# 准备工作

## 1、设置一一个发送者61616 和2个接受者分别是61616 和61617

## 2、两个接受者中都采用线程的方式读取消息，（比较哪个接受者接到的消息多，前提是在生产者运行之前运行者两个消费者，因为如果一旦有消费者率先读取到消息之后（先开启生产者就会这样），就会霸占所有的消息，除非回流。明显会发现，static静态的方式61617接受的方式多）

# 开始

## 1、发送者61616

|  |
| --- |
| **public** **class** QueueSenderThread {  **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.*CLIENT\_ACKNOWLEDGE*);    Destination destination = session.createQueue("Thread.Consumer.queue");      MessageProducer producer = session.createProducer(destination);    **for** (**int** i = 0; i < 10; i++) {  TextMessage message = session.createTextMessage("messageCC--" + i);  producer.send(message);  }    session.commit();  session.close();  connection.close();  }  } |

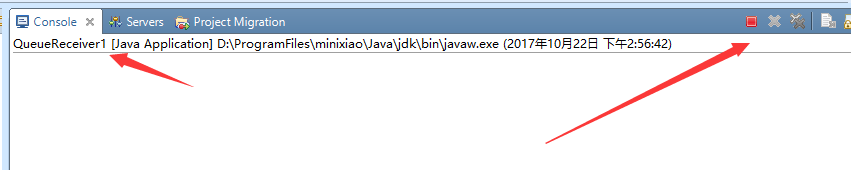
## 2、接受者61616 开启线程，为了防止线程太快，设置睡眠时间1秒，采用consumer.setMessageListener，设置监听的方式读取消息

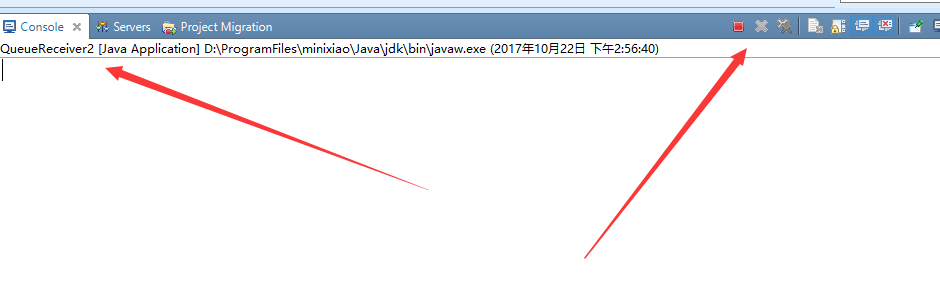
|  |
| --- |
| **public** **class** QueueReceiverThread1 {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");    **for**(**int** i=0;i<10;i++){  Thread t = **new** MyThread(cf);  t.start();  **try** {  Thread.*sleep*(1000L);  } **catch** (InterruptedException e1) {  // **TODO** Auto-generated catch block  e1.printStackTrace();  }  }  }    }  **class** MyThread **extends** Thread{  **private** ConnectionFactory cf = **null**;  **public** MyThread(ConnectionFactory cf){  **this**.cf = cf;  }  **public** **void** run(){  **try**{  **final** Connection connection = cf.createConnection();  connection.start();    **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  Destination destination = session.createQueue("Thread.Consumer.queue");//"my-queue");    MessageConsumer consumer = session.createConsumer(destination);  consumer.setMessageListener(**new** MessageListener() {    **public** **void** onMessage(Message msg) {  **try** {  TextMessage txtMsg = (TextMessage)msg;  System.*out*.println("Receiver11111===="+txtMsg.getText());  session.commit();  } **catch** (JMSException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  **try** {  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }  **try** {  session.close();  } **catch** (JMSException e) {  e.printStackTrace();  }  **try** {  connection.close();  } **catch** (JMSException e) {  e.printStackTrace();  }  }  });        }**catch**(Exception err){  err.printStackTrace();  }  }  } |

## 3、接受者61617，同上，只是端口不一样而已

|  |
| --- |
| **public** **class** QueueReceiverThread2 {  **public** **static** **void** main(String[] args) **throws** Exception {  String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61617");    **for**(**int** i=0;i<10;i++){  Thread t = **new** MyThread2(cf);  t.start();  **try** {  Thread.*sleep*(1000L);  } **catch** (InterruptedException e1) {  // **TODO** Auto-generated catch block  e1.printStackTrace();  }  }  }  }  **class** MyThread2 **extends** Thread{  **private** ConnectionFactory cf = **null**;  **public** MyThread2(ConnectionFactory cf){  **this**.cf = cf;  }  **public** **void** run(){  **try**{  **final** Connection connection = cf.createConnection();  connection.start();    **final** Session session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  Destination destination = session.createQueue("Thread.Consumer.queue");//"my-queue");    MessageConsumer consumer = session.createConsumer(destination);  consumer.setMessageListener(**new** MessageListener() {    **public** **void** onMessage(Message msg) {  **try** {  TextMessage txtMsg = (TextMessage)msg;  System.*out*.println("Receiver11111===="+txtMsg.getText());  session.commit();  } **catch** (JMSException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  **try** {  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }  **try** {  session.close();  } **catch** (JMSException e) {  e.printStackTrace();  }  **try** {  connection.close();  } **catch** (JMSException e) {  e.printStackTrace();  }  }  });        }**catch**(Exception err){  err.printStackTrace();  }  } |

## 4、运行这个两个接受者，等待发送者运行





## 5、运行发送者，发现61617收到的消息明显比61616收到的消息多，这样就说明static快

# 代码位置

