# Dubbo

各个应用节点中的url管理维护很困难、 依赖关系很模糊

每个应用节点的性能、访问量、响应时间，没办法评估

# Dubbo的使用入门

**dubbo://177.1.1.82:20880/com.gupao.vip.mic.dubbo.order.IOrderServices**

?anyhost=true&application=order-provider&dubbo=2.5.3&interface=com.gupao.vip.mic.dubbo.order.IOrderServices&methods=doOrder&owner=mic&pid=12500&side=provider&timestamp=1502889986089, dubbo version: 2.5.3, current host: 127.0.0.1

dubbo://177.1.1.82/20880/com.gupao.vip.mic.dubbo.order.IOrderServices%3Fanyhost%3Dtrue%26application%3Dorder-provider%26dubbo%3D2.5.3%26interface%3Dcom.gupao.vip.mic.dubbo.order.IOrderServices%26methods%3DdoOrder%26owner%3Dmic%26pid%3D10804%26side%3Dprovider%26timestamp%3D1502890818766

# Main方法怎么启动的

# 日志怎么集成

# admin控制台的安装

1. 下载dubbo的源码
2. 找到dubbo-admin
3. 修改webapp/WEB-INF/dubbo.properties

**dubbo.registry.address=zookeeper的集群地址**

控制中心是用来做服务治理的，比如控制服务的权重、服务的路由、。。。

# simple监控中心

Monitor也是一个dubbo服务，所以也会有端口和url

修改/conf目录下dubbo.properties /order-provider.xml

dubbo.registry.address=zookeeper://192.168.11.129:2181?backup=192.168.11.137:2181,192.168.11.138:2181,192.168.11.139:2181

监控服务的调用次数、调用关系、响应事件

# telnet命令

telnet ip port

ls、cd、pwd、clear、invoker

记得录屏

# 启动服务检查

如果提供方没有启动的时候，默认会去检测所依赖的服务是否正常提供服务

如果check为false，表示启动的时候不去检查。当服务出现循环依赖的时候，check设置成false

dubbo:reference 属性： check 默认值是true 、false

dubbo:consumer check=”false” 没有服务提供者的时候，报错

dubbo:registry check=false 注册订阅失败报错

dubbo:provider

# 多协议支持

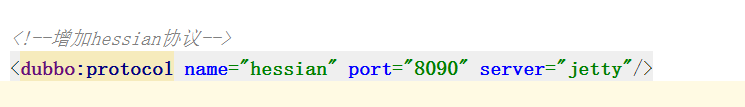
dubbo支持的协议： dubbo、RMI、**hessian**、webservice、http、Thrift

## hessian协议演示

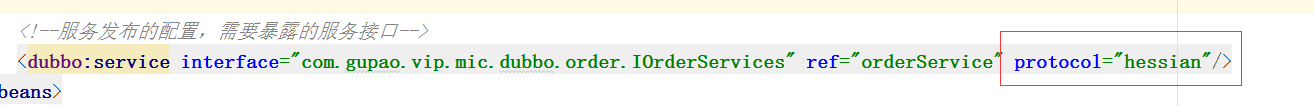
### 引入jar包

<**dependency**>  
 <**groupId**>com.caucho</**groupId**>  
 <**artifactId**>hessian</**artifactId**>  
 <**version**>4.0.38</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>javax.servlet</**groupId**>  
 <**artifactId**>servlet-api</**artifactId**>  
 <**version**>2.5</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.mortbay.jetty</**groupId**>  
 <**artifactId**>jetty</**artifactId**>  
 <**version**>6.1.26</**version**>  
</**dependency**>

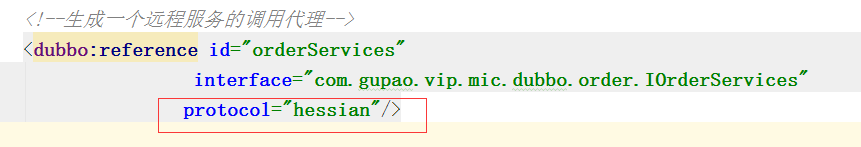
### 修改provider.xml



### 指定service服务的协议版本号



## 消费端改造

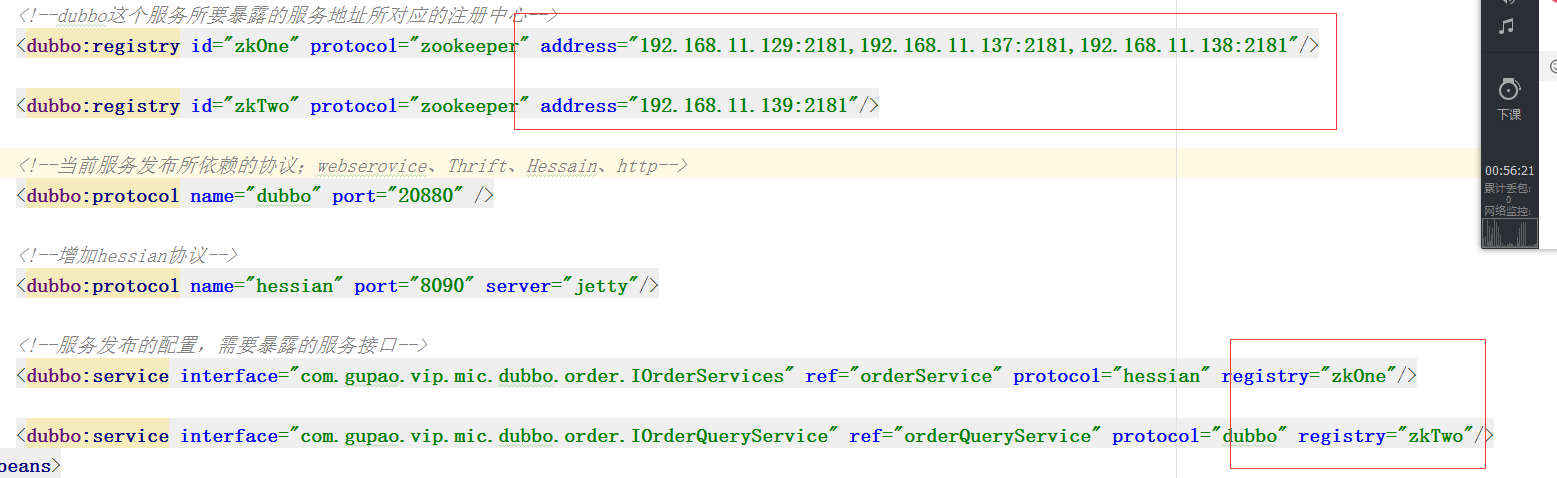


hessian%3A%2F%2F177.1.1.82%3A8090%2Fcom.gupao.vip.mic.dubbo.order.IOrderServices%3Fanyhost%3Dtrue%26application%3Dorder-provider%26dubbo%3D2.5.3%26interface%3Dcom.gupao.vip.mic.dubbo.order.IOrderServices%26methods%3DdoOrder%26owner%3Dmic%26pid%3D3116%26server%3Djetty%26side%3Dprovider%26timestamp%3D1503145940360,

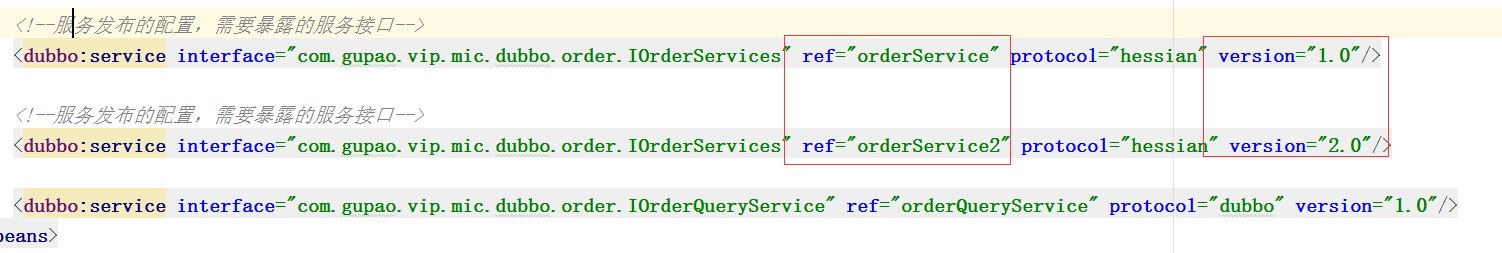
dubbo%3A%2F%2F177.1.1.82%3A20880%2Fcom.gupao.vip.mic.dubbo.order.IOrderServices%3Fanyhost%3Dtrue%26application%3Dorder-provider%26dubbo%3D2.5.3%26interface%3Dcom.gupao.vip.mic.dubbo.order.IOrderServices%26methods%3DdoOrder%26owner%3Dmic%26pid%3D3116%26side%3Dprovider%26timestamp%3D1503145950346]

hessian://177.1.1.82:8090/com.gupao.vip.mic.dubbo.order.IOrderServices

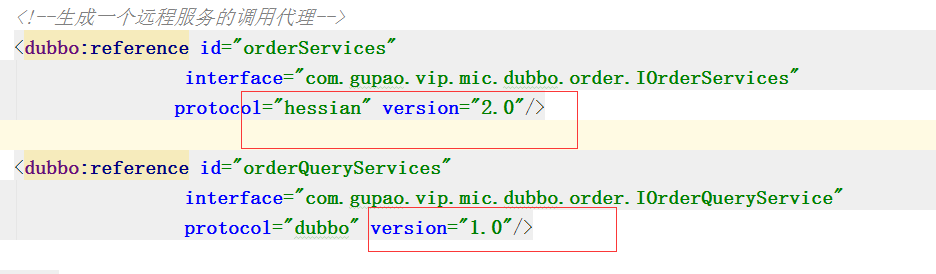
# 多注册中心支持



# 多版本支持



客户端调用的时候



hessian%3A%2F%2F177.1.1.82%3A8090%2Fcom.gupao.vip.mic.dubbo.order.IOrderServices%3Fanyhost%3Dtrue%26application%3Dorder-provider%26dubbo%3D2.5.3%26interface%3Dcom.gupao.vip.mic.dubbo.order.IOrderServices%26methods%3DdoOrder%26owner%3Dmic%26pid%3D7704%26revision%3D1.0%26server%3Djetty%26side%3Dprovider%26timestamp%3D1503147499144%26version%3D1.0,

hessian%3A%2F%2F177.1.1.82%3A8090%2Fcom.gupao.vip.mic.dubbo.order.IOrderServices2%3Fanyhost%3Dtrue%26application%3Dorder-provider%26dubbo%3D2.5.3%26interface%3Dcom.gupao.vip.mic.dubbo.order.IOrderServices%26methods%3DdoOrder%26owner%3Dmic%26pid%3D7704%26revision%3D2.0%26server%3Djetty%26side%3Dprovider%26timestamp%3D1503147510114%26version%3D2.0

# 异步调用

**async="true"表示接口异步返回**

hessian协议，使用async异步回调会报错

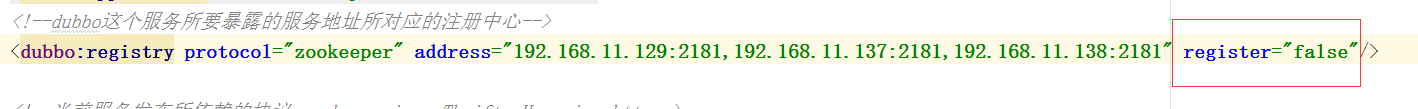
# 主机绑定

provider://177.1.1.82:20880

1. 通过<dubbo:protocol host配置的地址去找
2. 
3. 通过socket发起连接连接到注册中心的地址。再获取连接过去以后本地的ip地址
4. host = NetUtils.*getLocalHost*();

|  |
| --- |
| **if** (NetUtils.*isInvalidLocalHost*(host)) {  anyhost = **true**;  **try** {  host = InetAddress.*getLocalHost*().getHostAddress();  } **catch** (UnknownHostException e) {  ***logger***.warn(e.getMessage(), e);  }  **if** (NetUtils.*isInvalidLocalHost*(host)) {  **if** (registryURLs != **null** && registryURLs.size() > 0) {  **for** (URL registryURL : registryURLs) {  **try** {  Socket socket = **new** Socket();  **try** {  SocketAddress addr = **new** InetSocketAddress(registryURL.getHost(), registryURL.getPort());  socket.connect(addr, 1000);  host = socket.getLocalAddress().getHostAddress();  **break**;  } **finally** {  **try** {  socket.close();  } **catch** (Throwable e) {}  }  } **catch** (Exception e) {  ***logger***.warn(e.getMessage(), e);  }  }  }  **if** (NetUtils.*isInvalidLocalHost*(host)) {  host = NetUtils.*getLocalHost*();  } } |

# dubbo服务只订阅



# dubbo服务只注册

只提供服务

<**dubbo:registry subscribe="false"**/>

# 负载均衡

在集群负载均衡时，Dubbo提供了多种均衡策略，缺省为random随机调用。可以自行扩展负载均衡策略

## 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。

# 连接超时timeout

必须要设置服务的处理的超时时间

# 集群容错

**Failover cluster 失败的时候自动切换并重试其他服务器。 通过retries=2。 来设置重试次数**

failfast cluster 快速失败，只发起一次调用 ; 写操作。比如新增记录的时候， 非幂等请求

failsafe cluster 失败安全。 出现异常时，直接忽略异常

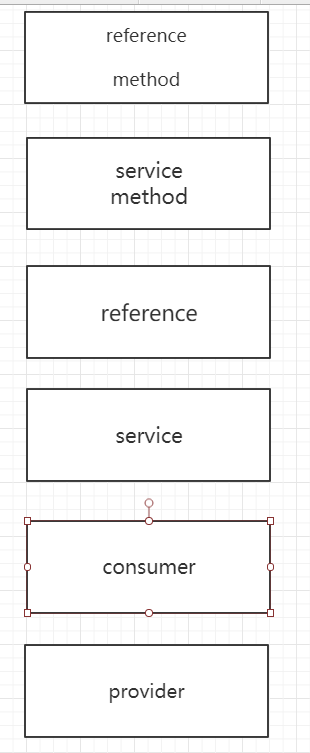
failback cluster 失败自动恢复。 后台记录失败请求，定时重发

forking cluster 并行调用多个服务器，只要一个成功就返回。 只能应用在读请求

broadcast cluster 广播调用所有提供者，逐个调用。其中一台报错就会返回异常

# 配置的优先级

消费端有限最高 – 服务端



# 服务改造

|  |  |
| --- | --- |
|  |  |

# 服务的最佳实践

## 分包

1. 服务接口、请求服务模型、异常信息都放在api里面，符合重用发布等价原则，共同重用原则
2. api里面放入spring 的引用配置。 也可以放在模块的包目录下。 com.gupao.vip.mic.order/\*\*\*-reference.xml

## 粒度

1. 尽可能把接口设置成粗粒度，每个服务方法代表一个独立的功能，而不是某个功能的步骤。否则就会涉及到分布式事务
2. 服务接口建议以业务场景为单位划分。并对相近业务做抽象，防止接口暴增
3. 不建议使用过于抽象的通用接口 T T<泛型>， 接口没有明确的语义，带来后期的维护

## 版本

1. 每个接口都应该定义版本，为后续的兼容性提供前瞻性的考虑 version （maven -snapshot）
2. 建议使用两位版本号，因为第三位版本号表示的兼容性升级，只有不兼容时才需要变更服务版本
3. 当接口做到不兼容升级的时候，先升级一半或者一台提供者为新版本，再将消费全部升级新版本，然后再将剩下的一半提供者升级新版本

预发布环境

# 推荐用法

## 在provider端尽可能配置consumer端的属性

比如timeout、retires、线程池大小、LoadBalance

## 配置管理员信息

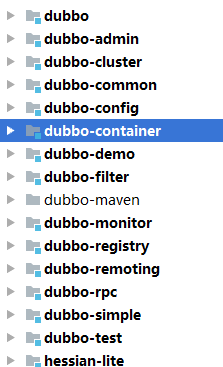
application上面配置的owner 、 owner建议配置2个人以上。因为owner都能够在监控中心看到

# 配置dubbo缓存文件

注册中心的列表

服务提供者列表

# 源码分析



基于spring 配置文件的扩展的话

NamespaceHandler: 注册BeanDefinitionParser， 利用它来解析

BeanDefinitionParser： 解析配置文件的元素

spring会默认加载jar包下/META-INF/spring.handlers 找到对应的NamespaceHandler

initializingBean, 当spring容器初始化完以后，会调用afterPropertiesSet方法

DisposableBean,

ApplicationContextAware,

ApplicationListener,

BeanNameAware

# 课后作业

如下，pay-center做了四台机器的集群，每个机器上都运行了一个定时任务。

需求是：

按照如下架构搭建一个分布式应用架构，使用dubbo做为rpc

保证定时任务的触发只在其中一台机器上进行，其他机器不执行，通过zookeeper去实现

