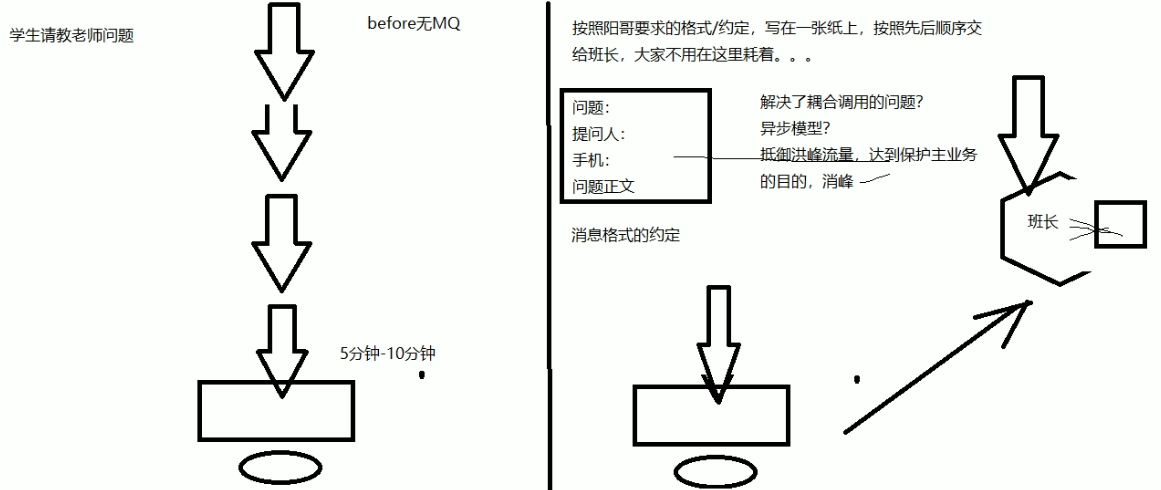
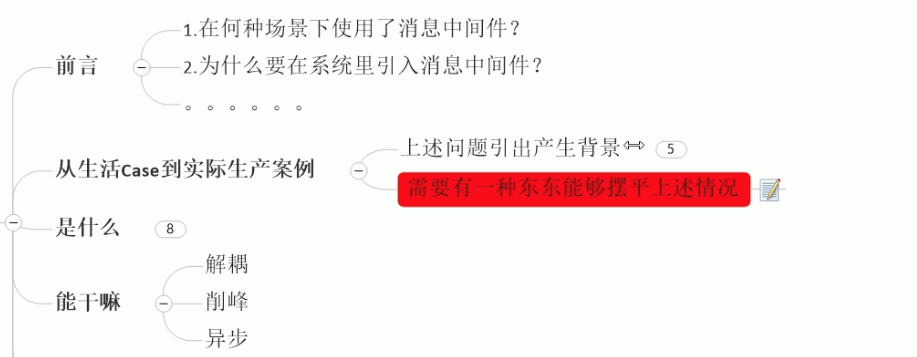
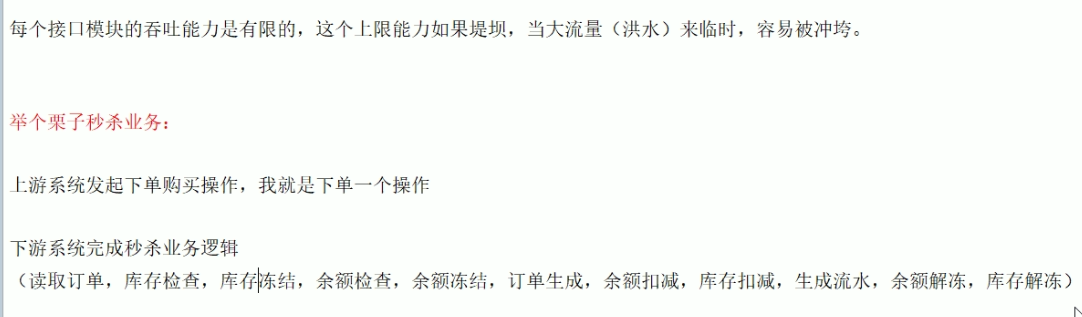
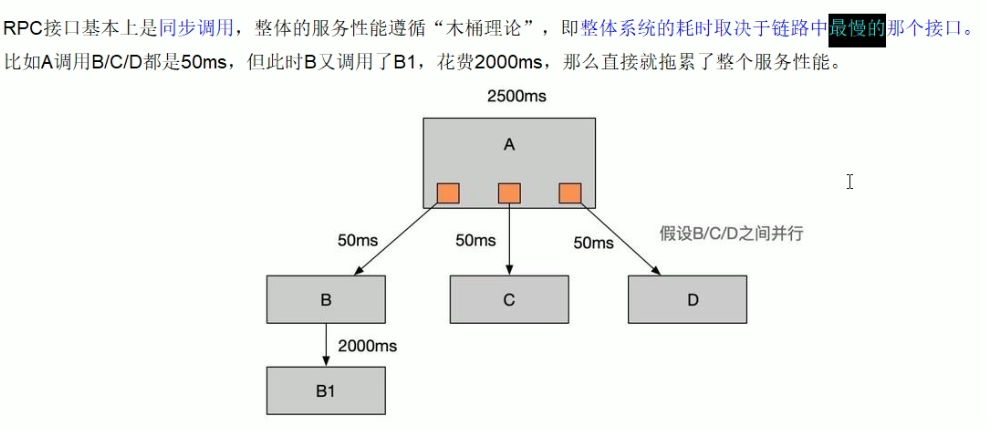


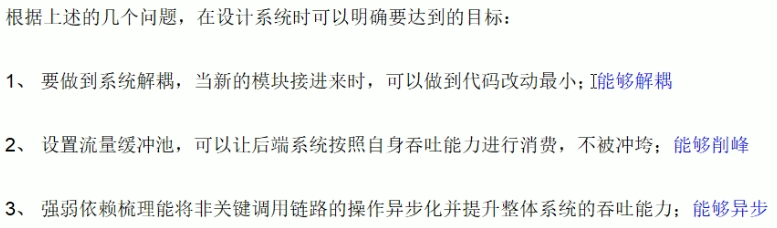
使用MQ前后对比



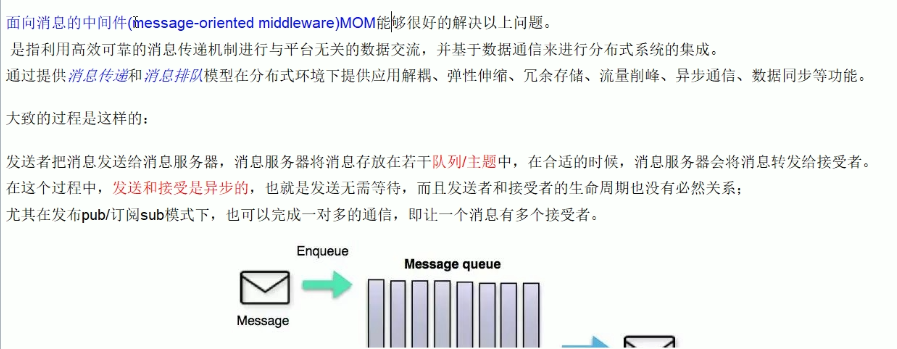


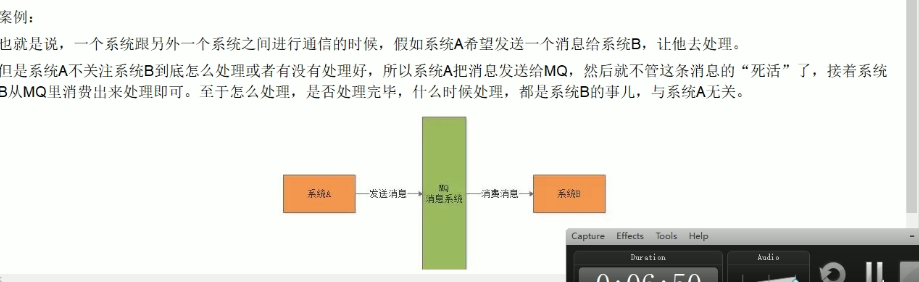






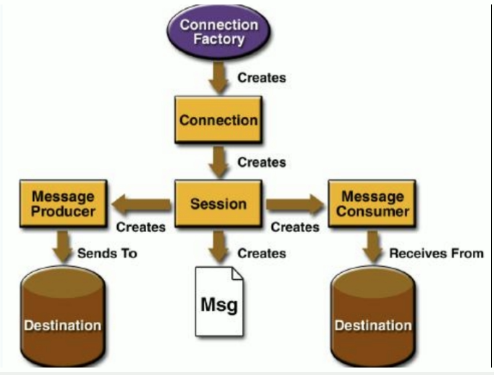
只要是消息发送，尽量使用异步。只要是系统解偶都可以使用中间件。















消息生产者原码：

**public class** JmsProduce ***{* public static final** String ***ACTIVEMQ\_URL***=**"tcp://localhost:61616"**;  
 **public static final** String ***QUEUE\_NAME***=**"queue01"**;  
  
 **public static void** main***(***String***[]*** args***)* throws** JMSException ***{*** *//1、创建连接工厂,按照给定的url地址，采用默认用户名和密码* ActiveMQConnectionFactory activeMQConnectionFactory = **new** ActiveMQConnectionFactory***(ACTIVEMQ\_URL)***;  
 *//2、通过连接工厂，获得连接connection* Connection connection = activeMQConnectionFactory.createConnection***()***;  
 connection.start***()***;  
 *//3、创建会话session  
 //3.1 两个参数：第一个是 事务，第二个是签收* Session session = connection.createSession***(*false**, Session.***AUTO\_ACKNOWLEDGE)***;  
 *//4、创建目的地（具体是队列还是主题topic）* Queue queue = session.createQueue***(QUEUE\_NAME)***;  
 *//5、创建消息的生产者* MessageProducer messageProducer = session.createProducer***(***queue***)***;  
 *//6、通过使用MessageProducer生产3条消息发送到MQ的队列里面* **for*(*int** i = 1; i<=3;i++***){*** *//7、创建消息* TextMessage textMessage = session.createTextMessage***(*"msg -- "**+i***)***;*//理解为一个字符串  
 //8、通过MessageProducer发送给MQ* messageProducer.send***(***textMessage***)***;  
 ***}*** *//9、关闭资源* messageProducer.close***()***;  
 session.close***()***;  
 connection.close***()***;  
  
 System.***out***.println***(*"\*\*\*\*\*\*\*\*消息发布到mq完成"*)***;  
 ***}  
}***

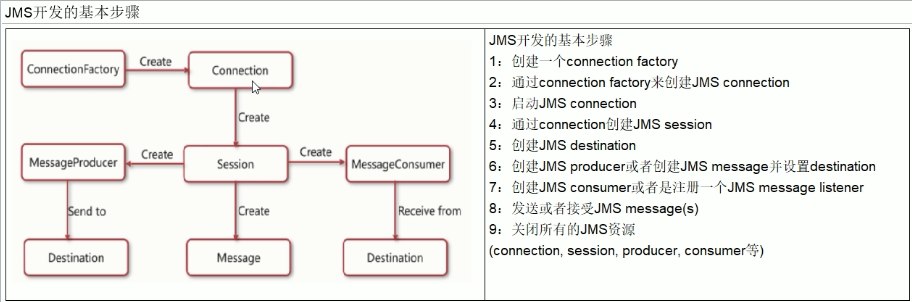
消息消费者源码：

**public class** JmsConsumer ***{* public static final** String ***ACTIVEMQ\_URL***=**"tcp://localhost:61616"**;  
 **public static final** String ***QUEUE\_NAME***=**"queue01"**;  
  
 **public static void** main***(***String***[]*** args***)* throws** JMSException ***{*** *//1、创建连接工厂,按照给定的url地址，采用默认用户名和密码* ActiveMQConnectionFactory activeMQConnectionFactory = **new** ActiveMQConnectionFactory***(ACTIVEMQ\_URL)***;  
 *//2、通过连接工厂，获得连接connection* Connection connection = activeMQConnectionFactory.createConnection***()***;  
 connection.start***()***;  
 *//3、创建会话session  
 //3.1 两个参数：第一个是 事务，第二个是签收* Session session = connection.createSession***(*false**, Session.***AUTO\_ACKNOWLEDGE)***;  
 *//4、创建目的地（具体是队列还是主题topic）* Queue queue = session.createQueue***(QUEUE\_NAME)***;  
 *//5、创建消费者* MessageConsumer messageConsumer = session.createConsumer***(***queue***)***;  
 **while *(*true*){*** TextMessage textMessage = ***(***TextMessage***)*** messageConsumer.receive***()***;  
 **if *(***textMessage != **null*){*** System.***out***.println***(*"消费者接收消息："** +textMessage.getText***())***;  
 ***}*else *{* break**;  
 ***}  
 }*** messageConsumer.close***()***;  
 session.close***()***;  
 connection.close***()***;  
  
 ***}  
}***

*//第二种方法接收消息*messageConsumer.setMessageListener***(*new** MessageListener***() {*** @Override  
 **public void** onMessage***(***Message message***) {* if *(***message != **null** && message **instanceof** TextMessage***){*** TextMessage textMessage = ***(***TextMessage***)*** message;  
 **try *{*** System.***out***.println***(*"消费者接收消息："** +textMessage.getText***())***;  
 ***}* catch *(***JMSException e***) {*** e.printStackTrace***()***;  
 ***}  
 }  
 }  
})***;  
System.***in***.read***()***;*//由于速度太快，还没有接收到消息，下面3条关闭资源的代码就执行了，所以会导致没有消费消息。因此需要加入此行代码。*messageConsumer.close***()***;  
session.close***()***;  
connection.close***()***;

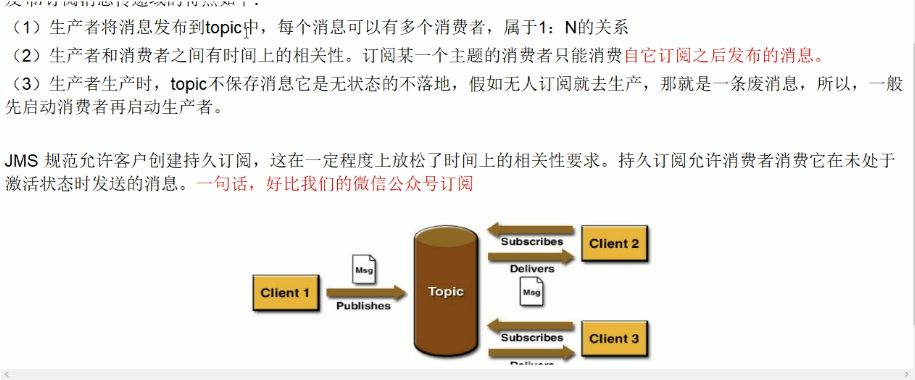
ActiveMQ所需要依赖：

*<!-- https://mvnrepository.com/artifact/org.apache.xbean/xbean-spring -->****<*dependency*>  
 <*groupId*>***org.apache.xbean***</*groupId*>  
 <*artifactId*>***xbean-spring***</*artifactId*>  
 <*version*>***3.1***</*version*>  
</*dependency*>****<!-- https://mvnrepository.com/artifact/org.apache.activemq/activemq-all -->****<*dependency*>  
 <*groupId*>***org.apache.activemq***</*groupId*>  
 <*artifactId*>***activemq-all***</*artifactId*>  
 <*version*>***5.15.9***</*version*>  
</*dependency*>***





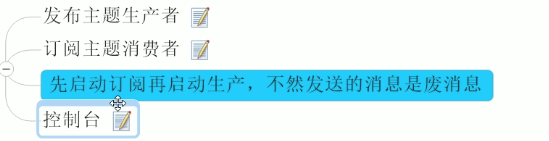
消息订阅 topic 了解





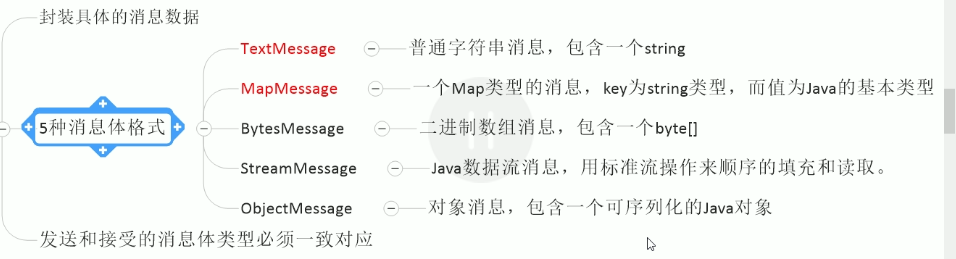


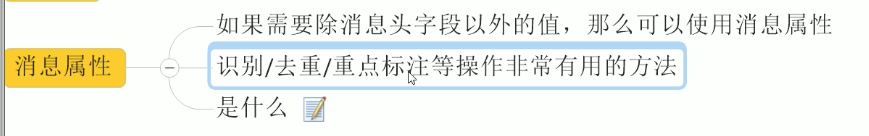
代码只改这里

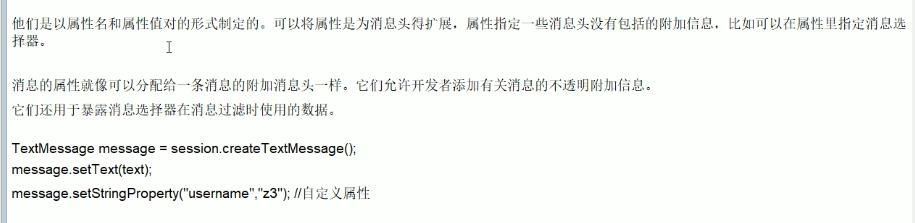


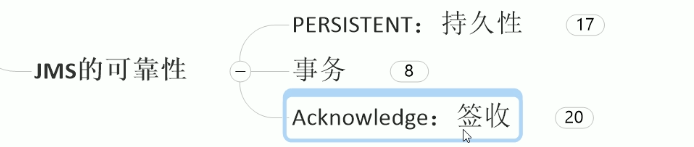
如果在未订阅之前发布的消息，则后来订阅的用户不能接收到历史消息

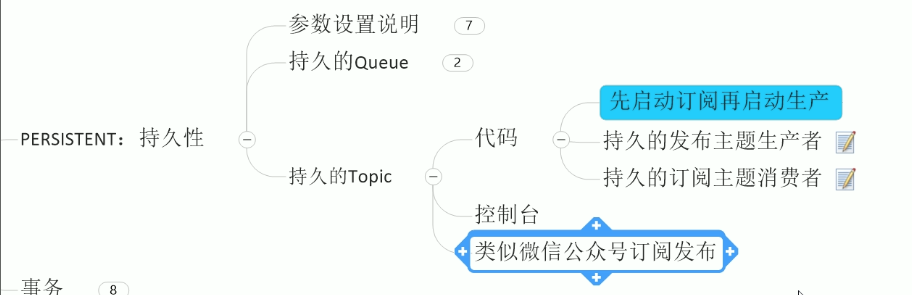


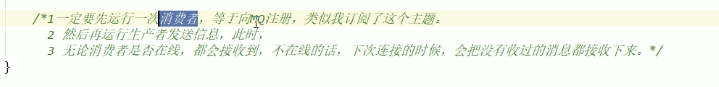




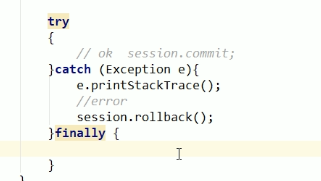




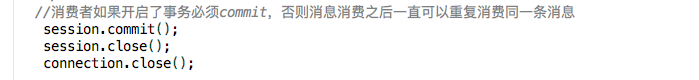


持久化topic

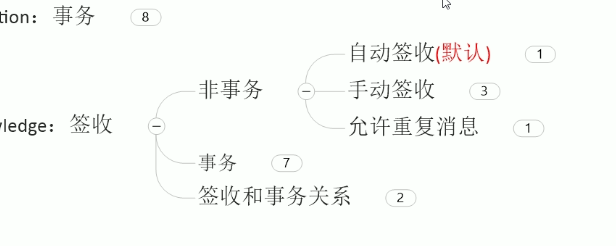
生产者事务：



消费者事务：



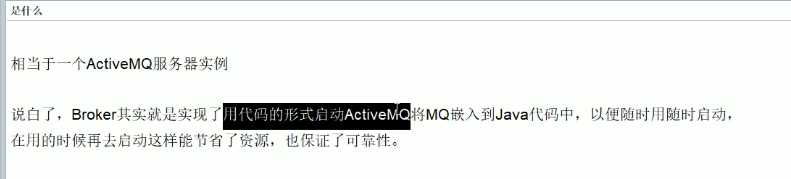
非事务情况下签收：



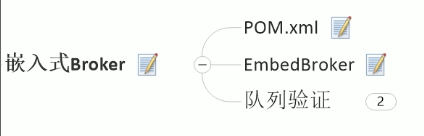
手动签收：textMessage.acknowledge();

事务情况下签收：尽管是手动签收，有事务commit都会变为自动签收。如果没有commit，使用了ack手动签收也是会出现重复消费情况。









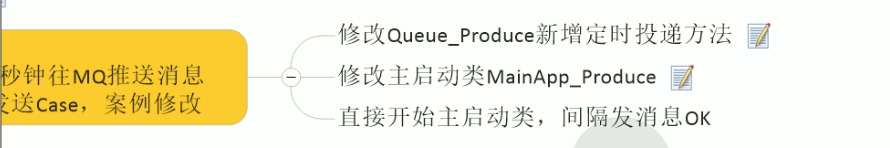


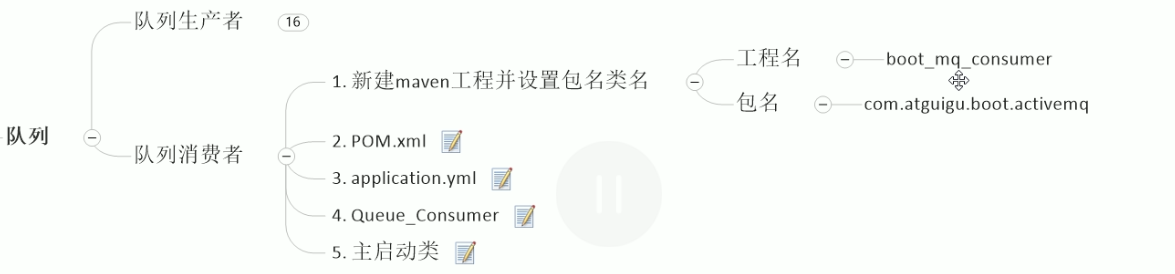


mini版MQ，直接在java中运行。在工作中一般还是选用linux安装

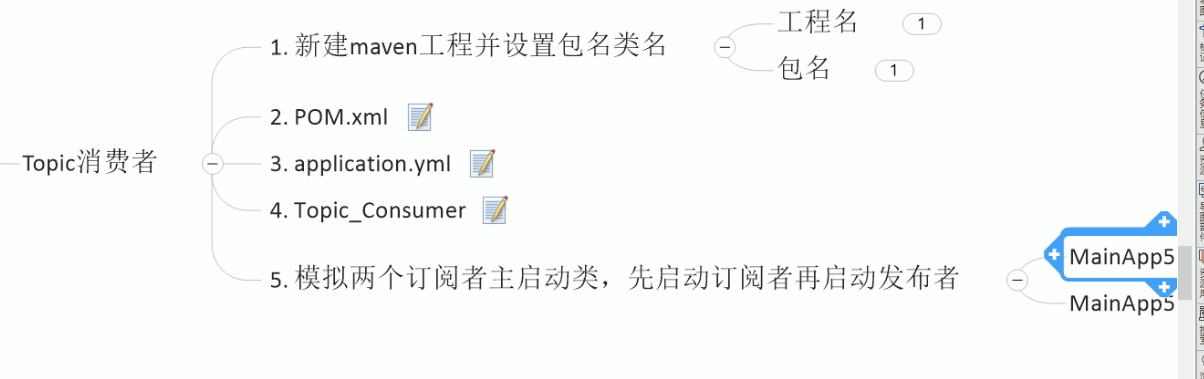
Springboot整合ActiveMQ

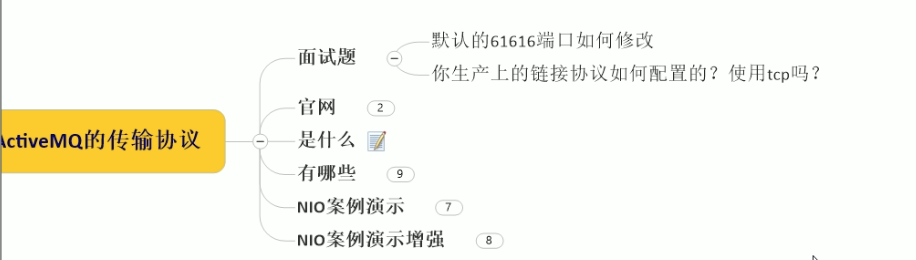




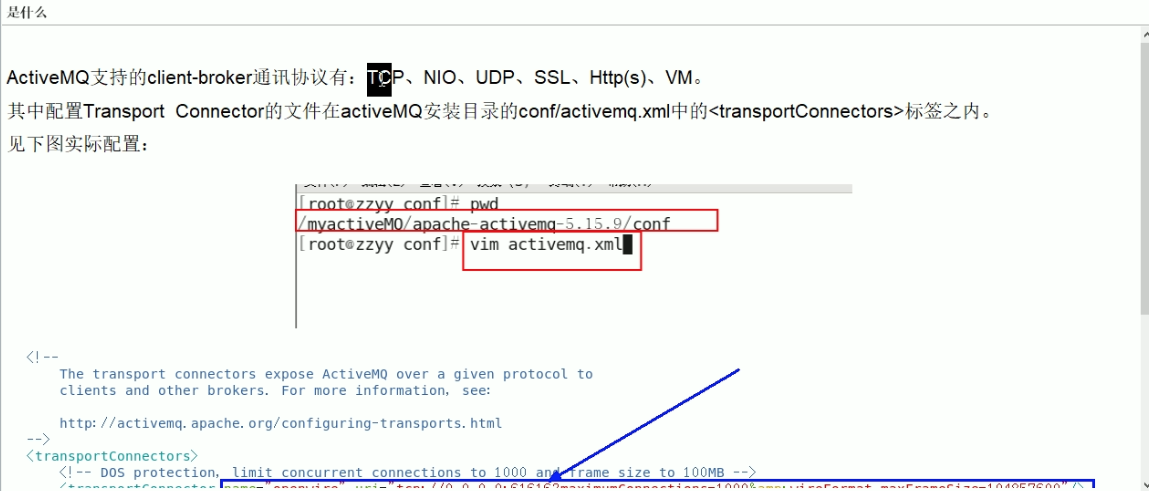








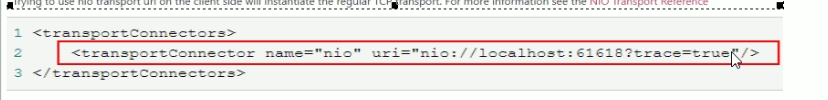






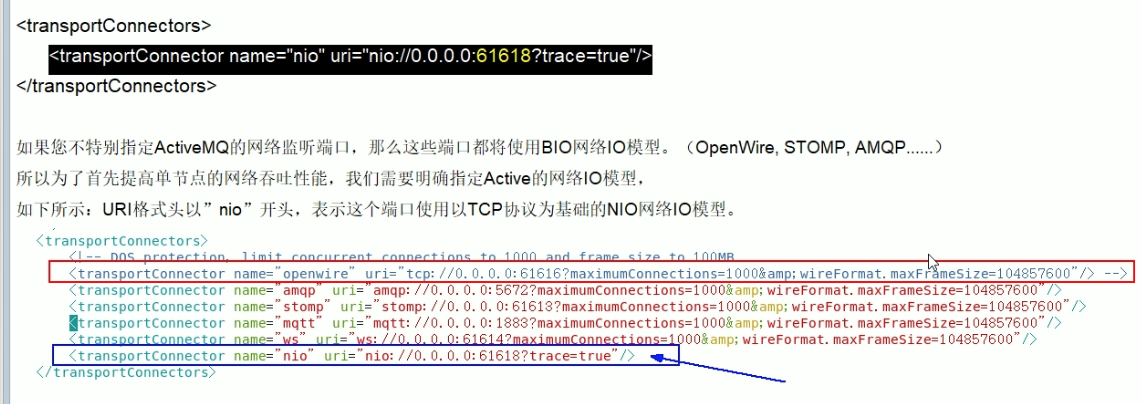








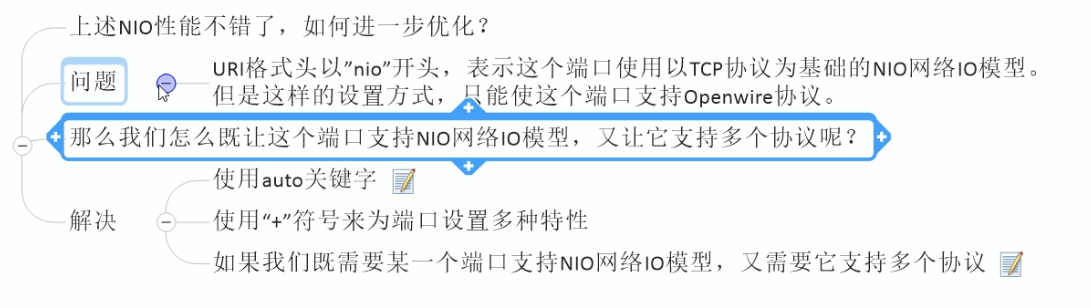
配置nio连接方式：

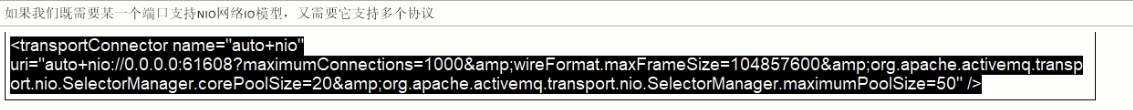


<transportConnector name="nio" uri="nio://0.0.0.0:61618?trace=true"/>



nio增强：



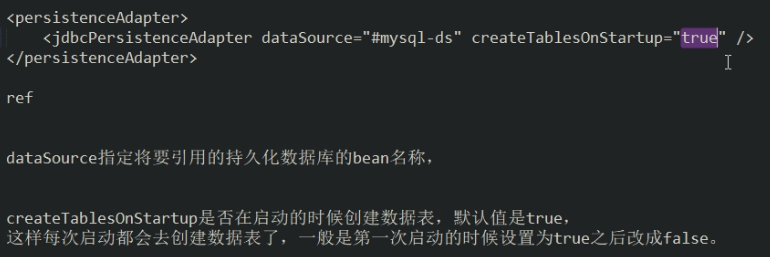


<transportConnector name="auto+nio" uri="nio://0.0.0.0:61608?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600&amp;org.apache.activemq.transport.nio.SelectorManager.corePoolSize=20&amp;org.apache.activemq.transport.nio.SelectorManager.maximumPoolSize=50"/>











<bean id="mysql-ds" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">

<property name="driverClassName" value="com.mysql.jdbc.Driver"/>

<property name="url" value="jdbc:mysql://localhost/activemq?relaxAutoCommit=true"/>

<property name="username" value="root"/>

<property name="password" value="123456"/>

<property name="poolPreparedStatements" value="true"/>

</bean>

上面的mysql拦截配置放在conf中activemq.xml图中方框位置。

3张表字段：





