**What is JMS**

An interface implemented by JEE containers to provide point-to-point publish/subscribe behavior

@Resource(name="jms/XATbiCF", type=javax.jms.ConnectionFactory.class)  
@MessageDriven(mappedName="jms/PITCreateCaseRequestQueue")

**What is MDB, message driven bean**

It is a class that imlements MessageListener

public class PitCreateCaseMDB implements MessageListener {

Instead of sending an email directly from an Enterprise JavaBean, the bean may choose to put the message onto a JMS queue to be handled by a Message-Driven Bean (another type of EJB) or another system in the enterprise.

**What type messaging is provided by JMS?**  
Both synchronous and asynchronous.

**How may messaging models do JMS provide?**JMS provides for two messaging models, publish-and-subscribe and point-to-point queuing.

**What is the point-to-point model in JMS?  PTP**  
A point-to-point model is based on the concept of a message queue: Senders send messages into the queue, and the receiver reads messages from this queue.

In the point-to-point model, several receivers can exist, attached to the same queue. However, (Message Oriented Middleware)MOM will deliver the message only to one of them. To which depends on the MOM implementation.

**What is the publish-and-subscribe model in JMS?**  
A publish-subscribe model is based on the message topic concept: Publishers send messages in a topic, and all subscribers of the given topic receive these messages.  
There is no guarantee consumer can receive the messages send by the publisher.

**How does a typical client perform the communication? Queue**

**1- Create Connection Factory**

A ConnectionFactory object is a JMS administered object and supports concurrent use.

General setting : JNDI name, pool name, Resouce type javax.jms.connectionFactory,   
 Pool setting : poolsize,IDLE time out, Transactionsuppot and etc  
 Additional properties. AddressList , user name password and etc

**2- Create Destination resources** modifies the associated admin object resources

**What is Producer and Consumer?**  
A1:   
Messaging lets a servlet delegate processing to a batch process either on the same machine or on a separate machine. The servlet creates a message and sends it to a queue. The servlet immediately completes and when the batch process is ready, it processes the message.   
Messaging is therefore comprised of three main components:   
A Producer creates messages and sends them to a Queue. The Producer could be something like a Servlet.   
A Queue stores the messages from the Produces and provides them to a Consumer when ready. The Queue is implemented by the messaging provider.   
A Consumer processes messages as they become available in the Queue. The Consumer is typically a bean implementing the MessageListener interface.

**Can JMS utilities automatically re-establish a connection if one side of the communication link (i.e. an application that's sending/receiving messages) goes down and is restarted? Are there APIs to help detect that the other side broke a connection (went down)?**  
Yes. You can write a snooper files to detect the service and restart the node upon node fail and a server instance fail.

**What is the Role of the JMS Provider?**  
The JMS provider handles security of the messages, data conversion and the client triggering. The JMS provider specifies the level of encryption and the security level of the message, the best data type for the non-JMS client

**Does JMS specification define transactions? Queue**  
JMS defines a transaction mechanisms allowing clients to send and receive groups of logically bounded messages as a single unit of information

A Session may be marked as transacted. It means that all messages sent in a session are considered as parts of a transaction. A set of messages can be committed (commit() method) or rolled back (rollback() method). If a provider supports distributed transactions, it's recommended to use XAResource API.

**What are the different types of messages available in the JMS API?**  
Message, TextMessage, BytesMessage, StreamMessage, ObjectMessage, MapMessage are the different messages available in the JMS API.

**What is the difference between topic and queue?**  
A topic is typically used for one to many messaging i.e. it supports publish subscribe model of messaging. While queue is used for one-to-one messaging i.e. it supports Point to Point Messaging.

Steps to produce the message :

**Definition :**

JMS : Java Message Service is an API that is part of Java EE for sending messages between two or more clients.  There are many JMS providers such as OpenMQ (glassfish’s default), HornetQ(Jboss), and ActiveMQ.

RabbitMQ: is an open source message broker software which uses the AMQP standard and is written by Erlang.

**Messaging Model:**

JMS supports two models: one to one and publish/subscriber. RabbitMQ supports the AMQP model which has 4 models : direct, fanout, topic, headers.

**Data types:**

JMS supports 5 different data types but RabbitMQ supports only the binary data type.

**Workflow strategy:**

In AMQP, producers send to the exchange then the queue, but in JMS, producers send to the queue or topic directly.

**Technology compatibility:**

JMS is specific for java users only, but RabbitMQ supports many technologies.

**Performance:**

If you would like to know more about their performance, [this benchmark](http://x-aeon.com/wp/2013/04/10/a-quick-message-queue-benchmark-activemq-rabbitmq-hornetq-qpid-apollo/) is a good place to start, but look for others as well.

Published at DZone with permission of its author, [Saeid Siavashi](http://java.dzone.com/users/phoenix87).