Extended Messaging Support in WebSphere EE
Agenda

- Messaging in an n-Tier environment
- Messaging support in WebSphere
- JMS Listener in WebSphere Enterprise Extensions
  - Programming model
  - Configuration file
  - Transaction support
  - Connection pooling
  - Prerequisites
  - Further reading
Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility: a messaging client can send messages to, and receive messages from, any other client. Each client connects to a messaging agent that provides facilities for creating, sending, and receiving messages. Servlets and EJBs can send and receive messages to/from a messaging vendor. Messaging technologies provide an additional means of communicating with Server Side Components.
Loose Coupling

- Enables application programs to communicate *asynchronously* with each other
- Application data sent as a message
- If the receiver, or the communication channel to it, is temporarily unavailable, the message can be forwarded at a later time.

Messaging in an n-Tier environment

Messaging

Middleware

(QUEUE)

PUT

Messaging

GET

Middleware

(QUEUE)

IBM

Messaging enables distributed communication that is loosely coupled. A component sends a message to a destination, and the recipient can retrieve it from the destination. However, the sender and receiver do not have to be available at the same time in order to communicate. In fact, the sender does not need to know anything about the receiver, nor does the receiver need to know anything about the sender; they only need to know what message format and what destination to use. In this respect, messaging differs from tightly coupled technologies such as Remote Method Invocation (RMI), which require an application to know a remote application’s methods.
Applications running in WebSphere can create, send and receive messages using JMS
Inbound and Outbound messages

**Inbound Messages**
- Inject work into WebSphere Application Server
- An EJB or Servlet programmatically receives messages from a JMS Destination (QUEUE)
  ... or uses a JMS Listener

**Outbound messages**
- Delegate work to application components outside WebSphere
- WebSphere sends messages to a JMS Destination

A JMS Listener automates tasks that would otherwise be done by the application code. In JMS, a Message Queue is referred to as a JMS Destination. JMS Destinations can be of two types
  - Queue: Used for Point-To-Point messaging
  - Topic: Used for Publish/Subscribe messaging
Message Listener in an n-Tier environment

JMS Listener
--> adds function to the server
--> new source of inbound work
--> target is EJBs
Benefits of JMS Listeners

- Automatic consumption of messages
  - No polling needed in the application code
- Reduce application code
- Synchronous communication between Queue and Listener
- Leave resource management to the container
Send data to Queue

Wait till WebSphere is ready

Deliver data to WebSphere

Listener

EJB Container

Message Bean

Business Bean

PUT

GET

Messaging Middleware (QUEUE)
"Application component that accepts **INBOUND** messages"

A Message Bean is a Stateless Session Bean that follows certain guidelines

- Automatic synchronous delivery of messages
- Injects work into a WebSphere application
- Provides transactional support
- Follows direction for Message Driven Beans
  - Message Driven Beans defined in J2EE 1.3 spec
  - Supported in the next WebSphere release
- Built upon MQSeries and JMS
By providing a clear separation between message and business processing, it is easier to implement the Message Beans as Message Driven Beans (J2EE 1.3)
public void onMessage(javax.jms.Message msg) {
    try {
        System.out.println("Input message = 
             + ((TextMessage) msg).getText().);
        MyEJBHome home = (MyEJBHome)PortableRemoteObject.narrow
             (ic.lookup("com/mycom/MyEJBHome"),MyEJBHome.class);
        MyEJBHome obj = home.create();
        obj.myBusinessMethod(msg);  // business method
    } catch(Exception err) {
        err.printStackTrace();
    }
}
Listener Manager

- Every AppServer has a Listener Manager
- Enabled during AppServer initialization
- Creates JMSListeners for destinations
  - Based upon a user provided XML configuration file
- Interfaces with JMS Application Server Facilities (ASF)
- Controls the cleanup of resources during server shutdown
In order to enable Message Beans, the AppServer has to be provided a configuration file. This configuration file is read when the AppServer starts up.
A sample configuration file is provided with WebSphere.
In the config file, the underlined values are the defaults
The config file can have multiple <Listener></Listener> tag pairs.
The config file can have only one <Pooling></Pooling> tag pair.
Specifying the location of the XML config file in AEs
Specifying the location of the XML config file in AE
A single Application Server can have multiple JMS Listeners.
JMSAdmin.bat is a configuration tool provided by MQSeries as part of the MA88 install. The tool uses a configuration file (default name JMSAdmin.config).

The foil above shows the differences between AE and AEs when providing JNDI options in the config file. In AEs, we use a file system (C:\...) to configure the JMS administered objects. The reason being that AEs does not persist its JNDI namespace.

In AE however, we can rely on using WebSphere namespace (iiop://localhost/)
One Listener Manager per App Server
Many Listeners per Listener Manager
One connection per Listener
Many Sessions per connection
Listener threads are managed by the Listener Manager
Message Bean Session threads are managed by the Listener
One Listener corresponding to a `<Listener>`/<listener> tag pair in the XML config file
Message Bean has the onMessage() method
Message headers can be read before pulling messages off a Queue. The filtering criteria is specified in the XML config file.
Queues/Topics

Destination:
- Q1
- Q2
- Q3
- T1
- T2

MessageBeans:
- M1
- M2
- M3
- M4

BusinessBeans:
- B1
- B2
- B3
- B4
Transactional Support

- Allows JMS Listener participation in a 2-phase commit
- Configuration
  - In XML configuration file: Transactional = true
    - `<Transactional> true | false </Transactional>`
  - Use JMSAdmin to configure WebSphere JMS / XA connection factories

The JMSAdmin tool is provided by MQSeries
Detailed information on this tool can be found in the redbook "MQSeries Using Java"
Connection Pooling

- Enable MQSeries connection pooling
  - Provides improved JMS system performance
  - Maximum benefit when connections used for short time
  - Enabled by Listener during initialization

Configuration

- Values specified via configuration file
- Will use default values if not specified
  - Timeout = 300000 milliseconds
  - Threshold = 10 Connections
Support pack ma0c needs to be installed in MQSeries to enable Publish/Subscribe. This is in addition to ma88 support pack.
Summary - JMS Listener

- Part of WebSphere 4.0 EE
- Provides ability to inject work into the system
- Provides transaction support
- Provides connection pooling with MQSeries
- Follows direction for Message Driven Beans in J2EE 1.3
WebSphere EE 4.0 InfoCenter

"MQSeries Using Java"  SC34-5456-06
  ➤ http://www.redbooks.ibm.com

White Paper- WebSphere 3.5 JMS/JTA support for MQSeries
  ➤ http://www-4.ibm.com/software/webservers/appserv/whitepapers/was_mqseries.pdf

http://www.javasoft.com/jms

IBM