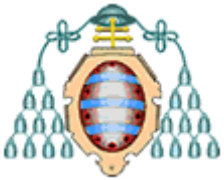
PATRONES

MVC

Ingeniería del Software

José Ramón de Diego

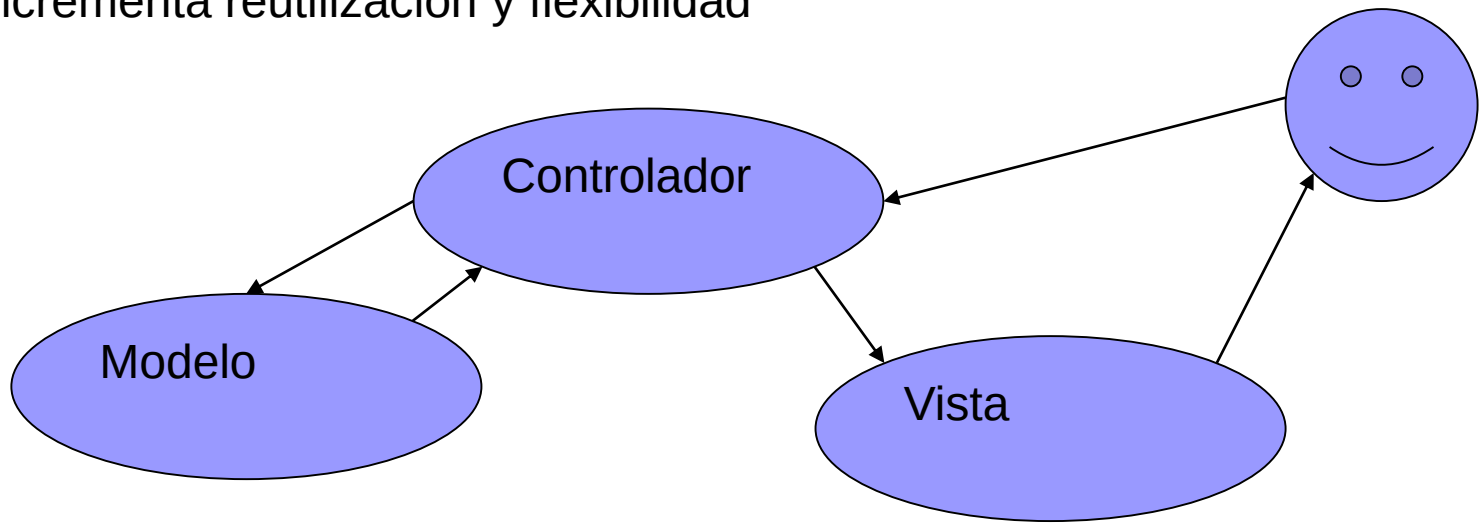


Contenido

- MVC

MVC:

- Patrón de arquitectura
- Separa la lógica de negocio de la interfaz de usuario
- Facilita la evolución por separado de ambos aspectos
- Incrementa reutilización y flexibilidad



MVC:

MODELO

- Representación específica de la información

VISTA

- Capa de la aplicación que ve el usuario

CONTROLADOR

- Controla todo lo que puede realizar nuestra aplicación

MVC:

Ejemplo:

Base de datos con una tabla donde se almacenará la información de los clientes. Desde una interfaz gráfica, un usuario podrá listar, añadir, modificar o eliminar.

MVC:

Base de datos:

Conecta con el servidor.
Métodos:
Constructor
Obtener objeto conexión

MVC:

```
package MVC;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Bd {
    private String  maquina     = "localhost";
    private String  usuario     = "root";
    private String  clave       = "";
    private int puerto          = 3306;
    private String  servidor    = "";
    private static Connection conexion  = null;

//Devuelve el objeto Connection que se usará en
la clase Controller
    public static Connection getConexion() {
        return conexion;    }

//CONSTRUCTOR
//Recibe el nombre de la base de datos
    Bd(String baseDatos){
      this.servidor="jdbc:mysql://"+this.maquina+":"+
                this.puerto+"/"+baseDatos;

//Registrar el driver
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            System.err.println("ERROR AL REGISTRAR EL
DRIVER");
            System.exit(0); //parar la ejecución       }

//Establecer la conexión con el servidor
        try {
            conexion = DriverManager.getConnection
                            (this.servidor, this.usuario,
this.clave);
        } catch (SQLException e) {
            System.err.println("ERROR AL CONECTAR
CON EL SERVIDOR");
            System.exit(0); //parar la ejecución
        }
} }
```

MVC:

Vista:

Genera el IU

MVC:

```
package MVC;

import javax.swing.JFrame;
   …
import javax.swing.JTable;

public class View extends JFrame {

//CONTENEDOR PRINCIPAL
   private JPanel contenedor;

   //DEFINICIÓN DE LAS ETIQUETAS
   private JLabel lblNombre;

         . . .
   //DEFINICIÓN DE LOS CUADROS DE TEXTO
   protected JTextField txtNombre;

   //DEFINICIÓN DE LOS BOTONES
   protected JButton btnAdd;

         . . .
   //DEFINICIÓN DE LOS OBJETOS PARA LA TABLA
   private JScrollPane scroll; //Panel de scroll que contiene la tabla

         . . .
```

MVC:

```
…
//CONSTRUCTOR
   View(){
      //Métodos de la JFrame
      setBounds(100, 100, 450, 300);
      setTitle("GESTIÓN DE CLIENTES ");
      setDefaultCloseOperation(EXIT_ON_CLOSE);

      //CREAR EL CONTENEDOR PRINCIPAL Y AÑADIRLO A LA VENTANA
      contenedor = new JPanel();
      getContentPane().add(contenedor);

      //INDICAR QUE SE QUIERE USAR SPRINGLAYOUT
      SpringLayout sp = new SpringLayout();
      contenedor.setLayout(sp);
…
```

MVC:

```
...
  //ETIQUETA NOMBRE
       lblNombre = new JLabel("Nombre:");
       contenedor.add(lblNombre);
       sp.putConstraint(SpringLayout.NORTH, lblNombre, 10,
                    SpringLayout.NORTH, contenedor);
       sp.putConstraint(SpringLayout.WEST, lblNombre,  10,
                    SpringLayout.WEST, contenedor);

      //ETIQUETA APELLIDOS
    ...
      //ETIQUETA NIF
     ...

     ...
```

MVC:

```
...
//CUADRO DE TEXTO PARA EL NOMBRE
        txtNombre       = new JTextField();
        contenedor.add(txtNombre);
        sp.putConstraint(SpringLayout.NORTH, txtNombre, 10,
                        SpringLayout.NORTH, contenedor);
        sp.putConstraint(SpringLayout.WEST, txtNombre, 100,
                        SpringLayout.WEST, contenedor);
        sp.putConstraint(SpringLayout.EAST, txtNombre, 300,
                        SpringLayout.WEST, contenedor);

        //CUADRO DE TEXTO PARA EL NIF
      ...
       //CUADRO DE TEXTO PARA LOS APELLIDOS
      ...
```

MVC:

```
...
        scroll      = new JScrollPane();
        cabecera    = new String[] {"ID","NOMBRE","NIF"};
        dtm         = new DefaultTableModel(datos,cabecera);
        tabla       = new JTable(dtm);
        scroll.setViewportView(tabla);
        //y ahora se coloca el scrollpane...
        contenedor.add(scroll); //añadir al contenedor
        sp.putConstraint(SpringLayout.NORTH, scroll, 120,
                        SpringLayout.NORTH, contenedor);
        sp.putConstraint(SpringLayout.WEST, scroll,   10,
                        SpringLayout.WEST, contenedor);
        sp.putConstraint(SpringLayout.EAST, scroll,  -10,
                        SpringLayout.EAST, contenedor);
        sp.putConstraint(SpringLayout.SOUTH, scroll, -50,
                        SpringLayout.SOUTH, contenedor);
...
```

MVC:

```
        ...
//BOTÓN AÑADIR
    btnAdd          = new JButton("Añadir");
    contenedor.add(btnAdd);
    sp.putConstraint(SpringLayout.SOUTH, btnAdd, -10,
                SpringLayout.SOUTH, contenedor);//colocarlo
    sp.putConstraint(SpringLayout.WEST, btnAdd,   60,
                SpringLayout.WEST, contenedor);
    //BOTÓN BORRAR

   //BOTÓN MODIFICAR


   //Se hace visible la ventana
    setVisible(true);

}  // Fin constructor
```

MVC:

```
    public void conectaControlador(  Controller c  ){

        btnAdd.addActionListener(c);
        btnAdd.setActionCommand("INSERTAR");

        btnDel.addActionListener(c);
        btnDel.setActionCommand("BORRAR");

        btnUpd.addActionListener(c);
        btnUpd.setActionCommand("MODIFICAR");

        tabla.addMouseListener(c);
        //sólo se permite pulsar una fila a la vez.
        tabla.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    }
}
```

MVC:

Controlador:

Comunicación entre IU y BD

MVC:

```
package MVC;

import java.awt.event.ActionEvent;
...
import java.util.Vector;


public class Controller implements ActionListener,
                                   MouseListener {
    private View view;

    //CONSTRUCTOR
    Controller( View view ){
        this.view   = view;
        cargarTabla();
    }
```

MVC:

```
…
public void actionPerformed(ActionEvent arg0) {
        //Objeto para ejecutar los procedimientos almacenados en la base de datos
        CallableStatement cs;

        String comando  = arg0.getActionCommand();
        switch (comando) {
            case "INSERTAR":
                try{
                     cs  = Bd.getConexion().prepareCall("{CALL insertarCliente(?,?,?)}");
                     cs.setString(1, this.view.txtNombre.getText());
                    cs.setString(2, this.view.txtApellido.getText());
                    cs.setString(3, this.view.txtNIF.getText());
                    //Ejecutar el procedimiento
                    cs.execute(); }catch (SQLException e) {
                    System.err.println("Error en la INSERCIÓN");
                }
            break;
            case "BORRAR":
                 …   }
       limpia();
      cargarTabla();
   }
```

MVC:

```
...
    private void limpia(){
       this.view.txtNombre.setText("");
       this.view.txtApellido.setText("");
       this.view.txtNIF.setText("");
    }

     protected void cargarTabla(){
       CallableStatement cs;
       ResultSet rs;
       Vector<Object> fila;
       for(int i=this.view.dtm.getRowCount(); i>0; i--){
          this.view.dtm.removeRow(i-1);        }

       //Cargar datos en la tabla
       try {
          cs  = Bd.getConexion().prepareCall(
                       "{CALL getClientes()}");
          //Ejecutarla y recoger el resultado
          rs  = cs.executeQuery();
          //Recorrer el resultado
          …
```

MVC:

```
...
while(rs.next()){
            //Añadir registro a registro en el vector
            fila    = new Vector<Object>();
            fila.add(rs.getInt("id"));
            fila.add(rs.getString("nombre"));
            fila.add(rs.getString("nif"));
            //Añadir el vector a la tabla de la clase View
            this.view.dtm.addRow(fila);
        }

      } catch (SQLException e) {
         System.err.println("Error al CARGAR DATOS");
      }
   }

public void mouseClicked(MouseEvent arg0) {

   ...
}
```

MVC:

Principal:

```
package MVC;

public class Principal {
    public static void main(String[] args) {

        //Invocar al constructor de la clase Bd
        new Bd("mvc");

        //Crear un objeto de la clase View
        View vista  = new View();

        //Crear un objeto de la clase Controller
        Controller controlador  = new Controller(vista);

        //Vincular la vista y el controlador
        vista.conectaControlador(controlador);

    }
}
```