ActiveMQ

(V5.11)

# 概述

JMS是一个Java平台中关于面向消息中间件的API，用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。Java消息服务是一个与具体平台无关的API。

ActiveMQ是JMS(Java Message Server)规范的一个具体实现。

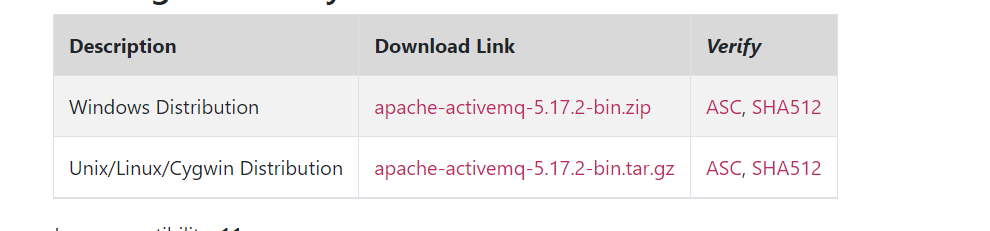
JMS是一种与厂商无关的 API，用来访问收发系统消息，类似JDBC(JDBC是Java提供的一套访问关系型数据库的规范API)

JMS的产品有ActiceMQ、Kafka、RebitMQ、RocketMQ等

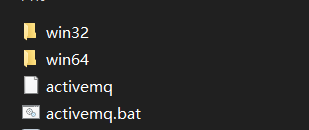
# 安装

下载地址：<https://github.com/apache/activemq>

或 <https://activemq.apache.org/activemq-5017002-release>



下载完成后根据当前系统选择对应的启动文件



启动之后可通过<http://localhost:8161> 访问控制台

登录账号/密码 admin/admin

# 依赖包

<dependency>

<groupId>org.apache.activemq</groupId>

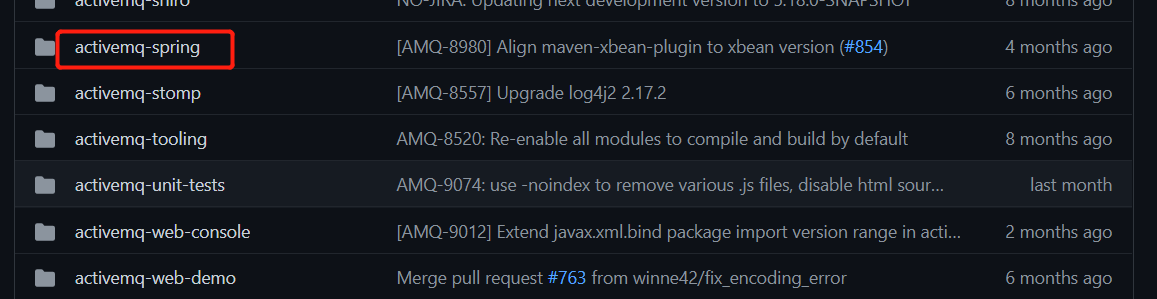
<artifactId>activemq-core</artifactId>

<version>5.7.0</version>

</dependency>

如果是与spring集成 可以参考activemq在github中对应的demo

https://github.com/apache/activemq.git



# 点对点消息发送与接收

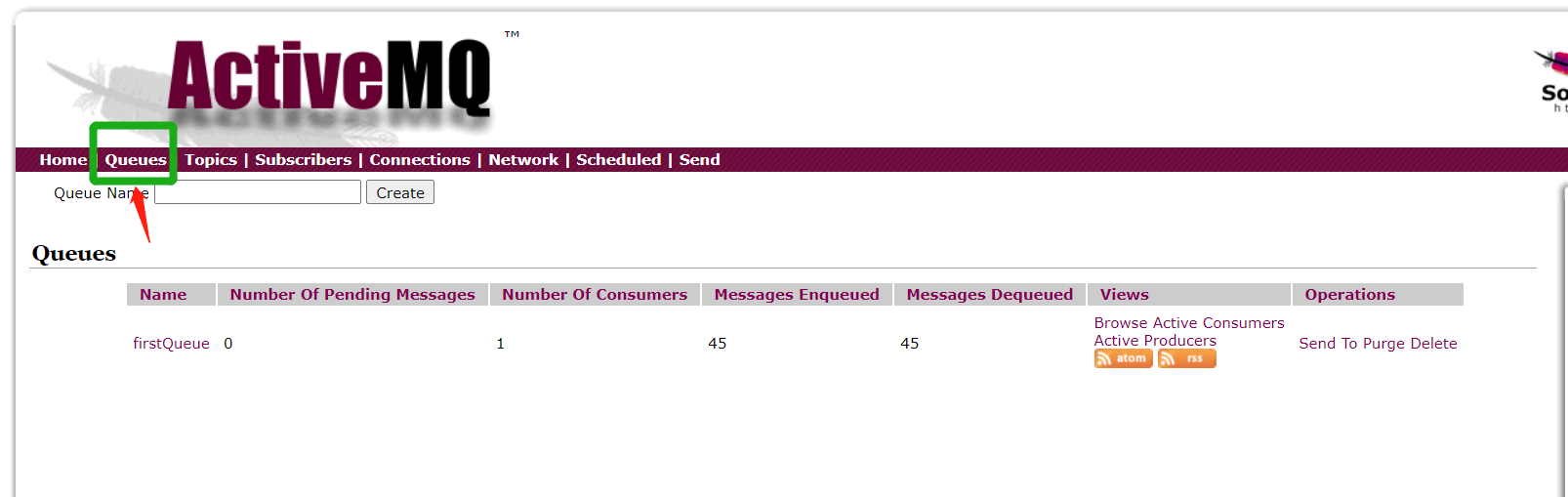
点对点模式发送消息与接收消息 指的是生产者与消费者都到指定的队列中读取消息 即 生产者指定消息队列并发送消息，然后消费者到同一个队列中获取消息。

分为两种方式 receive和listener

receive方式即消费者通过MessageConsumer.receive()方法获取消息内容 此方式是一直在与生产者相同的队列中等待生产者生产消息。

listener方式 即 通过实现MessageListener接口中的onMessage方法来获取消息，消费者只需要将监听注入进来(消息消费者)即可 不用一直在等待生产者生产消息

点对点生产者和消费者都是在Queues中发送和获取消息

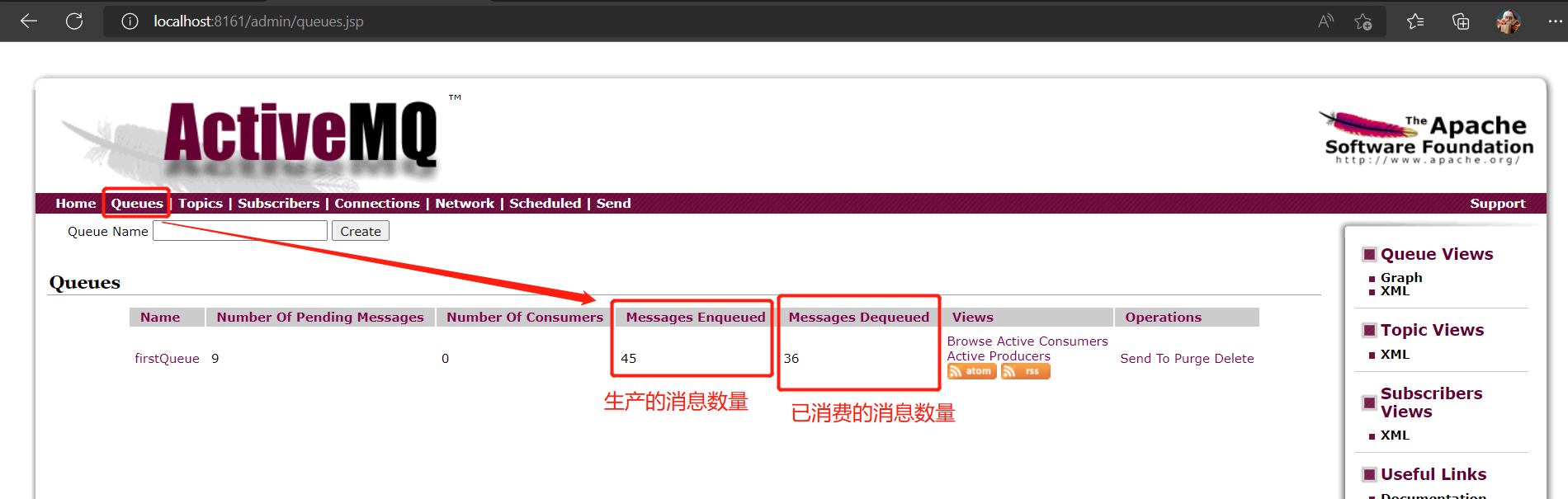


## 生产者

receive和listener两种方式只是针对消费者 ，那么生产者生产消息是共用的

## 消息生产者

|  |
| --- |
| **package** jms;  **import** javax.jms.Connection;  **import** javax.jms.ConnectionFactory;  **import** javax.jms.Destination;  **import** javax.jms.JMSException;  **import** javax.jms.MessageProducer;  **import** javax.jms.Session;  **import** javax.jms.TextMessage;  **import** org.apache.activemq.ActiveMQConnection;  **import** org.apache.activemq.ActiveMQConnectionFactory;  **import** lombok.extern.slf4j.Slf4j;  /\*\*  \* 消息生产者  \* 点对点 发送和接收消息  \* **@author** etjava  \*  \*/  @Slf4j  **public** **class** ProducerTest {  **private** **static** **final** String ***USERNAME*** = ActiveMQConnection.***DEFAULT\_USER***; // admin  **private** **static** **final** String ***PASSWORD*** = ActiveMQConnection.***DEFAULT\_PASSWORD***; // admin  **private** **static** **final** String ***BROKEURL*** = ActiveMQConnection.***DEFAULT\_BROKER\_URL***;// 默认的连接地址 tcp://localhost:61616  **private** **static** **final** **int** ***NUMS*** = 10; // 消息数量    **public** **static** **void** main(String[] args) {  // 创建连接Factory  ConnectionFactory factory = **null**;  // 获取连接  Connection conn = **null**;  // 创建Session 用于发送和接收消息的线程  Session session = **null**;  // 定义消息的路由 - 消息的目的地  Destination destination = **null**;  // 创建消息生产者  MessageProducer msgProducer = **null**;      // 实例化连接工厂  factory = **new** ActiveMQConnectionFactory(***USERNAME***, ***PASSWORD***, ***BROKEURL***);    **try** {  // 在连接工厂中获取连接  conn = factory.createConnection();  // 打开连接  conn.start();  /\*  \* 获取session - session会话是根据具体的每个连接获取的  \* 参数解释：  \* 参数1 表示是否开启事务，true为开启  \* 参数2 消息的确认方式 Session.AUTO\_ACKNOWLEDGE 表示自动确认  \*  \* 直接receive(接收)消息方式 有三种 分别如下  \* Session.AUTO\_ACKNOWLEDGE [推荐]  \* 当客户端成功的接收方法返回的时候 或是从监听(MessageListener.onMessage)方法成功返回的时候 会话会自动确认客户端收到了消息  \*  \* Session.CLIENT\_ACKNOWLEDGE  \* 客户端通过消息的acknowledge方法确认接收到消息，该模式是在会话层上进行确认的  \* 例如 生产了10条消息 当第五条消息被确认消费后 那么这10条消息都会被确认为已消费  \*  \* Session.DUPS\_OK\_ACKNOWLEDGE  \* 这种模式当出现会话迟钝(一直接收不到消息被消费状态)会出现重复确认消息 如果生产者确认失败 那么可能就会再次发送确认消息  \*  \*/  session = conn.createSession(**true**, Session.***AUTO\_ACKNOWLEDGE***);  // 创建消息队列 - 会返回消息的目的地  destination = session.createQueue("firstQueue");  // session.createQueue("secondQueue");  // 创建消息生产者 - 返回消息生产者实例  msgProducer = session.createProducer(destination);  // 发送消息到队列中  *sendMsg*(session, msgProducer);  // 提交事务 - session在创建的时候设置了事务 因此这里需要提交事务才能真正的发送到队列  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }**finally** {  **try** {  **if**(conn!=**null**)  conn.close();  } **catch** (JMSException e) {  ***log***.error("连接关闭异常 ",e);  }  }  }    // 发送消息  **private** **static** **void** sendMsg(Session session,MessageProducer produce) **throws** JMSException {  // 模拟发送消息  **for**(**int** i = 1; i< ***NUMS***; i++) {  // 创建简单的文本消息  TextMessage textMessage = session.createTextMessage("消息信息 - "+i);  System.***out***.println("消息信息 - "+i);  // 发送消息  produce.send(textMessage);  }  }  } |



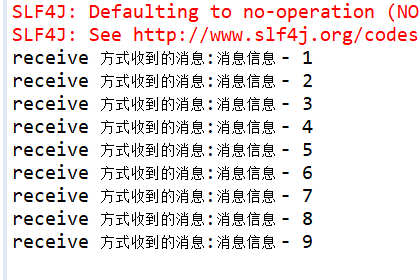
## receive方式

### 消息消费者

receive方式是直接获取消息的方式 但这种方式需要一直等待对应的队列中消息，也就是说 始终有一个线程一直处于等待和接收消息

while(true){}

|  |
| --- |
| package jms;  import javax.jms.Connection;  import javax.jms.ConnectionFactory;  import javax.jms.Destination;  import javax.jms.JMSException;  import javax.jms.MessageConsumer;  import javax.jms.Session;  import javax.jms.TextMessage;  import org.apache.activemq.ActiveMQConnection;  import org.apache.activemq.ActiveMQConnectionFactory;  import lombok.extern.slf4j.Slf4j;  /\*\*  \* 消息消费者  \* 点对点 发送和接收消息  \* @author etjav  \*  \*/  @Slf4j  public class ConsumerTest {  private static final String USERNAME = ActiveMQConnection.DEFAULT\_USER; // admin  private static final String PASSWORD = ActiveMQConnection.DEFAULT\_PASSWORD; // admin  private static final String BROKEURL = ActiveMQConnection.DEFAULT\_BROKER\_URL;// 默认的连接地址 tcp://localhost:61616  public static void main(String[] args) {  // 创建连接Factory  ConnectionFactory factory = null;  // 获取连接  Connection conn = null;  // 创建Session 用于发送和接收消息的线程  Session session = null;  // 定义消息的路由 - 消息的目的地  Destination destination = null;  // 消息消费者  MessageConsumer consumer = null;    // 实例化连接工厂  factory = new ActiveMQConnectionFactory(USERNAME, PASSWORD, BROKEURL);    try {  // 在连接工厂中获取连接  conn = factory.createConnection();  // 打开连接  conn.start();  // 获取session - 每个连接对应一个session 获取消息不需要添加事务控制  session = conn.createSession(false, Session.AUTO\_ACKNOWLEDGE);  // 创建连接的消息队列 - 会返回消息的目的地  destination = session.createQueue("firstQueue");  // 创建消息消费者 - 返回消息消费者实例  consumer = session.createConsumer(destination);  // 发送消息到队列中  getMsg(consumer);  // 提交事务 - session在创建的时候设置了事务 因此这里需要提交事务才能真正的发送到队列  session.commit();  } catch (JMSException e) {  e.printStackTrace();  }finally {  try {  if(conn!=null)  conn.close();  } catch (JMSException e) {  log.error("连接关闭异常 ",e);  }  }  }  /\*\*  \* 接收消息  \* @param consumer 消息消费者  \* @throws JMSException  \*/  private static void getMsg(MessageConsumer consumer) throws JMSException {  // receive 方式【不推荐使用】  while(true) {  TextMessage msg = (TextMessage) consumer.receive();  if(msg!=null) {  System.out.println("receive 方式收到的消息:"+msg.getText());  }else {  // 没有收到消息 结束循环  break;  }  }  }  } |





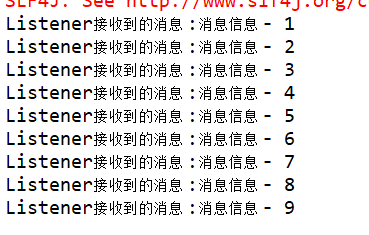
## listener方式

### 消息消费者

先创建监听 通过实现MessageListener.onMessage()方法来创建

然后在消费者获取消息时通过  
consumer.setMessageListener(**new** Linstener());注入监听来获取消息

|  |
| --- |
| **package** jms;  **import** javax.jms.Connection;  **import** javax.jms.ConnectionFactory;  **import** javax.jms.Destination;  **import** javax.jms.JMSException;  **import** javax.jms.Message;  **import** javax.jms.MessageConsumer;  **import** javax.jms.MessageListener;  **import** javax.jms.Session;  **import** javax.jms.TextMessage;  **import** org.apache.activemq.ActiveMQConnection;  **import** org.apache.activemq.ActiveMQConnectionFactory;  /\*\*  \* 点对点 发送和接收消息  \* 消息消费者 - 监听方式获取消息  \* 会监听是否存在未消费的消息 有则取出  \* **@author** etjav  \*  \*/  **public** **class** ConsumerWithListenerTest {  **private** **static** **final** String ***USERNAME*** = ActiveMQConnection.***DEFAULT\_USER***; // admin  **private** **static** **final** String ***PASSWORD*** = ActiveMQConnection.***DEFAULT\_PASSWORD***; // admin  **private** **static** **final** String ***BROKEURL*** = ActiveMQConnection.***DEFAULT\_BROKER\_URL***;// 默认的连接地址 tcp://localhost:61616  **public** **static** **void** main(String[] args) {  // 创建连接工厂  ConnectionFactory factory = **new** ActiveMQConnectionFactory(***USERNAME***, ***PASSWORD***, ***BROKEURL***);  // 获取连接  Connection conn = **null**;  **try** {  // 获取连接  conn = factory.createConnection();  // 启动连接  conn.start();  // 创建session  Session session = conn.createSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);  // 获取连接的目的地  Destination destination = session.createQueue("firstQueue");  // 创建消费者  MessageConsumer consumer = session.createConsumer(destination);  // 接收消息 - 监听方式  consumer.setMessageListener(**new** Linstener());  } **catch** (JMSException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  }  /\*\*  \* 消息监听  \* **@author** etjav  \*  \*/  **class** Linstener **implements** MessageListener{  @Override  **public** **void** onMessage(Message message) {  **try** {  // 接收到的消息  System.***out***.println("Listener:"+((TextMessage)message).getText());  } **catch** (JMSException e) {  e.printStackTrace();  }  }    } |

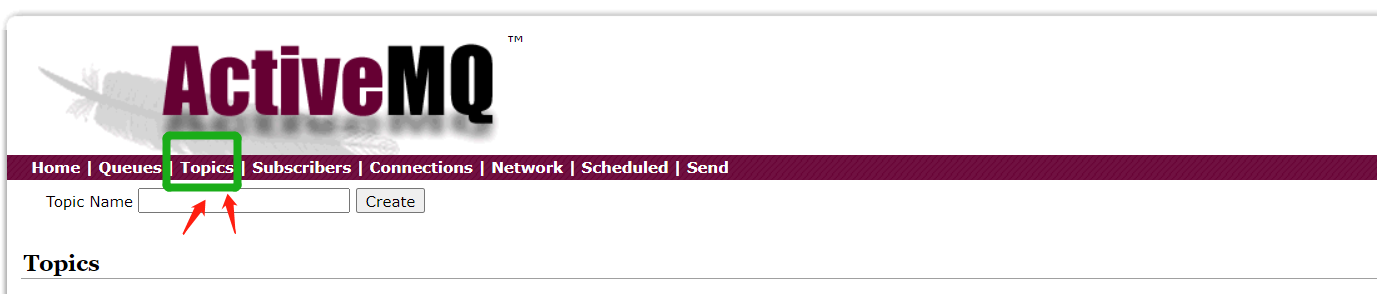




# 订阅与发布

一个生产者对应多个消费者

生产者和消费者对应的是Topics



注意：**先订阅才能接收到生产者发布的消息**

## 生产者

与点对点创建方式相同 区别只在于创建队列改为创建topic   
即session.createTopic("topic\_1");

|  |
| --- |
| **package** jms.topic;  **import** javax.jms.Connection;  **import** javax.jms.ConnectionFactory;  **import** javax.jms.Destination;  **import** javax.jms.JMSException;  **import** javax.jms.MessageProducer;  **import** javax.jms.Session;  **import** javax.jms.TextMessage;  **import** org.apache.activemq.ActiveMQConnection;  **import** org.apache.activemq.ActiveMQConnectionFactory;  **import** lombok.extern.slf4j.Slf4j;  /\*\*  \* 消息生产者 - 发布订阅模式  \* **@author** etjav  \*  \*/  @Slf4j  **public** **class** ProducerWithTopicTest {  **private** **static** **final** String ***USERNAME*** = ActiveMQConnection.***DEFAULT\_USER***; // admin  **private** **static** **final** String ***PASSWORD*** = ActiveMQConnection.***DEFAULT\_PASSWORD***; // admin  **private** **static** **final** String ***BROKEURL*** = ActiveMQConnection.***DEFAULT\_BROKER\_URL***;// 默认的连接地址 tcp://localhost:61616  **private** **static** **final** **int** ***NUMS*** = 10; // 消息数量    **public** **static** **void** main(String[] args) {  // 创建连接Factory  ConnectionFactory factory = **null**;  // 获取连接  Connection conn = **null**;  // 创建Session 用于发送和接收消息的线程  Session session = **null**;  // 定义消息的路由 - 消息的目的地  Destination destination = **null**;  // 创建消息生产者  MessageProducer msgProducer = **null**;      // 实例化连接工厂  factory = **new** ActiveMQConnectionFactory(***USERNAME***, ***PASSWORD***, ***BROKEURL***);    **try** {  // 在连接工厂中获取连接  conn = factory.createConnection();  // 打开连接  conn.start();  /\*  \* 获取session - session会话是根据具体的每个连接获取的  \* 参数解释：  \* 参数1 表示是否开启事务，true为开启  \* 参数2 消息的确认方式 Session.AUTO\_ACKNOWLEDGE 表示自动确认  \*  \* 直接receive(接收)消息方式 有三种 分别如下  \* Session.AUTO\_ACKNOWLEDGE [推荐]  \* 当客户端成功的接收方法返回的时候 或是从监听(MessageListener.onMessage)方法成功返回的时候 会话会自动确认客户端收到了消息  \*  \* Session.CLIENT\_ACKNOWLEDGE  \* 客户端通过消息的acknowledge方法确认接收到消息，该模式是在会话层上进行确认的  \* 例如 生产了10条消息 当第五条消息被确认消费后 那么这10条消息都会被确认为已消费  \*  \* Session.DUPS\_OK\_ACKNOWLEDGE  \* 这种模式当出现会话迟钝(一直接收不到消息被消费状态)会出现重复确认消息 如果生产者确认失败 那么可能就会再次发送确认消息  \*  \*/  session = conn.createSession(**true**, Session.***AUTO\_ACKNOWLEDGE***);  // 创建消息发布(topic) - 会返回消息的目的地  destination = session.createTopic("topic\_1");  // 创建消息生产者 - 返回消息生产者实例  msgProducer = session.createProducer(destination);  // 发送消息到队列中  *sendMsg*(session, msgProducer);  // 提交事务 - session在创建的时候设置了事务 因此这里需要提交事务才能真正的发送到队列  session.commit();  } **catch** (JMSException e) {  e.printStackTrace();  }**finally** {  **try** {  **if**(conn!=**null**)  conn.close();  } **catch** (JMSException e) {  ***log***.error("连接关闭异常 ",e);  }  }  }    // 发送消息  **private** **static** **void** sendMsg(Session session,MessageProducer produce) **throws** JMSException {  // 模拟发送消息  **for**(**int** i = 1; i< ***NUMS***; i++) {  // 创建简单的文本消息  TextMessage textMessage = session.createTextMessage("发布的消息信息 - "+i);  System.***out***.println(textMessage.getText());  // 发送消息  produce.send(textMessage);  }  }  } |

## 订阅者

订阅者可以有多个 只需要订阅同一个topic中的消息就可以接收到

多个订阅者代码是完全相同的 这里只添加一份 如果测试 可以复制多份订阅者

|  |
| --- |
| **package** jms.topic;  **import** javax.jms.Connection;  **import** javax.jms.ConnectionFactory;  **import** javax.jms.Destination;  **import** javax.jms.JMSException;  **import** javax.jms.Message;  **import** javax.jms.MessageConsumer;  **import** javax.jms.MessageListener;  **import** javax.jms.Session;  **import** javax.jms.TextMessage;  **import** org.apache.activemq.ActiveMQConnection;  **import** org.apache.activemq.ActiveMQConnectionFactory;  /\*\*  \* 消息订阅者 1  \* **@author** etjav  \*  \*/  **public** **class** Consumer1 {  **private** **static** **final** String ***USERNAME*** = ActiveMQConnection.***DEFAULT\_USER***; // admin  **private** **static** **final** String ***PASSWORD*** = ActiveMQConnection.***DEFAULT\_PASSWORD***; // admin  **private** **static** **final** String ***BROKEURL*** = ActiveMQConnection.***DEFAULT\_BROKER\_URL***;// 默认的连接地址 tcp://localhost:61616    **public** **static** **void** main(String[] args) {  // 创建连接工厂  ConnectionFactory factory = **new** ActiveMQConnectionFactory(***USERNAME***, ***PASSWORD***, ***BROKEURL***);  // 获取连接  Connection conn = **null**;  **try** {  // 获取连接  conn = factory.createConnection();  // 启动连接  conn.start();  // 创建session  Session session = conn.createSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);  // 获取连接的目的地  Destination destination = session.createTopic("topic\_1");  // 创建消费者  MessageConsumer consumer = session.createConsumer(destination);  // 接收消息 - 监听方式  consumer.setMessageListener(**new** Linstener());  } **catch** (JMSException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  }  /\*\*  \* 消息监听  \* **@author** etjav  \*  \*/  **class** Linstener **implements** MessageListener{  @Override  **public** **void** onMessage(Message message) {  **try** {  // 接收到的消息 - 如果有多份 这里改下打印的备注信息即可  System.***out***.println("订阅者1 接收到的消息 :"+((TextMessage)message).getText());  } **catch** (JMSException e) {  e.printStackTrace();  }  }    } |

