

[DZone](#) > [Integration Zone](#) > [JMS Queue Server and Client Example Based On an ActiveMQ Provider](#)

JMS Queue Server and Client Example Based On an ActiveMQ Provider

The basics of messaging services for Java apps, along with a client example based on ActiveMQ.



by Tomasz J-ski · Jan. 16, 16 · Integration Zone · Opinion

Like (3) Comment (0) Save Tweet

18.93k Views

What is JMS?

Java Message Service could be considered as high level framework that communicates two or more clients each other by sending messages. JMS is fully supported by Java EE and has been described in JSR -914 (see links section).

Here are the base operations that we can do:

- Send messages
- Receive messages

Typically, one client creates a message and sends it to the JMS server, and another client receives it and performs some kind of operation. Of course, this is a high level overview, and low level processes are more complicated.

Where Do We Use It?

Well we could use JMS everywhere where we need to increase scalability, integrate different environments, include batch processing support, or asynchronous communication without time pressure.

A common use is to send email, the client creates an email message from the template, and sends it to the MQ server. The receiver then receives the message from MQ server and sends the email.

But as you can see there are many other processes that could be implemented that way.

Sample MQ Providers List?

- Amazon SQS
- ActiveMQ

- Oracle Weblogic
- HornetQ (previously JBoss messaging) see link number 2.

What is ActiveMQ?

ActiveMQ is an implementation of Message Oriented Middleware which is used to communicate via JMS clients. As we can read at the official page (see link number 3), it is an open source system that is fast, supports cross language clients and protocols, and, most importantly, fully supports JMS 1.1 specifications.

The latest version is 5.13.0, and this version will be used in the demo.

ActiveMQ Server

From the download section of the official ActiveMQ page (see link number 3), you can download the latest server binary package. After un-zipping, it took almost 100MB of disk space. Unzipped binaries are all you need to start your server.

My Windows machine starting script is located at: `apache-activemq-5.13.0\bin`. I use `activemq.bat` but there are similar scripts for Linux too.

Bat script setup local env and finally run the jar file called `activemq.jar`, which is located at bin folder.

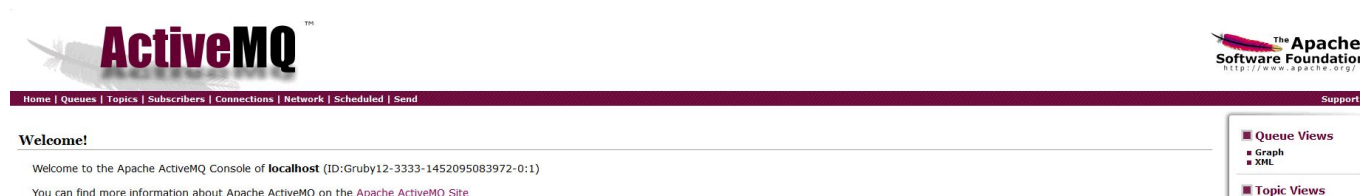
After running in the command line, starting the script, and successfully starting the server you should see something similar to this:

server you should see something similar to this:

```
Java Runtime: Oracle Corporation 1.8.0_65 C:\Program Files\Java\jdk1.8.0_65\jre
Heap sizes: current=1005056k free=984084k max=1005056k
JVM args: -Dcom.sun.management.jmxremote -Xms1G -Xmx1G -Djava.util.logging.config.file=logging.properties -Djava.
security.auth.login.config=E:\Tools\apache-activemq-5.13.0\bin\..\conf\login.config -Dactivemq.classpath=E:\Tools\apa
che-activemq-5.13.0\bin\..\conf;E:\Tools\apache-activemq-5.13.0\bin\..\conf;E:\Tools\apache-activemq-5.13.0\bin\..\co
nf; -Dactivemq.home=E:\Tools\apache-activemq-5.13.0\bin\..\ -Dactivemq.base=E:\Tools\apache-activemq-5.13.0\bin\..\ -Da
ctivemq.conf=E:\Tools\apache-activemq-5.13.0\bin\..\conf -Dactivemq.data=E:\Tools\apache-activemq-5.13.0\bin\..\data
-Djava.io.tmpdir=E:\Tools\apache-activemq-5.13.0\bin\..\data\tmp
Extensions classpath:
[E:\Tools\apache-activemq-5.13.0\bin\..\lib;E:\Tools\apache-activemq-5.13.0\bin\..\lib\camel;E:\Tools\apache-activem
q-5.13.0\bin\..\lib\optional;E:\Tools\apache-activemq-5.13.0\bin\..\lib\web;E:\Tools\apache-activemq-5.13.0\bin\..\l
ib\extra]
ACTIVEMQ_HOME: E:\Tools\apache-activemq-5.13.0\bin\..
ACTIVEMQ_BASE: E:\Tools\apache-activemq-5.13.0\bin\..
ACTIVEMQ_CONF: E:\Tools\apache-activemq-5.13.0\bin\..\conf
ACTIVEMQ_DATA: E:\Tools\apache-activemq-5.13.0\bin\..\data
Loading message broker from: xbean:activemq.xml
INFO | Refreshing org.apache.activemq.xbean.XBeanBrokerFactory$1@20e2cbe0: startup date [Wed Jan 06 16:44:42 CET 201
6]; root of context hierarchy
INFO | PListStore[E:\Tools\apache-activemq-5.13.0\bin\..\data\localhost\tmp_storage] started
INFO | Using Persistence Adapter: KahaDBPersistenceAdapter[E:\Tools\apache-activemq-5.13.0\bin\..\data\kahadb]
INFO | KahaDB is version 6
INFO | Recovering from the journal @1:503
INFO | Recovery replayed 52 operations from the journal in 0.025 seconds.
INFO | Apache ActiveMQ 5.13.0 (localhost, ID:Gruby12-3333-1452095083972-0:1) is starting
INFO | Listening for connections at: tcp://Gruby12:61616?maximumConnections=1000&wireFormat.maxFrameSize=104857600
INFO | Connector openwire started
INFO | Listening for connections at: amqp://Gruby12:5672?maximumConnections=1000&wireFormat.maxFrameSize=104857600
INFO | Connector amqp started
INFO | Listening for connections at: stomp://Gruby12:61613?maximumConnections=1000&wireFormat.maxFrameSize=104857600
INFO | Connector stomp started
INFO | Listening for connections at: mqtt://Gruby12:1883?maximumConnections=1000&wireFormat.maxFrameSize=104857600
INFO | Connector mqtt started
INFO | Listening for connections at ws://Gruby12:61614?maximumConnections=1000&wireFormat.maxFrameSize=104857600
INFO | Connector ws started
INFO | Apache ActiveMQ 5.13.0 (localhost, ID:Gruby12-3333-1452095083972-0:1) started
```

Now you could type <http://localhost:8161/admin/> to access Your ActiveMQ server administration console, note that the default login and password is **admin**.

It should look like this:



Broker	
Name	localhost
Version	5.13.0
ID	ID:Grubyl2-3333-1452095083972-01
Uptime	2 minutes
Store percent used	0
Memory percent used	0
Temp percent used	0

- XML
- Subscribers
- Views
- XML
- Useful Links
 - Documentation
 - FAQ
 - Downloads
 - Forums

Copyright 2005-2019 The Apache Software Foundation.

Now you could add your first queue in the Queues menu.

ActiveMQ Client Example

I created a sample client for an ActiveMQ server in Java 8 using threads and lambdas. You can find all the code at my GitHub: <https://github.com/tjancz/jms-activemq-example.git>

Below you can see the sender code:

```
1 package it.janczewski.examples.utils;
2
3 import org.apache.activemq.ActiveMQConnectionFactory;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6
7 import javax.jms.*;
8 import java.math.BigInteger;
9 import java.security.SecureRandom;
10
11 public class Sender {
12     private final Logger logger = LoggerFactory.getLogger(Sender.class);
13     private SecureRandom random = new SecureRandom();
14
15     public void createTask(){
16         String taskName = generateTaskName();
17         Runnable sendTask = () -> {
18             ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFac
```

```

19     Connection connection = null;
20     try {
21         connection = connectionFactory.createConnection();
22         connection.start();
23         Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
24         Destination destination = session.createQueue("TJ Test");
25         MessageProducer producer = session.createProducer(destination);
26         producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
27         String text = "Hello from: " + taskName + " : " + this.hashCode();
28         TextMessage message = session.createTextMessage(text);
29         logger.info("Sent message hash code: " + message.hashCode() + " : " + taskName);
30         producer.send(message);
31         session.close();
32         connection.close();
33     } catch (JMSEException e) {
34         logger.error("Sender createTask method error", e);
35     }
36 };
37     new Thread(sendTask).start();
38 }
39
40     private String generateTaskName() {
41         return new BigInteger(20, random).toString(16);
42     }
43 }

```

And the receiver code:

```

1 package it.janczewski.examples.utils;
2
3 import org.apache.activemq.ActiveMQConnectionFactory;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import javax.jms.*;
7
8 public class Reciver implements ExceptionListener{

```

```

9     private final Logger logger = LoggerFactory.getLogger(Reciver.class);
10
11     public void createRecieveTask() {
12         Runnable recTask = () -> {
13             try {
14                 ActiveMQConnectionFactory connectionFactory = new ActiveMQConnection
15                 Connection connection = connectionFactory.createConnection();
16                 connection.start();
17                 connection.setExceptionListener(this);
18                 Session session = connection.createSession(false, Session.AUTO_ACKNO
19                 Destination destination = session.createQueue("TJ Test");
20                 MessageConsumer consumer = session.createConsumer(destination);
21                 Message message = consumer.receive(4000);
22                 if (message instanceof TextMessage) {
23                     TextMessage textMessage = (TextMessage) message;
24                     String text = textMessage.getText();
25                     logger.info("Received TextMessage object: " + text);
26                 } else {
27                     logger.info("Received other object type with message: " + messa
28                 }
29                 consumer.close();
30                 session.close();
31                 connection.close();
32
33                 } catch (JMSEException e) {
34                     logger.error("Reciver createRecieveTask method error", e);
35                 }
36             };
37             new Thread(recTask).start();
38         }
39
40     @Override
41     public void onException(JMSEException exception) {
42         logger.error("Recieve error occured.");
43     }
44 }

```

If you would like to see all working stuff, just clone it from my GitHub repo.

There is also great documentation on the ActiveMQ page.

Summary

JMS is not scary as it looks and, ActiveMQ is a great MOM that could easily help you set up message queues and integrate different systems.

Links

1. <https://jcp.org/aboutJava/communityprocess/final/jsr914/index.html>
2. <http://hornetq.jboss.org/>
3. <http://activemq.apache.org/>

Like This Article? Read More From DZone

 [DZone Article](#)
related article thumbnail
Mercury: Highly-scalable and Distributed Socket Messaging Library

 [DZone Article](#)
related article thumbnail
CSV to XML using DataWeave and Active MQ configuration

 [DZone Article](#)
related article thumbnail
Mule 4: JMS Pub and Sub With Transformation

 [Free DZone Refcard](#)
related refcard thumbnail
Introduction to Digital Asset Management via APIs

Topics: ACTIVE MQ, CLIENT, JAVA 8, JMS, MESSAGING MIDDLEWARE, MIDDLEWARE, SERVER

 Like (3)  Comment (0)  Save  Tweet

 18.93k Views

Opinions expressed by DZone contributors are their own.

Integration Partner Resources

ABOUT US

[About DZone](#)
[Send feedback](#)
[Careers](#)

CONTRIBUTE ON DZONE

[MVB Program](#)
[Zone Leader Program](#)
[Become a Contributor](#)
[Visit the Writers' Zone](#)

LEGAL

[Terms of Service](#)
[Privacy Policy](#)


ADVERTISE

[Developer Marketing Blog](#)
[Advertise with DZone](#)
[+1 \(919\) 238-7100](#)

CONTACT US

[600 Park Offices Drive](#)
[Suite 150](#)
[Research Triangle Park, NC](#)
[27709](#)
[support@dzone.com](#)
[+1 \(919\) 678-0300](#)

Let's be friends:    

DZone.com is powered by  AnswerHub logo