CONEXIÓN A BASES DE DATOS RELACIONALES (MYSQL)

Acceso a Datos

3 DE NOVIEMBRE DE 2024

ERIK AMO TOQUERO

Contenido

Miniintroducción 2

Acceso a la Base de Datos 2

Métodos CRUD 3

- Inserción (Create) 3
- Edición (Update) 4
- Eliminación (Delete) 4
- Búsqueda (Read) 4
- Listado (Read) 5

Miniintroducción

Para acceder a bases de datos relacionales en Java se utiliza JDBC. Promociona una API que permite trabajar con bases de datos, siendo la misma para todas las bases, pero diferenciándose en el Driver que se instala

En nuestro caso, utilizamos MySQL Connector/J, un Driver de MySQL para acceder a sus bases de datos desde aplicaciones Java.

```
<!--Dependencia de MySQL Connector-->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.33</version>
</dependency>
```

Acceso a la Base de Datos

Para acceder a una base de datos de MySQL habrá que crear un **Conector**, un objeto de acceso a datos que permite establecer conexión entre la aplicación y la base de datos, se hará de la siguiente forma:

```
public static Connection conectar(){
    Connection con = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://" + host + ":"
+ port + "/" + dbname + "?serverTimezone=UTC", user, passwd);
    }catch (Exception e){
        System.out.println(e.getMessage());
    }
    return con;
}
```

Donde:

- **Host:** Es la dirección de la base de datos relacional
- **Port:** Puerto abierto para la conexión
- **dbname:** Nombre de la base de datos
- **user:** Nombre de usuario que puede acceder a la base
- **passwd:** Contraseña.

Estos parámetros se suelen guardar en un archivo de propiedades:

```
host=localhost
port=3306
uname=root
passwd=toor
dbname=CochesMultas
```

Métodos CRUD

CRUD son las siglas Create, Read, Update, Delete, que hacen referencia a las acciones que puede tomar una base de datos. En MySQL todas estas siguen una estructura parecida:

1. Creas un String con la acción que quieras tomar
2. Creas un PreparedStatement (un objeto de JavaSQL que prepara la sentencia SQL que vas ahacer) e introduces el String
3. Pasas los parámetros que sean convenientes
4. Ejecutas la sentencia

Cuando se busca hacer CRUD con varias clases, se crean lo que se llaman Clases DAO, Data Access Object. Cada una de ellas se forma con distintos métodos CRUD dependiendo de lo que interese al programador y de la estructura del objeto.

Para estos ejemplos, se va a operar con la clase DAO de la siguiente clase (**Aviso: Para evitar errores, debe tener un constructor vacío y sus getter y setter)**:

```
public class Coche {
    private int id;
    private String matricula;
    private String marca;
    private String modelo;
    private String tipo;

}
SQL:
CREATE TABLE Coches(
Id int unsigned auto_increment primary key,
Matricula varchar(10),
Marca varchar(20),
Modelo varchar(20),
Tipo enum('Familiar','Monovolumen','Deportivo','SUV'));
```

Inserción (Create)

La sentencia de inserción permite introducir los datos de un objeto en una tabla. Lo usual es que reciba por parámetro el objeto entero que quiera introducir a la base de datos, prepare la sentencia de inserción, introduzca los datos y ejecute la actualización. Esta ejecución devuelve un entero que guarda las filas introducidas.

```
public void guardarCoche(Coche coche) throws SQLException {
    String sql = "INSERT INTO coches (matricula, marca, modelo, tipo)
VALUES (?, ?, ?, ?)";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setString(1, coche.getMatricula());
    sentencia.setString(2, coche.getMarca());
    sentencia.setString(3, coche.getModelo());
    sentencia.setString(4, coche.getTipo());
    sentencia.executeUpdate();
}
```

Edición (Update)

La sentencia de edición o actualización permite actualizar una ocurrencia (o varias) de una tabla. Se tiene que utilizar un parámetro de condición para elegir qué editar y qué no. Para ello se utiliza el WHERE. La ejecución de la sentencia devuelve un entero que guarda el número de filas actualizadas

```
public void modificarCoche(Coche cocheAntiguo, Coche cocheNuevo)
throws SQLException {
    String sql = "UPDATE coches SET matricula = ?, marca = ?, modelo =
?, tipo = ? WHERE id = ?";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setString(1, cocheNuevo.getMatricula());
    sentencia.setString(2, cocheNuevo.getMarca());
    sentencia.setString(3, cocheNuevo.getModelo());
    sentencia.setString(4, cocheNuevo.getTipo());
    sentencia.setInt(5, cocheAntiguo.getId());
    sentencia.executeUpdate();
}
```

Eliminación (Delete)

La sentencia de eliminación permite borrar una o varias ocurrencias de una tabla en la base de datos. Necesita usar un parámetro de condición para no borrar absolutamente toda la tabla. La ejecución devuelve un entero que guarda el número de filas borradas.

```
public void eliminarCoche(Coche coche) throws SQLException {
    String sql = "DELETE FROM coches WHERE matricula = ?";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setString(1, coche.getMatricula());
    sentencia.executeUpdate();
}
```

Búsqueda (Read)

La sentencia de búsqueda permite seleccionar uno o varios objetos por medio de un valor como parámetro. La ejecución de esta sentencia devuelve un objeto de tipo ResultSet, el cual permite recoger distintos tipos de valores en función del número o el nombre de columna en el que se encuentren. Suelen devolver un objeto de la clase que se busca

```
public Coche buscarCoche(String matricula) throws SQLException {
    Coche coche = null;
    String sql = "SELECT * FROM coches WHERE matricula = ?";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    ResultSet resultado = sentencia.executeQuery();
    if (resultado.next()){ //Si encuentra algún resultado:
        coche = new Coche();
        coche.setId(resultado.getInt(1));
        coche.setMatricula(resultado.getString(2));
        coche.setMarca(resultado.getString(3));
        coche.setModelo(resultado.getString(4));
        coche.setTipo(resultado.getString(5));
    }
    return coche;
}
```

Listado (Read)

Es la misma sentencia de búsqueda, pero esta está hecha para devolver una lista de datos. Lo único que cambia es que el resultset devuelve (normalmente) más de un dato, por lo que se almacena todo el contenido en una lista

```
public List<Coche> obtenerCoches() throws SQLException {
    List<Coche> coches = new ArrayList<>();
    String sql = "SELECT * FROM coches";

    PreparedStatement sentencia = conexion.prepareStatement(sql);
    ResultSet resultado = sentencia.executeQuery();
    while (resultado.next()) {
        Coche coche = new Coche();
        coche.setId(resultado.getInt(1));
        coche.setMatricula(resultado.getString(2));
        coche.setMarca(resultado.getString(3));
        coche.setModelo(resultado.getString(4));
        coche.setTipo(resultado.getString(5));

        coches.add(coche);
    }

    return coches;
}
```