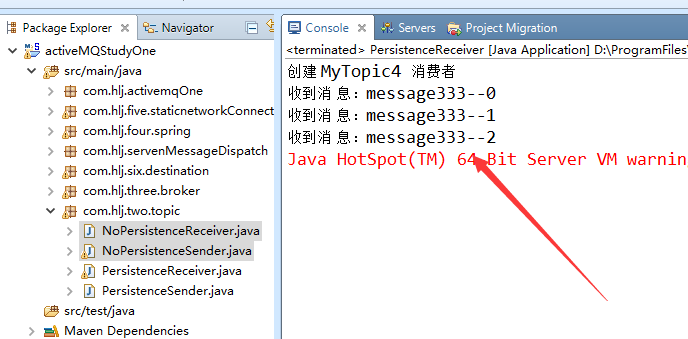
# 1、advisory，顾问，消费者获取生产者的信息

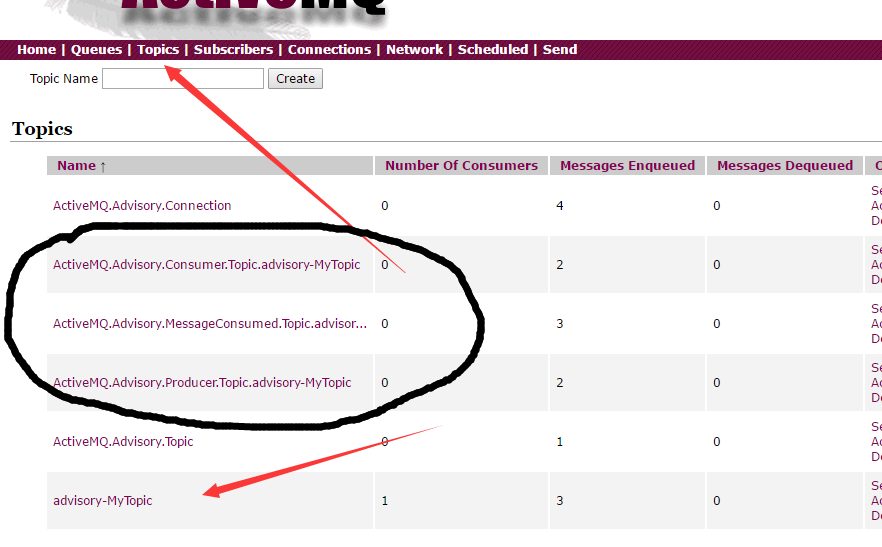
## 1、下面配置advisoryForConsumed为true代表访问topic的时候，advisory的功能是打开的，默认是false

|  |
| --- |
| <destinationPolicy>  <policyMap>  <policyEntries>  <policyEntry topic=">" advisoryForConsumed ="true"> |

## 2、创建生产者和消费者，这里使用的是之前的代码非持久化的topic，进行的测试，先运行消费者，后运行生产者。advisory-MyTopic为destinection



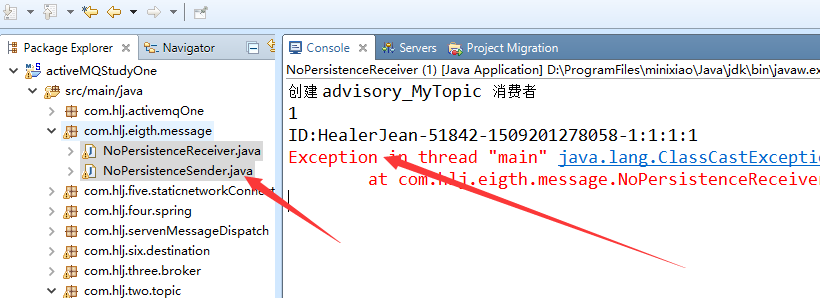
## 3、浏览器中观察，会发现多出来很多系统的topic



## 4、现在我们来获取下系统的topic的一些信息，消费者中获取这个topic的信息，下面获取message的时候直接用TestMessage会报异常，说是消息转化失败。所以不能用TestMessage来获取消息，下面是获取生产者的信息

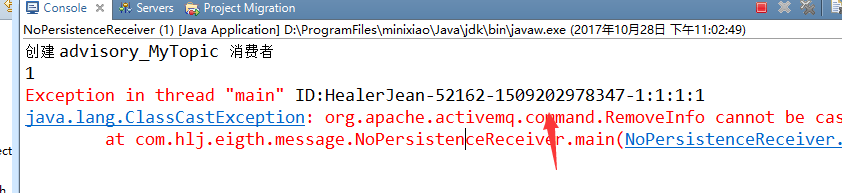
|  |
| --- |
| Destination destination = session.createTopic("ActiveMQ.Advisory.Producer.Topic.advisory\_MyTopic");    MessageConsumer consumer = session.createConsumer(destination);  System.*out*.println("创建 advisory\_MyTopic 消费者");  Message message = consumer.receive();  **while**(message!=**null**) {    **if**(message **instanceof** ActiveMQMessage){  ActiveMQMessage amsg = (ActiveMQMessage)message;  ProducerInfo prod = (ProducerInfo) amsg.getDataStructure();  System.*out*.println(amsg.getProperty("producerCount"));  System.*out*.println(prod.getProducerId());    }  message = consumer.receive();  } |

## 5、控制台打印信息



## 6、第二种方式获取信息，测试运行，和上面是一模一样的

|  |
| --- |
| Topic topic =session.createTopic("advisory\_MyTopic");  Destination destination = AdvisorySupport.*getProducerAdvisoryTopic*(topic); |



# 2、message 时间



## 1、xml开启时间属性

|  |
| --- |
| <broker xmlns="http://activemq.apache.org/schema/core"  brokerName="localhost"  dataDirectory="${activemq.data}"  schedulePeriodForDestinationPurge="2000"  schedulerSupport="true"  > |

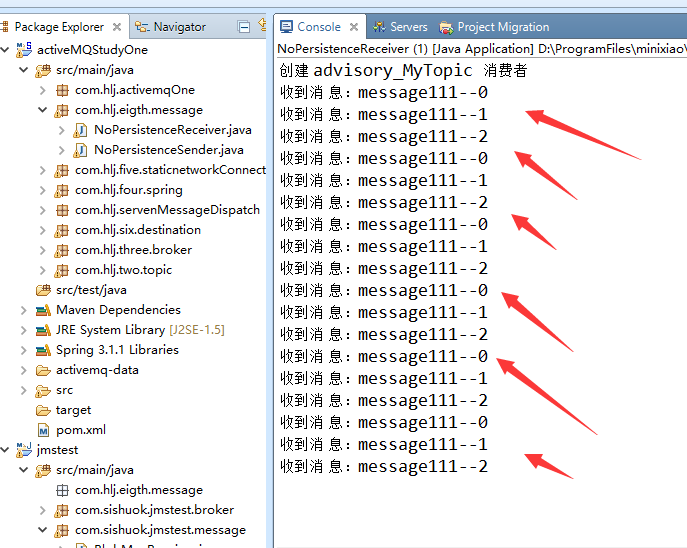
## 2、生产者设置 延迟3秒，周期3秒，重复5次

|  |
| --- |
| MessageProducer producer = session.createProducer(destination);  System.*out*.println("创建 advisory\_MyTopic 生产者");  **for** (**int** i = 0; i < 3; i++) {    TextMessage message = session.createTextMessage("message111--" + i);    **long** delay = 3 \* 1000; //延迟3秒  **long** period = 3 \* 1000; //周期3秒  **int** repeat = 5; //重复5次  message.setLongProperty(ScheduledMessage.*AMQ\_SCHEDULED\_DELAY*, delay);  message.setLongProperty(ScheduledMessage.*AMQ\_SCHEDULED\_PERIOD*, period);  message.setIntProperty(ScheduledMessage.*AMQ\_SCHEDULED\_REPEAT*, repeat);    // 通过消息生产者发出消息  producer.send(message);  } |

## 3、消费者接收

|  |
| --- |
| **while**(message!=**null**) {  TextMessage txtMsg = (TextMessage)message;  System.*out*.println("收到消 息：" + txtMsg.getText());  message = consumer.receive();  } |

## 4、测试成功，会发现出现了6次，其中有5次是重复的。



# 以上文件



# 3、Blog Message（文件）

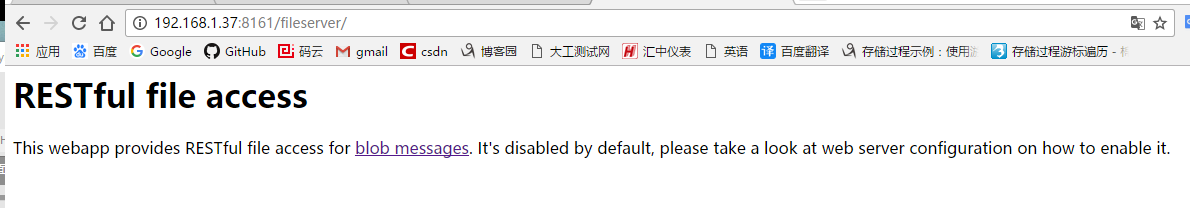
## 2、建立一个发送文件的生产者

|  |
| --- |
| **public** **class** BlobMsgSend {  **public** **static** **void** main(String[] args)**throws** Exception {    String linuxIp = "myLinuxQj";  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616?"  + "jms"  + ".blobTransferPolicy"  + ".uploadUrl=http://"+linuxIp+":8161/fileserver/"  );    Connection connection = connectionFactory.createConnection();  connection.start();  ActiveMQSession session =(ActiveMQSession) connection.createSession(Boolean.*TRUE*,  Session.*CLIENT\_ACKNOWLEDGE*);    Destination destination = session.createQueue("my-queue");      MessageProducer producer = session.createProducer(destination);    BlobMessage bm = session.createBlobMessage(**new** File("pom.xml"));    producer.send(bm);    session.commit();  session.close();  connection.close();  }  } |

## 3、创建消费者接收文件

|  |
| --- |
| **public** **class** BlobMsgReceiver {  **public** **static** **void** main(String[] args) **throws** Exception{      String linuxIp = "myLinuxQj";  ConnectionFactory cf = **new** ActiveMQConnectionFactory(  "tcp://"+linuxIp+":61616");    Connection connection = cf.createConnection();  connection.start();    **final** Session session = connection.createSession(Boolean.*FALSE*,  Session.*CLIENT\_ACKNOWLEDGE*);  Destination destination = session.createQueue("my\_queue\_blobFile");  MessageConsumer consumer = session.createConsumer(destination);    consumer.setMessageListener(**new** MessageListener() {    **public** **void** onMessage(Message msg) {  **if**(msg **instanceof** BlobMessage){  BlobMessage message = (BlobMessage) msg;  **try** {  InputStream in = message.getInputStream();  **byte** bs[] = **new** **byte**[in.available()];    in.read(bs);  in.close();    System.*out*.println("content===="+**new** String(bs));    } **catch** (Exception e) {  e.printStackTrace();  }    }  }  });  }  } |

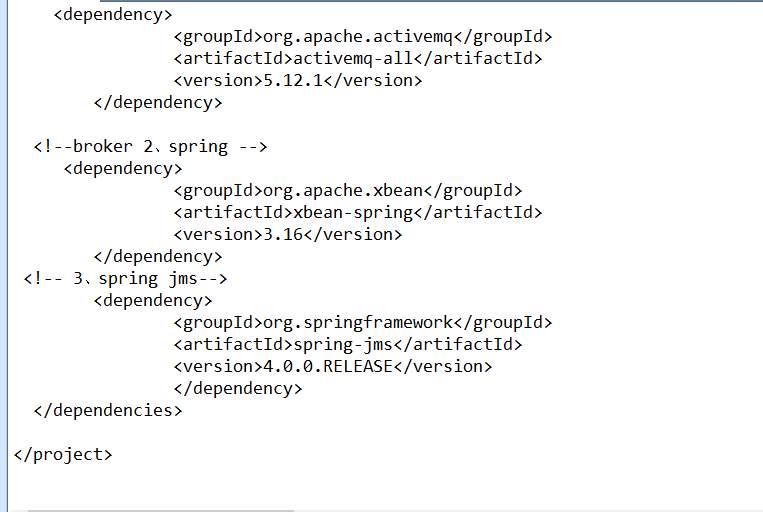
## 3、测试运行，失败了，访问<http://192.168.1.37:8161/fileserver/>因为没有开启对这个的支持，



## 4、打开jetty开启支持，将下面的注释打开即可

|  |
| --- |
| <!-- Enable embedded file server for Blob messages -->  <!-- -->  <bean class="org.eclipse.jetty.webapp.WebAppContext">  <property name="contextPath" value="/fileserver" />  <property name="resourceBase" value="${activemq.home}/webapps/fileserver" />  <property name="logUrlOnStart" value="true" />  <property name="parentLoaderPriority" value="true" />  </bean> |

## 4、测试成功



# 4、MessageTransformer

## 1、生产者，直接会将上面的TextMessage读取成为msg

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

## 2、测试，消费者

|  |
| --- |
| **int** i = 0;  **while** (i < 3) {  MapMessage message = (MapMessage) consumer.receive();    System.*out*.println("收到消 息：" + message.getString("messageAA--" + i)  +" , property=="+message.getStringProperty("extra"));    i++;  } |

## 3、控制台，成功

