protocol		string	可选		配置关联	使用指定的协议暴露服务，在多协议时使用，值为<dubbo:protocol>的 id 属性，多个协议 ID 用逗号分隔	2.0.5 以上版本
<dubbo:service>	layer	layer	string	可选		服务治理	服务提供者所在的分层。如：biz、dao、intl:web、china:acton。	2.0.7 以上版本
<dubbo:service>	register	register	boolean	可选	true	服务治理	该协议的服务是否注册到注册中心	2.0.8 以上版本

<dubbo:reference/>

服务消费者引用服务配置：

配置类：com.alibaba.dubbo.config.ReferenceConfig

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:reference>	id		string	必填		配置关联	服务引用 BeanId	1.0.0 以上版本
<dubbo:reference>	interface		class	必填		服务发现	服务接口名	1.0.0 以上版本
<dubbo:reference>	version	version	string	可选		服务发现	服务版本，与服务提供者的版本一致	1.0.0 以上版本
<dubbo:reference>	group	group	string	可选		服务发现	服务分组，当一个接口有多个实现，可以用分组区分，必需和服务提供方一致	1.0.7 以上版本
<dubbo:reference>	timeout	timeout	long	可选	缺省使用<dubbo:consumer>的 timeout	性能调优	服务方法调用超时时间(毫秒)	1.0.5 以上版本
<dubbo:reference>	retries	retries	int	可选	缺省使用<dubbo:consumer>的 retries	性能调优	远程服务调用重试次数，不包括第一次调用，不需要重试请设为 0	2.0.0 以上版本
<dubbo:reference>	connections	connections	int	可选	缺省使用<dubbo:consumer>的 connections	性能调优	对每个提供者的最大连接数，rmi、http、hessian 等短连接协议表示限制连接数，dubbo 等长连接协表示建立的长连接个数	2.0.0 以上版本
<dubbo:reference>	loadbalance	loadbalance	string	可选	缺省使用<dubbo:consumer>的 loadbalance	性能调优	负载均衡策略，可选值：random,roundrobin,leastactive，分别表示：随机，轮循，最少活跃调用	2.0.0 以上版本
<dubbo:reference>	async	async	boolean	可选	缺省使用<dubbo:consumer>的 async	性能调优	是否异步执行，不可靠异步，只是忽略返回值，不阻塞执行线程	2.0.0 以上版本
<dubbo:reference>	generic	generic	boolean	可选	缺省使用<dubbo:consumer>的 generic	服务治理	是否缺省泛化接口，如果为泛化接口，将返回 GenericService	2.0.0 以上版本
<dubbo:reference>	check	check	boolean	可选	缺省使用<dubbo:consumer>的 check	服务治理	启动时检查提供者是否存在，true 报错，false 忽略	2.0.0 以上版本
<dubbo:reference>	url	<url>	string	可选		服务治理	点对点直连服务提供者地址，将绕过注册中心	1.0.6 以上版本
<dubbo:reference>	stub	stub	class/boolean	可选		服务治理	服务接口客户端本地代理类名，用于在客户端执行本地逻辑，如本地缓存等，该本	2.0.0 以上版本

							地代理类的构造函数必须允许传入远程代理对象，构造函数如：public XxxServiceLocal(XxxService xxxService)	
<dubbo:reference>	mock	mock	class/boolean	可选		服务治理	服务接口调用失败 Mock 实现类名，该 Mock 类必须有一个无参构造函数，与 Local 的区别在于，Local 总是被执行，而 Mock 只在出现非业务异常(比如超时，网络异常等)时执行，Local 在远程调用之前执行，Mock 在远程调用后执行。	Dubbo1.0.13 及其以上版本支持
<dubbo:reference>	cache	cache	string/boolean	可选		服务治理	以调用参数为 key，缓存返回结果，可选：lru, threadlocal, jcache 等	Dubbo2.1.0 及其以上版本支持
<dubbo:reference>	validation	validation	boolean	可选		服务治理	是否启用 JSR303 标准注解验证，如果启用，将对方法参数上的注解进行校验	Dubbo2.1.0 及其以上版本支持
<dubbo:reference>	proxy	proxy	boolean	可选	javassist	性能调优	选择动态代理实现策略，可选：javassist, jdk	2.0.2 以上版本
<dubbo:reference>	client	client	string	可选		性能调优	客户端传输类型设置，如 Dubbo 协议的 netty 或 mina。	Dubbo2.0.0 以上版本支持
<dubbo:reference>	registry		string	可选	缺省将从所有注册中心获服务列表后合并结果	配置关联	从指定注册中心注册获取服务列表，在多个注册中心时使用，值为 <dubbo:registry>的 id 属性，多个注册中心 ID 用逗号分隔	2.0.0 以上版本
<dubbo:reference>	owner	owner	string	可选		服务治理	调用服务负责人，用于服务治理，请填写负责人公司邮箱前缀	2.0.5 以上版本
<dubbo:reference>	actives	actives	int	可选	0	性能调优	每服务消费者每服务每方法最大并发调用数	2.0.5 以上版本
<dubbo:reference>	cluster	cluster	string	可选	failover	性能调优	集群方式，可选：failover/failfast/failsafe/failback/forking	2.0.5 以上版本
<dubbo:reference>	filter	reference.filter	string	可选	default	性能调优	服务消费方远程调用过程拦截器名称，多个名称用逗号分隔	2.0.5 以上版本
<dubbo:reference>	listener	invoker.listener	string	可选	default	性能调优	服务消费方引用服务监听器名称，多个名称用逗号分隔	2.0.5 以上版本
<dubbo:reference>	layer	layer	string	可选		服务治理	服务调用者所在的分层。如：biz、dao、intl:web、china:acton。	2.0.7 以上版本
<dubbo:reference>	init	init	boolean	可选	false	性能调优	是否在 afterPropertiesSet()时饥饿初始化引用，否则等到有人注入或引用该实例时再初始化。	2.0.10 以上版本
<dubbo:reference>	protocol	protocol	string	可选		服力治理	只调用指定协议的服务提供方，其它协议忽略。	2.2.0 以上版本

<dubbo:protocol/>

服务提供者协议配置：

配置类：com.alibaba.dubbo.config.ProtocolConfig

说明：如果需要支持多协议，可以声明多个<dubbo:protocol>标签，并在<dubbo:service>中通过 protocol 属性指定使用的协议。

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:protocol>	id		string	可选	dubbo	配置关联	协议 BeanId，可以在<dubbo:service protocol="">中引用此 ID，如果 ID 不填，缺省和 name 属性值一样，重复则在 name 后加序号。	2.0.5 以上版本
<dubbo:protocol>	name	<protocol>	string	必填	dubbo	性能调优	协议名称	2.0.5 以上版本
<dubbo:protocol>	port	<port>	int	可选	dubbo 协议缺省端口为 20880，rmi 协议缺省端口为 1099，http 和 hessian 协议缺省端口为 80 如果配置为-1 或者 没有配置 port，则会分配一个没有被占用的端口。 Dubbo 2.4.0+，分配的端口在协议缺省端口的基础上增长，确保端口段可控。	服务发现	服务端口	2.0.5 以上版本
<dubbo:protocol>	host	<host>	string	可选	自动查找本机 IP	服务发现	-服务主机名，多网卡选择或指定 VIP 及域名时使用，为空则自动查找本机 IP，-建议不要配置，让 Dubbo 自动获取本机 IP	2.0.5 以上版本
<dubbo:protocol>	threadpool	threadpool	string	可选	fixed	性能调优	线程池类型，可选：fixed/cached	2.0.5 以上版本
<dubbo:protocol>	threads	threads	int	可选	100	性能调优	服务线程池大小(固定大小)	2.0.5 以上版本
<dubbo:protocol>	iothreads	threads	int	可选	cpu 个数+1	性能调	io 线程池大小(固定大小)	2.0.5 以上版本

						优		
<dubbo:protocol>	accepts	accepts	int	可选	0	性能调优	服务提供方最大可接受连接数	2.0.5 以上版本
<dubbo:protocol>	payload	payload	int	可选	88388608(=8M)	性能调优	请求及响应数据包大小限制，单位：字节	2.0.5 以上版本
<dubbo:protocol>	codec	codec	string	可选	dubbo	性能调优	协议编码方式	2.0.5 以上版本
<dubbo:protocol>	serialization	serialization	string	可选	dubbo 协议缺省为 hessian2，rmi 协议缺省为java，http 协议缺省为 json	性能调优	协议序列化方式，当协议支持多种序列化方式时使用，比如：dubbo 协议的 dubbo,hessian2,java,compactedjava，以及 http 协议的 json 等	2.0.5 以上版本
<dubbo:protocol>	accesslog	accesslog	string/boolean	可选		服务治理	设为 true，将向 logger 中输出访问日志，也可填写访问日志文件路径，直接把访问日志输出到指定文件	2.0.5 以上版本
<dubbo:protocol>	path	<path>	string	可选		服务发现	提供者上下文路径，为服务 path 的前缀	2.0.5 以上版本
<dubbo:protocol>	transporter	transporter	string	可选	dubbo 协议缺省为 netty	性能调优	协议的服务端和客户端实现类型，比如：dubbo 协议的 mina,netty 等，可以分拆为 server 和 client 配置	2.0.5 以上版本
<dubbo:protocol>	server	server	string	可选	dubbo 协议缺省为 netty，http 协议缺省为 servlet	性能调优	协议的服务器端实现类型，比如：dubbo 协议的 mina,netty 等，http 协议的 jetty,servlet 等	2.0.5 以上版本
<dubbo:protocol>	client	client	string	可选	dubbo 协议缺省为 netty	性能调优	协议的客户端实现类型，比如：dubbo 协议的 mina,netty 等	2.0.5 以上版本
<dubbo:protocol>	dispatcher	dispatcher	string	可选	dubbo 协议缺省为 all	性能调优	协议的消息派发方式，用于指定线程模型，比如：dubbo 协议的 all, direct, message, execution, connection 等	2.1.0 以上版本
<dubbo:protocol>	queues	queues	int	可选	0	性能调	线程池队列大小，当线程池满时，排队等待执行的队列大小，建议不要设置，当线程程池时应立即失败，重试其它服务提供	2.0.5 以上版本

						优	机器，而不是排队，除非有特殊需求。	
<dubbo:protocol>	charset	charset	string	可选	UTF-8	性能调优	序列化编码	2.0.5 以上版本
<dubbo:protocol>	buffer	buffer	int	可选	8192	性能调优	网络读写缓冲区大小	2.0.5 以上版本
<dubbo:protocol>	heartbeat	heartbeat	int	可选	0	性能调优	心跳间隔，对于长连接，当物理层断开时，比如拔网线，TCP 的 FIN 消息来不及发送，对方收不到断开事件，此时需要心跳来帮助检查连接是否已断开	2.0.10 以上版本
<dubbo:protocol>	telnet	telnet	string	可选		服务治理	所支持的 telnet 命令，多个命令用逗号分隔	2.0.5 以上版本
<dubbo:protocol>	register	register	boolean	可选	true	服务治理	该协议的服务是否注册到注册中心	2.0.8 以上版本
<dubbo:protocol>	contextpath	contextpath	String	可选	缺省为空串	服务治理		2.0.6 以上版本

<dubbo:registry/>

注册中心配置：

配置类：com.alibaba.dubbo.config.RegistryConfig

说明：如果有多个不同的注册中心，可以声明多个<dubbo:registry>标签，并在<dubbo:service>或<dubbo:reference>的 registry 属性指定使用的注册中心。

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:registry>	id		string	可选		配置关联	注册中心引用 BeanId，可以在<dubbo:service registry="">或<dubbo:reference registry="">中引用此 ID	1.0.16 以上版本
<dubbo:registry>	address	<host:port>	string	必填		服务发	注册中心服务器地址，如果地址没有端口缺省为 9090，同一集群内的多个地址用逗号分隔，如：ip:port,ip:port，不同集群的注册中心，请配置多个	1.0.16 以上版本

						现	<dubbo:registry>标签	
<dubbo:registry>	protocol	<protocol>	string	可选	dubbo	服务发现	注同中心地址协议，支持 dubbo, http, local 三种协议，分别表示，dubbo 地址，http 地址，本地注册中心	2.0.0 以上版本
<dubbo:registry>	port	<port>	int	可选	9090	服务发现	注册中心缺省端口，当 address 没有带端口时使用此端口做为缺省值	2.0.0 以上版本
<dubbo:registry>	username	<username>	string	可选		服务治理	登录注册中心用户名，如果注册中心不需要验证可不填	2.0.0 以上版本
<dubbo:registry>	password	<password>	string	可选		服务治理	登录注册中心密码，如果注册中心不需要验证可不填	2.0.0 以上版本
<dubbo:registry>	transport	registry.transporter	string	可选	netty	性能调优	网络传输方式，可选 mina,netty	2.0.0 以上版本
<dubbo:registry>	timeout	registry.timeout	int	可选	5000	性能调优	注册中心请求超时时间(毫秒)	2.0.0 以上版本
<dubbo:registry>	session	registry.session	int	可选	60000	性能调优	注册中心会话超时时间(毫秒)，用于检测提供者非正常断线后的脏数据，比如用心跳检测的实现，此时间就是心跳间隔，不同注册中心实现不一样。	2.1.0 以上版本
<dubbo:registry>	file	registry.file	string	可选		服务治理	使用文件缓存注册中心地址列表及服务提供者列表，应用重启时将基于此文件恢复，注意：两个注册中心不能使用同一文件存储	2.0.0 以上版本
<dubbo:registry>	wait	registry.wait	int	可选	0	性能调优	停止时等待通知完成时间(毫秒)	2.0.0 以上版本
<dubbo:registry>	check	check	boolean	可选	true	服务治理	注册中心不存在时，是否报错	2.0.0 以上版本
<dubbo:registry>	register	register	boolean	可选	true	服务治	是否向此注册中心注册服务，如果设为 false，将只订阅，不注册	2.0.5 以上版本

						理		
<dubbo:registry>	subscribe	subscribe	boolean	可选	true	服务治理	是否向此注册中心订阅服务，如果设为 false，将只注册，不订阅	2.0.5 以上版本
<dubbo:registry>	dynamic	dynamic	boolean	可选	true	服务治理	服务是否动态注册，如果设为 false，注册后将显示后 disable 状态，需人工启用，并且服务提供者停止时，也不会自动取消册，需人工禁用。	2.0.5 以上版本

<dubbo:monitor/>

监控中心配置：

配置类：com.alibaba.dubbo.config.MonitorConfig

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:monitor>	protocol	protocol	string	可选	dubbo	服务治理	监控中心协议，如果为 protocol="registry"，表示从注册中心发现监控中心地址，否则直连监控中心。	2.0.9 以上版本
<dubbo:monitor>	address	<url>	string	可选	N/A	服务治理	直连监控中心服务器地址，address="10.20.130.230:12080"	1.0.16 以上版本

<dubbo:application/>

应用信息配置：

配置类：com.alibaba.dubbo.config.ApplicationConfig

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:application>	name	application	string	必填		服务治理	当前应用名称，用于注册中心计算应用间依赖关系，注意：消费者和提供者应用名不要一样，此参数不是匹配条件，你当前项目叫什么名字就填什么，和提供者消费者角色无关，比如：kylin 应用调用了 morgan 应用的服务，则 kylin 项目配成 kylin，morgan 项目配成 morgan，可能 kylin 也提供其它服务给别人使用，但 kylin 项目永远配成 kylin，这样注册中心将显示 kylin 依赖于 morgan	1.0.16 以上版本
<dubbo:application>	version	application.version	string	可选		服务治理	当前应用的版本	2.2.0 以上版本

<dubbo:application>	owner	owner	string	可选		服务治理	应用负责人，用于服务治理，请填写负责人公司邮箱前缀	2.0.5以上版本
<dubbo:application>	organization	organization	string	可选		服务治理	组织名称(BU 或部门)，用于注册中心区分服务来源，此配置项建议不要使用 autoconfig，直接写死在配置中，比如 china,intl,itu,crm,asc,dw,aliexpress 等	2.0.0以上版本
<dubbo:application>	architecture	architecture	string	可选		服务治理	用于服务分层对应的架构。如，intl、china。不同的架构使用不同的分层。	2.0.7以上版本
<dubbo:application>	environment	environment	string	可选		服务治理	应用环境，如：develop/test/product，不同环境使用不同的缺省值，以及作为只用于开发测试功能的限制条件	2.0.0以上版本
<dubbo:application>	compiler	compiler	string	可选	javassist	性能优化	Java 字节码编译器，用于动态类的生成，可选：jdk 或 javassist	2.1.0以上版本
<dubbo:application>	logger	logger	string	可选	slf4j	性能优化	日志输出方式，可选：slf4j,jcl,log4j,jdk	2.2.0以上版本

<dubbo:module/>

模块信息配置：

配置类：com.alibaba.dubbo.config.ModuleConfig

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:module>	name	module	string	必填		服务治理	当前模块名称，用于注册中心计算模块间依赖关系	2.2.0以上版本
<dubbo:module>	version	module.version	string	可选		服务治理	当前模块的版本	2.2.0以上版本
<dubbo:module>	owner	owner	string	可		服	模块负责人，用于服务治理，请填写负责人公司邮箱前缀	2.2.0

				选		务治理		以上版本
<dubbo:module>	organization	organization	string	可选		服务治理	组织名称(BU 或部门)，用于注册中心区分服务来源，此配置项建议不要使用 autoconfig，直接写死在配置中，比如 china,intl,itu,crm,asc,dw,aliexpress 等	2.2.0 以上版本

<dubbo:provider/>

服务提供者缺省值配置：

配置类：com.alibaba.dubbo.config.ProviderConfig

说明：该标签为<dubbo:service>和<dubbo:protocol>标签的缺省值设置。

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:provider>	id		string	可选	dubbo	配置关联	协议 BeanId，可以在<dubbo:service proivder="">中引用此 ID	1.0.16 以上版本
<dubbo:provider>	protocol	<protocol>	string	可选	dubbo	性能调优	协议名称	1.0.16 以上版本
<dubbo:provider>	host	<host>	string	可选	自动查找本机 IP	服务发现	服务主机名，多网卡选择或指定 VIP 及域名时使用，为空则自动查找本机 IP，建议不要配置，让 Dubbo 自动获取本机 IP	1.0.16 以上版本
<dubbo:provider>	threads	threads	int	可选	100	性能调优	服务线程池大小(固定大小)	1.0.16 以上版本
<dubbo:provider>	payload	payload	int	可选	88388608(=8M)	性能调优	请求及响应数据包大小限制，单位：字节	2.0.0 以上版本
<dubbo:provider>	path	<path>	string	可选		服务发现	提供者上下文路径，为服务 path 的前缀	2.0.0 以上版本
<dubbo:provider>	server	server	string	可选	dubbo 协议缺省为 netty，http 协	性能	协议的服务器端实现类型，比如：dubbo 协议的 mina,netty 等，http 协议的	2.0.0 以上

					议缺省为 servlet	调优	jetty,servlet 等	版本
<dubbo:provider>	client	client	string	可选	dubbo 协议缺省为 netty	性能调优	协议的客户端实现类型，比如：dubbo 协议的 mina,netty 等	2.0.0 以上版本
<dubbo:provider>	codec	codec	string	可选	dubbo	性能调优	协议编码方式	2.0.0 以上版本
<dubbo:provider>	serialization	serialization	string	可选	dubbo 协议缺省为 hessian2，rmi 协议缺省为 java，http 协议缺省为 json	性能调优	协议序列化方式，当协议支持多种序列化方式时使用，比如：dubbo 协议的 dubbo,hessian2,java,compactedjava，以及 http 协议的 json,xml 等	2.0.5 以上版本
<dubbo:provider>	default		boolean	可选	false	配置关联	是否为缺省协议，用于多协议	1.0.16 以上版本
<dubbo:provider>	filter	service.filter	string	可选		性能调优	服务提供方远程调用过程拦截器名称，多个名称用逗号分隔	2.0.5 以上版本
<dubbo:provider>	listener	exporter.listener	string	可选		性能调优	服务提供方导出服务监听器名称，多个名称用逗号分隔	2.0.5 以上版本
<dubbo:provider>	threadpool	threadpool	string	可选	fixed	性能调优	线程池类型，可选：fixed/cached	2.0.5 以上版本
<dubbo:provider>	accepts	accepts	int	可选	0	性能调优	服务提供者最大可接受连接数	2.0.5 以上版本
<dubbo:provider>	version	version	string	可选	0.0.0	服务发现	服务版本，建议使用两位数字版本，如：1.0，通常在接口不兼容时版本号才需要升级	2.0.5 以上版本
<dubbo:provider>	group	group	string	可选		服务发现	服务分组，当一个接口有多个实现，可以用分组区分	2.0.5 以上版本
<dubbo:provider>	delay	delay	int	可	0	性	延迟注册服务时间(毫秒)- ，设为-1 时，	2.0.5

				选		能调优	表示延迟到 Spring 容器初始化完成时暴露服务	以上版本
<dubbo:provider>	timeout	default.timeout	int	可选	1000	性能调优	远程服务调用超时时间(毫秒)	2.0.5以上版本
<dubbo:provider>	retries	default.retries	int	可选	2	性能调优	远程服务调用重试次数，不包括第一次调用，不需要重试请设为 0	2.0.5以上版本
<dubbo:provider>	connections	default.connections	int	可选	0	性能调优	对每个提供者的最大连接数，rmi、http、hessian 等短连接协议表示限制连接数，dubbo 等长连接协表示建立的长连接个数	2.0.5以上版本
<dubbo:provider>	loadbalance	default.loadbalance	string	可选	random	性能调优	负载均衡策略，可选值：random,roundrobin,leastactive，分别表示：随机，轮循，最少活跃调用	2.0.5以上版本
<dubbo:provider>	async	default.async	boolean	可选	false	性能调优	是否缺省异步执行，不可靠异步，只是忽略返回值，不阻塞执行线程	2.0.5以上版本
<dubbo:provider>	stub	stub	boolean	可选	false	服务治理	设为 true，表示使用缺省代理类名，即：接口名 + Local 后缀。	2.0.5以上版本
<dubbo:provider>	mock	mock	boolean	可选	false	服务治理	设为 true，表示使用缺省 Mock 类名，即：接口名 + Mock 后缀。	2.0.5以上版本
<dubbo:provider>	token	token	boolean	可选	false	服务治理	令牌验证，为空表示不开启，如果为 true，表示随机生成动态令牌	2.0.5以上版本
<dubbo:provider>	registry	registry	string	可选	缺省向所有 registry 注册	配置关联	向指定注册中心注册，在多个注册中心时使用，值为<dubbo:registry>的 id 属性，多个注册中心 ID 用逗号分隔，如果不想将该服务注册到任何 registry，可将值设为 N/A	2.0.5以上版本
<dubbo:provider>	dynamic	dynamic	boolean	可选	true	服务治理	服务是否动态注册，如果设为 false，注册后将显示后 disable 状态，需人工启用，并且服务提供者停止时，也不会自动取消册，需人工禁用。	2.0.5以上版本

<dubbo:provider>	accesslog	accesslog	string/boolean	可选	false	服务治理	设为 true，将向 logger 中输出访问日志，也可填写访问日志文件路径，直接把访问日志输出到指定文件	2.0.5 以上版本
<dubbo:provider>	owner	owner	string	可选		服务治理	服务负责人，用于服务治理，请填写负责人公司邮箱前缀	2.0.5 以上版本
<dubbo:provider>	document	document	string	可选		服务治理	服务文档 URL	2.0.5 以上版本
<dubbo:provider>	weight	weight	int	可选		性能调优	服务权重	2.0.5 以上版本
<dubbo:provider>	executes	executes	int	可选	0	性能调优	服务提供者每服务每方法最大可并行执行请求数	2.0.5 以上版本
<dubbo:provider>	actives	default.actives	int	可选	0	性能调优	每服务消费者每服务每方法最大并发调用数	2.0.5 以上版本
<dubbo:provider>	proxy	proxy	string	可选	javassist	性能调优	生成动态代理方式，可选：jdk/javassist	2.0.5 以上版本
<dubbo:provider>	cluster	default.cluster	string	可选	failover	性能调优	集群方式，可选：failover/failfast/failsafe/failback/forking	2.0.5 以上版本
<dubbo:provider>	deprecated	deprecated	boolean	可选	false	服务治理	服务是否过时，如果设为 true，消费方引用时将打印服务过时警告 error 日志	2.0.5 以上版本
<dubbo:provider>	queues	queues	int	可选	0	性能调优	线程池队列大小，当线程池满时，排队等待执行的队列大小，建议不要设置，当线程程池时应立即失败，重试其它服务提供机器，而不是排队，除非有特殊需求。	2.0.5 以上版本
<dubbo:provider>	charset	charset	string	可选	UTF-8	性能调	序列化编码	2.0.5 以上版本

						优		
<dubbo:provider>	buffer	buffer	int	可选	8192	性能调优	网络读写缓冲区大小	2.0.5以上版本
<dubbo:provider>	iothreads	iothreads	int	可选	CPU + 1	性能调优	IO 线程池，接收网络读写中断，以及序列化和反序列化，不处理业务，业务线程池参见 threads 配置，此线程池和 CPU 相关，不建议配置。	2.0.5以上版本
<dubbo:provider>	telnet	telnet	string	可选		服务治理	所支持的 telnet 命令，多个命令用逗号分隔	2.0.5以上版本
<dubbo:service>	contextpath	contextpath	String	可选	缺省为空串	服务治理		2.0.6以上版本
<dubbo:provider>	layer	layer	string	可选		服务治理	服务提供者所在的分层。如：biz、dao、intl:web、china:acton。	2.0.7以上版本

<dubbo:consumer/>

服务消费者缺省值配置：

配置类：com.alibaba.dubbo.config.ConsumerConfig

说明：该标签为<dubbo:reference>标签的缺省值设置。

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:consumer>	timeout	default.timeout	int	可选	1000	性能调优	远程服务调用超时时间(毫秒)	1.0.16 以上版本
<dubbo:consumer>	retries	default.retries	int	可选	2	性能调优	远程服务调用重试次数，不包括第一次调用，不需要重试请设为 0	1.0.16 以上版本
<dubbo:consumer>	loadbalance	default.loadbalance	string	可选	random	性能调优	负载均衡策略，可选值：random,roundrobin,leastactive，分别表示：随机，轮循，最少活跃调用	1.0.16 以上版本

<dubbo:consumer>	async	default.async	boolean	可选	false	性能调优	是否缺省异步执行，不可靠异步，只是忽略返回值，不阻塞执行线程	2.0.0 以上版本
<dubbo:consumer>	connections	default.connections	int	可选	100	性能调优	每个服务对每个提供者的最大连接数，rmi、http、hessian 等短连接协议支持此配置，dubbo 协议长连接不支持此配置	1.0.16 以上版本
<dubbo:consumer>	generic	generic	boolean	可选	false	服务治理	是否缺省泛化接口，如果为泛化接口，将返回 GenericService	2.0.0 以上版本
<dubbo:consumer>	check	check	boolean	可选	true	服务治理	启动时检查提供者是否存在，true 报错，false 忽略	1.0.16 以上版本
<dubbo:consumer>	proxy	proxy	string	可选	javassist	性能调优	生成动态代理方式，可选：jdk/javassist	2.0.5 以上版本
<dubbo:consumer>	owner	owner	string	可选		服务治理	调用服务负责人，用于服务治理，请填写负责人公司邮箱前缀	2.0.5 以上版本
<dubbo:consumer>	actives	default.actives	int	可选	0	性能调优	每服务消费者每服务每方法最大并发调用数	2.0.5 以上版本
<dubbo:consumer>	cluster	default.cluster	string	可选	failover	性能调优	集群方式，可选：failover/failfast/failsafe/failback/forking	2.0.5 以上版本
<dubbo:consumer>	filter	reference.filter	string	可选		性能调优	服务消费方远程调用过程拦截器名称，多个名称用逗号分隔	2.0.5 以上版本
<dubbo:consumer>	listener	invoker.listener	string	可选		性能调优	服务消费方引用服务监听器名称，多个名称用逗号分隔	2.0.5 以上版本
<dubbo:consumer>	registry		string	可选	缺省向所有 registry 注册	配置关联	向指定注册中心注册，在多个注册中心时使用，值为<dubbo:registry>的 id 属性，多个注册中心 ID 用逗号分隔，如果不想将该服务注册到任何 registry，	2.0.5 以上版本

							可将值设为 N/A	
<dubbo:consumer>	layer	layer	string	可选		服务治理	服务调用者所在的分层。如：biz、dao、intl:web、china:acton。	2.0.7 以上版本
<dubbo:consumer>	init	init	boolean	可选	false	性能调优	是否在 afterPropertiesSet()时饥饿初始化引用，否则等到有人注入或引用该实例时再初始化。	2.0.10 以上版本
<dubbo:consumer>	cache	cache	string/boolean	可选		服务治理	以调用参数为 key，缓存返回结果，可选：lru, threadlocal, jcache 等	Dubbo2.1.0 及其以上版本支持
<dubbo:consumer>	validation	validation	boolean	可选		服务治理	是否启用 JSR303 标准注解验证，如果启用，将对方法参数上的注解进行校验	Dubbo2.1.0 及其以上版本支持

<dubbo:method/>

方法级配置：

配置类：com.alibaba.dubbo.config.MethodConfig

说明：该标签为<dubbo:service>或<dubbo:reference>的子标签，用于控制到方法级，

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:method>	name		string	必填		标识	方法名	1.0.8 以上版本
<dubbo:method>	timeout	<metodName>.timeout	int	可选	缺省为的 timeout	性能调优	方法调用超时时间(毫秒)	1.0.8 以上版本
<dubbo:method>	retries	<metodName>.retries	int	可选	缺省为<dubbo:reference>的 retries	性能调优	远程服务调用重试次数，不包括第一次调用，不需要重试请设为 0	2.0.0 以上版本
<dubbo:method>	loadbalance	<metodName>.loadbalance	string	可选	缺省为的 loadbalance	性能调优	负载均衡策略，可选值：random,roundrobin,leastactive，分别表示：随机，轮循，最少活跃调用	2.0.0 以上版本
<dubbo:method>	async	<metodName>.async	boolean	可选	缺省为<dubbo:reference>	性能	是否异步执行，不可靠异步，只是忽略返回值，不阻塞执行线程	1.0.9 以上版本

					的 async	调优		
<dubbo:method>	sent	<methodName>.sent	boolean	可选	true	性能调优	异步调用时，标记 sent=true 时，表示网络已发出数据	2.0.6 以上版本
<dubbo:method>	actives	<metodName>.actives	int	可选	0	性能调优	每服务消费者最大并发调用限制	2.0.5 以上版本
<dubbo:method>	executes	<metodName>.executes	int	可选	0	性能调优	每服务每方法最大使用线程数限制- -，此属性只在 <dubbo:method>作为 <dubbo:service>子标签时有效	2.0.5 以上版本
<dubbo:method>	deprecated	<methodName>.deprecated	boolean	可选	false	服务治理	服务方法是否过时，此属性只在 <dubbo:method>作为 <dubbo:service>子标签时有效	2.0.5 以上版本
<dubbo:method>	sticky	<methodName>.sticky	boolean	可选	false	服务治理	设置 true 该接口上的所有方法使用同一个 provider.如果需要更复杂的规则，请使用用路由	2.0.6 以上版本
<dubbo:method>	return	<methodName>.return	boolean	可选	true	性能调优	方法调用是否需要返回值,async 设置为 true 时才生效，如果设置为 true，则返回 future，或回调 onreturn 等方法，如果设置为 false，则请求发送成功后直接返回 Null	2.0.6 以上版本
<dubbo:method>	oninvoke	attribute 属性，不在 URL 中体现	String	可选		性能调优	方法执行前拦截	2.0.6 以上版本
<dubbo:method>	onreturn	attribute 属性，不在 URL 中体现	String	可选		性能调优	方法执行返回后拦截	2.0.6 以上版本
<dubbo:method>	onthrow	attribute 属性，不在 URL 中体现	String	可选		性能调优	方法执行有异常拦截	2.0.6 以上版本
<dubbo:method>	cache	<methodName>.cache	string/boolean	可选		服务治理	以调用参数为 key，缓存返回结果，可选：lru, threadlocal, jcache 等	Dubbo2.1.0 及其以上版本支持

<dubbo:method>	validation	<methodName>.validation	boolean	可选		服务治理	是否启用 JSR303 标准注解验证，如果启用，将对方法参数上的注解进行校验	Dubbo2.1.0及其以上版本支持

比如：

```
<dubbo:reference interface="com.xxx.XxxService">
    <dubbo:method name="findXxx" timeout="3000" retries="2" />
</dubbo:reference>
```

<dubbo:argument/>

方法参数配置：

配置类：com.alibaba.dubbo.config.ArgumentConfig

说明：该标签为<dubbo:method>的子标签，用于方法参数的特征描述，比如：

```
<dubbo:method name="findXxx" timeout="3000" retries="2">
    <dubbo:argument index="0" callback="true" />
<dubbo:method>
```

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:argument>	index		int	必填		标识	方法名	2.0.6以上版本
<dubbo:argument>	type		String	与index二选一		标识	通过参数类型查找参数的 index	2.0.6以上版本
<dubbo:argument>	callback	<metodName><index>.retries	boolean	可选		服务治理	参数是否为 callback 接口，如果为 callback，服务提供方将生成反向代理，可以从服务提供方反向调用消费方，通常用于事件推送.	2.0.6以上版本

<dubbo:parameter/>

选项参数配置：

配置类：java.util.Map

说明：该标签为<dubbo:protocol>或<dubbo:service>或<dubbo:provider>或<dubbo:reference>或<dubbo:consumer>的子标签，用于配置自定义参数，该配置项将作为扩展点设置自定义参数使用。

标签	属性	对应 URL 参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:parameter>	key	key	string	必填		服务治理	路由参数键	2.0.0 以上版本
<dubbo:parameter>	value	value	string	必填		服务治理	路由参数值	2.0.0 以上版本

3.1.3 xml 配置使用样例

xml 的配置使用样例，以及与 Springmvc 的集成 demo 。可参见源码程序包 busi-xml-server-client 和 busi-mvc 项目

3.2注解方式

注解方式的底层与 XML 一致，只是表现形式上的不同。目标都是将 Dubbo 基础信息配入，主要涉及以下五个必不可少的信息：**ApplicationConfig、ProtocolConfig 、 RegistryConfig、service、reference**

3.2.1 EnableDubbo 开启服务

@EnableDubbo：开启注解 Dubbo 功能 ，其中可以加入 scanBasePackages 属性配置包扫描的路径，用于扫描并注册 bean。其中 封装了组件 @DubboComponentScan，来扫描 Dubbo @Service 注解暴露 Dubbo 服务，以及扫描 Dubbo @Reference 字段或者方法注入 Dubbo 服务代理。

其它 Dubbo 三种公共信息的配置，有两种方式，根据自己喜好选用

3.2.2 Configuration 方式配置公共信息

@Configuration 方式：分别将 **ApplicationConfig、ProtocolConfig 、 RegistryConfig** 创建到 IOC 容器中即可，如下：

```
@Configuration
@EnableDubbo(scanBasePackages = "com.enjoy.service")
class ProviderConfiguration {
    @Bean
    public ApplicationConfig applicationConfig() {
        ApplicationConfig applicationConfig = new ApplicationConfig();
        applicationConfig.setName("busi-provider");
        return applicationConfig;
    }
    @Bean
    public RegistryConfig registryConfig() {
        RegistryConfig registryConfig = new RegistryConfig();
        registryConfig.setProtocol("zookeeper");
        registryConfig.setAddress("192.168.0.128");
        registryConfig.setPort(2181);
        return registryConfig;
    }
    @Bean
    public ProtocolConfig protocolConfig() {
        ProtocolConfig protocolConfig = new ProtocolConfig();
```

```
        protocolConfig.setName("dubbo");
        protocolConfig.setPort(20880);
        return protocolConfig;
    }
}
```

3.2.3 Property 方式自动装配公共信息

Property 方式：使用 Springboot 属性文件方式，由 Dubbo 自动将文件信息配置入容器，示例如下：

```
@Configuration
@EnableDubbo(scanBasePackages = "com.enjoy.service")
@PropertySource("classpath:/dubbo-provider.properties")
static class ProviderConfiguration {
}
```

dubbo-provider.properties 文件内容

```
dubbo.application.name=busi-provider
dubbo.registry.address=zookeeper://192.168.0.128:2181
dubbo.protocol.name=dubbo
dubbo.protocol.port=20880
```

完整的项目示例，参见 busi-xml-server-client 源码包