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 C2.1**

# Objetivo

Enviar un mensaje de forma síncrona.

# 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\LABM201\_P2P.

## 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/QueueCF

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/JAADQueue

Physical Name: JAADQueue

Resource Type: java.jms.Queue

Description: Cola de Mensajes

Le damos clic en “OK”

## Prueba de conexión con Metadatos

Creamos la clase Metada:

|  |
| --- |
| package edu.cibertec.jaad.jms;  public class MetaData {  private static Logger LOG = Logger.getLogger(MetaData.class);  public static void main(String[] args) {  try {  Context ctx = new InitialContext();  ConnectionFactory cnFactory = (ConnectionFactory) ctx  .lookup("jms/QueueCF");  Connection connection = cnFactory.createConnection();  ConnectionMetaData metadata = connection.getMetaData();  LOG.info("JMS Version:" + metadata.getJMSMajorVersion() + "."  + metadata.getJMSMinorVersion());  LOG.info("JMS Provider:" + metadata.getJMSProviderName());  LOG.info("JMS Provider Version:"  + metadata.getProviderMajorVersion() + "."  + metadata.getProviderMinorVersion());  connection.close();  //FIXME: Solo para ejecuciones locales  System.exit(0);  } catch (Exception ex) {  LOG.error("Error la procesar la metadata", ex);  }  }  } |

Código 1 – Clase MetaData

Junto con el proyecto se acompaña un archivo jndi.properties. Este archivo es el que tiene los datos de la ubicación del servidor:

|  |
| --- |
| #Propiedades especificas segun el proveedor del servicio JNDI  java.naming.factory.initial = com.sun.enterprise.naming.SerialInitContextFactory  java.naming.factory.url.pkgs = com.sun.enterprise.naming  java.naming.factory.state = com.sun.corba.ee.impl.presentation.rmi.JNDIStateFactoryImpl  org.omg.CORBA.ORBInitialHost = 0.0.0.0  org.omg.CORBA.ORBInitialPort = 3700 |

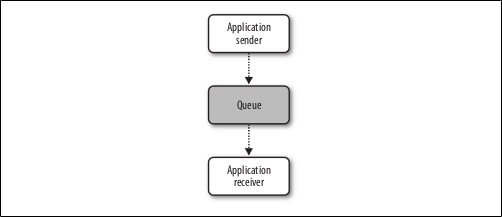
Código 2 – Archivo jndi.properties

Ejecutamos el archivo de la forma habitual y debería mostrarnos información básica de la conexión, versión del JMS, entre otros.

Discuta en clase el código utilizado.

## Creación del JMSSender y JMSReceiver

Los elementos que lo integran tienen la siguiente distribución:

****

La siguiente clase representará el Application Sender y colocará un archivo en la cola creada anteriormente:

|  |
| --- |
| package edu.cibertec.jaad.jms;  public class JMSSender {  private static Logger LOG = Logger.getLogger(JMSSender.class);  private static String CONNECTION\_FACTORY\_NAME = "jms/QueueCF";  private static String QUEUE\_NAME = "jms/JAADQueue";  public static void main(String[] args) {  try {  //Obteniendo de JNDI  Context ctx = new InitialContext();  ConnectionFactory cnFactory = (ConnectionFactory) ctx  .lookup(CONNECTION\_FACTORY\_NAME);  Destination destination = (Destination)ctx.lookup(QUEUE\_NAME);    Connection connection = cnFactory.createConnection();  Session session = connection  .createSession(false, Session.AUTO\_ACKNOWLEDGE);    connection.start();    MessageProducer producer = session.createProducer(destination);    TextMessage message = session.createTextMessage("Hola Mundo JMS");  producer.send(message);  LOG.info("Mensaje enviado=[" + message + "]");    producer.close();  session.close();  connection.close();  //FIXME: Solo para ejecuciones locales  System.exit(0);  } catch (Exception ex) {  LOG.error("Error la procesar la metadata", ex);  }  }  } |

Código 3 - JMSSender

Ahora creamos el JMSReceiver el cual hace de Application Receiver y que recibirá el mensaje:

|  |
| --- |
| package edu.cibertec.jaad.jms;  public class JMSReceiver {  private static Logger LOG = Logger.getLogger(JMSReceiver.class);  private static String CONNECTION\_FACTORY\_NAME = "jms/QueueCF";  private static String QUEUE\_NAME = "jms/JAADQueue";  public static void main(String[] args) {  try {  //Obteniendo de JNDI  Context ctx = new InitialContext();  ConnectionFactory cnFactory = (ConnectionFactory) ctx  .lookup(CONNECTION\_FACTORY\_NAME);  Destination destination = (Destination)ctx.lookup(QUEUE\_NAME);    Connection connection = cnFactory.createConnection();  Session session = connection  .createSession(false, Session.AUTO\_ACKNOWLEDGE);    connection.start();    MessageConsumer consumer = session.createConsumer(destination);    TextMessage message = (TextMessage)consumer.receive(2000);  LOG.info("Mensaje recibido=[" + message + "]");    consumer.close();  session.close();  connection.close();  //FIXME: Retirarme por favor  System.exit(0);  } catch (Exception ex) {  LOG.error("Error la procesar la metadata", ex);  }  }  } |

Código 4 - JMSReceiver

Ejecute el código de la forma habitual.

Revise el funcionamiento y discútalo en clase.

## Envío de un objeto vía JMS

Para enviar un objeto, éste debe implementar la interfaz Serializable y se debe enviar mediante el objeto ObjectMessage. Creamos el objeto a enviar:

|  |
| --- |
| package edu.cibertec.jaad.jms;  import java.io.Serializable;  public class Profesor implements Serializable{  private static final long serialVersionUID = 1L;  private String nombre;  private String dni;  // Completar getter/setters, Constructores y toString  } |

Modificamos en la clase JSMSender, en lugar de:

TextMessage message = ...

Colocamos:

ObjectMessage message = session.createObjectMessage();

message.setObject(new Profesor("Manuel", "0001"));

Modificamos el JMSReceiver, en lugar de:

TextMessage message = ...

Colocamos:

ObjectMessage message = (ObjectMessage)consumer.receive(2000);

Profesor profesor = (Profesor)message.getObject();

LOG.info("Profesor recibido=[" + profesor+ "]");

Ejecutamos de la forma habitual.

Revise el funcionamiento y discútalo en clase.