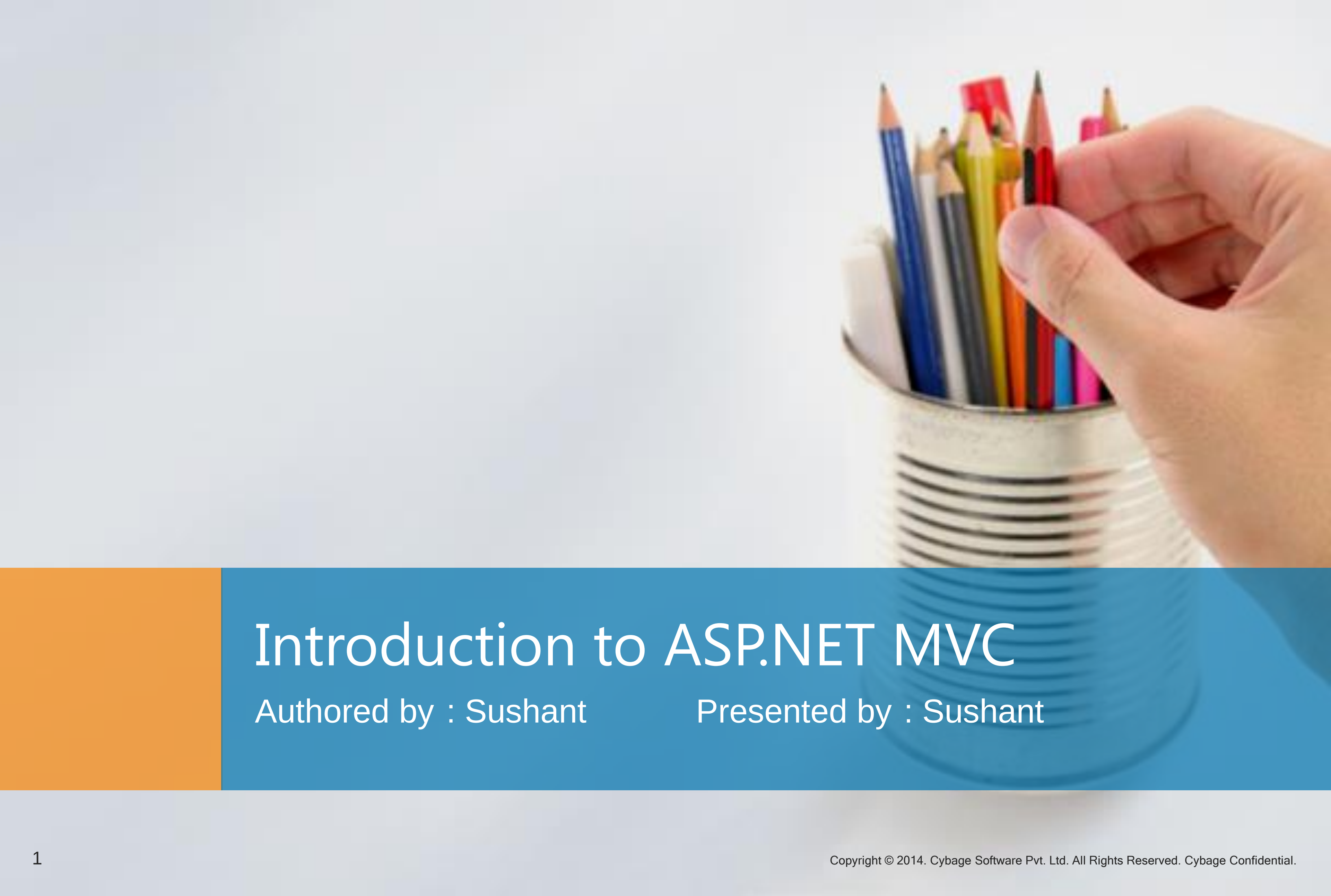


Introduction to ASP.NET MVC

Authored by : Sushant Presented by : Sushant

Agenda

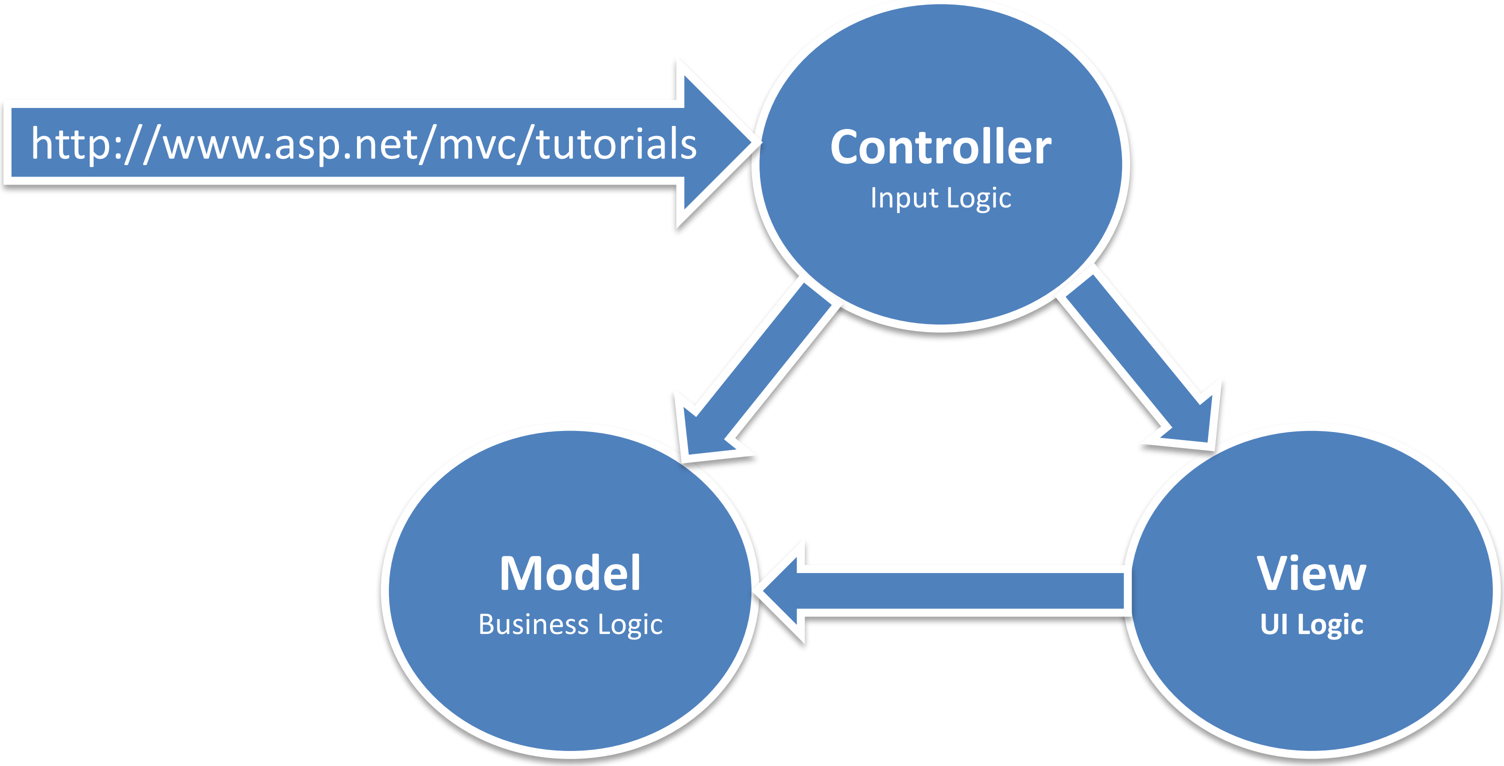
- History of Web
- MVC Pattern
- Controllers
- Routing
- Views
- Models
- Unit Testing

History of Web

- HTML
  - Static pages only
- HTML with Script
  - Dynamic user interactive pages
- Classic ASP
  - Supports strongly typed language such as C#
  - Difficult to maintain
- ASP.NET Web Forms
  - Server controls makes life easy
  - Uses Viewstate for managing state
  - Confusing page life cycle
  - Less control over application behavior

What is MVC

- Model-View-Controller is a design pattern



http://www.asp.net/mvc/tutorials

Controller
Input Logic

Model
Business Logic

View
UI Logic

ASP.NET MVC framework

- Developed by Microsoft
- Does not replace Web Forms
- Built on top of ASP.NET
- Defined in System.Web.Mvc

Why MVC

- Separation of concern
- Convention over configuration
- Test Driven Development (TDD)
- Highly extensible framework

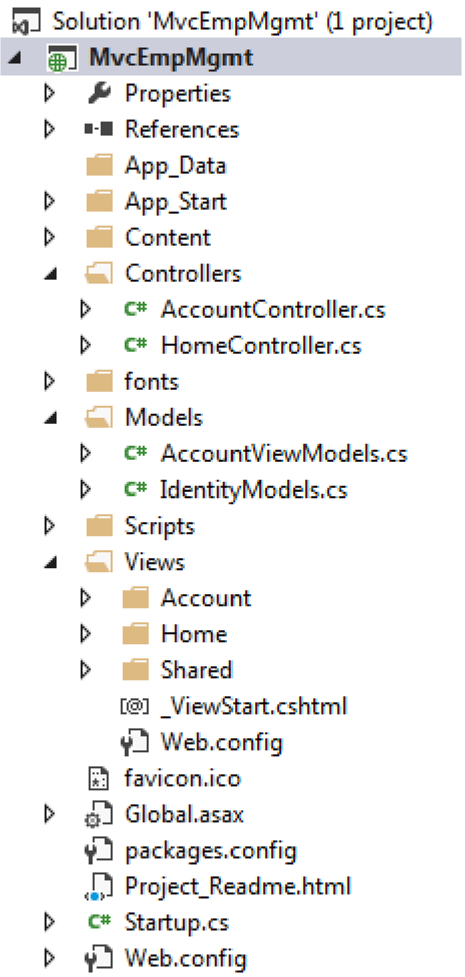
Web Forms Vs. MVC

- When to use Web Forms
  - Useful for small team of developers and designers
  - Rapid application development using inbuilt components
  - Unit testing is not a concern
  - Want to write less code

- When to use MVC
  - Want full control over the application behavior
  - Useful for large team of developers and designers
  - Unit testing is the major concern
  - Ready to write more code

File > New > Project

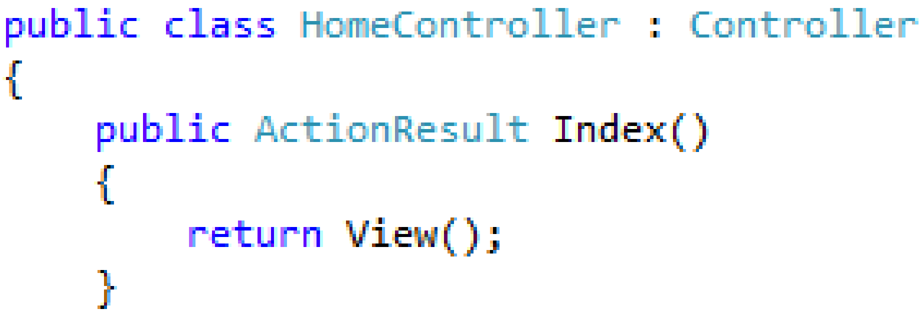
- Understanding MVC project structure
  - Controllers Folder
  - Models Folder
  - Views Folder



```
Solution 'MvcEmpMgmt' (1 project)
◢ MvcEmpMgmt
    ▷ Properties
    ▷ References
      App_Data
    ▷ App_Start
    ▷ Content
    ◢ Controllers
        ▷ AccountController.cs
        ▷ HomeController.cs
    ▷ fonts
    ◢ Models
        ▷ AccountViewModels.cs
        ▷ IdentityModels.cs
    ▷ Scripts
    ◢ Views
        ▷ Account
        ▷ Home
        ▷ Shared
          _ViewStart.cshtml
          Web.config
      favicon.ico
    ▷ Global.asax
      packages.config
      Project_Readme.html
    ▷ Startup.cs
    ▷ Web.config
```

Controllers

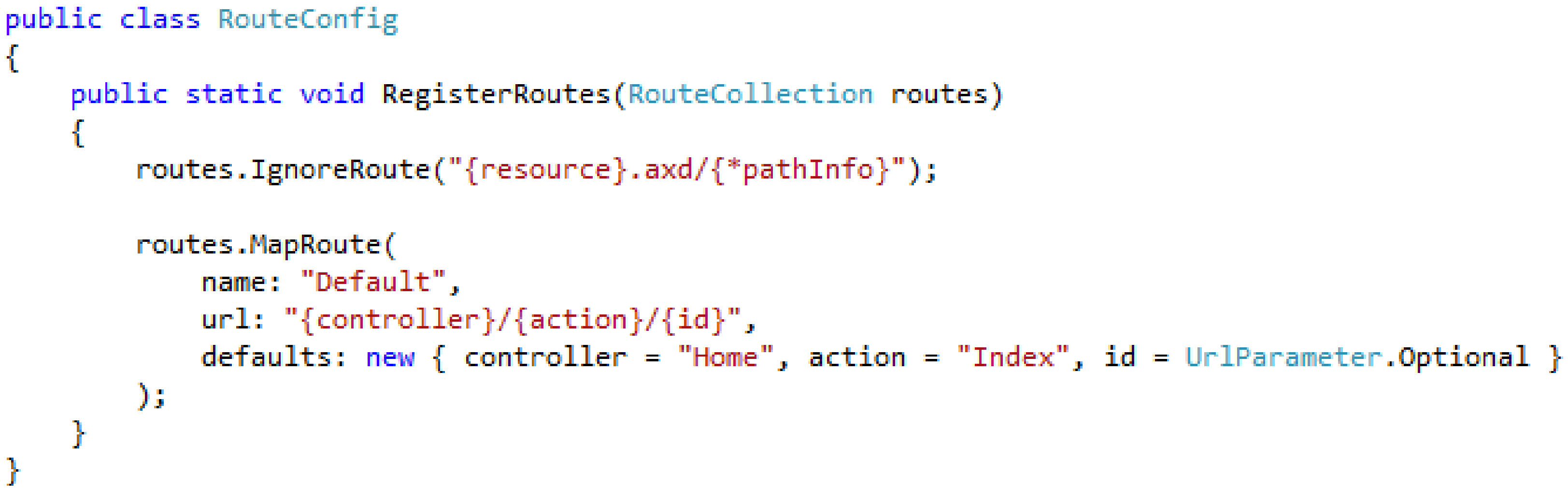
- A class responsible for
  - Accepting requests from client
  - Calling appropriate action methods
  - Interacting with models when required
  - Deciding which view to return



```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
```

Routing

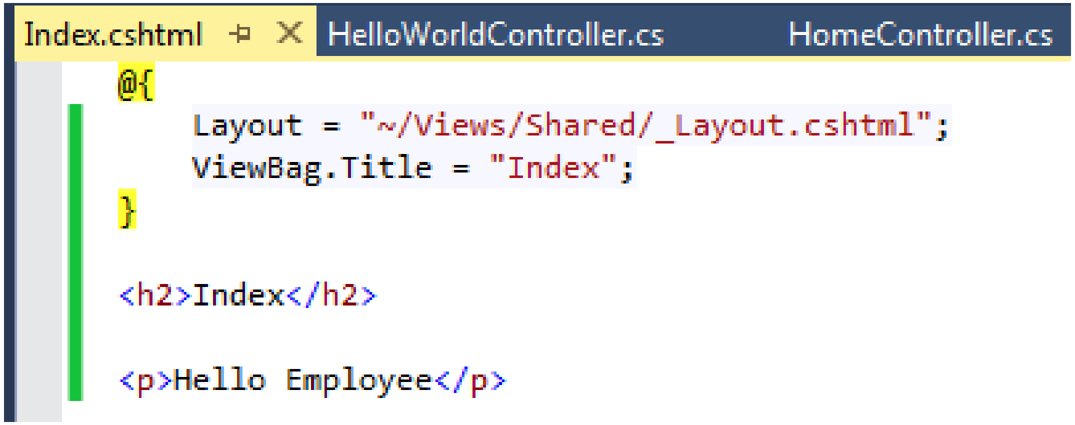
- App_Start/RouteConfig.cs file contains
  - A default URL routing logic
  - A route name
  - A URL pattern
  - Default values



```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

Views

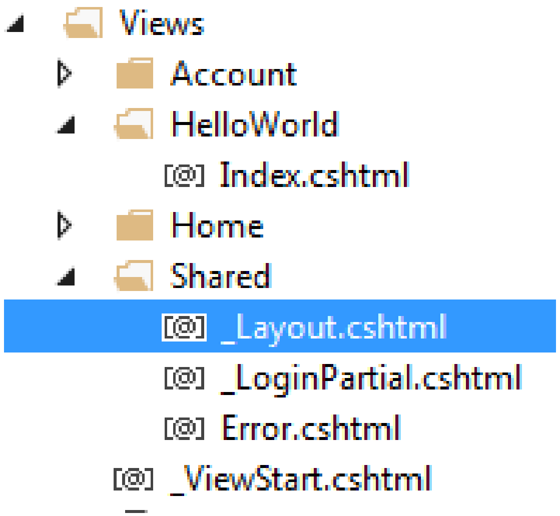
- Views are just template files
  - Provides UI to users
  - Generates HTML response to clients
  - View templates use Razor View Engine
  - Razor views has .cshtml file extension
  - Razor is a syntax to write a view template



```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>Hello Employee</p>
```

Layout Template

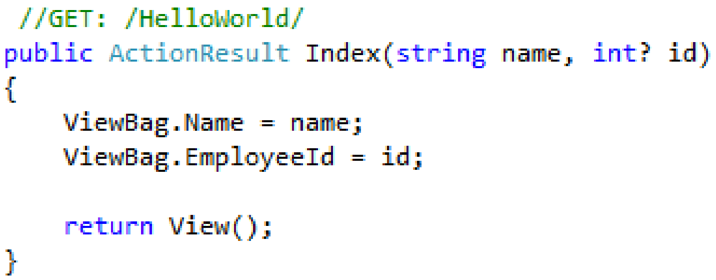
- A layout template
  - Named _Layout.cshtml exists in shared folder
  - All views can use this template
  - Keeps common stuffs
  - @RenderBody() wraps view specific content
  - _ViewStart.cshtml sets the layout to every view



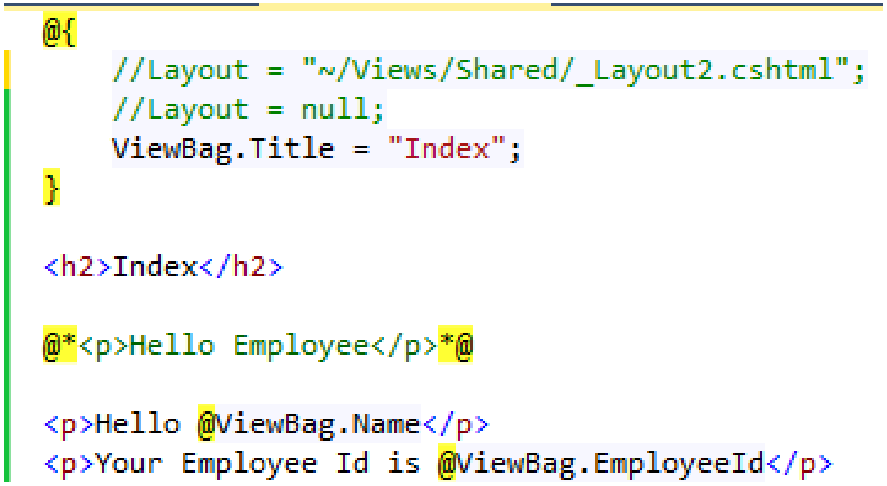
```
◢ Views
    ▷ Account
    ◢ HelloWorld
          [@] Index.cshtml
    ▷ Home
    ◢ Shared
          [@] _Layout.cshtml
          [@] _LoginPartial.cshtml
          [@] Error.cshtml
      [@] _ViewStart.cshtml
```

Passing Data

- Controllers pass data to view using ViewBag
- ViewBag is a dynamic object
- Allows adding any property
  - Set property values in controller
  - Access same properties in views



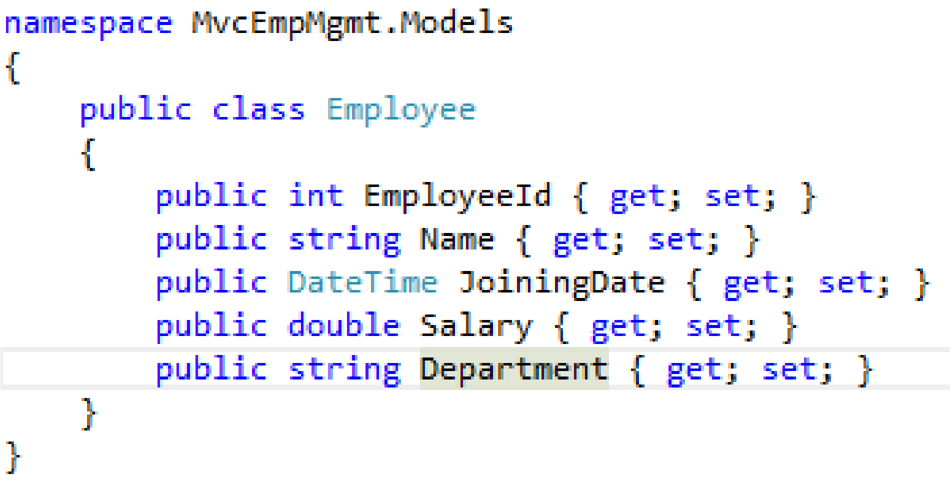
```
 //GET: /HelloWorld/
public ActionResult Index(string name, int? id)
{
    ViewBag.Name = name;
    ViewBag.EmployeeId = id;

    return View();
}
```



```
@{
    //Layout = "~/Views/Shared/_Layout2.cshtml";
    //Layout = null;
    ViewBag.Title = "Index";
}

<h2>Index</h2>

@*<p>Hello Employee</p>*@

<p>Hello @ViewBag.Name</p>
<p>Your Employee Id is @ViewBag.EmployeeId</p>
```

Models

- Models are classes represent entities
- MVC uses EF as data access technology
- EF uses models to map database objects



```
namespace MvcEmpMgmt.Models
{
    public class Employee
    {
        public int EmployeeId { get; set; }
        public string Name { get; set; }
        public DateTime JoiningDate { get; set; }
        public double Salary { get; set; }
        public string Department { get; set; }
    }
}
```

Unit Testing

- MVC supports TDD – Test Driven Development
- Arrange
  - Creating controller instance
- Act
  - Calling action methods
- Assert
  - Check for something such as return value etc.

Bibliography, Important Links

- http://www.asp.net/mvc
- http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview
- http://www.asp.net/mvc/tutorials/mvc-5/introduction/getting-started

Any Questions?

Thank you!