# 1、删除不活跃的订阅者，

## 解释：某些主题消息因为客户端在订阅后却长时间离线，而一直进驻内存，影响系统内存量及稳定性

## 1、activemq.xml中配置

### offlineDurableSubscriberTimeout 离线多长时间就过去删除，缺省是—1,就是不删除

### offlineDurableSubscriberTaskSchedule，多长时间检查一次 ，缺省是300000

|  |
| --- |
| <broker xmlns="http://activemq.apache.org/schema/core"  brokerName="localhost"  dataDirectory="${activemq.data}"  offlineDurableSubscriberTimeout="200000"  offlineDurableSubscriberTaskSchedule="10000"  > |

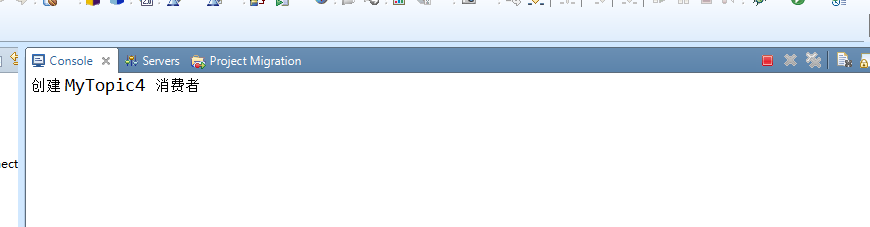
## 2、创建持久化的topic生产者

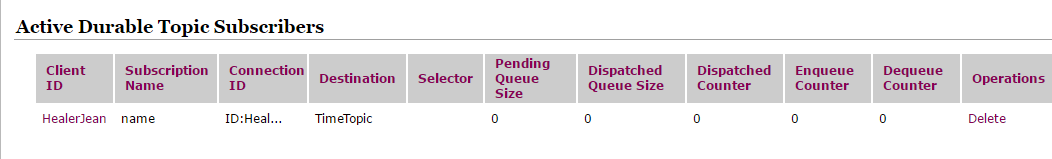
|  |
| --- |
| **public** **class** PersistenceSender {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");  Connection connection = connectionFactory.createConnection();  Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  Destination destination = session.createTopic("TimeTopic");    System.*out*.println("创建 MyTopic4 生产者");  MessageProducer producer = session.createProducer(destination);  producer.setDeliveryMode(DeliveryMode.*PERSISTENT*);  connection.start();    **for** (**int** i = 0; i < 3; i++) {  TextMessage message = session.createTextMessage("message333--" + i);    // 通过消息生产者发出消息  producer.send(message);  }  session.commit();  session.close();  connection.close();  }  } |

## 3、创建消费者读取

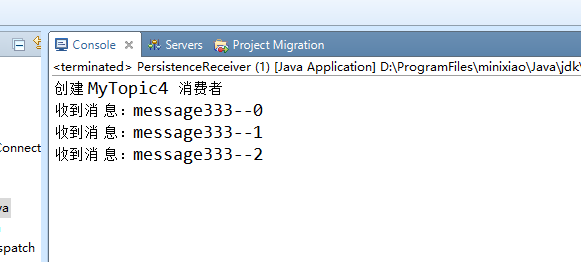
|  |
| --- |
| **public** **class** PersistenceReceiver {  **public** **static** **void** main(String[] args) **throws** Exception {    String linuxIp = "myLinuxQj";  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");  Connection connection = connectionFactory.createConnection();  //设置连接客户端 id  connection.setClientID("HealerJean");    **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);    Topic topic = session.createTopic("TimeTopic");  TopicSubscriber consumer = session.createDurableSubscriber(topic, "name");  System.*out*.println("创建 MyTopic4 消费者");  connection.start();    Message message = consumer.receive();  **while**(message!=**null**) {  TextMessage txtMsg = (TextMessage)message;  System.*out*.println("收到消 息：" + txtMsg.getText());  message = consumer.receive(1000L);  }  session.commit();  session.close();  connection.close();  }  } |

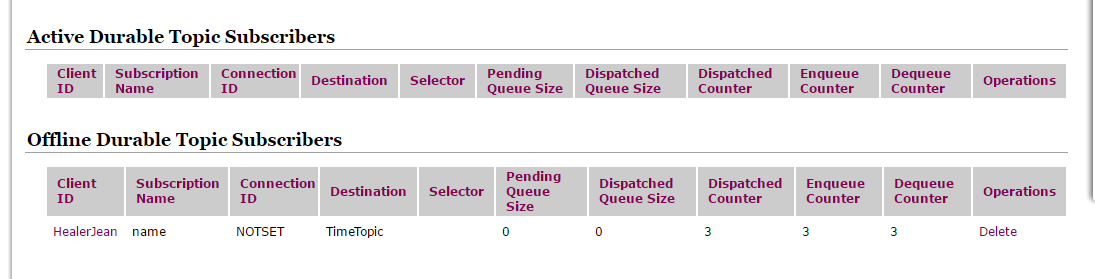
## 4、运行消费者，准备读取消息，并且观察浏览器，是否存在订阅者，发现有一个激活的订阅者。



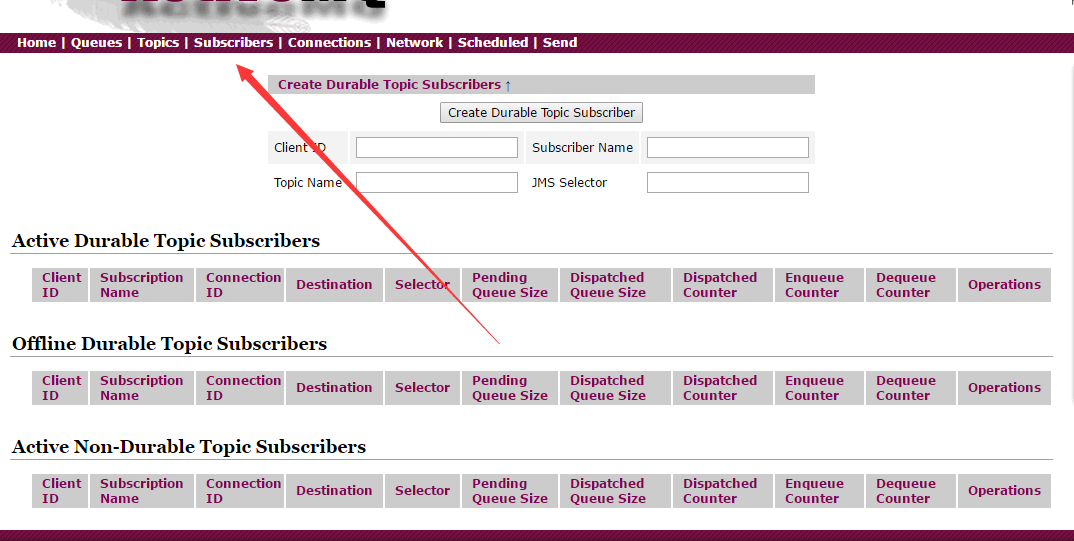


## 5、运行生产者，观察消费者，是否能够读取到消息，观察到读取到消息了，紧接着，这个读取到消息的消费者，就会进去不在线的订阅者、





## 6、再过20秒就会自动删除掉这个订阅者



# 2、Message Groups

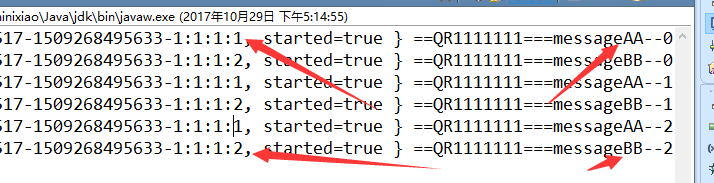
## 1、建立一个queue生产者，TestMessage发送group，这里是创建了两个组A和B

|  |
| --- |
| **public** **class** QueueGroupSender {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");  Connection connection = connectionFactory.createConnection();  connection.start();  Session session = connection.createSession(Boolean.*TRUE*,  Session.*CLIENT\_ACKNOWLEDGE*);  Destination destination = session.createQueue("GroupQueue");    ActiveMQMessageProducer producer = (ActiveMQMessageProducer)session.createProducer(destination);    **for** (**int** i = 0; i < 3; i++) {  TextMessage message = session.createTextMessage("messageAA--" + i);      message.setStringProperty("JMSXGroupID","GroupA");    producer.send(message);    TextMessage message2 = session.createTextMessage("messageBB--" + i);  message2.setStringProperty("JMSXGroupID","GroupB");  producer.send(message2);  }    session.commit();  session.close();  connection.close();  }  } |

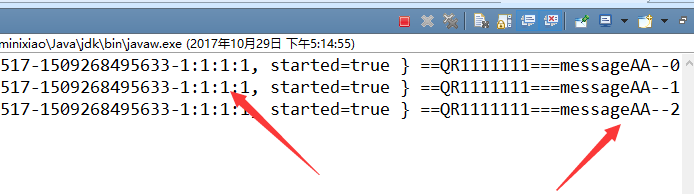
## 2、创建两个消费者开始接

|  |
| --- |
| **public** **class** QueueGroupReceiver {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");  Connection connection = connectionFactory.createConnection();    connection.start();  **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  Destination destination = session.createQueue("GroupQueue");  **for**(**int** i=0;i<2;i++){  **final** MessageConsumer consumer = session.createConsumer(destination);  consumer.setMessageListener(**new** MessageListener() {  **public** **void** onMessage(Message m) {  TextMessage msg = (TextMessage)m;  **try** {  System.*out*.println(consumer+ " ==QR1111111==="+msg.getText());  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }  }  });  }  }  } |

## 3、测试接收，观察控制台，会观察到，消费者1接收组A的消息，消费者2接收组B的消息，自动分组，分工明确

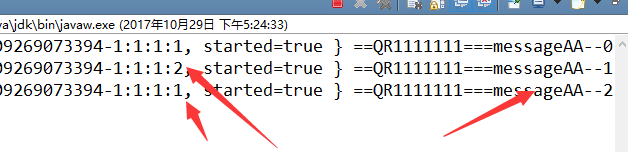


## 4、如果只创建一个组，两个消费者，则只有一个消费者接收，另一个消费者休息



## 5、如果没有组，则是两个消费者都接

|  |
| --- |
| ActiveMQMessageProducer producer = (ActiveMQMessageProducer)session.createProducer(destination);    **for** (**int** i = 0; i < 3; i++) {  TextMessage message = session.createTextMessage("messageAA--" + i);  producer.send(message);  } |

otd

# 3、message Selectors

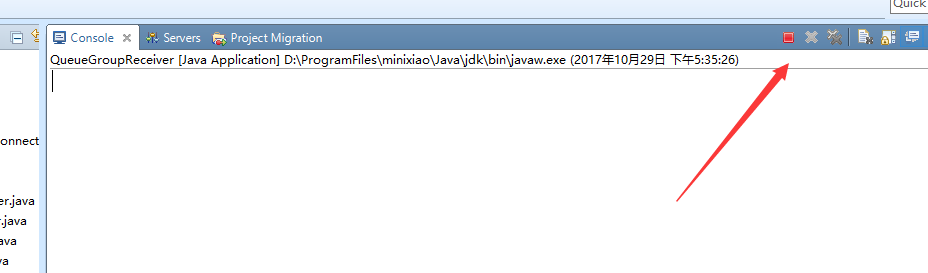
## 1、生产者，TextMessage ，创建int 定为数字23

|  |
| --- |
| **for** (**int** i = 0; i < 3; i++) {  TextMessage message = session.createTextMessage("messageAA--" + i);    message.setIntProperty("age", 23);    message.setStringProperty("JMSXGroupID","GroupA");    producer.send(message);  } |

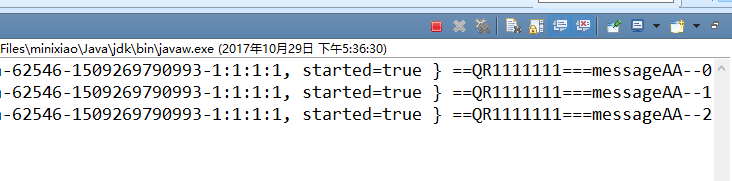
## 2、消费者，只接大于24的

|  |
| --- |
| **for**(**int** i=0;i<2;i++){  **final** MessageConsumer consumer = session.createConsumer(destination,"age >24");  consumer.setMessageListener(**new** MessageListener() {  **public** **void** onMessage(Message m) {  TextMessage msg = (TextMessage)m;  **try** {  System.*out*.println(consumer+ " ==QR1111111==="+msg.getText());  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }  }  });  } |

## 3、测试运行，没有接收到消息



## 4、改成大于等于23接收，成功



# 4、消费者事务不提交，默认只能接收6次，

## 解释

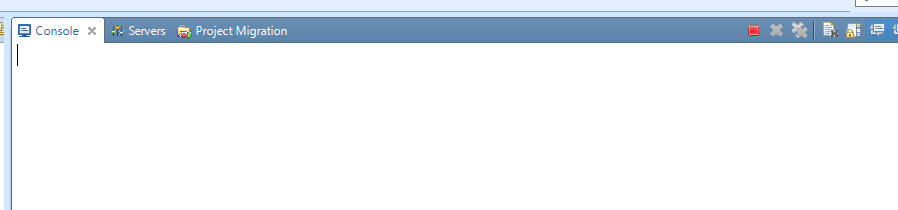
## 1、创建生产者

|  |
| --- |
| **public** **class** Sender {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");    Connection connection = connectionFactory.createConnection();  connection.start();  Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);    Destination destination = session.createQueue("CountTestQueue");      ActiveMQMessageProducer producer = (ActiveMQMessageProducer)session.createProducer(destination);    **for** (**int** i = 0; i < 3; i++) {  TextMessage message = session.createTextMessage("messageAA--" + i);    producer.setTransformer(**new** MessageTransformer() {    **public** Message producerTransform(Session session, MessageProducer producer,  Message msg) **throws** JMSException {    MapMessage message = session.createMapMessage();  message.setString(((TextMessage)msg).getText(), "my map message AAA=="+((TextMessage)msg).getText());  message.setStringProperty("extra", "okok");  **return** message;  }    **public** Message consumerTransform(Session arg0, MessageConsumer arg1,  Message arg2) **throws** JMSException {  // **TODO** Auto-generated method stub  **return** **null**;  }  });    producer.send(message);  }    session.commit();  session.close();  connection.close();  }  } |

## 2、创建消费者，采用事务，但是最后不提交

|  |
| --- |
| **public** **class** Receiver {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ActiveMQConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");      RedeliveryPolicy policy = **new** RedeliveryPolicy();  policy.setMaximumRedeliveries(3);  cf.setRedeliveryPolicy(policy);    Connection connection = cf.createConnection();  connection.start();  **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  Destination destination = session.createQueue("CountTestQueue");  ActiveMQMessageConsumer consumer = (ActiveMQMessageConsumer)session.createConsumer(destination);  **int** i = 0;  **while** (i < 2) {  MapMessage message = (MapMessage) consumer.receive();  System.*out*.println("收到消 息：" + message.getString("messageAA--" + i)  +" , property=="+message.getStringProperty("extra"));    i++;  }  session.close();  connection.close();  }  } |

## 3、测试运行，发现可以一直运行消费者接收消息，但是却只能接收，7次，因为重复了6次。1+6



## 4、添加代码策略，只能重复接收3次，在消费者代码中添加

|  |
| --- |
| **public** **class** Receiver {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ActiveMQConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");      RedeliveryPolicy policy = **new** RedeliveryPolicy();  policy.setMaximumRedeliveries(3);  cf.setRedeliveryPolicy(policy); |

## 5、运行发现，确实只能接受4次

