Copyright © Todos los Derechos Reservados - Cibertec Perú SAC



|  |  |  |
| --- | --- | --- |
| **Programa** | **:** | Java 7.0 Architect Application Developer |
| **Curso** | **:** | Implementando una Arquitectura Orientada a Mensajes |
| **Duración** | **:** | 120 minutos |

**Laboratorio Nº C2.3**

# Objetivo

Publicar mensajes como un Tópico.

# Requisitos

Los siguientes elementos son necesarios para la realización del laboratorio:

1. JDK instalado (versión 1.6+)
2. Eclipse Luna instalado y configurado
3. Maven 3.0 instalado.
4. Glassfish 3.0 instalado

# Ejecución del Laboratorio

## Importar el proyecto

1. Inicie el Eclipse y seleccione el workspace wksm02. Puede crear esta carpeta y luego seleccionarlo en caso no haberlo hecho previamente.
2. Elija la opción **File –> Import…**
3. En la ventana mostrada, seleccione Maven, luego “Existing maven project” y finalmente “Next”.
4. Clic en “Browse” y seleccione $JAAD\03\_Laboratorios\Modulo02\LABM203\_Topic.

## Creación de la factoría de conexiones y las colas

Ejecutaremos los siguientes pasos:

1. Para iniciar el glassfish nos debemos ubicar en la línea de comandos en la ruta:

$JAAD\02\_Programas\glassfish3\bin

1. Ejecutamos:

asadmin start-domain --verbose

1. Ingresamos a la ruta:

http://localhost:4848

1. En el menú lateral seleccionamos:

JMS Resources🡪Connection Factory

En el listado mostrado, elegimos “New”.

1. Completamos los datos siguientes y el resto lo dejamos con los datos por omisión:

Pool Name: jms/TopicCF

Resource Type: javax.jms.ConnectionFactory

Descripción: Factoría de conexiones JMS

Le damos clic en “OK”

1. En el menú lateral seleccionamos:

JMS Resources🡪Destination Resources

En el listado mostrado, elegimos “New”.

1. Completamos los datos siguientes y el resto lo dejamos con los datos por omisión:

JNDI Name: jms/JAADTopic

Physical Name: JAADTopic

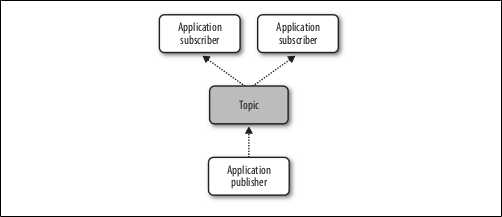
Resource Type: java.jms.Topic

Description: Topico de Mensajes

Le damos clic en “OK”

## Creando el Publicador y el Objeto a enviar

En este caso, el comportamiento del tópico se refleja por la cantidad de suscriptores:

****

Creamos primero el objeto a ser enviado como tópico:

|  |
| --- |
| package edu.cibertec.jaad.jms;  import java.io.Serializable;  public class Oferta implements Serializable{  private static final long serialVersionUID = 1L;  private String descripcion;  private Double monto;  private String producto;  // Getters/Setter, Constructores, toString  } |

Código 1 – Clase Oferta

Creamos la clase JMSPublish encargada de publicar el tópico:

|  |
| --- |
| package edu.cibertec.jaad.jms;  public class JMSPublish {  private static final int WAITING\_MSG = 60;  private static final String JMS\_TOPIC = "jms/JAADTopic";  private static final String JMS\_CONNFACT = "jms/TopicCF";  private static final Logger LOG = Logger.getLogger(JMSPublish.class);  public static void main(String[] args) {  try {  Context ctx = new InitialContext();  ConnectionFactory factory = (ConnectionFactory)ctx  .lookup(JMS\_CONNFACT);  Connection connection = factory.createConnection();  Session session = connection  .createSession(false, Session.AUTO\_ACKNOWLEDGE);    Destination topic = (Destination)ctx.lookup(JMS\_TOPIC);  connection.start();    //Enviando el mensaje  MessageProducer producer = session.createProducer(topic);  ObjectMessage msgReq = session.createObjectMessage();  msgReq.setObject(  new Oferta("Recien llegado", 20.0d, "Televisor"));  msgReq.setStringProperty("OPERACION", "Oferta");  producer.send(msgReq);    LOG.info("Mensaje enviado=" + msgReq);    producer.close();  connection.close();  System.exit(0);  } catch (Exception ex) {  LOG.error("Error al enviar/recibir el mensaje", ex);  }  }  } |

Código 2 – Clase JMSPublish

Discuta en clase el código utilizado.

## Creación de los Suscriptores

Los suscriptores serán creados como una instancia para luego ser ejecutados una cantidad configurable de estos.

Creamos la clase JMSSubscriber:

|  |
| --- |
| package edu.cibertec.jaad.jms;  public class JMSSubscriber implements MessageListener{  private static final String JMS\_TOPIC = "jms/JAADTopic";  private static final String JMS\_CONNFACT = "jms/TopicCF";  private static final Logger LOG = Logger.getLogger(JMSSubscriber.class);    private String id;  private Session session;    public JMSSubscriber(String id) {  super();  this.id = id;  }  public void start() {  try {  Context ctx = new InitialContext();  ConnectionFactory factory = (ConnectionFactory)ctx.lookup(JMS\_CONNFACT);  Connection connection = factory.createConnection();  session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);  Destination colaIN = (Destination)ctx.lookup(JMS\_TOPIC);  connection.start();  MessageConsumer consumer = session.createConsumer(colaIN);  consumer.setMessageListener(this);  LOG.info( "[" + id + "]Esperando por mensaje...");  } catch (Exception ex) {  LOG.error("Error al iniciar el lector", ex);  }  }    @Override  public void onMessage(Message message) {  try {  ObjectMessage msg = (ObjectMessage)message;  LOG.info("[" + id + "]Recibido=[" + msg.getObject() + "]");  } catch (Exception ex) {  LOG.error("Error al recibir el mensaje", ex);  }  }  public static void main(String[] args) {  JMSSubscriber rs = new JMSSubscriber("A");  rs.start();  }  } |

Código 3 - Clase JMSSubscriber

Creamos el ejecutable que cree varias instancias de estos suscriptores:

|  |
| --- |
| package edu.cibertec.jaad.jms;  public class JMSSubscribers {    public static void main(String[] args) {  int size = 3;  for (int i = 0; i < size; i++) {  JMSSubscriber subs = new JMSSubscriber("SUSB-" + i);  subs.start();  }  }  } |

Código 4 - JMSSubscribers

Para ver el funcionamiento del mismos, probaremos las siguientes dos situaciones:

* + - 1. Publicar el tópico (ejecutando JMSPublish) y luego los suscriptores (JMSSubscribers).
      2. Primero los suscriptores (JMSSubscribers) y luego publicamos el tópico (JMSPublish).

Revise el funcionamiento de cada uno de estos casos y discútalo en clase.