

Introducción al Patrón de Diseño MVC (Model-View-Controller)

¿Qué es MVC?

- MVC es un patrón de diseño de software.
- Divide una aplicación en tres componentes principales: Modelo, Vista y Controlador.
- Mejora la separación de responsabilidades.

Componentes del MVC

1. **Modelo (Model):**

- Gestiona los datos y la lógica del negocio.
- Es independiente de la interfaz de usuario.
- Incluye la lógica de negocio.

2. **Vista (View):**

- Se encarga de la presentación de datos / muestra los datos al usuario.
- Escucha cambios en el modelo y se actualiza.

3. **Controlador (Controller):**

- Coordina la interacción entre el modelo y la vista.
- Controla el flujo de la aplicación.

Flujo de Trabajo en MVC

- El usuario interactúa con la Vista.
- La Vista notifica al Controlador.
- El Controlador actualiza el Modelo.
- El Modelo notifica a la Vista de los cambios.
- La Vista se actualiza y muestra los datos actualizados.

Ejemplo básico de MVC en Java

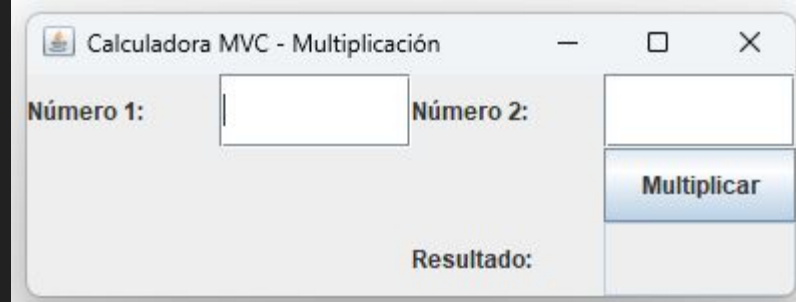
```
class Modelo {  
    private String mensaje = "Hola, Mundo";  
  
    public String getMensaje() {  
        return mensaje;  
    }  
}
```

```
class Controlador {  
    private Modelo modelo;  
    private Vista vista;  
  
    public Controlador(Modelo modelo, Vista vista) {  
        this.modelo = modelo;  
        this.vista = vista;  
    }  
  
    public void actualizarVista() {  
        String mensaje = modelo.getMensaje();  
        vista.mostrarMensaje(mensaje);  
    }  
}
```

```
// Clase Vista  
class Vista {  
    public void mostrarMensaje(String mensaje) {  
        System.out.println("Mensaje: " + mensaje);  
    }  
}
```

```
public class EjemploMVC {  
    public static void main(String[] args) {  
        Modelo modelo = new Modelo();  
        Vista vista = new Vista();  
        Controlador controlador = new Controlador(modelo, vista);  
        controlador.actualizarVista();  
    }  
}
```

Ejemplo Calculadora (Multiplicación)



Calculadora MVC - Multiplicación

Número 1:

Número 2:

Resultado:

Aplicación

```
public class Main {  
    public static void main(String[] args) {  
        // Crear el Modelo, la Vista y el Controlador  
        CalculadoraModelo modelo = new CalculadoraModelo();  
        CalculadoraVista vista = new CalculadoraVista();  
        new CalculadoraControlador(modelo, vista);  
  
        // Mostrar la Vista  
        vista.setVisible(true);  
    }  
}
```


Modelo

```
public class CalculadoraModelo {  
    // Método que realiza la multiplicación de dos números  
    public int multiplicar(int numero1, int numero2) {  
        return numero1 * numero2;  
    }  
}
```

Vista

```
import javax.swing.*;
import java.awt.*;

public class CalculadoraVista extends JFrame {

    private JTextField campoNumero1 = new JTextField(10);
    private JTextField campoNumero2 = new JTextField(10);
    private JButton botonMultiplicar = new JButton("Multiplicar");
    private JTextField campoResultado = new JTextField(10);

    public CalculadoraVista() {
        setTitle("Calculadora MVC - Multiplicación");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 150);
        setLocationRelativeTo(null);
        setLayout(new GridLayout(3, 5));

        add(new JLabel("Número 1:")); add(campoNumero1); add(new JLabel("Número 2:")); add(campoNumero2);
        add(new JLabel("")); add(new JLabel("")); add(new JLabel("")); add(botonMultiplicar);
        add(new JLabel("")); add(new JLabel("")); add(new JLabel("Resultado:")); add(campoResultado);

        campoResultado.setEditable(false);
    }
}
```

Vista

```
// Getters para los campos y botón
public int getNumero1() {
    return Integer.parseInt(campoNumero1.getText());
}

public int getNumero2() {
    return Integer.parseInt(campoNumero2.getText());
}

public void setResultado(int resultado) {
    campoResultado.setText(Integer.toString(resultado));
}

public JButton getBotonMultiplicar() {
    return botonMultiplicar;
}

public void mostrarMensajeError(String mensaje) {
    JOptionPane.showMessageDialog(this, mensaje, "Error", JOptionPane.ERROR_MESSAGE);
}
}
```

Controlador

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CalculadoraControlador {
    private CalculadoraModelo modelo;
    private CalculadoraVista vista;

    public CalculadoraControlador(CalculadoraModelo modelo, CalculadoraVista vista) {
        this.modelo = modelo;
        this.vista = vista;

        // Asigna el controlador al botón de la vista
        this.vista.getBotonMultiplicar().addActionListener(new MultiplicarListener());
    }

    // Clase interna que implementa la acción de multiplicación
    class MultiplicarListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            try {
                int numero1 = vista.getNumero1();
                int numero2 = vista.getNumero2();

                // Llamada al modelo para realizar la operación
                int resultado = modelo.multiplicar(numero1, numero2);

                // Actualiza la vista con el resultado
                vista.setResultado(resultado);
            } catch (NumberFormatException ex) {
                vista.mostrarMensajeError("Por favor ingrese números válidos.");
            }
        }
    }
}
```

Ventajas de MVC

- Separación clara de responsabilidades.
- Facilita el mantenimiento del código.
- Facilita la prueba de componentes individuales.
- Escalable y flexible.