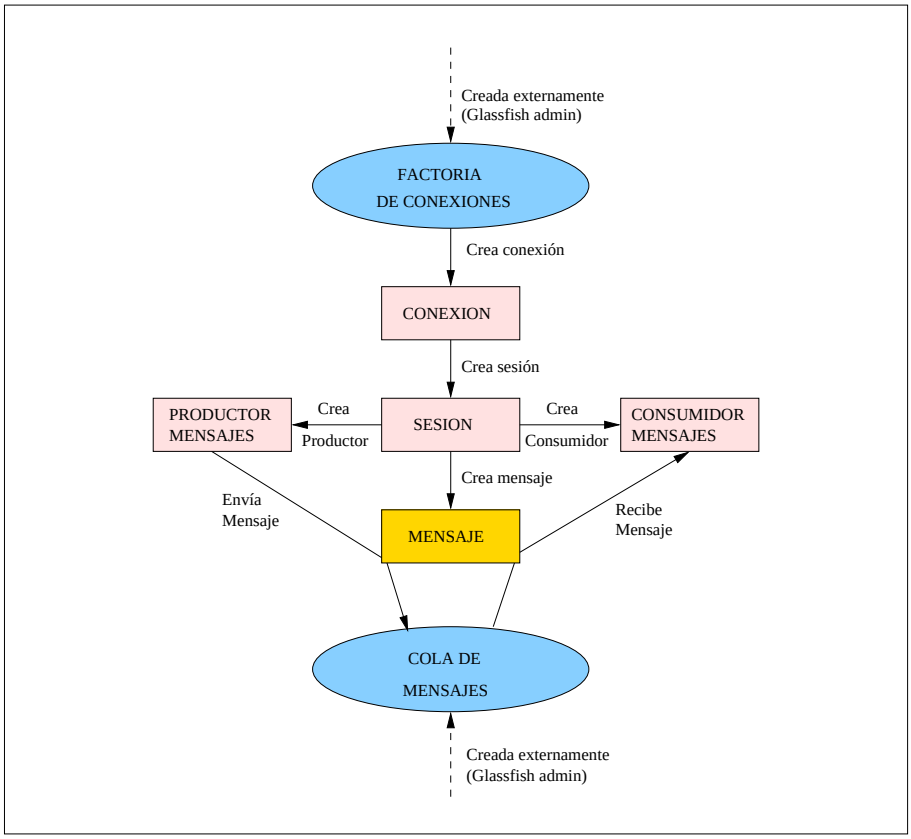
PRACTICAS DE SISTEMAS DISTRIBUIDOS.

Práctica 1: Envío de mensajes con Java Message Service (JMS)

Java Message Service es un API de Java para el envío y recepción de mensajes mediante un sistema de sesiones, conexiones, colas de mensajes y tópicos. En esta práctica trabajaremos con el envío de mensajes entre pares a través de una cola de mensajes, si bien JMS también ofrece soporte para el modelo *Publish/Subscribe*, sobre la base de *tópicos*, a los cuales se pueden suscribir los clientes y recibir los mensajes destinados a un tópico en concreto. Básicamente esta segunda opción es una forma de multienvío mediante una cola de mensajes (tópico).

Esta práctica la realizaremos con *Netbeans*, en concreto ha sido probada con Netbeans 7.2.1, y soporte Glassfish 3.1.2.2.

1. La gráfica siguiente ilustra esquemáticamente el funcionamiento del envío y recepción de mensajes en JMS:



Observamos que hemos de crear manualmente (mediante la herramienta de administración de recursos de *Glassfish*) una factoría de conexiones de colas de mensajes y una cola de mensajes. La factoría de conexiones después tendrá que ser localizada dentro de la aplicación mediante un servicio de directorio (JNDI). A través de la factoría de conexiones podremos crear una conexión, la cual a su vez nos permitirá crear una sesión. Las sesiones se emplean para diversos fines en JMS, en particular nos permiten crear *transacciones*, es decir, podemos agrupar una serie de envíos y recepciones para garantizar sobre ellos la atomicidad (todo o nada en su

ejecución). Para una sesión ya podemos crear uno o varios productores/consumidores de mensajes, los cuales ya pueden (respectivamente) enviar y recibir mensajes de la cola de mensajes.

A continuación se describen los pasos que el alumno debe seguir para realizar la práctica:

a) Crear un nuevo proyecto "ProyectoJMS1" (Java EE) con *Netbeans*, de tipo *Enterprise application client* (es importante elegir ese tipo para evitar problemas de conectividad posteriores). Dejar las opciones por defecto, pulsar *ok* y después *finish*.

b) Seleccionar "ProyectoJMS1-ejb", y con el botón derecho *New->*Java class. Le pondremos por nombre *MsgConexion*, que será una clase que usaremos para encapsular toda la problemática relativa a la factoría, conexión y sesión, y podremos usarla tanto para el productor como para el consumidor. Podemos crearla en un paquete denominado *paquete1*.

Necesitarás importar los paquetes *javax.jms.** y *javax.naming.**. Después, dentro de la clase debes declarar las variables:

```
public String nombreCola="miCola";           // Nombre externo de la cola
public Context contexto=null;                // Contexto JNDI
public QueueConnectionFactory factoria=null; // Factoria de conexiones
public QueueConnection conexionCola=null;    // conexion
public QueueSession sesionCola=null;         // sesion
public Queue cola = null;                    // cola de mensajes
```

Las declaramos públicas por sencillez en su manejo posterior, aunque sería más correcto hacerlas privadas y facilitar métodos para su acceso. El constructor de la clase lo dejaremos vacío, y crearemos un método *inicializaCola* sin parámetros, que elevará la excepción *JMSException* y que retornará un booleano (indicando si acaba bien o mal), cuyo código principal será el siguiente:

```
if (contexto == null){
  // Aun no se ha realizado la inicializacion. Una vez realizada
  // no tiene sentido volver a realizarla.
  contexto = new InitialContext();  // Obtiene contexto JNDI
  // Obtiene factoria de conexion a colas (ha debido ser creada externamente)
  factoria = (QueueConnectionFactory) contexto.lookup("QueueConnectionFactory");
  // Obtiene la cola (ha debido ser creada externamente)
  cola = (Queue) contexto.lookup(nombreCola);
  // Ahora crea la conexion a la cola
  conexionCola=factoria.createQueueConnection();
  // Crea la sesion
  sesionCola = conexionCola.createQueueSession(false,Session.AUTO_ACKNOWLEDGE);
  conexionCola.start(); // Hay que activar la conexion para empezar.
}
```

Queda para el alumno completar el código con un mínimo tratamiento de excepciones. Si se produce algún error la variable *contexto* debe quedar a *null*.

Con el método *lookup* localizamos la factoría de conexiones de colas, con el nombre indicado, así como la cola de mensajes. Una vez obtenida la factoría, usamos el método *createQueueConnection* para crear la conexión y desde ella, la sesión, con el método *createQueueSession*. El primer argumento indica que no es *transaccional* (no usamos

transacciones), mientras que el segundo indica uso automático de ACKs para las recepciones.

Las conexiones pueden ser temporalmente detenidas con un método *stop*, y **deben** ser cerradas al concluir la aplicación (método *close*). El alumno debe escribir un método *cerrarConexion* que realizará esta operación.

c) Desde *paquete1* crearemos con *New-Java Class* una clase llamada *Productor*, cuyo contenido se rellenará como sigue:

Es necesario importar el paquete *javax.jms.**, y la clase únicamente contará con el método *main* para poderla ejecutar posteriormente. Declararemos en dicho método *main* las variables:

```
int i;              // Para enviar varios mensajes
MsgConexion mc;     // Conexion propia a cola
QueueSender emisor; // Emisor
TextMessage m;      // Mensaje a enviar
boolean ok;         // Para comprobar retorno de metodo
```

El cuerpo del programa es básicamente el siguiente:

```
mc = new MsgConexion();  // Crea objeto MsgConexion
ok=mc.inicializaCola();  // Prepara los parametros
if (ok){
  // Prepara emisor
  emisor = mc.sesionCola.createSender(mc.cola);
  // Prepara mensaje
  m=mc.sesionCola.createTextMessage(); // Crea mensaje
  // Hace varios envios
  for (i=0; i<5; i++){
   m.setText("Hola Mundo"+i);    // Contenido mensaje
   emisor.send(m);               // ENVIO MENSAJE
  }
}
```

El emisor se crea con el método *createSender*, indicando la cola como argumento, mientras que el mensaje se crea a partir de la sesión, con el método *createTextMesage*. Pueden crearse también otros tipos de mensajes:

TextMessage	Mensaje de texto (String)
MapMessage	Un conjunto de pares nombre/valor
BytesMessage	Array de bytes
StreamMessage	Array de valores primitivos
ObjectMessage	Objeto serializado

d) De forma análoga se crea la clase *Consumidor*, cuyo contenido será como se indica:

Es necesario declarar las variables:

```
int i;                  // Para recibir varios mensajes
MsgConexion mc;         // Conexion propia a cola
QueueReceiver receptor; // Receptor
TextMessage m;          // Mensaje recibido
boolean ok;             // Comprobacion retorno
```

El cuerpo de este segundo programa (método main) será básicamente:

```
mc = new MsgConexion();   // Crea su objeto MsgConexion
ok=mc.inicializaCola();   // Inicializa parametros
if (ok){
 // Prepara receptor sobre cola
 receptor = mc.sesionCola.createReceiver(mc.cola);
 // Recibe los mensajes:
 for (i=0; i<5; i++){
   m = (TextMessage)receptor.receive(1000);
   System.out.println("Mensaje recibido:" + m.getText());
 }
}
```

La primera parte es idéntica a la del productor. Después, a partir de la sesión crea el receptor con el método *createReceiver*, indicando la cola de mensajes como argumento. A partir de ahí puede recibir mensajes mediante el método *receive* del receptor.

El método *receive* puede ser invocado con un número entero como argumento, en este caso se está indicando una espera máxima (en milisegundos), transcurrida la cual se retorna *null*. Indicando como parámetro 0 la espera es indefinida, como si se invoca sin parámetros.

El alumno debe incluir el código necesario para el tratamiento de excepciones, así como para cerrar las conexiones.

e) Para probar el ejercicio debe lanzarse *Glassfish* (pestaña "Services-servers", con el botón derecho pulsar *start*). Una vez ejecutándose el servidor podremos acceder a su entorno de administración (de nuevo con el botón derecho nos permitirá lanzar el navegador con el entorno de administración).

Desde ahí podemos acceder a los recursos JMS del servidor, y crear la factoría de conexiones y la cola de mensajes, con nombres *QueueConnectionFactory* (tipo javax.jms.QueueConnectionFactory) y *miCola* (cola de mensajes).

f) Finalmente ejecutar "Run File" sobre *Productor* y después sobre *Consumidor* para probar la práctica.

2. En este segundo ejercicio realizaremos la práctica anterior en un entorno distribuido, de modo que los alumnos se dividirán en grupos de 2, para que uno de ellos actúe como Productor y el otro como Consumidor. La cola de mensajes la ubicaremos en el nodo del Consumidor, de modo que el productor necesitará localizarla, según veremos.

   Pasos a seguir:

   a) El productor debe conocer la IP del consumidor, para ello puedes usar la orden */sbin/ifconfig* en linux o *ipconfig* en windows. Llamaremos IP_CONSUMIDOR a dicha IP.

   b) Del lado del PRODUCTOR:

   - Crear un nuevo proyecto ProyectoJMS2, de forma similar a la seguida en el ejercicio anterior, y crear una clase *MsgConexion* como antes, cambiando los nombres de la factoría de colas por "*jms/factoria*" y el de la cola por "*cola*".
   - Crear la clase *Productor* exactamente igual que en el ejercicio anterior, cambiando el contenido del mensaje por "Hola Mundo Distante".

- Lanzar el servidor *Glassfish* y crear una factoría de conexión de colas llamada *jms/factoria*, pero **incluyendo (abajo) la propiedad** *addresslist* con valor *mq://IP_CONSUMIDOR:7676/jms*.
  De esta forma estamos dirigiendo nuestra petición de obtención de la cola remota a esa dirección. El puerto 7676 es el que utiliza Glassfish por defecto para el servicio JMS, puedes comprobarlo en el registro de *log* que genera el servidor al lanzarlo.
- Crear la cola local "*cola*", ésta se conecta en realidad a la ubicada en el nodo del consumidor, pero la necesitamos para enviar el mensaje.
- Dependiendo de la configuración del cortafuegos es posible que haya que detenerlo momentáneamente para probar el ejercicio.

c) Del lado del CONSUMIDOR:

- Crear un nuevo proyecto ProyectoJMS2, de forma similar a la seguida en el ejercicio anterior, y crear una clase *MsgConexion* como antes, cambiando los nombres de la factoría de colas por "*jms/factoria*" y el de la cola por "*cola*".
- Crear la clase *Consumidor* exactamente igual que en el ejercicio anterior.
- Lanzar el servidor *Glassfish* y crear una factoría de conexión de colas llamada *jms/factoria*, y crear la cola local "*cola*".
- Modificar el *receive* del consumidor para que la espera sea ahora indefinida (parámetro 0).
- Dependiendo de la configuración del cortafuegos es muy posible que haya que detenerlo momentáneamente para probar el ejercicio.

d) Ejecutar ambos programas *Productor* y *Consumidor* en sus respectivos nodos.