Práctica Java JMS Middleware

Víctor Portals Lorenzo - R090218

8 de julio de 2014

1. Introducción

En esta práctica se actualiza la práctica realizada con Java RMI. Se mantiene el funcionamiento de toda la aplicación a excepción de la comunicación de los mensajes de estado, que ahora se realiza mediante el sistema de mensajes JMS.

Como servidor de mensajes JMS se ha empleado Apache ActiveMQ por la sencillez de su instalación, configuración y la posibilidad de crear Topics dinámicamente.

El funcionamiento es el siguiente, cada vez que un nuevo usuario realiza el proceso de registro, desde el servidor se crea un nuevo Topic con el nombre único del usuario. A partir de este momento cada vez que un usuario inicie sesión el servidor creará una subscripción al topic correspondiente.

El usuario al iniciar sesión se conectará a su topic propio como publicador y como subscriptor a los topics de todos sus amigos. Cuando escriba y envíe un nuevo mensaje de estado, lo que estará haciendo es enviar un mensaje JMS de tipo TextMessage a su topic. Entonces, el servidor y los clientes amigos del usuario recibirán el mensaje, el servidor mediante el método onMessage implementado en Sesion.java, escribirá en la base de datos el mensaje recibido. El cliente amigo, mediante el método onMessage implementado en la clase notificará la recepción de un nuevo mensaje, y en caso de estar ya en la vista de novedades actualizará además la vista para mostrar el nuevo mensaje.

2. Código JMS

Servidor:

MainServer.java

```
.
.
.
private static String url = ActiveMQConnection.DEFAULT_BROKER_URL;
        public static ConnectionFactory connectionFactory;

        public static void main (String[] args){
.
.
.
connectionFactory = new ActiveMQConnectionFactory(url);
```

Conexión.java

```
.
.
.
public class Conexion extends UnicastRemoteObject implements ConexionInterfaz, Serializable {
        .
        .
        .
        public boolean iniciarSesion(String email, String pass) throws RemoteException {
```

```
                Connection connection2 = MainServer.connectionFactory.createConnection();
                javax.jms.TopicSession           sessionjms           =           (TopicSession)
connection2.createSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
                connection2.start();
                Topic messageTopic = sessionjms.createTopic(email + "_topic");
                TopicSubscriber  subscriber  =  sessionjms.createSubscriber(messageTopic,  null,
true);
                subscriber.setMessageListener(stub);
                connection2.start();
        .
        .
        .
        public boolean crearUsuario(String email, String name, String apellidos, String pass,
byte[] foto, String web, boolean publico) throws RemoteException{
        .
        .
        .
                Connection connection;
                javax.jms.Session session;
                try {
                        connection = MainServer.connectionFactory.createConnection();
                        connection.start();
                        session                 =                 connection.createSession(false,
javax.jms.Session.AUTO_ACKNOWLEDGE);
                       Destination destination = session.createQueue(email + "_queue");
                        Destination destination2 = session.createTopic(email + "_topic");
                        MessageProducer producer = session.createProducer(destination);
                        MessageProducer producer2 = session.createProducer(destination2);
                        producer.getDestination();
                        producer2.getDestination();
                        connection.close();
                } catch (JMSException e) {
                        e.printStackTrace();
                }
                boolean ini_sesion = iniciarSesion(email, pass);
                return ini_sesion;
        }
}
```

Sesion.java

```
public  class  Sesion  extends  UnicastRemoteObject  implements  SesionInterfaz,  Serializable,
MessageListener {
        .
        .
        .
        public void onMessage(javax.jms.Message message) {
                System.out.println("Recibe el mensaje");
                try {
            TextMessage textMessage = (TextMessage) message;
            Session session3 = this.sessionFac.openSession();
                session3.beginTransaction();
                session3.save(new Message(email, textMessage.getText(), nombre));
                session3.getTransaction().commit();
                session3.close();
                } catch (JMSException jmse){ jmse.printStackTrace(); }

        }
}
```

Cliente:

MainWindow.java

```
public class MainWindow extends JFrame implements MessageListener {
        .
        .
        .
        static ConnectionFactory connectionFactory;
        public  MainWindow(Point  location,Dimension  dim,  String  email)  throws  IOException,
Exception{
        .
        .
        .
        connectionFactory = new ActiveMQConnectionFactory("tcp://"+ MainClient.host + ":61616");
        .
        .
        .
@Override
        public void onMessage(Message message) {
                //TextMessage textMessage = (TextMessage) message;
                //String usuario = textMessage.getStringProperty("usuario");
                newsNotification();
        }
```

WallPamel.java

```
public class WallPanel extends JPanel {
        .
        .
        .
        public WallPanel(String email){
        .
        .
        .
btnSend.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent arg0) {
        .
        .
        .
        Connection connection;
        javax.jms.Session session;
        try {
                connection = MainWindow.connectionFactory.createConnection();
                connection.start();
                session = connection.createSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
                Destination destination = session.createTopic(email_user + "_topic");
                MessageProducer producer = session.createProducer(destination);
                TextMessage message = session.createTextMessage(txtNewState.getText());
                message.setStringProperty("usuario", email_user);
                producer.send(message);
                connection.close();
        } catch (JMSException e) {
                e.printStackTrace();
        }
        .
        .
        .
```

3. Conclusiones y opinión personal

Finalmente no he tenido tiempo para implementar la funcionalidad completa mediante JMS aunque me hubiera gustado hacerlo. Con JMS se simplifica mucho el sistema de notificaciones en vivo respecto a RMI, que da problemas con cortafuegos además de requerir bastante más código para lo mismo.

La principal dificultad que he encontrado ha sido con la instalación y configuración del servidor JMS. Inicialmente he perdido bastante tiempo intentando instalar y configurar Gassfish v4 y al final decidí probar con el siguiente servidor propuesto en la asignatura, ActiveMQ, cuya instalación ha sido muy sencilla y no ha necesitado ninguna configuración.