Conectores

Como ya sabemos, las aplicaciones que desarrollamos rara vez trabajan solas: la mayoría necesitan **almacenar y recuperar información**. A menudo, esto se consigue mediante Bases de Datos. Pero, estas Bases de Datos son externas a nuestro programa; suelen estar disponibles mediante un Servidor. Ya utilizamos en 1ºDAM el MySQL, luego esto ya te lo sabes.

Para conectarse a una Base de Datos, nuestro programa necesita de un **conector**, un componente software que permite que un programa comunicarse con dicha BBDD. La gracia del asunto es, que dichos conectores sirven para evitar que nuestro programa 'conozca' cómo es esa BBDD internamente. Tú puedes usar un conector para una BBDD de un fabricante, pero tú te comunicas con ese conector. Por tanto, el acceso a las BBDD está de alguna forma estandarizado gracias al uso de conectores.

Por ejemplo:

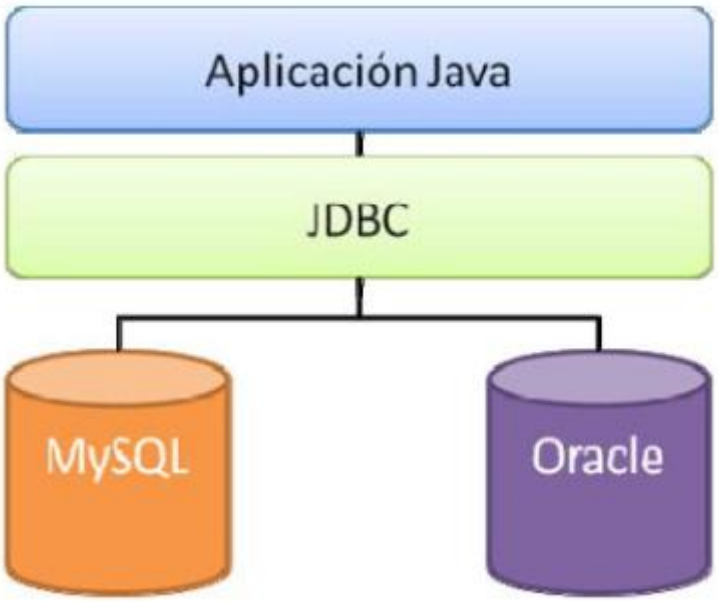
- ODBC (Open Database Connectivity): Estándar creado por Microsoft para C/C++ y .Net

- JDBC (Java Database Connectivity): Equivalente al ODBC, pero para Java y su ecosistema.

En el caso concreto de Java, **JDBC** ofrece una **API** que proporciona todos los métodos necesarios para:

- Conectarse a una base de datos.
- Enviar sentencias SQL.
- Recuperar resultados.
- Procesar los datos obtenidos.

Por tanto, mientras exista un **driver JDBC compatible**, podremos acceder a cualquier BBDD de la misma forma, sin cambiar una línea de código. La mayoría de los fabricantes ofrecen conectores para sus propias BBDD.



La arquitectura básica de **JDBC** consiste en:

- **API JDBC**: Incluida en el JDK. Proporciona las clases e interfaces necesarias para conectarse a la BBDD, como Connection, Statement, ResultSet, etc.

- **Driver JDBC**: Una librería externa que traduce las órdenes JDBC a las instrucciones específicas del SGBD.

Para poder usar JDBC en un proyecto Java, tenemos que:

1. Descargar el driver JDBC (.jar) y añadirlo al proyecto.
2. Registrar el driver JDBC en la aplicación Java
3. Establecer conexión
4. Usar la conexión
    - Preparar las sentencias SQL
    - Ejecutar la sentencia
    - Procesar resultados
5. Cerrar los recursos: resultados, statements, conexión (en orden inverso a creación).

De nuevo, esto es algo que ya hemos trabajado en **1ºDAM** en la asignatura de Programación. Por tanto, ya sabemos que o bien podemos bajar el conector a mano (**mysql-connector-java-5.1.48.jar**) o podemos incluirlo en el pom.xml de nuestro proyecto Maven.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="ht
  <modelVersion>4.0.0</modelVersion>
  <groupId>PruebaEmpleados</groupId>
  <artifactId>PruebaEmpleados</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
      <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.48</version>
    </dependency>
  </dependencies>
  <build>
    <sourceDirectory>src</sourceDirectory>
```

Conexión con la BBDD

Habitualmente, para gestionar una conexión a una BBDD mediante JDBC usamos una clase de configuración similar a esta:

```
package myProject.bbdd.config;

// Configuraciones necesarias para la Base de Datos de MySql
public class DBUtils {

    // La URL donde esta la Base de Datos. Se descompone en:
    // driver : bbd : // IP : Puerto / Schema
    public static final String URL = "jdbc:mysql://localhost:3306/myProject";

    // El Driver que vamos a usar
    public static final String DRIVER = "com.mysql.cj.jdbc.Driver";

    // Nombre y Pass de acceso a la Base de Datos
    public static final String USER = "root";
    public static final String PASS = "";

}
```

Todos estos atributos son imprescindibles para poder hacer uso del conector.

- **URL**: indica dónde se encuentra la base de Datos. Nótese que en la mayoría de los proyectos de tamaño considerable, la Base de Datos suele estar alojada en un servidor dedicado de la red. El formato para especificar dicha URL es:

  driver : bbd : // IP : Puerto / Schema

- **Driver**: El nombre del Driver que vamos a utilizar.

- **USER & PASS**: Por motivos de seguridad, las BBDD suelen estar protegidas con usuarios y password.

Una vez configurados los parámetros de acceso a la BBDD, podemos establecer una conexión:

```
public ArrayList<Client> getAll() {

    ArrayList<Client> ret = null;

    String sql = "select * from t_clients";

    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;

    try {
        Class.forName(DBUtils.DRIVER);
        connection = DriverManager.getConnection(DBUtils.URL,
            DBUtils.USER, DBUtils.PASS);
        statement = connection.createStatement();
        resultSet = statement.executeQuery(sql);
```

A tener en cuenta:

- **Class.forName(DBUtils.DRIVER)**: Búsqueda del Driver dentro del .jar del Conector, y registro del Driver con el DriverManager.

- **Connection** ofrecen un enlace a la BBDD a través del cual un programa puede leer y escribir datos.

- **DriverManager** registran los controladores JDBC y proporcionan las conexiones que permiten manejar las URL específicas de JDBC.

- **Statement** proporciona los métodos para que las sentencias SQL se ejecuten sobre la BBDD.

Uso el Statement

Se realiza en dos pasos: una preparación y una ejecución de la sentencia SQL. Podemos preparar dicha sentencia usando:

- **Statement**, que permite especificar un String con una sentencia SQL
- **PreparedStatement**, que especifica la sentencia mediante variables, de forma que es más flexible y reutilizable que **Statement**.
- **CallableStatement**, usado para llamar a procedimientos almacenados de la BBDD.

Finalmente, métodos como **executeQuery, executeUpdate o excecute** permiten lanzar la sentencia SQL contra la BBDD.

Procesar la respuesta

Si la BBDD devuelve información lo hace mediante un **ResultSet**. Esta clase en realidad es similar a una lista que contiene en cada uno de sus nodos una 'fila' producto de la ejecución de la sentencia SQL. El acceso a cada una de las 'columnas' se hace mediante su nombre. Por tanto, a modo de ejemplo:

```
while (resultSet.next()) {

    ret = ret == null? new ArrayList<Client>() : client;

    Client client = new Client();
    client.setId(resultSet.getInt("id"));
    client.setName(resultSet.getString("name"));

    ret.add(client);
}
```

Cerrar los recursos

Finalmente, es importante cerrar los recursos cuando no los utilicemos. De no hacerlo, podrían quedarse 'cogidos' y no podrían volver ser reutilizados por la aplicación (o incluso por terceras aplicaciones que usen la BBDD). Se suelen cerrar los accesos en el propio método de consulta en orden inverso al que fueron abiertos.

El ejemplo completo es:

```
public ArrayList<Client> getAll() {

    ArrayList<Client> ret = null;
    String sql = "select * from t_clients";
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName(DBUtils.DRIVER);
        connection = DriverManager.getConnection(DBUtils.URL,
            DBUtils.USER, DBUtils.PASS);
        statement = connection.createStatement();
        resultSet = statement.executeQuery(sql);

        while (resultSet.next()) {

            ret = ret == null? new ArrayList<Client>() : client;

            Client client = new Client();
            client.setId(resultSet.getInt("id"));
            client.setName(resultSet.getString("name"));
            ret.add(client);
        }
    } catch (Exception e) {
        System.out.println("Error - " + e.getMessage());
    } finally {
        // Cerramos al reves de como las abrimos
        try {
            if (resultSet != null)
                resultSet.close();
        } catch (Exception e) {
        }
        try {
            if (statement != null)
                statement.close();
        } catch (Exception e) {
        }
        try {
            if (connection != null)
                connection.close();
        } catch (Exception e) {
        }
    }
    return ret;
}
```

Práctica - 1

Instala un MySQL en tu equipo. A continuación, importa el fichero **BBDD Empresa** en tu MySQL para generar la BBDD. Crea un programa en Java que se conecte a la BBDD que has creado y realice las siguientes operaciones:

- Mostrar el contenido de la tabla departamentos.

- Mostrar los siguientes campos de los empleados del departamento 10

    - Apellido
    - Oficio
    - Salario

- Mostrar los siguientes campos del empleado con el máximo salario
    - Apellido
    - Salario
    - Departamento