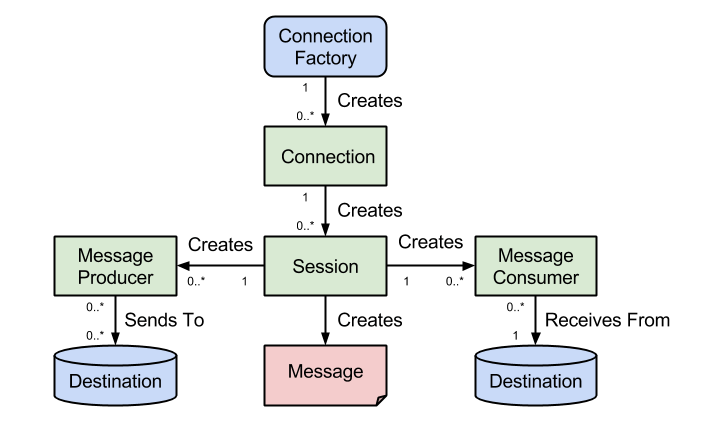
JMS API

JMS API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. It is a Java API that allows applications to create, send, receive, and read messages. The JMS API enables communication that is loosely coupled, asynchronous and reliable. To use JMS, we need to have a JMS provider that can manage the sessions, queues, and topics.

**What is Messaging?**

Messaging means exchanging information between different components in the same system or different systems. It can happen in either synchronous manner or asynchronous manner.

**The JMS API Programming Model**



At the top, we have the ConnectionFactory object which is the object a client uses to create a connection to a JMS provider. A connection factory encapsulates a set of connection configuration parameters like for example the broker URL. A connection factory is a JMS administered object that is typically created by an administrator and later used by JMS clients.

When you have a ConnectionFactory object, you can use it to create a connection. A Connection object encapsulates a virtual connection with a JMS provider. For example, a connection could represent an open TCP/IP socket between a client and a provider service daemon. Before an application completes, it must close any connections that were created. Failure to close a connection can cause resources not to be released by the JMS provider.

A session is a single-threaded context for producing and consuming messages. A session provides a transactional context with which to group a set of sends and receives into an atomic unit of work. Session objects are created on top of connections.

A MessageProducer is an object that is created by a session and used for sending messages to a destination. You use a Sessionobject to create a message producer for a destination.

 A MessageConsumer is an object that is created by a session and used for receiving messages sent to a destination. After you have created a message consumer it becomes active, and you can use it to receive messages. Message delivery does not begin until you start the connection you created by calling its start() method.

A Destination is the object a client uses to specify the target of messages it produces and the source of messages it consumes. In the point-to-point messaging domain, destinations are called queues. In the publish/subscribe messaging domain, destinations are called topics.

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

#### Point-to-Point Messaging Domain

In the point-to-point messaging domain the application is built on the basis of message queues, senders and receivers. Each and every message is addressed to a particular [queue](http://docs.oracle.com/javaee/6/api/javax/jms/Queue.html). Queues retain all messages sent to them until the messages are consumed or expired. There are some characteristics of PTP messaging:

1. There is only one client for each message.
2. There is no timing dependency for sender and receiver of a message.
3. The receiver can fetch message whether it is running or not when the sender sends the message.
4. The receiver sends the acknowledgement after receiving the message.

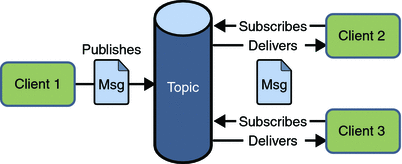
## jms point to point model

#### Publish/Subscribe Messaging Domain

In publish/subscribe messaging domain, only one message is published which is delivered to all clients through [Topic](http://docs.oracle.com/javaee/6/api/javax/jms/Topic.html) which acts as a bulletin board. Publishers and subscribers are generally anonymous and can dynamically publish or subscribe to the topic. The Topic is responsible to hold and deliver messages. The topic retains messages as long as it takes to distribute to the present clients.

Some of the characteristics are:

1. There can be multiple subscribers for a message.
2. The publisher and subscribe 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.



1.The point-to-point or queue model works by the sender to receiver setup. On the other hand, publisher/subscriber or topic model works by bulletin setup.

2.In the queue model there is acknowledgement of the identity of the receiver and oftentimes the sender. In the topic model there is anonymity in the identities of both the subscriber and publisher.

3.Queue model is only allowed one recipient; topic, on the other hand, can have multiple recipients.

4.In queue model, the sender and receiver do not have to be both active at the same time. In the topic model, timing is very vital.

5.In the queue model, the sender will receive a notification when the message gets to the receiver. The topic model, on the other hand, will not notify you with such, and there is even a risk that you will have no subscribers.