

[articles](#) [Q&A](#) [forums](#) [lounge](#)

Search for articles, questions, tips



Simple Example of MVC (Model View Controller) Design Pattern for Abstraction

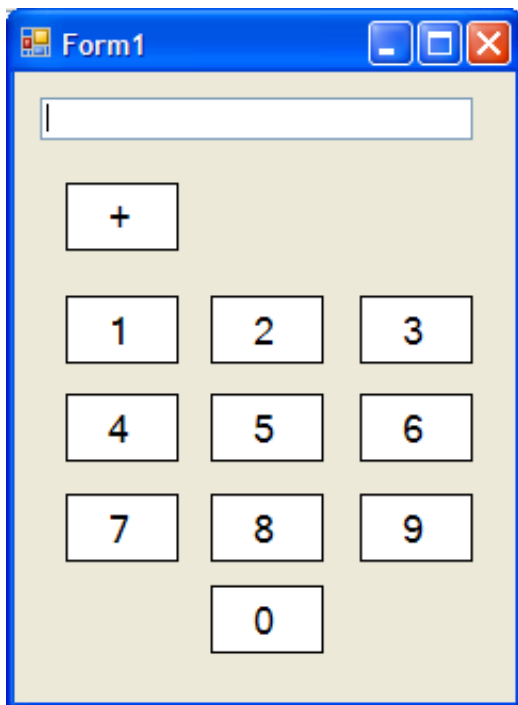


rj45, 8 Apr 2008 CPOL

Rate this:

 2.97 (38 votes)

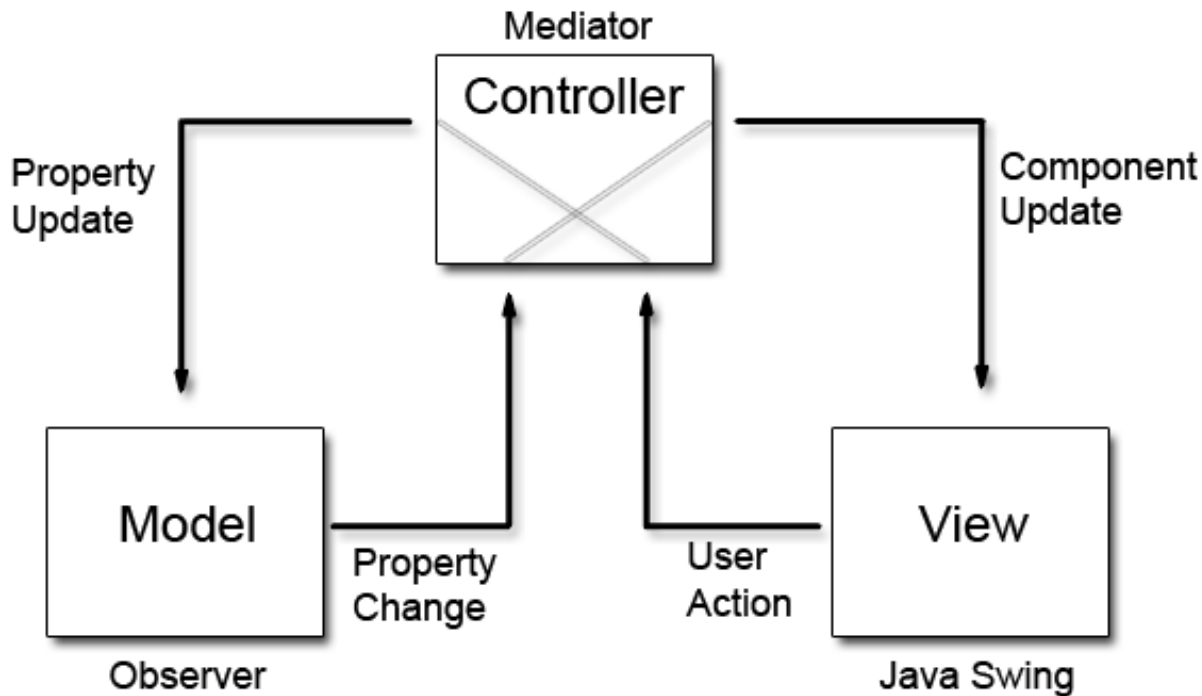
Simple example of MVC (Model View Controller) design pattern for abstraction

[Download MVC example - 22.16 KB](#)

Introduction

Model-view-controller (MVC) is a pattern used to isolate business logic from the user interface. Using MVC, the Model represents the information (the data) of the application and the business rules used to manipulate the data, the View corresponds to elements of the user interface such as text, checkbox items, and so forth, and the Controller manages details involving the communication between the model and view. The controller

handles user actions such as keystrokes and mouse movements and pipes them into the model or view as required.



Background

- [Java SE Application Design With MVC](#)

Using the Code

Note: I strongly recommend you download the code to view it, it will be much easier.

Here I will show an example of our good old friend calculator in a MVC architecture. A brief overview; the Form will house the view and events will be passed to the controller who will then call methods on our model such as ADD/Subtract/NumberPress. The model takes care of all the work and it holds the current state of the calculator. The tough thing about MVC is where to slice it apart can be confusing. The end goal is a pluggable UI and perhaps multiple controllers attached to the same model. So one way to test if you did it right is to quickly write another UI and plug it in.

A typical MVC patterns instantiation looks something like the following. A few important things to notice; the controller takes an interface to the view and model. It is important to know that the view will typically interact with the controller if it needs notification of events which are fired via the view (such as a button click). In this case, I have the controllers constructor pass a reference to itself to the view class.

```

static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        // Note: The view should not send to the model but it is often useful
        // for the view to receive update event information from the model.
        // However you should not update the model from the view.
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        frmCalcView view = new frmCalcView();
        CalculatorModel model = new CalculatorModel();
        CalcController controller = new CalcController(model, view);
        Application.Run(view);
    }
}
/// <summary>
/// The controller process the user requests.
/// Based on the user request, the Controller calls methods in the View and
/// Model to accomplish the requested action.
/// </summary>
class CalcController : IController
{
    ICalcModel model;
    ICalcView view;

    public CalcController( ICalcModel model, ICalcView view)
    {
        this.model = model;
        this.view = view;
        this.view.AddListener(this); // Pass controller to view here.
    }

    public void OnClick( int number )
    {
        view.Total = model.SetInput(number).ToString();
    }

    public void OnAdd()
    {
        model.ChangeToAddState();
    }
}

```

Notice that the view does not interact with the model, it simply receives update requests from the controller. The controller will access the view through the Total property. The view also passes click events on to the controller.

This View shouldn't know about the Controller, except we need to give it notification of some events so we pass in a **IController**. We will invoke event handlers on the controller via **IController**.

Hide Shrink ▲ Copy Code

```

/// <summary>
/// Windows Form that will host our MVC based functionality.
///
/// </summary>
public partial class frmCalcView : Form, ICalcView
{
    IController controller;

```

```

public frmCalcView( )
{
    InitializeComponent();
}
/// <summary>
/// The view needs to interact with the controller to pass the click events
/// This could be done with delegates instead.
/// </summary>
/// <param name="controller"></param>
public void AddListener( IController controller )
{
    this.controller = controller;
}
private void lbl_Click(object sender, EventArgs e)
{
    // Get the text out of the label to determine the letter and pass the
    // click info to the controller to distribute.
    controller.OnClick((Int32.Parse(((Label)sender).Text)));
}
private void lblPlus_Click(object sender, EventArgs e)
{
    controller.OnAdd();
}

#region ICalcView Members
public string Total
{
    get
    {
        return textBox1.Text;
    }
    set
    {
        textBox1.Text = value;
    }
}
#endregion
}

```

Now we will take a look at the model. Notice that it should do the "work" of the calculator and it handles the state.

Hide Shrink ▲ Copy Code

```

/// <summary>
/// Calculator model, The model is independent of the user interface.
/// It doesn't know if it's being used from a text-based, graphical, or web interface
/// This particular model holds the state of the application and the current value.
/// The current value is updated by SetInput
/// </summary>
class CalculatorModel : ICalcModel
{
    public enum States { NoOperation, Add, Subtract };
    States state;
    int currentValue;
    public States State
    {
        set { state = value; }
    }
    public int SetInput ( int number )
    {
        if (state == States.NoOperation)
        {
            currentValue = number;
        }
    }
}

```

```
    }  
    else if (state == States.Add)  
    {  
        currentValue = Add(currentValue , number );  
    }  
    return currentValue;  
}  
public void ChangeToAddState()  
{  
    this.state = States.Add;  
}  
public int Add( int value1, int value2 )  
{  
    return value1 + value2;  
}  
public int Subtract(int value1, int value2)  
{  
    throw new System.ApplicationException(" Not implemented yet");  
}  
}
```

Other Information

MVC is often seen in web applications, where the view is the actual HTML page, and the controller is the code that gathers dynamic data and generates the content within the HTML. Finally, the model is represented by the actual content, usually stored in a database or XML files, and the business rules that transform that content based on user actions.

Though MVC comes in different flavors, control flow generally works as follows:

1. The user interacts with the user interface in some way (e.g. presses a button).
2. A controller handles the input event from the user interface, often via a registered handler or callback.
3. The controller notifies the model of the user action, possibly resulting in a change in the model's state. (e.g. controller updates user's shopping cart).
4. A view uses the model (indirectly) to generate an appropriate user interface (e.g. the view produces a screen listing the shopping cart contents). The view gets its own data from the model. The model has no direct knowledge of the view.
5. The user interface waits for further user interactions, which begins the cycle anew.

By decoupling models and views, MVC helps to reduce the complexity in architectural design, and to increase flexibility and reuse.

History

- 8th April, 2008: Initial post

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License](#)

(CPOL)

Share

EMAIL

TWITTER

About the Author

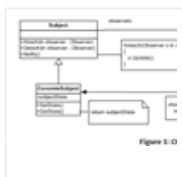
**rj45**

Software Developer (Senior)

Canada 🇨🇦

No Biography provided

You may also be interested in...



Model View Controller, Model View Presenter, and Model View ViewModel Design Patterns



Enterprise Integration Patterns Flash Cards



RIATasks: A Simple Silverlight CRUD Example (using View Model)



Integrating HL7 into Medical Applications with LEADTOOLS



Design Patterns 3 of 3 - Behavioral Design Patterns



Using Intel® RealSense™ Technology in combination with the Intel® Edison Development Platform

Comments and Discussions

You must [Sign In](#) to use this message board.

Search Comments

Go

First Prev Next

multiple digit input for addition 

Member 10754466 20-May-14 9:30

Add Clear , Add Division, Add Multiplication 

kelvinct 24-Nov-13 16:18

thx your example , i try to make subtract function working . it is done 

kelvinct 24-Nov-13 15:19

My vote of 1 

sreejith 20-Mar-13 1:40

My vote of 5 

zzfima 17-Jul-12 0:58

My vote of 4 

warlie 15-Sep-10 4:05

new architecture 

heartsofsea 2-Jul-10 4:54

Where is the Obverser pattern ? 

sks1607 14-Jun-10 0:32

Re: Where is the Obverser pattern ? 

heartsofsea 2-Jul-10 4:57

One way, but I think there is a better way to do MVC. 

Member 4538437 20-Feb-10 9:54

Re: One way, but I think there is a better way to do MVC. 

heartsofsea 2-Jul-10 4:57

Access incompatibility 

Aldo Abril Ricalde 13-Jan-09 6:47

Re: Access incompatibility 

rj45 13-Jan-09 8:47

Re: Access incompatibility 

Aldo Abril Ricalde 13-Jan-09 15:26

MVC or MVP


ppopadiyn 7-Dec-08 0:44

Re: MVC or MVP 

rj45 13-Jan-09 8:46

what about web application

Member 360350 28-Jun-08 12:39

Re: what about web application 

rj45 23-Feb-09 13:30

Does Controller be aware of the operation?

vaibhavgadian 5-May-08 11:33

Re: Does Controller be aware of the operation? 

rj45 5-May-08 11:47

Re: Does Controller be aware of the operation? [modified] 

vaibhavgadian 5-May-08 12:18

I Don't Understand

John Simmons / outlaw programmer 8-Apr-08 23:55

Re: I Don't Understand 

rj45 9-Apr-08 9:43

Refresh

1

 General  News  Suggestion  Question  Bug  Answer  Joke  Rant  Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | [Mobile](#)
Web04 | 2.8.150624.2 | Last Updated 8 Apr 2008

Select Language | ▼

Layout: [fixed](#) | [fluid](#)

Article Copyright 2008 by rj45
Everything else Copyright © [CodeProject](#), 1999-2015