## 1、直接send

|  |
| --- |
| **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);  }  } |

## 2、roducer.setTransformer(new MessageTransformer()

|  |
| --- |
| **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**;  }  });    // Thread.sleep(1000);  // 通过消息生产者发出消息  producer.send(message);  }    session.commit();  session.close();  connection.close();  } |

## 3、spring，queueSender.jmsTemplate.send(new MessageCreator()

|  |
| --- |
| queueSender.jmsTemplate.send(**new** MessageCreator() {  **public** Message createMessage(Session session) **throws** JMSException {  TextMessage msg = session.createTextMessage("Spring msg（ spring 创建生产者成功）");  System.*out*.println("spring 创建 queque生产者成功");  **return** msg;  }  }); |

# 接收，

## 1、spring queueReceiver.jmsTemplate.receiveAndConvert();

|  |
| --- |
| @Service  **public** **class** QueueReceiver {  @Autowired  **private** JmsTemplate jmsTemplate = **null**;  **public** **static** **void** main(String[] args) **throws** Exception {  ApplicationContext context = **new** ClassPathXmlApplicationContext("spring-queue-applicationContext.xml");  QueueReceiver queueReceiver = (QueueReceiver) context.getBean("queueReceiver");    String msg = (String) queueReceiver.jmsTemplate.receiveAndConvert();    System.*out*.println("msg===" + msg);  }  } |

## 2、监听器接收

|  |
| --- |
| 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();  }  }  }); |