



PRÁCTICA 03 - MVC

Patrón Modelo-Vista-Controlador (MVC)

DESARROLLO RÁPIDO DE APLICACIONES

Por Juan José Camacho Hidalgo
UNIVERSIDAD DE ALMERÍA

Contenido

1.	Introducción	2
2.	Patrón Modelo-Vista-Controlador (MVC)	2
2.1.	MVCConsoleApplication.....	2
2.2.	MVCWindowsFormsApplication	4
2.3.	MVCWpfApplication.....	5

1. Introducción

Este documento tiene como objetivo la explicación y justificación del uso del patrón Modelo-Vista-Controlador, mediante un ejemplo programado en C#.

2. Patrón Modelo-Vista-Controlador (MVC)

El patrón Modelo-Vista-Controlador (MVC) se trata de un tipo de patrón compuesto integrado en la arquitectura de software cuyo cometido es separar los datos, la interfaz de usuario y la inteligencia de control en tres componentes (modelo: datos de la aplicación, vista: interfaz de usuario, y controlador: maneja la interacción del usuario con la interfaz). Esta integrado por tres patrones, Strategy, que corresponde al controlador, Observer, que corresponde al controlador y puntualmente a la vista, y por último, a Composite, como patrón para la vista.

2.1. MVCConsoleApplication

El patrón MVC separa el modelo, de la vista y el controlador. En este caso, el programa se trata de un conversor de euros a pesetas y viceversa. Para ello hace uso de la biblioteca de clases MVCPasivoClassLibrary creada, que define una interfaz para el modelo, IModelo, donde presenta la propiedad Cantidad, que define la cantidad de moneda a convertir. Por otro lado, presenta una interfaz IVista, que como su nombre indica, se trata de la interfaz de la vista. En este caso, la vista se trata de los mensajes requeridos para la conversión por pantalla, así como reflejar la cantidad. Tanto el modelo como el controlador ya están definidos en la biblioteca de clases que se ha creado (clases Modelo y Controlador, respectivamente). Controlador es el responsable de tomar las entradas del usuario y mandarlas a Modelo para su procesamiento. Modelo implementa IModelo y cada vista que creamos tiene que tener IVista implementado.

El controlador hace uso del modelo y de la vista.

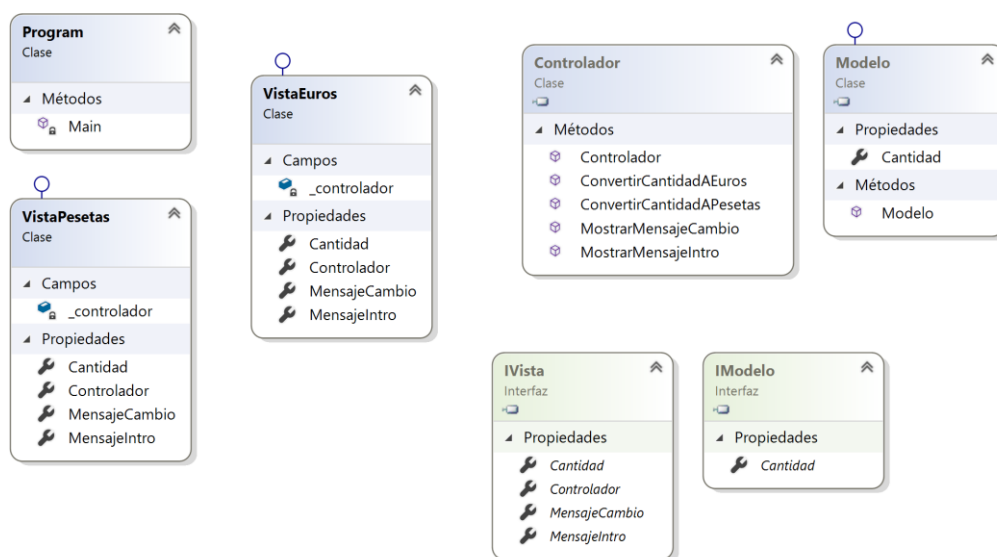



Diagrama de clases de MVCConsoleApplication

En el programa creado por consola se definen dos tipos de vistas, VistaPesetas, para la conversión a pesetas, y VistaEuros, para la conversión a euros. No se toca ni el controlador ni el modelo, ya que pertenecen a la biblioteca de clases. La vista en cuestión llama al controlador, y el controlador llama al modelo. Para hacer uso de una o de otra de las vistas, tenemos que estar comentando la línea que no deseamos usar, como podemos observar:


```
class Program
{
    0 referencias | Camacho Hidalgo, Juan Jose, Hace 1 hora | 1 autor, 1 cambio
    static void Main(string[] args)
    {
        //IVista vista = new VistaPesetas();
        IVista vista = new VistaEuros();
        IModelo modelo = new Modelo();
        Controlador controlador = new Controlador(modelo, vista);
        controlador.MostrarMensajeIntro();
    }
}
```

Aquí mostramos la ejecución de las vistas:

 C:\Users\Zéin\Desktop\DRA\Camacho Hidalgo Ju:

```
Introduzca la cantidad inicial: 30
Son 0,18 euros
```

VistaEuros

 C:\Users\Zéin\Desktop\DRA\Camacho Hidalgo Juan José (jch004

```
Introduzca la cantidad inicial: 30
Son 4991,58 pesetas
```

VistaPesetas.

2.2. MVCWindowsFormsApplication

De igual forma que la aplicación de consola, hace uso de la biblioteca de clases creada. Lo único que cambia es la vista. En este caso tenemos una única vista que incorpora los dos tipos de conversión, mediante Windows Forms. Vista implementa la interfaz IVista.

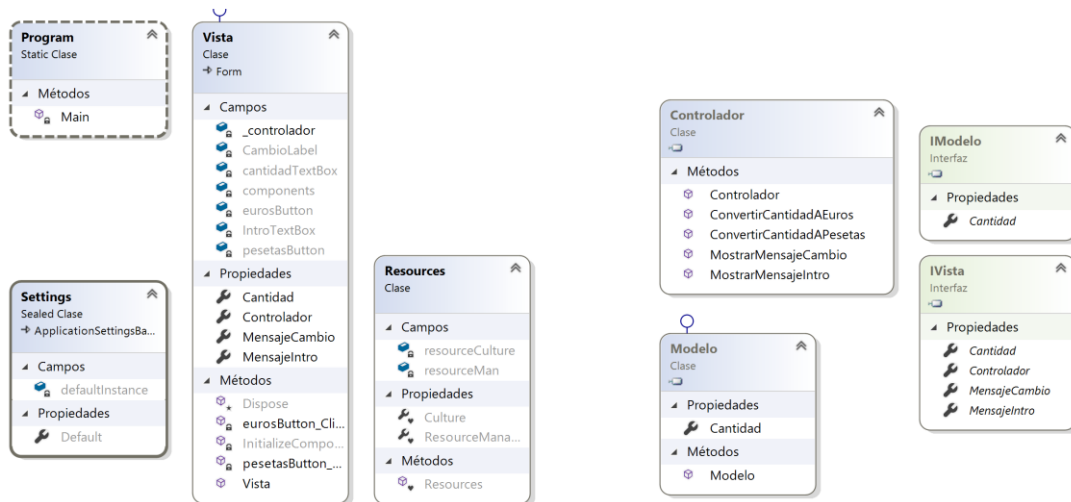


Diagrama de clases de MVCWindowsFormsApplication

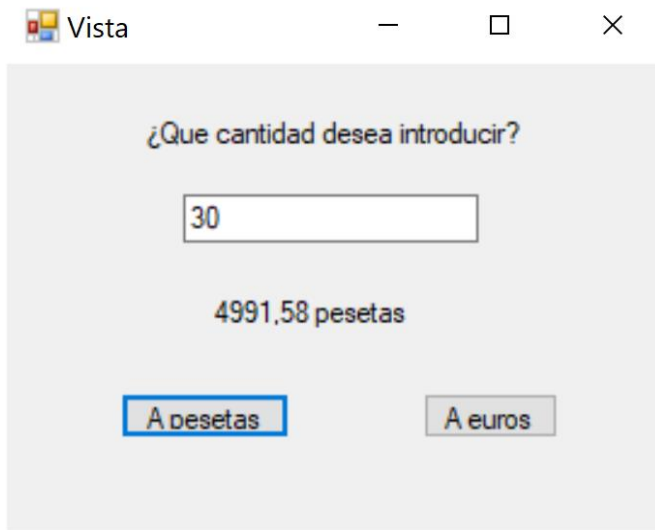
Como vemos el funcionamiento de esta vista, aparte de implementar el diseño desarrollado de Windows Forms, realiza lo mismo que la aplicación anterior. La vista llama al controlador, que se encuentra en la biblioteca de clases, y este llama al Modelo de la misma forma que antes.

```
3 referencias | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
public partial class Vista : Form, IVista
{
    private Controlador _controlador;
    1 referencia | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
    public Vista()
    {
        InitializeComponent();
    }

    5 referencias | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
    public Controlador Controlador
    {
        get
        {
            return _controlador;
        }
        set
        {
            _controlador = value;
        }
    }

    10 referencias | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
    public string Cantidad
    {
        get
        {
            return cantidadTextBox.Text;
        }
    }
}
```

Mostramos la ejecución del programa:



2.3. MVCWpfApplication

Por último, hablamos de esta nueva implementación de IVista, de la misma forma que la anterior, esta aplicación incorpora un diseño, y hace uso de la biblioteca de clases creada. Realiza exactamente lo mismo que la aplicación de Windows Forms, pero esta es WPF, y también hace uso de XAML. Se define una nueva vista (MainWindow), definida con ayuda de las clases de Resources y Settings.

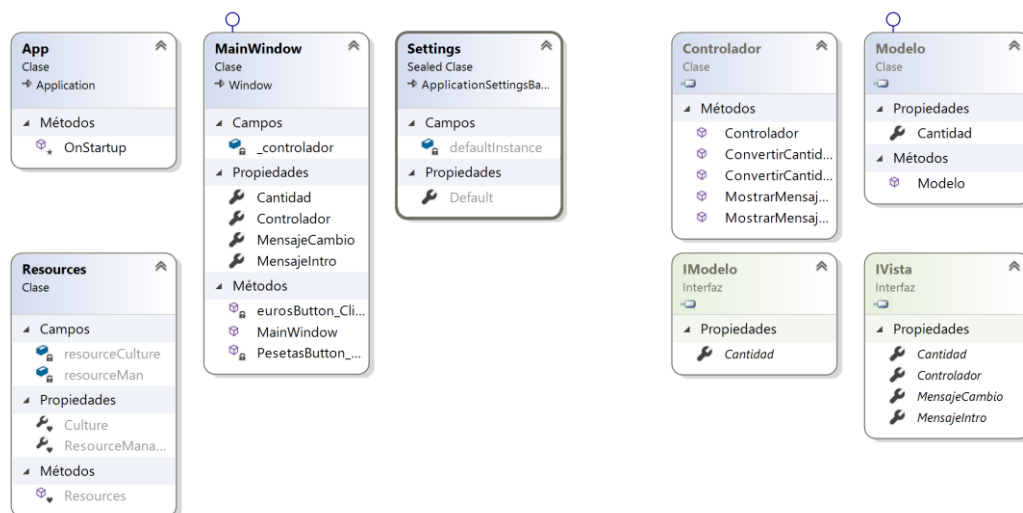


Diagrama de clases de MVCWpfApplication

De igual forma, MainWindow llama al controlador de la biblioteca de clases, y éste llama al Modelo.

```
4 referencias | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
public partial class MainWindow : Window, IVista
{
    private Controlador _controlador;
    1 referencia | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
    public MainWindow()
    {
        InitializeComponent();
    }

    5 referencias | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
    public Controlador Controlador
    {
        get
        {
            return _controlador;
        }
        set
        {
            _controlador = value;
        }
    }

    10 referencias | Camacho Hidalgo, Juan Jose, Hace 2 horas | 1 autor, 1 cambio
    public string Cantidad
    {
        get
        {
            return CantidadTextBox.Text;
        }
        set
        {
            CantidadTextBox.Text = value;
        }
    }
}
```

Mostramos la ejecución del programa:

MainWindow

¿Qué cantidad desea introducir?

30

4991,58 pesetas

A pesetas

a Euros