# 1、queue

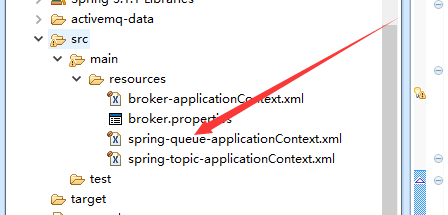
## 1、添加spring的jar包依赖（不知道为什么，我通过pom引入的spring依赖包，一直测试没有成功，后来用eclipse导入的spring3.0）

|  |
| --- |
| <!-- 2、spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jms</artifactId>  <version>4.0.0.RELEASE</version>  </dependency> |

## 2、配置spring配置文件spring-queue-applicationContext.xml

### 1、配置activemq

|  |
| --- |
| <bean id=*"jmsFactory"* class=*"org.apache.activemq.pool.PooledConnectionFactory"*  destroy-method=*"stop"*>  <property name=*"connectionFactory"*>  <bean class=*"org.apache.activemq.ActiveMQConnectionFactory"*>  <property name=*"brokerURL"*>  <value>tcp://localhost:61616</value>  </property>  </bean>  </property>  <property name=*"maxConnections"* value=*"100"*></property>  </bean>      <bean id=*"destination"* class=*"org.apache.activemq.command.ActiveMQQueue"*>  <constructor-arg index=*"0"* value=*"spring-queue"* />  </bean>    <bean id=*"jmsTemplate"* class=*"org.springframework.jms.core.JmsTemplate"*>  <property name=*"connectionFactory"* ref=*"jmsFactory"* />  <property name=*"defaultDestination"* ref=*"destination"* />  <property name=*"messageConverter"*>  <bean class=*"org.springframework.jms.support.converter.SimpleMessageConverter"* />  </property>  </bean> |



### 2、添加包的扫描

|  |
| --- |
| <context:component-scan base-package=*"com.hlj.four.spring"*>  <context:exclude-filter type=*"annotation"*  expression=*"org.springframework.stereotype.Controller"* />  </context:component-scan>  <aop:aspectj-autoproxy proxy-target-class=*"true"*/> |

### 3、spring 配置文件头部链接（不会的话记得复制）

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd"*  > |

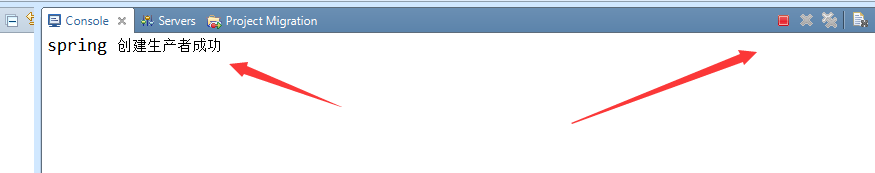
## 3、测试创建生产者

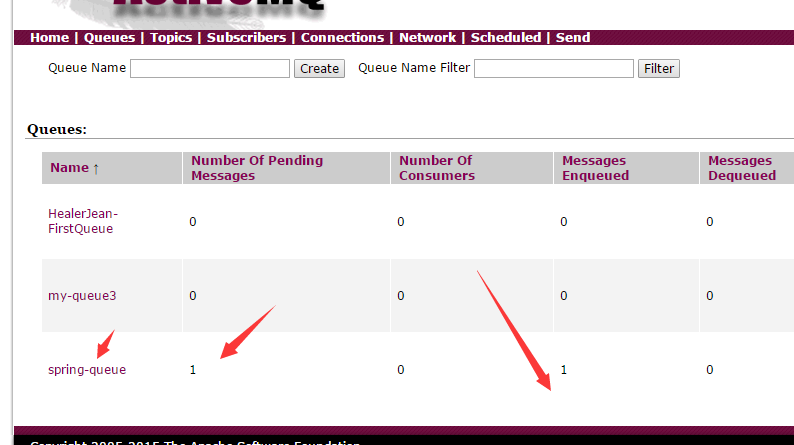
|  |
| --- |
| @Service  **public** **class** QueueSender {  @Autowired  **private** JmsTemplate jmsTemplate = **null**;  **public** **static** **void** main(String[] args)**throws** Exception {  ApplicationContext ctx = **new** ClassPathXmlApplicationContext("spring-queue-applicationContext.xml");  QueueSender queueSender = (QueueSender)ctx.getBean("queueSender");  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 创建生产者成功"); return msg; }  });  }  } |

## 4、测试创建消费者

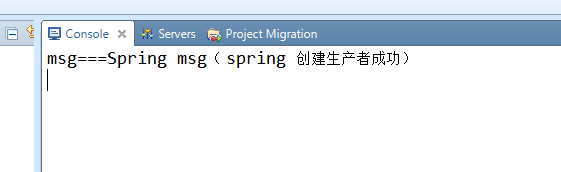
|  |
| --- |
| @Service  **public** **class** QueueReceiver {  @Autowired  **private** JmsTemplate jmsTemplate = **null**;  **public** **static** **void** main(String[] args) **throws** Exception {  ApplicationContext context = **new** ClassPathXmlApplicationContext("spring- queue-applicationContext.xml");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

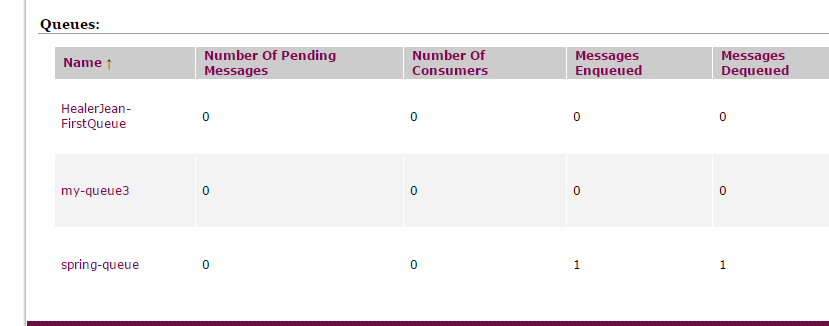
## 5、运行生产者，控制台、浏览器中观察





## 6、运行消费者，接收消息

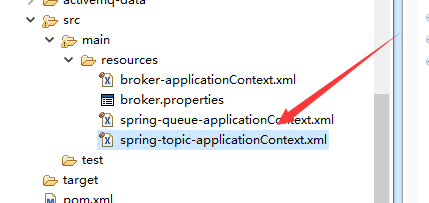




# 2、topic

## 1、spring配置文件修改目的地， spring-topic-applicationContext.xml

|  |
| --- |
| <bean id=*"destinationTopic"* class=*"org.apache.activemq.command.ActiveMQTopic"*>  <constructor-arg index=*"0"* value=*"spring-topic"* />  </bean>    <bean id=*"jmsTemplate"* class=*"org.springframework.jms.core.JmsTemplate"*>  <property name=*"connectionFactory"* ref=*"jmsFactory"* />  <property name=*"defaultDestination"* ref=*"destinationTopic"* />  <property name=*"messageConverter"*>  <bean class=*"org.springframework.jms.support.converter.SimpleMessageConverter"* />  </property>  </bean> |



## 2、生产者和消费者不变，测试成功

# 3、MessgeListener

## 1、spring配置，添加一个类似于监听器的东西spring-msglistener-applicationContext.xml

|  |
| --- |
| <bean id=*"jmsContainer"*  class=*"org.springframework.jms.listener.DefaultMessageListenerContainer"*>  <property name=*"connectionFactory"* ref=*"jmsFactory"* />  <property name=*"destination"* ref=*"destination"* />  <property name=*"messageListener"* ref=*"messageListener"* />  </bean>  <bean id=*"messageListener"*  class=*"com.hlj.four.spring.MyMessageListener"*>  </bean> |

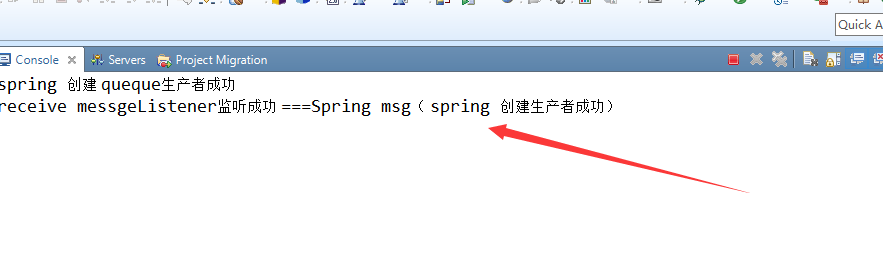
## 2、创建自定义的MymessageListener

|  |
| --- |
| **public** **class** MyMessageListener **implements** MessageListener {  **public** **void** onMessage(Message msg) {  TextMessage txtMsg = (TextMessage)msg;  **try** {  System.*out*.println("receive txt msg==="+txtMsg.getText());  } **catch** (JMSException e) {  e.printStackTrace();  }  }  } |

## 3、只建立一个生产者，上面的message监听器会自动监听

|  |
| --- |
| @Service  **public** **class** QueueMessageListenerSender {  @Autowired  **private** JmsTemplate jmsTemplate = **null**;    **public** **static** **void** main(String[] args)**throws** Exception {  ApplicationContext ctx = **new** ClassPathXmlApplicationContext("spring-msglistener-applicationContext.xml");  QueueMessageListenerSender queueSender = (QueueMessageListenerSender)ctx.getBean("queueMessageListenerSender");  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;  }  });  }  } |

## 4、运行测试



# 4、后期加的。使用propetries

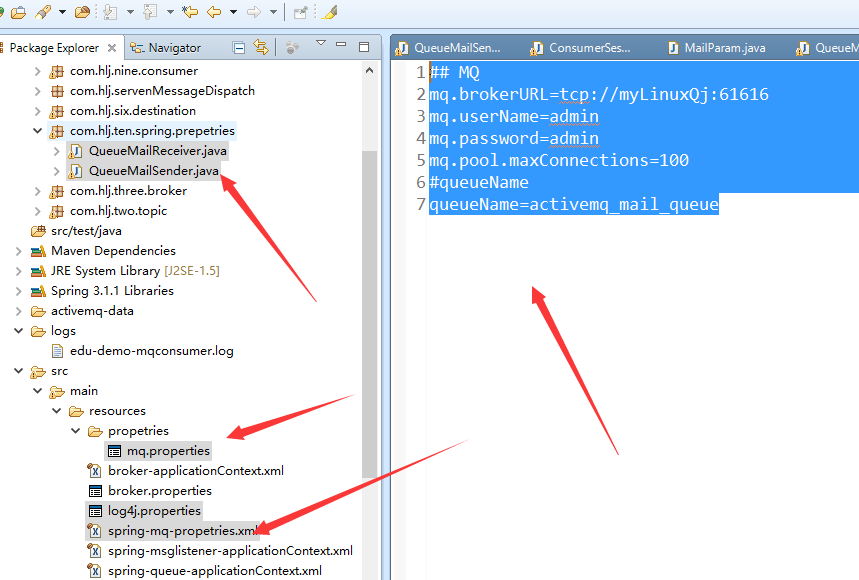
## 1、spring-mq-prepetries.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.2.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.2.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-3.2.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.2.xsd"*  default-autowire=*"byName"* default-lazy-init=*"false"*>      <!-- 采用注释的方式配置bean -->  <context:annotation-config />  <!-- 配置要扫描的包 -->  <context:component-scan base-package=*"com.hlj.ten.spring.mail"* />  <!-- 读入配置属性文件 -->  <context:property-placeholder location=*"classpath:propetries/mq.properties"* />  <!-- proxy-target-class默认"false",更改为"ture"使用CGLib动态代理 -->  <aop:aspectj-autoproxy proxy-target-class=*"true"* />    <!-- 真正可以产生Connection的ConnectionFactory，由对应的 JMS服务厂商提供 -->  <bean id=*"targetConnectionFactory"* class=*"org.apache.activemq.ActiveMQConnectionFactory"*>  <!-- ActiveMQ服务地址 -->  <property name=*"brokerURL"* value=*"${mq.brokerURL}"* />  <property name=*"userName"* value=*"${mq.userName}"*></property>  <property name=*"password"* value=*"${mq.password}"*></property>  </bean>  <!--  ActiveMQ为我们提供了一个PooledConnectionFactory，通过往里面注入一个ActiveMQConnectionFactory  可以用来将Connection、Session和MessageProducer池化，这样可以大大的减少我们的资源消耗。  要依赖于 activemq-pool包  -->  <bean id=*"pooledConnectionFactory"* class=*"org.apache.activemq.pool.PooledConnectionFactory"*>  <property name=*"connectionFactory"* ref=*"targetConnectionFactory"* />  <property name=*"maxConnections"* value=*"${mq.pool.maxConnections}"* />  </bean>  <!-- Spring用于管理真正的ConnectionFactory的ConnectionFactory -->  <bean id=*"connectionFactory"* class=*"org.springframework.jms.connection.SingleConnectionFactory"*>  <!-- 目标ConnectionFactory对应真实的可以产生JMS Connection的ConnectionFactory -->  <property name=*"targetConnectionFactory"* ref=*"pooledConnectionFactory"* />  </bean>    <!-- Spring提供的JMS工具类，它可以进行消息发送、接收等 -->    <!-- 队列模板 -->  <bean id=*"activeMqJmsTemplate"* class=*"org.springframework.jms.core.JmsTemplate"*>  <!-- 这个connectionFactory对应的是我们定义的Spring提供的那个ConnectionFactory对象 -->  <property name=*"connectionFactory"* ref=*"connectionFactory"*/>  <property name=*"defaultDestinationName"* value=*"${queueName}"*></property>  </bean>  </beans> |

## 2、mq.properties

|  |
| --- |
| ## MQ  mq.brokerURL=tcp://myLinuxQj:61616  mq.userName=admin  mq.password=admin  mq.pool.maxConnections=100  #queueName  queueName=activemq\_mail\_queue |

## 3、项目结构



## 4、正常测试生产者

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