Introduction to Messaging using JMS

Evan Mamas
emamas@ca.ibm.com
IBM Toronto Lab

CASCON 2002

Outline
- Basic Concepts
- API Architecture
- API Programming Model
- Advanced features
- Integration with J2EE
- Simple applications
- References
Messaging

- "Method of communication between software components"
- Point to Point
  - A client can send and receive messages to and from another client
  - A messaging agent provides facilities for creating, sending, and receiving messages
- Loosely coupled
  - Sender / Receiver are not required to be available at the same time
  - They do not have to know anything about each other

What is JMS?

- Standard API for messaging
  - Provides access to Message Oriented Middleware (MOM)
  - Analogous to JDBC and DBMS
- Support for distributed transactions
- Integration with J2EE
  - Synchronous message send and receive
  - Message driven beans
- Latest spec version is 1.1
  - Part of J2EE 1.3 (1.4)
Why use JMS?

- JMS vs. Vendor Specific APIs
  - Abstraction from vendor specific APIs
  - Message Oriented Middleware (MOM) vendors offer similar functionality and APIs
  - Easy to upgrade components

- JMS vs. RPC
  - Decoupled components
    - Easy to replace components of the system
  - Application does not require all components to be up and running
  - Asynchronous messaging scales better when multiple systems need to be integrated

RPC vs. JMS
JMS Basic Concepts

- **Architecture**
  - Providers, Clients, Messages
- **Messaging domains**
  - Point to Point
  - Publish / Subscribe
  - J2EE implementers must support both
- **Message consumption**
  - Synchronous and Asynchronous

JMS API Architecture

- **JMS Provider**
  - Implements JMS interfaces
  - A J2EE 1.3 implementation
- **JMS Client**
  - Produces and consumes messages
- **Messages**
  - Multiple message formats
- **Administered Objects**
  - Required by clients
  - Connection and destination factories
- **Native Clients**
JMS API Architecture

Point to Point Messaging domain

- Message queue, sender, receiver
- Clients must know which queue to use
- Queue retains the message until consumed or expires
- One consumer per message
- No timing dependencies
Point to Point Messaging domain

- Topic based
  - Hierarchy of topics is possible
- Publishers and subscribers are anonymous
- Messages are retained as long as necessary to get distributed to subscribers
- One message multiple consumers
- Timing constraints
  - Messages received only after subscription
  - Subscriber must remain active
  - Durable messages
Message Consumption

- **Synchronous**
  - Receiver explicitly fetches the message from the queue by calling the `receive` method
  - Call can be blocking
  - A timeout could be used if message does not arrive in specified time

- **Asynchronous**
  - Register a message listener with a consumer
  - `onMessage` method is invoked when a message becomes available
JMS API Programming Model

- Administered Objects
- Connections
- Sessions
- Message Producers
- Message Consumers
- Messages

Administered Objects

- Configured by the administrator
- Usually looked up through JNDI
- Connection Factories
  - Used by the client to create a connection with a provider
  - Has preconfigured parameters
  - QueueConnectionFactory or TopicConnectionFactory
- Destination Factories
  - Where messages are put/read from
  - Point To Point: Queues
  - Publish/Subscribe: Topics
Connections

- A virtual connection with a JMS provider
- Queue or Topic connections are created using the corresponding connection factory
- start needs to be called before messages can be consumed
- close needs to be called to release resources
- stop can be used to temporarily stop message delivery without closing the connection

Sessions

- Single threaded context for consuming/producing messages
- Sessions are used to create message consumers, message providers, and messages
- Sessions provide a context to group as one atomic operation a number of sends and receives
- QueueSession and TopicSession
- Can be transactional (local)
- Can provide acknowledgment
Message Producers

- Send messages to a destination
- Unidentified producers can be used when we want to specify the destination at send time
- QueueSender interface
  - send
- TopicPublisher interface
  - publish

Message Consumers

- Receives messages sent to a destination
- Message Receivers
  - QueueReceiver and TopicSubscriber
  - start and close
  - Used synchronously by calling the receive method
- Message Listeners
  - QueueListener and TopicListener
  - Used asynchronously (onMessage)
- Message Selectors
  - Filter messages using an SQL92 conditional expression
  - Only headers and properties can be filtered (not content)
Messages

- Header
  - Unique ID, timestamp, destination, priority, ...
- Properties (optional)
- Body (optional)
  - TextMessage
  - MapMessage
  - BytesMessage
  - StreamMessage
  - ObjectMessage
  - Message (no body)

JMS API Programming Model

Connection Factory creates a Connection, which creates a Session. The Session sends to a Destination and creates a Message. The Message is received from a Destination by a Message Consumer. A Connection Factory also creates a Message Producer, which creates a Message (Msg).
Reliability

- Acknowledgment
  - Client, Provider
- Persistence
- Priority
  - 0-9 (default is 4)
- Expiration
- Temporary destinations
  - Valid within the scope of a connection
  - Allow client to programmatically create destinations

Reliability (advanced)

- Durable subscriptions
- Local transactions
  - Using the session as the context
- Distributed transactions
  - Container managed
  - Bean managed
A Simple J2EE Application: Client to Message-Driven Bean
ERROR: ioerror
OFFENDING COMMAND: %image_file_continue

STACK: