

# ActiveMQ学习笔记

## 一. 初步安装使用

ActiveMQ 的官网：<http://activemq.apache.org>

MQ：消息中间件/消息队列

为什么要使用 MQ？

解决了耦合调用、异步模型、抵御洪峰流量，保护了主业务，消峰。

直接进入myactivemq 的文件下的activemq 下的 bin 目录，使用 `./activemq start` 命令启动

检查activemq 是否启动的三种方法：也是三种查看后台进程的方法

`ps -ef|grep activemq|grep -v grep` // `grep -v grep` 可以不让显示grep 本来的信息

`netstat -anp|grep 61616` // activemq 的默认后台端口是61616

`lsof -i:61616`

让启动的日志信息不在控制台打印，而放到专门的文件中：

`./activemq start > /myactivemq/myrunmq.log`

g

## 二、部署和代码尝试

部署在linux 上的activeMQ 要可以通过前台windows 的页面访问，必须把linux 的IP和 windows的IP 地址配置到同一个网关下。这种情况一般都是修改 linux 的IP 地址，修改网卡文件对应的IP 地址

修改linux 的ip 地址：`cd /etc/sysconfig/network-scripts`

`vi ifcfg-eth0`

这是修改之后的网卡文件配置，IP 地址为：192.168.17.3 （因为我的windows 的IP 地址为192.168.17.1，将他们配置在了同一个网关下）

配置成功后，可以用 windows ping linux，linux ping windows，当全部ping通后，可以使用图形化界面访问activeMQ

// ActiveMQ 的前台端口为 8161，提供控制台服务 后台端口为61616，提供 JMS 服务

// 192.168.17.3 为 linux 的IP 地址，使用 IP+端口 访问了ActiveMQ，登陆之后的样子如上。（能访问成功首先得在linux 上启动activeMQ 的服务），首次登录的默认账户密码为 账号: admin 密码: admin

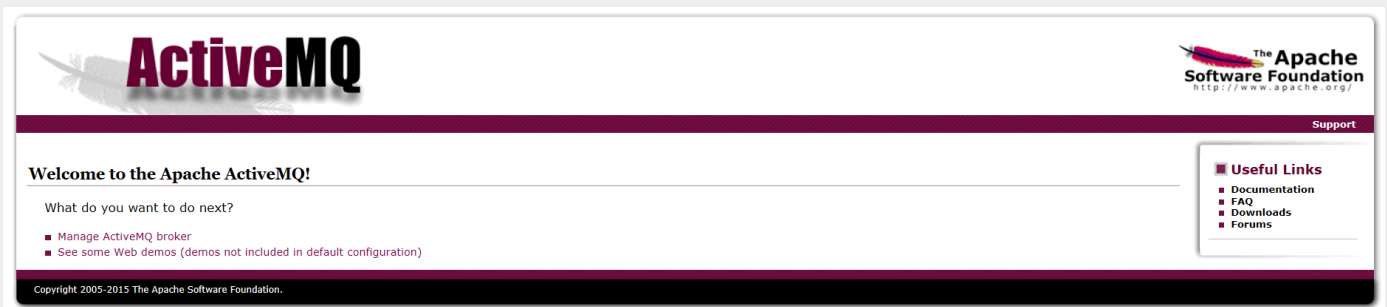
访问不到的坑： 1 可能是你的linux 和 windows 没有在一个网关下

2 可能你windows 的防火墙或者 linux 的防火墙没有关掉（是的，先得关掉防火墙）

3 你忘记启动activemq 的服务了

4 你启动失败了，可能是你得java 环境没配好，必须是jdk 8 或者

以上



```
package cn.scct.helloworld;
```

```
import org.apache.activemq.ActiveMQConnectionFactory;
```

```
import javax.jms.*;
```

```
public class JmsProduce {
```

```
    // linux 上部署的activemq 的 IP 地址 + activemq 的端口号，如果用自己的需要改动
```

```
    public static final String ACTIVEMQ_URL =  
"tcp://192.168.150.128:61616";
```

```
    // public static final String ACTIVEMQ_URL = "nio://192.168.17.3:61608";
```

```
    public static final String QUEUE_NAME = "myqueue";
```

```

public static void main(String[] args) throws Exception{

    // 1 按照给定的url创建连接工程，这个构造器采用默认的用户名密码
    ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQC
onnectionFactory(ACTIVEMQ_URL);

    // 设置允许有数据丢失
    // activeMQConnectionFactory.setUseAsyncSend(true);

    // 2 通过连接工厂连接 connection 和 启动
    Connection connection = activeMQConnectionFactory.createConnection
();
    // 启动
    connection.start();
    // 3 创建回话 session
    // 两个参数，第一个事务， 第二个签收
    Session session = connection.createSession(false,Session.AUTO_ACKNO
WLEDGE);
    // 4 创建目的地 （两种： 队列/主题 这里用队列）
    Queue queue = session.createQueue(QUEUE_NAME);
    // 5 创建消息的生产者
    MessageProducer messageProducer = session.createProducer(queue);
    // 非持久化消息 和持久化消息演示
    messageProducer.setDeliveryMode(DeliveryMode.PERSISTENT); // 持久
化 如果开启
                                                                    // 就会存入文
件或数据库中

    // 6 通过messageProducer 生产 6 条 消息发送到消息队列中
    for (int i = 1; i < 7; i++) {
        // 7 创建字消息
        TextMessage textMessage = session.createTextMessage("msg--" +
i);
        // 8 通过messageProducer发布消息
        messageProducer.send(textMessage);
    }
    // 9 关闭资源
    messageProducer.close();
    session.close();
    connection.close();
    // session.commit();
    System.out.println(" **** send MQ finished ****");
}

```

```
}  
}
```

```
package cn.scct.helloworld;  
  
import org.apache.activemq.ActiveMQConnectionFactory;  
  
import javax.jms.*;  
  
// 消息的消费者 也就是回答消息的系统  
public class JmsConsumer {  
    public static final String ACTIVEMQ_URL =  
"tcp://192.168.150.128:61616";  
  
    public static final String QUEUE_NAME = "myqueue";  
  
    public static void main(String[] args) throws Exception{  
        System.out.println(" 这里是 2号 消费者 ");  
  
        // 1 按照给定的url创建连接工程，这个构造器采用默认的用户名密码  
        ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQC  
onnectionFactory(ACTIVEMQ_URL);  
        // 2 通过连接工厂连接 connection 和 启动  
        javax.jms.Connection connection = activeMQConnectionFactory.createC  
onnection();  
        // 启动  
        connection.start();  
        // 3 创建回话 session  
        // 两个参数，第一个事务， 第二个签收  
        Session session = connection.createSession(false, Session.AUTO_ACKN  
OWLEDGE);  
        // 4 创建目的地 （两种： 队列/主题 这里用队列）  
        Queue queue = session.createQueue(QUEUE_NAME);  
        // 5 创建消息的消费者  
        MessageConsumer messageConsumer = session.createConsumer(queue);  
  
        //  
        // 同步阻塞方式receive() 空参数的receive方法是阻塞，有参数的为等待时  
        间
```

```
// 订阅者或消费者使用MessageConsumer 的receive() 方法接收消息，receive 在接收之前一直阻塞
/* while(true){
    // 这里是 TextMessage 是因为消息发送者是 TextMessage ， 接受处理的
    // 也应该是这个消息
    TextMessage message = (TextMessage)messageConsumer.receive(4000L); // 4秒
    if (null != message){
        System.out.println("****consumer: "+message.getText());
    }else {
        break;
    }
}*/
```

// 通过监听的方式来消费消息  
 // 通过异步非阻塞的方式消费消息  
 // 通过messageConsumer 的setMessageListener 注册一个监听器，  
 // 当有消息发送来时，系统自动调用MessageListener 的 onMessage 方法处理消息

```
messageConsumer.setMessageListener(new MessageListener(){

    @Override
    public void onMessage(Message message) {
        if (null != message && message instanceof TextMessage) {
            TextMessage textMessage = (TextMessage) message;
            try {
                System.out.println("****消费者的消息: " + textMessage.getText());
            } catch (JMSEException e) {
                e.printStackTrace();
            }
        }
    }
});
```

/\*

1.先生产 只启动1号消费者 消费者能消费消息吗

Y

2.先生产 先启动1号消费者，再启动2号消费者，问题：2号消费者还能消费消息

吗？

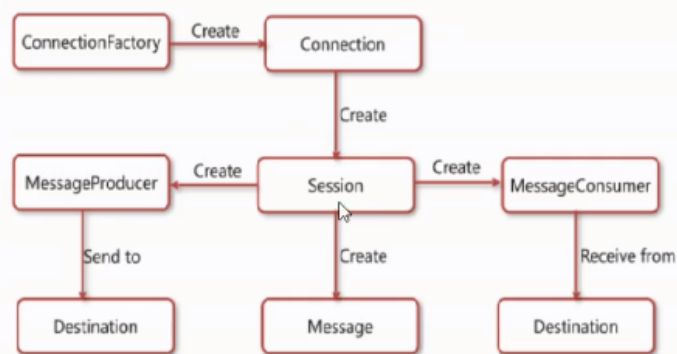
N

3.先启动2个消费者，再生产六条消息，请问消费情况如何？

\*/

```
// 保证控制台不灭 不然activemq 还没连上就关掉了连接
System.in.read();
messageConsumer.close();
session.close();
connection.close();
}
}
```

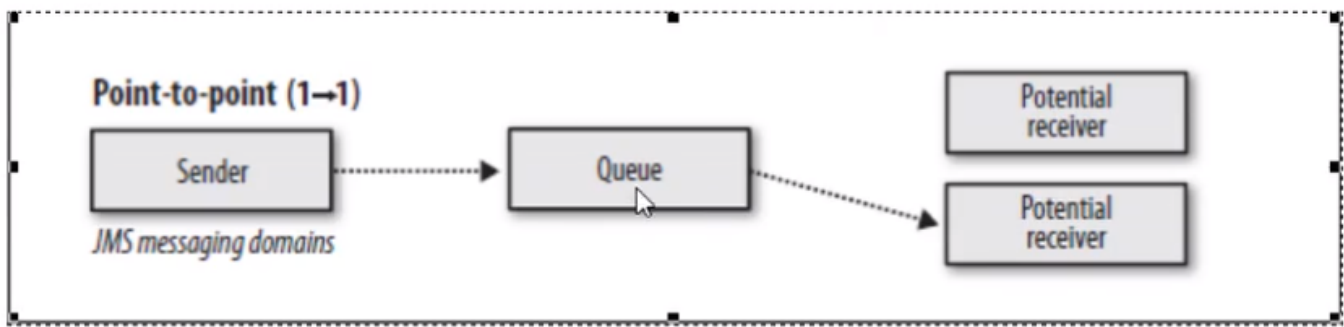
#### JMS开发的基本步骤



#### JMS开发的基本步骤

- 1: 创建一个connection factory
- 2: 通过connection factory来创建JMS connection
- 3: 启动JMS connection
- 4: 通过connection创建JMS session
- 5: 创建JMS destination
- 6: 创建JMS producer或者创建JMS message并设置destination
- 7: 创建JMS consumer或者是注册一个JMS message listener
- 8: 发送或者接受JMS message(s)
- 9: 关闭所有的JMS资源  
(connection, session, producer, consumer等)

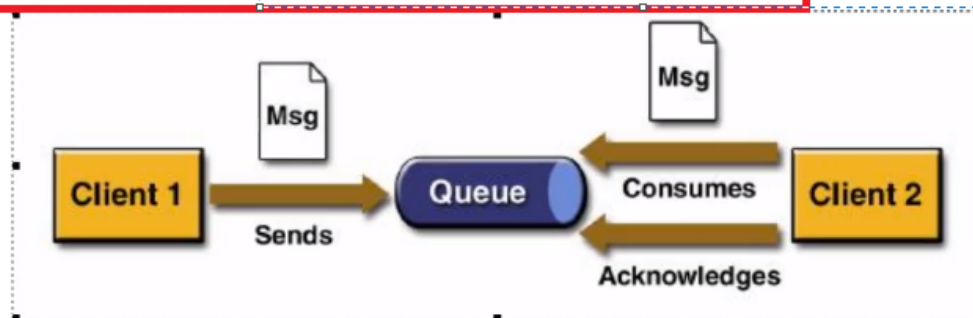
### 三、消息队列总结



在点对点的消息传递域中，目的地被称为队列（Queue）

点对点消息传递域的特点如下：

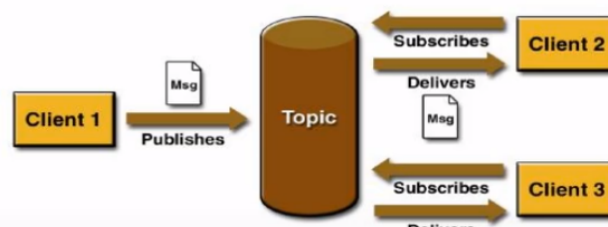
- （1）每个消息只能有一个消费者，类似1对1的关系。好比个人快递自己领取自己的。
- （2）消息的生产者和消费者之间没有时间上的相关性。无论消费者在生产者发送消息的时候是否处于运行状态，消费者都可以提取消息。好比我们的发送短信，发送者发送后不见得接收者会即收即看。
- （3）消息被消费后队列中不会再存储，所以消费者不会消费到已经被消费掉的消息。



## 四、订阅

- （1）生产者将消息发布到topic中，每个消息可以有多个消费者，属于1: N的关系
- （2）生产者和消费者之间有时间上的相关性。订阅某一个主题的消费者只能消费自它订阅之后发布的消息。
- （3）生产者生产时，topic不保存消息它是无状态的不落地，假如无人订阅就去生产，那就是一条废消息，所以，一般先启动消费者再启动生产者。

JMS 规范允许客户创建持久订阅，这在一定程度上放松了时间上的相关性要求。持久订阅允许消费者消费它在未处于激活状态时发送的消息。一句话，好比我们的微信公众号订阅



topic01

2

3

6

Send To Active Subscribers  
Active Producers  
Delete

```
package cn.scct.topic;
```

```
import org.apache.activemq.ActiveMQConnectionFactory;
```

```

import javax.jms.*;

public class JmsProduceTopic {
    // linux 上部署的activemq 的 IP 地址 + activemq 的端口号, 如果用自己的需要改动
    public static final String ACTIVEMQ_URL =
"tcp://192.168.150.128:61616";
    // public static final String ACTIVEMQ_URL = "nio://192.168.17.3:61608";
    public static final String TOPIC_NAME = "topic01";

    public static void main(String[] args) throws Exception{

        // 1 按照给定的url创建连接工程, 这个构造器采用默认的用户名密码
        ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQC
onnectionFactory(ACTIVEMQ_URL);
        // 2 通过连接工厂连接 connection 和 启动
        Connection connection = activeMQConnectionFactory.createConnection
();
        // 启动
        connection.start();
        // 3 创建回话 session
        // 两个参数, 第一个事务, 第二个签收
        Session session = connection.createSession(false, Session.AUTO_ACKN
OWLEDGE);
        // 4 创建目的地 (两种 : 队列/主题 这里用主题)
        Topic topic = session.createTopic(TOPIC_NAME);
        // 5 创建消息的生产者
        MessageProducer messageProducer = session.createProducer(topic);

        for (int i = 1; i < 4 ; i++) {
            // 7 创建字消息
            TextMessage textMessage = session.createTextMessage("topic_name
--" + i);
            // 8 通过messageProducer发布消息
            messageProducer.send(textMessage);

            MapMessage mapMessage = session.createMapMessage();
            //      mapMessage.setString("k1","v1");
            //      messageProducer.send(mapMessage);
        }
    }
}

```



```

        // 9 关闭资源
        messageProducer.close();
        session.close();
        connection.close();

        System.out.println(" topic01 message send to MQ finished");
    }
}

```

```

package cn.scct.topic;

import org.apache.activemq.ActiveMQConnectionFactory;

import javax.jms.*;

// 消息的消费者 也就是回答消息的系统
public class JmsConsumerTopic {
    public static final String ACTIVEMQ_URL =
"tcp://192.168.150.128:61616";

    public static final String TOPIC_NAME = "topic01";

    public static void main(String[] args) throws Exception{
        System.out.println(" NO.2 consumer ");

        // 1 按照给定的url创建连接工程，这个构造器采用默认的用户名密码
        ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQC
onnectionFactory(ACTIVEMQ_URL);
        // 2 通过连接工厂连接 connection 和 启动
        javax.jms.Connection connection = activeMQConnectionFactory.createC
onnection();
        // 启动
        connection.start();
        // 3 创建回话 session
        // 两个参数，第一个事务， 第二个签收
        Session session = connection.createSession(false, Session.AUTO_ACKN
OWLEDGE);
        // 4 创建目的地 （两种 ： 队列/主题 这里用主题）
        Topic topic = session.createTopic(TOPIC_NAME);
        // 5 创建消息的消费者

```

```

MessageConsumer messageConsumer = session.createConsumer(topic);

messageConsumer.setMessageListener( (message) -> {
    if (null != message && message instanceof TextMessage){
        TextMessage textMessage = (TextMessage)message;
        try {
            System.out.println("consumer recieve the topic from M
Q:"+textMessage.getText());
        }catch (JMSEException e) {
        }
    }
    if (null != message && message instanceof MapMessage){
        MapMessage mapMessage = (MapMessage)message;
        try {
            System.out.println("consumer recieve the topic of map t
ype from MQ :"+mapMessage.getString("k1"));
        }catch (JMSEException e) {
        }
    }
});

// 保证控制台不灭 不然activemq 还没连上就关掉了连接
System.in.read();
messageConsumer.close();
session.close();
connection.close();
}
}

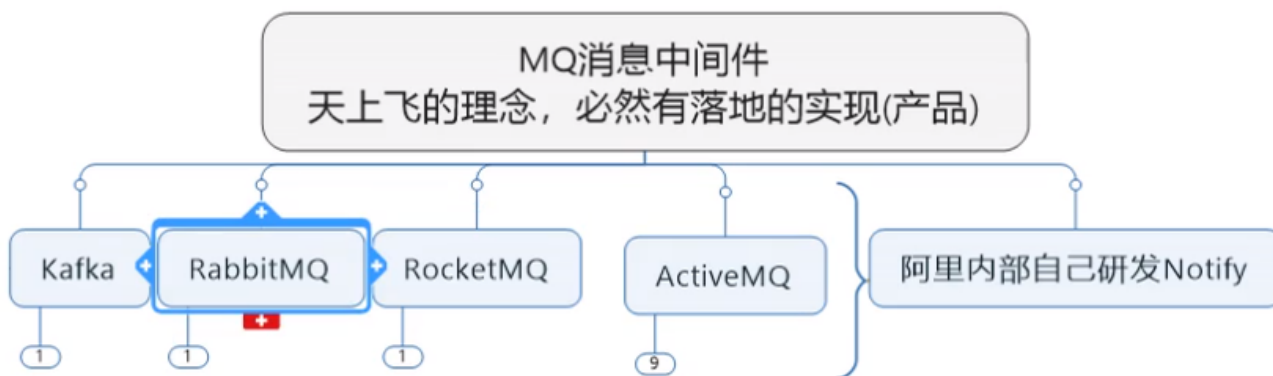
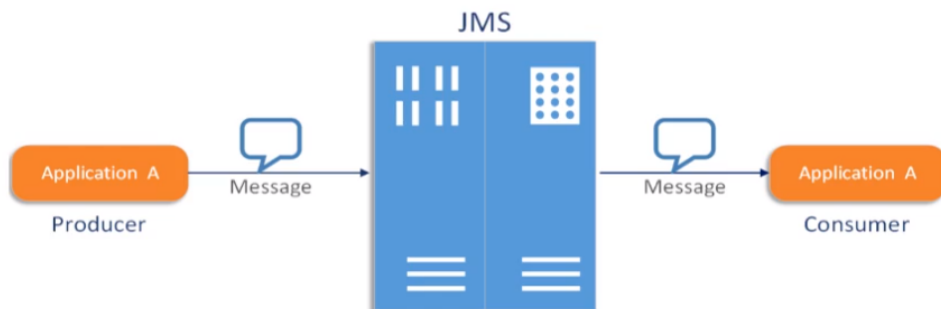
```

比较项目	Topic 模式队列	Queue 模式队列
工作模式	只有先订阅才能收到消息，而且收不到未订阅前的消息 “订阅-发布”模式，如果当前没有订阅者，消息将会被丢弃。如果有多个订阅者，那么这些订阅者都会收到消息	“负载均衡”模式。如果当前没有消费者，消息也不会丢弃；如果有多个消费者，那么一条消息也只会发送给其中一个消费者，并且要求消费者ack信息。
有无状态	无状态	Queue数据默认会在mq服务器上以文件形式保存，比如Active MQ一般保存在\$AMQ_HOME\data\kr-store\data下面，也可以配置成DB存储。
传递完整性	如果没有订阅者，消息会被丢弃	消息不会丢弃
处理效率	由于消息要按照订阅者的数量进行复制，所以处理性能会随着订阅者的增加而明显降低，并且还要结合不同消息协议自身的性能差异	由于一条消息只发送一个消费者，所以就算消费者再多，性能也不会有明显降低。当然不同消息协议的具体性能也是有差异的

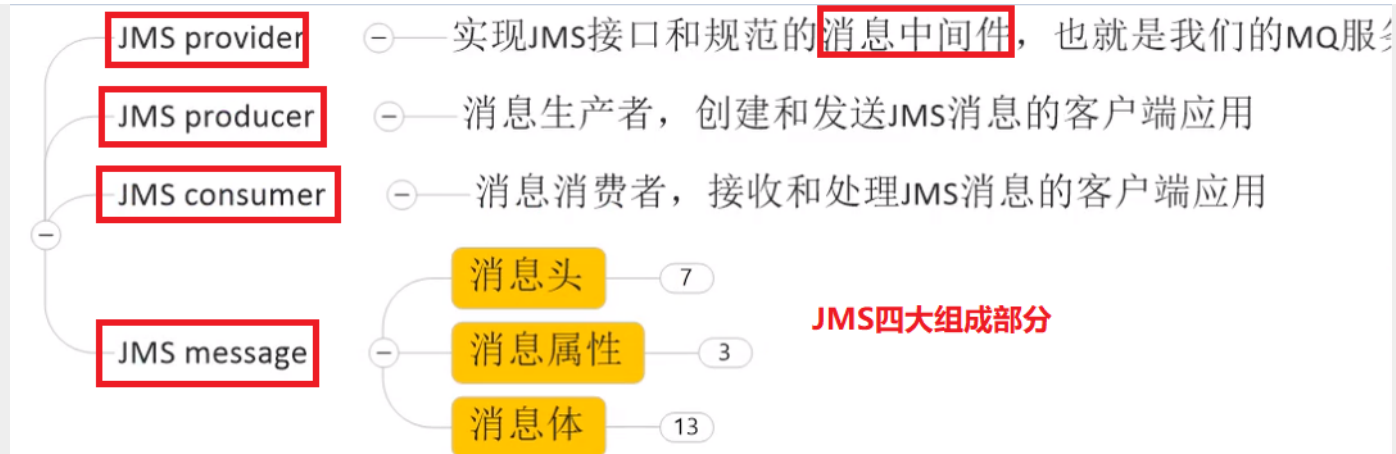
五、JMS（Java消息服务）

## 什么是Java消息服务

Java消息服务指的是两个应用程序之间进行异步通信的API。它为标准消息协议和消息服务提供了一组通用接口，包括创建、发送、读取消息等，用于支持JAVA应用程序开发。在JavaEE中，当两个应用程序使用JMS进行通信时，它们之间并不是直接相连的，而是通过一个共同的消息收发服务组件关联起来以达到解耦/异步削峰的效果。



## 5.1 JMS四大组成部分



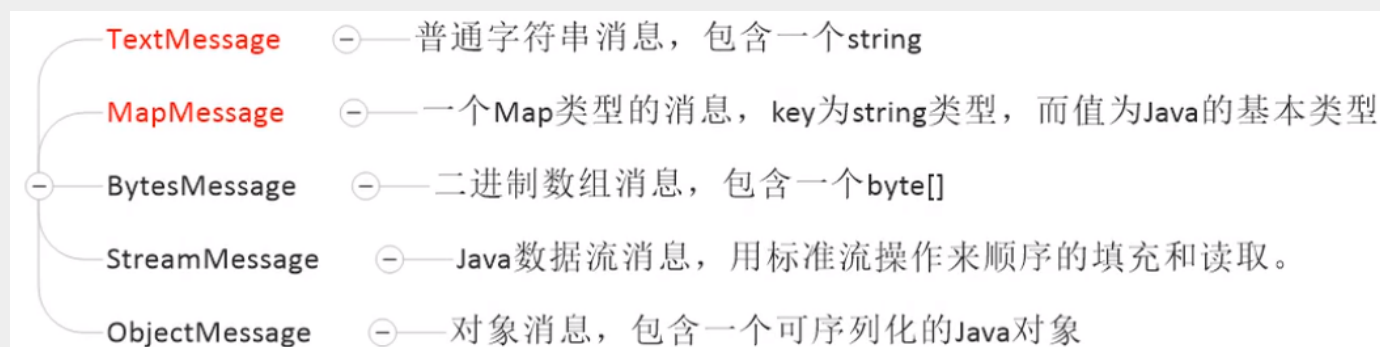
## 5.2 JMS消息

### 5.2.1 消息头

5 个主要的消息头


消息头	JMSDestination	JMSDeliveryMode	JMSExpiration	JMSPriority	JMSMessageId
含义	头在哪儿	是持久还是非持久	过期时间，默认永久	优先级，默认是4 有0~9，5-9 是紧急的，0-4 是普通的	唯一的消息ID

### 5.2.2 消息体



### 5.2.3 消息属性

## 消息属性

如果需要除消息头字段以外的值，那么可以使用消息属性识别/去重/重点标注等操作非常有用的方法是什么 

他们是以属性名和属性值对的形式制定的。可以将属性是为消息头得扩展，属性指定一些消息头没有包括的附加信息，比如可以在属性里指定消息选择器。

消息的属性就像可以分配给一条消息的附加消息头一样。它们允许开发者添加有关消息的不透明附加信息。它们还用于暴露消息选择器在消息过滤时使用的数据。

```
TextMessage message = session.createTextMessage();
message.setText(text);
message.setStringProperty("username","z3"); //自定义属性
```

## 5.3 如何保证消息的可靠性

### 5.3.1 持久化

```
// 在队列为目的地的时候持久化消息
messageProducer.setDeliveryMode(DeliveryMode.PERSISTENT);

// 队列为目的地的非持久化消息
messageProducer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
```

持久化的消息，服务器宕机后消息依旧存在，**只是没有入队**，当服务器再次启动，消息仍旧会被消费。

但是**非持久化的消息**，服务器宕机后消息永远丢失。

而当你没有注明是否是持久化还是非持久化时，**队列默认是持久化的消息**。

对于目的地为主题（topic）来说，**默认就是非持久化的**，让主题的订阅持久化的意义在于：**对于订阅了公众号的人来说，当用户手机关机，在开机后仍然可以接受到关注公众号之前发送的消息。**

## 持久化主题生产者

```
// 1 按照给定的url创建连接工程，这个构造器采用默认的用户名密码
ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQConnectionFactory(ACTIVEMQ_URL);
// 2 通过连接工厂连接 connection 和 启动
Connection connection = activeMQConnectionFactory.createConnection();

// 3 创建会话 session
// 两个参数，第一个事务， 第二个签收
Session session = connection.createSession(b: false, Session.AUTO_ACKNOWLEDGE);

Topic topic = session.createTopic(TOPIC_NAME);

// 5 创建消息的生产者
MessageProducer messageProducer = session.createProducer(topic);
// 6 通过messageProducer 生产 3 条 消息发送到消息队列中

// 设置持久化topic 再启动
messageProducer.setDeliveryMode(DeliveryMode.PERSISTENT);
//
connection.start();

for (int i = 1; i < 4 ; i++) {
    // 7 创建字消息
    TextMessage textMessage = session.createTextMessage(s: "topic_name--" + i);
    // 8 通过messageProducer发布消息
    messageProducer.send(textMessage);

    MapMessage mapMessage = session.createMapMessage();
    // mapMessage.setString("k1", "v1");
    // messageProducer.send(mapMessage);
}

// 9 关闭资源
messageProducer.close();
```

以前可以在3之前启动

## 主题消费者

```
// 1 按照给定的url创建连接工程，这个构造器采用默认的用户名密码
ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQConnectionFactory(ACTIVEMQ_URL);
// 2 通过连接工厂连接 connection
Connection connection = activeMQConnectionFactory.createConnection();
connection.setClientID("marrry");

// 3 创建会话 session
// 两个参数，第一个事务， 第二个签收
Session session = connection.createSession(b: false, Session.AUTO_ACKNOWLEDGE);
// 4 创建目的地（两种： 队列/主题 这里用主题）
Topic topic = session.createTopic(TOPIC_NAME);
TopicSubscriber topicSubscriber = session.createDurableSubscriber(topic, s: "remark...");

// 5 接收订阅
connection.start();

Message message = topicSubscriber.receive(); // 一直等
while (null != message){
    TextMessage textMessage = (TextMessage)message;
    System.out.println(" 收到的持久化 topic : "+textMessage.getText());
    message = topicSubscriber.receive(1: 3000L);
}

session.close();
connection.close();
```

注意recieve（时间）方法可以设置在线时间

```
/*1 一定要先运行一次消费者，等于向MQ注册，类似我订阅了这个主题。
2 然后再运行生产者发送信息，此时，
3 无论消费者是否在线，都会接收到，不在线的话，下次连接的时候，会把没有收过的消息都接收下来。*/
```

## 5.3.2 事务



createSession的第一个参数为true 为开启事务，开启事务之后必须在将消息提交，才可以在队列中看到消息

```
Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);
```

提交：

```
session.commit();
```

事务开启的意义在于，如果对于多条必须同批次传输的消息，可以使用事务，如果一条传输失败，可以将事务回滚，再次传输，保证数据的完整性。

对于消息消费者来说，开启事务的话，可以避免消息被多次消费，以及后台和服务器数据的不一致性。

举个栗子：

如果消息消费的 createSession 设置为 true，但是没有 commit，此时就会造成非常严重的后果，那就是在后台看来消息已经被消费，但是对于服务器来说并没有接收到消息被消费，此时就有可能被多次消费。

### 5.3.3 签收

非事务：

Session.AUTO_ACKNOWLEDGE	自动签收，默认
--------------------------	---------

Session.CLIENT_ACKNOWLEDGE	手动签收
----------------------------	------

手动签收需要acknowledge

```
textMessage.acknowledge();
```

而对于开启事务时，设置手动签收和自动签收没有多大的意义，都默认自动签收，也就是说事务的优先级更高一些。

```
Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);  
//Session session = connection.createSession(true, Session.CLIENT_ACKNOWLEDGE);  
// 也是自动签收
```

```
.....
```

```
session.commit();
```

但是开启事务没有commit 任就会重复消费



## 5.4 broker (了解)

小知识: broker

broker 就是实现了用代码形式启动 ActiveMQ 将 MQ 内嵌到 Java 代码中, 可以随时启动, 节省资源, 提高了可靠性。

就是将 MQ 服务器作为了 Java 对象

使用多个配置文件启动 activemq

```
cp activemq.xml activemq02.xml
```

// 以active02 启动mq 服务器

```
./activemq start xbean:file:/myactivemq/apache-activemq-5.15.9/conf/activemq02.xml
```

把小型 activemq 服务器嵌入到 java 代码: 不再使用linux 的服务器

需要的包:

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.9.5</version>
</dependency>
```

代码实现:

```
public class Embebroker {
    public static void main(String[] args) throws Exception {
        // broker 服务
        BrokerService brokerService = new BrokerService();
        // 把小型 activemq 服务器嵌入到 java 代码
        brokerService.setUseJmx(true);
        // 原本的是 192..... 是linux 上的服务器, 而这里是本地windows 的小型mq 服务器
        brokerService.addConnector("tcp://localhost:61616");
        brokerService.start();
    }
}
```