# TEMA 5. MANEJO DE CONECTORES

## 5.1 INTRODUCCIÓN A J2EE

### 5.1.1 ARQUITECTURA (2 capas y 3 capas)

## 5.2 COMPONENTES J2EE

### 5.2.1 CONCEPTO DE COMPONENTE.

### 5.2.2 JAVABEANS

## 5.3 INTRODUCCIÓN A JDBC

### 2.3.1 DRIVERS, TIPOS

## 5.4 CONECTAR CON LA BASE DE DATOS

### 5.4.1 PARA ORACLE Y MYSQL

- cargar controlador
- crear conexión
- recuperar datos de la bbdd
- obtener datos de un conjunto de resultados (cursor)
- cerrar conexión

## 5.5 PATRON DAO (ver anexo tema3)

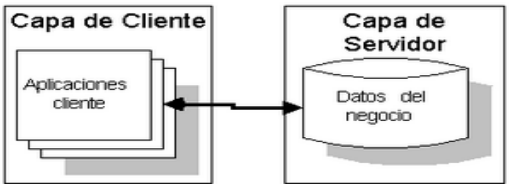
## 5.1 INTRODUCCIÓN A J2EE

J2EE (Java 2Enterprice Edition) versión ampliada de J2SE que incluye APIS necesarias para construir aplicaciones para arquitecturas distribuidas multicapa.

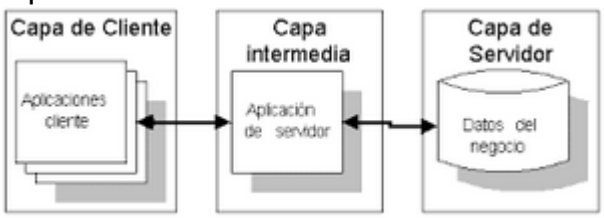
Capa→ un grupo de tecnologías que proporcionan uno o más servicios.

### 5.1.1 ARQUITECTURA (2 capas y 3 capas)

Tradicionalmente los sistemas se han venido diseñando usando el modelo cliente-servidor, entendiendo por cliente una aplicación que inicia el diálogo con otra denominada servidor para solicitarle servicios.




Actualmente, la lógica de negocio se ha convertido también en un servicio y puede residir en otro servidor, conocido como servidor de aplicaciones, dando lugar a una arquitectura 3 capas.




## 5.2 COMPONENTES J2EE

### 5.2.1 CONCEPTO DE COMPONENTE.

Definición de componente por Szyperski 1998 "*Un componente es una unidad de composición, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio"*

El objetivo de desarrollo basado en componentes es construir aplicaciones mediante ensamblado de módulos software reutilizables, que han sido diseñados previamente con independencia de las aplicaciones en las que van a ser utilizados.

Ejemplos de tecnologías de componentes son las siguientes:

- La plataforma .NET de Mocrosoft.
- JavaBeans y EJB (Enterprice JavaBeans) de Oracle Corporation.

### 5.2.2 JAVABEANS

Un JavaBeans es un componente de software reutilizable que está escrito en lenguaje java. A menudo nos referimos a un JavaBean como un Bean.

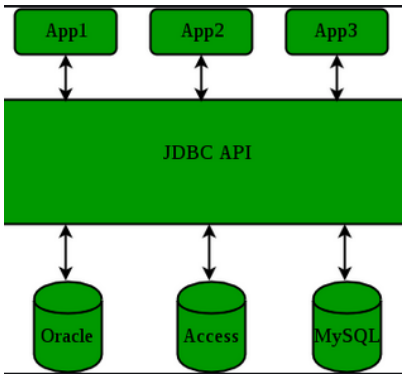
Un JavaBean ha de cumplir las siguientes características:

- **Introspección**. Mecanismo mediante el cual los propios JavaBeans proporcionan información sobre sus características (propiedades, métodos y eventos)
- **Manejo de Eventos.** Un Bean se comunica con otro Bean utilizando eventos.
- **Propiedades.**
- **Persistencia.**

## 5.3 INTRODUCCIÓN A JDBC (Java Database Connectivity)

JDBC es un API (Interfaz de Programación de Aplicaciones) de Java, proporciona un conjunto de clases que permiten ejecutar instrucciones SQL.

Para que una aplicación java pueda hacer operaciones sobre una base de datos, previamente tiene que establecer una conexión con la misma. Esta conexión se realiza a través de un controlador (Driver).




La API JDBC se proporciona en los paquetes: **java.sql y javax.sql**

### 5.3.1 DRIVERS (Controladores) y TIPOS

Un driver permite conectarse a una base de datos, cada base de datos requiere uno específico.

**ACCESS:** **Sun.jdbc.odbc.jdbc_odbcDriver**

**MySQL:** **com.mysql.jdbc.Driver**

**ORACLE:** **Oracle.jdbc.driver.oracleDriver**

**TIPOS:**

**Tipo 1: JDBC_ODBC Bridge**

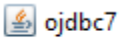
Permite el acceso a bases de datos JDBC mediante un driver ODBC. Exige la instalación y configuración de ODBC en la máquina cliente.

**Tipo 2: THIN**

Controlador de java puro con protocolo nativo. Traduce las llamadas al API de JDBC en llamadas propias del protocolo de red usado por el motor de base de datos. No existe instalación en el cliente.

### 5.4 CONECTAR CON LA BASE DE DATOS

El primer paso para crear una conexión entre una aplicación java y una base de datos es registrar el controlador JDBC adecuado. Lo da cada fabricante, en el caso de Oracle es el fichero .jar




En java será necesario importar las librerías de **java.sql.***

| Tipo de dato SQL | Tipo de dato Java |
|---|---|
| INTEGER o INT | int |
| SMALLINT | short |
| FLOAT | float |
| DOUBLE | double |
| CHARACTER(n) o CHAR(n) | String |
| VARCHAR(n) | String |
| BOOLEAN | boolean |
| DATE | java.sql.Date |
| TIME | java.sql.Time |
| TIMESTAMP | java.sql.Timestamp |
| BLOB | java.sql.Blob |

### 5.4.1 PARA ORACLE Y MYSQL

**Paso 1) Cargar controlador**

Invocar al método forName de Class que devuelve un objeto Class asociado con la clase especificada (controlador).

```
String controlador = "com.mysql.jdbc.Driver"
                   = "oracle.jdbc.driver.OracleDriver"
Class.forName(controlador);
```

**Paso 2) Crear conexión.**

Una vez registrado el controlador JDBC, se solicitará a DriverManager que proporcione una conexión para una fuente de datos.

Para ello hay que enviarle un mensaje getConnectionm método estático de la clase DriverManager.

Si DriverManager no encuentra un controlador acorde al URL especificado, lanza una excepción SQLexception. (usaremos throw para propagar las excepciones)

Las URL serán:

**Mysql:** "jdbc:mysql://127.0.0.1:3306/basesdatos", "root","");

**Oracle:** "jdbc:oracle:thin@localhost:1521:orcl","usu","pw");

```
Connection conexión=DriverManager.getConnection("URL","usuario","pw");
```

conexión será un objeto Connection que representa la conexión a la BD

**Paso 3) Recuperar datos de la bbdd.**

La interfaz Connection tiene varios métodos:

**a) createStatement** devuelve un objeto de tipo Statement, se usa para sentencias SQL sin parámetros.

**b) prepareStatement** devuelve un objeto de tipo PrepareStatement, se usa para sentencias SQL con parámetros

a) Statement sentencia= conexión.**createStatement()**;

```
sentencia = conexión.createStatement()
sentencia.execute("UPDATE alumno set nombre=""+nombre+""where
numexpdte=""+this.numexpdte+""");

//Execute se usa para INSERT, UPDATE Y DELETE no devuelven datos

Resulset resultado= sentencia.executeQuery("Select* from
alumno"+"where numexpdte=' " + numexpdte+ "'"");
//executeQuery se usa para select que devuelve objetos de tipo resulset
  que son cursores
```

b) PrepareStatement sentencia=conexión.**prepareStatement();**

```
Int e = 120;
String s="pepe";

PrepareStatement sentencia=conexión.prepareStatement("Select * from
tabla where nombre=?, sueldo=?);

sentencia.setString(1,s);
sentencia.setInt(2,e);
sentencia. execute();
```

## Paso 4) Obtener datos de un conjunto de resultados (cursor).

Los resultados de una consulta son recogidos en un objeto ResulSet, que es un cursor.
La interfaz ResulSet tiene varios métodos que devuleven un boolean(true/false)

**Métodos para moverse por las filas:**

**beforeFirst()→** mover el cursor antes de la primera fila
**first()→**mover el cursor a la primera fila
**last()→**mover el cursor a la última fila
**next()→** mover el cursor a la siguiente fila
**previous()→**mover el cursor a la fila anterior

**Métodos para obtener datos de la fila donde está el cursor:**

**getString(String)→** recupera la columna especificada por String

```
resultado.getString("nombre");
```

**getString(int)→** recupera la columna especificada por el índice.

```
resultado.getString(i); -- la primera columna tiene índice 1.
```

**getInt()→** obtener el valor de una columna de la fila actual cuando esta es int.
**getLong(), getFloat(), getDouble().**

**Ej)**

**resultado**

| nombre | apellidos |
|---|---|
| | |
| | |
| | |

```
While(resultado.next()){
  String n = resultado.getString("nombre");
  String a = resultado.getString("apellidos");
}
```

**Métodos para obtener Metadatos (datos sobre otros datos):**

La interfaz **DataBAseMetaData** contine una serie de métodos que nos permiten saber desde una aplicación las características de una determinada BD.
Mediante el método **getMetaData()** del objeto Connection, podemos obtener un objeto que implemente la interfaz anterior.

```
DataBaseMetaData  md  = conexión.getMetaData()
Resulset rs=md.getTables(null, null, null, null);
While(rs.next()){
   System.out.Println(rs.getString(3));
}
```

Algunos métodos de esta interfaz son:

**getColumnsCount()→** número de columnas del cursor

```
int ncol=resultado.getMetaData().getColumnCount();
```

**getUserName()→** devuelve el nombre del usuario actual
**getURL()→** devuelve la URL de la BD actual
**getPrimaryKeys()→** devuelve la lista de columnas que constituyen la clave primaria
**gerImportedKeys()→** devuelve las claves ajenas existentes en la tabla
**getTables()→** devuelve las tablas de una BD

## Paso 5) Cerrar conexión.

Invocamos al método **close()** del objeto Connection.

Al cerrar la conexión también se cierran los objetos Resulset y Statement.

```
If(resultado!=null)
        resultado.close();
  If(sentencia!=null)
        sentencia.close();
If(conexion!=null)
            conexión.close();
```