Servicios de Mensajes en JEE

Sesión3: JMS en aplicaciones reales (1)

Indice

- EJBs de Mensajes
- Interacción con SSBs

Indice

- EJBs de mensajes
- Interacción SSBs

EJBs de Mensajes

- Message-Driven Beans (MDBs):
  - Funcionamiento vs clientes *standalone (1)*:
    - Ambos tienen un método `OnMessage()`.
    - El contenedor de EJBs realiza automáticamente tareas de setup:
      - Crear consumidores.
      - El usuario se limita a asociar el MDB con un destination y una connection factory en tiempo de despliegue.
      - Si se quiere especificar una durable subscription o un selector de mensajes se puede hacer también en tiempo de despliegue.
      - El MDB no tiene que registrar el message listener
      - No es necesario especificar el acknowledgment mode.

EJBs de Mensajes

- Message-Driven Beans (MDBs):
  - Funcionamiento vs clientes *standalone (2)*:
    - El MDB debe implementar las interfaces
      `javax.ejb.MessageDrivenBean`
      `javax.jms.MessageListener`
    - El MDB debe implementar el método `ejbCreate()`
      - Este método se utiliza si el MBD produce mensajes o bien los recibe de forma síncrona de otra destination. Se usa este método para buscar las factorías de conexión destinations y para crear una conexión.
    - El MDB debe implementar `ejbRemove()`
      - Si se usa el `ejbCreate()` para crear la conexión, éste método se usa para cerrarla.

EJBs de Mensajes

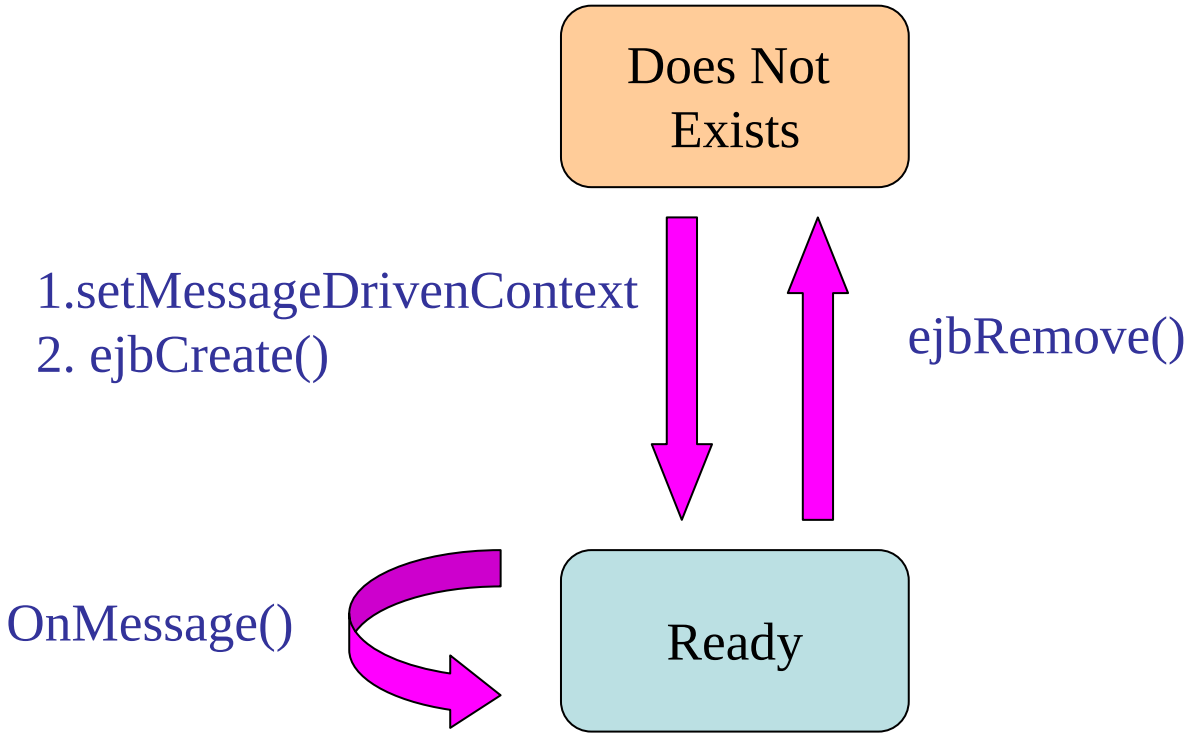
- Message-Driven Beans (MDBs):
  - Funcionamiento vs clientes *standalone (3)*:
    - El MDB debe implementar el método `setMessageDrivenContext`
      - Un `MessageDrivenContext` proporciona métodos adicionales que pueden usarse para el manejo de transacciones.
  - Funcionamiento respecto a stateless session beans:
    - La diferencia fundamental con cualquier otro EJB es que el MDB no tiene interface local o remota. Solo la clase bean.
    - Es similar a un SSB porque sus instancias son short-lived y no retienen estado para un cliente específico. Pero sus variables pueden contener info de estado mediante clientes JMS (una conexión, p.e.).

EJBs de Mensajes

- Message-Driven Beans (MDBs):
  - Funcionamiento respecto a stateless session beans:
    - Como un SSB un MDB puede tener varias instancias intercambiables ejecutándose al mismo tiempo. El contenedor puede hacer un *pooling* de instancias para permitir que los mensajes se procesen concurrentemente, lo cual puede afectar al orden en que se reciben los mensajes.
  - Creación de un MDB:
    1. Llamar al método `setMessageDrivenContext` para pasar el contexto del objeto a la instancia.
    2. Llamar al método `ejbCreate()` de la instancia.

EJBs de Mensajes

- Message-Driven Beans (MDBs):
  - Ciclo de vida:



EJBs de Mensajes

- Ejemplo: `MessageTraderBean.java` (en WL 9.2)
  - Funcionamiento:
    - Supogamos que tenemos un cliente llamado Client.java que publica mensajes en un tópico en el que está *escuchando* el MDB MessageTraderBean.java
    - El MDB MessageTraderBean.java simplemente lee el mensaje y lo escribe en pantalla.
    - Se supone que el mensaje es de texto.
    - Los métodos `ejbCreate()` y `ejbRemove()` se implementan pero se dejan vacíos en este caso.
    - Hay dos ficheros XML de despliegue. El general y otro específico.

Indice

- EJBs de mensajes
- Interacción con SSBs

Interacción con SSBs

- Ejemplo: `TraderReceive.java` (en WL 7.0)
  - Funcionamiento:
    - Supogamos que tenemos un servlet llamado TraderServlet.java que envía mensajes MAP (atributo-valor) de compra (“sell”) o de venta (“buy”) a un `Topic`.
    - El cliente `TraderReceive.java` invoca un EJB de compra o de venta para realizar la tarea especificada en el mensaje MAP.
    - Utilizamos transacciones, pero no en el contexto JMS sino en el contexto de `javax.transaction` con lo cual, utilizaremos los métodos `begin()`, `commit()` y `rollback()` de la clase `javax.transaction.UserTransaction`.

Interacción con SSBs

- Ejemplo: `TraderReceive.java`
  1. Inicialización: `init()`

```
public void init(Context ctx, String topicName)
       throws NamingException, JMSException, RemoteException,
CreateException
  {
    connectionFactory = (TopicConnectionFactory) ctx.lookup(JMS_FACTORY);

    connection = connectionFactory.createTopicConnection();
    connection.setClientID("traderReceive");
    session = connection.createTopicSession(false,
Session.AUTO_ACKNOWLEDGE);
    topic = (Topic) ctx.lookup(topicName);
    subscriber = session.createDurableSubscriber(topic, "traderReceive");
    TraderHome brokerage = (TraderHome) ctx.lookup(EJB_HOME);
    ejbTrader = brokerage.create();
    tx = (javax.transaction.UserTransaction) ctx.lookup(TX);
    connection.start();
  }
```

Interacción con SSBs

- Ejemplo: `TraderReceive.java`
  2. Recepción y tratamiento: `processMessages()`
     1. Modo recepción e iniciar transacción

```
Message msg = subscriber.receive();
tx.begin();
```

     2. Leer mensaje MAP

```
MapMessage m = (MapMessage) msg;
String customerName = m.getString("CustomerName");
String tradeType = m.getString("TradeType");
String symbol = m.getString("Symbol");
int numberOfShares = m.getInt("Shares");
```

Interacción con SSBs

- Ejemplo: `TraderReceive.java`
  2. Recepción y tratamiento: `processMessages()`
     3. Realizar operación correspondiente:

```
if ("buy".equalsIgnoreCase(tradeType)) {
  tr = ejbTrader.buy(symbol, numberOfShares);
  System.out.println("Bought " + tr.getNumberTraded());
  tx.commit();
} else {
  if ("sell".equalsIgnoreCase(tradeType)) {
    tr = ejbTrader.sell(symbol, numberOfShares);
    System.out.println("Sold " + tr.getNumberTraded());
    tx.commit();
  } else {
    System.out.println("Rolling Back Transaction");
    tx.rollback();
    System.out.println("Unknown TradeType: "+tradeType);
  }
}
```

Ejercicios…

- Ejemplo de beans de mensajes

  Compilad, desplegad y probad el ejemplo `examples.ejb.ejb20.message` para ver el funcionamiento del MDB. Interaccionad con él mediante otros clientes (p.e. el servlet)

- Ejemplo de beans de mensajes (opcional)
  - Haced que el beans de mensajes envíe a su vez el mensaje recibido a otra destination en la que espera otro MDB.