EJB3: Message-Driven Beans

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/msajsp.html

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at your location.

For live Java EE training, please see training courses at http://courses.coreservlets.com/. Servlets, JSP, Struts, JSF, Ajax, GWT, Java 6, Spring, Hibernate, JPA, EJB3, Web Services, & customized combinations of topics.

Taught by the author of Core Servlets and JSP, More Servlets and JSP, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization. Contact hall@coreservlets.com for details.
Agenda

• Distinguishing message-driven beans (MDBs) from session beans
• Types of messages
• Making an MDB for receiving TextMessages
• Making a client to send TextMessages

Overview
Session Beans vs. Message-Driven Beans (MDBs)

**Session beans**
- Called directly
  - Client accesses specific bean (via interface)
- Invoked synchronously
  - Client waits until bean finishes response
- Can return values to client
  - Client can receive return value from method call
- Based on interfaces
  - EJB is interface plus concrete class
  - Client refers to interface

**Message-driven beans**
- Called indirectly
  - Client puts message in queue. Queue later decides on which bean gets the message.
- Invoked asynchronously
  - No waiting for bean to process request. Messages not always handled in order sent.
  - Do not return values to client ("fire and forget")
    - Client can send data, but cannot get response value
  - Not based on interfaces
    - EJB can be concrete class only
    - Client does not refer to class in any way (only to queue)

General Approach

**Server**
- Class implements MessageListener
  - Overrides onMessage
  - Tagged with @MessageDriven
  - Usually with activationConfig giving queue type & name
- Blah-service.xml
  - Gives server-specific definition of message queue

**Client**
- Main class
  - Finds ConnectionFactory and Queue from InitialContext
  - Creates Session and MessageProducer
  - Repeatedly sends TextMessage, ObjectMessage, or other Message type
- jndi.properties
  - Same as in section on session beans
Message Types

- There are five predefined Message types
  - TextMessage
    - To send Strings
  - ObjectMessage
    - To send Serializable Objects
  - MapMessage
    - To send sets of names (Strings) and values (primitives).
  - ByteMessage
    - To send binary messages (streams of bytes)
  - StreamMessage
    - To send a series of Java primitives
- **Server casts onMessage’s arg to this type**
  - Server can do instanceof if same MDB will receive more than one type of Message

© 2009 Marty Hall

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at your location.
Approach

- **Make new EJB project**
  - File → New → EJB Project

- **Implement MessageListener**
  - Override `onMessage`, cast argument to `TextMessage`, extract message String with `getText`
  - Tag with `@MessageDriven`
  - Usually define `activationConfig` in annotation, but can be defined programmatically
    - `destinationType` is usually `javax.jms.Queue`
    - `destination` is `queue/YourQueueName`

- **Put `YourQueueName-service.xml` in src**
  - Contains server-specific info (see upcoming slide)

- **Deploy to app server**
  - R-click server, Add/Remove Projects, start server

EJB Project

- **Making project**
  - File → New → EJB Project
  - Pick a name
  - JBoss as target runtime
    - Deploying to different server later requires changing server-specific `Blah-service.xml` file
  - Optional: add to an EAR
    - If you make Dynamic Web project later that wants to use `@EJB`, you can add it to the same EAR

- **Deploying project**
  - R-click on JBoss, Add and Remove Projects, choose Project, Add, Finish
  - R-click on JBoss, Start
Listener Class

```java
@MessageDriven(activationConfig =
{
    @ActivationConfigProperty(propertyName="destinationType",
        propertyValue="javax.jms.Queue"),
    @ActivationConfigProperty(propertyName="destination",
        propertyValue="queue/SimpleMessageQ")
})
public class SimpleMessageBean implements MessageListener {
    public void onMessage(Message msg) {
        TextMessage txtMsg = (TextMessage)msg;
        try {
            String message = txtMsg.getText();
            System.out.printf("Received message: %s", message);
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

SimpleQ-service.xml (JBoss-Specific!)

```xml
<server>
    <mbean code="org.jboss.jms.server.destination.QueueService"
        name="jboss.messaging.destination:service=Queue,name=SimpleMessageQ"
        xmbean-dd="xmdesc/Queue-xmbean.xml">
        <depends optional-attribute-name="ServerPeer">
            jboss.messaging:service=ServerPeer
        </depends>
        <depends>jboss.messaging:service=PostOffice</depends>
    </mbean>
</server>
```
Clients for MDBs that Expect TextMessages

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at your location.

Approach

• Make new project
  – File → New → Java Project or
    • For standalone/desktop client
  – File → New → Dynamic Web Project
    • For Web apps that invoke MDBs

• Send messages
  – Get InitialContext, look up ConnectionFactory & Queue
  – Make Connection, Session, and MessageProducer
  – Use MessageProducer to send TextMessages

• Put jndi.properties in src
  – As in section on session beans

• Deploy to app server
  – R-click server, Add/Remove Projects, start server
    • For Web projects but not for desktop Java projects
public class SimpleMessageClient {
    public static void main(String[] args) throws Exception {
        InitialContext context = new InitialContext();
        ConnectionFactory cf =
            (ConnectionFactory)context.lookup("/ConnectionFactory");
        Queue queue = (Queue)context.lookup("queue/SimpleMessageQ");
        Connection connection = cf.createConnection();
        Session session =
            connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageProducer sender = session.createProducer(queue);

        for(int i=1; i<=10; i++) {
            String messageText =
                String.format("This is message number %s.", i);
            TextMessage message =
                session.createTextMessage(messageText);
            sender.send(message);
        }
    }
}
Remote Client: jndi.properties

- For JBoss
  java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
  java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
  java.naming.provider.url=localhost:1099

- For Glassfish
  org.omg.CORBA.ORBInitialHost=localhost

- Notes
  - Put in “src” folder in Eclipse to be sure it is in classpath
  - The Eclipse project you can download has both versions. Just copy/rename jboss-jndi.properties or glassfish-jndi.properties.

Remote Standalone Client: Deploying

- messages (EJB Project)
  - Deployed to JBoss.
  - SimpleQ-service.xml has JBoss-specific info

- messages-client (Dynamic Web Project)
  - Not yet deployed to any server.
  - jndi.properties has JBoss-specific info
Remote Standalone Client: Results (JBoss server log)

2009-06-24 16:31:28,472 INFO  [STDOUT] (WorkManager(2)-7) Received message: This is message number 5.
2009-06-24 16:31:28,473 INFO  [STDOUT] (WorkManager(2)-9) Received message: This is message number 7.
2009-06-24 16:31:28,475 INFO  [STDOUT] (WorkManager(2)-10) Received message: This is message number 8.
2009-06-24 16:31:28,476 INFO  [STDOUT] (WorkManager(2)-11) Received message: This is message number 9.
2009-06-24 16:31:28,479 INFO  [STDOUT] (WorkManager(2)-11) Received message:
2009-06-24 16:31:28,488 INFO  [STDOUT] (WorkManager(2)-8) Received message: This is message number 6.
2009-06-24 16:31:28,493 INFO  [STDOUT] (WorkManager(2)-5) Received message: This is message number 3.
2009-06-24 16:31:28,494 INFO  [STDOUT] (WorkManager(2)-3) Received message: This is message number 1.
2009-06-24 16:31:28,494 INFO  [STDOUT] (WorkManager(2)-6) Received message: This is message number 4.
2009-06-24 16:31:28,496 INFO  [STDOUT] (WorkManager(2)-4) Received message: This is message number 2.

Wrap-up

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at your location.
Summary

- **Server**
  - Implement MessageListener
    - Override `onMessage`, cast argument to `TextMessage`, extract message String with `getText`
    - Tag with `@MessageDriven`
    - Usually define `activationConfig` in annotation
  - Put `YourQueueName-service.xml` in `src`
- **Client**
  - Java class
    - Get `InitialContext`, look up `ConnectionFactory` & `Queue`
    - Make `Connection`, `Session`, and `MessageProducer`
    - Use `MessageProducer` to send `TextMessages`
  - Put `jndi.properties` in `src`