Programación Frontend y Backend

BLOQUE ACCESO A DATOS

JDBC

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial
Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

01 Conceptos Básicos

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

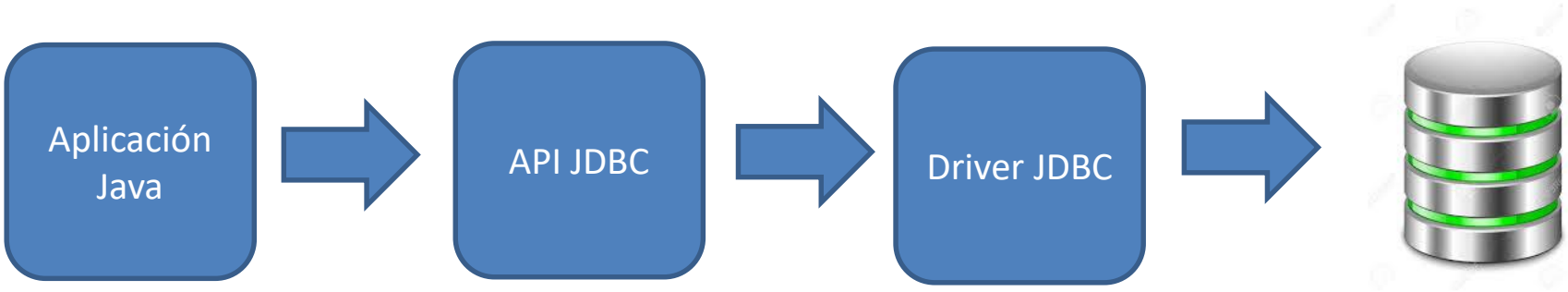
JDBC (Java DataBase Connectivity)

Para realizar las operaciones comunes de recuperar o almacenar información en una base de datos la mayoría de lenguajes de programación utilizan un API o una librería, clases auxiliares que permiten el envío de instrucciones SQL a la base de datos y la posterior manipulación de los resultados.

En el caso de Java, este API se llama **JDBC**, se trata de un API independiente del tipo de base de datos . Esto significa que podríamos cambiar con el mismo código contra una base de datos SQL server u Oracle con tan sólo cambiar la cadena de conexión y el driver utilizado.

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Esta compatibilidad es posible gracias a que JDBC no accede directamente a las BBDD, si no que en su lugar, accede a un software intermedio que traducirá las llamadas procedentes de la aplicación a instrucciones específicas de la Base de datos.



02

API JDBC

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Las clases principales:
Paquete java.sql

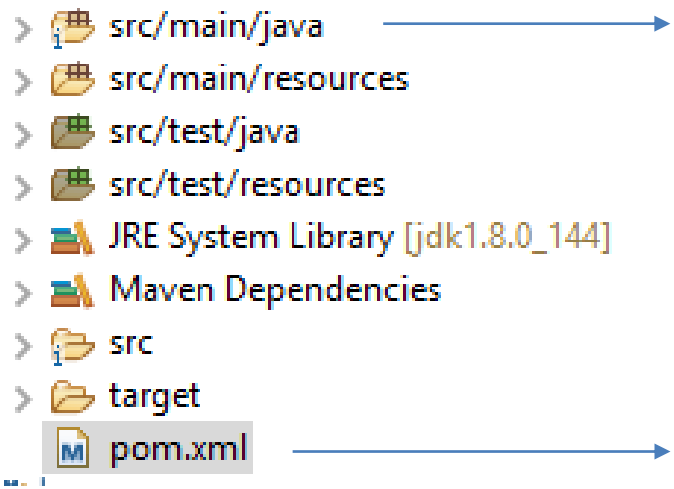
- **DrivesManager** (Es la encargada de establecer la conexión con la BD)
- **Connection** (Todas la acciones de la BD se realiza a través de este tipo de objeto)
- **Statement** (Proporciona los métodos necesarios para la comunicación con la BD)
- **PreparedStatement** (Parecido al anterior para casos concretos)
- **ResultSet** (Representa a los objetos devueltos de la BD)
- **ResultSetMetada** (Representa a los metadatos de los objetos devueltos de la BD)

Interacción con la BD

Para realizar una operación con la Base de Datos, requerirá realizar los siguiente:

- Establecer la conexión con la BD
- Ejecución de código SQL
- Recuperación del resultado (Si procede)
- Cierre de la conexión

Ejemplo



es.eoi.jdbc (Crearemos este paquete)

```
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.11</version>
</dependency>
```

Ejemplo

Conexión

Consulta SQL

Resultados

```java
package es.eoi.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Main {

    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/ej_eoi?serverTimezone=UTC";
        String user = "root";
        String pass = "1234";

        try {
            //Establecemos Conexión
            Connection conexion = DriverManager.getConnection(url, user, pass);
            System.out.println("Conectado a ".concat(url));

            //Inicializamos Statement
            Statement st = conexion.createStatement();

            //Ejecutamos y Recuperamos la consulta SQL
            ResultSet rs = st.executeQuery("SELECT * FROM CLIENTES");

            //Recorremos los datos obtenidos (si los hay)
            while (rs.next()) {
                System.out.print(rs.getString("DNI").concat(" - "));
                System.out.print(rs.getString("NOMBRE").concat(" - "));
                System.out.println(rs.getString("DIR"));
            }
```

Obteniendo la conexión

- El primer paso es crear la conexión con la BD, para ello utilizamos el método getConnection de la clase DriverManager al que le pasaremos por parámetro la URL de la BD y los datos necesarios para conectarnos.
- La URL de una base de datos depende del driver utilizado, siendo su sintaxis general:

JDBC:SUBPROTOCOLO:DATOS DE CONEXIÓN

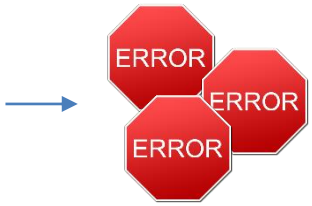
```
String url = "jdbc:mysql://localhost:3306/ej_eoi?serverTimezone=UTC";
String user = "root";
String pass = "1234";

try {
    //Establecemos Conexión
    Connection conexion = DriverManager.getConnection(url, user, pass);
    System.out.println("Conectado a ".concat(url));
```

Consultas SQL

- Tras crear la conexión en el paso anterior, el resto de operaciones a realizar serán ya independientes del tipo de base de datos y del driver que estemos utilizando.
- Utilizaremos el objeto Connection para crear un objeto Statement que nos permita enviar consultas SQL a la base de datos. Para ello deberemos invocar al método *createStatement()* proporcionado por la interfaz Connection:

Statement st = cn.createStatement();




```
//Inicializamos Statement
Statement st = conexion.createStatement();

//Ejecutamos y Recuperamos la consulta SQL
ResultSet rs = st.executeQuery("SELECT * FROM CLIENTES");
```

Statement

- Representa a una sentencia de código SQL, necesita un objeto Connection para ejecutarse, cuando ejecutamos un objeto de este tipo obtenemos un ResultSet que contiene el resultado devuelto por la BD.

- Existen 3 tipos de Statement:
  - Statement (Instrucciones SQL simples)
  - PreparedStatement (Instrucciones SQL con parámetros)
  - CallableStatement (Ejecutar Procedimientos)

Executions

- Para ejecutar una sentencia SQL necesitamos utilizar alguno de los siguientes métodos de la Interfaz Statement:

  - .execute Devuelve Utilizado para trabajar con más de un ResultSet.
  - .executeQuery Devuelve un ResultSet, se utiliza con consultas.
  - .executeUpdate Devuelve un número entero que representa el número de filas afectadas por la sentencia SQL, se utiliza para sentencias INSERT, DELETE o UPDATE.

ResultSet

- Un objeto ResultSet representa a una tabla de datos obtenida como resultado de la ejecución de una consulta, cuando accedemos a los datos de este tipo de objetos, el ResultSet lo realiza mediante el uso de un cursor.
- Inicialmente el ResultSet tiene el cursor posicionado antes de la primera fila, cuando ejecutamos .next() se mueve a la siguiente fila y devuelve true si hay más filas por recorrer o false si ya no hay más filas.

```
//Recorremos los datos obtenidos (si los hay)
while (rs.next()) {
    System.out.print(rs.getString("DNI").concat(" - "));
    System.out.print(rs.getString("NOMBRE").concat(" - "));
    System.out.println(rs.getString("DIR"));
}
```

Equivalencia de datos

SQL	Java
SMALLINT	short
INTEGER	int
INT	int
REAL	float
FLOAT	double
DOUBLE	double
BOOLEAN	boolean

Consumo de Recursos

Los objetos Connection consumen una gran cantidad de recursos del sistema, por lo que es necesario realizar el cierre de las conexiones con la base de datos una vez realizadas todas las operaciones pertinentes sobre la misma.

Si no se han cerrado explícitamente los objeto Statement y ResultSet, el cierre del objeto Connection provocará también el cierre automático de estos objetos.

SQLException

Como hemos visto anteriormente, la llamada a los distintos métodos del API JDBC puede provocar alguna excepción del tipo SQL. Esta clase dispone de tres métodos que pueden aportar información valiosa sobre los errores producidos.

getMessage() : Devuelve una cadena de caracteres con la información asociada a la excepción producida.

getErrorCode(): Devuelve un código de error específico del fabricante, por lo que habrá que consultar la información sobre el driver utilizado para conoce los posibles códigos de error que puede devolver el método y su significado.

getSqlState(): Devuelve una cadena de caracteres de cinco cifras con un código de error estandarizado que indica el estado de la sentencia SQL al producirse la excepción.

03 Parametrización

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Parametrización

Aunque podemos ejecutar sentencias SQL estáticas, fijas, que no cambian, el mayor potencial de las bases de datos consiste en la interacción con dichas sentencias, es decir no siempre nos interesará obtener todos los productos:

SELECT * FROM PRODUCTOS

Quizá necesitemos una lista de los últimos productos introducidos en la tienda o incluso obtener aquellos que valen cierto precio o cualquier criterio que nos podamos imaginar.

SELECT * FROM PRODUCTOS WHERE PRECIO > 100

Parametrización

Estos criterios pueden ser variables, imaginemos que tenemos una tienda online y nuestros usuarios pueden ver nuestros productos y filtrarlos por talla, color, precio, color y precio, precio y talla, etc …

No podemos hacer una query por cada posibilidad que tenga el usuario de consultar información en nuestra web, esto lo podemos solucionar con la parametrización de las sentencias SQL, en lugar de usar **Statement** utilizaremos **PreparedStatement**.

Cada parámetro que vayamos a introducir en la sentencia SQL se sustituirá por una "**?**"

SELECT * FROM CLIENTES WHERE DNI = ? AND NOMBRE = ?

PreparedStatement

```
//Establecemos Conexión
Connection conexion = DriverManager.getConnection(url, user, pass);
System.out.println("Conectado a ".concat(url));

StringBuilder sql = new StringBuilder();
sql.append("SELECT * ");
sql.append("FROM CLIENTES ");
sql.append("WHERE DNI = ? ");
sql.append("OR NOMBRE LIKE ? ");
System.out.println(sql.toString());

//Inicializamos PreparedStatement
PreparedStatement pst = conexion.prepareStatement(sql.toString());
pst.setString(1, "1112");
pst.setString(2, "%rio");

//Ejecutamos y Recuperamos la consulta SQL
ResultSet rs = pst.executeQuery();

//Recorremos los datos obtenidos (si los hay)
while (rs.next()) {
    System.out.print(rs.getString("DNI").concat(" - "));
    System.out.print(rs.getString("NOMBRE").concat(" - "));
    System.out.println(rs.getString("DIR"));
}
```

04 Procedimientos Almacenados

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Programación con JDBC

Procedimientos Almacenados

En ocasiones los desarrolladores de aplicaciones deciden almacenar parte del código SQL en la propia base de datos en forma de procedimientos almacenados, para posteriormente ejecutarlos desde la aplicación en lugar de generar las instrucciones de la misma.

Este sistema tiene como ventaja el aumento del rendimiento y la simplificación del código de la aplicación. Por el contrario se tiene una mayor dependencia del tipo de base de datos, dificultando la portabilidad.

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO EOI Escuela de organización industrial
Unión Europea Fondo Social Europeo Iniciativa de Empleo Juvenil El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Programación con JDBC

Procedimientos Almacenados

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SP_CLIENTES`()
BEGIN
SELECT * FROM CLIENTES;
END
```

```
1 •  call ej_eoi.SP_CLIENTES();
2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DNI	NOMBRE	DIR
1111	Mario	C/. Mayor, 3
1114	Juan	C/. Mayor, 4

Programación con JDBC

CallableStatement

La ejecución de un procedimiento almacenado desde una aplicación JDBC se realiza mediante un objeto CallableStatement el cual puede obtenerse invocando al método **prepareCall()** de la interfaz Connection.

Este método requiere como parámetro el nombre del procedimiento que se quiere ejecutar.

La interfaz CallableStatement es una subinterfaz de preparedStatement, siendo su uso bastante similar ella.

Programación con JDBC

CallableStatement

```
//Establecemos Conexión
Connection conexion = DriverManager.getConnection(url, user, pass);
System.out.println("Conectado a ".concat(url));

//Inicializamos CallableStatement
CallableStatement cst = conexion.prepareCall("CALL SP_CLIENTES");

//Ejecutamos y Recuperamos la consulta SQL
ResultSet rs = cst.executeQuery();

//Recorremos los datos obtenidos (si los hay)
while (rs.next()) {
    System.out.print(rs.getString("DNI").concat(" - "));
    System.out.print(rs.getString("NOMBRE").concat(" - "));
    System.out.println(rs.getString("DIR"));
}
```

05 Ejercicios

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Ejercicio 1 - PARTE I

- Crearemos una tabla con la siguiente estructura:
  ALUMNO (DNI, NOMBRE, APELLIDOS, EDAD)

- A continuación nos crearemos los siguientes paquetes:
  es.eoi.jdbc.entity
  es.eoi.jdbc.service
  es.eoi.jdbc.repository

- Crearemos las siguientes clases:
  es.eoi.jdbc.entity.Alumno.java (Tendrá los mismos atributos que la tabla, constructores, …)
  es.eoi.jdbc.service.AlumnoService.java
  es.eoi.jdbc.repository.AlumnoRepository.java
  es.eoi.jdbc.app.GestionInstituto.java

Ejercicio 1 - PARTE II

- La clase **AlumnoRepository.java** no tendrá ningún atributo, únicamente estará compuesta por los siguientes métodos:

  - private java.sql.Connection openConnection()
  - public Alumno findByDni(String dni)
  - public List<Alumno> findAll()
  - public boolean create(Alumno)
  - public boolean delete(String dni)
  - public boolean update(String dni, String nombre, String apellidos)

Ejercicio 1 - PARTE III

- La clase **AlumnoService.java** estará compuesta por un único atributo de la clase AlumnoRepository llamado repository, no tendrá get ni set, pero en el constructor inicializaremos nuestro atributo repository, también tendrá los siguientes métodos que a su vez utilizarán los correspondientes métodos declarados en AlumnoRepository:

  - public Alumno findByDni(String dni)
  - public List<Alumno> findAll()
  - public boolean create(Alumno)
  - public boolean delete(String dni)
  - public boolean update(String dni, String nombre, String apellidos)

Ejercicio 1 - PARTE IV

- Para probar toda la lógica anterior utilizaremos **GestionInstituto.java**, este clase será la encargada de mostrar el siguiente menú por consola, se mostrará tantas veces como sea necesario hasta pulsar 0.

```
GESTION INSTITUTO V1
------------------------------
1 – Listado Alumnos
2 – Busca Alumno (DNI)
3 – Crear Alumno
4 – Modificar Alumno
5 – Eliminar Alumno
0 – SALIR
------------------------------
```

Ejercicio 1 - PARTE V

- Cada opción de menú utilizará los métodos correspondientes de la clase AlumnoService.java, para ello se solicitará al usuario los datos necesarios para buscar, modificar, etc. En caso de producirse algún error, se le notificará al usuario, así como el resultado de cada acción que solicite, en caso de no encontrar el alumno indicado se notificará también.

FLUJO DE LA APLICACIÓN