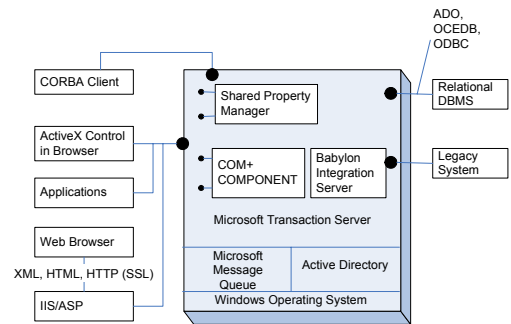


## CmpE 473 Internet Programming

Pinar Yolum  
[pyolum@cmpe.boun.edu.tr](mailto:pyolum@cmpe.boun.edu.tr)

Department of  
Computer Engineering  
Boğaziçi University

## .NET Technology



Spring 2005— Pinar Yolum

4

## Chapter 6 Enterprise Architectures

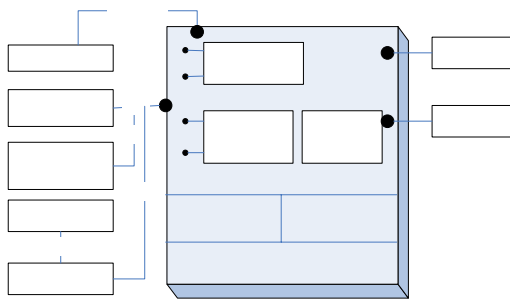
## Messaging

- Between loosely-coupled systems
  - A component sends a message to a destination
  - The receiver picks it up whenever it wants
- Asynchronous
  - Messages delivered as recipients become online
- Contrast with RMI
  - No need to know the internal details of others
  - No need to be available at the same time
- JMS: Java API

Spring 2005— Pinar Yolum

5

## J2EE Technology



Spring 2005— Pinar Yolum

3

## When to use it?

- Components should not depend on others' interfaces
- Application should run even when some components are not available
- Components can work without receiving immediate responses (non-blocking)

Spring 2005— Pinar Yolum

6

## JMS API

- JMS Provider implements JMS interface
  - Comes in J2EE
- JMS Clients produce and consume messages
- Messages are the communication objects
- Providers implement point-to-point and publish/subscribe

Spring 2005— Pinar Yolum

7

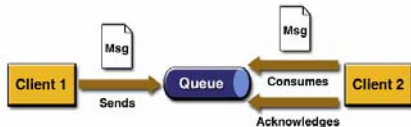
## Message Consuming

- Conceptually asynchronous:
  - No need to be available at the same time
- Synchronous access:
  - Subscriber or receiver fetches messages by calling `receive`
  - Either blocks or wait for a time period
- Asynchronous access:
  - Consumer registers with a `message listener`
  - When a message arrives at the destination, JMS provider calls client's `onMessage` method
  - Similar to the JAVA delegation event model

Spring 2005— Pinar Yolum

10

## Point-to-Point



- Client1 produces messages for a particular queue
- Client2 consumes the messages and acknowledges receipt
- Queue keeps messages until they are consumed or until they expire

Spring 2005— Pinar Yolum

8

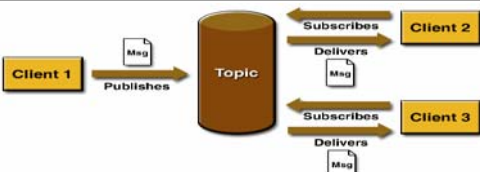
## Administered Objects

- Connection Factories
  - Object used by the client to connect to the provider
  - Contains configuration information
  - Instance of the following interfaces
    - `QueueConnectionFactory`
    - `TopicConnectionFactory`
  - Default `QueueConnectionFactory` and `TopicConnectionFactory` objects

Spring 2005— Pinar Yolum

11

## Publish/Subscribe



- Multiple producers; multiple consumers
- Topic retains messages until all subscribers are receive them
- Consumers start consuming after they subscribe

Spring 2005— Pinar Yolum

9

## Connection Factories

```
Context ctx = new InitialContext();
```

- Looks for `jndi.properties` file
  - JNDI implementation and namespaces to use
- ```
QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory) ctx.lookup("QueueConnectionFactory");
TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory) ctx.lookup("TopicConnectionFactory");
```

Spring 2005— Pinar Yolum

12

## Destination

- Target for produced messages
- Source for consumed messages
- Create a queue or a topic
  - `j2eeadmin -addJmsDestination queue_name queue`
  - `j2eeadmin -addJmsDestination topic_name topic`
- To look up a destination
  - `Queue myQueue = (Queue) ctx.lookup("MyQueue");`

Spring 2005— Pinar Yolum

13

## Message Producer

- Used for sending messages to a destination
- Create a message producer
  - `QueueSender queueSender = queueSession.createSender(myQueue);`
  - `TopicPublisher topicPublisher = topicSession.createPublisher(myTopic);`
- Send a message
  - `queueSender.send(message);`

Spring 2005— Pinar Yolum

16

## Connection

- Denotes a virtual connection with the provider
- Use connection to create sessions
  - `QueueConnection queueConnection = queueConnectionFactory.createQueueConnection();`
- Close the connection at the end
  - `queueConnection.close();`

Spring 2005— Pinar Yolum

14

## Message Consumer

- Used for receiving messages from a destination
- Create a message consumer
  - `QueueReceiver queueReceiver = queueSession.createReceiver(myQueue);`
  - `TopicSubscriber topicSubscriber = topicSession.createSubscriber(myTopic);`
- Start the connection
  - `queueConnection.start();`
- Receive messages synchronously
  - `Message m = queueReceiver.receive();`

Spring 2005— Pinar Yolum

17

## Session

- Context for producing and consuming messages
- Used to create messages, message producers, and consumers
- Transaction option
  - Two values: Transaction set and acknowledgement
  - `QueueSession queueSession = queueConnection.createQueueSession(true, 0);`
  - `TopicSession topicSession = topicConnection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);`
- `Session.CLIENT_ACKNOWLEDGE`:
  - Acknowledges all messages consumed so far

Spring 2005— Pinar Yolum

15

## Message Listener

- Asynchronous message handler object
- Implements `MessageListener` interface
  - `onMessage (Message)` method
  - `TopicListener` implements `MessageListener`
  - `TopicListener topicListener = new TopicListener();`
- Register the message listener with a `TopicSubscriber`
  - `topicSubscriber.setMessageListener(topicListener);`
- Start the connection
  - `queueConnection.start();`
- Consumer calls `messageListener`'s `onMessage`

Spring 2005— Pinar Yolum

18

## Message

- Contains header (required), properties, body
- Message header
  - Send or publish method sets the following
    - JMSDestination, JMSDeliveryMode (Default persistent)
    - JMSExpiration (Default: No expiration), setTimeToLive
    - JMSPriority (Range: 0—9; Default:4)
    - JMSMessageID, JMSTimestamp
  - Client sets the following
    - JMSCorrelationID, JMSReplyTo, JMSType
  - JMS Provider sets: JMSRedelivered

Spring 2005— Pinar Yolum

19

## Exceptions

- JMS exceptions descend from `JMSException`
  - `IllegalStateException`
  - `InvalidClientIDException`
  - `InvalidDestinationException`
  - `InvalidSelectorException`
  - `JMSecurityException`
  - `MessageEOFException`
  - `MessageFormatException`
  - `MessageNotReadableException`
  - `MessageNotWriteableException`
  - `ResourceAllocationException`
  - `TransactionInProgressException`
  - `TransactionRolledBackException`

Spring 2005— Pinar Yolum

22

## Message

- Message body
  - `TextMessage`: `java.lang.String` object (for XML)
  - `MapMessage`: Name (string)/ value (primitive) pairs
  - `BytesMessage`: A stream of bytes
  - `StreamMessage`: A stream of primitive values
  - `ObjectMessage`: A serializable Java object
  - `Message`: No body

Spring 2005— Pinar Yolum

20

## Point-to-Point Example

- `SimpleQueueSender.java`; `SimpleQueueReceiver.java`
- CLASSPATH contains `$J2EE_HOME/lib/j2ee.jar`; `$J2EE_HOME/lib/locale`
- Start J2EE server to start JMS Provider
  - `j2ee -verbose`
- Create JMS administered objects (`j2eeadmin` → `asadmin`)
  - `j2eeadmin -addJmsDestination MyQueue queue`
- Start the sender
  - `java -Djms.properties=%J2EE_HOME%\config\jms_client.properties SimpleQueueSender MyQueue 3`
- Start the receiver
  - `java -Djms.properties=%J2EE_HOME%\config\jms_client.properties SimpleQueueReceiver MyQueue`

Spring 2005— Pinar Yolum

23

## Example

- To create and send a text message

```
TextMessage message = queueSession.createTextMessage();
message.setText(msg_text); // msg_text is a String
queueSender.send(message);
```
- On the receiving side, cast the message as needed

```
Message m = queueReceiver.receive();
if (m instanceof TextMessage) {
    TextMessage message = (TextMessage) m;
    System.out.println("Reading message: " +
        message.getText());
} else { // Handle error }
```

Spring 2005— Pinar Yolum

21

## Publish/Subscribe Example

- `SimpleTopicPublisher.java`; `SimpleTopicSubscriber.java`; `TextListener.java`
- Create JMS administered objects
  - `j2eeadmin -addJmsDestination MyTopic topic`
- Start the sender
  - `java -Djms.properties=%J2EE_HOME%\config\jms_client.properties SimpleTopicSubscriber MyTopic`
- Start the receiver
  - `java -Djms.properties=%J2EE_HOME%\config\jms_client.properties SimpleTopicPublisher MyTopic 3`
- Stop the topic
  - `j2eeadmin -removeJmsDestination MyTopic`
- Easier to use: <http://localhost:4848/asadmin>

Spring 2005— Pinar Yolum

24

## Clients on different hosts

- Both hosts are running the J2EE Server
  - Start the server on both hosts (earth, mars)
- Create a connection factory on Earth
  - `j2eeadmin -addJmsFactory jms/EarthQCF queue`
- Create a connection factory on Mars and have it point to Earth's connection factory
  - `j2eeadmin -addJmsFactory jms/EarthQCF queue -props url=corbaname:iiop:earth:1050#earth`
- Modify the source codes so they show
  - `queueConnectionFactory = (QueueConnectionFactory) jndiContext.lookup("jms/EarthQCF");`

Spring 2005— Pinar Yolum

25

## Clients on different hosts

- One host is running the J2EE Server
  - Start the server on Earth
- Create a connection factory on Earth
  - `j2eeadmin -addJmsFactory jms/EarthQCF queue`
- Use the following as an extra parameter to access the server
  - `-Dorg.omg.CORBA.ORBInitialHost=hostname`
  - `hostname: Server`
- Start the subscriber
  - `java -Djms.properties=%J2EE_HOME%\config\jms_client.properties -Dorg.omg.CORBA.ORBInitialHost=earth SimpleTopicSubscriber MyTopic`

Spring 2005— Pinar Yolum

26