ABOUT RAM MOKKAPATY

Ram holds a master's degree in Machine Design from IT B.H.U. His expertise lies in test driven development and re-factoring. He is passionate about open source technologies and actively blogs on various java and open-source technologies like spring. He works as a principal Engineer in the logistics domain.

JMS Queue Example

Posted by: Ram Mokkapaty in jms November 5th, 2015 0 50 Views

JMS Message queue is a destination to which producers send messages. Consumer connects to the broker to receive the message sitting in the queue. Queue is used in point-to-point messaging. In point-to-point messaging, there may be more than one receiver connected to the queue but each message in the queue may only be consumed by one of the queue's receivers.

The messages can be sent and received either synchronously or asynchronously.

In this article, we will see some examples of JMS Queue.

1. Dependencies

In order to send and receive JMS messages to and from a JMS message broker, we need to include the message service library. In this example we are using activeMq so our pom.xml will have dependencies related to spring as well as activeMq.

pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.javacodegeeks.jms</groupId>
    <artifactId>springJmsQueue</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.apache.activemq</groupId>
            <artifactId>activemq-all</artifactId>
            <version>5.12.0</version>
        </dependency>
    </dependencies>

</project>
```

2. Creating a Queue

First let's see how to create a queue.

In order to create a queue object, you need to first create a session and then call

```
createQueue()
```

on the session object. You need to pass the queue name to it.

```
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
Queue queue = session.createQueue("customerQueue");
```

The queue stores all messages until they're delivered or until they expire.

3. Sending message to a Queue

```
2  producer.send(msg);
```

As you can see from above, a producer sends a message to the queue.

4. Receive message from Queue

Each message received on the queue is delivered once and only once to a single consumer which is why this style of messaging is called point-to-point messaging. Consumer will first connect to the broker to receive the message from the queue. Just like the producer, consumer also needs a session using which it will connect to the queue.

```
1  MessageConsumer consumer = session.createConsumer(queue);
2  connection.start();
```

Note that connection is started so that any message listener registered will get the notification as soon as a message lands in the queue.

Consumer receives the message using

`MessageConsumer.receive()`

method or asynchronously by registering a

`MessageListener`

implementation using the

`MessageConsumer.setMessageListener()`

method. Multiple consumers can be registered on a single queue but only one consumer will receive a given message.

```
1  TextMessage textMsg = (TextMessage) consumer.receive();
2  System.out.println(textMsg);
3  System.out.println("Received: " + textMsg.getText());
```

JmsMessageQueueExample:

```
01 package com.javacodegeeks.jms;
02
03 import java.net.URI;
04 import java.net.URISyntaxException;
05
06 import javax.jms.Connection;
07 import javax.jms.ConnectionFactory;
08 import javax.jms.Message;
09 import javax.jms.MessageConsumer;
10 import javax.jms.MessageProducer;
11 import javax.jms.Queue;
12 import javax.jms.Session;
13 import javax.jms.TextMessage;
14
15 import org.apache.activemq.ActiveMQConnectionFactory;
16 import org.apache.activemq.broker.BrokerFactory;
17 import org.apache.activemq.broker.BrokerService;
18
19 public class JmsMessageQueueExample {
20     public static void main(String[] args) throws URISyntaxException, Exception {
21         BrokerService broker = BrokerFactory.createBroker(new URI(
22                 "broker:(tcp://localhost:61616)"));
23         broker.start();
24         Connection connection = null;
25         try {
26             // Producer
27             ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(
28                     "tcp://localhost:61616");
29             connection = connectionFactory.createConnection();
30             Session session = connection.createSession(false,
31                     Session.AUTO_ACKNOWLEDGE);
32             Queue queue = session.createQueue("customerQueue");
33             String payload = "Important Task";
34             Message msg = session.createTextMessage(payload);
35             MessageProducer producer = session.createProducer(queue);
36             System.out.println("Sending text '" + payload + "'");
37             producer.send(msg);
38
39             // Consumer
40             MessageConsumer consumer = session.createConsumer(queue);
41             connection.start();
42             TextMessage textMsg = (TextMessage) consumer.receive();
43             System.out.println(textMsg);
44             System.out.println("Received: " + textMsg.getText());
45             session.close();
46         } finally {
47             if (connection != null) {
48                 connection.close();
49             }
50                         broker.stop();
51         }
52     }
53
54 }
```

Output:

```
01 INFO | JMX consoles can connect to service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
02  INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-
   data\localhost\tmp_storage] started
03  INFO | Using Persistence Adapter:
```

```
 INFO | Recovery replayed 1 operations from the journal in 0.011 seconds.
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-57715-1446468253396-0:1) is starting
 INFO | Listening for connections at: tcp://127.0.0.1:61616
 INFO | Connector tcp://127.0.0.1:61616 started
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-57715-1446468253396-0:1) started
 INFO | For help or more information please see: http://activemq.apache.org
 WARN | Store limit is 102400 mb (current store usage is 0 mb). The data directory:
C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\KahaDB only has 34512 mb of
usable space - resetting to maximum available disk space: 34512 mb
 WARN | Temporary Store limit is 51200 mb, whilst the temporary data directory:
C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage only has 34512 mb
of usable space - resetting to maximum available 34512 mb.
Sending text 'Important Task'
ActiveMQTextMessage {commandId = 5, responseRequired = true, messageId = ID:INMAA1-L1005-57715-
1446468253396-3:1:1:1:1, originalDestination = null, originalTransactionId = null, producerId =
ID:INMAA1-L1005-57715-1446468253396-3:1:1:1, destination = queue://customerQueue, transactionId =
null, expiration = 0, timestamp = 1446468253638, arrival = 0, brokerInTime = 1446468253639,
brokerOutTime = 1446468253663, correlationId = null, replyTo = null, persistent = true, type =
null, priority = 4, groupID = null, groupSequence = 0, targetConsumerId = null, compressed =
false, userID = null, content = org.apache.activemq.util.ByteSequence@77be656f,
marshalledProperties = null, dataStructure = null, redeliveryCounter = 0, size = 0, properties =
null, readOnlyProperties = true, readOnlyBody = true, droppable = false,
jmsXGroupFirstForConsumer = false, text = Important Task}
Received: Important Task
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-57715-1446468253396-0:1) is shutting
down
 INFO | Connector tcp://127.0.0.1:61616 stopped
 INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-
data\localhost\tmp_storage] stopped
 INFO | Stopping async queue tasks
 INFO | Stopping async topic tasks
 INFO | Stopped KahaDB
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-57715-1446468253396-0:1) uptime 0.906
seconds
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-57715-1446468253396-0:1) is shutdown
```

5. Receiving a message Asynchronously

In our last example, we have seen consumer receiving message explicitly using MessageConsumer.receive(). In in this section, we will see ow a consumer can register a message listener. Instead of explicitly receiving the message, consumer just registers a message listener. Moment a message lands in the queue, the broker passes on the message to one of the message listeners.

Let's first create a message listener.

A message listener is created by implementing

```
javax.jms.MessageListener
```

and implementing

```
onMessage(Message)
```

.

ConsumerMessageListener:

```
package com.javacodegeeks.jms;

import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;

public class ConsumerMessageListener implements MessageListener {
    private String consumerName;
    public ConsumerMessageListener(String consumerName) {
        this.consumerName = consumerName;
    }

    public void onMessage(Message message) {
        TextMessage textMessage = (TextMessage) message;
        try {
            System.out.println(consumerName + " received " + textMessage.getText());
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }

}
```

Consumer will register its own message listener. It will pass a name to it so we know which consumer is consuming the message.

```
// Consumer
MessageConsumer consumer = session.createConsumer(queue);
consumer.setMessageListener(new ConsumerMessageListener("Consumer"));
```

Next, we need to make sure

```
start()
```

is called on connection object. This is an important step for the broker to make sure the message is passed on to one of the listeners.

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

JmsMessageAsynchronousQueueExample:

```
package com.javacodegeeks.jms;
```

```
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.jms.MessageProducer;
import javax.jms.Queue;
import javax.jms.Session;

import org.apache.activemq.ActiveMQConnectionFactory;
import org.apache.activemq.broker.BrokerFactory;
import org.apache.activemq.broker.BrokerService;

public class JmsMessageAsynchronousQueueExample {
    public static void main(String[] args) throws URISyntaxException, Exception {
        BrokerService broker = BrokerFactory.createBroker(new URI(
                "broker:(tcp://localhost:61616)"));
        broker.start();
        Connection connection = null;
        try {
            // Producer
            ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(
                    "tcp://localhost:61616");
            connection = connectionFactory.createConnection();
            Session session = connection.createSession(false,
                    Session.AUTO_ACKNOWLEDGE);
            Queue queue = session.createQueue("customerQueue");
            String payload = "Important Task";
            Message msg = session.createTextMessage(payload);
            MessageProducer producer = session.createProducer(queue);
            System.out.println("Sending text '" + payload + "'");
            producer.send(msg);

            // Consumer
            MessageConsumer consumer = session.createConsumer(queue);
            consumer.setMessageListener(new ConsumerMessageListener("Consumer"));
            connection.start();
            Thread.sleep(1000);
            session.close();
        } finally {
            if (connection != null) {
                connection.close();
            }
            broker.stop();
        }
    }

}
```

Output:

```
INFO | JMX consoles can connect to service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
 INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage] started
 INFO | Using Persistence Adapter: KahaDBPersistenceAdapter[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\KahaDB]
 INFO | KahaDB is version 6
 INFO | Recovering from the journal @1:153817
 INFO | Recovery replayed 1 operations from the journal in 0.011 seconds.
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-56230-1446467650870-0:1) is starting
 INFO | Listening for connections at: tcp://127.0.0.1:61616
 INFO | Connector tcp://127.0.0.1:61616 started
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-56230-1446467650870-0:1) started
 INFO | For help or more information please see: http://activemq.apache.org
 WARN | Store limit is 102400 mb (current store usage is 0 mb). The data directory: C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\KahaDB only has 34515 mb of usable space - resetting to maximum available disk space: 34515 mb
 WARN | Temporary Store limit is 51200 mb, whilst the temporary data directory: C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage only has 34515 mb of usable space - resetting to maximum available 34515 mb.
Sending text 'Important Task'
Consumer received Important Task
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-56230-1446467650870-0:1) is shutting down
 INFO | Connector tcp://127.0.0.1:61616 stopped
 INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage] stopped
 INFO | Stopping async queue tasks
 INFO | Stopping async topic tasks
 INFO | Stopped KahaDB
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-56230-1446467650870-0:1) uptime 1.928 seconds
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-56230-1446467650870-0:1) is shutdown
```

6. Multiple Consumers

The workload of message processing can be distributed among more than one consumer. When multiple receivers are attached to a queue, each message in the queue is delivered to one receiver. The absolute order of messages cannot be guaranteed, since one receiver may process messages faster than another.

Storage for queue is on the basis of first in, first out order (FIFO). One message is dispatched to a single consumer at a time. Only when that message has been consumed and acknowledged, it is deleted from the queue.
In the below example, we create multiple consumers, each one registered with a message listener. Next, we create a producer and make it send multiple messages. Each message is received by just one consumer and the order in which the messages are received is according to FIFO.

Each consumer will register its own message listener. It will pass a name to it so we know which consumer is consuming the message.

JmsMultipleCustomersMessageQueueExample:

```
05
06 import javax.jms.Connection;
07 import javax.jms.ConnectionFactory;
08 import javax.jms.Message;
09 import javax.jms.MessageConsumer;
10 import javax.jms.MessageProducer;
11 import javax.jms.Queue;
12 import javax.jms.Session;
13
14 import org.apache.activemq.ActiveMQConnectionFactory;
15 import org.apache.activemq.broker.BrokerFactory;
16 import org.apache.activemq.broker.BrokerService;
17
18 public class JmsMultipleCustomersMessageQueueExample {
19     public static void main(String[] args) throws URISyntaxException, Exception {
20         BrokerService broker = BrokerFactory.createBroker(new URI(
21                 "broker:(tcp://localhost:61616)"));
22         broker.start();
23         Connection connection = null;
24         try {
25             // Producer
26             ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(
27                     "tcp://localhost:61616");
28             connection = connectionFactory.createConnection();
29             Session session = connection.createSession(false,
30                     Session.AUTO_ACKNOWLEDGE);
31             Queue queue = session.createQueue("customerQueue");
32
33             // Consumer
34             for (int i = 0; i < 4; i++) {
35                 MessageConsumer consumer = session.createConsumer(queue);
36                 consumer.setMessageListener(new ConsumerMessageListener(
37                         "Consumer " + i));
38             }
39             connection.start();
40
41             String basePayload = "Important Task";
42             MessageProducer producer = session.createProducer(queue);
43             for (int i = 0; i < 10; i++) {
44                 String payload = basePayload + i;
45                 Message msg = session.createTextMessage(payload);
46                 System.out.println("Sending text '" + payload + "'");
47                 producer.send(msg);
48             }
49
50             Thread.sleep(1000);
51             session.close();
52         } finally {
53             if (connection != null) {
54                 connection.close();
55             }
56             broker.stop();
57         }
58     }
59
60 }
```

You can see from output, the messages are delivered in a round-robin fashion between all the message consumers.

Output:

```
01 INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-
   data\localhost\tmp_storage] started
02  INFO | Using Persistence Adapter:
   KahaDBPersistenceAdapter[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-
   data\localhost\KahaDB]
03  INFO | JMX consoles can connect to service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
04  INFO | KahaDB is version 6
05  INFO | Recovering from the journal @1:173161
06  INFO | Recovery replayed 1 operations from the journal in 0.012 seconds.
07  INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-62099-1446469937715-0:1) is starting
08  INFO | Listening for connections at: tcp://127.0.0.1:61616
09  INFO | Connector tcp://127.0.0.1:61616 started
10  INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-62099-1446469937715-0:1) started
11  INFO | For help or more information please see: http://activemq.apache.org
12  WARN | Store limit is 102400 mb (current store usage is 0 mb). The data directory:
   C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\KahaDB only has 34555 mb of
   usable space - resetting to maximum available disk space: 34556 mb
13  WARN | Temporary Store limit is 51200 mb, whilst the temporary data directory:
   C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage only has 34555 mb
   of usable space - resetting to maximum available 34555 mb.
14 Sending text 'Important Task0'
15 Consumer 0 received Important Task0
16 Sending text 'Important Task1'
17 Consumer 1 received Important Task1
18 Sending text 'Important Task2'
19 Consumer 2 received Important Task2
20 Sending text 'Important Task3'
21 Consumer 3 received Important Task3
22 Sending text 'Important Task4'
23 Consumer 0 received Important Task4
24 Sending text 'Important Task5'
25 Consumer 1 received Important Task5
26 Sending text 'Important Task6'
27 Consumer 2 received Important Task6
28 Sending text 'Important Task7'
29 Consumer 3 received Important Task7
30 Sending text 'Important Task8'
31 Consumer 0 received Important Task8
32 Sending text 'Important Task9'
33 Consumer 1 received Important Task9
34  INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-62099-1446469937715-0:1) is shutting
   down
35  INFO | Connector tcp://127.0.0.1:61616 stopped
36  INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-
   data\localhost\tmp_storage] stopped
```

```
seconds
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-62099-1446469937715-0:1) is shutdown
```

7. Creating a Temporary Queue

A temporary queue is a queue which can only be consumed by the JMS client that created it. It is created using the

```
createTemporaryQueue()
```

method on

```
QueueSession/code> object.
```

```
QueueSession.createTemporaryQueue();
```

8. Browsing a Queue

JMS allows you to peek ahead at pending messages on a Queue without actually consuming them using

```
QueueBrowser
```

object. Since we can browse through the messages without actually consuming them, this is very unique and important feature to point-to-point messaging.

We create the

```
QueueBrowser
```

object using the below statement on session object.

```
QueueBrowser browser = session.createBrowser(queue);
```

As you can see

```
createBrowser()
```

takes the

```
Queue
```

object that we are interested to browse.

To enumerate through the messages, we will call

```
QueueBrowser.getEnumeration()
```

.

```
Enumeration e = browser.getEnumeration();
while (e.hasMoreElements()) {
    TextMessage message = (TextMessage) e.nextElement();
    System.out.println("Get [" + message.getText() + "]");
}
```

When we are done with the browser we should close it.

```
QueueBrowser.close();
```

In the below example, we create a producer and post a bunch of messages to a queue. Next we create a consumer. In order to browse, we create a

```
QueueBrowser
```

object and navigate through the messages.

Finally, we call

```
consumer.receive()
```

to receive one of the messages from queue.

```
package com.javacodegeeks.jms;

import java.net.URI;
import java.net.URISyntaxException;
import java.util.Enumeration;

import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.jms.MessageProducer;
import javax.jms.Queue;
import javax.jms.QueueBrowser;
import javax.jms.Session;
import javax.jms.TextMessage;

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

```
        BrokerService broker = BrokerFactory.createBroker(new URI(
                "broker:(tcp://localhost:61616)"));
        broker.start();
        Connection connection = null;
        try {
            // Producer
            ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(
                    "tcp://localhost:61616");
            connection = connectionFactory.createConnection();
            Session session = connection.createSession(false,
                    Session.AUTO_ACKNOWLEDGE);
            Queue queue = session.createQueue("browseQueue");

            String basePayload = "A";
            MessageProducer producer = session.createProducer(queue);
            for (int i = 0; i < 4; i++) {
                String payload = basePayload + i;
                Message msg = session.createTextMessage(payload);
                System.out.println("Sending text '" + payload + "'");
                producer.send(msg);
            }

            MessageConsumer consumer = session.createConsumer(queue);
            connection.start();

            System.out.println("Browse through the elements in queue");
            QueueBrowser browser = session.createBrowser(queue);
            Enumeration e = browser.getEnumeration();
            while (e.hasMoreElements()) {
                TextMessage message = (TextMessage) e.nextElement();
                System.out.println("Get [" + message.getText() + "]");
            }
            System.out.println("Done");
            browser.close();

            TextMessage textMsg = (TextMessage) consumer.receive();
            System.out.println(textMsg);
            System.out.println("Received: " + textMsg.getText());
            session.close();
        } finally {
            if (connection != null) {
                connection.close();
            }
            broker.stop();
        }
    }

}
```

Messages obtained from a QueueBrowser are copies of messages contained in the queue and are not considered to be consumed as they are merely for browsing. Below is the output.

Output:

```
INFO | JMX consoles can connect to service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
 INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage] started
 INFO | Using Persistence Adapter: KahaDBPersistenceAdapter[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\KahaDB]
 INFO | KahaDB is version 6
 INFO | Recovering from the journal @1:260856
 INFO | Recovery replayed 1 operations from the journal in 0.012 seconds.
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-53401-1446474681874-0:1) is starting
 INFO | Listening for connections at: tcp://127.0.0.1:61616
 INFO | Connector tcp://127.0.0.1:61616 started
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-53401-1446474681874-0:1) started
 INFO | For help or more information please see: http://activemq.apache.org
 WARN | Store limit is 102400 mb (current store usage is 0 mb). The data directory: C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\KahaDB only has 34326 mb of usable space - resetting to maximum available disk space: 34327 mb
 WARN | Temporary Store limit is 51200 mb, whilst the temporary data directory: C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage only has 34326 mb of usable space - resetting to maximum available 34326 mb.
Sending text 'A0'
Sending text 'A1'
Sending text 'A2'
Sending text 'A3'
Browse through the elements in queue
Get [A0]
Get [A1]
Get [A2]
Get [A3]
Done
ActiveMQTextMessage {commandId = 5, responseRequired = true, messageId = ID:INMAA1-L1005-53401-1446474681874-3:1:1:1:1, originalDestination = null, originalTransactionId = null, producerId = ID:INMAA1-L1005-53401-1446474681874-3:1:1:1, destination = queue://browseQueue, transactionId = null, expiration = 0, timestamp = 1446474682340, arrival = 0, brokerInTime = 1446474682341, brokerOutTime = 1446474682383, correlationId = null, replyTo = null, persistent = true, type = null, priority = 4, groupID = null, groupSequence = 0, targetConsumerId = null, compressed = false, userID = null, content = org.apache.activemq.util.ByteSequence@ba8d91c, marshalledProperties = null, dataStructure = null, redeliveryCounter = 0, size = 0, properties = null, readOnlyProperties = true, readOnlyBody = true, droppable = false, jmsXGroupFirstForConsumer = false, text = A0}
Received: A0
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-53401-1446474681874-0:1) is shutting down
 INFO | Connector tcp://127.0.0.1:61616 stopped
 INFO | PListStore:[C:\javacodegeeks_ws\jmsMessageTypesExample\activemq-data\localhost\tmp_storage] stopped
 INFO | Stopping async queue tasks
 INFO | Stopping async topic tasks
 INFO | Stopped KahaDB
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-53401-1446474681874-0:1) uptime 1.446 seconds
 INFO | Apache ActiveMQ 5.12.0 (localhost, ID:INMAA1-L1005-53401-1446474681874-0:1) is shutdown
```

Download
You can download the full source code of this example here: **jmsMessageQueueExample.zip**

Leave a Reply

Be the First to Comment!

Start the discussion

KNOWLEDGE BASE

- Courses
- Minibooks
- News
- Resources
- Tutorials

THE CODE GEEKS NETWORK

- .NET Code Geeks
- Java Code Geeks
- System Code Geeks
- Web Code Geeks

HALL OF FAME

- Android Alert Dialog Example
- Android OnClickListener Example
- How to convert Character to String and a String to Character Array in Java
- Java Inheritance example
- Java write to File Example
- java.io.FileNotFoundException – How to solve File Not Found Exception
- java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception
- java.lang.NoClassDefFoundError – How to solve No Class Def Found Error
- JSON Example With Jersey + Jackson
- Spring JdbcTemplate Example

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused ultimate Java to Java developers resource center; targeted at the techn technical team lead (senior developer), project manager and junior deve JCGs serve the Java, SOA, Agile and Telecom communities with daily domain experts, articles, tutorials, reviews, announcements, code snipp source projects.

DISCLAIMER