* Go to configuration and open the standalone-full.xml file and edit it
* <subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">

<server name="default">

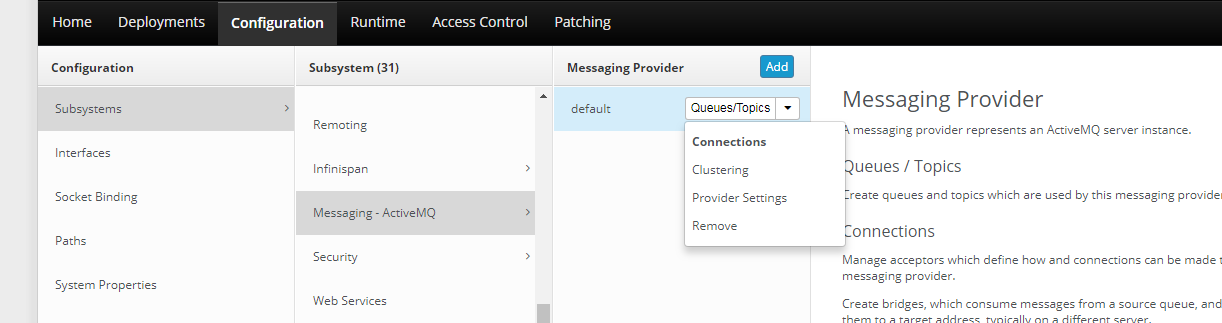
<security enabled="false" />

* ./standalone.bat --server-config=standalone-full.xml

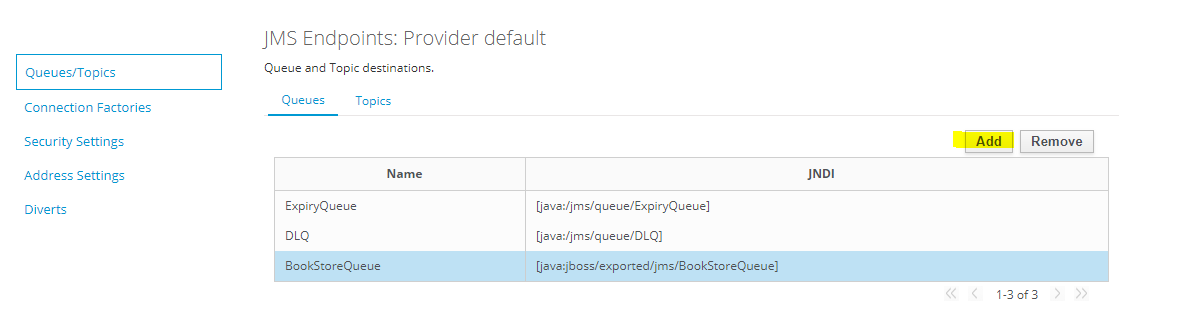
Make a project  
Add a jar fie i.e:jboss-client

How to add a message in JMS:

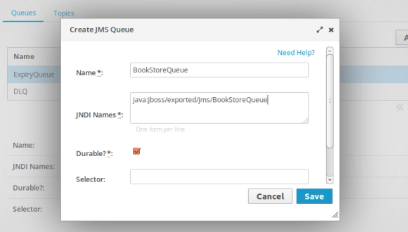
First go to configuration->subsystem->Messaginf-ActiveMq->default



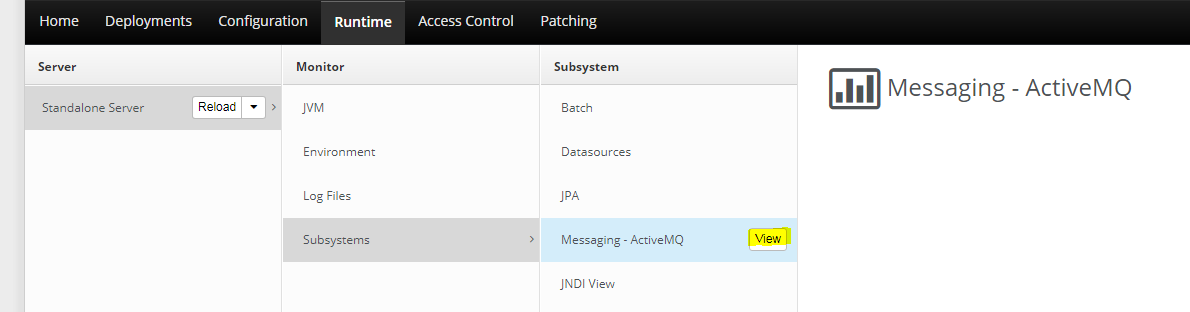
Click on queue topics:

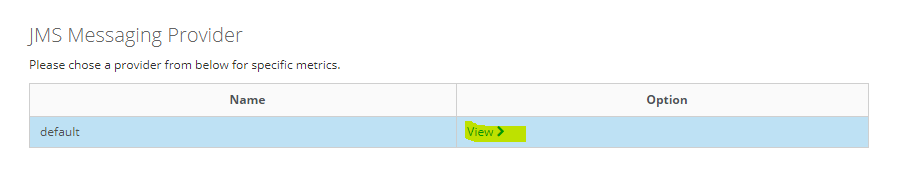


Click on add button and add one queue:

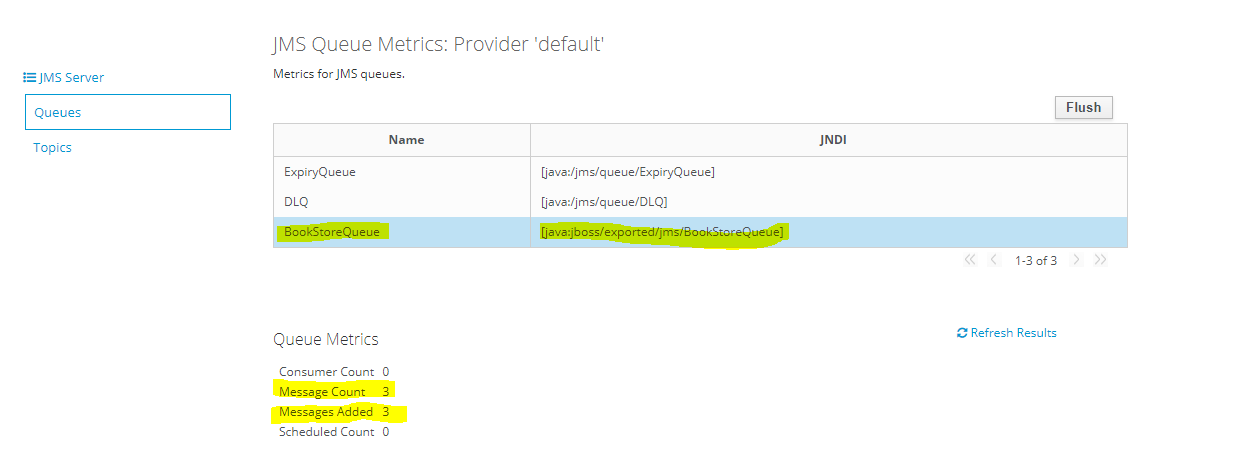


Go to runtime->StandaloneServer->Subsystem->Messaging-ActiveMQ->view





Then click on view:



So while running the below code three times so messages got added:

Create a java project:BookStorePOS

Inside create a class: SendTextMessage

**import java.util.Date;**

**import java.util.Properties;**

**import javax.jms.Connection;**

**import javax.jms.ConnectionFactory;**

**import javax.jms.JMSException;**

**import javax.jms.MessageProducer;**

**import javax.jms.Queue;**

**import javax.jms.Session;**

**import javax.jms.TextMessage;**

**import javax.naming.Context;**

**import javax.naming.InitialContext;**

**import javax.naming.NamingException;**

**public class SendTextMessage {**

**public static void main(String[] args) {**

**Properties jndiProperties = new Properties();**

**jndiProperties.put(Context.URL\_PKG\_PREFIXES, "org.jboss.ejb.client.naming");**

**jndiProperties.put("jboss.naming.client.ejb.context","true");**

**jndiProperties.put(Context.INITIAL\_CONTEXT\_FACTORY,"org.jboss.naming.remote.client.InitialContextFactory");**

**jndiProperties.put(Context.PROVIDER\_URL,"http-remoting://localhost:8080");**

**Connection connection = null;**

**try {**

**Context ctx = new InitialContext(jndiProperties);**

**Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");**

**ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");**

**connection = cf.createConnection();**

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

**MessageProducer messageProducer = session.createProducer(queue);**

**TextMessage message = session.createTextMessage("Hello World. The time is now " + new Date());**

**messageProducer.send(message);**

**} catch (NamingException e) {**

**e.printStackTrace();**

**} catch (JMSException e) {**

**e.printStackTrace();**

**}**

**finally {**

**try {**

**connection.close();**

**} catch (JMSException e) {**

**}**

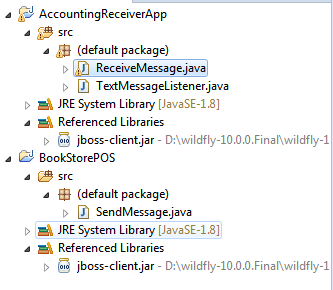
**}**

**}**

**}**

**2)Receiving a message():**

**Project structure:**



Now create a new project:AccountingReceiverApp

And add a jboss-client.jar in build path

This is for **TextMessage** only

* Create a class: ReceiveTextMessage

**import** java.util.Date;

**import** java.util.Properties;

**import** java.util.Scanner;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.JMSException;

**import** javax.jms.MessageConsumer;

**import** javax.jms.MessageProducer;

**import** javax.jms.Queue;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** javax.naming.Context;

**import** javax.naming.InitialContext;

**import** javax.naming.NamingException;

**public** **class** ReceiveTextMessage {

**private** **static** **boolean** lookForUserPressingX() {

Scanner scanner = **new** Scanner(System.***in***);

**return** scanner.nextLine().toLowerCase().equals("x");

}

**public** **static** **void** main(String[] args) {

Properties jndiProperties = **new** Properties();

jndiProperties.put(Context.***URL\_PKG\_PREFIXES***, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context","true");

jndiProperties.put(Context.***INITIAL\_CONTEXT\_FACTORY***,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.***PROVIDER\_URL***,"http-remoting://localhost:8080");

Connection connection = **null**;

**try** {

Context ctx = **new** InitialContext(jndiProperties);

Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

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

MessageConsumer consumer = session.createConsumer(queue);

consumer.setMessageListener(**new** TextMessageListener());

connection.start();

**boolean** finished = **false**;

**while** (!finished) {

finished = *lookForUserPressingX*();

}

} **catch** (NamingException e) {

e.printStackTrace();

} **catch** (JMSException e) {

e.printStackTrace();

}

**finally** {

**try** {

connection.close();

} **catch** (JMSException e) {

}

}

}

}

* Create a class: TextMessageListener

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.TextMessage;

public class TextMessageListener implements MessageListener {

@Override

public void onMessage(Message message) {

if (message instanceof TextMessage) {

TextMessage textMessage = (TextMessage) message;

String text;

System.out.println("aman");

try {

text = textMessage.getText();

System.out.println(text);

System.out.println("aman\_try");

} catch (JMSException e) {

throw new RuntimeException(e);

}

}

else {

System.out.println("Invalid message received.");

}

}

}

Run this ReceiverMessage class:

o/p: in console:

Jan 28, 2018 11:05:41 AM org.xnio.Xnio <clinit>

INFO: XNIO version 3.3.4.Final

Jan 28, 2018 11:05:41 AM org.xnio.nio.NioXnio <clinit>

INFO: XNIO NIO Implementation Version 3.3.4.Final

Jan 28, 2018 11:05:42 AM org.jboss.remoting3.EndpointImpl <clinit>

INFO: JBoss Remoting version 4.0.18.Final

Jan 28, 2018 11:05:43 AM org.jboss.ejb.client.remoting.VersionReceiver handleMessage

INFO: EJBCLIENT000017: Received server version 2 and marshalling strategies [river]

Jan 28, 2018 11:05:43 AM org.jboss.ejb.client.remoting.RemotingConnectionEJBReceiver associate

INFO: EJBCLIENT000013: Successful version handshake completed for receiver context EJBReceiverContext{clientContext=org.jboss.ejb.client.EJBClientContext@6e1ec318, receiver=Remoting connection EJB receiver [connection=Remoting connection <29a652d0>,channel=jboss.ejb,nodename=aman-pc]} on channel Channel ID cb945e15 (outbound) of Remoting connection 210366b4 to localhost/127.0.0.1:8080

Jan 28, 2018 11:05:43 AM org.jboss.ejb.client.EJBClient <clinit>

INFO: JBoss EJB Client version 2.1.4.Final

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

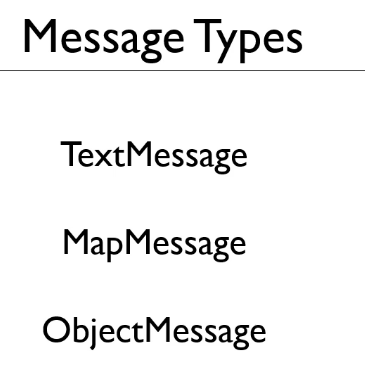
SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

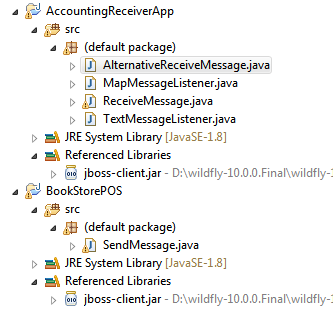
Hello World. The time is now Sun Jan 28 11:15:08 IST 2018

o/p in wildfly:





**3)Receiving a message(MapMessage):**



* Create a class SendMapMessage:

**import** java.util.Date;

**import** java.util.Properties;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.JMSException;

**import** javax.jms.MapMessage;

**import** javax.jms.MessageProducer;

**import** javax.jms.Queue;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** javax.naming.Context;

**import** javax.naming.InitialContext;

**import** javax.naming.NamingException;

**public** **class** SendMapMessage {

**public** **static** **void** main(String[] args) {

Properties jndiProperties = **new** Properties();

jndiProperties.put(Context.***URL\_PKG\_PREFIXES***, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context","true");

jndiProperties.put(Context.***INITIAL\_CONTEXT\_FACTORY***,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.***PROVIDER\_URL***,"http-remoting://localhost:8080");

Connection connection = **null**;

**try** {

Context ctx = **new** InitialContext(jndiProperties);

Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

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

MessageProducer messageProducer = session.createProducer(queue);

MapMessage message = session.createMapMessage();

message.setInt("sku", 10296);

message.setString("title", "Mastering Messaging");

message.setDouble("price",10.99);

message.setLong("date", **new** Date().getTime());

messageProducer.send(message);

} **catch** (NamingException e) {

e.printStackTrace();

} **catch** (JMSException e) {

e.printStackTrace();

}

**finally** {

**try** {

connection.close();

} **catch** (JMSException e) {

}

}

}

}

* Create a class:**ReceiveMapMessage**

import java.util.Date;

import java.util.Properties;

import java.util.Scanner;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.JMSException;

import javax.jms.MessageConsumer;

import javax.jms.MessageProducer;

import javax.jms.Queue;

import javax.jms.Session;

import javax.jms.TextMessage;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class RecieveMapMessage {

private static boolean lookForUserPressingX() {

Scanner scanner = new Scanner(System.in);

return scanner.nextLine().toLowerCase().equals("x");

}

public static void main(String[] args) {

Properties jndiProperties = new Properties();

jndiProperties.put(Context.URL\_PKG\_PREFIXES, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context","true");

jndiProperties.put(Context.INITIAL\_CONTEXT\_FACTORY,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.PROVIDER\_URL,"http-remoting://localhost:8080");

Connection connection = null;

try {

Context ctx = new InitialContext(jndiProperties);

Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

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

MessageConsumer consumer = session.createConsumer(queue);

consumer.setMessageListener(new MapMessageListener());

connection.start();

boolean finished = false;

while (!finished) {

finished = lookForUserPressingX();

}

} catch (NamingException e) {

e.printStackTrace();

} catch (JMSException e) {

e.printStackTrace();

}

finally {

try {

connection.close();

} catch (JMSException e) {

}

}

}

}

* Create a class: **MapMessageListener**

import java.util.Date;

import javax.jms.JMSException;

import javax.jms.MapMessage;

import javax.jms.Message;

import javax.jms.MessageListener;

public class MapMessageListener implements MessageListener {

public void onMessage(Message message) {

if (message instanceof MapMessage) {

MapMessage mapMessage = (MapMessage) message;

try {

String title = mapMessage.getString("title");

int sku = mapMessage.getInt("sku");

double price = mapMessage.getDouble("price");

long longDate = mapMessage.getLong("date");

Date date = new Date(longDate);

System.out.println("Sale of " + title + " (" + sku + ") at $" + price + " on " + date);

} catch (JMSException e) {

throw new RuntimeException(e);

}

}

else {

System.out.println("Invalid message received.");

}

}

}

Note:

* First run the ReceiveMessage.class this class will wait till any message in there in active-mq.
* Second run SendMessage.class this class will run and send the message on active mq.
* As stated in the first point that ReceiveMessage is currently running

So as soon as the message is sent on the queue it will retrieve it and display on the console

Lastly I don’t want Receive message always running I want if the message is not there

It should terminate and say no message is available:

So for this an alternative approach is there i.e

* Create a class: AlternativeReceiveMessageTextMapBasic

import java.util.Date;

import java.util.Properties;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.JMSException;

import javax.jms.MapMessage;

import javax.jms.MessageConsumer;

import javax.jms.Queue;

import javax.jms.Session;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class AlternativeReceiveMessage {

public static void main(String[] args) {

Properties jndiProperties = new Properties();

jndiProperties.put(Context.URL\_PKG\_PREFIXES, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context","true");

jndiProperties.put(Context.INITIAL\_CONTEXT\_FACTORY,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.PROVIDER\_URL,"http-remoting://localhost:8080");

Connection connection = null;

try {

Context ctx = new InitialContext(jndiProperties);

Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

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

MessageConsumer consumer = session.createConsumer(queue);

connection.start();

MapMessage mapMessage = (MapMessage) consumer.receive(100);

if (mapMessage != null ) {

String title = mapMessage.getString("title");

int sku = mapMessage.getInt("sku");

double price = mapMessage.getDouble("price");

long longDate = mapMessage.getLong("date");

Date date = new Date(longDate);

System.out.println("Sale of " + title + " (" + sku + ") at $" + price + " on " + date);

}

else {

System.out.println("There were no messages");

}

} catch (NamingException e) {

e.printStackTrace();

} catch (JMSException e) {

e.printStackTrace();

}

finally {

try {

connection.close();

} catch (JMSException e) {

}

}

}

}

o/p:

Jan 28, 2018 2:53:46 PM org.xnio.Xnio <clinit>

INFO: XNIO version 3.3.4.Final

Jan 28, 2018 2:53:46 PM org.xnio.nio.NioXnio <clinit>

INFO: XNIO NIO Implementation Version 3.3.4.Final

Jan 28, 2018 2:53:47 PM org.jboss.remoting3.EndpointImpl <clinit>

INFO: JBoss Remoting version 4.0.18.Final

Jan 28, 2018 2:53:48 PM org.jboss.ejb.client.remoting.VersionReceiver handleMessage

INFO: EJBCLIENT000017: Received server version 2 and marshalling strategies [river]

Jan 28, 2018 2:53:48 PM org.jboss.ejb.client.remoting.RemotingConnectionEJBReceiver associate

INFO: EJBCLIENT000013: Successful version handshake completed for receiver context EJBReceiverContext{clientContext=org.jboss.ejb.client.EJBClientContext@dc24521, receiver=Remoting connection EJB receiver [connection=Remoting connection <16f5448d>,channel=jboss.ejb,nodename=aman-pc]} on channel Channel ID b21e253d (outbound) of Remoting connection 67784306 to localhost/127.0.0.1:8080

Jan 28, 2018 2:53:48 PM org.jboss.ejb.client.EJBClient <clinit>

INFO: JBoss EJB Client version 2.1.4.Final

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.

Sale of Mastering Messaging (10296) at $10.99 on Sun Jan 28 14:53:58 IST 2018

**4) Tunning a message:**

As in the queue metrics:

Consumer count:0

Message Count:0

Message added:0

Scheduled count:0

* Send the two message so the queue metrics will be reflected:

By running SendMessage.class

queue metrics:

Consumer count:0

Message Count:2

Message added:2

Scheduled count:0

* Now restart the server the message count and message added will be unchanged because both the message are persisted.

queue metrics:

Consumer count:0

Message Count:2

Message added:2

Scheduled count:0

* Now receive the message message so the queue metrics will be reflected:

By running ReceiveMessage.class

queue metrics:

Consumer count:0

Message Count:0

Message added:2

Scheduled count:0

* Now again Send the one message so the queue metrics will be reflected:

By running SendMessage.class

queue metrics:

Consumer count:0

Message Count:1

Message added:3

Scheduled count:0

* Now restart the server you will find change in queue metrics:

i.e Message added == message count so message added is reseted

queue metrics:

Consumer count:0

Message Count:1

Message added:1

Scheduled count:0

Note:It is called persistence of message.

* Now again Send the one message so the queue metrics will be reflected:

By running SendMessage.class

But update

messageProducer.setDeliveryMode(DeliveryMode.***NON\_PERSISTENT***);

messageProducer.send(message);

queue metrics:

Consumer count:0

Message Count:2

Message added:2

Scheduled count:0

* Now restart the server you will find change in queue metrics:

i.e now u have added non-persistent message so this will not get stored   
as soon as you restarted the server you will get only the persisted content

queue metrics:

Consumer count:0

Message Count:1

Message added:1

Scheduled count:0

Note:It is called non-persistence of message

**4)a ) Priprities/timetolive:**

Change the class sendMessage.class

import java.util.Date;

import java.util.Properties;

import java.util.Random;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.DeliveryMode;

import javax.jms.JMSException;

import javax.jms.MapMessage;

import javax.jms.MessageProducer;

import javax.jms.Queue;

import javax.jms.Session;

import javax.jms.TextMessage;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class SendMessage {

public static void main(String[] args) {

Properties jndiProperties = new Properties();

jndiProperties.put(Context.URL\_PKG\_PREFIXES, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context","true");

jndiProperties.put(Context.INITIAL\_CONTEXT\_FACTORY,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.PROVIDER\_URL,"http-remoting://localhost:8080");

Connection connection = null;

try {

Context ctx = new InitialContext(jndiProperties);

Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

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

MessageProducer messageProducer = session.createProducer(queue);

Random random = new Random();

for (int i = 0; i < 20; i++) {

int priority = random.nextInt(9);

MapMessage message = session.createMapMessage();

message.setInt("sku", 10296);

message.setString("title", "Mastering Messaging with Priority " + priority);

message.setDouble("price",10.99);

message.setLong("date", new Date().getTime());

messageProducer.send(message, DeliveryMode.PERSISTENT,priority,10000);

}

} catch (NamingException e) {

e.printStackTrace();

} catch (JMSException e) {

e.printStackTrace();

}

finally {

try {

connection.close();

} catch (JMSException e) {

}

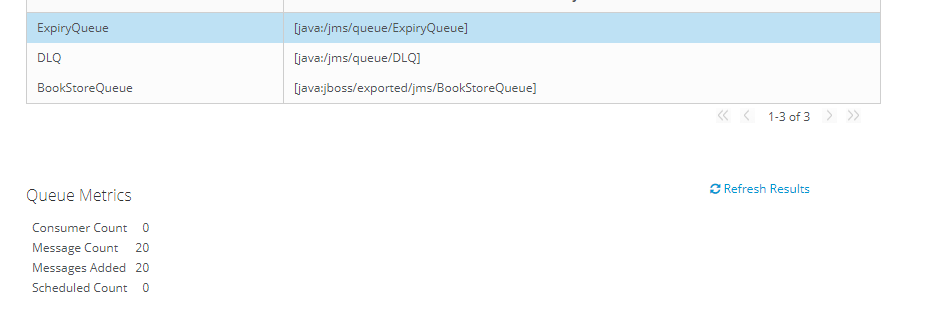
}

}

}

Now run both sendMessage for sending into the queue  
then run ReceiveMessage you will see the o/p in the console with the priorities

So as the message will expire Active-mq replaces the expired message in the expiry queue:



**5) Browsing queue:**

Create one project:BookStoreQueueMonitor:

Create a class->Monitor.java

**import** java.util.Enumeration;

**import** java.util.Properties;

**import** javax.jms.Connection;

**import** javax.jms.ConnectionFactory;

**import** javax.jms.JMSException;

**import** javax.jms.Message;

**import** javax.jms.Queue;

**import** javax.jms.QueueBrowser;

**import** javax.jms.Session;

**import** javax.naming.Context;

**import** javax.naming.InitialContext;

**public** **class** Monitor

{

**public** **static** **void** main(String[] args)

{

Connection connection = **null**;

Session session = **null**;

Properties jndiProperties = **new** Properties();

jndiProperties.put(Context.***URL\_PKG\_PREFIXES***, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context",**true**);

jndiProperties.put(Context.***INITIAL\_CONTEXT\_FACTORY***,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.***PROVIDER\_URL***,"http-remoting://localhost:8080");

**try**

{

Context ctx = **new** InitialContext(jndiProperties);

Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

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

connection.start();

QueueBrowser browser = session.createBrowser(queue);

Boolean finished = **false**;

**while** (!finished) {

Enumeration<Message> e = browser.getEnumeration();

**int** numMsgs =0;

**while** (e.hasMoreElements()) {

Message message = e.nextElement();

numMsgs++;

}

System.***out***.println("There are currently " + numMsgs + " messages in the queue.");

System.***out***.println("Press enter to refresh or x to exit");

String input = System.*console*().readLine();

**if** (input.toLowerCase().trim().equals("x")) {

finished = **true**;

}

}

}

**catch** (Exception e)

{

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

}

**finally** {

**try**

{

connection.close();

} **catch** (JMSException e)

{

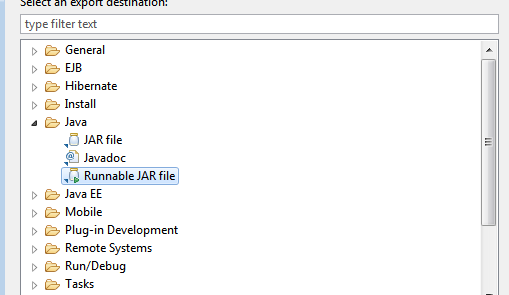
e.printStackTrace();

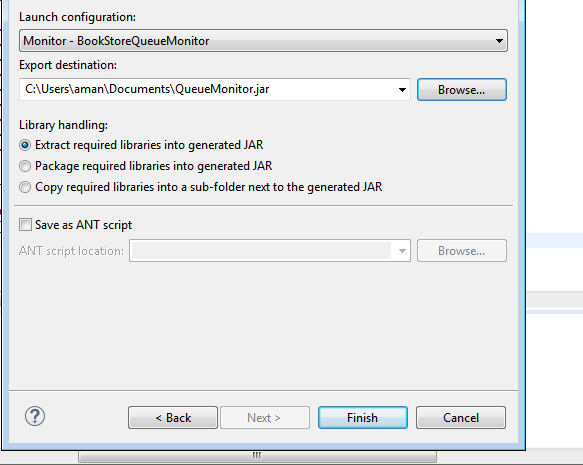
}

}

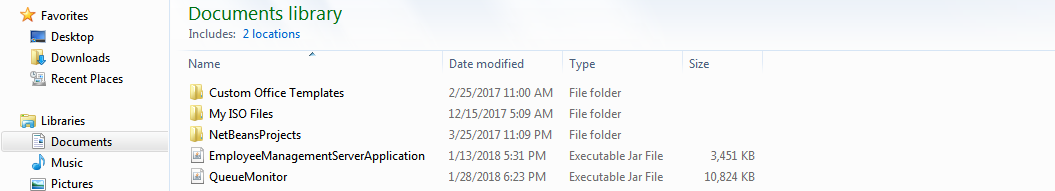
}}

Now export this jar file:



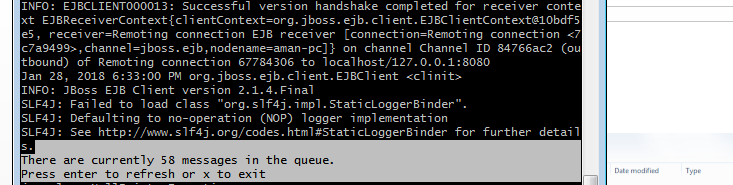


Now you can see the jar file will be saved in the documents foilder

i.e: 

so open gitbash/cmd and run the command: java –jar queueMonitor.jar

o/p:



So now if you want to read the expiry queue:

So we know we can only give a reuest if that file is exported,  
As in jms the Expiry queue is not exported,

So we can add one more jndi name to it and then can fetch

Open standalone.xml

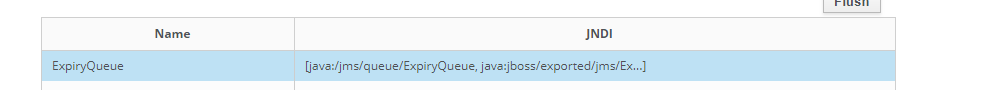
Search for ->expiry

line:324 <jms-queue name="ExpiryQueue" entries="java:/jms/queue/ExpiryQueue java:jboss/exported/jms/ExpiryQueue"/>

Now again run the server

U will fing the second jndi name:

Now u can see the messages of expiery queue:



Now in monitor.java

Just change one line:

i.e->Queue queue = (Queue)ctx.lookup("jms/ExpiryQueue");

then again run u will fing how many messages are there in expiryQueue:

o/p:

Jan 28, 2018 6:45:48 PM org.xnio.Xnio <clinit>

INFO: XNIO version 3.3.4.Final

Jan 28, 2018 6:45:49 PM org.xnio.nio.NioXnio <clinit>

INFO: XNIO NIO Implementation Version 3.3.4.Final

Jan 28, 2018 6:45:51 PM org.jboss.remoting3.EndpointImpl <clinit>

INFO: JBoss Remoting version 4.0.18.Final

Jan 28, 2018 6:45:53 PM org.jboss.ejb.client.remoting.VersionReceiver handleMessage

INFO: EJBCLIENT000017: Received server version 2 and marshalling strategies [river]

Jan 28, 2018 6:45:53 PM org.jboss.ejb.client.remoting.RemotingConnectionEJBReceiver associate

INFO: EJBCLIENT000013: Successful version handshake completed for receiver context EJBReceiverContext{clientContext=org.jboss.ejb.client.EJBClientContext@55d56113, receiver=Remoting connection EJB receiver [connection=Remoting connection <68953ed>,channel=jboss.ejb,nodename=aman-pc]} on channel Channel ID a8ff76aa (outbound) of Remoting connection 6c3708b3 to localhost/127.0.0.1:8080

Jan 28, 2018 6:45:54 PM org.jboss.ejb.client.EJBClient <clinit>

INFO: JBoss EJB Client version 2.1.4.Final

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

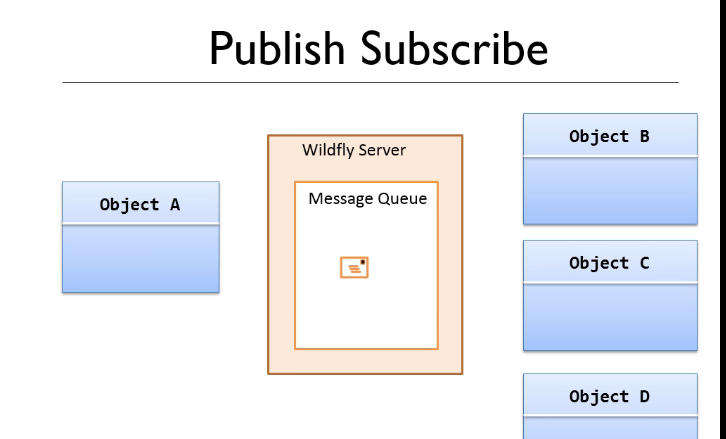
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

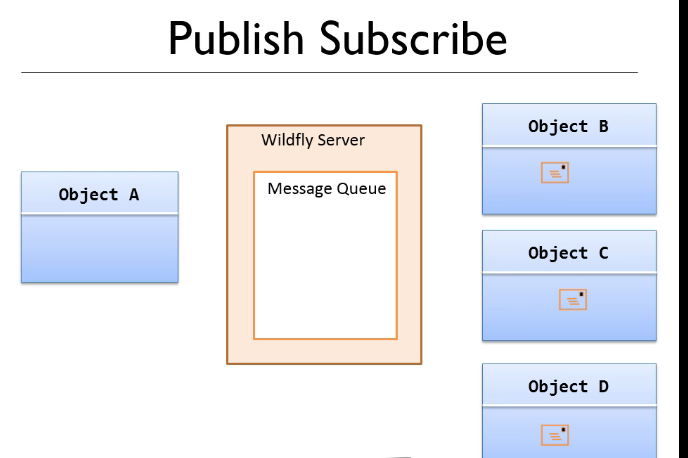
There are currently 40 messages in the queue.

Press enter to refresh or x to exit

**6)Publish-subscribe:**

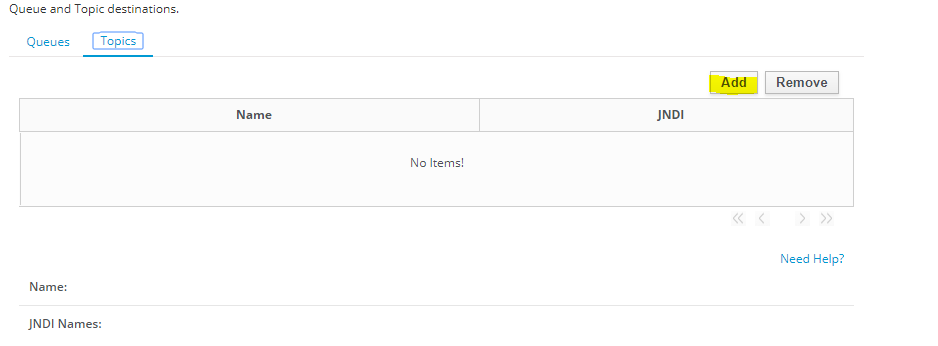
In this we can have more than one receiver and these multiple receiver will each receive a copy of the message that are delievered to the queue only when all the receiver are picked up the message will the message be removed.

Here we don’t call the destination on the server queue.Instead,we cal it a topic and technically the receiver should be called subscriber.



**Topic->It is simple a queue that can have multiple receiver required to it and each receiver will receive a copy of every message sent to that topic.**

**Got to Configuration->subsystem->Messaging-ActiveMQ->default->queueTopics**



**Change two lines of code in SendMessage and ReceiveMessage:**

**Topic topic = (Topic) ctx.lookup("jms/BookStoreTopic");**

**MessageProducer messageProducer = session.createProducer(topic);**

**Then run first receive message:**

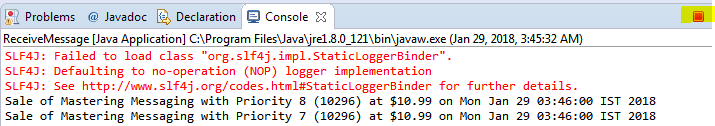
**And then sendMessage:**

**U will find the change in topic:**

|  |
| --- |
| **Topic Metrics** |
| |  |  | | --- | --- | | Delivering Count | 0 | | Durable Message Count | 0 | | Durable Subscription Count | 0 | | Message Count | 0 | | Messages Added | 0 | | Subscription Count | 1 | |

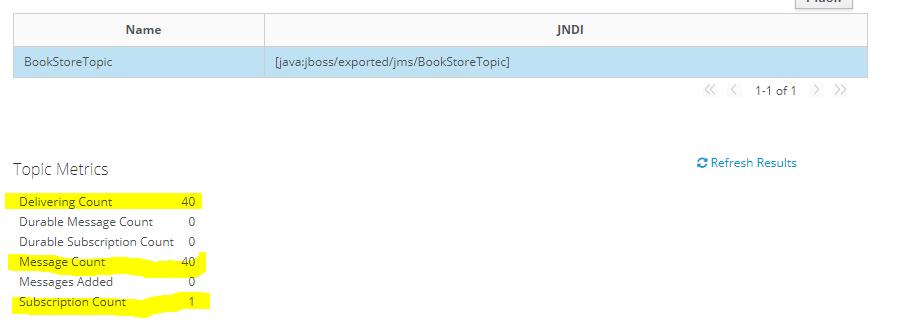
And all the 20 messages will be printed in the console.

Now do crash the receiver message code:



But you will find the subscriber count is still there in topic metrics

Now again run the sendMessage.class the topic metrics will be updated:



But now run the code ReceiverMessage again:

It will not receive anymessage and when you will refresh the result,all the unsubscribed message will be deleted.  
and a new subscriber count will be shown:



Default behavior Topic-> if a subscriber disconnects from a topic or it unsubscribes maybe because its crashed and its being rebooted then any messages received in the topic while the subscriber is not subscribed will never be received by that subscriber.It will only get the messages that arrive in the topic while it is subscribed.

To make the message durable and change the default behavior just change two lines in receiveMessage.class

i.e

connection.setClientID("accounting1");

MessageConsumer consumer = session.createDurableConsumer(topic, "accounting");

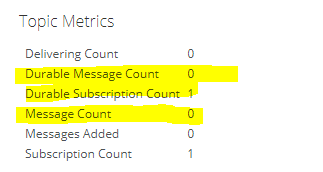


now first run :receiveMessage then run sendMessage you will find all the send messages will be received in the console.

Now just terminate the receiveMessage and again run the sendMessage.



Now run the receiveMessage:u will find all the meesages send will be received despite you were unsubscribed.



Code:

SendMessage.class

import java.util.Date;

import java.util.Properties;

import java.util.Random;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.DeliveryMode;

import javax.jms.JMSException;

import javax.jms.MapMessage;

import javax.jms.MessageProducer;

import javax.jms.Queue;

import javax.jms.Session;

import javax.jms.TextMessage;

import javax.jms.Topic;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class SendMessage {

public static void main(String[] args) {

Properties jndiProperties = new Properties();

jndiProperties.put(Context.URL\_PKG\_PREFIXES, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context","true");

jndiProperties.put(Context.INITIAL\_CONTEXT\_FACTORY,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.PROVIDER\_URL,"http-remoting://localhost:8080");

Connection connection = null;

try {

Context ctx = new InitialContext(jndiProperties);

//Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

Topic topic = (Topic) ctx.lookup("jms/BookStoreTopic");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

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

MessageProducer messageProducer = session.createProducer(topic);

Random random = new Random();

for (int i = 0; i < 20; i++) {

int priority = random.nextInt(9);

MapMessage message = session.createMapMessage();

message.setInt("sku", 10296);

message.setString("title", "Mastering Messaging with Priority " + priority);

message.setDouble("price",10.99);

message.setLong("date", new Date().getTime());

messageProducer.send(message, DeliveryMode.PERSISTENT,priority,0);

}

} catch (NamingException e) {

e.printStackTrace();

} catch (JMSException e) {

e.printStackTrace();

}

finally {

try {

connection.close();

} catch (JMSException e) {

}

}

}

}

ReceiveMessage.class:

import java.util.Date;

import java.util.Properties;

import java.util.Scanner;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.JMSException;

import javax.jms.MessageConsumer;

import javax.jms.MessageProducer;

import javax.jms.Queue;

import javax.jms.Session;

import javax.jms.TextMessage;

import javax.jms.Topic;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

public class ReceiveMessage {

private static boolean lookForUserPressingX() {

Scanner scanner = new Scanner(System.in);

return scanner.nextLine().toLowerCase().equals("x");

}

public static void main(String[] args) {

Properties jndiProperties = new Properties();

jndiProperties.put(Context.URL\_PKG\_PREFIXES, "org.jboss.ejb.client.naming");

jndiProperties.put("jboss.naming.client.ejb.context","true");

jndiProperties.put(Context.INITIAL\_CONTEXT\_FACTORY,"org.jboss.naming.remote.client.InitialContextFactory");

jndiProperties.put(Context.PROVIDER\_URL,"http-remoting://localhost:8080");

Connection connection = null;

try {

Context ctx = new InitialContext(jndiProperties);

//Queue queue = (Queue)ctx.lookup("jms/BookStoreQueue");

Topic topic = (Topic) ctx.lookup("jms/BookStoreTopic");

ConnectionFactory cf = (ConnectionFactory)ctx.lookup("jms/RemoteConnectionFactory");

connection = cf.createConnection();

connection.setClientID("accounting1");

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

//MessageConsumer consumer = session.createConsumer(topic);

MessageConsumer consumer = session.createDurableConsumer(topic, "accounting");

consumer.setMessageListener(new MapMessageListener());

connection.start();

boolean finished = false;

while (!finished) {

finished = lookForUserPressingX();

}

} catch (NamingException e) {

e.printStackTrace();

} catch (JMSException e) {

e.printStackTrace();

}

finally {

try {

connection.close();

} catch (JMSException e) {

}

}

}

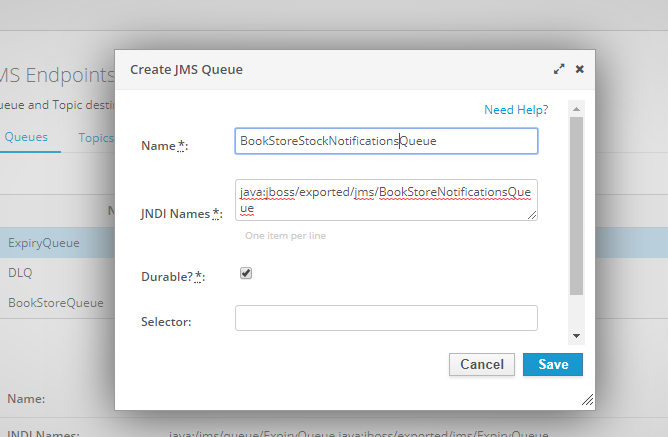
}

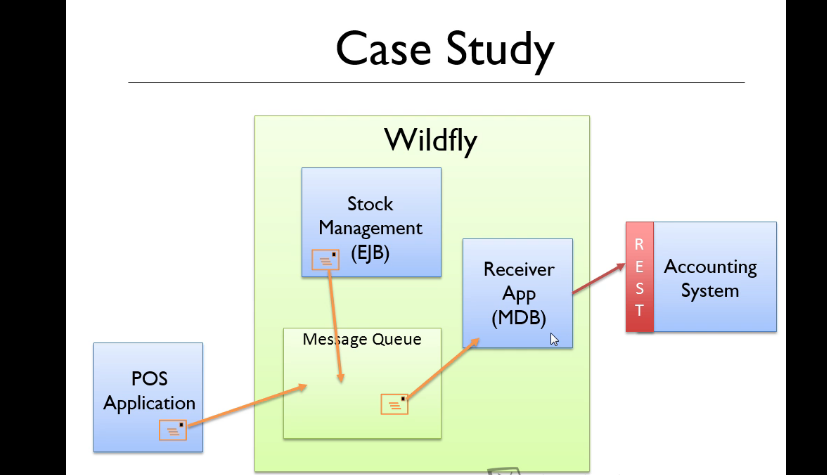
**Transactions-Rollbacks:**

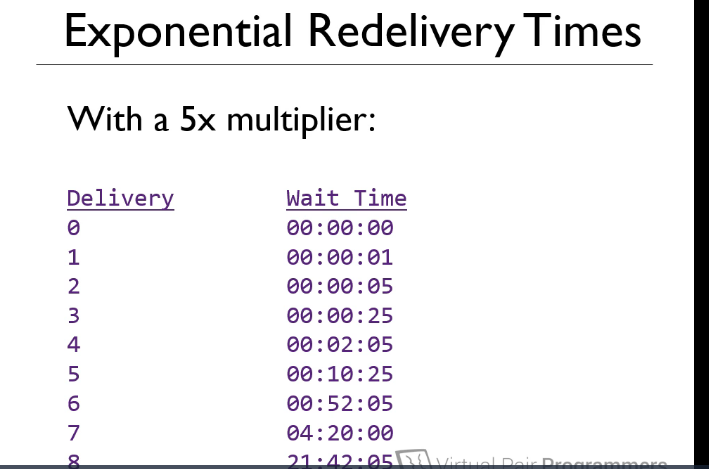
**EJBS:**

Create a project->BookStoreManagement

Add a queue in configuration->subsystem->queue







While Retrying:

Line number:312

We have to change three things in tags in standalone-full.xml

<address-setting name="#" message-counter-history-day-limit="10" page-size-bytes="2097152" max-size-bytes="10485760" max-delivery-attempts="8" redelivery-multiplier="5.0" redelivery-delay="1000" expiry-address="jms.queue.ExpiryQueue" dead-letter-address="jms.queue.DLQ"/>