What is JMS?
A1:
Java Message Service: An interface implemented by most J2EE containers to provide point-to-point queueing and topic (publish/subscribe) behavior. JMS is frequently used by EJB's that need to start another process asynchronously.
For example, 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. This technique allows the EJB to return to handling requests immediately instead of waiting for a potentially lengthy process to complete.

A2:
The Java Message Service (JMS) defines the standard for reliable Enterprise Messaging. Enterprise messaging, often also referred to as Messaging Oriented Middleware (MOM), is universally recognized as an essential tool for building enterprise applications. By combining Java technology with enterprise messaging, the JMS API provides a powerful tool for solving enterprise computing problems.

A3:
JMS stands for Java Messaging Service which is developed by Sun Microsystems . JMS Provider allow applications which are running on different systems can communicate with each other asynchronously . Many EAI tools support JMS as their standard messaging service.

A4:
The Java Message Service is a Java API that allows applications to create, send, receive, and read

messages. Designed by Sun and several partner companies, the JMS API defines a common set of interfaces and associated semantics that allow programs written in the Java programming language to communicate with other messaging implementations.

A5;
JMS stands for Java Message Service. It allows applications to communicate through reliable, scaleable, and asynchronous text messages and objects over the network.

A6;
Java Message Service (JMS) is the new standard for interclient communication. It allows J2EE application components to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

A7;
JMS is an acronym used for Java Messaging Service. It is Java's answer to creating software using asynchronous messaging. It is one of the official specifications of the J2EE technologies and is a key technology.

Must I place all my class files in the WEB-INF folder and all JSP's outside?
A1:
The class files should place into WEB-INF/classes folder and the JSP files should place within a separate folder.

A2:
Yes! Otherwise the web server/ application server cannot access the .jsp files and classes.
The java class files can be placed either in WEB-INF/lib or WEB-INF/classes. But it is recommended to put the class files in WEB-INF/classes. The server will load the files from the classpath so it just will not matter where the class is.

A3:
Yes, class files is private resources, so you must store class in WEB-INF/classes folder.
JSP and HTML files should be placed outside.

A4:
Class files inside web-inf cannot be access by browsers, while the JSP files are meant for accessible by browsers so, it may be strictly place outside the web-inf only.

A5:
Here is structure of web app.
web (this folder is Accessible from www)
Store all your JSP and HTML files here
WEB-INF (this folder is not Accessible)
classes (store your classes here, classes you are using in jsp
lib (store 3rd party jars)

A6:

1. Class files - Either they must be in WEB-INF\classes directory OR you can package them as JAR and put in WEB-INF\lib
2. JSP files - Depends how do you design your arch. If you have controller/delegator that can forward requests to JSPs, you can keep them under WEB-INF directory also. If not, you have to keep them outside WEB-INF.

A7:
The Java Message Service (JMS) API is a messaging standard that allows application components based on the Java 2 Platform, Enterprise Edition (J2EE) to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous

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

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

What is the point-to-point model in JMS?
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 are the advantages of JMS?
One of the principal advantages of JMS messaging is that it's asynchronous. Thus not all the pieces need to be up all the time for the application to function as a whole.

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.

What is JMS administered object?
A preconfigured JMS object (a resource manager connection factory or a destination) created by an administrator for the use of JMS clients and placed in a JNDI namespace

What is publish/subscribe messaging?
With publish/subscribe message passing the sending application/client establishes a named topic in the JMS broker/server and publishes messages to this queue. The receiving clients register (specifically, subscribe) via the broker to messages by topic; every subscriber to a topic receives each message published to that topic. There is a one-to-many relationship between the publishing client and the subscribing clients.

What is the main parts of JMS applications?
The main parts of JMS applications are:
--ConnectionFactory and Destination
--Connection
--Session
--MessageProducer
--MessageConsumer
--Message

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.
Messaging enables distributed communication that is loosely coupled. A component sends a message to a destination, and the recipient can retrieve the message from the destination. However, the sender and the receiver do not have to be available at the same time in order to communicate. In fact, the sender does not need to know anything about the receiver; nor does the receiver need to know anything about the sender. The sender and the receiver need to know only what message format and what destination to use. In this respect, messaging differs from tightly coupled technologies, such as Remote Method Invocation (RMI), which require an application to know a remote application's methods.
Messaging also differs from electronic mail (e-mail), which is a method of communication between people or between software applications and people. Messaging is used for communication between software applications or software components.
Messaging is a mechanism by which data can be passed from one application to another application.

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.

What is the diffrence between Java Mail and JMS Queue
JMS is the ideal high-performance messaging platform for intrabusiness messaging, with full programmatic control over quality of service and delivery options.
JavaMail provides lowest common denominator, slow, but human-readable messaging using infrastructure already available on virtually every computing platform.

Does JMS specification define transactions? Queue
JMS specification 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 is synchronous messaging? Queue
Synchronous messaging involves a client that waits for the server to respond to a message. So if one end is down the entire communication will fail.

What is asynchronous messaging? Queue

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. So even if the client is down , the messaging will complete successfully.

How does a typical client perform the communication? Queue
1. Use JNDI to locate administrative objects.
2. Locate a single ConnectionFactory object.
3. Locate one or more Destination objects.
4. Use the ConnectionFactory to create a JMS Connection.

5. Use the Connection to create one or more Session(s).
6. Use a Session and the Destinations to create the MessageProducers and MessageConsumers needed.
7. Perform your communication.

What is JMS session? Queue
A single-threaded context for sending and receiving JMS messages. A JMS session can be nontransacted, locally transacted, or participating in a distributed transaction.

What is the use of JMS? In which situations we are using JMS? Can we send message from one server to another server using JMS? Queue
JMS is the ideal high-performance messaging platform for intrabusiness messaging, with full programmatic control over quality of service and delivery options.

What is the difference between durable and non-durable subscriptions?
Point-To-Point (PTP). This model allows exchanging messages via queues created for some purposes. A client can send and receive messages from one or several queues. PTP model is easier than pub/sub model.
A durable subscription gives a subscriber the freedom of receiving all messages from a topic, whereas a non-durable subscription doesn't make any guarantees about messages sent by others when a client was disconnected from a topic.

What is the difference between Message producer and Message consumer?
A1:
Messaging systems provide a host of powerful advantages over other conventional distributed computing models. Primarily, they encourage "loose coupling" between message consumers and message producers. There is a high degree of anonymity between producer and consumer: to the message consumer, it doesn't matter who produced the message, where the producer lives on the network, or when the message was produced.

A2:
In Publish/Subscribe model:
A publish/subscribe (pub/sub) messaging system supports an event driven model where information consumers and producers participate in the transmission of messages. Producers "publish" events, while consumers "subscribe" to events of interest, and consume the events. Producers associate messages with a specific topic, and the messaging system routes messages to consumers based on the topics the consumers register interest in.
In Point-To-Point model:
In point to point messaging systems, messages are routed to an individual consumer which maintains a queue of "incoming" messages. Messaging applications send messages to a specified queue, and clients retrieve messages from a queue.

A3:
In Point-To-Point model, one client can send message to the another client through the Destination. There is a guarantee to recieve the message whenever reciever is connected.
example:your telephone answering machine ,outer send a message to u,but you can recieve those msg whenever u coonected to answering machine.
In pub/sub model. one publisher,many no.of clients willbe there,publisher publish the message,subscriber or consumer can recieve those messages when he got subscription through the

topic.There is no guarantee consumer can recieve the messages send by the publisher.

What is JMS application ?
One or more JMS clients that exchange messages.

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

How JMS is different from RPC?
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. 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.

What Is the JMS API?
The Java Message Service is a Java API that allows applications to create, send, receive, and read messages. Designed by Sun and several partner companies, the JMS API defines a common set of interfaces and associated semantics that allow programs written in the Java programming language to communicate with other messaging implementations.
The JMS API minimizes the set of concepts a programmer must learn to use messaging products but provides enough features to support sophisticated messaging applications. It also strives to maximize the portability of JMS applications across JMS providers in the same messaging domain.
The JMS API enables communication that is not only loosely coupled but also
* Asynchronous. A JMS provider can deliver messages to a client as they arrive; a client does not have to request messages in order to receive them.
* Reliable. The JMS API can ensure that a message is delivered once and only once. Lower levels of reliability are available for applications that can afford to miss messages or to receive duplicate messages.
The JMS Specification was first published in August 1998. The latest version of the JMS Specification is Version 1.1, which was released in April 2002. You can download a copy of the Specification from the JMS Web site, http://java.sun.com/products/jms/.

What is JMS client ?
A Java language program that sends or receives messages.

Give an example of using the point-to-point model
The point-to-point model is used when the information is specific to a single client. For example, a client can send a message for a print out, and the server can send information back to this client after completion of the print job.

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.

A2:
A producer is the client application that plays the role of a message sender in JMS API.
A consumer is the client application that plays the role of a message receiver in JMS API.

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.

What is JMS provider?
A messaging system that implements the Java Message Service as well as other administrative and control functionality needed in a full-featured messaging product.

What is Byte Message ?
Byte Messages contains a Stream of uninterrupted bytes. Byte Message contains an array of primitive bytes in it's payload. Thus it can be used for transfer of data between two applications in their native format which may not be compatible with other Message types. It is also useful where JMS is used purely as a transport between two systems and the message payload is opaque to the JMS client.

What is the difference between Byte Message and Stream Message?
Bytes Message stores data in bytes. Thus the message is one contiguous stream of bytes. While the Stream Message maintains a boundary between the different data types stored because it also stores the type information along with the value of the primitive being stored. Bytes Message allows data to be read using any type. Thus even if your payload contains a long value, you can invoke a method to read a short and it will return you something. It will not give you a semantically correct data but the call will succeed in reading the first two bytes of data. This is strictly prohibited in the Stream Message. It maintains the type information of the data being stored and enforces strict conversion rules on the data being read.

What are the advantages of JMS?
JMS is asynchronous in nature. Thus not all the pieces need to be up all the time for the application to function as a whole. Even if the receiver is down the MOM will store the messages on it's behalf and will send them once it comes back up. Thus at least a part of application can still function as there is no blocking.

Are you aware of any major JMS products available in the market?
IBM's MQ Series is one of the most popular product used as Message Oriented Middleware. Some of the other products are SonicMQ, iBus etc. Weblogic application server also comes with built in support for JMS messaging.

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 are the different messaging paradigms JMS supports?
Publish and Subscribe i.e. pub/suc and Point to Point i.e. p2p.

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.

How Does the JMS API Work with the J2EE Platform?
When the JMS API was introduced in 1998, its most important purpose was to allow Java applications to access existing messaging-oriented middleware (MOM) systems, such as MQSeries from IBM. Since that time, many vendors have adopted and implemented the JMS API, so that a JMS product can now provide a complete messaging capability for an enterprise.
Since the 1.3 release of the J2EE platform ("the J2EE 1.3 platform"), the JMS API has been an integral part of the platform, and application developers can use messaging with components using J2EE APIs ("J2EE components").
The JMS API in the J2EE platform has the following features.
* 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 may optionally implement concurrent processing of messages by message-driven beans.
* Message sends and receives can participate in distributed transactions.

The JMS API enhances the J2EE platform by simplifying enterprise development, allowing loosely coupled, reliable, asynchronous interactions among J2EE components and legacy systems capable of messaging. A developer can easily add new behavior to a J2EE application with existing business events by adding a new message-driven bean to operate on specific business events. The J2EE platform's EJB container architecture, moreover, enhances the JMS API by providing support for distributed transactions and allowing for the concurrent consumption of messages.
Another J2EE platform technology, the J2EE Connector Architecture, provides tight integration between J2EE applications and existing Enterprise Information (EIS) systems. The JMS API, on the other hand, allows for a very loosely coupled interaction between J2EE applications and existing EIS systems.
At the 1.4 release of the J2EE platform, the JMS provider may be integrated with the application server using the J2EE Connector Architecture. You access the JMS provider through a resource adapter. For more information, see the Enterprise JavaBeans Specification, v2.1, and the J2EE Connector Architecture Specification, v1.5.

What is the role of JMS in enterprise solution development?
JMS is typically used in the following scenarios
1. Enterprise Application Integration: - Where a legacy application is integrated with a new application via messaging.
2. B2B or Business to Business: - Businesses can interact with each other via messaging because JMS allows organizations to cooperate without tightly coupling their business systems.
3. Geographically dispersed units: - JMS can ensure safe exchange of data amongst the geographically

dispersed units of an organization.
4. One to many applications: - The applications that have to push data in packet to huge number of clients in a one-to-many fashion are good candidates for the use JMS. Typical such applications are Auction Sites, Stock Quote Services etc.

What is the use of Message object?
Message is a light weight message having only header and properties and no payload. Thus if the received are to be notified abt an event, and no data needs to be exchanged then using Message can be very efficient.

What is the basic difference between Publish Subscribe model and P2P model?
Publish Subscribe model is typically used in one-to-many situation. It is unreliable but very fast. P2P model is used in one-to-one situation. It is highly reliable.

What is the use of BytesMessage?
BytesMessage contains an array of primitive bytes in it's payload. Thus it can be used for transfer of data between two applications in their native format which may not be compatible with other Message types. It is also useful where JMS is used purely as a transport between two systems and the message payload is opaque to the JMS client. Whenever you store any primitive type, it is converted into it's byte representation and then stored in the payload. 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 and hence it is advisable that the payload be read in the same order and using the same type in which it was created by the sender.

What is the use of StreamMessage?
StreamMessage carries a stream of Java primitive types as it's payload. It contains some conveient methods for reading the data stored in the payload. However StreamMessage prevents reading a long value as short, something that is allwed in case of BytesMessage. This is so because the StreamMessage also writes the type information alonwgith the value of the primitive type and enforces a set of strict conversion rules which actually prevents reading of one primitive type as another.

What is the use of TextMessage?
TextMessage contains instance of java.lang.String as it's payload. Thus it is very useful for exchanging textual data. It can also be used for exchanging complex character data such as an XML document.

Why do the JMS dbms_aqadm.add_subscriber and dbms_aqadm.remove_subscriber calls sometimes hang when there are concurrent enqueues or dequeues happening on the same queue to which these calls are issued?
Add_subscriber and remove_subscriber are administrative operations on a queue. Though AQ does not prevent applications from issuing administrative and operational calls concurrently, they are executed serially. Both add_subscriber and remove_subscriber will block until pending transactions that have enqueued or dequeued messages commit and release the resources they hold. It is expected that adding and removing subscribers will not be a frequent event. It will mostly be part of the setup for the application. The behavior you observe will be acceptable in most cases. The solution is to try to isolate the calls to add_subscriber and remove_subscriber at the setup or cleanup phase when there are no other operations happening on the queue. That will make sure that they will not stay blocked waiting for operational calls to release resources.

Why do the TopicSession.createDurableSubscriber and TopicSession.unubscribe calls raise JMSException with the message "ORA - 4020 - deadlock detected while trying to lock object"?

CreateDurableSubscriber and unsubscribe calls require exclusive access to the Topics. If there are pending JMS operations (send/publish/receive) on the same Topic before these calls are issued, the ORA - 4020 exception is raised.
There are two solutions to the problem:
1. Try to isolate the calls to createDurableSubscriber and unsubscribe at the setup or cleanup phase when there are no other JMS operations happening on the Topic. That will make sure that the required resources are not held by other JMS operational calls. Hence the error ORA - 4020 will not be raised.
2. Issue a TopicSession.commit call before calling createDurableSubscriber and unsubscribe call.

Why doesn't AQ_ADMINISTRATOR_ROLE or AQ_USER_ROLE always work for AQ applications using Java/JMS API?
In addition to granting the roles, you would also need to grant execute to the user on the following packages:
* grant execute on sys.dbms_aqin to <userid>
* grant execute on sys.dbms_aqjms to <userid>

Why do I get java.security.AccessControlException when using JMS MessageListeners from Java stored procedures inside Oracle8i JServer?
To use MessageListeners inside Oracle8i JServer, you can do one for the following
1. GRANT JAVASYSPRIV to <userid>

Call dbms_java.grant_permission ('JAVASYSPRIV', 'SYS:java.net.SocketPermission', '*', 'accept,connect,listen,resolve');

What is the use of ObjectMessage?
ObjectMessage contains a Serializable java object as it's payload. Thus it allows exchange of Java objects between applications. This in itself mandates that both the applications be Java applications. The consumer of the message must typecast the object received to it's appropriate type. Thus the consumer should before hand know the actual type of the object sent by the sender. Wrong type casting would result in ClassCastException. Moreover the class definition of the object set in the payload should be available on both the machine, the sender as well as the consumer. If the class definition is not available in the consumer machine, an attempt to type cast would result in ClassNotFoundException. Some of the MOMs might support dynamic loading of the desired class over the network, but the JMS specification does not mandate this behavior and would be a value added service if provided by your vendor. And relying on any such vendor specific functionality would hamper the portability of your application. Most of the time the class need to be put in the classpath of both, the sender and the consumer, manually by the developer.

What is the use of MapMessage?
A MapMessage carries name-value pair as it's payload. Thus it's payload is similar to the java.util.Properties object of Java. The values can be Java primitives or their wrappers.

What is the difference between BytesMessage and StreamMessage?
BytesMessage stores the primitive data types by converting them to their byte representation. Thus the message is one contiguous stream of bytes. While the StreamMessage maintains a boundary between the different data types stored because it also stores the type information along with the value of the primitive being stored. BytesMessage allows data to be read using any type. Thus even if your payload contains a long value, you can invoke a method to read a short and it will return you something. It will not give you a semantically correct data but the call will succeed in reading the first two bytes of data.

This is strictly prohibited in the StreamMessage. It maintains the type information of the data being stored and enforces strict conversion rules on the data being read.

What is object message ?
Object message contains a group of serializeable java object. So it allows exchange of Java objects between applications. sot both the applications must be Java applications.

What is text message?
Text messages contains String messages (since being widely used, a separate messaging Type has been supported) . It is useful for exchanging textual data and complex character data like XML.

What is Map message?
map message contains name value Pairs. The values can be of type primitives and its wrappers. The name is a string.

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

What are the three components of a Message ?
A1:
A jms message has three components
1. A header
2. Properties (Optional)
3. A body (Optional)

A2:
A JMS message consists of three parts:
Message header - For message identification. For example, the header is used to determine if a given message is appropriate for a "subscriber"
Properties - For application-specific, provider-specific, and optional header fields
Body - Holds the content of the message. Several formats are supported, including TextMessage, which wrap a simple String, that wrap arbitrary Java objects (which must be serializable). Other formats are supported as well.

What is the difference between queue and topic?

A1:
A connection is created between the client and the server from a connection factory. Connections can be shared by several threads. The user credentials are supplied at this level. It is probably common for a client application to share access to a single connection to the server (unless different security credentials are required for different destinations). A session is created from a connection. The session may only be used by one thread at one time. The user credentials are inherited from the parent connection.
It is probably common for each MessageProducer (TopicPublisher or QueueSender) or MessageConsumer (TopicSubscriber or QueueReceiver) to have their own session due to the threading restriction. You can look at it like a tree. At the top you have a connection factory, beneath this you have your connections and then beneath the connections you have sessions. The leaves of the tree are the JMS actors (MessageProducers and MessageConsumers).

A2:
Both work on 2 different comminication models. Queue is point-to-point and topic is publish-subscriber.

A3:
In queues, one message can be consumed by only one client. But in the topics, one message can be consumed by many clients. Both are separate domains in MOM.
Queue represent Point-To-Point domain and Topic represent Pub/Sub domain

A4:
A point-to-point (PTP) product or application is built around the concept of message queues, senders, and receivers. Each message is addressed to a specific queue, and receiving clients extract messages from the queue(s) established to hold their messages. Queues retain all messages sent to them until the messages are consumed or until the messages expire.
In a publish/subscribe (pub/sub) product or application, clients address messages to a topic. Publishers and subscribers are generally anonymous and may dynamically publish or subscribe to the content hierarchy. The system takes care of distributing the messages arriving from a topic's multiple publishers to its multiple subscribers. Topics retain messages only as long as it takes to distribute them to current subscribers.

What are the types of messaging?
There are two kinds of Messaging. Synchronous messaging involves a client that waits for the server to respond to a message. 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.

What is the difference between Point to Point and Publish/Subscribe
Point-to-point (P2P)
In point-to-point, messages are sent via queues. Messages are put onto the queues by the message producers (the clients). The message consumer is responsible for pulling the message from the queue. Point-to-point is typically used when a given message must be processed (received) only once by a given consumer. In this way, there is only one consumer of the given message.
Publish-and-subscribe (pub/sub)
In publish-and-subscribe, messages are sent through topics. Messages are published to topics by the message producers. The messages may be received by any consumers that subscribe to the given topic. In this way, a message may be received, or processed, by multiple consumers.

Why doesn't the JMS API provide end-to-end synchronous message delivery and notification of delivery?
Some messaging systems provide synchronous delivery to destinations as a mechanism for implementing reliable applications. Some systems provide clients with various forms of delivery notification so that the clients can detect dropped or ignored messages. This is not the model defined by the JMS API. JMS API messaging provides guaranteed delivery via the once-and-only-once delivery semantics of PERSISTENT messages. In addition, message consumers can insure reliable processing of messages by using either CLIENT_ACKNOWLEDGE mode or transacted sessions. This achieves reliable delivery with minimum synchronization and is the enterprise messaging model most vendors and developers prefer. The JMS API does not define a schema of systems messages (such as delivery notifications). If an application requires acknowledgment of message receipt, it can define an

application-level acknowledgment message.

What are the core JMS-related objects required for each JMS-enabled application?
Each JMS-enabled client must establish the following:
* A connection object provided by the JMS server (the message broker)
* Within a connection, one or more sessions, which provide a context for message sending and receiving
* Within a session, either a queue or topic object representing the destination (the message staging area) within the message broker
* Within a session, the appropriate sender or publisher or receiver or subscriber object (depending on whether the client is a message producer or consumer and uses a point-to-point or publish/subscribe strategy, respectively). Within a session, a message object (to send or to receive)

How does the Application server handle the JMS Connection?
1. App server creates the server session and stores them in a pool.
2. Connection consumer uses the server session to put messages in the session of the JMS.
3. Server session is the one that spawns the JMS session.
4. Applications written by Application programmers creates the message listener.

What is Stream Message ?
Stream messages are a group of java primitives. It contains some convenient methods for reading the data. However Stream Message prevents reading a long value as short. This is so because the Stream Message also writes the type information along with the value of the primitive type and enforces a set of strict conversion rules which actually prevents reading of one primitive type as another.