# 1、消费者不使用线程，（线程中创建的消费者数量和消息数量一致，这样能够区分静态比正常的快）下面的则不能区分

## 1、创建两个消费者类61616和61617，每个只创建一个消费者

|  |
| --- |
| **public** **class** QR1 {  **public** **static** **void** main(String[] args) **throws** Exception {    String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");  Connection connection = cf.createConnection();  connection.start();    **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  Destination destination = session.createQueue("QR");//"my-queue");  **for**(**int** i=0;i<1;i++){  MessageConsumer consumer = session.createConsumer(destination);  consumer.setMessageListener(**new** MessageListener() {    **public** **void** onMessage(Message m) {  TextMessage msg = (TextMessage)m;  **try** {  System.*out*.println("QR1111111==="+msg.getText());  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }  }  });  }  } |

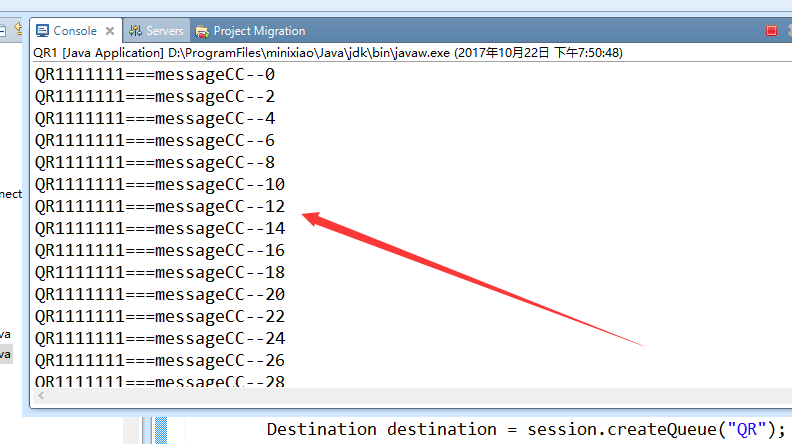
## 2、61617消费者，每个只创建一个消费者

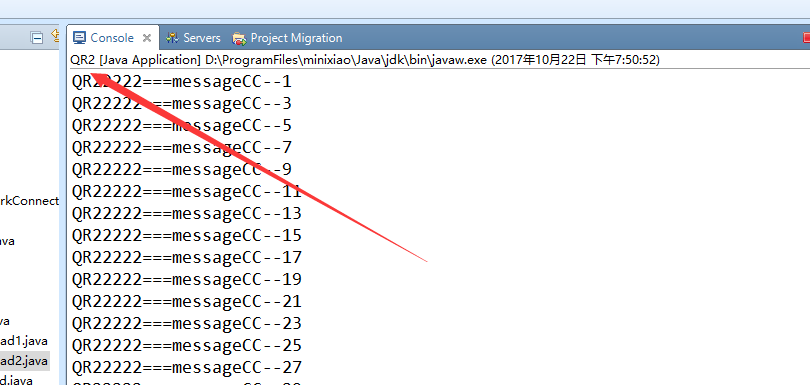
|  |
| --- |
| **public** **class** QR2 {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61617");    Connection connection = cf.createConnection();  connection.start();    **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  Destination destination = session.createQueue("QR");  **for**(**int** i=0;i<1;i++){  MessageConsumer consumer = session.createConsumer(destination);  consumer.setMessageListener(**new** MessageListener() {    **public** **void** onMessage(Message m) {  TextMessage msg = (TextMessage)m;  **try** {  System.*out*.println("QR22222==="+msg.getText());  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }  }  });  }      }  } |

## 3.、运行这连个消费者，然后再运行生产者，生产30个消息

|  |
| --- |
| **public** **class** QRQueueSender {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "failover:(tcp://"+linuxIp+":61616,tcp://"+linuxIp+":61617)?randomize=false");  Connection connection = cf.createConnection();  connection.start();  Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);    Destination destination = session.createQueue("QR");      MessageProducer producer = session.createProducer(destination);    **for** (**int** i = 0; i < 30; i++) {  TextMessage message = session.createTextMessage("messageCC--" + i);  producer.send(message);  }    session.commit();  session.close();  connection.close();  }  } |

## 4、观察控制台，发现两个消费者将消息均分了





# 2、均衡消费

## 这个时候，我们将QR2中创建两个消费者，上面的for循环设置为2发现控制台结果还是一样的，并没有分到多的消息，显然这和我们的集群的最终设想是违背的

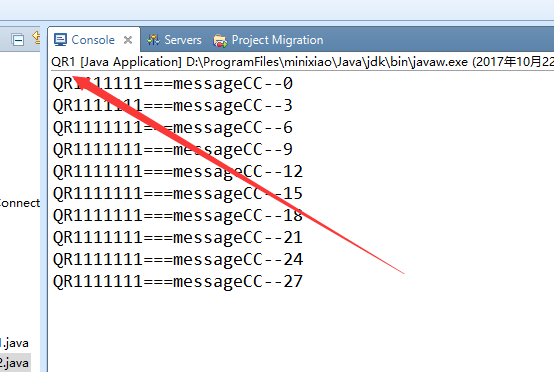
### 解释：conduitSubscriptions多个网络消费者是否被当做一个消费者来对待。，默认为true

### 1、现在将它设为false，不要让一个broker的多个客户端（消费者），被当做一个对待，

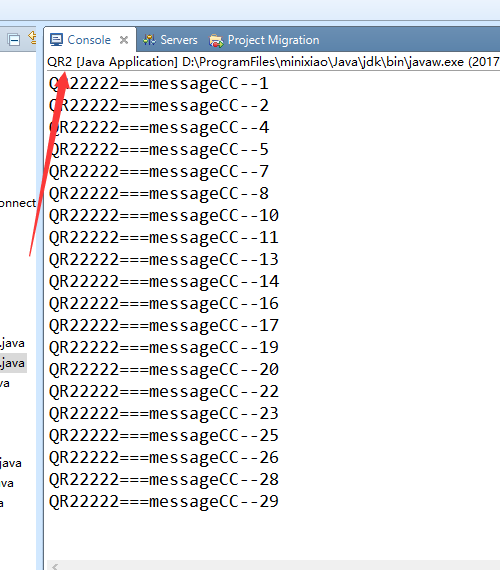
|  |
| --- |
|  |
| <networkConnectors>  <networkConnector  duplex="true"  conduitSubscriptions="false"  name="local network"  uri="static://(tcp://localhost:61616,tcp://localhost:61617)"/>  </networkConnectors> | | |

### 2、再次运行这两个消费者，然后再运行生产者，控制台结果如下

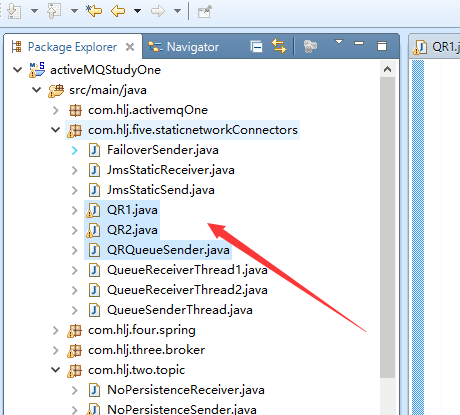
#### 1、QR1,收到10个消息



#### 2、QR2，收到20个消息



### 3、代码位置



# 3、mysql 数据源的均衡消费，master slave

## 1、两个broker的activemq.xml文件中配置好mysql的数据源，也就是不需要之前static的任何的东西，包括networkConnection和回流的配置。

## 2、broker1 61616

|  |
| --- |
| <persistenceAdapter>  <!-- <kahaDB directory="${activemq.data}/kahadb"/> -->  <jdbcPersistenceAdapter dataSource="#mysql-ds" />  </persistenceAdapter> |

|  |
| --- |
| <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?useUnicode=true&amp;characterEncoding=UTF-8"/>  <property name="username" value="root"/>  <property name="password" value="147094"/>  </bean> |