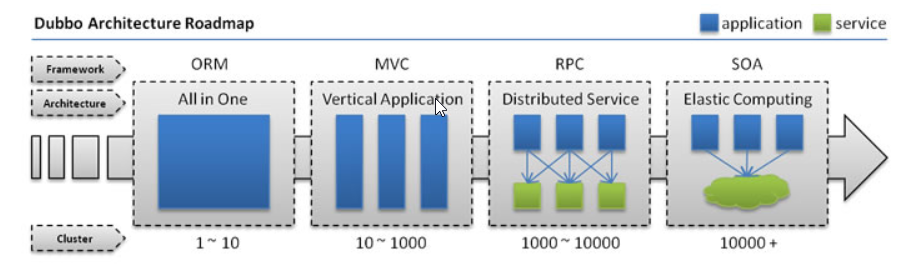
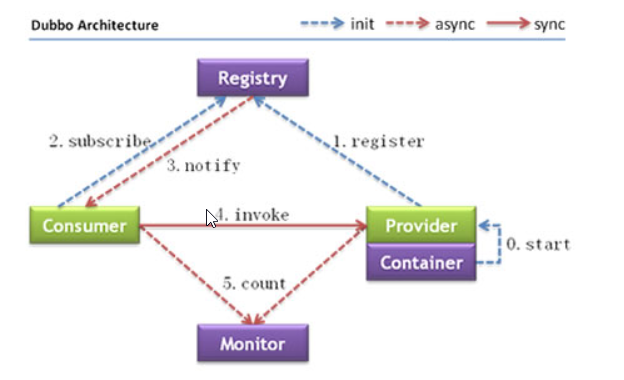
快速搭建dubbo+zookeeper架构

# 1．Dubbo说明

首先，看下一般网站架构随着业务的发展，逻辑越来越复杂，数据量越来越大，交互越来越多之后的常规方案演进历程



是dubbo的架构图



# 2.注册中心的选择

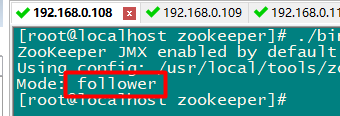


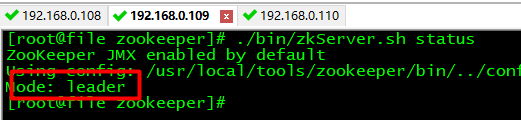
本次搭建用三台虚拟机来模拟

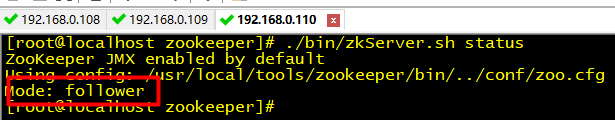
192.168.0.108

192.168.0.109

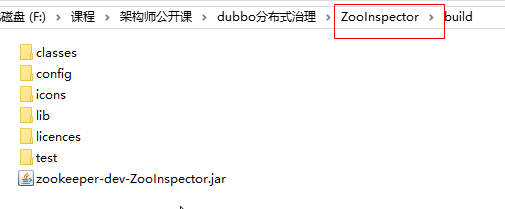
192.168.0.110



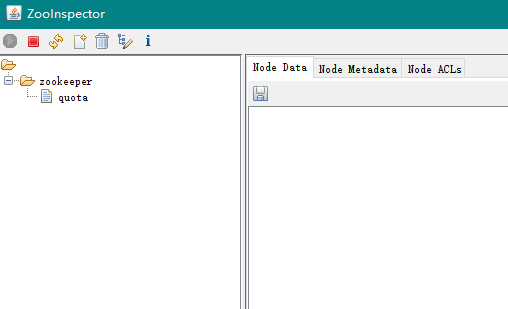




通过工具查看zk视图数据

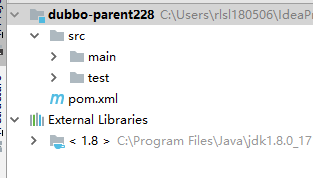


Dos命令执行：java -jar zookeeper-dev-ZooInspector.jar



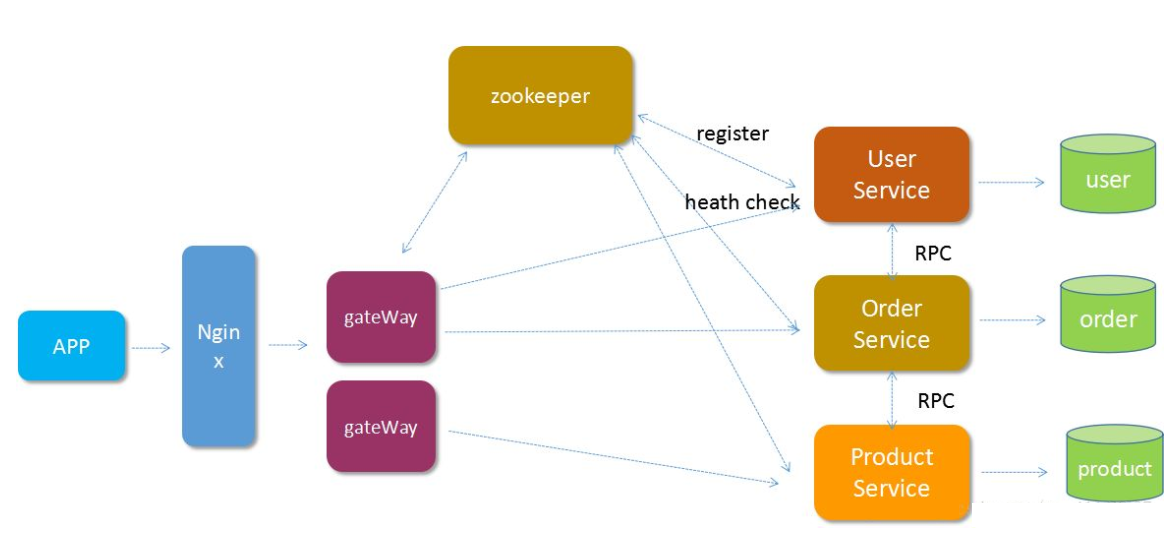
# 3.项目搭建

1.创建maven项目



2. 创建model模块

本例中我们需要生产者和消费者



Zk

订阅

注册

OrderService（服务提供者）

UseService（服务消费者）

调用

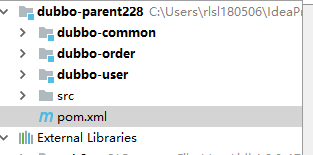
创建Model：

Dubbo-order 服务提供者

Dubbo-user 服务消费者

Dubbo-common 公共模块

公共模块为了减少代码的重复性，为多个model的功用模块。



3.导入依赖

父工程pom.xml的依赖

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.6.2</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.101tec</**groupId**>  <**artifactId**>zkclient</**artifactId**>  <**version**>0.9</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.zookeeper</**groupId**>  <**artifactId**>zookeeper</**artifactId**>  <**version**>3.4.9</**version**>  <**exclusions**>  <**exclusion**>  <**groupId**>log4j</**groupId**>  <**artifactId**>log4j</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  <**dependency**>  <**groupId**>log4j</**groupId**>  <**artifactId**>log4j</**artifactId**>  <**version**>1.2.16</**version**>  </**dependency**>  <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  <**version**>4.12</**version**>  <**scope**>test</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-test</**artifactId**>  <**version**>4.3.16.RELEASE</**version**>  <**scope**>test</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.curator</**groupId**>  <**artifactId**>curator-framework</**artifactId**>  <**version**>4.0.0</**version**>  </**dependency**> </**dependencies**> |

Dubbo-order中的pom.xml

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>cn.tx.dubbo</**groupId**>  <**artifactId**>dubbo-common</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> |

Dubbo-user中的pom.xml

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>cn.tx.dubbo</**groupId**>  <**artifactId**>dubbo-common</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> |

4.在model中创建公共代码

创建Order的Model类，注意要序列化

|  |
| --- |
| **public class** Order **implements** Serializable {   **private** Integer **orderId**;   **private double price**;   **private** String **addr**;   **public** Integer getOrderId() {  **return orderId**;  }   **public void** setOrderId(Integer orderId) {  **this**.**orderId** = orderId;  }   **public double** getPrice() {  **return price**;  }   **public void** setPrice(**double** price) {  **this**.**price** = price;  }   **public** String getAddr() {  **return addr**;  }   **public void** setAddr(String addr) {  **this**.**addr** = addr;  }   @Override  **public** String toString() {  **return "Order{"** +  **"orderId="** + **orderId** +  **", price="** + **price** +  **", addr='"** + **addr** + **'\''** +  **'}'**;  } } |

创建OrderService服务接口

|  |
| --- |
| **public interface** OrderService {   **public** List<Order> getOrders(); } |

5.在dubbo-order的model中创建服务接口的实现类

|  |
| --- |
| **public class** OrderServiceImpl **implements** OrderService {      **public** List<Order> getOrders() {    Order order = **new** Order();  order.setOrderId(1);  order.setPrice(100);  order.setAddr(**"北京"**);   Order order1 = **new** Order();  order1.setOrderId(2);  order1.setPrice(200);  order1.setAddr(**"上海"**);      List<Order> orderList = **new** ArrayList<Order>();  orderList.add(order);  orderList.add(order1);    **return** orderList;  } } |

6.创建服务提供端的Spring配置文件

在dubbo-resources下创建spring包，然后创建order-provider.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://code.alibabatech.com/schema/dubbo  http://code.alibabatech.com/schema/dubbo/dubbo.xsd"**>    <**bean id="orderService" class="cn.tx.service.impl.OrderServiceImpl"**></**bean**>   <**dubbo:application name="orderServiceProvider"**></**dubbo:application**>   <**dubbo:registry address="zookeeper://192.168.0.109:2181?backup=192.168.0.109:2181,192.168.0.110:2181"**></**dubbo:registry**>   <**dubbo:protocol port="20880"**></**dubbo:protocol**>   <**dubbo:service interface="cn.tx.service.OrderService" ref="orderService"**/> </**beans**> |

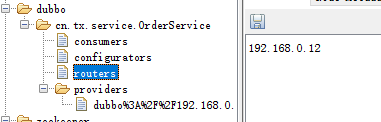
7.创建测试类

创建单元测试

在test资源包下创建cn.tx.test.OrderServiceTest

|  |
| --- |
| @RunWith(SpringJUnit4ClassRunner.**class**) @ContextConfiguration(locations = **"classpath:spring/order-provider.xml"**) **public class** OrderServiceTest {   @Test  **public void** test() **throws** IOException {  System.***out***.println(**"服务器启动了"**);  System.***in***.read();  } } |

服务提供端启动后可以观察zookeeper上的注册情况



8.服务消费端spring配置文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd">     <dubbo:application name="orderServiceProvider"></dubbo:application>  <dubbo:registry address="zookeeper://192.168.0.109:2181?backup=192.168.0.108:2181,192.168.0.110:2181"></dubbo:registry>   <dubbo:reference id="orderService" interface="cn.tx.service.OrderService"></dubbo:reference> </beans> |

9.在dubbo-user端创建单元测试类

|  |
| --- |
| @RunWith(SpringJUnit4ClassRunner.**class**) @ContextConfiguration(locations = **"classpath:spring/user-consumer.xml"**) **public class** UserConsumerTest {   @Autowired  **private** OrderService **orderService**;    @Test  **public void** test(){    List<Order> orders = **orderService**.getOrders();  System.***out***.println(orders);   }   } |

测试

