

Simple Application

In this recipe we are going to create a very simple JMS application that places a message on a queue in ActiveMQ and consumes that message from the queue immediately after. This example demonstrates many of the JMS APIs that we will be using in the future recipes.

1. Create a Maven Project
2. Add dependencies for ActiveMQ:

pom.xml

```
<dependencies>
  <dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-client</artifactId>
    <version>5.8.0</version>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.2</version>
  </dependency>
</dependencies>
```

3. Create a class to connect to ActiveMQ
SimpleJMS.java

```
public class SimpleJMS {

    private final String connectionUri = "tcp://localhost:
61616";
    private ActiveMQConnectionFactory connectionFactory;
    private Connection connection;
    private Session session;
    private Destination destination;

    public void before() throws Exception {
        connectionFactory = new
ActiveMQConnectionFactory(connectionUri);
        connection = connectionFactory.createConnection();
        connection.start();
        session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
        destination = session.createQueue("MyQueue");
    }

    public void after() throws Exception {
        if (connection != null) {
            connection.close();
        }
    }

    public void after() throws Exception {
        if (connection != null) {
            connection.close();
        }
    }
}
```

```

    public void run() throws Exception {

        MessageProducer producer =
session.createProducer(destination);
        try {
            TextMessage message =
session.createTextMessage();
            message.setText("We sent a Message!");
            producer.send(message);
        } finally {
            producer.close();
        }

        MessageConsumer consumer =
session.createConsumer(destination);
        try {
            TextMessage message = (TextMessage)
consumer.receive();
            System.out.println(message.getText());
        } finally {
            consumer.close();
        }
    }

    public static void main(String[] args) {
        SimpleJMS example = new SimpleJMS();
        System.out.print("\n\n\n");
        System.out.println("Starting SimpleJMS example
now...");
        try {
            example.before();
            example.run();
            example.after();
        } catch (Exception e) {
            System.out.println("Caught an exception during
the example: " + e.getMessage());
        }
        System.out.println("Finished running the SimpleJMS
example.");
        System.out.print("\n\n\n");
    }
}

```

4. Start ActiveMQ broker

```
bin# activemq start
```

NOTE: If broker is already running, you do not need to start again.

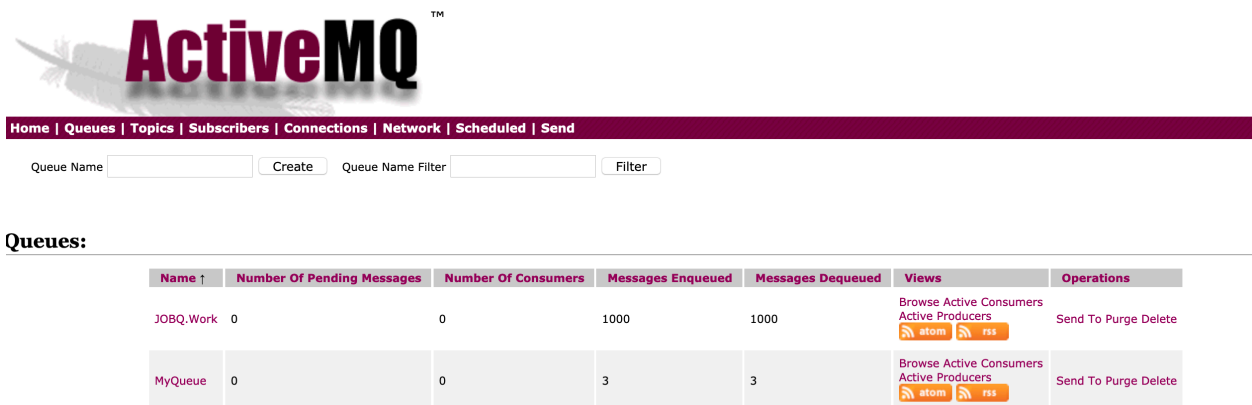
5. RUN the Application

SimpleJMS → Run As → Java Application

6. OUTPUT

```
Starting SimpleJMS example now...
log4j:WARN No appenders could be found for logger (org.apache.activemq.transport.WireFormatNegotiator).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
We sent a Message!
Finished running the SimpleJMS example.
```

7. Observe Queue in the Admin Console



The screenshot shows the ActiveMQ Admin Console interface. At the top is the ActiveMQ logo. Below it is a navigation bar with links: Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send. Under the 'Queues' link, there is a search bar with 'Queue Name' and 'Filter' buttons, and a 'Create' button. The main content area is titled 'Queues:' and contains a table with the following data:

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
JOBQ.Work	0	0	1000	1000	Browse Active Consumers Active Producers atom rss	Send To Purge Delete
MyQueue	0	0	3	3	Browse Active Consumers Active Producers atom rss	Send To Purge Delete

Possible Warnings

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See <http://www.slf4j.org/codes.html#StaticLoggerBinder>
for further details.

log4j:WARN No appenders could be found for logger
(org.apache.activemq.transport.WireFormatNegotiator).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See <http://logging.apache.org/log4j/1.2/faq.html#noconfig> for more info.

8. Add log4j.properties and append logger statements to console
src/main/resources/log4j.properties

```
# Root logger option
log4j.rootLogger=INFO, console

# CONSOLE appender not used by default
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.out
log4j.appender.console.immediateFlush=true
log4j.appender.console.encoding=UTF-8

# File appender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.conversionPattern=%d [%t] %-5p
%C - %m%n
```

9. Change **System.out** lines to **logger.info()** in **SimpleJMS.java**
10. RUN the APP & no warnings must be displayed.

