MODELO VISTA CONTROLADOR

Ejercicio de clase

ES UN PATRON DE ARQUITECTURA DE SOFTWARE MUY UTILIZADO PARA SEPARAR EL CODIGO EN VARIAS PARTES:

|  |  |  |
| --- | --- | --- |
| MODELO | VISTA | CONTROLADOR |
| Tiene la lógica de la aplicación, gestiona los datos, realiza operaciones lógicas, es el encargado en nuestro ejemplo de interactuar con la base de datos: | Es la interfaz de usuario que muestra los datos al usuario y recibe las interacciones del usuario. | Es el intermediario entre el controlador y la vista |

Hay que tener en cuenta que nos hará falta en nuestro ejemplo una clase Cliente y la clase Main.

modelo

**package** modelovistacontrolador;

**import** java.util.List;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** java.sql.PreparedStatement;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ClienteModel {

**private** Connection conn;

// Constructor que recibe la URL de la base de datos, el usuario y la contraseña

**public** ClienteModel(String dbURL, String dbUser, String dbPassword) **throws** SQLException {

// Establece la conexión con la base de datos

conn = DriverManager.*getConnection*(dbURL, dbUser, dbPassword);

}

// Método para obtener todos los clientes de la base de datos y devolverlos como una lista

**public** List<Cliente> getAllClientes() **throws** SQLException {

List<Cliente> clientes = **new** ArrayList<>(); // Inicializa una lista para almacenar los clientes

PreparedStatement stmt = conn.prepareStatement("SELECT \* FROM clientes"); // Crea una declaración SQL

// Ejecuta una consulta SQL para obtener todos los clientes de la tabla "clientes"

ResultSet rs = stmt.executeQuery();

// Itera a través de los resultados y crea objetos Cliente para cada fila

**while** (rs.next()) {

**int** id = rs.getInt("id");

String nif = rs.getString("nif");

String nombre = rs.getString("nombre");

String edad = rs.getString("edad");

clientes.add(**new** Cliente(id, nif, nombre, edad)); // Agrega cada Cliente a la lista

}

stmt.close(); // Cierra la declaración

rs.close(); // Cierra el resultado

**return** clientes; // Devuelve la lista de clientes

}

// Método para cerrar la conexión a la base de datos

**public** **void** close() **throws** SQLException {

conn.close();

}

}

|  |  |  |
| --- | --- | --- |
| **clase** | Tipo | Métodos |
| ClienteModel | MODELO | * Constructor (dbURL,dbUser,dbPassword) * Método getAllClientes() obtiene todos los clientes de bbdd los almacena en una lista. preparedStatement, resulset, while….cierra los objetos y devuelve clientes que se ha definido como lista. * Importante:   List<Cliente>  List<Cliente> clientes = **new** ArrayList<>();  TIENE LA LOGICA DE LA APLICACION |
| ClienteView | VISTA | * Método mostrarclientes, argumento lista clientes, se usan system.out.printl para mostrar los datos, cliente.getId() * importante   mostrarClientes(List<Cliente> clientes) {  **for** (Cliente cliente : clientes) {  VIENE A SER EL FRONTEND |
| ClienteController | CONTROLADOR | * definimos dos instancias una para clientemodel que llamamos model y otra instancia de clienteview que se llama view * el método constructor Constructor (dbURL,dbUser,dbPassword) * model será igual al clientemodel con los parámetros del constructor (dbURL,dbUser,dbPassword) * view será igual a new de la clase clienteview sin argumentos. * Método mostrarclientes, defines un list<clientes> con el nombre de clientes, que es igual al método getllallclientes de la instancia model y después invocamos al método mostrarclientes con el argumento clientes * ES LA INTERMEDIARIA ENTRE LA VISTA Y EL MODELO |
| Cliente | * Constructor id, nif, nombre, edad * métodos getter y setter * CLASE CON METODOS ABSTRACTOS | |
| Main | * definimos las variables dburl,dbuser, dbpassword, y usando try catch, instanciamos desde la clase ClienteController que llamamos controler que será igual a ClienteController con los parámetros creados previamente, después llamamos a mostrarclientes de la instancia controller que será igual a new clientecontroller con los parámetros dburl, dbuser y dbpassword. * EN EL MAIN INSTANCIAMOS LA CLASE CONTROLADOR | |

VISTA

**import** java.util.List; **package** modelovistacontrolador;

**import** java.sql.Connection;

**public** **class** ClienteView {

// Método para mostrar los clientes en la vista

**public** **void** mostrarClientes(List<Cliente> clientes) {

**for** (Cliente cliente : clientes) {

System.***out***.println("ID: " + cliente.getId()); // Imprime el ID del cliente

System.***out***.println("NIF: " + cliente.getNif()); // Imprime el NIF del cliente

System.***out***.println("Nombre: " + cliente.getNombre()); // Imprime el nombre del cliente

System.***out***.println("Edad: " + cliente.getEdad()); // Imprime la edad del cliente

System.***out***.println(); // Imprime una línea en blanco para separar los clientes

}

}

}

CONTROLADOR

**package** modelovistacontrolador;

**import** java.util.List;

**import** java.sql.\*;

**public** **class** ClienteController {

**private** ClienteModel model; // Instancia del modelo

**private** ClienteView view; // Instancia de la vista

// Constructor que recibe la URL de la base de datos, el usuario y la contraseña

**public** ClienteController(String dbURL, String dbUser, String dbPassword) **throws** SQLException {

model = **new** ClienteModel(dbURL, dbUser, dbPassword); // Inicializa el modelo con la conexión a la base de datos

view = **new** ClienteView(); // Inicializa la vista

}

// Método para mostrar los clientes

**public** **void** mostrarClientes() **throws** SQLException {

List<Cliente> clientes = model.getAllClientes(); // Obtiene la lista de clientes desde el modelo

view.mostrarClientes(clientes); // Muestra los clientes en la vista

}

// Método para cerrar la conexión a la base de datos

**public** **void** cerrarConexion() **throws** SQLException {

model.close(); // Cierra la conexión en el modelo

}

}

CLIENTE

**package** modelovistacontrolador;

**public** **class** Cliente {

**private** **int** id;

**private** String nif;

**private** String nombre;

**private** String edad;

**public** Cliente(**int** id, String nif, String nombre, String edad) {

**this**.id = id;

**this**.nif = nif;

**this**.nombre = nombre;

**this**.edad = edad;

}

**public** **int** getId() {

**return** id;

}

**public** String getNif() {

**return** nif;

}

**public** String getNombre() {

**return** nombre;

}

**public** String getEdad() {

**return** edad;

}

}

MAIN

**package** modelovistacontrolador;

**import** java.sql.SQLException;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

String dbURL = "jdbc:mysql://localhost/empresa"; // URL de la base de datos

String dbUser = "root"; // Usuario de la base de datos

String dbPassword = ""; // Contraseña del usuario

**try** {

ClienteController controller = **new** ClienteController(dbURL, dbUser, dbPassword); // Crea una instancia del controlador

controller.mostrarClientes(); // Llama al método del controlador para mostrar los clientes

controller.cerrarConexion(); // Llama al método del controlador para cerrar la conexión a la base de datos

} **catch** (SQLException e) {

e.printStackTrace(); // Maneja las excepciones de SQL imprimiendo el error

}

}

}

Diagrama

Descripción generada automáticamente