JMS

### **What is JMS?**

**Answer**  
JMS is an acronym used for Java Messaging Service. The Java Message Service defines a intra-business messaging API, designed to be easily and efficiently supported by a wide range of enterprise messaging products. Enterprises need to send and receive multiple messages. JMS supports this requirement by using asynchronous messaging.

### **Explain the Role of the JMS Provider?**

**Answer**  
JMS Provider is an implementation of the JMS interface for a Message Oriented Middleware (MOM). Providers are implemented as either a Java JMS implementation or an adapter to a non-Java MOM.  
Their functions are:

- Handling security of messages  
- Data Conversion   
- Client triggering  
- Specifying encryption level, security level, best data type for JMS Client.

### **What is the difference between Java Mail and JMS Queue?**

**Answer**  
JMS queue is a store that contains messages that have been sent and are waiting to be read. JMS is the ideal high-performance messaging platform, with full programmatic control over quality of service and delivery options.

JavaMail on the other hand provides lowest common denominator; slow, but human-readable messaging using infrastructure already available on virtually every computing platform.

### **What is the difference between JMS and RPC?**

**Answer**  
The basic difference between RPC and JMS lies in the way they message. RPC uses synchronous messaging while JMS uses asynchronous messaging approach.

In RPC the method invoker waits for the method to finish execution and return the control back to the invoker.  
In JMS the message sender just sends the message to the destination and continues it's own processing.

### **What are the advantages of JMS?**

**Answer**  
The characteristics of JMS are:

Asynchronous messaging:  
Due to asynchronous messaging, all the pieces don’t need to be up for the application to function as a whole.

Storage:  
MOM stores the messages on behalf of the receiver when it is down and then sends them once it is up.

### **Explain the use of Message object.**

**Answer**  
Message Object is a light weight entity that comprises of only header and properties. It does not comprise of payload. Thus, no data needs to be transferred when the receivers just need to be notified. Due to this using message becomes a very efficient way.

### **What is the use of BytesMessage?**

**Answer**  
BytesMessage contains an array of primitive bytes in it's payload. When the datatypes of two applications are not compatible with other message types,, they can send them using ByteMessage. JMS can be simply used as a transport medium with the payload being opaque to the JMS client.

However, it is important that the payload be read in the same order and type when it was created. Since there is no boundary line between the different data types stored. Thus you can even read a long as short. This would result in erroneous data.

### **Describe the use of StreamMessage?**

**Answer**  
StreamMessage carries a stream of Java primitive types as it's payload. Unlike ByteMessage, there are restrictions on the way the data is read to avoid erroneous output. StreamMessage prevents reading a long value as short, something that is allowed in case of BytesMessage. SreamMessage enforces a set of strict conversion rules which actually prevents reading of one primitive type as another.

### **What is the difference between Point to Point and Publish and Subscribe method?**

**Answer**  
In point to point communication, the producers and consumers of the message are defined. The producers push the messages in the queues and the receivers pull them.

In publish and subscribe method, the producers publish the message and the subscribers who have subscribed to that topic receive them. In this way, a message may be received, or processed, by multiple consumers.

### **Q. When to Remove messages from the queue ?**

##### When an application uses ConnectionConsumers, JMS might need to remove messages from the queue in a number of situations: 1) Badly formatted message A message might arrive that JMS cannot parse. 2) Poison message A message might reach the backout threshold, but the ConnectionConsumer fails to requeue it on the backout queue. 3)No interested ConnectionConsumer For point-to-point messaging, when the QueueConnectionFactory is set so that it does not retain unwanted messages, a message arrives that is unwanted by any of the ConnectionConsumers.

**Q. What is poison messages? And how to handle poison messages?**

Poison messages, messages the application can never successfully process.   
A badly-formatted message arrives on a queue. Such a message might make the receiving application fail and back out the receipt of the message. In this situation, such a message might be received, then returned to the queue, repeatedly. These messages are known as poison messages. The ConnectionConsumer must be able to detect poison messages and reroute them to an alternative destination.

### **What are the Message Headers in JMS message ?**

JMSDestination - send or publish method   
JMSDeliveryMode - send or publish method   
JMSExpiration - send or publish method   
JMSPriority - send or publish method   
JMSMessageID - send or publish method   
JMSTimestamp - send or publish method   
JMSCorrelationID - Client   
JMSReplyTo - Client   
JMSType - Client   
JMSRedelivered - JMS provider

**What are the steps to send and receive JMS message?**

Step 1.   
lookup the ConnectionFactory .   
(A connection factory is the object a client uses to create a connection to a provider. A connection factory encapsulates a set of connection configuration parameters that has been defined by an administrator. Each connection factory is an instance of the ConnectionFactory, QueueConnectionFactory, or TopicConnectionFactory interface)   
  
Context ctx = new InitialContext();   
ConnectionFactory connectionFactory = (ConnectionFactory)   
ctx.lookup("jms/ConnectionFactory");   
  
Step 2.   
lookup the destination (A destination is the object a client uses to specify the target of messages it produces and the source of messages it consumes.)   
  
Destination myDest = (Destination) ctx.lookup("jms/MyTopic");   
or   
Queue myQueue = (Queue) ctx.lookup("jms/MyQueue");   
  
  
Step 3.   
Create the Connection   
Connection connection = connectionFactory.createConnection();   
  
Step 4.   
Session session = connection.createSession(false,   
Session.AUTO\_ACKNOWLEDGE);   
  
The first argument means that the session is not transacted; the second means that the session automatically acknowledges messages when they have been received successfully   
  
Step 5.   
Create Message Producers   
  
You use a Session to create a MessageProducer for a destination. Here, the first example creates a producer for the destination myQueue, and the second for the destination myTopic:   
  
MessageProducer producer = session.createProducer (myQueue);   
or   
MessageProducer producer = session.createProducer(myTopic);   
  
Step 6.   
Create message and send message   
  
TextMessage message = session.createTextMessage();   
message.setText("testmsg");   
producer.send(message);   
  
Step 7.   
create Message Consumers and receive message   
MessageConsumer consumer = session.createConsumer(myQueue);   
or   
MessageConsumer consumer = session.createConsumer(myTopic);   
  
Message m = consumer.receive();   
if (m instanceof TextMessage) {   
TextMessage message = (TextMessage) m;   
System.out.println("Reading message: " + message.getText());   
} else {   
// Handle error   
}

**What is Publish/Subscribe Messaging in JMS ?**

Pub/sub messaging has the following characteristics :   
  
a) Each message can have multiple consumers.   
  
b)Publishers and subscribers have a timing dependency. A client that subscribes to a topic can consume only messages published after the client has created a subscription, and the subscriber must continue to be active in order for it to consume messages.

**What Point-to-Point Messaging in JMS ?**

) Each message has only one consumer.   
b) A sender and a receiver of a message have no timing dependencies. The receiver can fetch the message whether or not it was running when the client sent the message.   
c)The receiver acknowledges the successful processing of a message.

**What are the main component in JMS**?

1)A JMS provider is a messaging system that implements the JMS interfaces and provides administrative and control features.   
  
2)JMS clients are the programs or components, written in the Java programming language, that produce and consume messages. Any J2EE application component can act as a JMS client.   
  
3)Messages are the objects that communicate information between JMS clients.   
  
4)Administered objects are preconfigured JMS objects created by an administrator for the use of clients. The two kinds of JMS administered objects are destinations and connection factories

**How Does the JMS Work with the J2EE ?**

Application clients, Enterprise JavaBeans (EJB) components, and web components can send or synchronously receive a JMS message. Application clients can in addition receive JMS messages asynchronously. (Applets, however, are not required to support the JMS API.)   
  
Message-driven beans, which are a kind of enterprise bean, enable the asynchronous consumption of messages. A JMS provider can optionally implement concurrent processing of messages by message-driven beans.   
  
Message send and receive operations can participate in distributed transactions, which allow JMS operations and database accesses to take place within a single transaction.

**What Is Messaging?**

Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility: A messaging client can send messages to, and receive messages from, any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages.

### **Q. What is the difference between pub/sub and Point to Point i.e. p2p. ?**

pub/sub ? Publish/Subscriber   
Publish/Subscriber can have multiple Publisher who publish message and multiple Subscriber who subscribe the messages. This is like watching TV multiple channel and multiple users .   
One message can have more then one subscriber.   
All the publish messages have the header and subscriber consume the message based on the header.   
In JMS we say Topic.   
P2p ? Point to point   
One message have only one subscriber. Multiple publisher can send multiple messages to the Queue but each message has only one subscriber.   
In JMS we say Queue.

**What is the different between JMS and RPC?**

RPC : Remote procedure Call   
In RPC the method invoker waits for the method to finish execution and return the control back to the invoker. Thus it is completely synchronous in nature.   
  
Example : In a website click on any link ( country list link) and then website jsp call to database and get all the country list and shows in the screen.   
  
JMS : Java Messaging System   
While in JMS the message sender just sends the message to the destination and continues it's own processing. The sender does not wait for the receiver to respond. This is asynchronous behavior.   
  
Example : Send a postal letter in post office and doing other works. Not waiting for post to receive.

**What are the types of messaging?**

There are two TYPES of Messaging.   
Synchronous Messaging: Synchronous messaging involves a client that waits for the server to respond to a message.   
Example : Call to database and got data   
Asynchronous Messaging: Asynchronous messaging involves a client that does not wait for a message from the server. An event is used to trigger a message from a server.   
  
Example : JMS

**What are the various message types supported by JMS?**

Stream Messages = Group of Java Primitives   
Map Messages = Name Value Pairs. Name being a string& Value being a java primitive   
Text Messages = String messages (since being widely used a separate messaging Type has been supported)   
Object Messages = Group of serialize able java object   
Bytes Message = Stream of uninterrupted bytes

##### 