|  |
| --- |
| **1. JMS is:** |
| http://www.academictutorials.com/images/cross.gif  Java Multimedia Service |
| http://www.academictutorials.com/images/tick.jpg  Java Message Service |
| http://www.academictutorials.com/images/cross.gif  Java Message Server |
| http://www.academictutorials.com/images/cross.gif  Java Multimedia Server |
| **2. Java Message Service(JMS) specification v1.0.1 was released in:** |
| http://www.academictutorials.com/images/tick.jpg  October 1998 |
| http://www.academictutorials.com/images/cross.gif  October 1997 |
| http://www.academictutorials.com/images/cross.gif  October 1999 |
| http://www.academictutorials.com/images/cross.gif  December 1998 |
| **3. Java Message Service(JMS) specification v1.0.1 was released by \_\_\_\_\_\_\_\_ .** |
| http://www.academictutorials.com/images/cross.gif  Sun Microsystems |
| http://www.academictutorials.com/images/cross.gif  Javaloft |
| http://www.academictutorials.com/images/cross.gif  Micrsoft |
| http://www.academictutorials.com/images/tick.jpg  JavaSoft |
| **4. For sending messages between two or more clients the Java Message Service(JMS) API uses a Java's \_\_\_\_\_\_\_\_\_\_ .** |
| http://www.academictutorials.com/images/cross.gif  Message Open Middleware(MOM) |
| http://www.academictutorials.com/images/cross.gif  Multimedia Oriented Middleware(MOM) |
| http://www.academictutorials.com/images/tick.jpg  Message Oriented Middleware(MOM) |
| http://www.academictutorials.com/images/cross.gif  None of the above |
| **5. OMG is:** |
| http://www.academictutorials.com/images/tick.jpg  Object Management Group |
| http://www.academictutorials.com/images/cross.gif  Open Management Group |
| http://www.academictutorials.com/images/cross.gif  Organizational Management Group |
| http://www.academictutorials.com/images/cross.gif  None of the above |
| **6. Messages that contain XML pages are supported by JMS API's.** |
| http://www.academictutorials.com/images/cross.gif  Not always |
| http://www.academictutorials.com/images/cross.gif  False |
| http://www.academictutorials.com/images/tick.jpg  True |
| http://www.academictutorials.com/images/cross.gif  None of the above |
| **7. JMS supports messages that contain \_\_\_\_\_\_\_\_\_ Java objects** |
| http://www.academictutorials.com/images/cross.gif  nonserialized |
| http://www.academictutorials.com/images/tick.jpg  serialized |
| http://www.academictutorials.com/images/cross.gif  dynamic |
| http://www.academictutorials.com/images/cross.gif  A and C are true |
| **8. JMS is purely written in Java.** |
| http://www.academictutorials.com/images/tick.jpg  True |
| http://www.academictutorials.com/images/cross.gif  False |
| http://www.academictutorials.com/images/cross.gif  Can't say |
| http://www.academictutorials.com/images/cross.gif  None of the above |
| **9. You'll need the \_\_\_\_\_\_\_\_\_ to compile the programs.** |
| http://www.academictutorials.com/images/cross.gif  Java interpreter |
| http://www.academictutorials.com/images/cross.gif  Java browser |
| http://www.academictutorials.com/images/tick.jpg  Java compiler |
| http://www.academictutorials.com/images/cross.gif  Java editor |
| **10. Which of these are not JMS elements ?** |
| http://www.academictutorials.com/images/cross.gif  JMS provider |
| http://www.academictutorials.com/images/cross.gif  JMS client |
| http://www.academictutorials.com/images/cross.gif  JMS consumer |
| http://www.academictutorials.com/images/tick.jpg  JMS producer |
| **11. Which of these are not JMS API"s ?** |
| http://www.academictutorials.com/images/cross.gif  ConnectionFactory interfac |
| http://www.academictutorials.com/images/tick.jpg  Source interface |
| http://www.academictutorials.com/images/cross.gif  MessageConsumer interface |
| http://www.academictutorials.com/images/cross.gif  MessageProducer interface |
| **12. The JMS API are provided in the Java package \_\_\_\_\_\_\_\_ package.** |
| http://www.academictutorials.com/images/cross.gif  system.jms |
| http://www.academictutorials.com/images/cross.gif  java.jms |
| http://www.academictutorials.com/images/tick.jpg  javax.jms |
| http://www.academictutorials.com/images/cross.gif  jms.\* |
| **13. \_\_\_\_\_\_\_\_\_\_ a common software construct, are used to verify the continual operation of a specific component or service.** |
| http://www.academictutorials.com/images/tick.jpg  Heartbeats |
| http://www.academictutorials.com/images/cross.gif  Heartbeats sounds |
| http://www.academictutorials.com/images/cross.gif  Heart |
| http://www.academictutorials.com/images/cross.gif  None of the above |
| **14. Which of these are not the JMS Weblogic features ?** |
| http://www.academictutorials.com/images/cross.gif  Provides a unified and single messaging API. |
| http://www.academictutorials.com/images/cross.gif  Supports clustering. |
| http://www.academictutorials.com/images/cross.gif  Supports messages containing Extensible Markup Language(XML). |
| http://www.academictutorials.com/images/tick.jpg |
| **15. Which of these are not major components JMS Weblogic ?** |
| http://www.academictutorials.com/images/cross.gif  WebLogic JMS servers implementing the messaging facility. |
| http://www.academictutorials.com/images/tick.jpg  Server applications. |
| http://www.academictutorials.com/images/cross.gif  The Java Naming and Directory Interface(JNDI), which provides a server lookup facility. |
| http://www.academictutorials.com/images/cross.gif  Backing stores (file or database) for storing persistent data |
| **16. JMS-DOF is means:** |
| http://www.academictutorials.com/images/tick.jpg  JMS Distributed Object Framework |
| http://www.academictutorials.com/images/cross.gif  JMS Developed Object Framework |
| http://www.academictutorials.com/images/cross.gif  JMS Data Object Framework |
| http://www.academictutorials.com/images/cross.gif  None of the above |
| **17. Which are JMS messaging models? A. Publish-and-subscribe B. Point-to-point C. Store-and-forward D. Peer-to-peer** |
| http://www.academictutorials.com/images/cross.gif  Only A |
| http://www.academictutorials.com/images/tick.jpg  A and B |
| http://www.academictutorials.com/images/cross.gif  Only B |
| http://www.academictutorials.com/images/cross.gif  A and C |
| **18. Which of the following are JMS administered objects ? A. Destination B. Connection C. ConnectionFactory D. Session** |
| http://www.academictutorials.com/images/tick.jpg  A,C |
| http://www.academictutorials.com/images/cross.gif  A,B |
| http://www.academictutorials.com/images/cross.gif  A,D |
| http://www.academictutorials.com/images/cross.gif  A,C,D |
| **19. Which following services are NOT provided by JMS? A. Clustering: Load balancing and fail-over B. Asynchronous message delivery C. Error notifications D. Security** |
| http://www.academictutorials.com/images/cross.gif  Only A and C |
| http://www.academictutorials.com/images/tick.jpg  A, C, D |
| http://www.academictutorials.com/images/cross.gif  A,B,C |
| http://www.academictutorials.com/images/cross.gif  A,B,D |
| **20. A JMS message is composed of which of the following parts: A. Header B. Properties C. Footer D. Body** |
| http://www.academictutorials.com/images/tick.jpg  A, B, D |
| http://www.academictutorials.com/images/cross.gif  Only A and B |
| http://www.academictutorials.com/images/cross.gif  All of the above |
| http://www.academictutorials.com/images/cross.gif  None of these |

**1) What is JMS?**

JMS means Java Messaging Service.  It is the new standard for inter client communication. It allows the J2EE application component to create, send, read and receive the messages.

**2) What type of messaging is provided by JMS?**

JMS provides both type of messaging,

* synchronous
* Asynchronous

**3) What do you mean by Synchronous and Asynchronous type of messaging?**

Synchronous: In this type of messaging, client waits for the server to respond to a message. Ex: Telephone call, two way radio communication.

Asynchronous: In this type of messaging, client does not wait for a message from the server, but automatically an event is created to trigger a message from a server. Ex: email, text messaging, blog posting.

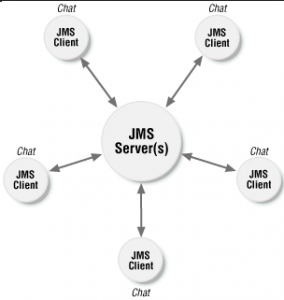
**4) How many types of messaging model do JMS provide for and what are they?**

There are two types of messaging models that JMS provides –

* Point to point queuing
* Second one is publish and subscribe

**5) Explain the difference between topic and queue?**

Queue technique is used for one to one messaging, and it supports point to point messaging. While topic is typically used for one to many messaging and it supports public subscribe model of messaging.

*[](https://career.guru99.com/wp-content/uploads/2013/10/jms.png)*

*Java Message Service*

**6) What is the role of the JMS provider?**

The JMS provider handles data conversion, security of the messages and the client triggering.  It specifies the level of encryption, security level of the message and the best-data type for the non-JMS client.

**7) What are the components of JMS?**

* JMS provider
* JMS client
* Messages
* Administered objects
* Native clients

**8) Give an example of using point to point model in JMS?**

Example for point to point model, would be a print out. When you select a print-out option, your system sends the message to the server, and once the print-out is taken out, again this server will send the message back to you. Point to point model is used, when the information is specific to a single client.

**9) For JMS-enabled application, what are the core JMS-related objects required?**

* The core JMS-related objects that are required are –
* The connection object
* One or more sessions within a connection that provides a context for message sending and receiving.
* A topic or queue object within a session representing the destination within the message broker.
* Appropriate sender or publisher or receiver within a session.

**10) What is JMS administered object?**

JMS administered object is a pre-configured JMS object that is created by an administrator for the use of JMS clients and placed in JNDI namespace.

**11) What is the important part of JMS applications?**

* Session
* Connection
* Message
* Message Producer
* Message Consumer
* Connection factory and destination

**12) What is JMS session?**

A JMS session is a single-threaded context for sending and receiving JMS messages.  A JMS session could be a locally transacted, non-transacted or distributed transacted.

**13) Mention the difference between durable and non-durable subscription?**

Durable subscription gives a subscriber the freedom of receiving all messages from a topic, while a non-durable subscription does not make any guarantees about messages sent by others when a client get disconnected by others.

**14)  What is Byte Message?**

Byte message is a stream of uninterrupted bytes. It contains an array of primitive bytes in its payload.  For the transfer of data between two applications in their native format, byte message is used, which may be not possible with other message types.

**15)  Mention different types of messages available in JMS API?**

The different types of messages available in JMS API are Message, TextMessage, BytesMessage, ObjectMessage and MapMessage.

**16) What is the difference between the P2P (Peer to Peer) model and subscribe model?**

P2P model is highly reliable and it is used in a one-to-one situation, while subscribe model is used in one-to-many situation. It is very fast but less reliable.

**17) What is a JMS client?**

JMS client is a language program that sends or receives messages.

**18) Can we send e-mail messages using JMS?**

JMS has no inherent support for email operations.

**19) Explain how Application server handles the JMS Connection?**

* With the help of Application server, the server session is created and it stores them in a pool
* To put messages in JMS session, connection consumer, uses the Server session
* Server session is the one that creates the JMS session
* Application written by Application programmers creates the message listener.

**20) What is the difference between JMS and RPC (Remote Procedure Call)?**

The basic difference between JMS and RPC lies in the way they message. JMS uses asynchronous messaging type while, RPC creates synchronous messaging type. The method invoker in RPC, waits for the method to finish execution and return back the control to the invoker. In JMS the message sender just sends the message to the destination and continues its own processing.

**21) Explain how does the JMS work with the J2EE?**

The application client like enterprise JavaBeans components and web components can send or receive JMS message synchronously. In addition, the application clients can also receive message asynchronously.   With the help of message-driven beans, JMS provider can optionally implement the processing of messages. Message-driven beans are a type of enterprise bean that enables the asynchronous consumption of messages.

The operation of sending and receiving message is carried out in distributed operation, which allows JMS operations and database accesses within a single transaction.

**22) What are the types of messages that are supported by JMS?**

The types of messages that are supported by JMS are

* Stream Messages
* Text Messages
* Map Messages
* Bytes Messages
* Object Messages

**23) What is MOM in reference to JMS?**

The MOM ( Message Oriented Middleware) is a software that works as an intermediate between two communicating components.  It is placed between the client and server, MOM provides the facility of passing message by using the technique queuing. Until the client does not request to read the message, the messages will be stored in queue.  By using this technique, the software component can work independently of time.

**24) How you can deliver a java message to a non-java client?**

First of all, after receiving the message from Topic or Queue, the message has to be converted into a non-java client according to their specification. The message once converted to non-java client, it can be delivered.

**25) For sending messages through JMS, what encryption options are there?**

The encryption and decryption of the messages is handled by JMS provider and not JMS specifications. Sonic MQ by Progress Software is a leading JMS provider and they do encryption through encryption mechanisms called Quality of Protection.

**JMS Exception Handling**

The root class for exceptions thrown by JMS API methods is JMSException. Catching JMSException provides a generic way of handling all exceptions related to the JMS API.

The JMSException class includes the following subclasses, which are described in the API documentation:

* IllegalStateException
* InvalidClientIDException
* InvalidDestinationException
* InvalidSelectorException
* JMSSecurityException
* MessageEOFException
* MessageFormatException
* MessageNotReadableException
* MessageNotWriteableException
* ResourceAllocationException
* TransactionInProgressException
* TransactionRolledBackException

### **Message Bodies**

The JMS API defines five message body formats, also called message types, which allow you to send and to receive data in many different forms and provide compatibility with existing messaging formats. [Table 30–2](https://docs.oracle.com/cd/E19798-01/821-1841/6nmq2cpps/index.html#bncex) describes these message types.

Table 30–2 JMS Message Types

| **Message Type** | **Body Contains** |
| --- | --- |
| TextMessage | A java.lang.String object (for example, the contents of an XML file). |
| MapMessage | A set of name-value pairs, with names as String objects and values as primitive types in the Java programming language. The entries can be accessed sequentially by enumerator or randomly by name. The order of the entries is undefined. |
| BytesMessage | A stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. |
| StreamMessage | A stream of primitive values in the Java programming language, filled and read sequentially. |
| ObjectMessage | A Serializable object in the Java programming language. |
| Message | Nothing. Composed of header fields and properties only. This message type is useful when a message body is not required. |

### **Message Headers**

A JMS message header contains a number of predefined fields that contain values that both clients and providers use to identify and to route messages. [Table 30–1](https://docs.oracle.com/cd/E19798-01/821-1841/6nmq2cppq/index.html#bnceu) lists the JMS message header fields and indicates how their values are set. For example, every message has a unique identifier, which is represented in the header field JMSMessageID. The value of another header field, JMSDestination, represents the queue or the topic to which the message is sent. Other fields include a timestamp and a priority level.

Each header field has associated setter and getter methods, which are documented in the description of the Message interface. Some header fields are intended to be set by a client, but many are set automatically by the send or the publish method, which overrides any client-set values.

Table 30–1 How JMS Message Header Field Values Are Set

| **Header Field** | **Set By** |
| --- | --- |
| 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 |