Programación Orientada a Servicios

Programación de Servicios Web con Acceso a Bases de Datos

Programa de
Ingeniería en Computación
UAM – Azcapotzalco

A cargo de:
Dra. Maricela Claudia Bravo Contreras
mcbc@correo.azc.uam.mx

CBI DIVISIÓN CIENCIAS BÁSICAS E INGENIERÍA
UAM - Azcapotzalco

Casa abierta al tiempo
Universidad Autónoma Metropolitana
Azcapotzalco

CONEXIÓN A BASE DE DATOS

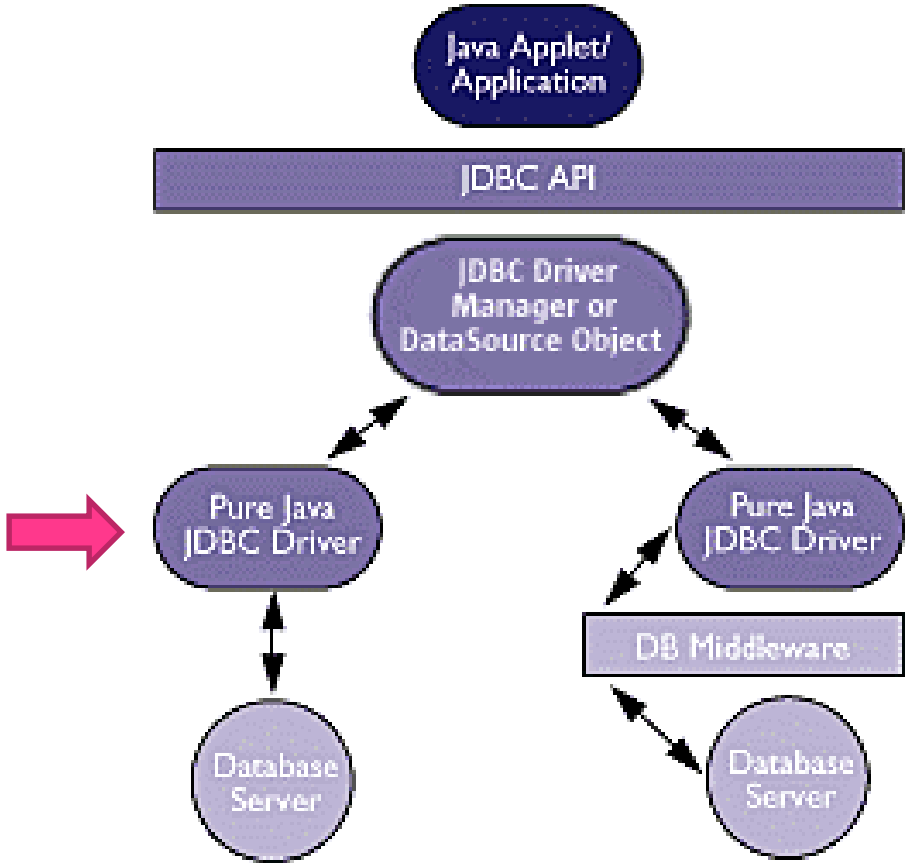
¿Qué es JDBC?

- Es una API que permite el acceso a cualquier fuente de datos "tabular" desde el lenguaje de programación Java.
- Una fuente de datos tabular puede ser desde bases de datos relacionales, hojas de cálculo hasta archivos planos.
- El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos.

Arquitectura General

Tipo 4: Driver directo a la base de datos en Java puro

Este tipo de driver convierte las llamadas JDBC en el protocolo usado directamente por el DBMS, permitiendo llamadas directas desde el cliente al servidor del DBMS.



Pasos básicos para conectarse a una base de datos con Java

1. Establecer una **conexión**
2. Crear JDBC **Statements**
3. Ejecutar **SQL** Statements
4. Obtener el **ResultSet**
5. Cerrar la conexión

1. Establecer una conexión

- **import java.sql.*;**
- **Cargar el driver específico**
  - Class.forName("oracle.jdbc.driver.OracleDriver");
    - Carga dinámica de una clase driver para la base de datos Oracle
- **Crear la conexión**
  - Connection con = DriverManager.getConnection("jdbc:oracle:thin:@oracle-prod:1521:OPROD", username, passwd);
    - Establece la conexión a la base de datos obteniendo un objeto *Connection*

2. Crear JDBC statement(s)

- Statement stmt = con.createStatement() ;
- Crea un objeto tipo Statement para enviar instrucciones SQL a la base de datos.

3. Ejecutar SQL Statements

- String createLehigh = "Create table Lehigh " +
  "(SSN Integer not null, Name VARCHAR(32), " + "Marks Integer)";
  stmt.**executeUpdate**(createLehigh);

- String insertLehigh = "Insert into Lehigh values“ +
  "(123456789,abc,100)";
  stmt.**executeUpdate**(insertLehigh);

Get ResultSet

```
String queryLehigh = "select * from Lehigh";

ResultSet rs = Stmt.executeQuery(queryLehigh);

while (rs.next()) {
   int ssn = rs.getInt("SSN");
   String name = rs.getString("NAME");
   int marks = rs.getInt("MARKS");
}
```

Close connection

- stmt.close();
- con.close();

Primer Ejemplo

- Crear una base de datos en MySQL llamada Agenda y una tabla llamada Alumnos

Nombre del atributo	Tipo de dato
Matricula	CHAR(9)
Nombre	VARCHAR(200)
Edad	INT
Genero	VARCHAR(20)

Instrucción SQL para crear una tabla

```
CREATE TABLE ALUMNOS (
        matricula CHAR(9) not NULL,
        nombre VARCHAR(200),
        edad INTEGER,
        genero VARCHAR(20),
        PRIMARY KEY ( matricula ));
```

Primer Ejemplo

```
import java.sql.*;
public class PrimerEjemplo
{
        public static void main(String[] args)
        {
            try
          {
            // Se registra el Driver de MySQL
              Class.forName("com.mysql.jdbc.Driver");

            Connection conexion = DriverManager.getConnection (
               "jdbc:mysql://localhost/Agenda","kevin", "kevin");
```

```
            Statement s = conexion.createStatement();

            ResultSet rs = s.executeQuery ("select * from alumnos");
            System.out.println("Nombre    Edad    Genero");
            while (rs.next())
            {
                System.out.println (
                        rs.getString("nombre") + " " + rs.getInt("edad") + " " +  rs.getString("genero")
                        );
            }
                    conexion.close();
             }
             catch (Exception e)
              {
                    e.printStackTrace();
               }
            }
}
```

```
public class CreaBaseDatos
{
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/";
   static final String DB_URL2 = "jdbc:mysql://localhost/students";

   static final String USER = "root";
   static final String PASS = "labsim";

   public static void main(String[] args)
   {
   Connection conn = null;
   Connection conn2 = null;
   Statement stmt = null;
   Statement stmt2 = null;

   try
   {
      Class.forName("com.mysql.jdbc.Driver");

      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      System.out.println("Creación de Base de Datos...");
      stmt = conn.createStatement();
```

```
sql = "CREATE TABLE REGISTRO " +
              "(id INTEGER not NULL, " +
              " nombre VARCHAR(255), " +
              " apellido VARCHAR(255), " +
              " edad INTEGER, " +
              " PRIMARY KEY ( id ))";

      stmt2.executeUpdate(sql);
      System.out.println("Tabla creada...");

   }catch(SQLException se){
      se.printStackTrace();
   }catch(Exception e){
      e.printStackTrace();
   }finally{
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }
   }
   System.out.println("Adios!");
}
}
```

Mapeo de tipos JDBC - Java

JDBC Type	Java Type
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT DOUBLE	double
BINARY VARBINARY LONGVARBINARY	byte[]
CHAR VARCHAR LONGVARCHAR	String

JDBC Type	Java Type
NUMERIC DECIMAL	BigDecimal
DATE	java.sql.Date
TIME TIMESTAMP	java.sql.Timestamp
CLOB	Clob*
BLOB	Blob*
ARRAY	Array*
DISTINCT	mapping of underlying type
STRUCT	Struct*
REF	Ref*
JAVA_OBJECT	underlying Java class

*SQL3 data type supported in JDBC 2.0

JDBC – Lectura de Metadatos del RS

```
public static void printRS(ResultSet rs) throws SQLException
{
   ResultSetMetaData md = rs.getMetaData();
   // Leer el número de columnas
   int nCols = md.getColumnCount();
   // Imprimir los nombres de las columnas
   for(int i=1; i < nCols; ++i)
         System.out.print( md.getColumnName( i)+",");
    / / Imprime los datos del RS
   while ( rs.next() )
   {     for(int i=1; i < nCols; ++i)
                  System.out.print( rs.getString( i)+",");
         System.out.println( rs.getString(nCols) );
   }
}
```

CONEXIÓN A BASES DE DATOS EN POSTGRESQL

¿Qué es PostgreSQL?

- Sistema de Bases de Datos Relacionales
- De código abierto
- Orientado a Objetos
- Ideas básicas acerca del funcionamiento
  - Modelo Cliente-Servidor
  - Postmaster
  - Backend
  - Clientes

PostgreSQL

- Con cerca de dos décadas de desarrollo tras él, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día
- Ofrece control de concurrencia multi-versión, soportando casi toda la sintaxis SQL
- Incluyendo subconsultas, transacciones, tipos, y funciones definidas por el usuario
- Cuenta también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python).

Comparación de PostgreSQL

	MySQL	PostgreSQL	SAP DB
Cumple con estándar SQL	Media	Alta	-
Velocidad	Media/Alta	Media	-
Integridad de Datos	No	Si	Si
Seguridad	Alta	Media	-
Soporte disparadores	No	Si	Si
Replicación	Si	Si	-
Integridad Referencial	No	Si	Si
Transacciones	Si	Si	-
Backups funcionando	Si	Si	-
Soporte Unicode	No	Si	-

Ventajas de PostgreSQL

- Estable
- Alto Rendimiento
- Flexibilidad
- Se puede extender su funcionalidad
- Gran Compatibilidad
- Permite crear o migrar aplicaciones desde Access, Visual Basic, Visual Fox Pro, Visual C/C++, Delphi para usar PostgreSQL como servidor de DB's.
- Cuenta con interfaces de Programación:
  ODBC, JDBC, C/C++, SQL Embebido, Tcl/Tk, Perl, Python, PHP.

Ejemplo PostgreSQL+ Java

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;
public class JDBCExample
{
public static void main(String[] argv)
{
    try
    {
          Class.forName("org.postgresql.Driver");
    }
    catch (ClassNotFoundException e)
    {
          System.out.println("Driver no localizado
          en el classpath!");
          e.printStackTrace();
          return;
   }
```

```
System.out.println("Driver Registrado!");
Connection connection = null;
try {
    connection = DriverManager.getConnection(
    "jdbc:postgresql://127.0.0.1:5432/testdb",
    "mkyong", "123456");   }
catch (SQLException e)
{
    System.out.println(“Falló la conexión");
    e.printStackTrace();
    return;
}
if (connection != null)
{
    System.out.println(“Conexión exitosa!");
}
else
{
    System.out.println("Falló la conexión!"); } }
}
```

DUDAS O COMENTARIOS???