

# ActiveMQ{消息队列}入门学习练习

## 1. ActiveMQ的安装

1. 从官网下载安装包, <http://activemq.apache.org/download.html>
2. 赋予运行权限 `chmod +x windows`可以忽略此步
3. 运行 `./active start | stop`

启动后, activeMQ会占用两个端口, 一个是负责接收发送消息的tcp端口:61616, 一个是基于web负责用户界面化管理的端口:8161。这两个端口可以在conf下面的xml中找到。http服务器使用了jetty。这里有个问题是启动mq后, 很长时间管理界面才可以显示出来。

### 一: JMS基本概念

#### 1. JMS的目标

为企业级的应用提供一种智能的消息系统, JMS定义了一整套的企业级的消息概念与工具, 尽可能最小化的Java语言概念去构建最大化企业消息应用。统一已经存在的企业级消息系统功能。

#### 2. JMS提供者

JMS提供者是指那些完全完成JMS功能与管理功能的JMS消息厂商, 理论上JMS提供者完成JMS消息产品必须是100%的纯Java语言实现, 可以运行在跨平台的架构与操作系统上, 当前一些JMS厂商包括IBM, Oracle, JBoss社区 (JBoss Community), Apache 社区(ApacheCommunity)。

#### 3. JMS应用程序, 一个完整的JMS应用应该实现以下功能:

JMS 客户端 – Java语言开发的接受与发送消息的程序

非JMS客户端 – 基于消息系统的本地API实现而不是JMS

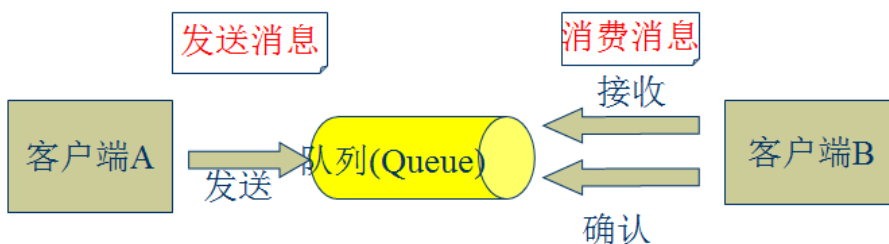
消息 – 应用程序用来相互交流信息的载体

被管理对象–预先配置的JMS对象, JMS管理员创建, 被客户端运用。如链接工厂, 主题等

JMS提供者–完成JMS功能与管理功能的消息系统

### 二: JMS的消息模式

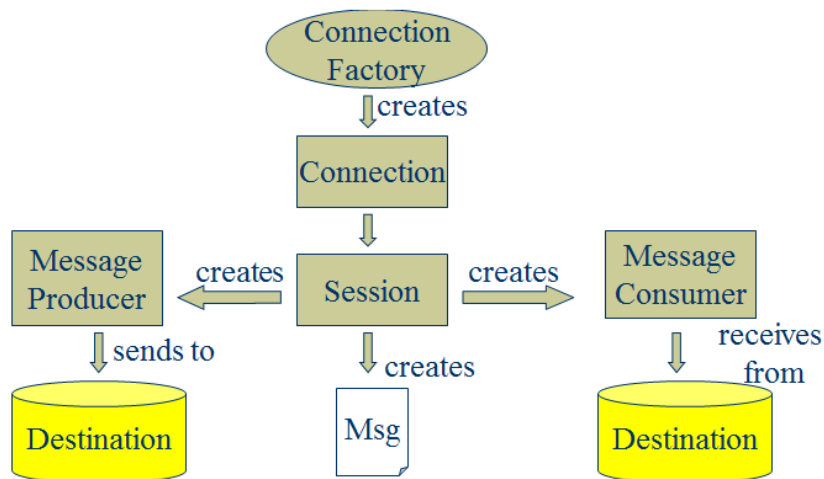
#### 1. 点对点的消息模式(Point to Point Messaging)



下面的JMS对象在点对点消息模式中是必须的:

- a. 队列(Queue) – 一个提供者命名的队列对象, 客户端将会使用这个命名的队列对象
- b. 队列链接工厂(QueueConnectionFactory) – 客户端使用队列链接工厂创建链接队列 ConnectionQueue来取得与JMS点对点消息提供者的链接。
- c. 链接队列(ConnectionQueue) – 一个活动的链接队列存在在客户端与点对点消息提供者之间, 客户用它创建一个或者多个JMS队列会话(QueueSession)
- d. 队列会话(QueueSession) – 用来创建队列消息的发送者与接受者(QueueSender and QueueReceiver)
- e. 消息发送者(QueueSender 或者 MessageProducer) – 发送消息到已经声明的队列
- f. 消息接受者(QueueReceiver 或者 MessageConsumer) – 接受已经被发送到指定队列的消息

## 2. 发布订阅模式(publish – subscribe Mode)



- 主题Topic(Destination) – 一个提供者命名的主题对象，客户端将会使用这个命名的主题对象
- 主题链接工厂(TopicConnectionFactory) – 客户端使用主题链接工厂创建链接主题 ConnectionTopic来取得与JMS消息Pub/Sub提供者的链接。
- 链接主题(ConnectionTopic) – 一个活动的链接主题存在发布者与订阅者之间
- 会话(TopicSession) – 用来创建主题消息的发布者与订阅者 (TopicPublisher and TopicSubscribers)
- 消息发送者MessageProducer) – 发送消息到已经声明的主题
- 消息接受者(MessageConsumer) – 接受已经被发送到指定主题的消息

### 介绍 ActiveMQ

ActiveMQ是apache社区完成的JMS开源消息组件，客户端支持多种语言调用，包括Java,C++, C#, Perl, **Python**等。支持**spring**配置集成等

#### 代码示例

##### 先运行ActiveMQ

解压缩apache-activemq-5.8.0-bin.zip，然后双击apache-activemq-5.5.1\bin\activemq.bat运行ActiveMQ程序。

启动ActiveMQ以后，登陆：http://localhost:8161/admin/，创建一个Queue，命名为FirstQueue。

##### 创建idea项目并运行

创建java project: ActiveMQ-Sduty

打开apache-activemq-5.8.0\lib目录

拷贝

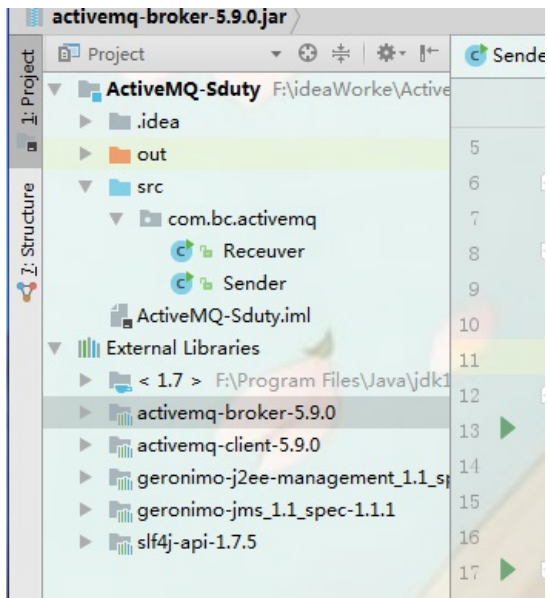
activemq-broker-5.9.0.jar

activemq-client-5.9.0.jar

geronimo-j2ee-management\_1.1\_spec-1.0.1.jar

geronimo-jms\_1.1\_spec-1.1.1.jar

slf4j-api-1.7.5.jar



Sender.java

---

```
package com.bc.activemq;
```

```
import org.apache.activemq.ActiveMQConnection;
```

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

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

```
/**
```

```
 * Created by BlackCat.
```

```
 * Date 2017/8/10.
```

```
 * Time 17:11
```

```
 */
```

```
public class Sender {
```

```
    private static final int SEND_NUMBER = 5;
```

```
    public static void main(String[] args) {
```

```
        // ConnectionFactory : 连接工厂, JMS用它创建连接
```

```
        ConnectionFactory connectionFactory; //Connection : JMS 客户端到JMS
```

```
        //Provider 的连接
```

```
        Connection connection = null;
```

```
        Session session; //Session: 一个发送或者接受消息的线程
```

```
        Destination destination; //Destiantion : 消息的目的地: 消息发送给谁
```

```
        MessageProducer producer; //MessageProducer 消息发送者
```

```
        //构造connectionFactory对象实例对象, 此处采用ActiveMQ的实现jar
```

```
        connectionFactory = new ActiveMQConnectionFactory(
```

```
            ActiveMQConnection.DEFAULT_USER,
```

```
            ActiveMQConnection.DEFAULT_PASSWORD,
```

```
            "tcp://localhost:61616");
```

```

try {
    //构造从工厂得到连接对象
    connection = connectionFactory.createConnection();
    //启动
    connection.start();
    //获取操作连接
    session = connection.createSession(Boolean.TRUE,
        Session.AUTO_ACKNOWLEDGE);
    //获取session注意参数是一个服务器的queue，须在ActiveMQ的Queues配置
    destination = session.createQueue("FirstQueue");
    //得到消息生成者（发送者）
    producer = session.createProducer(destination);
    //设置不持久化，此处学习，实际根据项目定
    producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
    //构造消息，此处写死，项目就是参数
    sendMessage(session,producer);
    session.commit();
} catch (JMSEException e) {
    e.printStackTrace();
}finally {
    try {
        if (null != connection){
            connection.close();
        }
    } catch (JMSEException e) {
        e.printStackTrace();
    }
}
}

```

```

private static void sendMessage(Session session, MessageProducer producer)
    throws JMSEException {
    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
        System.out.println("发送消息：" + "ActiveMQ 发送的消息" + i);
        producer.send(message);
    }
}
}

```

---



---

```
-----  
package com.bc.activemq;
```

```
import org.apache.activemq.ActiveMQConnection;  
import org.apache.activemq.ActiveMQConnectionFactory;  
import javax.jms.*;
```

```
/**
```

```
 * Created by BlackCat.
```

```
 * Date 2017/8/10.
```

```
 * Time 17:12
```

```
 */
```

```
public class Receiver {
```

```
    public static void main(String[] args) {
```

```
        // ConnectionFactory : 连接工厂, JMS 用它创建连接
```

```
        ConnectionFactory connectionFactory;
```

```
        // Connection : JMS 客户端到JMS Provider 的连接
```

```
        Connection connection = null;
```

```
        // Session: 一个发送或接收消息的线程
```

```
        Session session;
```

```
        // Destination : 消息的目的地;消息发送给谁.
```

```
        Destination destination;
```

```
        // 消费者, 消息接收者
```

```
        MessageConsumer consumer;
```

```
        connectionFactory = new ActiveMQConnectionFactory(  
            ActiveMQConnection.DEFAULT_USER,  
            ActiveMQConnection.DEFAULT_PASSWORD,  
            "tcp://localhost:61616");
```

```
    try {
```

```
        // 构造从工厂得到连接对象
```

```
        connection = connectionFactory.createConnection();
```

```
        // 启动
```

```
        connection.start();
```

```
        // 获取操作连接
```

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

```
        destination = session.createQueue("FirstQueue");
```

```
        consumer = session.createConsumer(destination);
```

```
        while (true){
```

```
            // 设置接收者接收消息的时间, 为了便于测试, 这里谁定为100s
```

```
            TextMessage message = (TextMessage) consumer.receive(100000);
```

```
            if (null != message){
```

```
                System.out.println("收到的消息" + message.getText());
```

```
            }else {
```

```

        break;
    }
}
} catch (JMSEException e) {
    e.printStackTrace();
} finally {
    try {
        if (null != connection) {
            connection.close();
        }
    } catch (JMSEException e) {
        e.printStackTrace();
    }
}
}
}
}

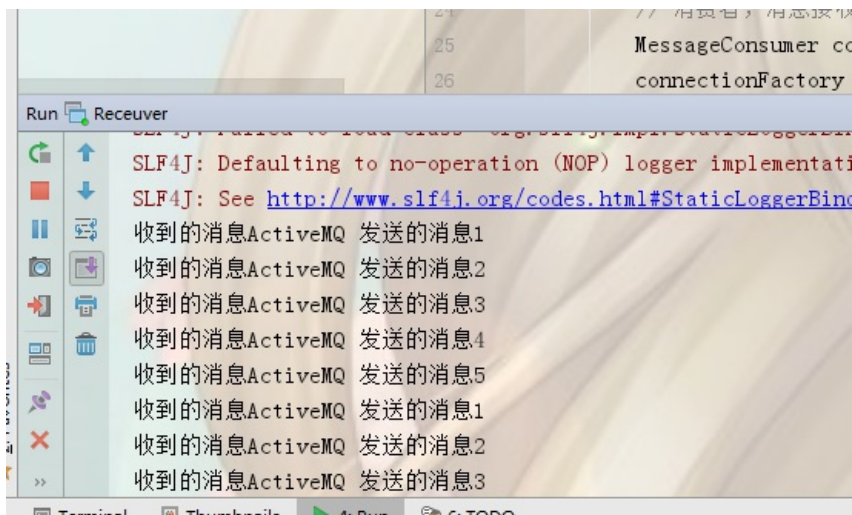
```

## 测试过程

先运行:Receiver.java







再运行:Sender.java

## 结果



Queue Name

## Queues

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
com.zheng.cms.admin.queue.default	0	0	0	0	Browse Active Consumers  	Send To Purge Delete
com.zheng.cms.queue.default	0	0	0	0	Browse Active Consumers  	Send To Purge Delete
FirstQueue	10	1	10	0	Browse Active Consumers  	Send To Purge Delete