Java Messaging Service (JMS)

- What is messaging?
- What is the JMS API?
- How can you use the JMS API?
- How does the JMS API work with the J2EE platform?
- See: jms_tutorial-1_3_1.pdf
Messaging

- Messaging is a method of communication between software components or applications.
- It enables distributed communication that is loosely coupled.
  - Sender and receiver need not know anything about each other (except the destination and message format)
  - Email is for people what messaging is for applications.
- Messaging is a peer to peer facility.
  - A messaging client can send mail to and receive messages from any other client.
  - Each messaging client connects to a messaging agent that provides facilities for creating, sending, receiving and reading messages
JMS API

- Allows applications to create, send, receive, and read messages.
- Messages are
  - Asynchronous: a client does not have to request them in order to receive them.
  - Reliable: can ensure message is delivered safely once and only once.
Using JMS API

- A provider wants the components not to depend on other components’ interfaces, so that components can be easily replaced.
- The provider wants the application to run whether or not all components are up and running simultaneously.
- The application business model allows a component to send information to another and to continue to operate without receiving immediate response.
Examples using JMS

- Inventory component that can send message to factory component when inventory level for a product goes below a certain level, so that they can make more cars.
- The parts components in turn can messages to update inventories and order new parts.
- Both factory and parts components can send messages to the accounting component to update their budget numbers.
- Business application can publish updated catalog to its sales force.
Messaging in an Enterprise Application

Inventory
- New items
  - Sales

Factory
- Number low
  - Sales

Parts
- Parts needed
  - Factory

Accounting
- bill
  - Parts
- Purchase order
  - Parts

Order
- Get from inventory
  - Parts Inventory
- Order From outside
  - Parts Order

New items
Number low
Parts needed
Bill
Purchase order
Get from inventory
Order From outside
JMS and J2EE

- Messaging is an integral part of J2EE 1.3.
- Application clients, EJB components, and web components can send or synchronously receive a JMS message.
- Application clients can receive JMS messages asynchronously.
- Message driven bean enables the asynchronous consumption of messages.
- Can be used in the place of connector architecture when interacting with existing Enterprise Information Systems (EIS).
JMS Architecture

- **JMS provider**: is a messaging system that implements JMS interfaces and provides administrative and control features.

- **JMS clients**: are programs or components written in Java that produce or consume messages.

- **Messages**: are objects that communicate information between the JMS clients.

- **Administered objects**: are preconfigured JMS objects created by an administration for the use of clients. Two kinds of administered objects are destinations and connection factories.
JMS API Architecture

1: Bind destinations and connection factories objects
2. Client looks up for administered objects in the namespace
3. Establish a logical connection to the looked up object thru’ a JMS provider.
Messaging Domains

- Point-to-Point (PTP) application is built around the concept of message queues, sender and receivers.
  - Each message is addressed to a specific queue and the receiving clients extract messages from the queues established to hold their messages.
  - Each message has only one consumer.
  - A sender and receiver have no time dependencies.
  - The receiver acknowledges the successful processing of a message.
  - Use PTP when every message you send must be processed successfully by one consumer.
Publish/Subscribe domain

- Clients address messages to a topic.
- Each message may multiple consumers.
- Publishers and subscribers have timing dependencies.
- Publishers and subscribers are generally anonymous and the system takes care of distributing the messages.
Message Consumption

- Synchronously: A subscriber or receiver explicitly fetches the message from the destination by calling the receive method.

- Asynchronous: A client can register a message listener with a consumer. A message listener is similar to an event listener. Whenever a message arrives at the destination, the JMS provider delivers the message by calling the listener’s onMessage method, which acts on the content.