# Dubbo XML 配置

| **标签** | **用途** | **解释** |
| --- | --- | --- |
| <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:module/> | 模块配置 | 用于配置当前模块信息，可选 |
| <dubbo:registry/> | 注册中心配置 | 用于配置连接注册中心相关信息 |
| <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/> | 提供方配置 | 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/> | 参数配置 | 用于指定方法参数配置 |

配置覆盖关系：

以 timeout 为例，显示了配置的查找顺序，其它 retries, loadbalance, actives 等类似：

方法级优先，接口级次之，全局配置再次之。

如果级别一样，则消费方优先，提供方次之。

其中，服务提供方配置，通过 URL 经由注册中心传递给消费方。

建议由服务提供方设置超时，因为一个方法需要执行多长时间，服务提供方更清楚，如果一个消费方同时引用多个服务，就不需要关心每个服务的超时设置。

# Dubbo 属性配置

覆盖策略：

JVM 启动 -D 参数优先，这样可以使用户在部署和启动时进行参数重写，比如在启动时需改变协议的端口。

XML 次之，如果在 XML 中有配置，则 dubbo.properties 中的相应配置项无效。

Properties 最后，相当于缺省值，只有 XML 没有配置时，dubbo.properties 的相应配置项才会生效，通常用于共享公共配置，比如应用名。

# 集群容错控制

Failfast Cluster

快速失败，只发起一次调用，失败立即报错。通常用于非幂等性的写操作，比如新增记录。

Failsafe Cluster

失败安全，出现异常时，直接忽略。通常用于写入审计日志等操作。

Failback Cluster

失败自动恢复，后台记录失败请求，定时重发。通常用于消息通知操作。

Forking Cluster

并行调用多个服务器，只要一个成功即返回。通常用于实时性要求较高的读操作，但需要浪费更多服务资源。可通过 forks="2" 来设置最大并行数。

Broadcast Cluster

广播调用所有提供者，逐个调用，任意一台报错则报错 2。通常用于通知所有提供者更新缓存或日志等本地资源信息。

# 负载均衡

Random LoadBalance

随机，按权重设置随机概率。

在一个截面上碰撞的概率高，但调用量越大分布越均匀，而且按概率使用权重后也比较均匀，有利于动态调整提供者权重。

RoundRobin LoadBalance

轮循，按公约后的权重设置轮循比率。

存在慢的提供者累积请求的问题，比如：第二台机器很慢，但没挂，当请求调到第二台时就卡在那，久而久之，所有请求都卡在调到第二台上。

LeastActive LoadBalance

最少活跃调用数，相同活跃数的随机，活跃数指调用前后计数差。

使慢的提供者收到更少请求，因为越慢的提供者的调用前后计数差会越大。

ConsistentHash LoadBalance

一致性 Hash，相同参数的请求总是发到同一提供者。

当某一台提供者挂时，原本发往该提供者的请求，基于虚拟节点，平摊到其它提供者，不会引起剧烈变动。

缺省只对第一个参数 Hash，如果要修改，请配置

<dubbo:parameter key="hash.arguments" value="0,1" />

缺省用 160 份虚拟节点，如果要修改，请配置

<dubbo:parameter key="hash.nodes" value="320" />

# 线程模型

如果事件处理的逻辑能迅速完成，并且不会发起新的 IO 请求，比如只是在内存中记个标识，则直接在 IO 线程上处理更快，因为减少了线程池调度。

但如果事件处理逻辑较慢，或者需要发起新的 IO 请求，比如需要查询数据库，则必须派发到线程池，否则 IO 线程阻塞，将导致不能接收其它请求。

如果用 IO 线程处理事件，又在事件处理过程中发起新的 IO 请求，比如在连接事件中发起登录请求，会报“可能引发死锁”异常，但不会真死锁。

<dubbo:protocol name="dubbo" dispatcher="all" threadpool="fixed" threads="100" />

Dispatcher:

all 所有消息都派发到线程池，包括请求，响应，连接事件，断开事件，心跳等。

direct 所有消息都不派发到线程池，全部在 IO 线程上直接执行。

message 只有请求响应消息派发到线程池，其它连接断开事件，心跳等消息，直接在 IO 线程上执行。

execution 只请求消息派发到线程池，不含响应，响应和其它连接断开事件，心跳等消息，直接在 IO 线程上执行。

connection 在 IO 线程上，将连接断开事件放入队列，有序逐个执行，其它消息派发到线程池。

ThreadPool:

fixed 固定大小线程池，启动时建立线程，不关闭，一直持有。(缺省)

cached 缓存线程池，空闲一分钟自动删除，需要时重建。

limited 可伸缩线程池，但池中的线程数只会增长不会收缩。只增长不收缩的目的是为了避免收缩时突然来了大流量引起的性能问题。

# 直连提供者

在开发及测试环境下，经常需要绕过注册中心，只测试指定服务提供者，这时候可能需要点对点直连，点对点直联方式，将以服务接口为单位，忽略注册中心的提供者列表，A 接口配置点对点，不影响 B 接口从注册中心获取列表。

xml配置：

<dubbo:reference id="xxxService" interface="com.alibaba.xxx.XxxService" url="dubbo://localhost:20890" />

通过文件映射：

果服务比较多，也可以用文件映射，用 -Ddubbo.resolve.file 指定映射文件路径，此配置优先级高于 <dubbo:reference> 中的配置。

# 多协议

Dubbo 允许配置多协议，在不同服务上支持不同协议或者同一服务上同时支持多种协议。

多协议暴露服务。中间用逗号隔开。

<!-- 多协议配置 -->

<dubbo:protocol name="dubbo" port="20880" />

<dubbo:protocol name="hessian" port="8080" />

<!-- 使用多个协议暴露服务 -->

<dubbo:service id="helloService" interface="com.alibaba.hello.api.HelloService" version="1.0.0" protocol="dubbo,hessian" />

# 多注册中心注册

Dubbo 支持同一服务向多注册中心同时注册，或者不同服务分别注册到不同的注册中心上去，甚至可以同时引用注册在不同注册中心上的同名服务。

<!-- 多注册中心配置 -->

<dubbo:registry id="hangzhouRegistry" address="10.20.141.150:9090" />

<dubbo:registry id="qingdaoRegistry" address="10.20.141.151:9010" default="false" />

<!-- 向多个注册中心注册 -->

<dubbo:service interface="com.alibaba.hello.api.HelloService" version="1.0.0" ref="helloService" registry="hangzhouRegistry,qingdaoRegistry" />

不同服务使用不同注册中心：

<!-- 多注册中心配置 -->

<dubbo:registry id="chinaRegistry" address="10.20.141.150:9090" />

<dubbo:registry id="intlRegistry" address="10.20.154.177:9010" default="false" />

<!-- 向中文站注册中心注册 -->

<dubbo:service interface="com.alibaba.hello.api.HelloService" version="1.0.0" ref="helloService" registry="chinaRegistry" />

<!-- 向国际站注册中心注册 -->

<dubbo:service interface="com.alibaba.hello.api.DemoService" version="1.0.0" ref="demoService" registry="intlRegistry" />

多注册中心引用：

<!-- 多注册中心配置 -->

<dubbo:registry id="chinaRegistry" address="10.20.141.150:9090" />

<dubbo:registry id="intlRegistry" address="10.20.154.177:9010" default="false" />

<!-- 引用中文站服务 -->

<dubbo:reference id="chinaHelloService" interface="com.alibaba.hello.api.HelloService" version="1.0.0" registry="chinaRegistry" />

<!-- 引用国际站站服务 -->

<dubbo:reference id="intlHelloService" interface="com.alibaba.hello.api.HelloService" version="1.0.0" registry="intlRegistry" />

# 服务分组

当一个接口有多种实现时，可以用 group 区分。

服务

<dubbo:service group="feedback" interface="com.xxx.IndexService" />

<dubbo:service group="member" interface="com.xxx.IndexService" />

引用

<dubbo:reference id="feedbackIndexService" group="feedback" interface="com.xxx.IndexService" />

<dubbo:reference id="memberIndexService" group="member" interface="com.xxx.IndexService" />

引用任意组

<dubbo:reference id="barService" interface="com.foo.BarService" group="\*" />

# 多版本

当一个接口实现，出现不兼容升级时，可以用版本号过渡，版本号不同的服务相互间不引用。

老版本服务提供者配置：

<dubbo:service interface="com.foo.BarService" version="1.0.0" />

新版本服务提供者配置：

<dubbo:service interface="com.foo.BarService" version="2.0.0" />

老版本服务消费者配置：

<dubbo:reference id="barService" interface="com.foo.BarService" version="1.0.0" />

新版本服务消费者配置：

<dubbo:reference id="barService" interface="com.foo.BarService" version="2.0.0" />

# 分组聚合

比如菜单服务，接口一样，但有多种实现，用group区分，现在消费方需从每种group中调用一次返回结果，合并结果返回，这样就可以实现聚合菜单项。

搜索所有分组

<dubbo:reference interface="com.xxx.MenuService" group="\*" merger="true" />

合并指定分组

<dubbo:reference interface="com.xxx.MenuService" group="aaa,bbb" merger="true" />

指定方法合并结果，其它未指定的方法，将只调用一个 Group

<dubbo:reference interface="com.xxx.MenuService" group="\*">

<dubbo:method name="getMenuItems" merger="true" /></dubbo:service>

某个方法不合并结果，其它都合并结果

<dubbo:reference interface="com.xxx.MenuService" group="\*" merger="true">

<dubbo:method name="getMenuItems" merger="false" /></dubbo:service>

指定合并策略，缺省根据返回值类型自动匹配，如果同一类型有两个合并器时，需指定合并器的名称

<dubbo:reference interface="com.xxx.MenuService" group="\*">

<dubbo:method name="getMenuItems" merger="mymerge" /></dubbo:service>

指定合并方法，将调用返回结果的指定方法进行合并，合并方法的参数类型必须是返回结果类型本身

<dubbo:reference interface="com.xxx.MenuService" group="\*">

<dubbo:method name="getMenuItems" merger=".addAll" /></dubbo:service>

# 回声测试

回声测试用于检测服务是否可用，回声测试按照正常请求流程执行，能够测试整个调用是否通畅，可用于监控。

所有服务自动实现 EchoService 接口，只需将任意服务引用强制转型为 EchoService，即可使用。

// 远程服务引用

MemberService memberService = ctx.getBean("memberService");

EchoService echoService = (EchoService) memberService; // 强制转型为EchoService

// 回声测试可用性

String status = echoService.$echo("OK");

assert(status.equals("OK"));

# 上下文信息

上下文中存放的是当前调用过程中所需的环境信息。所有配置信息都将转换为 URL 的参数。

RpcContext 是一个 ThreadLocal 的临时状态记录器，当接收到 RPC 请求，或发起 RPC 请求时，RpcContext 的状态都会变化。比如：A 调 B，B 再调 C，则 B 机器上，在 B 调 C 之前，RpcContext 记录的是 A 调 B 的信息，在 B 调 C 之后，RpcContext 记录的是 B 调 C 的信息。

# 异步调用

基于 NIO 的非阻塞实现并行调用，客户端不需要启动多线程即可完成并行调用多个远程服务，相对多线程开销较小。

<dubbo:reference id="fooService" interface="com.alibaba.foo.FooService">

<dubbo:method name="findFoo" async="true" /></dubbo:reference><dubbo:reference id="barService" interface="com.alibaba.bar.BarService">

<dubbo:method name="findBar" async="true" /></dubbo:reference>

你可以设置是否等待消息发出。

* sent="true" 等待消息发出，消息发送失败将抛出异常。
* sent="false" 不等待消息发出，将消息放入 IO 队列，即刻返回。

<dubbo:method name="findFoo" async="true" sent="true" />

如果你只是想异步，完全忽略返回值，可以配置 return="false"

<dubbo:method name="findFoo" async="true" return="false" />

# 延迟暴露

如果你的服务需要预热时间，比如初始化缓存，等待相关资源就位等，可以使用 delay 进行延迟暴露。

延迟 5 秒暴露服务

<dubbo:service delay="5000" />

延迟到 Spring 初始化完成后，再暴露服务

<dubbo:service delay="-1" />

为了规避spring的死锁问题：

1. 强烈建议不要在服务的实现类中有 applicationContext.getBean() 的调用，全部采用 IoC 注入的方式使用 Spring的Bean。
2. 如果实在要调 getBean()，可以将 Dubbo 的配置放在 Spring 的最后加载。
3. 如果不想依赖配置顺序，可以使用<dubbo:provider deplay=”-1”/>，使 Dubbo 在 Spring 容器初始化完后，再暴露服务。
4. 如果大量使用 getBean()，相当于已经把 Spring 退化为工厂模式在用，可以将 Dubbo 的服务隔离单独的 Spring 容器。

# 并发控制

样例 1

限制 com.foo.BarService 的每个方法，服务器端并发执行（或占用线程池线程数）不能超过 10 个：

<dubbo:service interface="com.foo.BarService" executes="10" />

样例 2

限制 com.foo.BarService 的 sayHello 方法，服务器端并发执行（或占用线程池线程数）不能超过 10 个：

<dubbo:service interface="com.foo.BarService">

<dubbo:method name="sayHello" executes="10" />

</dubbo:service>

样例 3

限制 com.foo.BarService 的每个方法，每客户端并发执行（或占用连接的请求数）不能超过 10 个：

<dubbo:service interface="com.foo.BarService" actives="10" />

或

<dubbo:reference interface="com.foo.BarService" actives="10" />

样例 4

限制 com.foo.BarService 的 sayHello 方法，每客户端并发执行（或占用连接的请求数）不能超过 10 个：

<dubbo:service interface="com.foo.BarService">

<dubbo:method name="sayHello" actives="10" />

</dubbo:service>

或

<dubbo:reference interface="com.foo.BarService">

<dubbo:method name="sayHello" actives="10" />

</dubbo:service>

如果 <dubbo:service> 和 <dubbo:reference> 都配了actives，<dubbo:reference> 优先。

# 连接控制

## **服务端连接控制**

限制服务器端接受的连接不能超过 10 个：

<dubbo:provider protocol="dubbo" accepts="10" />

或

<dubbo:protocol name="dubbo" accepts="10" />

## **客户端连接控制**

限制客户端服务使用连接不能超过 10 个：

<dubbo:reference interface="com.foo.BarService" connections="10" />

或

<dubbo:service interface="com.foo.BarService" connections="10" />

如果<dubbo:service>和<dubbo:reference>都配了 connections，<dubbo:reference>优先

# 令牌验证

通过令牌验证在注册中心控制权限，以决定要不要下发令牌给消费者，可以防止消费者绕过注册中心访问提供者，另外通过注册中心可灵活改变授权方式，而不需修改或升级提供者。

可以全局设置开启令牌验证：

<!--随机token令牌，使用UUID生成--><dubbo:provider interface="com.foo.BarService" token="true" />

或

<!--固定token令牌，相当于密码--><dubbo:provider interface="com.foo.BarService" token="123456" />

也可在服务级别设置：

<!--随机token令牌，使用UUID生成--><dubbo:service interface="com.foo.BarService" token="true" />

或

<!--固定token令牌，相当于密码--><dubbo:service interface="com.foo.BarService" token="123456" />

还可在协议级别设置：

<!--随机token令牌，使用UUID生成--><dubbo:protocol name="dubbo" token="true" />

或

<!--固定token令牌，相当于密码--><dubbo:protocol name="dubbo" token="123456" />

# schema 配置参考手册

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。

### dubbo:service服务提供者暴露服务配置

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| interface |  | class | **必填** |  | 服务发现 | 服务接口名 |
| ref |  | object | **必填** |  | 服务发现 | 服务对象实现引用 |
| 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 |
| 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 |
| 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) |
| 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 |
| timeout | timeout | int | 可选 | 1000 | 性能调优 | 远程服务调用超时时间(毫秒) |
| 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 |
| 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 |
| 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 |
| 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 |
| 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对象，构造函数如：public XxxServiceLocal(XxxService xxxService) |
| 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。 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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。 |
| 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 |
| 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 |
| document | document | string | 可选 |  | 服务治理 | 服务文档URL |
| weight | weight | int | 可选 |  | 性能调优 | 服务权重 |
| 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 |
| proxy | proxy | string | 可选 | javassist | 性能调优 | 生成动态代理方式，可选：jdk/javassist |
| cluster | cluster | string | 可选 | failover | 性能调优 | 集群方式，可选：failover/failfast/failsafe/failback/forking |
| 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 |
| 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 |
| 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 |
| 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。 |
| 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 |

### dubbo:reference服务消费者引用服务配置

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| id |  | string | **必填** |  | 配置关联 | 服务引用BeanId |
| interface |  | class | **必填** |  | 服务发现 | 服务接口名 |
| version | version | string | 可选 |  | 服务发现 | 服务版本，与服务提供者的版本一致 |
| 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 |
| 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) |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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) |
| 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。 |
| 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 |
| 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 |
| proxy | proxy | boolean | 可选 | javassist | 性能调优 | 选择动态代理实现策略，可选：javassist, jdk |
| 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。 |
| 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 |
| 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 |
| 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 |
| cluster | cluster | string | 可选 | failover | 性能调优 | 集群方式，可选：failover/failfast/failsafe/failback/forking |
| 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 |
| 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 |
| 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。 |
| 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。 |
| 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。 |

### dubbo:protocol 服务提供者协议配置。

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。

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| 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。 |
| name | <protocol> | string | **必填** | dubbo | 性能调优 | 协议名称 |
| 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 |
| 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 |
| threadpool | threadpool | string | 可选 | fixed | 性能调优 | 线程池类型，可选：fixed/cached |
| threads | threads | int | 可选 | 100 | 性能调优 | 服务线程池大小(固定大小) |
| iothreads | threads | int | 可选 | cpu个数+1 | 性能调优 | io线程池大小(固定大小) |
| accepts | accepts | int | 可选 | 0 | 性能调优 | 服务提供方最大可接受连接数 |
| 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 |
| codec | codec | string | 可选 | dubbo | 性能调优 | 协议编码方式 |
| 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 |
| 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 |
| path | <path> | string | 可选 |  | 服务发现 | 提供者上下文路径，为服务path的前缀 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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。 |
| charset | charset | string | 可选 | UTF-8 | 性能调优 | 序列化编码 |
| buffer | buffer | int | 可选 | 8192 | 性能调优 | 网络读写缓冲区大小 |
| 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 |
| 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 |
| register | register | boolean | 可选 | true | 服务治理 | 该协议的服务是否注册 |
| contextpath | contextpath | String | 可选 | 缺省为空串 | 服务治理 |  |

### dubbo:registry 注册中心配置。

同时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。

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |
| address | <host:port> | string | **必填** |  | 服务发现 | 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<dubbo:registry>标签 |
| 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 |
| 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 |
| 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 |
| 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 |
| transport | registry.transporter | string | 可选 | netty | 性能调优 | 网络传输方式，可选mina,netty |
| timeout | registry.timeout | int | 可选 | 5000 | 性能调优 | 注册中心请求超时时间(毫秒) |
| 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。 |
| 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 |
| wait | registry.wait | int | 可选 | 0 | 性能调优 | 停止时等待通知完成时间(毫秒) |
| check | check | boolean | 可选 | true | 服务治理 | 注册中心不存在时，是否报错 |
| register | register | boolean | 可选 | true | 服务治理 | 是否向此注册中心注册服务，如果设为false，将只订阅，不注册 |
| 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 |
| 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。 |

### dubbo:monitor 监控中心配置。

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| 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。 |
| 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" |

### dubbo:application 应用信息配置。

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |
| version | application.version | string | 可选 |  | 服务治理 | 当前应用的版本 |
| 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 |
| 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 |
| 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。 |
| 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 |
| 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 |
| logger | logger | string | 可选 | slf4j | 性能优化 | 日志输出方式，可选：slf4j,jcl,log4j,jdk |

### dubbo:module模块信息配置

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |
| version | module.version | string | 可选 |  | 服务治理 | 当前模块的版本 |
| owner | owner | string | 可选 |  | 服务治理 | 模块负责人，用于服务治理，请填写负责人公司邮箱前缀 |
| 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 |

### dubbo:provider服务提供者缺省值配置

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

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| id |  | string | 可选 | dubbo | 配置关联 | 协议BeanId，可以在<dubbo:service proivder="">中引用此ID |
| protocol | <protocol> | string | 可选 | dubbo | 性能调优 | 协议名称 |
| 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 |
| threads | threads | int | 可选 | 100 | 性能调优 | 服务线程池大小(固定大小) |
| 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 |
| path | <path> | string | 可选 |  | 服务发现 | 提供者上下文路径，为服务path的前缀 |
| 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 |
| 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 |
| codec | codec | string | 可选 | dubbo | 性能调优 | 协议编码方式 |
| 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 |
| default |  | boolean | 可选 | false | 配置关联 | 是否为缺省协议，用于多协议 |
| 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 |
| 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 |
| threadpool | threadpool | string | 可选 | fixed | 性能调优 | 线程池类型，可选：fixed/cached |
| accepts | accepts | int | 可选 | 0 | 性能调优 | 服务提供者最大可接受连接数 |
| 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 |
| 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 |
| 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 |
| timeout | default.timeout | int | 可选 | 1000 | 性能调优 | 远程服务调用超时时间(毫秒) |
| 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 |
| 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 |
| 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 |
| 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 |
| 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。 |
| 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。 |
| 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 |
| 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 |
| 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。 |
| 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 |
| 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 |
| document | document | string | 可选 |  | 服务治理 | 服务文档URL |
| weight | weight | int | 可选 |  | 性能调优 | 服务权重 |
| 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 |
| 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 |
| proxy | proxy | string | 可选 | javassist | 性能调优 | 生成动态代理方式，可选：jdk/javassist |
| cluster | default.cluster | string | 可选 | failover | 性能调优 | 集群方式，可选：failover/failfast/failsafe/failback/forking |
| 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 |
| 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。 |
| charset | charset | string | 可选 | UTF-8 | 性能调优 | 序列化编码 |
| buffer | buffer | int | 可选 | 8192 | 性能调优 | 网络读写缓冲区大小 |
| 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。 |
| 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 |
| <dubbo:service> | contextpath | contextpath | String | 可选 | 缺省为空串 | 服务治理 |
| 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。 |

### dubbo:consumer服务消费者缺省值配置

同时该标签为 <dubbo:reference> 标签的缺省值设置。

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| timeout | default.timeout | int | 可选 | 1000 | 性能调优 | 远程服务调用超时时间(毫秒) |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| proxy | proxy | string | 可选 | javassist | 性能调优 | 生成动态代理方式，可选：jdk/javassist |
| 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 |
| 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 |
| cluster | default.cluster | string | 可选 | failover | 性能调优 | 集群方式，可选：failover/failfast/failsafe/failback/forking |
| 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 |
| 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 |
| 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 |
| 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。 |
| 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。 |
| 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 |
| 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 |

### dubbo:method方法级配置。

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

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| name |  | string | **必填** |  | 标识 | 方法名 |
| 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) |
| 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 |
| 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 |
| async | <metodName>.async | boolean | 可选 | 缺省为<dubbo:reference>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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| 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 |
| oninvoke | attribute属性，不在URL中体现 | String | 可选 |  | 性能调优 | 方法执行前拦截 |
| 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 |
| 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 |
| 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 |
| 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 |

### dubbo:argument方法参数配置。

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

<dubbo:method name="findXxx" timeout="3000" retries="2">

<dubbo:argument index="0" callback="true" /></dubbo:method>

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| index |  | int | **必填** |  | 标识 | 方法名 |
| type |  | String | 与index二选一 |  | 标识 | 通过参数类型查找参数的index |
| 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. |

### dubbo:parameter选项参数配置

同时该标签为<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。

| **属性** | **对应URL参数** | **类型** | **是否必填** | **缺省值** | **作用** | **描述** |
| --- | --- | --- | --- | --- | --- | --- |
| key | key | string | **必填** |  | 服务治理 | 路由参数键 |
| value | value | string | **必填** |  | 服务治理 | 路由参数值 |

例如：

<dubbo:protocol name="napoli">

<dubbo:parameter key="http://10.20.160.198/wiki/display/dubbo/napoli.queue.name" value="xxx" /></dubbo:protocol>