# 1分发策略

## 1、创建生产者，创建2个生产者，这两个生产者分别创建3个消息，通常情况下，consumer会以相同的顺序来接收来自同一个producer的消息

|  |
| --- |
| **public** **class** NoPerSender {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");    Connection connection = cf.createConnection();  connection.start();  Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);    Destination destination = session.createTopic("MyTopic");  **for**(**int** i=0;i<2;i++){  MessageProducer producer = session.createProducer(destination);    **for** (**int** j = 0; j < 3; j++) {  TextMessage message = session.createTextMessage(i+"==message222--" + j);    // 通过消息生产者发出消息  producer.send(message);  }  session.commit();  }  session.close();  connection.close();  }  } |

## 2、消费者，线程接收，创建了2个消费者

|  |
| --- |
| **public** **class** NoPerReceiver {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");    **for**(**int** i=0;i<2;i++){  Thread t = **new** MyT(cf);  t.start();  }    }  }  **class** MyT **extends** Thread{  **private** ConnectionFactory cf = **null**;  **public** MyT(ConnectionFactory cf){  **this**.cf = cf;  }  **public** **void** run(){  **try**{  Connection connection = cf.createConnection();  connection.start();    **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);    Destination destination = session.createTopic("MyTopic");    **for**(**int** i=0;i<1;i++){  **final** MessageConsumer consumer = session.createConsumer(destination);  consumer.setMessageListener(**new** MessageListener() {  **public** **void** onMessage(Message message) {  TextMessage txtMsg = (TextMessage)message;  **try**{  System.*out*.println(consumer+"收到消 息：" + txtMsg.getText());  session.commit();  }**catch**(Exception err){  err.printStackTrace();  }  }  });  }  }**catch**(Exception err){  err.printStackTrace();  }  }  } |

## 3、保证严格执行让每个消费者以相同的顺序来接收消息，

|  |
| --- |
| <policyEntry topic=">" >    <!-- 严格的分发策略 -->  <dispatchPolicy>  <strictOrderDispatchPolicy />  </dispatchPolicy>  <pendingMessageLimitStrategy>  <constantPendingMessageLimitStrategy limit="1000"/>  </pendingMessageLimitStrategy>  </policyEntry> |

## 4、测试执行，不太理解就认为每个消费者按顺序接收消息就完事了，注意的是，这里的topic是非持久的。点对点接收，所以才会出现下面的12条消息。

