# DUBBO学习笔记

[DUBBO学习笔记 1](#_Toc20109)

[一、 环境部署 2](#_Toc8936)

[① Tomcat部署: 2](#_Toc16323)

[② Zookeeper部署: 3](#_Toc12580)

[③ Dubbo部署 3](#_Toc1046)

[④ 集群 3](#_Toc22828)

[二、 Dubbo简介 4](#_Toc26264)

[① Dubbo是1个散布式服务框 4](#_Toc29943)

[② 节点角色说明： 4](#_Toc20452)

[③ 调用关系说明： 5](#_Toc28463)

[④    dubbo:registry  5](#_Toc31374)

[⑤   dubbo:service 5](#_Toc28764)

[⑥ 开发者模式： 5](#_Toc3102)

[三、 DUBBO服务设计规则: 6](#_Toc12935)

[1.设计方式 6](#_Toc16044)

[2.接口类型 7](#_Toc5840)

[3.设计原则 7](#_Toc26779)

[四、 DUBBO 常用配置 9](#_Toc13510)

[五、超时配置 10](#_Toc23310)

# 环境部署

## **Tomcat部署:**

1. 下载：

wget https://edelivery.oracle.com/otn-pub/java/jdk/7u45-b18/jdk-7u45-linux-x64.rpm

wget <http://mirrors.shuosc.org/apache/tomcat/tomcat-7/v7.0.81/bin/apache-tomcat-7.0.81-windows-x64.zip>

1. 解压:

tar -zxvf apache-tomcat-7.0.81.tar.gz

mv apache-tomcat-7.0.81 tomcat

1. 防火墙:

firewall-cmd --zone=public --add-port=8080/tcp --permanent

firewall-cmd --reload

## **Zookeeper部署:**

$ tar -zxvf zookeeper-3.4.6.tar.gz

$ cd zookeeper/

$ mkdri data

$ mkdir logs

dataDir=/home/hadoop/zookeeper/data

dataLogDir=/home/hadoop/zookeeper/logs

$ ./zkServer.sh start

$ ./zkServer.sh status （查看状态）

$ ./zkServer.sh stop（关闭）

## **Dubbo部署**

War包解压ROOT下

## **集群**

配置:

|  |
| --- |
| vi conf/zoo.cfg |

如果不需要集群，zoo.cfg的内容如下：(其中data目录需改成你真实输出目录)

|  |
| --- |
| tickTime=2000  initLimit=10  syncLimit=5  dataDir=/home/dubbo/zookeeper-3.3.3/data  clientPort=2181 |

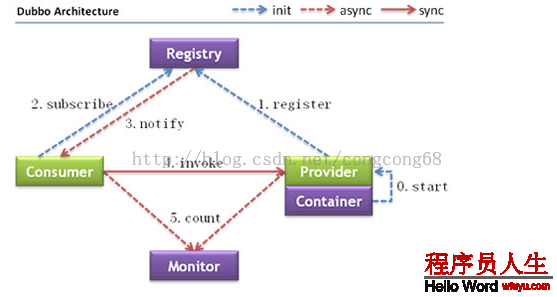
如果需要集群，zoo.cfg的内容如下：(其中data目录和server地址需改成你真实部署机器的信息)

|  |
| --- |
| tickTime=2000  initLimit=10  syncLimit=5  dataDir=/home/dubbo/zookeeper-3.3.3/data  clientPort=2181  server.1=10.20.153.10:2555:3555  server.2=10.20.153.11:2555:3555 |

并在data目录下放置myid文件：(上面zoo.cfg中的dataDir)

|  |
| --- |
| mkdir data  vi myid |

# **Dubbo简介**

1. **Dubbo是1个散布式服务框**架,

1. **节点角色说明：**
2. Provider: 暴露服务的服务提供方。
3. Consumer: 调用远程服务的服务消费方。
4. Registry: 服务注册与发现的注册中心。
5. Monitor: 统计服务的调用次调和调用时间的监控中心。
6. Container: 服务运行容器。

1. **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，向注册中心定阅自己所需的服务。

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1台提供者进行调用，如果调用失败，再选另外一台调用。

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1次统计数据到监控中心。

1. **dubbo:registry**

      1）register是不是向此注册中心注册服务，如果设为false，将只定阅，不注册。

      2）check注册中心不存在时，是不是报错。

      3）subscribe是不是向此注册中心定阅服务，如果设为false，将只注册，不定阅。

      4）timeout注册中心要求超时时间(毫秒)。

      5）address可以Zookeeper集群配置，地址可以多个以逗号隔开等。

1. **dubbo:service**

     1）interface服务接口的路径

     2）ref援用对应的实现类的Bean的ID

     3）registry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

     4）register 默许true ，该协议的服务是不是注册到注册中心。

1. **开发者模式：**
2. 在consumer配置指定服务器：

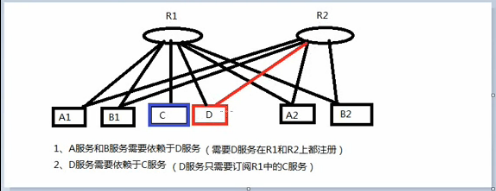
url:<dubbo:reference interface="com.dubbo.service.TestService" id="testService" check="false" url="dubbo://localhost:20880"/>

1. provider配置只订阅不注册：

register是不是向此注册中心注册服务，如果设为false，将只定阅，不注册。

1. provider配置只不注册不订阅：

subscribe是不是向此注册中心定阅服务，如果设为false，将只注册，不定阅。



# **DUBBO服务设计规则:**

### 1.设计方式

action->facade->biz->dao

好的Dubbo服务接口设计，并非只是纯粹的接口服务化

### 2.接口类型

简单的数据查询接口:action.facade、dao(例根据Id查询记录)

带业务逻辑的数据查询接口:action、facade、biz、dao(复杂的查询，带业务逻辑)

简单的数据写入接口：action、facade、dao(简单数据插入)

带业务逻辑的数据写入接口:action、facade、biz、dao(有业务逻辑的数据处理)

同步接口

异步接口

### 3.设计原则

1. 服务接口尽可能大粒度，每个服务方法应代表一个功能，而不是某功能的一个步骤，否则将地面临分布式事务问题，

Dubbo暂未提供分布式事务支持，同时可以减少系统间的网络交互

1. 服务接口建议以业务场景为单位划分，并对相近的业务做抽象,防止接口数量爆增(爆炸)。

例:某一个接口有多个实现，做成一个接口，再在dubbo分组中多实现

1. 不建议使用过于抽象的通用接口，如Map query(Map),这样的接口没有明确语义，会给后期维护带来不便
2. 接口版本:
3. 每个接口应定义版本号，为后续不兼容升级提供可能

如:

1. **[html]** [view plain](http://blog.csdn.net/l1028386804/article/details/72511650" \o "view plain" \t "http://blog.csdn.net/l1028386804/article/details/_blank) [copy](http://blog.csdn.net/l1028386804/article/details/72511650" \o "copy" \t "http://blog.csdn.net/l1028386804/article/details/_blank)

**<dubbo:service** interface="com.xxService" version="1.0"**/>**

1. 接口兼容性:

服务接口增加方法，或服务模型增加字段，可向后兼容；

删除方法或删除字段，将不兼容，枚举类型新增字段也不兼容，需要通过变更版本号升级。

1. 异常处理:
2. 建议使用异常汇报错误，而不是返回错误码，异常信息能携带更多的信息，以及语义更友好。
3. 如果担心性能问题，在必要时，可以通过override掉异常类的finlllnStackTrace()方法为空方法，使其不拷贝栈信息。
4. 查询方法不建议抛出checked异常，否则调用 方在查询 时将过多的try...catch,并且不能进行有效处理。
5. 服务提供方不应将DAO或者SQL等异常抛给消费方，应在服务实现中对消费方不关心的异常进行包装，否则可能出现消费方无法反序列化相应异常
6. 必要的接口输入参数校验
7. 在Provider上尽量多配置Consumer端属性:

原因如下:

1. 作为服务的提供者，比服务使用方更清楚服务性能参数，如调用的超时时间，合理的重试次数，并发控制数量，负载均衡 ，等等
2. 在Provider配置后，Consumer不配置则会使用Provider的配置值 ，
3. 即Provider配置可以作为Comsumer的缺省值，否则，Consumer会使用Consumer端的全局设置，这对于Provider不可控的，并且往往是不合理的
4. Provider上尽量多配置Consumer端的属性，让Provider实现者一开始就思考Provider服务特点、服务质量的问题

样例:



# **DUBBO 常用配置**

1. <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。

eg、<dubbo:service ref="demoService" interface="com.unj.dubbotest.provider.DemoService" />

1. <dubbo:reference/> 引用服务配置，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。

eg、<dubbo:reference id="demoService" interface="com.unj.dubbotest.provider.DemoService" />

1. <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。

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

1. <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。

eg、<dubbo:application name="xixi\_provider" />

<dubbo:application name="hehe\_consumer" />

1. <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。

eg、<dubbo:registry address="zookeeper://192.168.2.249:2181" />

1. <dubbo:monitor/> 监控中心配置，用于配置连接监控中心相关信息，可选。
2. <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。

**五、超时配置**  
