



# LẬP TRÌNH C# 4

## BÀI 5: VIEW

- ① Layout page Views
- ① ViewData
- ① ViewBag

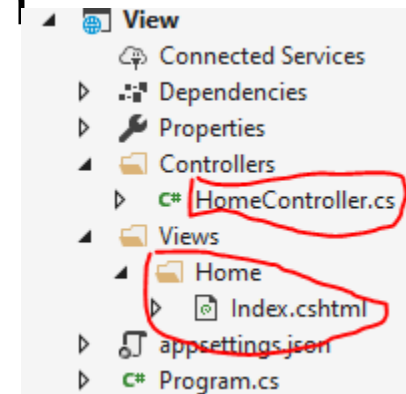


- ❑ View có trách nhiệm tạo ra giao diện cho người dùng từ model, controller
- ❑ Controller trong ASP.NET Core sẽ nhận request sau đó thực thi với logic tương ứng với dữ liệu đầu vào từ request. Sau đó nó trả về Model cho View
- ❑ Render ra giao diện và hiển thị model lên là trách nhiệm của View
- ❑ View có thể sử dụng bất cứ định dạng nào để trả về cho user. Định dạng có thể là HTML, JSON, XML hay là bất cứ định dạng nào khác

- ❑ Các View html được bố trí trong thư mục Views, nó là các template có phần mở rộng .cshtml
- ❑ Có thể nhúng mã C# trong view(Razor), những đoạn nhúng mã server side thì bắt đầu bằng ký hiệu @
- ❑ Khi một Action trả về đối tượng ActionResult thì nó sẽ sử dụng View .cshtml để tạo nội dung HTML. Nó sẽ tìm đến thư mục Views, tìm đến thư mục cùng tên Controller, ví dụ Home, sau đó là file .cshtml cùng tên với Action - file .cshtml tìm được sẽ sinh HTML

- ❑ Ví dụ có home controller và Index action

```
public class HomeController : Controller
{
    /...
    public IActionResult Index()
    {
        return View(); // Trả về ViewResult
    }
    /...
}
```



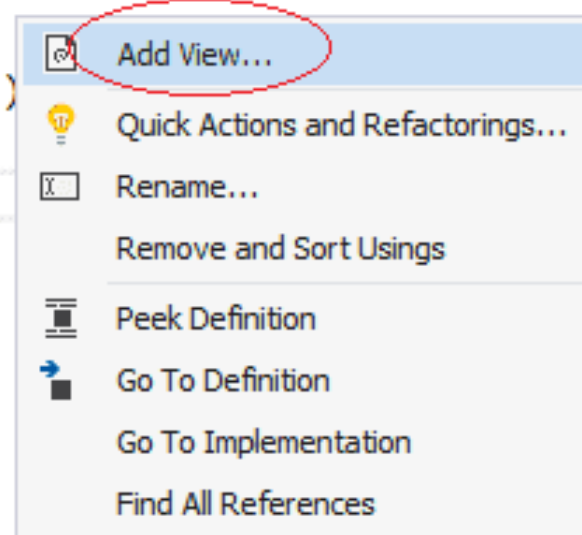
- ❑ Action trên sẽ sử dụng View tương ứng là Views/Home/Index.cshtml

```
<!doctype html>
<html>
  <head>
    <title>Trang View đầu tiên</title>
  </head>
  <body>
    <p>View .CSHTML là template <strong>Razor</strong> - nó chứa code C# bắt đầu bởi @@</p>
    <p><i>@DateTime.Now.ToLongDateString()</i></p>
  </body>
</html>
```

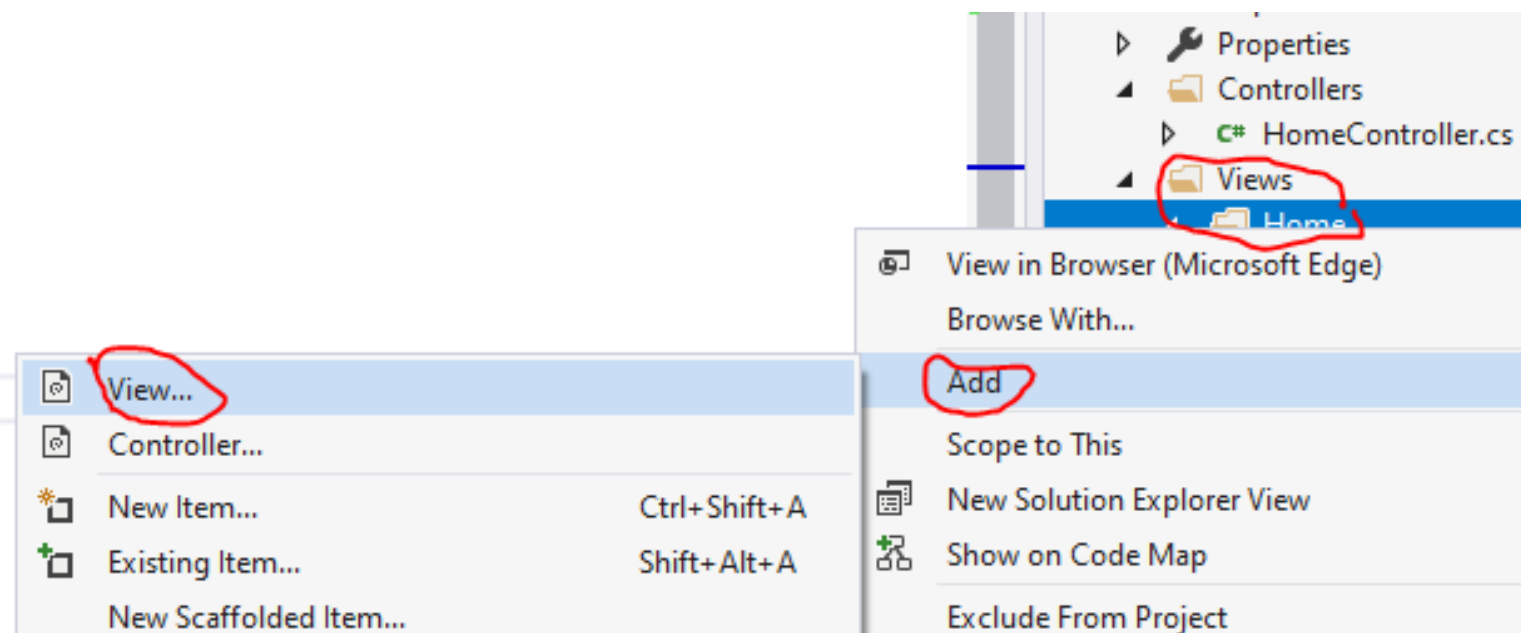
- ❑ Cách 1: Chuột phải vào bất cứ đâu trong phương thức và chọn Add View

```
public IActionResult Index()  
{  
    HomeModel message = new HomeModel();  
    return View(message);  
}
```

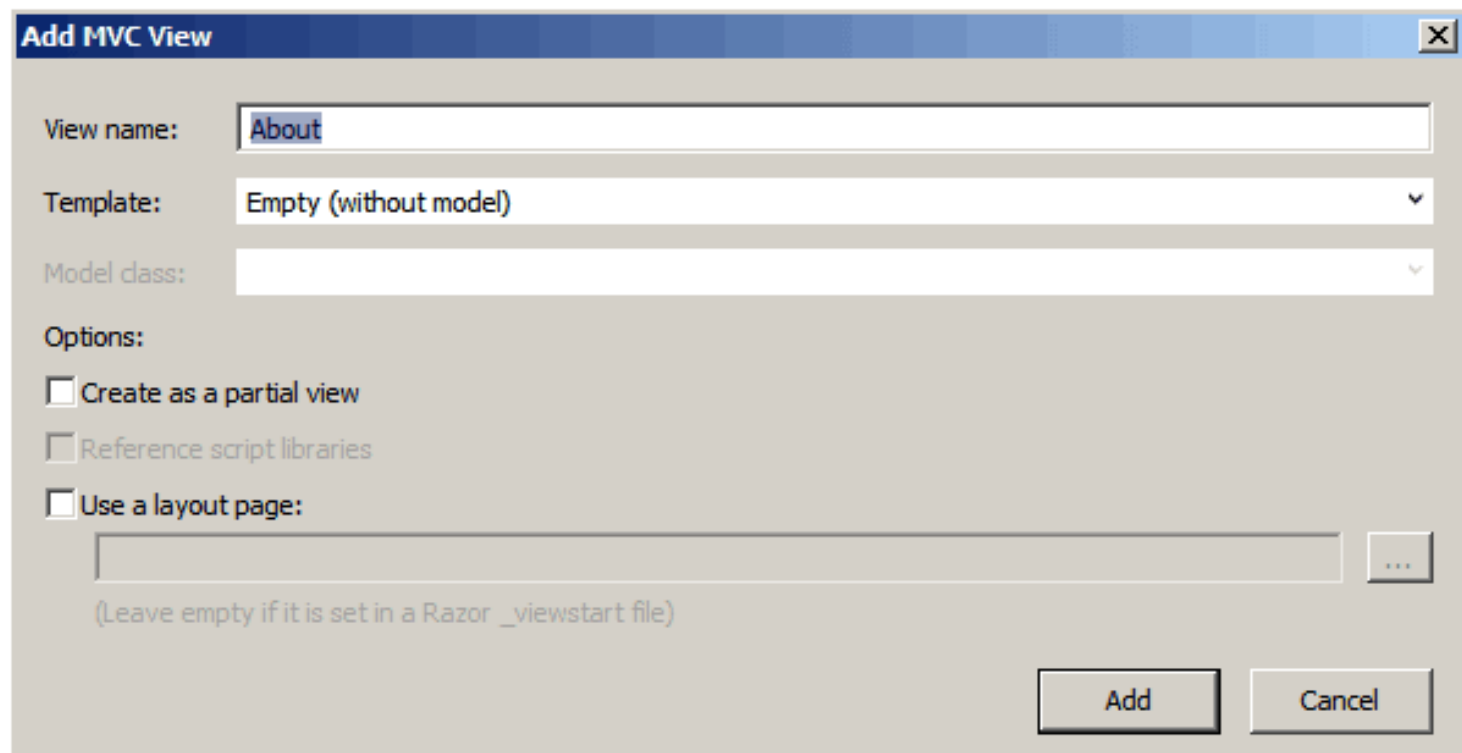
```
public IActionResult About()  
{  
    return View();  
}
```



## ❑ Cách 2: Tạo thư mục Views, bên trong tạo file View tương ứng



## □ Điền các thông tin



**Add MVC View**

View name:

Template:

Model class:

Options:

☐ Create as a partial view

☐ Reference script libraries

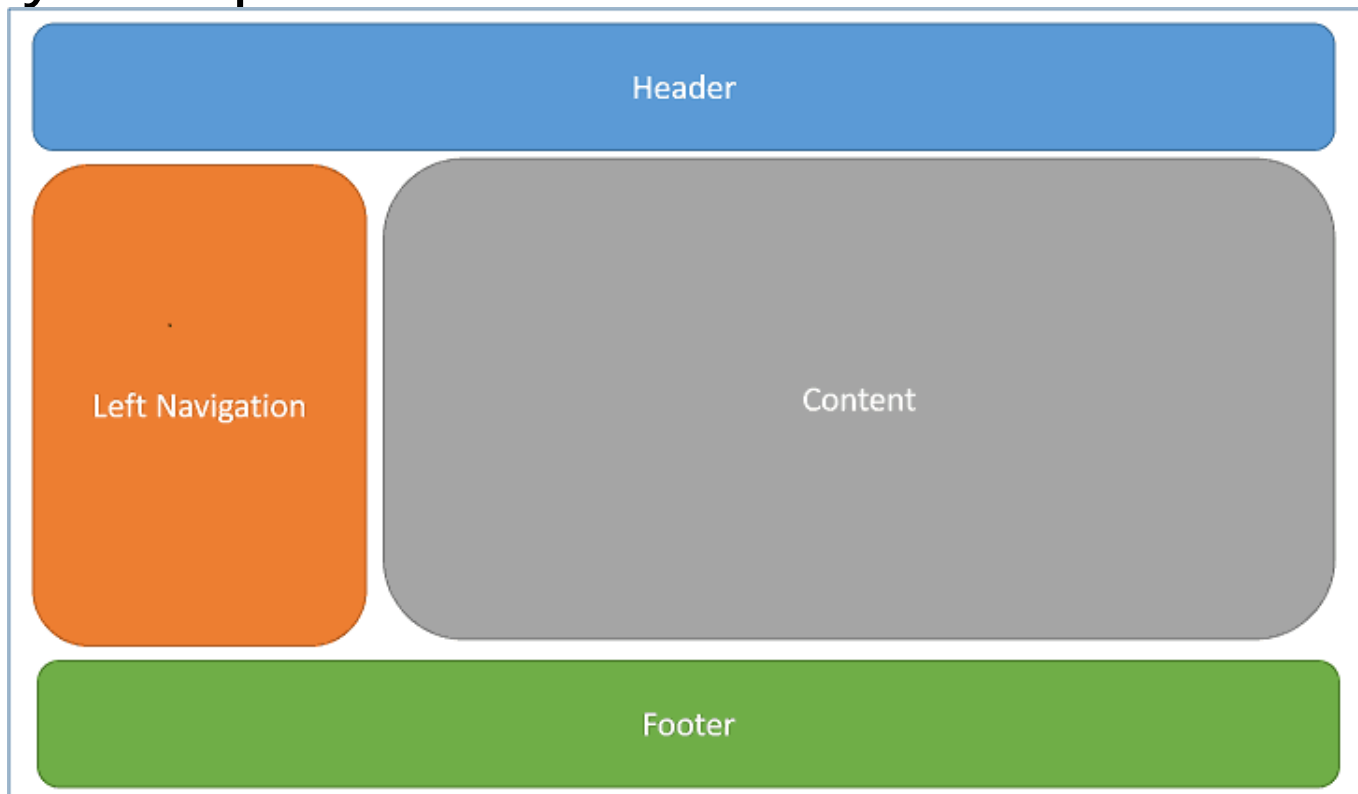
☒ Use a layout page:

(Leave empty if it is set in a Razor \_viewstart file)

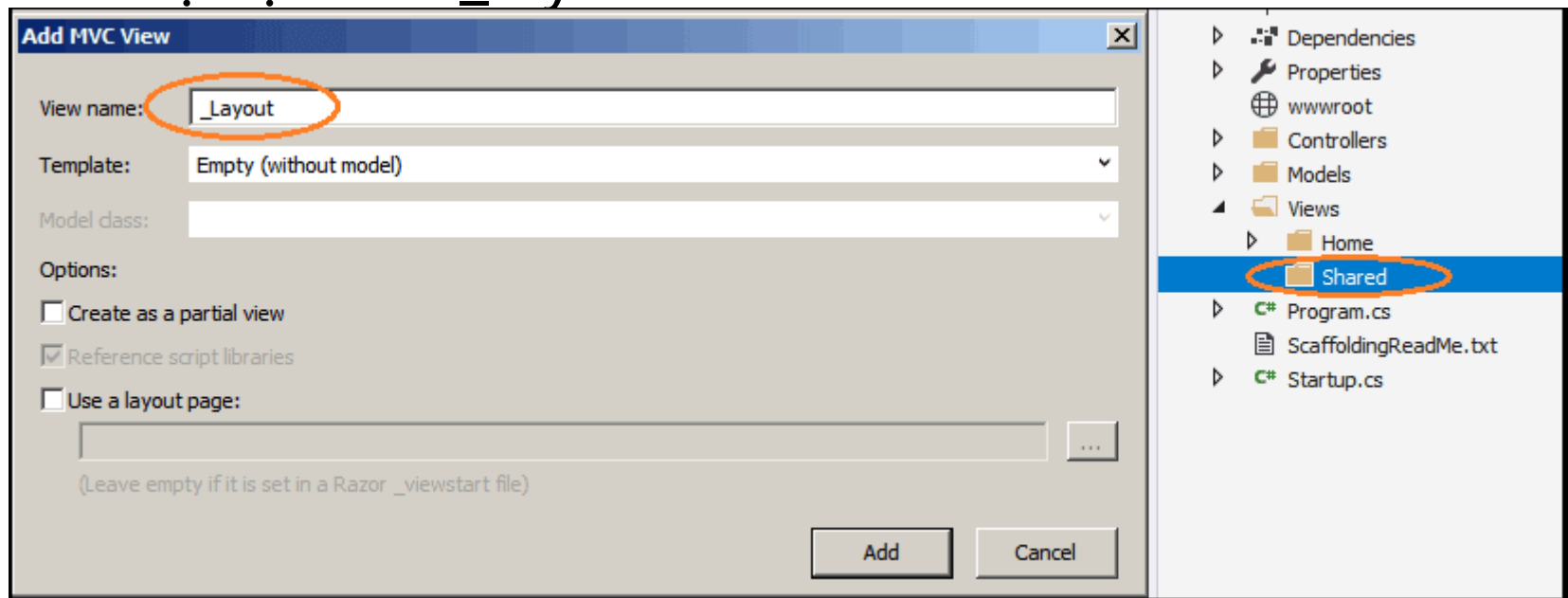


- ❑ View Name: Nhập tên View tại đây. Theo quy tắc, tên phải cùng tên với Action Method name
- ❑ Template: Template có vài tùy chọn như Create, Delete, Details, Edit, List, Empty (without Model). Các template ở trên cần một model, ngoại trừ Empty Template.
- ❑ Model class: Dropdown hiển thị tất cả các Model class trong project. Tùy chọn này được loại bỏ nếu bạn chọn Empty project.
- ❑ Partial View: Partial View tạo một phần của view và không phải view hoàn chỉnh
- ❑ Layout Page: Layout page được dùng để chia sẻ các thành phần dùng chung trong trang của bạn và cung cấp một giao diện đồng nhất trong toàn bộ hệ thống.

- ❑ Giúp đảm bảo sự thống nhất giữa các trang trong toàn bộ các view của ứng dụng.
- ❑ Các phần menu, header, footer thường sẽ không thay đổi qua các view

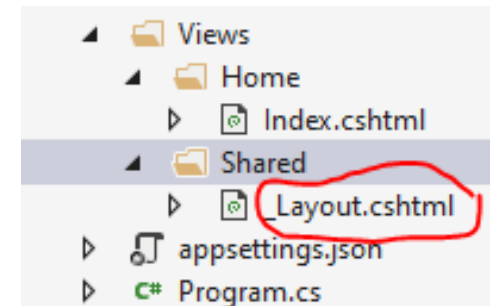


- ❑ \_Layout.cshtml định nghĩa một giao diện có các phần tử dùng chung như header, footer, navigation menu trên trang ở một vị trí mà có thể dùng cho mọi nơi.
- ❑ Layouts pages đặt trong thư mục Views/Shared, ví dụ tạo file \_layout.chtml



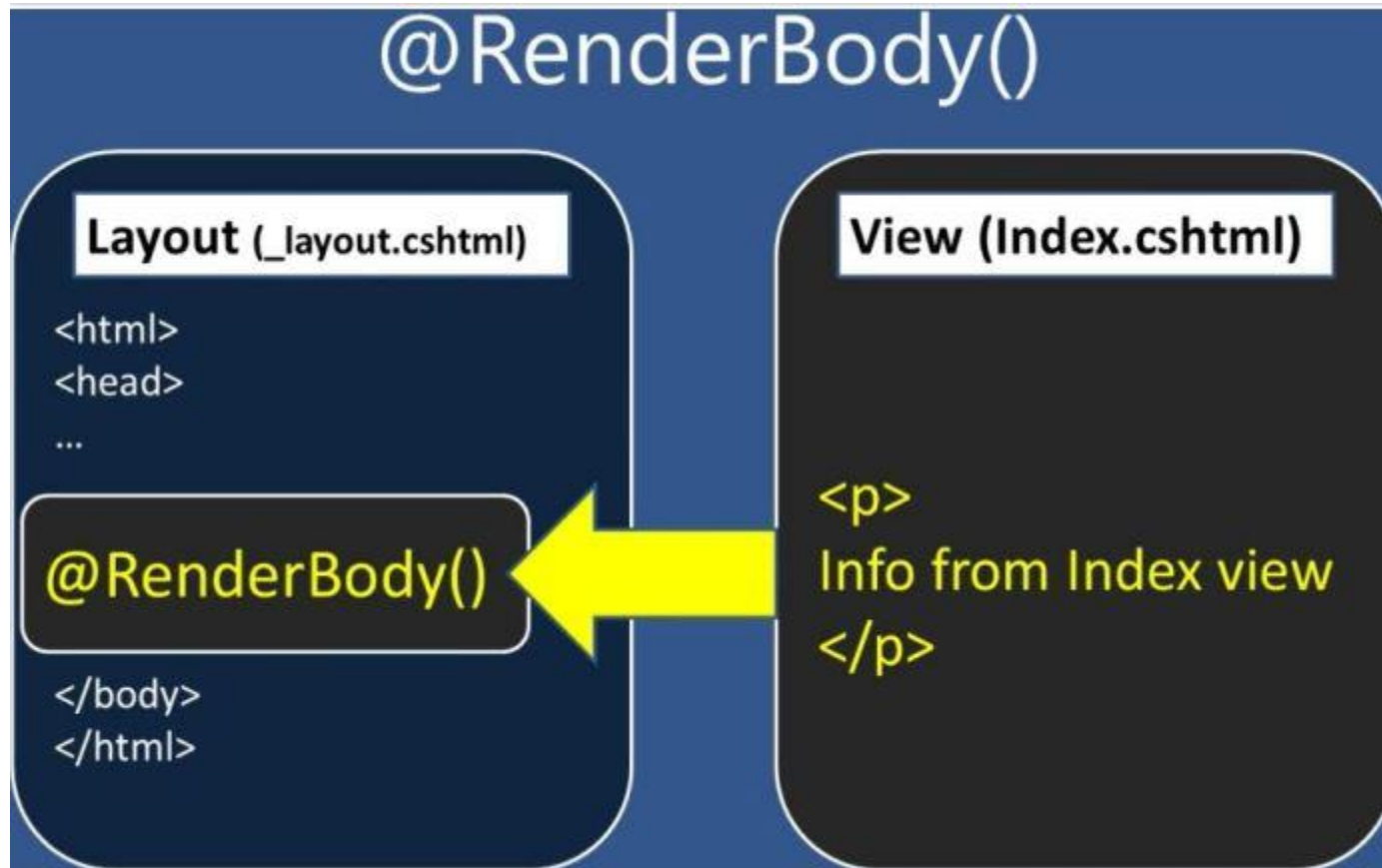
- \_layout.cshtml có 3 phần là header, content và footer. Content là phần nội dung thay đổi khác nhau ở các view khác nhau (@RenderBody())

```
@{  
    Layout = null;  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <meta name="viewport" content="width=device-width" />  
    <title>Layout Example</title>  
</head>  
<body>  
    <div id="header">  
        <h1>This is Layout example in HTML</h1>  
    </div>  
    <div id="content">  
        <p> this is content from layout pages</p>  
        @RenderBody()  
    </div>  
    <div id="Footer">  
        <p>This is Footer</p>  
    </div>  
</body>  
</html>
```

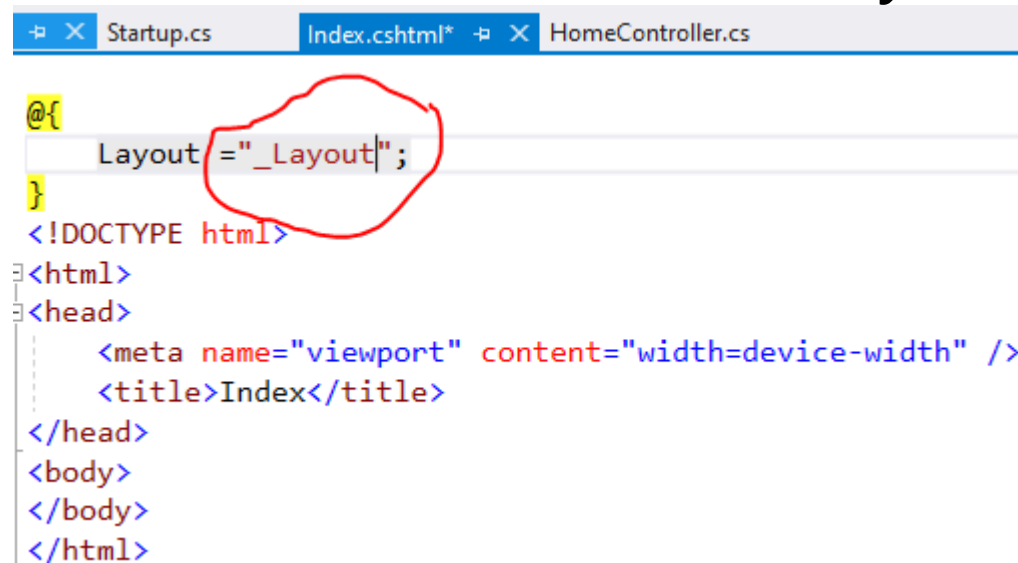


- ❑ Các view kế thừa `_layout.chnml` sẽ thừa hưởng các nội dung chứa trong `_layout.chnml`
- ❑ `RenderBody` là một phương thức đặc biệt đánh dấu vị trí nơi mà các view sử dụng Layout này sẽ được đặt vào đó khi chạy
- ❑ Cú pháp chỉ định view kế thừa layout pages

```
@{  
    Layout = "Layout page Name";  
}
```

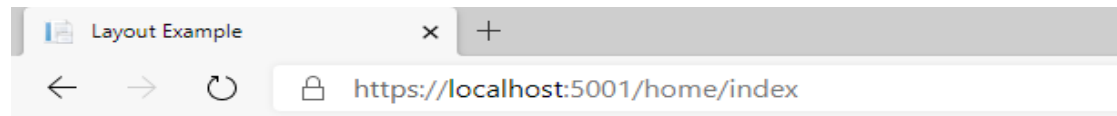


- ❑ Ví dụ view Index.html kế thừa \_Layout.html



```
@{  
    Layout = "_Layout";  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <meta name="viewport" content="width=device-width" />  
    <title>Index</title>  
</head>  
<body>  
</body>  
</html>
```

- ❑ Run ứng dụng với view Index được kết quả kế thừa các nội dung từ \_Layout



**This is content of Layout example in HTML**

this is content from layout pages

This is Footer of layout pages

- Ví dụ view Index.html kế thừa \_Layout.html và thêm nội dung riêng cho view Index

```
@{
    Layout = "_Layout";
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <h3>This is content private of view Index </h3>
</body>
</html>
```

- Run ứng dụng với view Index

**This is content of Layout example in HTML**

this is content from layout pages

**This is content private of view Index**

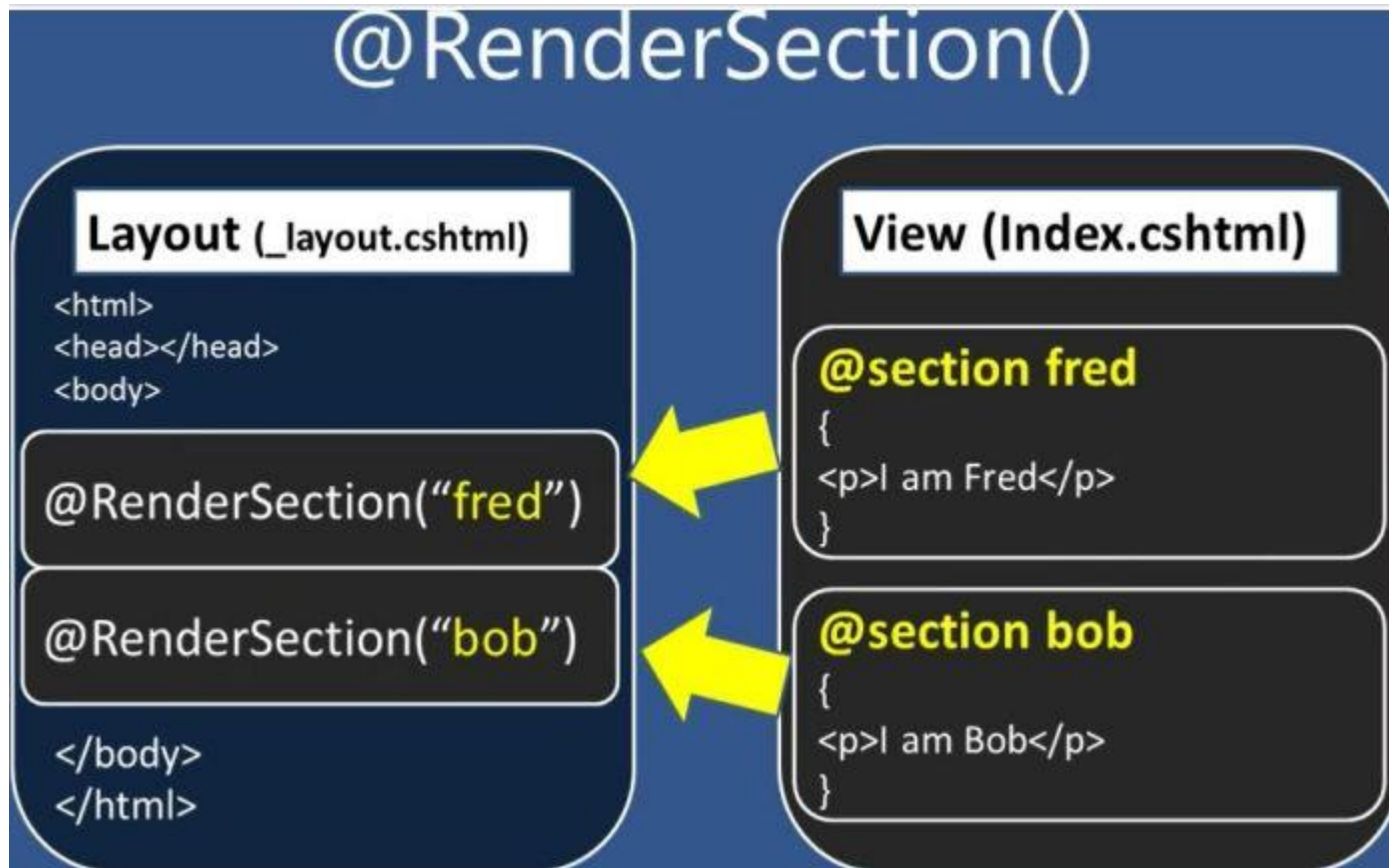
This is Footer of layout pages

*Inserted from  
the Layout  
Page*

*from view Index*



- ❑ RenderSection() phương thức tùy chọn cho phép view con hiển thị nội dung một thành phần trong layout
- ❑ ASP.NET Core cho phép tạo một hoặc nhiều Rendersection cho layout
- ❑ Cú pháp định nghĩa section trong layout pages:  
RenderSection(string name, bool required)
  - ❖ Name: tên section
  - ❖ Required: quy định có bắt buộc phải sử dụng section này trong các view kế thừa layout pages hay không



- ❑ Ví dụ định nghĩa một section tên "Fpoly" trong \_Layout.cshtml với tham số required là false tức là không yêu cầu bắt buộc phải có section này trong view con.

```

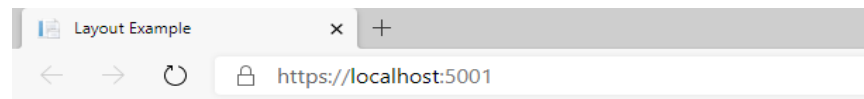
X _Layout.cshtml* X
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Layout Example</title>
</head>
<body>
    <div id="header">
        <h1>This is content of Layout example in HTML</h1>
    </div>
    <div>
        @RenderSection("Fpoly", required: false)
    </div>
    <div id="content">
        <p> this is content from layout pages</p>
        @RenderBody()
    </div>

```

- ❑ Ví dụ sử dụng section Fpoly và tạo nội dung cho nó trong Index view

```
Index.cshtml* - X
@{
    Layout = "_Layout";
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <h3>This is content private of view Index </h3>
    @section Fpoly
    {
        <p>Section is declared at layout pages but content of
    }
</body>
```

- ❑ Run ứng dụng:



**This is content of Layout example in HTML**

(Section is declared at layout pages but content of Index view)

this is content from layout pages

**This is content private of view Index**

This is Footer of layout pages



DEMO

- Hiện thực các ví dụ





# LẬP TRÌNH C# 4

## BÀI 5: VIEW (P2)

- ❑ ViewData là một thuộc tính của Controller base class, trả về một đối tượng ViewDataDictionary
- ❑ ViewDataDictionary là một đối tượng dictionary cho phép lưu dữ liệu dạng key-value
- ❑ Key phải là một chuỗi không phân biệt chữ hoa thường.
- ❑ ViewData truyền dữ liệu sang View từ Controller
- ❑ Tại View có thể truy cập giá trị được lưu trong ViewData thông qua key

## ❑ Ví dụ dùng viewdata kiểu chuỗi

```
ViewData["Title"] = " Student Details" ;
```

**Controller**

```
@ ViewData["Title"]
```

**View**

As we are accessing string data so there is no need to cast it to string type

## ❑ Ví dụ dùng viewdata kiểu đối tượng

```
Student student = new Student()  
{  
    StudentId = 101,  
    Name = "James",  
    Branch = "CSE",  
    Section = "A",  
    Gender = "Male"  
};
```

```
ViewData["Student"] = student;
```

**Controller**

```
@{  
    var student = ViewData["Student"]  
        as FirstCoreMVCApplication.Models.Student;  
}
```

**View**

As the data stored in the ViewData dictionary is of Student Type so we need to explicitly cast it to Student Type



- ❑ Ví dụ dùng viewdata truyền dữ liệu product từ controller sang view
- ❑ Trong View, chúng ta lấy sản phẩm từ ViewData và chuyển sang kiểu ProductModel

```
@{  
    // Since Product isn't a string, it requires a cast.  
    var product = ViewData["Product"] as ProductModel;  
}  
  
@ViewData["Greeting"]!  
  
@product.ProductID<br>  
@product.Name<br>  
@product.Brand<br>  
@product.Price<br>
```

```
public IActionResult Index()  
{  
    ViewData["Greeting"] = "Hello World";  
    ViewData["Product"] = new ProductModel()  
    {  
        ProductID=1,  
        Name = "Samsung galaxy Note",  
        Brand = "Samsung",  
        Price = 19000  
    };  
  
    return View();  
}
```

- ❑ ViewBag trong ASP.NET Core MVC là cơ chế truyền dữ liệu từ controller sang view
- ❑ Không cần chuyển kiểu dữ liệu khi sử dụng
- ❑ Ví dụ ViewBag với chuỗi

**Storing the data in ViewBag**

```
ViewBag.Header = "Student Details";
```

**Controller**

**Accessing the Data**

```
@ViewBag.Header
```

**View**

**Type Casting is not Required**

## ❑ Ví dụ ViewBag với đối tượng

```
Student student = new Student()
{
    StudentId = 101,
    Name = "James",
    Branch = "CSE",
    Section = "A",
    Gender = "Male"
};

ViewBag.Student = student;
```

**Controller**

```
@{
    var student = ViewBag.Student;
}
```

**Type Casting is not Required even though the data we are accessing is of Complex Type**

**View**

## ❑ Ví dụ sử dụng ViewBag tạo website:

### Employee Fpoly Education Data

Serial No	Name	Address	Phone
1	FplyHN	Tòa nhà FPT Polytechnic, Phố Trịnh Văn Bô, Nam Từ Liêm, Hà Nội	098 172 58 36
2	FpolyHcm	391A Nam Kỳ Khởi Nghĩa, Phường 8, Quận 3, Hồ Chí Minh	028 3526 8799
3	FpolyDN	137 Nguyễn Thị Thập, Thanh Khê Tây, Liên Chiểu, Đà Nẵng	098 172 58 36
4	FpolyTN	số 27, Tòa nhà VIB, đường, Nguyễn Tất Thành, TP. Buôn Mê Thuật, Đắk Lắk	0262 3555 678
5	FpolyCT	288 Đường Nguyễn Văn Linh, Hưng Lợi, Ninh Kiều, Cần Thơ	0292 7300 468

## ❑ B1: tạo model:

```
public class Employee
{
    [Display(Name = "Serial No")]
    5 references | 0 exceptions
    public byte EmployeeId { get; set; }

    [Display(Name = "Name")]
    5 references | 0 exceptions
    public string EmployeeName { get; set; }
    5 references | 0 exceptions
    public string Address { get; set; }
    5 references | 0 exceptions
    public string Phone { get; set; }
}
```

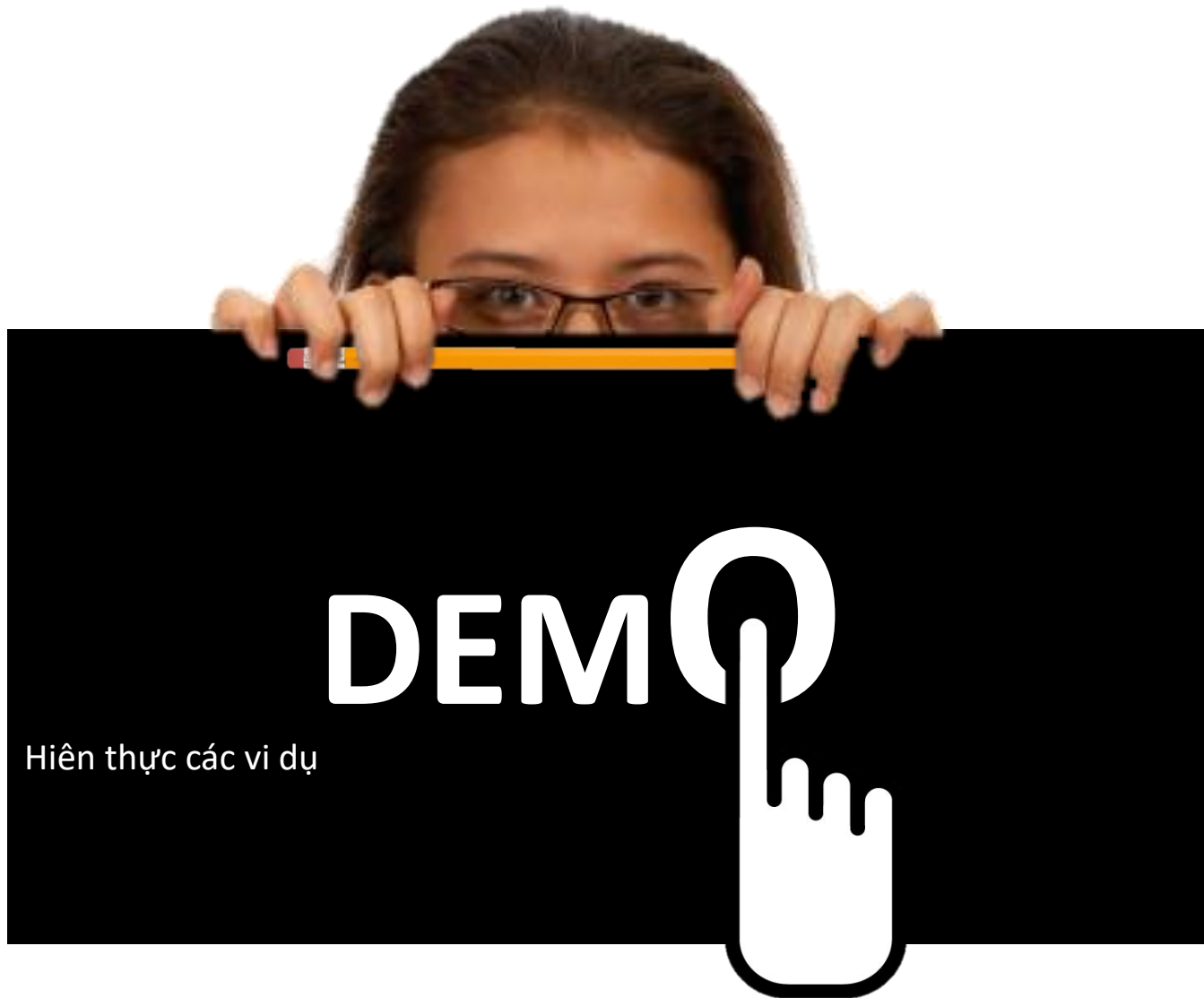
## ❑ B2: tạo controller:

```
public IActionResult GetEmployeeData()
{
    List<Models.Employee> emp = new List<Employee>
    {
        new Employee
        {
            EmployeeId = 2,
            EmployeeName = "FpolyHcm",
            Address = "391A Nam Kỳ Khởi Nghĩa, Phường 8, Quận 3, Hồ Chí Minh",
            Phone = "028 3526 8799"
        },
        new Employee
        {
            EmployeeId = 3,
            EmployeeName = "FpolyDN",
            Address = "137 Nguyễn Thị Thập, Thanh Khê Tây, Liên Chiểu, Đà Nẵng",
            Phone = "098 172 58 36"
        },
    };

    ViewBag.employee = emp;
    return View();
}
```

## ❑ B2: tạo view :

```
@model IEnumerable<ViewBag.Models.Employee>
<body>
<h3 style="color:rgb(11,77,163);">Employee Fpoly Education Data</h3>
<table>
<thead>
<tr>
<th>Serial No</th>
<th>Name</th>
<th>Address</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>@ViewBag.employee[0].EmployeeId</td>
<td>@ViewBag.employee[0].EmployeeName</td>
<td>@ViewBag.employee[0].Address</td>
<td>@ViewBag.employee[0].Phone</td>
</tr>
<tr>
<td>@ViewBag.employee[1].EmployeeId</td>
<td>@ViewBag.employee[1].EmployeeName</td>
<td>@ViewBag.employee[1].Address</td>
<td>@ViewBag.employee[1].Phone</td>
</tr>
```



# Tổng kết bài học

- ◎Layout page Views
- ◎ViewData
- ◎ViewBag







**KẾT THÚC**