ASP.NET MVC

@{ Introduction & Guidelines }

Speaker Introduction

Dev Raj Gautam

Application & Database Specialist

Nutrition Innovation Lab: Nepal

Active Contributor @ASP.NET Community

http://blogs.devgautam.com.np/

Have been in industry since 2009. Worked with government, local companies, outsourcing companies and Ingo's .

Agenda

- Introduction to MVC
- What Does MVC Offer?
- Demo
- Routing & Web API.
- Web Forms Or MVC?

Introduction

Asp.Net MVC1
Released on Mar 13, 2009

ASP.NET Web Forms is not part of ASP.NET 5

I was already there from 70's

MVC

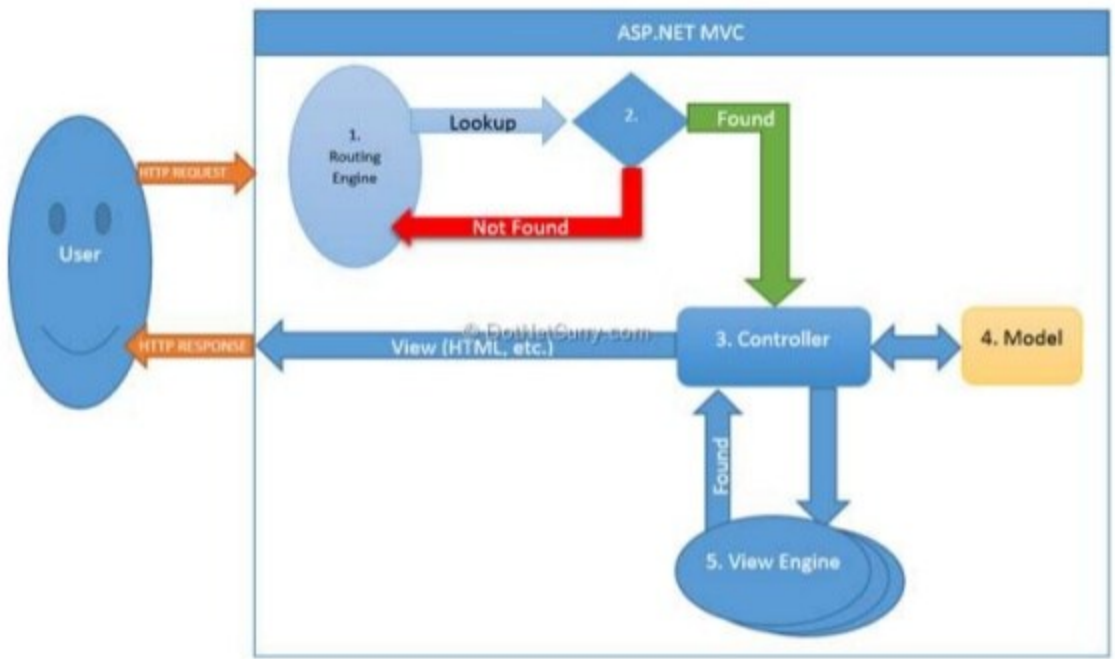
- Stands for Model View Controller. Common design pattern for 

- MVC is a software architectural pattern that follows the 'Separation of Concerns' principle. This principle divides an application into three interconnected parts as Model, View and Controller each with very specific set of responsibilities. The goal of this pattern is that each of these parts can be developed and tested in relative isolation and then combine together to create a very robust application.

Model-View-Controller

- ***Model***: These are the classes that contain data. They can practically be any class that can be instantiated and can provide some data. These can be entity framework generated entities, collection, generics or even generic collections too.
- ***Controllers***: These are the classes that will be invoked on user requests. The main tasks of these are to generate the model class object, pass them to some view and tell the view to generate markup and render it on user browser.
- ***Views***: The part of the application that handles user interaction. Typically controllers read data from a view, control user input, and send input data to the model.

How MVC Works?



What does MVC Offer?

Performance

- ASP.NET MVC don’t have support for view state, so there will not be any automatic state management which reduces the page size and so gain the performance.

Separation Of Concern

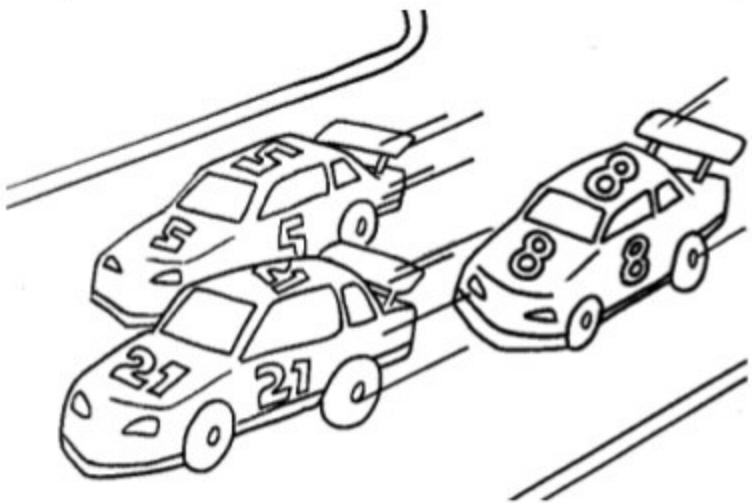
- When we talk about Views and Controllers, their ownership itself explains separation.The Views are just the presentation form of an application, it does not need to know specifically about the requests coming from the Controller. The Model is independent of Views and Controllers, it only holds business entities that can be ed to any View by the Controller as per the needs for exposing them to the end user. The Controller is independent of Views and Models, its sole purpose is to handle requests and them on as per the routes defined and as per the needs of the rendering Views. Thus our business entities (Model), business logic (Controllers) and presentation logic (Views) lie in logical/physical layers independent of each other.

Existing ASP.NET Features

- ASP.NET MVC framework is built on top of matured ASP.NET framework and thus provides developer to use many good features such as forms authentication, windows authentication, caching, session and profile state management etc.

Parallel Development

- In ASP.NET MVC layers are loosely coupled with each other, so one developer can work on Controller ,at the same time other on View and third developer on Model. This is called parallel development.



Full Control Over <HTML></HTML>

- ASP.NET MVC doesn’t support server controls, only option available is using html input controls, so we will be sure about final html rendered at the end. We will also be aware about 'id' of every element. And so integration of ASP.NET MVC application with third party JavaScript libraries like jQuery becomes easy.

Testing

- Asp.Net webforms is dependent on HttpContext.Current, our pages as part of the HttpApplication executing the request Which makes
- These are all easily mockable!
  - HttpContextBase, HttpResponseBase, HttpRequestBase

```
[TestMethod]
public void ShowPostsDisplayPostView()
{
   BlogController controller = new BlogController(…);

   var result = controller.ShowPost(2) as ViewResult;

   Assert.IsNotNull(result);
   Assert.AreEqual(result.ViewData["Message"], "Hello");
}
```

Testing

```
[TestMethod]
public void TestInvalidCredentials()
{
    LoginController controller = new LoginController();

    var mock = new Mock<System.Web.Security.MembershipProvider>();
    mock.Expect(m => m.ValidateUser("", "")).Returns(false);
    controller.MembershipProviderInstance = mock.Object;

    var result = controller.Authenticate("", "") as ViewResult;

    Assert.IsNotNull(result);
    Assert.AreEqual(result.ViewName, "Index");
    Assert.AreEqual(controller.ViewData["ErrorMessage"], "Invalid credentials!
Please verify your username and password.");
}
```

Clean URLS

- ASP.NET MVC is so tightly integrated with the ASP.NET Routing engine, it is simple to take control over the URLs that your application exposes.

Would you use:

/Products.aspx?CategoryID={3F2504E0-4F89-11D3-9A0C-0305E82C3301}

Or:

/Products/Books

Demo

What's In Demo

- Understanding the MVC Project Structure
- Understanding and Modifying the Layout and Templeates
- CRUD with project template
- CRUD DIY with code
- Basics of Routing
- Basics of Web API

Routing & Web API

Routing

- Routing maps request URL to a specific controller action using a Routing Table
- Routing Engine uses routing rules that are defined in *Global.asax* file in order to parse the URL and find out the path of corresponding controller.

WEB API

- Restful framework for Building HTTP Based services
- What is Rest? “Rest is an architectural style that abstracts the architectural elements within a distributed hypermedia system”
- https://api.twitter.com/1.1/statuses/retweets/2.json
- Web API is not MVC

What In the World Is WCF Then? HUH!!!!

Use When?

Web API

- Only working with HTTP
- Want RESTful endpoints
- Don’t need multiple protocols
- Don’t expose SOAP endpoints
- Want Simple Configuration

WCF

- Needs to support multiple protocols
- Need to expose SOAP endpoints
- Need Complex Configuration
- Need RPC Style Requests

Web Forms or MVC?

Web Forms or MVC?

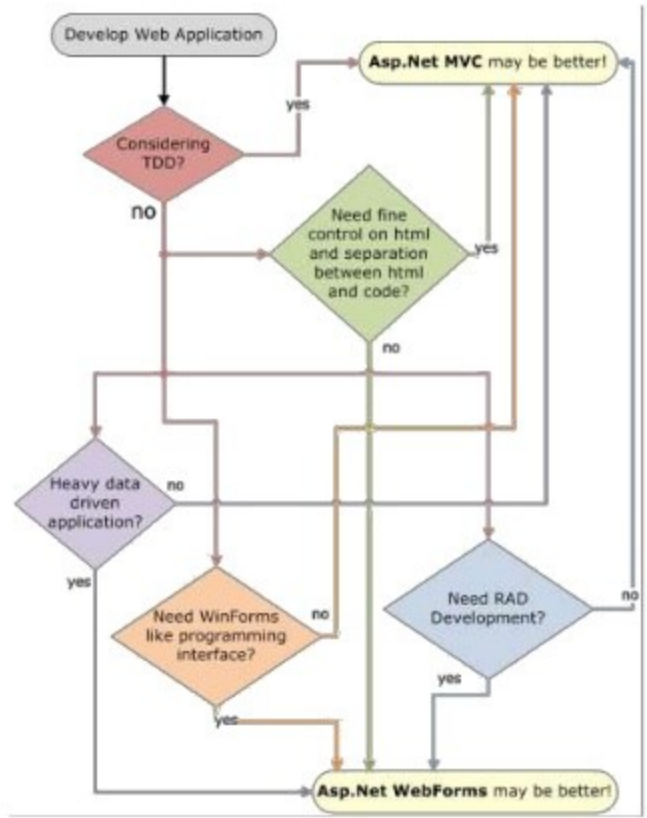
Web Froms

- Winforms-alike event model
- Familiar controls & Rich
- Familiar = rapid application development
- Functionality per page
- Uses viewstate for state management
- Less control over rendered HTML
- Event Driven Programming
- Oh wow! I can turn of or control View State Size.

MVC

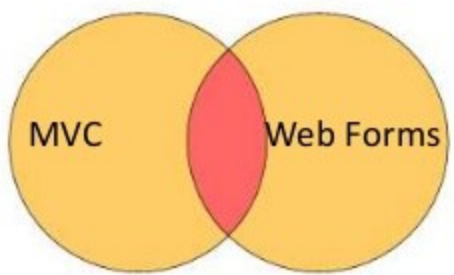
- Less complex: separation of concerns
- Easier parallel development
- TDD support
- No viewstate, …
- Full control over behavior and HTML
- Extensive Javascript
- Makes you think

Web Forms or MVC?



Web Forms or MVC?

- Mixing the Technologies can also do wonders for some solutions



Verdict?

- Please decide on your own on the basis of previous slides.

Thank You