JMS&ActiveMQ

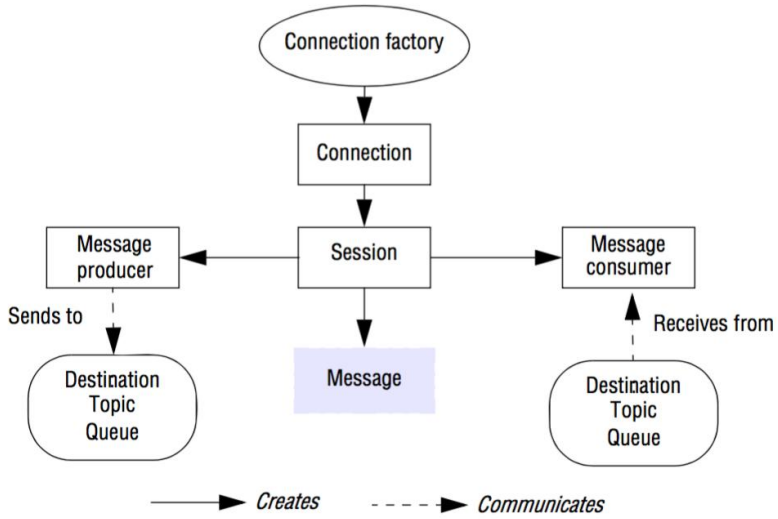
JMS基本角色

- JMS Message Producers : 消息生产者，向消息队列提供消息
- JMS Provider(Broker) : JMS提供商，如ActiveMQ，提供JMS Queues/Topics服务
- JMS Message listener/Consumer： 接收并使用消息

JMS编程模型

- JMS Provider有如数据库，Producers/Consumers发送/接收消息前需要先连接到Provider，与建立数据库连接相似，JMS ConnectionFactory负责创建JMS Connection，JMS Connection创建JMS Session
- JMS ConnectionFactory： Producers/Consumers用于创建一个到Provider的连接
- JMS Connection： 封装一个到Provider的连接
- JMS Session : 消息发送接收上下文
- 在JMS Provider上可以定义多个Queue和Topic，Producers发送消息到哪个Queue/Topic，称具体的那个Queue/Topic为Destination.
- JMS Destination : 一对一的Queue或者一对多的Topic

Figure 6.16 The programming model offered by JMS



- A JMS destination is an object supporting indirect communication in JMS. It is either a JMS topic or a JMS queue.

What is ActiveMQ

- The *Java Messaging Service* (JMS) [java.sun.com XI] is a specification of a standardized way for distributed Java programs to communicate indirectly. Most notably, as will be explained, the specification unifies the publish-subscribe and message queue paradigms at least superficially by supporting topics and queues as alternative destinations of messages.

- A wide variety of implementations of the common specification are now available. One of them is Apache ActiveMQ , others such as Joram from OW2, Java Messaging from JBoss, Sun's Open MQ and so on.

ActiveMQ的安装部署（mac）

- 在官网http://activemq.apache.org/download.html 下载最新的二进制包 apache-activemq-5.13.2-bin.tar.gz
- 解压到某个目录下，打开mac终端，输入以下命令切换到ActiveMQ主目录： cd /Users/xxxx/Downloads/apache-activemq-5.14.4/bin/macosx
- 输入以下命令启动ActiveMQ： Starting ActiveMQ Broker
- 在浏览器中输入如下地址: http://127.0.0.1:8161/admin/
- 然后输入用户名 admin 密码 admin便可登录控制台

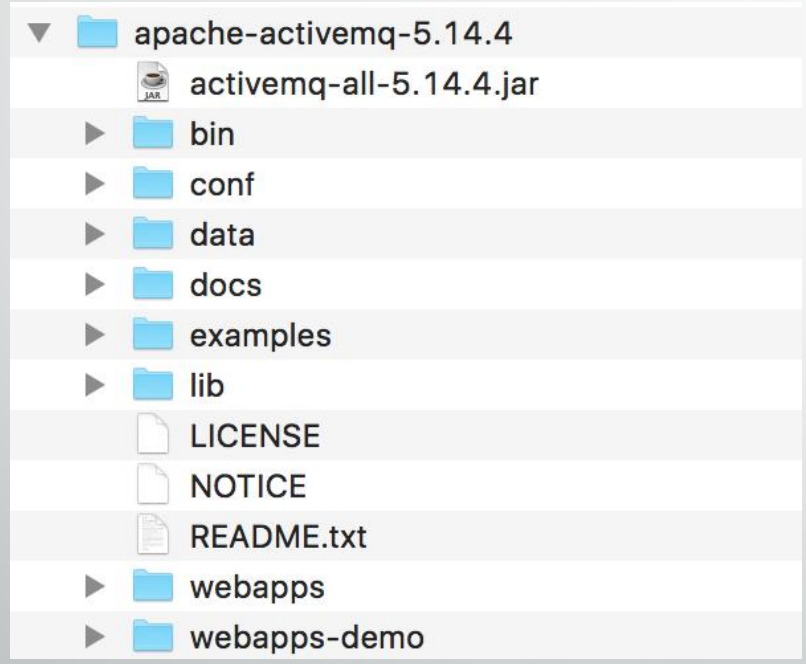
ActiveMQ的安装部署（windows）

- 在官网http://activemq.apache.org/download.html下载最新的windows版本：apache-activemq-5.14.4-bin.zip
- 解压下载的apache-activemq-5.10-20140603.133406-78-bin.zip压缩包到一个目录
- 根据自己的系统选择win32或win64文件夹
- 双击activemq.bat启动脚本
- 在浏览器中输入如下地址: http://127.0.0.1:8161/admin/
- 然后输入用户名 admin 密码 admin便可登录控制台

ActiveMQ安装问题

- Caused by: java.io.IOException: Failed to bind to server socket: tcp://0.0.0.0:61616?maximumConnections=1000&wireformat.maxFrameSize=104857600 due to: java.net.SocketException: Unrecognized Windows Sockets error: 0: JVM_Bind
- 可能是端口被占用了，需要对应的进程杀掉或者将对应的服务停止

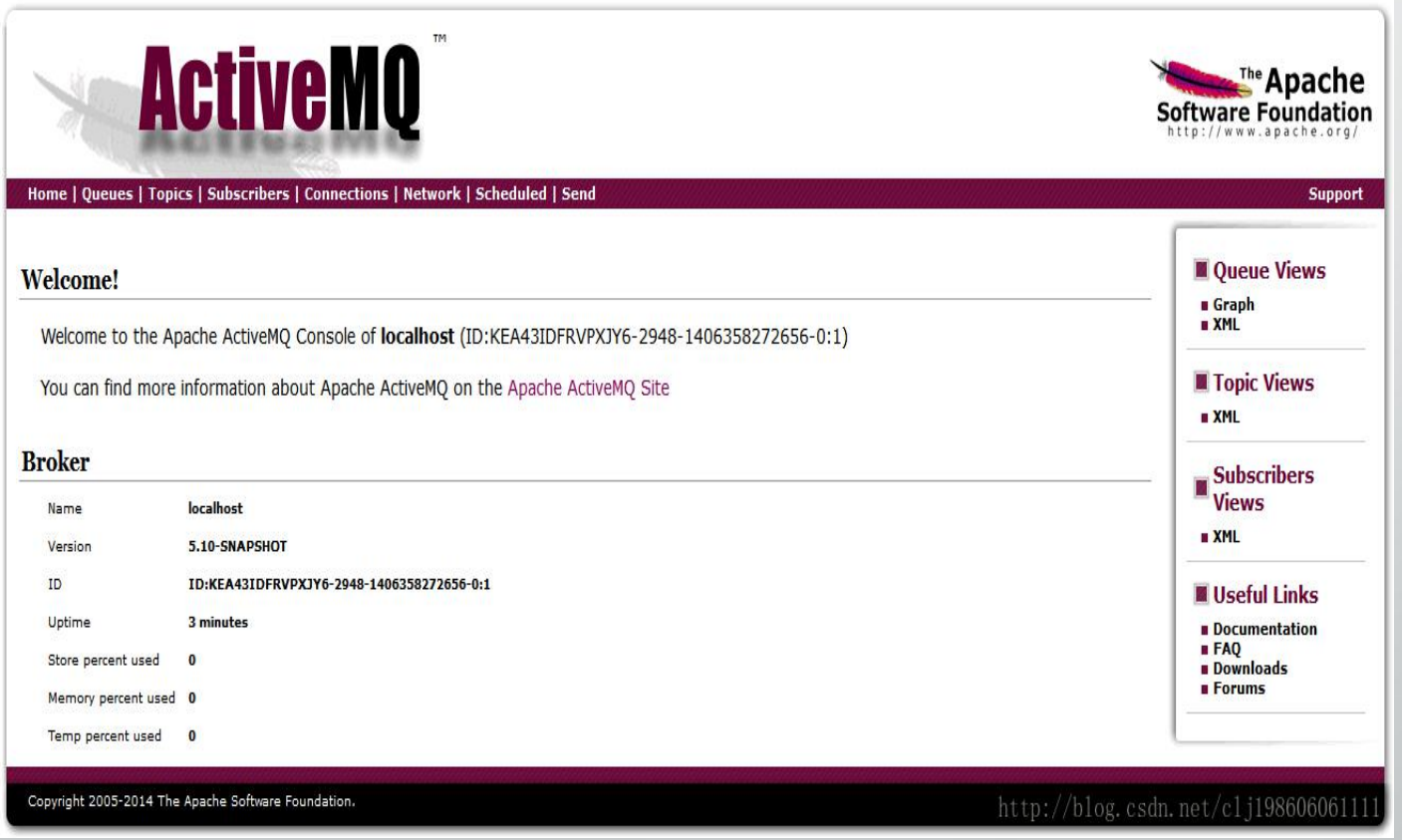
ActiveMQ目录结构



- bin存放的是脚本文件
- conf存放的是基本配置文件
- data存放的是日志文件
- docs存放的是说明文档
- examples存放的是简单的实例
- lib存放的是activemq所需jar包
- webapps用于存放项目的目录
- webapps-demo系统示例代码
- activemq-all-5.8.0.jar ActiveMQ的binary

ActiveMQ控制台界面

Queues：队列方式消息
Topics：主题方式消息
Subscribers：消息订阅监控查询
Connections：可以查看链接数，分别可以查看xmpp、ssl、stomp、openwire、ws和网络链接
Network：是网络链接数监控
Scheduled：查看消息调度
Send：可以发送消息数据



消息通信模式

- 发布—订阅
- 点到点
- 请求—应答

ActiveMQ消息类型

- TextMessage(文本消息)

将数据作为简单字符串存放在主体中(XML就可以作为字符串发)

TextMessage msg = session.createTextMessage();

msg.setText(text);

- MapMessage(映射消息)

使用一张映射表来存放其主体内容（参照Jms API）

MapMessage msg = session.createMapMessage();

msg.setString("CUSTOMER_NAME","John");

msg.setInt("CUSTOMER_AGE",12);

String s = msg.getString("CUSTOMER_NAME");

int age = msg.getInt("CUSTOMER_AGE");

- ObjectMessage()

用于往消息中写入可序列化的对象。消息中可以存放一个对象，如果要存放多个对象，需要建立一个对象集合，然后把这个

集合写入消息。

getObject()

setObject(Serializable s)

- ByteMessage(字节消息)

将字节流存放在消息主体中。适合于下列情况：必须压缩发送的大量数据、需要与现有消息格式保持一致等（参照Jms API）

byte[] data;

BytesMessage msg = session.createBytesMessage();

msg.wirte(data);

byte[] msgData = new byte[256];

int bytesRead = msg.readBytes(msgData);

- StreamMessage()

用于处理原语类型。这里也支持属性字段和MapMessage所支持的数据类型。使用这种消息格式时，收发双方事先协商好字段的顺序，以保证写读顺序相同（参照Jms API）

StringMessage msg = session.createStreamMessage();

msg.writeString("John");

msg.writeInt(12);

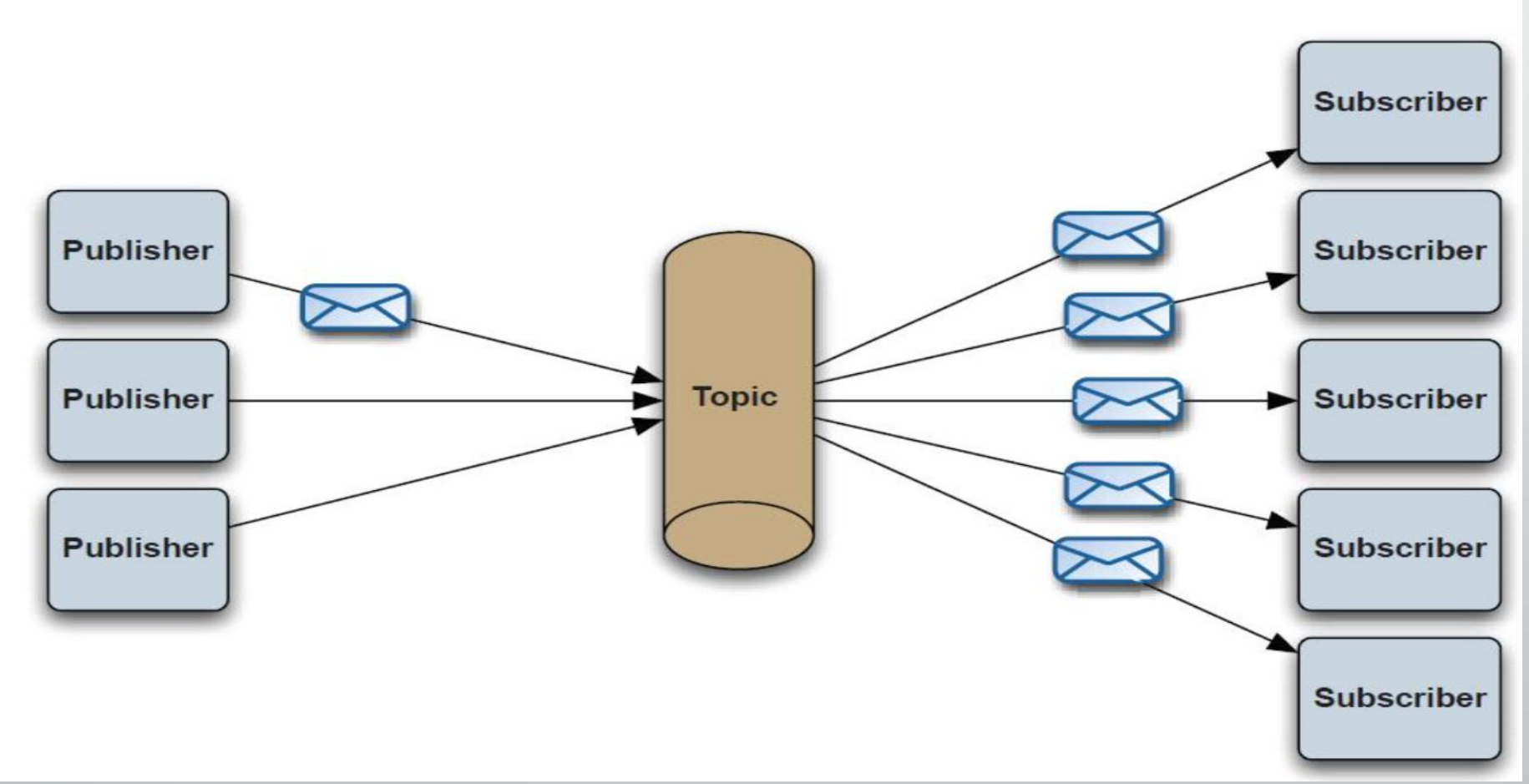
String s = msg.readString();

Int age = msg.readInt();

ActiveMQ基础流程（与JMS对应）

- 1. 获得JMS connection factory. 通过我们提供特定环境的连接信息来构造factory。
- 2. 利用factory构造JMS connection
- 3. 启动connection
- 4. 通过connection创建JMS session.
- 5. 指定JMS destination.
- 6. 创建JMS producer或者创建JMS message并提供destination.
- 7. 创建JMS consumer或注册JMS message listener.
- 8. 发送和接收JMS message.
- 9. 关闭所有JMS资源， 包括connection, session, producer, consumer等。

publish-subscribe



Publish流程

发布消息的步骤如下：

1. 从连接工厂中拿出Connecion对象。
2. 和服务器建立连接（Connection.start()）。
3. 创建会话（Session）对象。
4. 通过Session，在指定的Topic创建消息发布者（MessageProducer）。
5. 使用Session创建消息。
6. 使用消息生产者发布消息。

Subscribe流程

订阅消息的步骤如下：

1. 从连接工厂中拿出Connecion对象。
2. 和服务器建立连接（Connection.start()）。
3. 创建会话（Session）对象。
4. 通过Session，在指定的Topic创建消息订阅者（MessageConsumer）。

订阅消息：

- 调用messageConsumer.receive方法接受消息,如果队列上有消息，则receive方法返回该消息对象，如果队列上无消息，则该方法阻塞。
- 也可以以为Session指定MessageListener对象的方式来订阅消息，该方法的好处在于，一旦有新消息到来，会自动触发该对象的onMessage方法执行。

publish-subscribe sample

- 见ActiveMQSample.zip

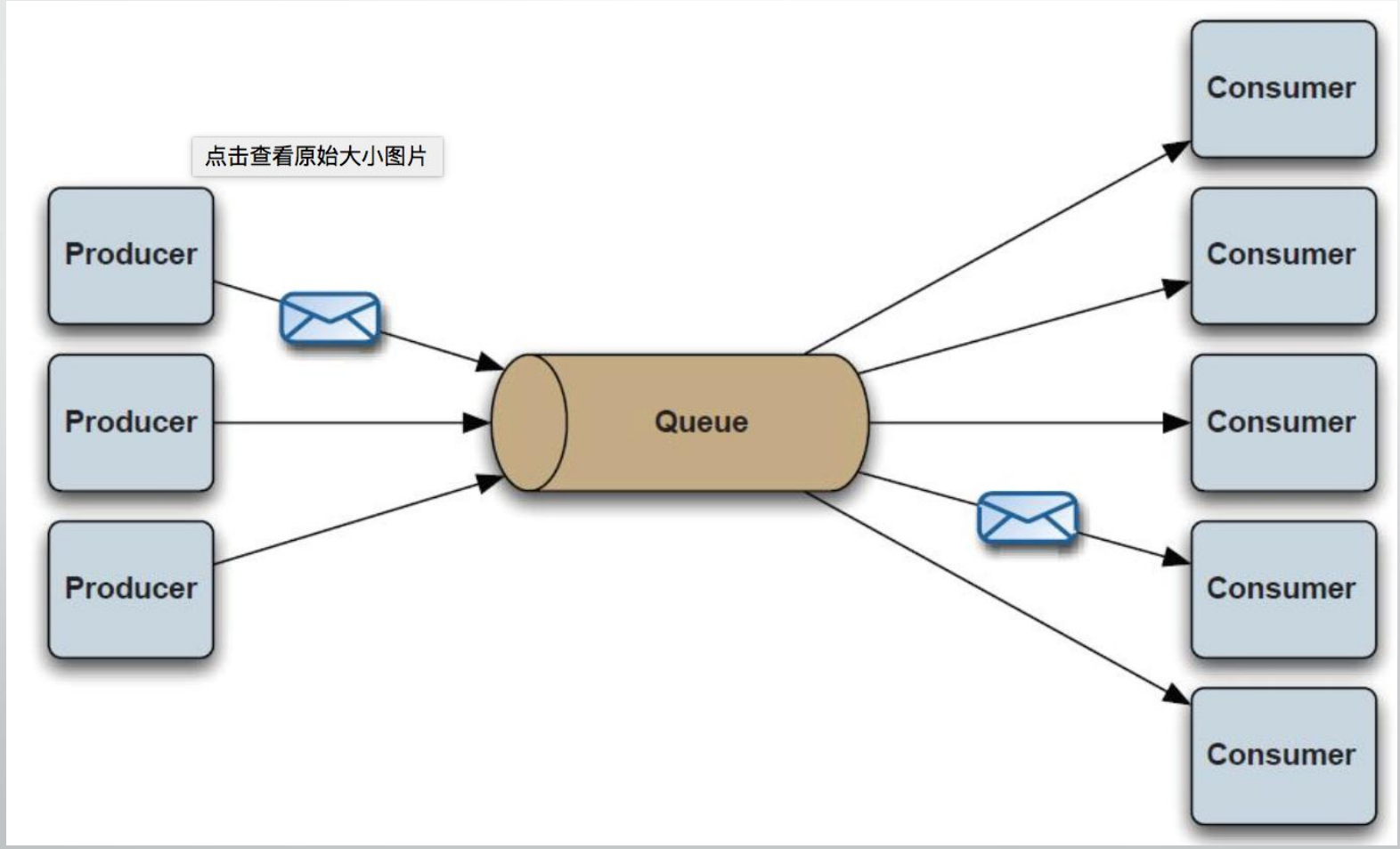
non-durable subscription & durable subscription

- 非持久订阅只有当客户端处于激活状态，也就是和 ActiveMQ 保持连接状态才能收到发送到某个主题的消息，而当客户端处于离线状态，这个时间段发到主题的消息将会丢失，永远不会收到。
- 持久订阅时，客户端向ActiveMQ 注册一个识别自己身份的 ID，当这个客户端处于离线时，ActiveMQ会为这个 ID 保存所有发送到主题的消息，当客户端再次连接到ActiveMQ 时，会根据自己的 ID 得到所有当自己处于离线时发送到主题的消息。

durable subscription

```
ConnFactory connectionFactory = new ConnFactory();
// 创建connection
Connection connection = connectionFactory.createConnection();
connection.setClientID(username); // 持久订阅需要多设置这个
connection.start();
// 创建session
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
// 创建destination
Topic topic = session.createTopic(topicname); // Topic名称
// MessageConsumer consumer = session.createConsumer(topic); //普通订阅
MessageConsumer consumer = session.createDurableSubscriber(topic, username);//持久订阅
consumer.setMessageListener(this);
```

P2P


点击查看原始大小图片
Producer
Producer
Producer
Queue
Consumer
Consumer
Consumer
Consumer
Consumer

pub-sub与P2p的区别

- pub-sub:一个topic有一个发送者和多个接收者
- P2P:一个queue只有一个发送者和一个接收者

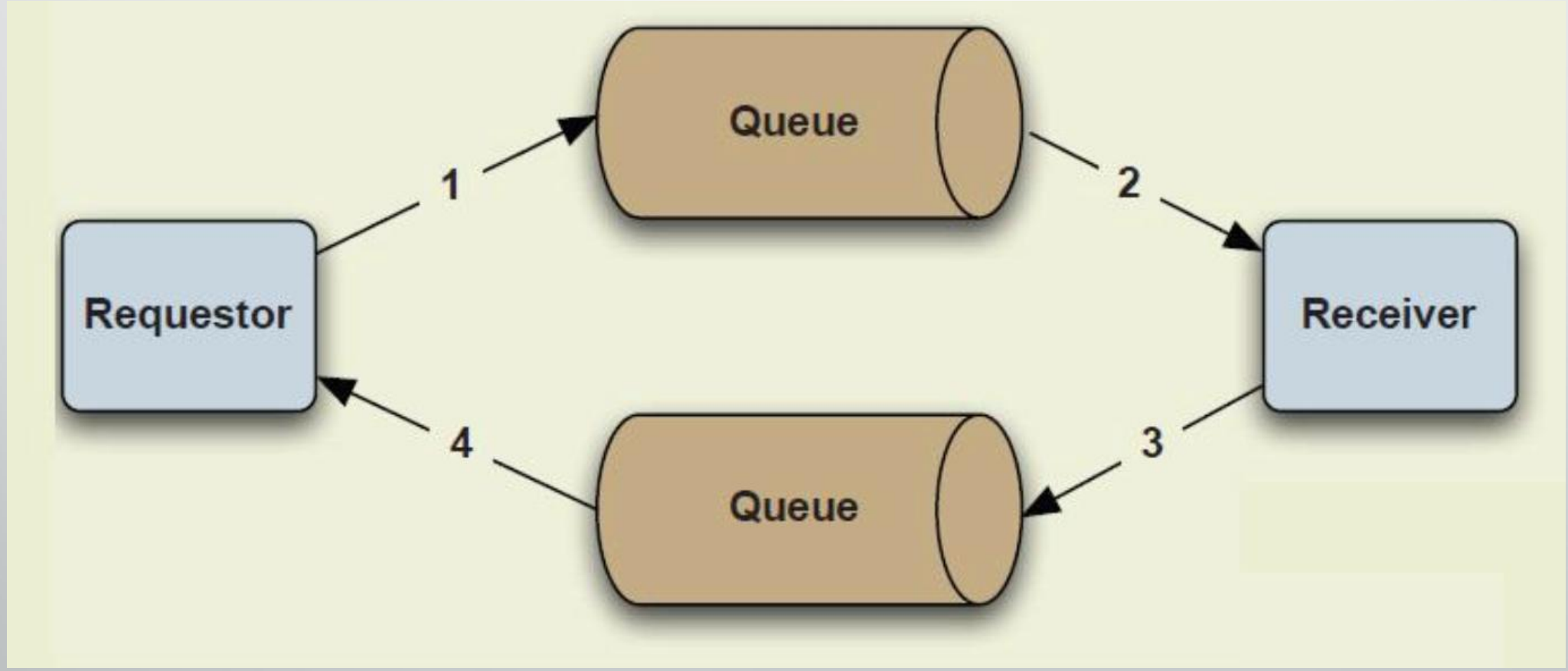
P2P的流程

- 生产和消费的流程是一样的
- 区别只在于创建的session是topic还是queue

P2P Sample

- 见ActiveMQSample.zip
- 如果你先执行发送消息的程序，在启动接受消息的程序，所有的消息都有可能被同一消费者消费，这是ActiveMQ为了提高效率，重用了同一个连接传输了所有的消息。其他的MQ产品未必会这么做，SnoicMQ它就会以一种随机的方式分发给不同的消费者。一旦你创建好消费者先监听消息队列，然后，再发送消息，由于这个时候，消费者与JMS Server之间的连接都已经建立，所以消息会随机的分发到不同的消费者。

Request-Response



Request-Response

- 前面的两种模式中都是一方负责发送消息而另外一方负责处理。而请求－应答为一种一应一答的过程，需要双方都能给对方发送消息。
- 请求-应答方式并不是JMS规范系统默认提供的一种通信方式，在JMS里面，如果要实现请求/应答的方式，可以利用JMSReplyTo和JMSCorrelationID消息头来将通信的双方关联起来。另外，QueueRequestor和TopicRequestor能够支持简单的请求/应答过程。

Request-Response Sample

- 见ActiveMQSample.zip

AcitveMQ相关的书

- ActiveMQ in Action
- Instant Apache ActiveMQ Messaging Application Development
- Mobile and Web Messaging
- Apache ActiveMQ Reference Guide

End