# 1、基本命令

## 1、启动命令

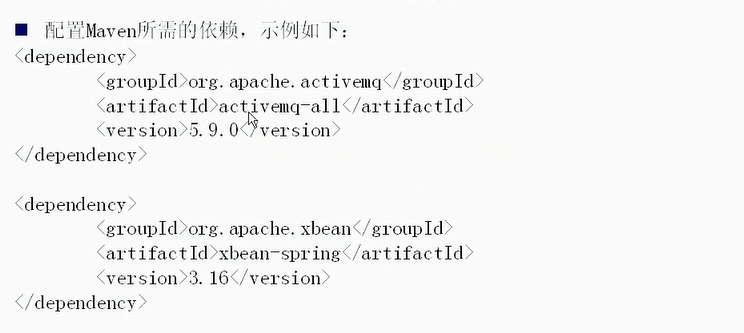
### ./activemq start 重启 restart

## 2、关闭命令

### activemq stop

# 1、开始第一个项目（maven项目）

## 1、配置Maven所需的依赖



## 2.建立生产者进行信息的发送

|  |
| --- |
| **public** **class** Sender {  **private** **static** **final** **int** *SEND\_NUMBER* = 5;  **public** **static** **void** main(String[] args) {  // ConnectionFactory ：连接工厂，JMS 用它创建连接  ConnectionFactory connectionFactory;  // Connection ：JMS 客户端到JMS Provider 的连接  Connection connection = **null**;  // Session： 一个发送或接收消息的线程  Session session;  // Destination ：消息的目的地;消息发送给谁.也就是创建队列  Destination destination;  // MessageProducer：消息发送者  MessageProducer producer;  // TextMessage message;  // 构造ConnectionFactory实例对象，此处采用ActiveMq的实现jar  connectionFactory = **new** ActiveMQConnectionFactory(  ActiveMQConnection.*DEFAULT\_USER*,  ActiveMQConnection.*DEFAULT\_PASSWORD*,  "tcp://localhost:61616");  **try** {  // 构造从工厂得到连接对象  connection = connectionFactory.createConnection();  // 启动  connection.start();  // 获取操作连接  session = connection.createSession(Boolean.*TRUE*,  Session.*AUTO\_ACKNOWLEDGE*);  // 获取session注意参数值xingbo.xu-queue是一个服务器的queue，须在在ActiveMq的console配置  destination = session.createQueue("HealerJean-FirstQueue");  // 得到消息生成者【发送者】  producer = session.createProducer(destination);  // 设置不持久化，此处学习，实际根据项目决定  producer.setDeliveryMode(DeliveryMode.*NON\_PERSISTENT*);  // 构造消息，此处写死，项目就是参数，或者方法获取  *sendMessage*(session, producer);  session.commit();  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (**null** != connection)  connection.close();  } **catch** (Throwable ignore) {  }  }  }  **public** **static** **void** sendMessage(Session session, MessageProducer producer)  **throws** Exception {  **for** (**int** i = 1; i <= *SEND\_NUMBER*; i++) {  TextMessage message = session  .createTextMessage("ActiveMq 发送的消息" + i);  // 发送消息到目的地方  System.*out*.println("发送消息：" + "ActiveMq 发送的消息" + i);  producer.send(message);  }  }  } |

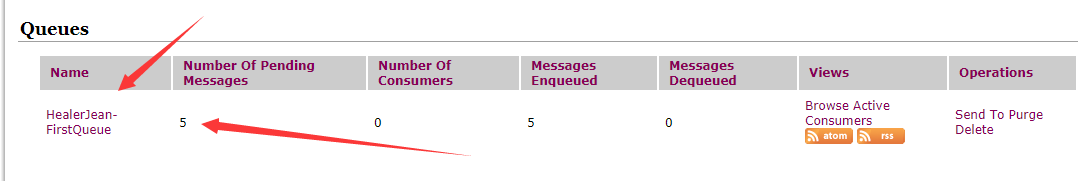
### [Number Of Pending Messages](http://localhost:8161/admin/queues.jsp)   表示等待的消息数量

### [Number Of Consumers](http://localhost:8161/admin/queues.jsp)   消费者的数量（目前消费者的数量）

### [Messages Enqueued](http://localhost:8161/admin/queues.jsp)   入队的数量（表示已经产生的消息的数量）

### [Messages Dequeued](http://localhost:8161/admin/queues.jsp)  出队的数量（表示已经被消费者使用的数量）

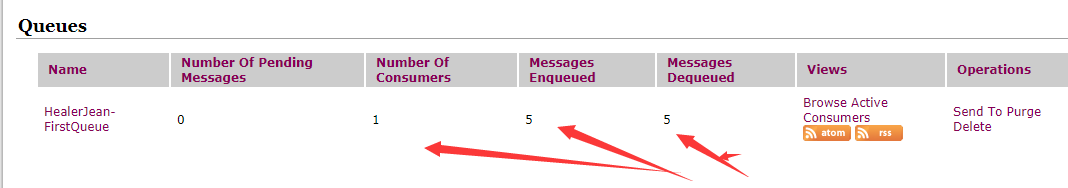
### 图解



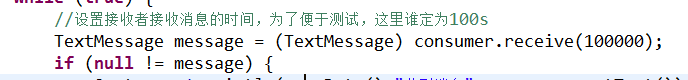
## 3、建立消费者进行信息的接收

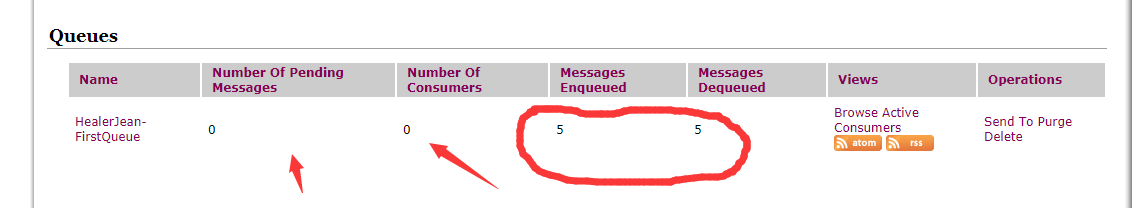
|  |
| --- |
| **public** **class** Receiver {  **public** **static** **void** main(String[] args) {  // ConnectionFactory ：连接工厂，JMS 用它创建连接  ConnectionFactory connectionFactory;  // Connection ：JMS 客户端到JMS Provider 的连接  Connection connection = **null**;  // Session： 一个发送或接收消息的线程  Session session;  // Destination ：消息的目的地;消息发送给谁.  Destination destination;  // 消费者，消息接收者  MessageConsumer consumer;  connectionFactory = **new** ActiveMQConnectionFactory(  ActiveMQConnection.*DEFAULT\_USER*,  ActiveMQConnection.*DEFAULT\_PASSWORD*,  "tcp://localhost:61616");  **try** {  // 构造从工厂得到连接对象  connection = connectionFactory.createConnection();  // 启动  connection.start();  // 获取操作连接  // 获取操作连接  // Boolean.FALSE, 是否开启事物 false 表示不开启事物，如果开启true之后，可以再最后面进行批量发送  // AUTO\_ACKNOWLEDGE 自动签收  session = connection.createSession(Boolean.*FALSE*,  Session.*AUTO\_ACKNOWLEDGE*);  // 获取session注意参数值xingbo.xu-queue是一个服务器的queue，须在在ActiveMq的console配置  destination = session.createQueue("HealerJean-FirstQueue");  consumer = session.createConsumer(destination);  **while** (**true**) {  //设置接收者接收消息的时间，为了便于测试，这里谁定为100s  TextMessage message = (TextMessage) consumer.receive(100000);  **if** (**null** != message) {  System.*out*.println("收到消息" + message.getText());  } **else** {  **break**;  }  }  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (**null** != connection)  connection.close();  } **catch** (Throwable ignore) {  }  }  }  } |

### 图解



## 4、自我理解，有了以下时间配置，不运行生产者，只运行消费者也是可以的。当有生产者出现的的时候，消费者会自动将消息队列中的数据读取过来，亲测是这样。最终如果二者都会消失





# 2、使用MapMessage

## 1、send

|  |
| --- |
| **public** **static** **void** sendMessage(Session session, MessageProducer producer)  **throws** Exception {  **for** (**int** i = 1; i <= 5; i++) {  // TextMessage message = session  // .createTextMessage("ActiveMq 发送的消息" + i);    MapMessage mapMessage = session.createMapMessage();  mapMessage.setStringProperty("username"+i, "HealerJean"+i);  mapMessage.setStringProperty("usernameZh"+i, "张宇晋"+i);    mapMessage.setString("mapMessage"+i, "mapMessage setString "+i);    // 发送消息到目的地方  System.*out*.println("发送消息：" + "ActiveMq 发送的消息" + i);  producer.send(mapMessage);  }  } |

## 2、receiver

|  |
| --- |
| **for** (**int** i = 1; i <= 5; i++) {  //设置接收者接收消息的时间，为了便于测试，这里谁定为100s  /\* TextMessage message = (TextMessage) consumer.receive(100000);  if (null != message) {  System.out.println(new Date()+"收到消息" + message.getText());  } else {  break;  }\*/  MapMessage message = (MapMessage) consumer.receive();    //  System.*out*.println("收到消 息：" + message.getString("mapMessage"+i)  +" , property=="+message.getStringProperty("username"+i)  +" , property=="+message.getStringProperty("usernameZh"+i));    } |

## 3、打印结果

