Use Java Messaging Service - Basics

In this practical you will learn how to send and receive JMS messages. We will use Apache ActiveMQ messaging server.

Table of Contents

[1. Download and install Apache ActiveMQ. 1](#_Toc525293268)

[2. Build a simple Sender application. 2](#_Toc525293269)

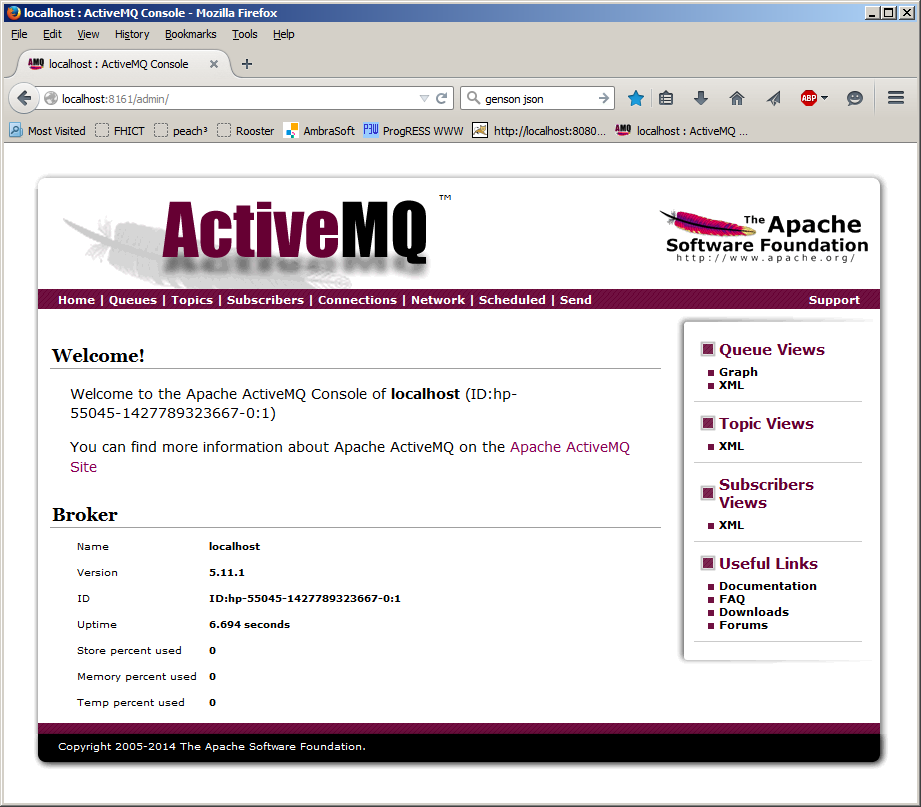
[4. Build a simple Receiver application 4](#_Toc525293270)

# 1. Download and install Apache ActiveMQ.

Download the latest stable release of Apache ActiveMQ from <http://activemq.apache.org/> . Just download the right .zip file, depending on your OS and unzip it.

You can use Command Prompt to start ActiveMQ : go to the bin folder in the installation folder (where you unzipped the downloaded file) and type “activemq start”. To stop ActiveMQ type "CTRL-C" in the console or command shell in which it is running.

If everything went well, you should be able to access ActiveMQ via your Internet Browser at <http://localhost:8161/>. You will be asked to log in. Default administrator username /password are amin/admin. Click on link “Manage ActiveMQ Broker” and you should see the page shown below. You can use links “Queues” and “Topics” to access your queues and topics.



# 2. Build a simple Sender application.

Start by creating an Empty Project in IntelliJ and creating a Gradle “jms-sender”module with Java plug-in. In order to be able to connect to ActiveMQ you need:

* If you do use gradle: add the “activemq-client” gradle dependency in the “jms-sender” module.
* If you do not use gradle: copy-paste file “activemq-all-5.15.6.jar” from the ActiveMQ folder to the lib folder of your module and add it as library in IntelliJ.

Code below shows how to create a TextMessage and send it via JMS:

Connection connection; // to connect to the ActiveMQ

Session session; // session for creating messages, producers and

Destination sendDestination; // reference to a queue/topic destination

MessageProducer producer; // for sending messages

**try** {

Properties props = **new** Properties();

props.setProperty(Context.***INITIAL\_CONTEXT\_FACTORY***, "org.apache.activemq.jndi.ActiveMQInitialContextFactory");

props.setProperty(Context.***PROVIDER\_URL***, "tcp://localhost:61616");

// connect to the Destination called “myFirstChannel”

// queue or topic: “queue.myFirstDestination” or “topic.myFirstDestination”

props.put(("queue.myFirstDestination"), "myFirstDestination");[[1]](#footnote-1)

Context jndiContext = **new** InitialContext(props);

ConnectionFactory connectionFactory = (ConnectionFactory) jndiContext

.lookup("ConnectionFactory");

connection = connectionFactory.createConnection();

session = connection.createSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);

// connect to the sender destination

sendDestination = (Destination) jndiContext.lookup("myFirstDestination");

producer = session.createProducer(sendDestination);

String body = "Hello, this is my first message!"; //or serialize an object!

// create a text message

Message msg **=** session.createTextMessage(body);

// send the message

producer.send(msg);

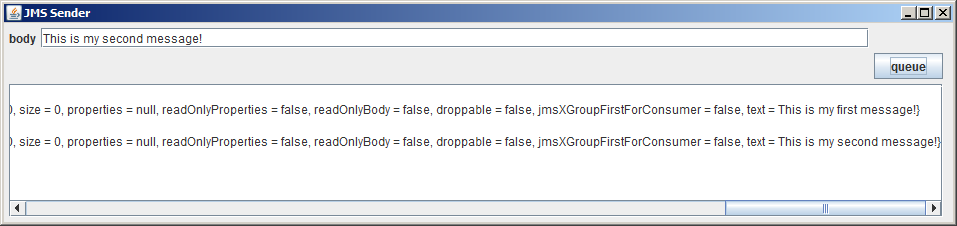
System.***out***.println("sent: " + msg);

} **catch** (NamingException | JMSException e) {

e.printStackTrace();

}

You should build a simple console or GUI application which can be used to send text messages and print them after sending:



Class Message has a lot of fields and methods: some ID’s, destination name that it was sent to, time stamps, etc. You can print the whole message to see which fields are available (next to field called “Text”). Note the following issues when printing a Message before and after it’s sent:

Message msg = session.createTextMessage(body);

// print all message attributes;

// but JMSDestination and JMSMessageID are both null

// Session makes the message via ActiveMQ. ActiveMQ assigns unique JMSMessageID

// to each message after sending the message.

System.***out***.println(msg);

producer.send(msg);

// print all message attributes: JMSMessgeID is set and J

// MSDestination is senderDestination name

System.***out***.println(msg);

**try** {

//print only the attributes you want to see

System.***out***.println("JMSMessageID=" + msg.getJMSMessageID()

+ " JMSDestination=" + msg.getJMSDestination()

+ " Text=" + ((TextMessage) msg).getText());

} **catch** (JMSException e) {

e.printStackTrace();

}

4. Build a simple Receiver application

Add a new Gradle “jms-receiver”module with Java plug-in to your project. In order to be able to connect to ActiveMQ you need the “activemq-client” gradle dependency in the “jms-receiver” module.

Code bellow shows how to receive JMS messages:

Connection connection; // to connect to the JMS

Session session; // session for creating consumers

Destination receiveDestination; reference to a queue/topic destination

MessageConsumer consumer; // for receiving messages

**try** {

Properties props = **new** Properties();

props.setProperty(Context.***INITIAL\_CONTEXT\_FACTORY***,

"org.apache.activemq.jndi.ActiveMQInitialContextFactory");

props.setProperty(Context.***PROVIDER\_URL***, "tcp://localhost:61616");

// connect to the Destination called “myFirstChannel”

// queue or topic: “queue.myFirstDestination” or

“topic.myFirstDestination”

props.put(("queue.myFirstDestination"), " myFirstDestination");

Context jndiContext = **new** InitialContext(props);

ConnectionFactory connectionFactory = (ConnectionFactory) jndiContext

.lookup("ConnectionFactory");

connection = connectionFactory.createConnection();

session = connection.createSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);

// connect to the receiver destination

receiveDestination = (Destination) jndiContext.lookup("myFirstDestination");

consumer = session.createConsumer(receiveDestination);

consumer.setMessageListener(**new** MessageListener() {

@Override

**public** **void** onMessage(Message msg) {

System.***out***.println("received: " + msg);

}

});

connection.start(); // this is needed to start receiving messages

} **catch** (NamingException | JMSException e) {

e.printStackTrace();

}

1. For topic: props.put(("topic.myFirstDestination"), "myFirstDestination"); [↑](#footnote-ref-1)