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

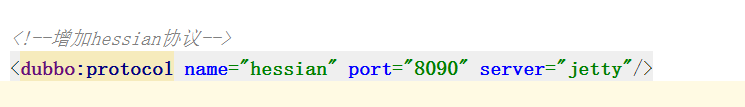
dubbo支持的协议： dubbo、RMI、**hessian**、webservice、http、Thrift

## hessian协议演示

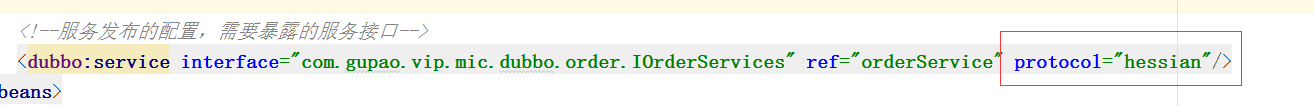
### 引入jar包

<**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**>

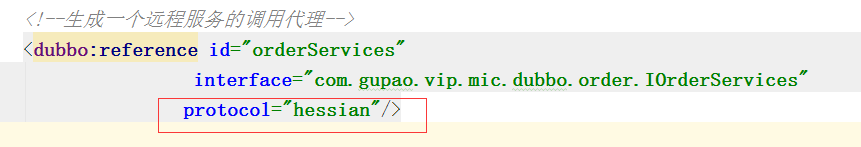
### 修改provider.xml



### 指定service服务的协议版本号



## 消费端改造

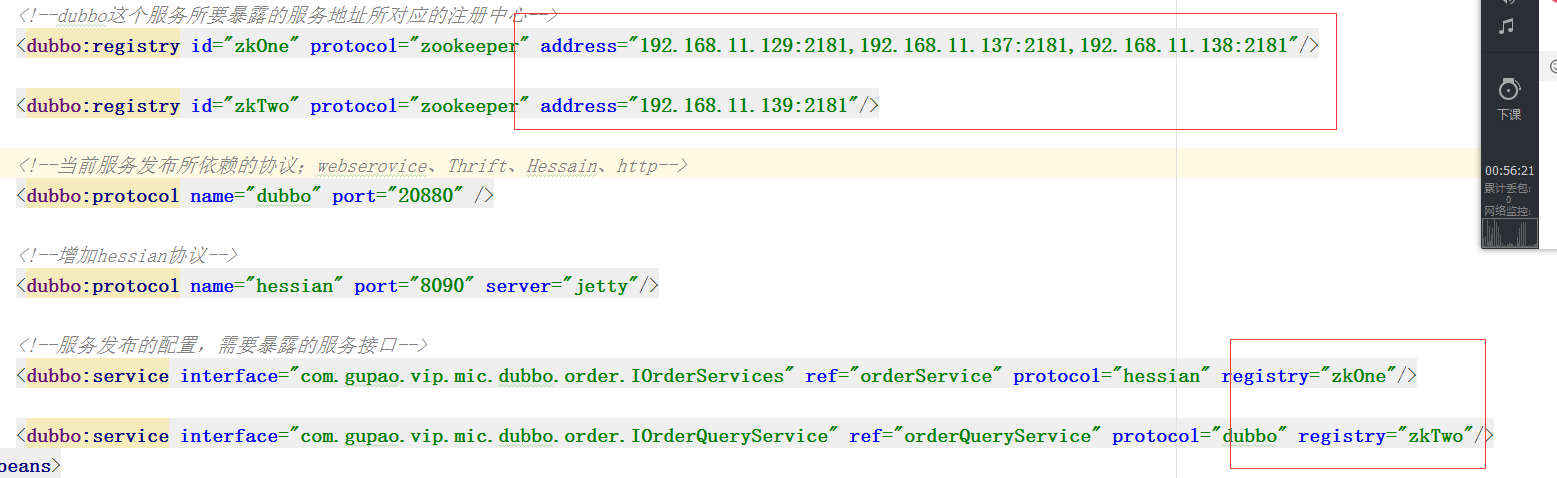


hessian%3A%2F%2F177.1.1.82%3A8090%2Fcom.gupao.vip.mic.dubbo.order.IOrderServices%3Fanyhost%3Dtrue%26application%3Dorder-provider%26dubbo%3D2.5.3%26interface%3Dcom.gupao.vip.mic.dubbo.order.IOrderServices%26methods%3DdoOrder%26owner%3Dmic%26pid%3D3116%26server%3Djetty%26side%3Dprovider%26timestamp%3D1503145940360,

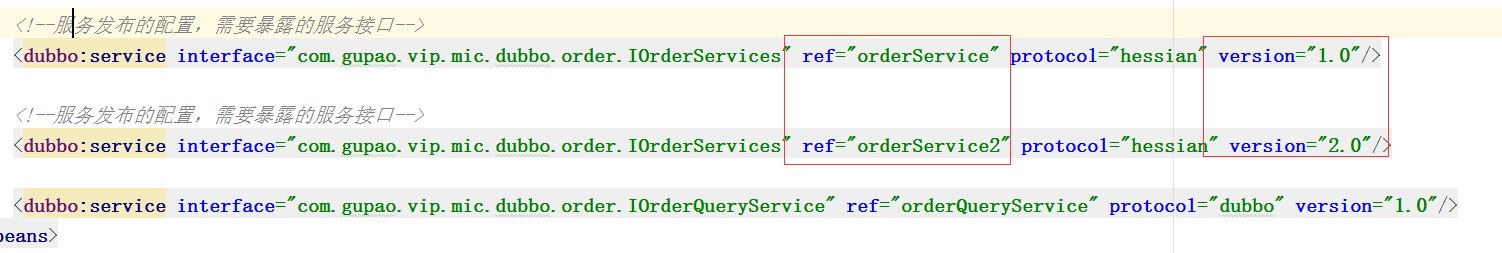
dubbo%3A%2F%2F177.1.1.82%3A20880%2Fcom.gupao.vip.mic.dubbo.order.IOrderServices%3Fanyhost%3Dtrue%26application%3Dorder-provider%26dubbo%3D2.5.3%26interface%3Dcom.gupao.vip.mic.dubbo.order.IOrderServices%26methods%3DdoOrder%26owner%3Dmic%26pid%3D3116%26side%3Dprovider%26timestamp%3D1503145950346]

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

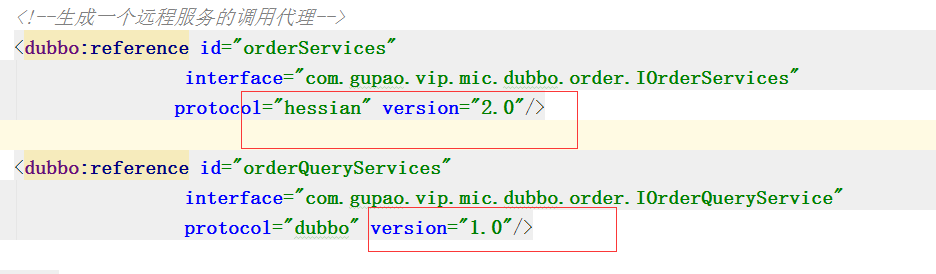
# 多注册中心支持



# 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

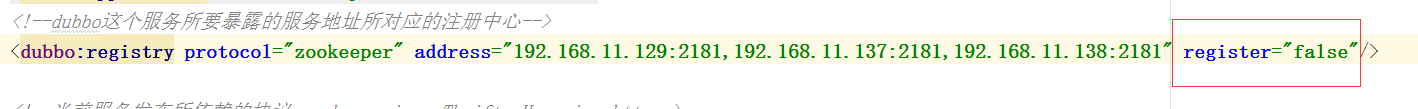
# 主机绑定

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