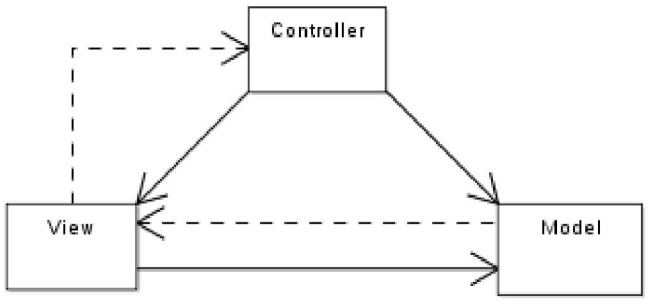
XÂY DỰNG PHẦN MỀM THEO MÔ HÌNH PHÂN LỚP

GV: ThS Phạm Thi Vương

NỘI DUNG

- Giới thiệu
- Mô hình dữ liệu (model)
- Giao diện (view)
- Điều khiển (controller)
- ASP.NET MVC

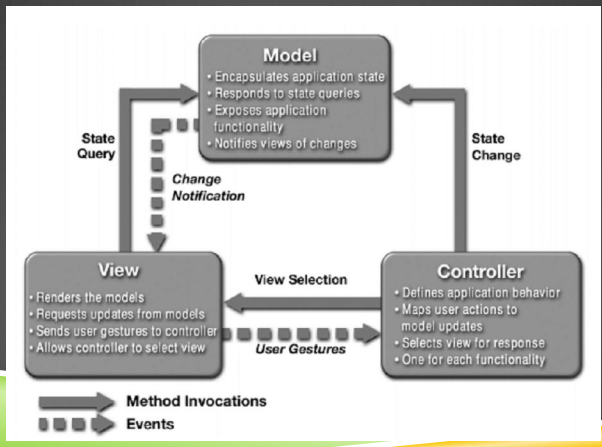


MVC PATTERN

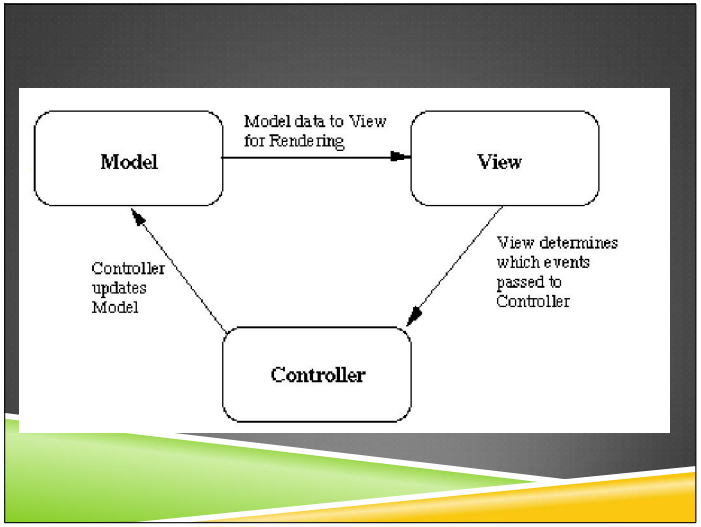
Model - View - Controller (MVC) is an architectural pattern used in software engineering.

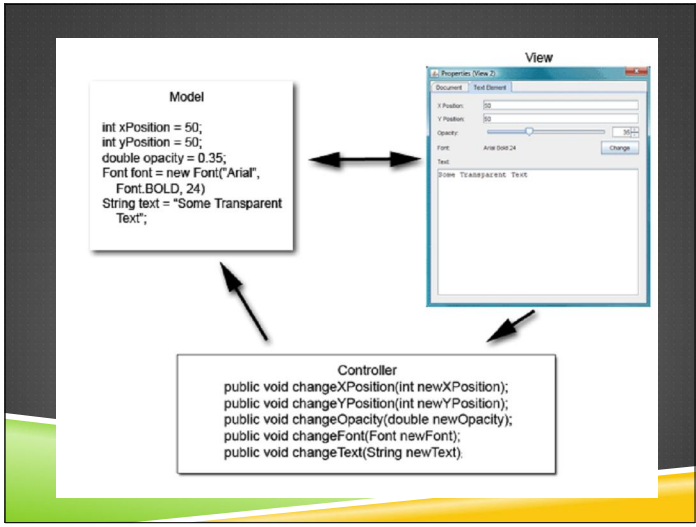
The MVC pattern, as the name implies, consists of three elements:

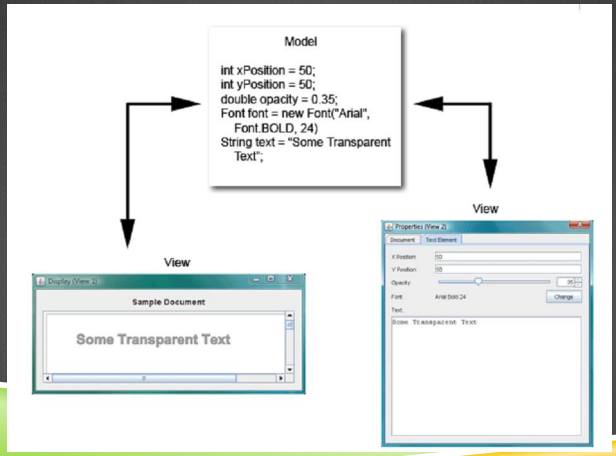
- Model: contains the application data and logic to manage the state of the application.
- View: present the user interface.
- Controller: Handles user inputs to change the state on the application.

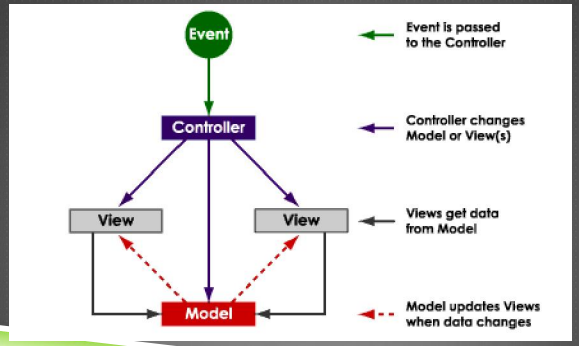


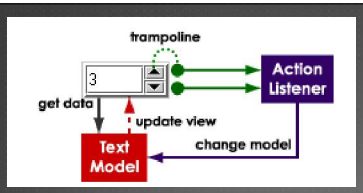
- **Model** -- *State* data for each component. Different for each component
  - *E.g.* Scrollbar – Max and minimum Values, Current Position, width of *thumb* position relative to values
  - *E.g.* Menu – Simple List of items.
  - Model data is *always* independent of visual representation.
- **View** -- How component is painted on the screen Can vary between platforms/look and feel
- **Controller** -- dictates how component interacts with events Many forms of events -- mouse clicks, keyboard, focus, repaint *etc.*.
  - Controller decides how component reacts to an event – if at all

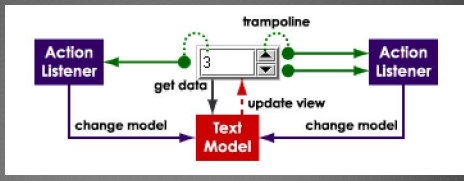




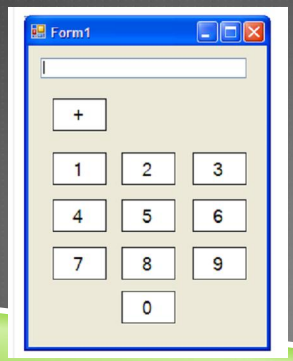


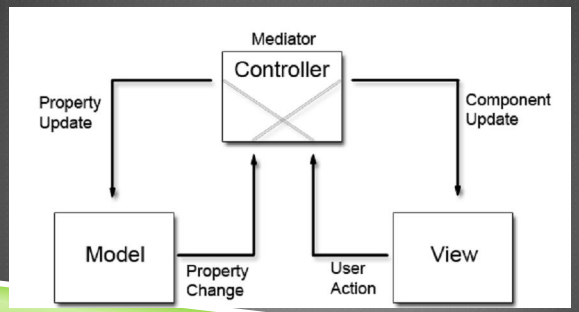






EXAMPLE






```
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();
    }
}
```

```
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
}
```

```
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");
    }
```

```
interface ICalcView
  {
     void AddListener( IController controller );

     string Total
     {
        get;
        set;
     }
  };
```

```
public interface IController
  {
     void OnClick(int number);
     void OnAdd();
  }
```

```
interface ICalcModel
  {
        int SetInput(int number);

        int Add(int value1, int value2);

        int Subtract(int value1, int value2);


        void ChangeToAddState();

  };
```

MVC

- The user interacts with the user interface in some way (e.g. presses a button).
- A controller handles the input event from the user interface, often via a registered handler or callback.
- 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).
- 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.
- The user interface waits for further user interactions, which begins the cycle anew.

MVC is often seen in web applications: :

- **View --** the HTML or XHTML generated by the application.
- **Controller**
  - receives GET or POST input and decides what to do with it
  - handing over to domain objects (i.e. the model)
- **Model --** domain objects that contain the business rules

26

MVC FRAMEWORK

- **ASP**
  - ASP Xtreme Evolution (AXE)
  - CodeCharge Studio
- **C++**
  - Wt - Web toolkit A library and application server for web applications using a desktop-like event-driven MVC pattern.
  - CppCMS - It is C++ MVC framework that had taken many ideas from Django.
- **Flex**
  - Cairngorm one of the primary open source frameworks for application architecture in Adobe Flex.
  - PureMVC ActionScript 3 MVC framework for Flex, Flash and AIR development.

MVC FRAMEWORK

Java MVC web application frameworks:

- JSF
- Oracle Application Development Framework
- Oracle Application Framework
- PureMVC, a framework for Java
- Spring MVC Framework
- Struts
- Struts2
- Wavemaker, a WYSIWYG development platform for Ajax web applications.[7]
- WebObjects
- WebWork
- .....

MVC FRAMEWORK

- **JavaScript** MVC web application frameworks:
  - SproutCore
  - PureMVC Framework for JavaScript
  - JavascriptMVC Javascript MVC framework based upon jQuery core.
  - cMVC is an MVC framework based upon Dojo Toolkit.

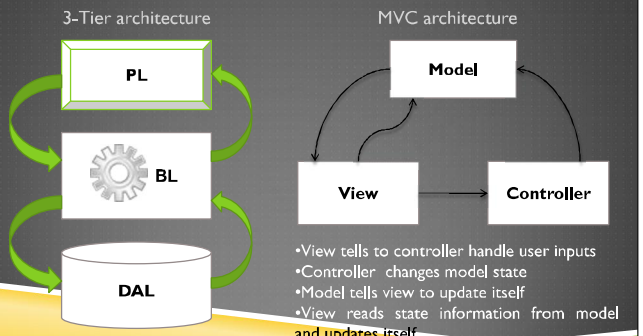
MVC FRAMEWORK

- **.NET**
  - ASP.NET MVC Framework
  - Bistro Framework
  - CodeCharge Studio
  - FubuMVC
  - Maverick.NET
  - MonoRail An ActionPack inspired MVC framework from the Castle Project
  - Naked Objects MVC
  - PureMVC Framework for C#
  - Spring Framework.NET

MVC FRAMEWORK

- **PHP**
  - CakePHP A web application framework modeled after the concepts of Ruby on Rails.
  - Drupal An open source content management system that uses MVC for its add-ons called modules.
  - eZ Publish Based on eZ Components is an object-oriented web application framework written in PHP that separates its functionality into several abstraction layers, following a strict MVC approach.
  - Joomla! v1.5.x is an open source content management system that employs the MVC model for its extensions, called components and modules.
  - Kohana v2.x is an open source MVC framework, while v3.x is HMVC (both supported).
  - PureMVC A framework for PHP.
  - Qcodo An open-source PHP 5 web application framework.
  - SilverStripe Contains a fully fledged PHP 5.2 ORM/MVC framework focused on building websites.
  - Spawn Framework An open source PHP5/OOP MVC framework
  - Symfony Framework A PHP 5 MVC framework modeled after the concepts of Ruby on Rails.
  - Yii An open source, object-oriented, high-performance component-based PHP web application framework.
  - Zend Framework An open source PHP 5-based framework featuring an MVC layer and a broad-spectrum of loosely coupled components.
  - .....

COMPARISON 3-TIER WITH MVC

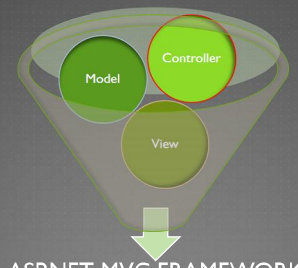


- View tells to controller handle user inputs
- Controller changes model state
- Model tells view to update itself
- View reads state information from model and updates itself.

32

ASP.NET MVC

ASP.NET MVC



MODEL ?

- **Models** trong các ứng dụng dựa trên **MVC** là những thành phần có nhiệm vụ lưu trữ thông tin, trạng thái của các đối tượng, thông thường nó là một lớp được ánh xạ từ một bảng trong **CSDL**. Lấy ví dụ, chúng ta có lớp Disc được sử dụng để mô tả dữ liệu từ bảng *Disc* trong **SQL**, bao gồm *DiscID*, *Title*…

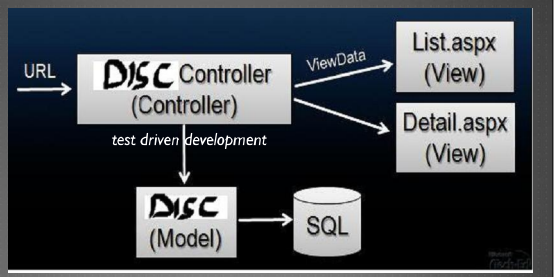
VIEWS ?

- **Views**, nó chính là các thành phần chịu trách nhiệm hiển thị các thông tin lên cho người dùng thông qua giao diện. Thông thường, các thông tin cần hiển thị được lấy từ thành phần Models. Ví dụ, đối tượng Disc có một "Edit" view bao gồm các textboxes, các dropdowns và checkboxes để chỉnh sửa các thuộc tính của sản phẩm; có một "Display" view gồm 2 dòng, cột dòng là DiscID, dòng sau là Title… để xem thông tin về sản phẩm.

CONTROLLER ?

- Controllers trong các ứng dụng kiểu MVC chịu trách nhiệm xử lý các tác động về mặt giao diện, các thao tác đối với models, và cuối cùng là chọn một view thích hợp để hiển thị ra màn hình. Trong kiến trúc MVC, view chỉ có tác dụng hiển thị giao diện mà thôi, còn điều kiển dòng nhập xuất của người dùng vẫn do Controllers đảm trách.

SỰ LIÊN KẾT GIỮA M-V-C



6. CẤU TRÚC THƯ MỤC I PROJECT MVC ASP.NET

1. Cấu trúc của một website sử dụng MVC Asp.net

NHỮNG HỖ TRỢ CỦA MVC ASP.NET

- MVC framework hỗ trợ sử dụng các các tập tin .ASPX, .ASCX và .Master như là thành phần View, điều đó có nghĩa là bạn vẫn có thể sử dụng các tính năng của ASP.NET như master pages, <%= %> snippets, server controls, templates, data-binding, localization...
- Tuy nhiên nó không sử dụng mô hình post-back từ giao diện gửi đến server nữa, thay vào đó, bạn có thể chủ động đưa những post-back từ giao diện đó đến thẳng lớp Controller. Tóm lại, không còn viewstate hay là page lifecycle còn tồn tại trong mô hình MVC.