JMS Tutorial

JMS (Java Message Service) is an API that provides the facility to create, send and read messages. It provides loosely coupled, reliable and asynchronous communication.

JMS is also known as a messaging service.

Understanding Messaging

Messaging is a technique to communicate applications or software components.

JMS is mainly used to send and receive message from one application to another.

Requirement of JMS

Generally, user sends message to application. But, if we want to send message from one application to another, we need to use JMS API.

Consider a scenario, one application A is running in INDIA and another application B is running in USA. To send message from A application to B, we need to use JMS.

Advantage of JMS

1) **Asynchronous:** To receive the message, client is not required to send request. Message will arrive automatically to the client.

2) **Reliable:** It provides assurance that message is delivered.

Messaging Domains

There are two types of messaging domains in JMS.

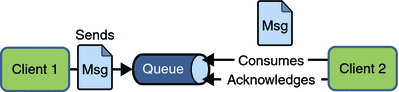
1. Point-to-Point Messaging Domain
2. Publisher/Subscriber Messaging Domain

1) Point-to-Point (PTP) Messaging Domain

In PTP model, one message is **delivered to one receiver** only. Here, **Queue** is used as a message oriented middleware (MOM).

The Queue is responsible to hold the message until receiver is ready.

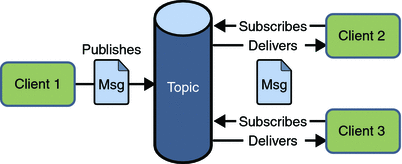
In PTP model, there is **no timing dependency** between sender and receiver.



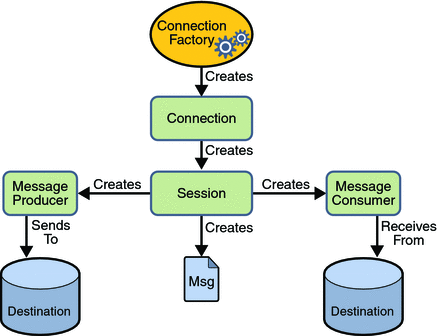
2) Publisher/Subscriber (Pub/Sub) Messaging Domain

In Pub/Sub model, one message is **delivered to all the subscribers**. It is like broadcasting. Here, **Topic** is used as a message oriented middleware that is responsible to hold and deliver messages.

In PTP model, there is **timing dependency** between publisher and subscriber.



JMS Programming Model



JMS Queue Example

To develop JMS queue example, you need to install any application server. Here, we are using **glassfish3** server where we are creating two JNDI.

1. Create connection factory named **myQueueConnectionFactory**
2. Create destination resource named **myQueue**

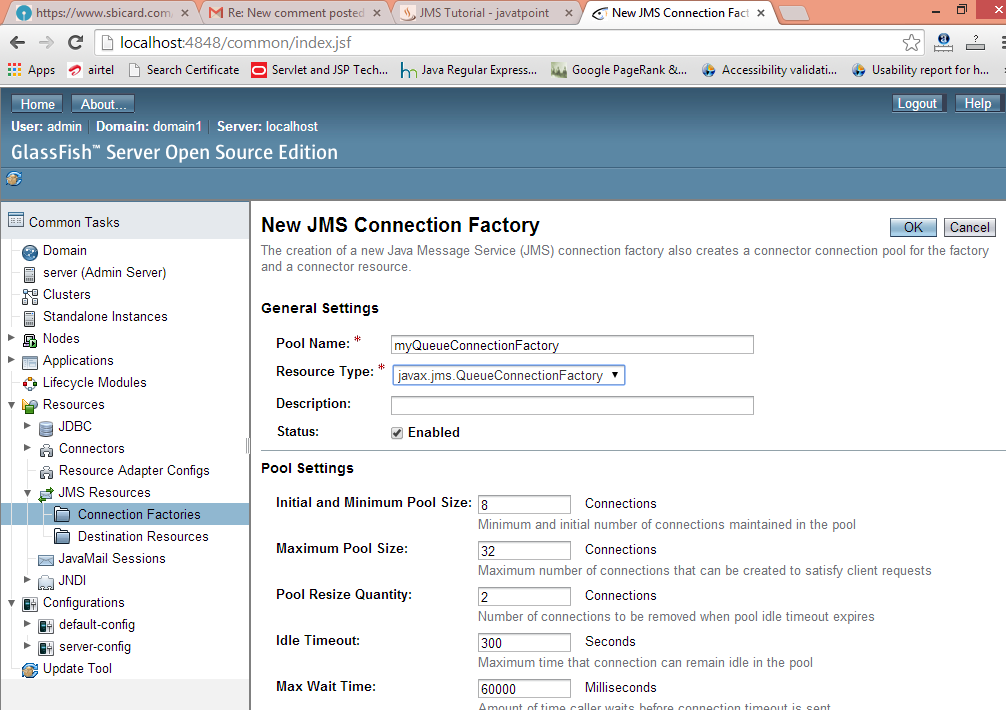
After creating JNDI, create server and receiver application. You need to run server and receiver in different console. Here, we are using eclipse IDE, it is opened in different console by default.

1) Create connection factory and destination resource

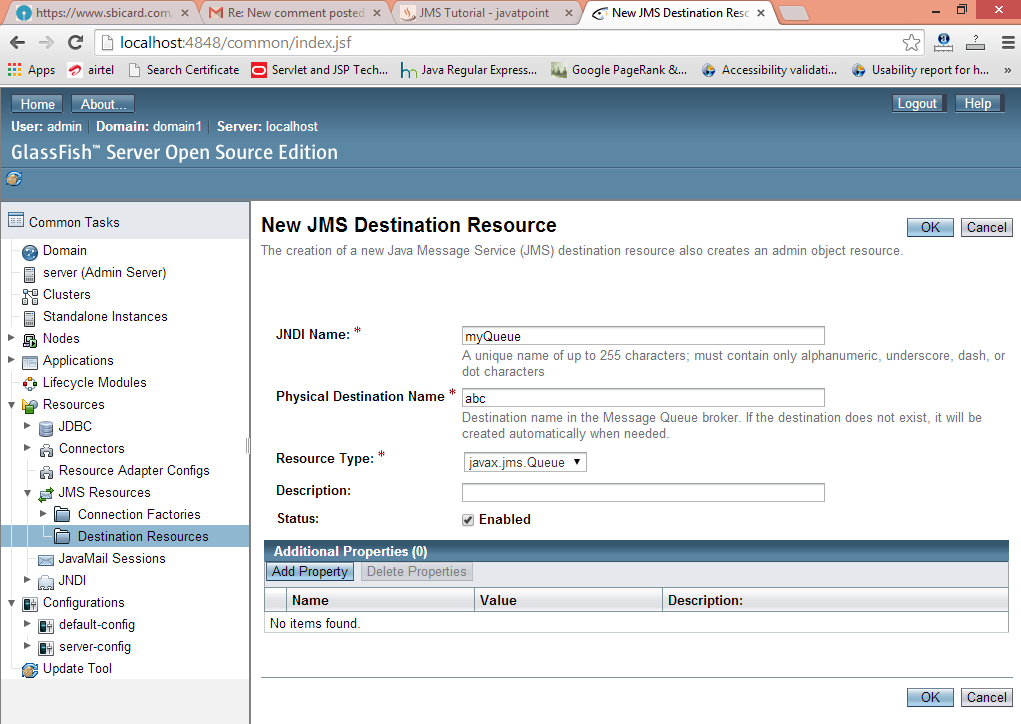
Open server admin console by the URL **http://localhost:4848**

Login with the username and password.

Click on the **JMS Resource -> Connection Factories -> New**, now write the pool name and select the Resource Type as QueueConnectionFactory then click on ok button.



Click on the **JMS Resource -> Destination Resources -> New**, now write the JNDI name and physical destination name then click on ok button.



2) Create sender and receiver application

Let's see the Sender and Receiver code. Note that Receiver is attached with listener which will be invoked when user sends message.

*File: MySender.java*

1. **import** java.io.BufferedReader;
2. **import** java.io.InputStreamReader;
3. **import** javax.naming.\*;
4. **import** javax.jms.\*;
6. **public** **class** MySender {
7. **public** **static** **void** main(String[] args) {
8. **try**
9. {   //Create and start connection
10. InitialContext ctx=**new** InitialContext();
11. QueueConnectionFactory f=(QueueConnectionFactory)ctx.lookup("myQueueConnectionFactory");
12. QueueConnection con=f.createQueueConnection();
13. con.start();
14. //2) create queue session
15. QueueSession ses=con.createQueueSession(**false**, Session.AUTO\_ACKNOWLEDGE);
16. //3) get the Queue object
17. Queue t=(Queue)ctx.lookup("myQueue");
18. //4)create QueueSender object
19. QueueSender sender=ses.createSender(t);
20. //5) create TextMessage object
21. TextMessage msg=ses.createTextMessage();
23. //6) write message
24. BufferedReader b=**new** BufferedReader(**new** InputStreamReader(System.in));
25. **while**(**true**)
26. {
27. System.out.println("Enter Msg, end to terminate:");
28. String s=b.readLine();
29. **if** (s.equals("end"))
30. **break**;
31. msg.setText(s);
32. //7) send message
33. sender.send(msg);
34. System.out.println("Message successfully sent.");
35. }
36. //8) connection close
37. con.close();
38. }**catch**(Exception e){System.out.println(e);}
39. }
40. }

*File: MyReceiver.java*

1. **import** javax.jms.\*;
2. **import** javax.naming.InitialContext;
4. **public** **class** MyReceiver {
5. **public** **static** **void** main(String[] args) {
6. **try**{
7. //1) Create and start connection
8. InitialContext ctx=**new** InitialContext();
9. QueueConnectionFactory f=(QueueConnectionFactory)ctx.lookup("myQueueConnectionFactory");
10. QueueConnection con=f.createQueueConnection();
11. con.start();
12. //2) create Queue session
13. QueueSession ses=con.createQueueSession(**false**, Session.AUTO\_ACKNOWLEDGE);
14. //3) get the Queue object
15. Queue t=(Queue)ctx.lookup("myQueue");
16. //4)create QueueReceiver
17. QueueReceiver receiver=ses.createReceiver(t);
19. //5) create listener object
20. MyListener listener=**new** MyListener();
22. //6) register the listener object with receiver
23. receiver.setMessageListener(listener);
25. System.out.println("Receiver1 is ready, waiting for messages...");
26. System.out.println("press Ctrl+c to shutdown...");
27. **while**(**true**){
28. Thread.sleep(1000);
29. }
30. }**catch**(Exception e){System.out.println(e);}
31. }
33. }

*File: MyListener.java*

1. **import** javax.jms.\*;
2. **public** **class** MyListener **implements** MessageListener {
4. **public** **void** onMessage(Message m) {
5. **try**{
6. TextMessage msg=(TextMessage)m;
8. System.out.println("following message is received:"+msg.getText());
9. }**catch**(JMSException e){System.out.println(e);}
10. }
11. }

Run the Receiver class first then Sender class.

JMS Topic Example

It is same as JMS Queue, but you need to change Queue to Topic, Sender to Publisher and Receiver to Subscriber.

You need to create 2 JNDI named **myTopicConnectionFactory** and **myTopic**.

*File: MySender.java*

1. **import** java.io.BufferedReader;
2. **import** java.io.InputStreamReader;
3. **import** javax.naming.\*;
4. **import** javax.jms.\*;
6. **public** **class** MySender {
7. **public** **static** **void** main(String[] args) {
8. **try**
9. {   //Create and start connection
10. InitialContext ctx=**new** InitialContext();
11. TopicConnectionFactory f=(TopicConnectionFactory)ctx.lookup("myTopicConnectionFactory");
12. TopicConnection con=f.createTopicConnection();
13. con.start();
14. //2) create queue session
15. TopicSession ses=con.createTopicSession(**false**, Session.AUTO\_ACKNOWLEDGE);
16. //3) get the Topic object
17. Topic t=(Topic)ctx.lookup("myTopic");
18. //4)create TopicPublisher object
19. TopicPublisher publisher=ses.createPublisher(t);
20. //5) create TextMessage object
21. TextMessage msg=ses.createTextMessage();
23. //6) write message
24. BufferedReader b=**new** BufferedReader(**new** InputStreamReader(System.in));
25. **while**(**true**)
26. {
27. System.out.println("Enter Msg, end to terminate:");
28. String s=b.readLine();
29. **if** (s.equals("end"))
30. **break**;
31. msg.setText(s);
32. //7) send message
33. publisher.publish(msg);
34. System.out.println("Message successfully sent.");
35. }
36. //8) connection close
37. con.close();
38. }**catch**(Exception e){System.out.println(e);}
39. }
40. }

*File: MyReceiver.java*

1. **import** javax.jms.\*;
2. **import** javax.naming.InitialContext;
4. **public** **class** MyReceiver {
5. **public** **static** **void** main(String[] args) {
6. **try** {
7. //1) Create and start connection
8. InitialContext ctx=**new** InitialContext();
9. TopicConnectionFactory f=(TopicConnectionFactory)ctx.lookup("myTopicConnectionFactory");
10. TopicConnection con=f.createTopicConnection();
11. con.start();
12. //2) create topic session
13. TopicSession ses=con.createTopicSession(**false**, Session.AUTO\_ACKNOWLEDGE);
14. //3) get the Topic object
15. Topic t=(Topic)ctx.lookup("myTopic");
16. //4)create TopicSubscriber
17. TopicSubscriber receiver=ses.createSubscriber(t);
19. //5) create listener object
20. MyListener listener=**new** MyListener();
22. //6) register the listener object with subscriber
23. receiver.setMessageListener(listener);
25. System.out.println("Subscriber1 is ready, waiting for messages...");
26. System.out.println("press Ctrl+c to shutdown...");
27. **while**(**true**){
28. Thread.sleep(1000);
29. }
30. }**catch**(Exception e){System.out.println(e);}
31. }
33. }

*File: MyListener.java*

1. **import** javax.jms.\*;
2. **public** **class** MyListener **implements** MessageListener {
4. **public** **void** onMessage(Message m) {
5. **try**{
6. TextMessage msg=(TextMessage)m;
8. System.out.println("following message is received:"+msg.getText());
9. }**catch**(JMSException e){System.out.println(e);}
10. }
11. }