



Razor Views

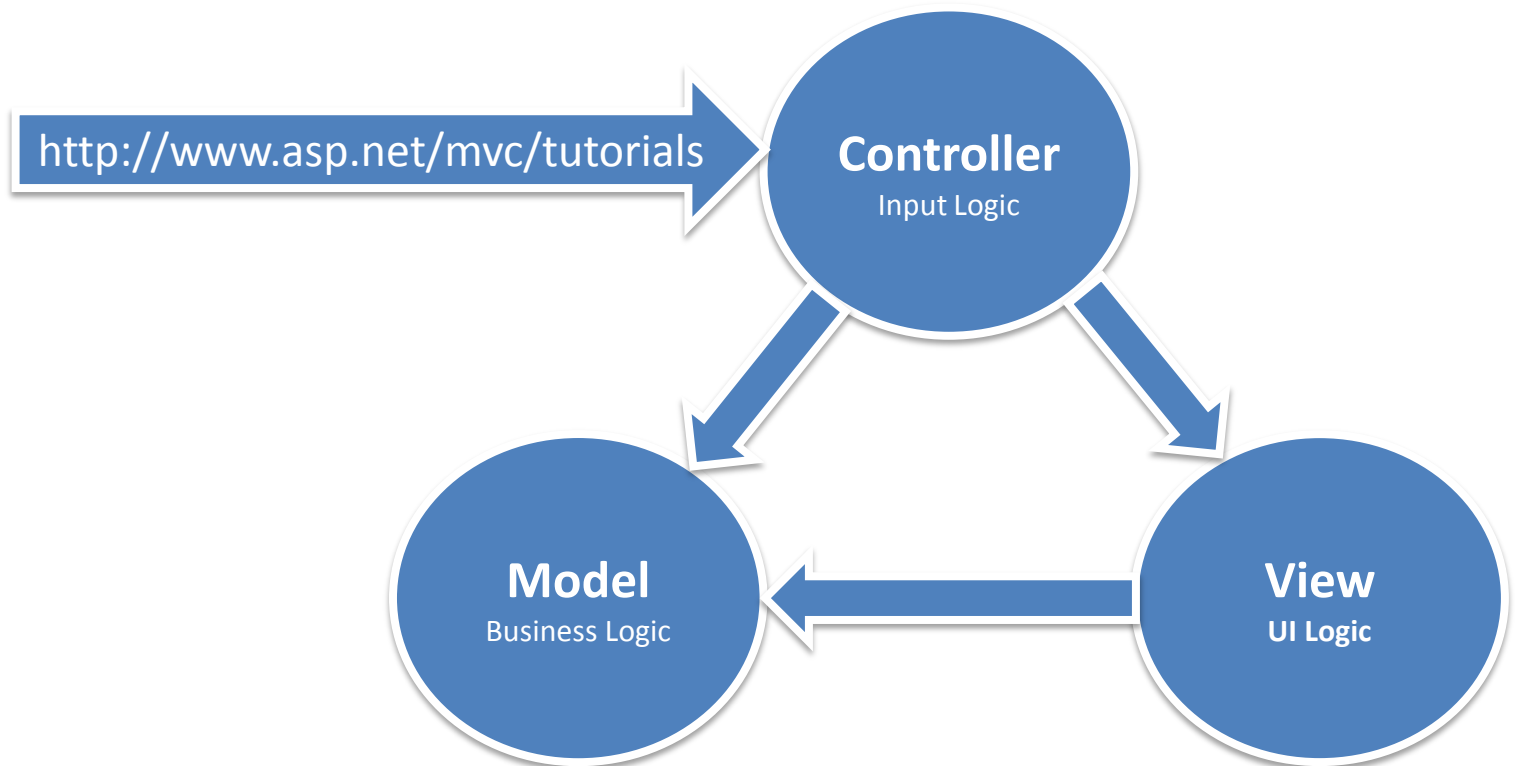
Author and Presenter : Sushant B.

Agenda

- Introduction
- Razor Syntax
- View Conventions
- Passing Data to Views
- Strongly Typed Views
- View Models
- Partial Views.

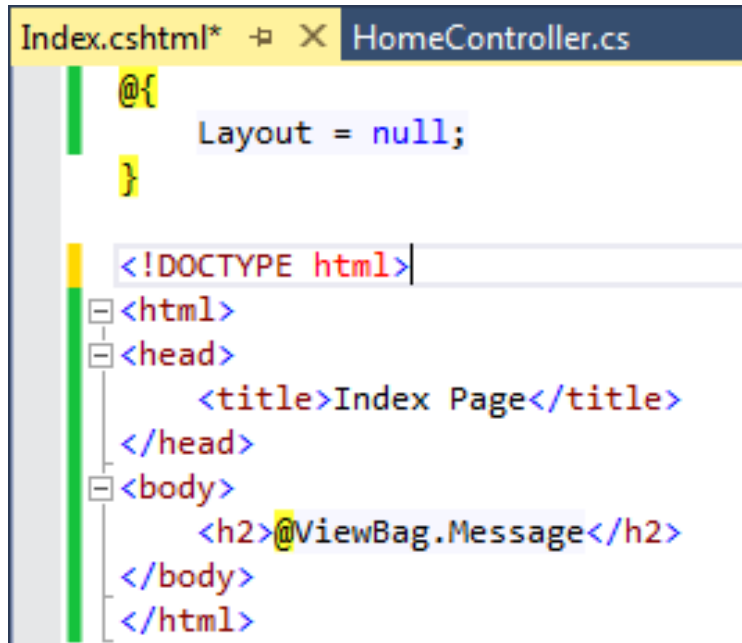


MVC – A Design Pattern



What is Razor View

- A template that generates HTML response
- A clean, lightweight and simple view engine
- A Razor view has .cshtml or .vbhtml file extensions
- Contains HTML along with data



```
@{
    Layout = null;
}

<!DOCTYPE html>
<html>
<head>
    <title>Index Page</title>
</head>
<body>
    <h2>@ViewBag.Message</h2>
</body>
</html>
```

Razor Syntax

- Razor view uses razor syntax
- @ is a key transition character
 - It switches execution between C# and HTML
- Code blocks
 - Can contain multiple C# lines of code
- Code expression
 - Inline C# expression

```
@{  
    ViewBag.Title = "Index";  
    var client = "Microsoft";  
}
```

Code Block

```
<h3>Client's Name is @client.</h3>
```

Code Expression

View Conventions

- Folder name : controller name – controller
- Above folder must be inside Views folder
- Overriding conventions is possible

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.Message = "Hello MVC";
        return View("About");
    }
}
```

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.Message = "Hello MVC";
        return View(@"~\Views\MyFolder\MyView.cshtml");
    }
}
```

- While returning view, can we pass data to view?

Passing Data to View

- ViewData
 - Dictionary object
 - Manually type casting is required
- ViewBag
 - Dynamic wrapper around ViewData
 - Dynamically type casted
- TempData
 - Can be used for subsequent request
 - Can pass data from one controller/action to another controller/action
 - Life is little more than ViewData and ViewBag
- Session
 - Data is available for multiple requests
- None of them are strongly typed.

Demo

Why Strong Typing

- ViewData/ViewBag produces performance issues
 - Internally stores data as object
 - Unboxing is required to read data
- Not a type safe technique
 - Compiler can't check type
 - Might be casted to wrong type
 - May be using wrong key name to retrieve value
- View doesn't know about incoming data
 - No connection between data passed and data received
 - View developer must remember what is passed
- Solution? Create strongly typed view.

Strongly Typed View

- When view knows about data type
- @ is used to write C# code
 - @model declares what is passed
 - @Model can access passed data
 - @using directive declares namespace
- Advantages
 - Compiler can detect errors
 - VS gives intellisense support
- View models are very useful.

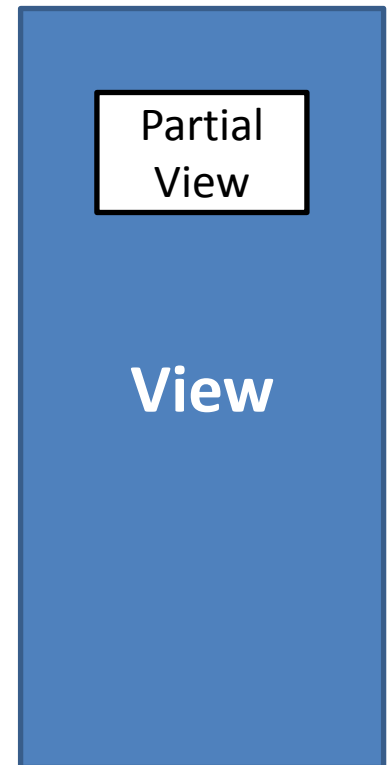
View Models

- What about model
 - Model contains data specific to business logic
- What about view model
 - View models contains data specific to views
 - View models are independent of models
- What a view model can do
 - Can return customized data
 - Can customize data without changing models
 - Can return data from multiple models.

Demo

Partial View

- Partial Views render a portion of a view content
- It simplifies complexity of a view
- Partial views are reusable in multiple views
- Partial views helps avoid duplicate code.



Demo

Layout Template

- `_Layout.cshtml` exists in shared folder
- Multiple views can use this template
- It keeps common stuffs
- `@RenderBody()` wraps view specific content
- `_ViewStart.cshtml` sets the layout to every view

Demo

Summary

- What is MVC
- Controllers
- Views
- Models.

Bibliography, Important Links

- <http://www.asp.net/mvc>

Any Questions?



Email : SushantBa@cybage.com
Extn. : 7221

Thank you!