Bài 3

Tìm hiểu về Views

Mục tiêu

- Giới thiệu về Views
- Cách tổ chức Views
- Tạo Views
- Ngôn ngữ Razor
- Razor Tag Helpers
- Tệp Layout và ViewStart
- Partial View
- View Compoment

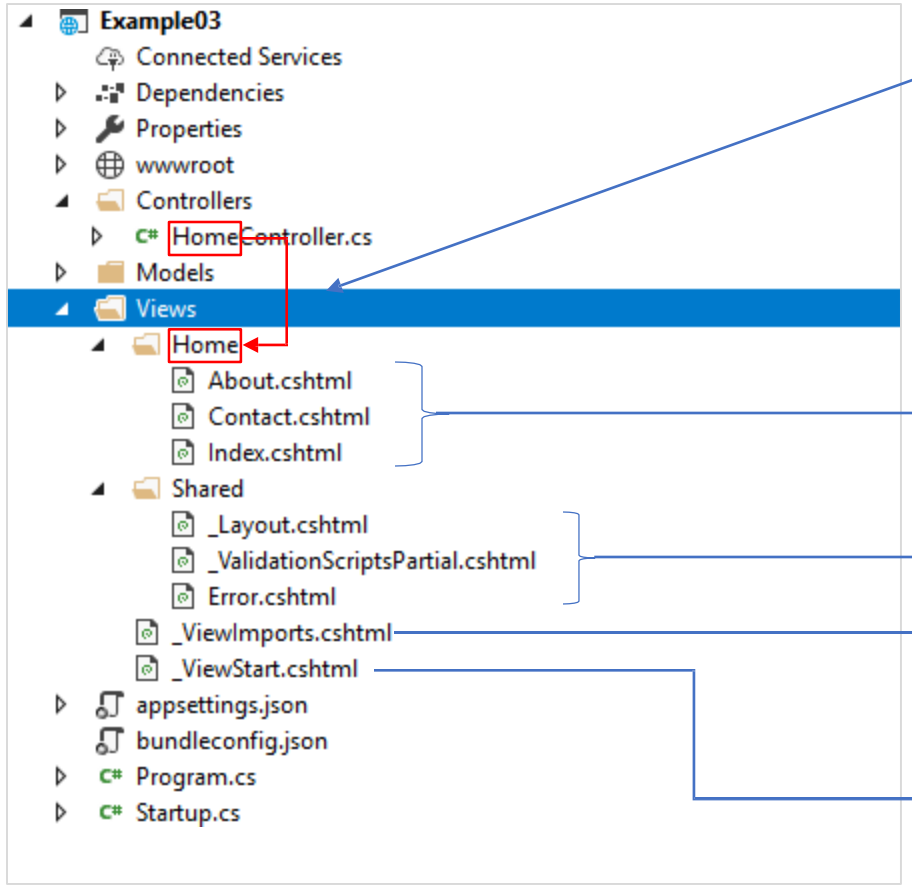
Giới thiệu về Views

MVC Pattern → View là thành phần hiểu thị dữ liệu tới người dùng

ASP.NET Core MVC

- View là 1 tệp tin với đuôi **.cshtml** chứa mã **HTML** và code **Razor C#**
- View trình bày dữ liệu tách biệt với phần logic của ứng dụng
- Hầu hết ứng dụng MVC sử dụng ngôn ngữ **Razor** cho View
- **Razor View Engine** làm nhiệm vụ convert code **Razor C#** sang dạng **HTML** để **response** về **Client**

Tổ chức các view trong ứng dụng ASP.NET Core



Các view được tổ chức lưu trữ trong thư mục **Views** của cấu trúc project

Mỗi **controller** tạo ra thì 1 thư mục trùng tên với **controller** sẽ được tạo ra trong thư mục **Views**

Các view ứng với các action trong controller

Các view dùng chung

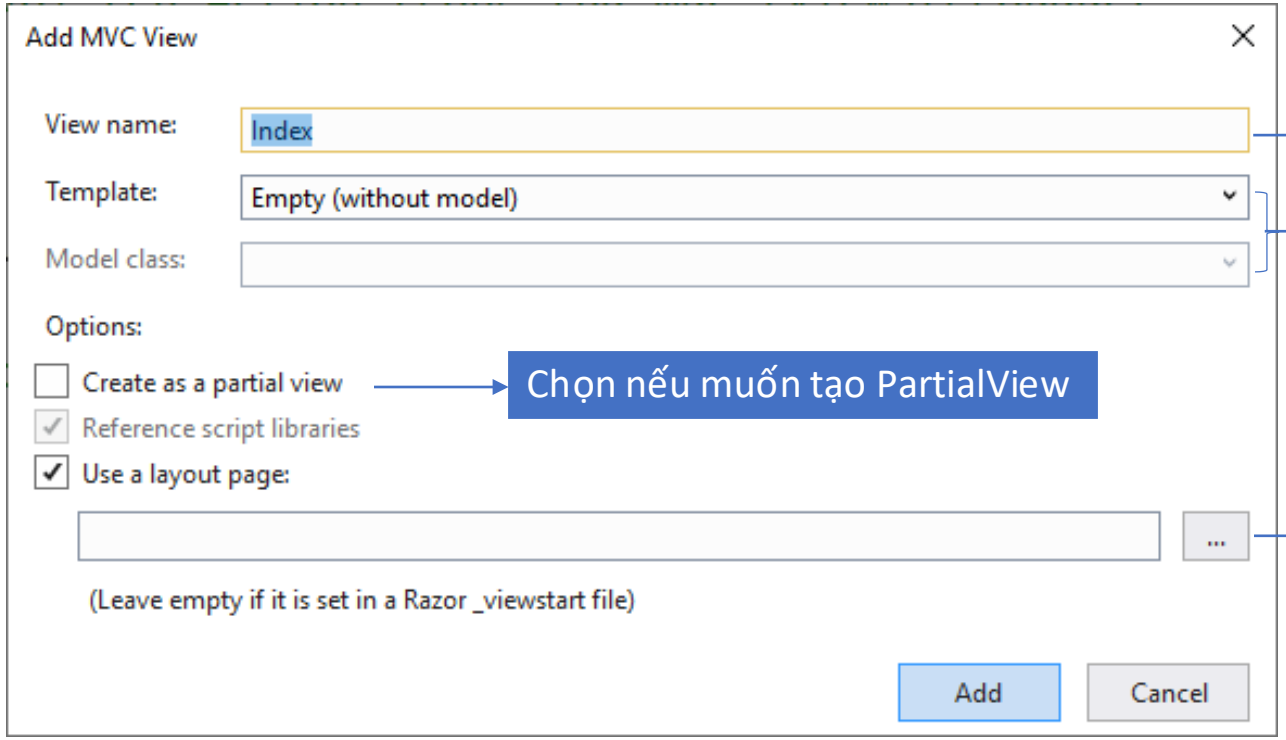
Import các namespace được sử dụng cho tất cả các view khác

Chỉ ra layout dùng chung cho các View, tệp này được thực thi tại thời điểm bắt đầu mỗi khi một view được rendering ra HTML.

Tạo view

Cách 1: Kích chuột phải vào **Action** trong **Controller** chọn **AddView**…

Cách 2: Kích chuột phải vào thư mục trong **Views** chọn **Add->View**…



Nhập tên View (thông thường trùng tên với tên Action)

Để trống nếu không dùng Template (xét sau)

Chọn nếu muốn tạo PartialView

Bỏ trống nếu muốn sử dụng Layout mặc định trong tệp _ViewStart hoặc kích vào …. để chọn layout khác cho view

Ngôn ngữ Razor

- Razor là ngôn ngữ phía server-side dùng để lập trình trên View trong ứng dụng ASP.NET Core MVC. Cú pháp của Razor dựa trên ngôn ngữ C# hoặc VB.NET.

Compact: Razor là nhỏ gọn cho phép bạn giảm thiểu số lượng ký tự và tổ hợp phím cần thiết để viết mã.

Intellisense: Razor hỗ trợ gợi ý gác câu lệnh.

Easy to Learn: Razor dễ học bạn có thể dụng các ngôn ngữ lập trình C#, Visual Basic.

Unit Testable: Razor hỗ trợ khả năng unit test cho các view mà không cần các controller hoặc web-server.

Cú pháp Razor C#

- Sử dụng biểu tượng @ để phân tách code HTML và code Razor

```
<!--Dạng inline -->
@expression
```

```
<!-- Dạng block -->
@{
    Statement;
    Statement;
}
```

```
<h2>Cú pháp Razor</h2>

<!-- Single statement blocks  -->
@{ var total = 7000; }
@{ var myMessage = "Biết người là trí, biết bản thân mới là trí tuệ thật sự. " +
                   "Thắng người là sự mạnh mẽ, thắng bản thân mới là sức mạnh thực sự"; }

<!-- Inline expressions -->
<p>Tài khoản của bạn đang có: $ @total  </p>
<p>Câu nói để đời của Lão Tử: @myMessage</p>

<!-- Multi-statement block -->
@{
    var greeting = "Chúc bạn thành công!";
    var weekDay = DateTime.Now.DayOfWeek;
    var greetingMessage = greeting + " Hôm nay là: " + weekDay;
}
<p>Lời chúc: @greetingMessage</p>
```

Cú pháp Razor

Tài khoản của bạn đang có: $ 7000

Câu nói để đời của Lão Tử: Biết người là trí, biết bản thân mới là trí tuệ thật sự. Thắng người là sự mạnh mẽ, thắng bản thân mới là sức mạnh thực sự

Lời chúc: Chúc bạn thành công! Hôm nay là: Wednesday

Cấu trúc if, if...else

```
<h2>Cấu trúc IF...ELSE</h2>

@if (total >= 5000)
{
    <p>Bạn nằm trong số 30% những người giàu nhất thế giới</p>
}
else if (total >= 2000)
{
    <p>Bạn nằm trong số 50% những người giàu nhất thế giới</p>
}
else
{
    <p>Chưa có số liệu thống kê</p>
}

@{
    if (total % 2 == 0)
    {
        <p>Số tiền trong tài khoản của bạn là số chẵn</p>
    }
    else
    {
        <p>Số tiền trong tài khoản của bạn là số lẻ</p>
    }
}
```

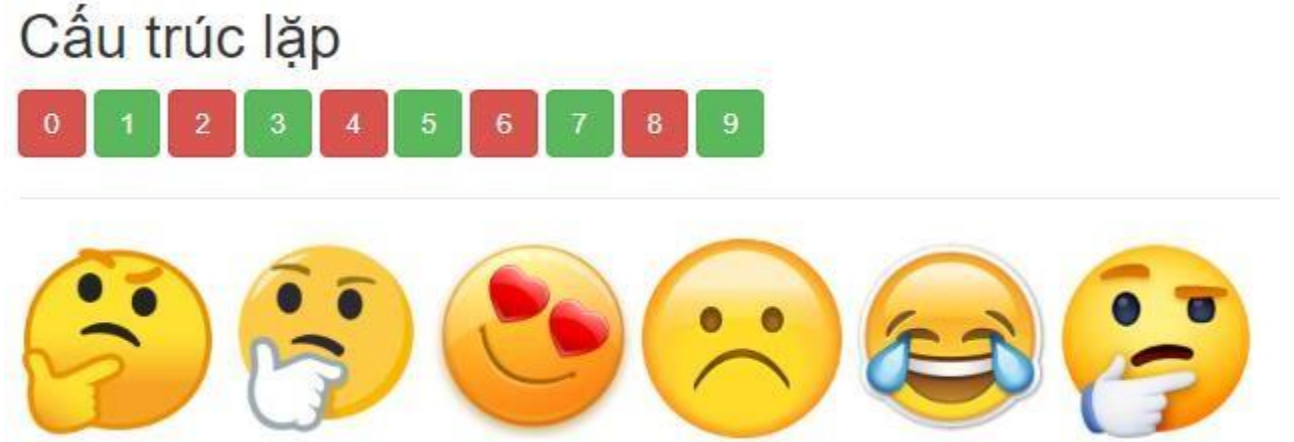
Cấu trúc IF...ELSE

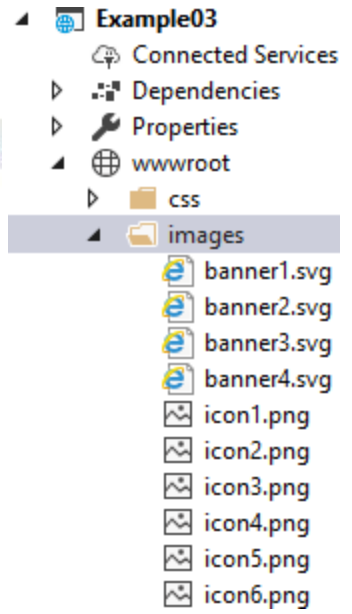
Bạn nằm trong số 30% những người giàu nhất thế giới

Số tiền trong tài khoản của bạn là số chẵn

Cấu trúc lặp (for, while, foreach, do…while)

```
<h2>Cấu trúc lặp</h2>
@{
    for (int i = 0; i < 10; i++)
    {
        if (i % 2 == 0)
        {
            <span class="btn btn-danger">@i</span>
        }
        else
        {
            <span class="btn btn-success">@i</span>
        }
    }
    string[] images = { "icon1.png", "icon2.png", "icon3.png", "icon4.png", "icon5.png", "icon6.png" };
    <hr />
    @foreach (var c in images)
    {
        <img src="/images/@c" width="100"/>
    }
}
```





HTML Helpers

- **HtmlHelpers** là các phương thức mở rộng viết bằng C# và sử dụng trong **View** để sinh ra nội dung **HTML**.
- Các phiên bản **ASP.NET MVC** cung cấp một loạt các **Build-in HtmlHelpers** và đó là những **HtmlHelper** truyền thống sử dụng **phức tạp và khó**.
- Phiên bản **ASP.NET Core MVC** giới thiệu một bộ thẻ mới có tên **Tag Helpers**, nó đính kèm **trực tiếp** vào các thẻ **HTML** tạo ra sự **đơn giản về dễ dàng** khi sử dụng
- Tuy nhiên **HtmlHelper** vẫn được hỗ trợ trên **ASP.NET Core MVC**.

Một số HtmlHelper

```
@Html.BeginForm()
@Html.EndForm()
@Html.Label()
@Html.TextBox()-Html.Hidden()
@Html.Password()
@Html.TextArea()
@Html.CheckBox()
@Html.RadioButton()
@Html.DropDownList ()
```

```
@Html.ListBox
@Html.EditorFor
@Html.ActionLink()
@Url.Action()
@Html.Partial()
@Html.RenderPartial()
@Html.Raw()
@Html.Display()
@Html.DisplayName()
```

Một số HtmlHelper Strongly Types

```
@Html.LabelFor()
@Html.TextBoxFor()-Html.HiddenFor()
@Html.PasswordFor()
@Html.TextAreaFor()
@Html.CheckBoxFor()
@Html.RadioButtonFor()
@Html.DropDownListFor()
@Html.ListBoxFor()
@Html.EditorFor()
@Html.DisplayFor()
@Html.DisplayNameFor()
```

Sử dụng cho Strongly Types View

Tag Helpers

- Tag Helpers là các lớp viết bằng C# và được đính kèm vào các thẻ HTML trong View để chạy phía server-side và thông dịch bởi Razor View Engine. Một số các Tag helps phổ biến được tạo sẵn là Anchor tag, Form Tags, Environment tag, Cache tag,....
- Để sử dụng các Tag Helpers bạn có thể lấy về từ Nuget và tham chiếu tới thư viện Microsoft.AspNetCore.Mvc.TagHelpers. Sau đó hãy nhập nội dung sau vào tệp _ViewImports.cshtml (nếu tạo project với template MVC thì nó được sinh ra sẵn)

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Một số Tag Helpers

Thẻ liên kết:

```
<a asp-area = "[area_name]" asp-controller = "controller_name" asp-action =
"action_name" asp-route-parame_name="value">Link Text</a>
```

Thẻ form:

```
<form asp-controller = "controller_name" asp-action = "action_name" method="post">
</form>
```

Thẻ lable:

```
<label asp-for="Property of Model"></label>
```

Thẻ input:

```
<input asp-for="Property of Model"/>
```

Một số Tag Helpers

Thẻ textarea:

```
<textarea asp-for="Property of Model"></textarea>
```

Thẻ span với validate data:

```
<span asp-validation-for="Property of Model"></span>
```

Thẻ select:

```
<select asp-for="Property of Model" asp-items="List of SelectListItem"></select>
```

Thẻ partial:

```
<partial name="name of partial view"/>
```

(Core 2.1)

Một số các phương thức khác dùng trong View

- **@RenderBody():** dùng trong các tệp layout để sinh ra các view tại vị trí đặt lệnh. RenderBody chỉ được xuất 1 lần duy nhất trong layout.
- **@RenderSection():** dùng trong các layout để sinh ra 1 phần của view tại vị trí đặt lệnh. RenderSection được thực thi 1 lần duy nhất.
- **@await component.InvokeAsync**(): Gọi 1 component thực thi

Tệp _Layout.cshtml và _ViewStart.cshtml

- **_Layout.cshtml** là tệp dùng trình bày bố cục trang web, trong đó chứa cấu trúc trang HTML và những phần dùng chung cho các View
- **_Layout.cshtml** được sinh tự động và đặt trong thư mục Views/Shared khi người dùng tạo ứng dụng ASP.NET Core MVC với template có sẵn
- **_Layout.cshtml** được gọi trong tệp _ViewStart.cshtml
- **_ViewStart.cshtml** là tệp mặc định được gọi đầu tiên trước khi các view được render

```
@{
    Layout = "_Layout";
}
```

Bố cục tệp _Layout.cshtml

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
        <!--link css-->
</head>
<body>
        @RenderSection("S1")

        @RenderBody()

        <!--link js-->

        @RenderSection("S2")

</body>
</html>
```

index.cshtml

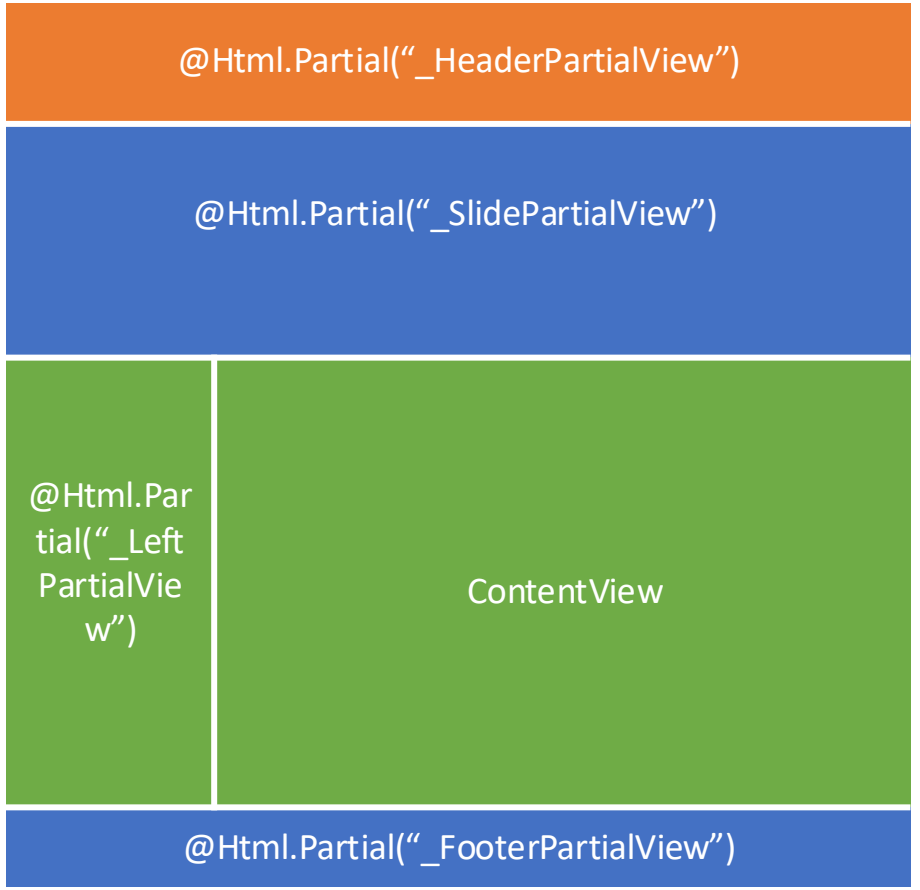
```
<h1>Nội dung sinh ra tại vị trí RenderBody!</h1>
```

```
@section S1{
        <h2>Phần đầu trang</h2>
}
```

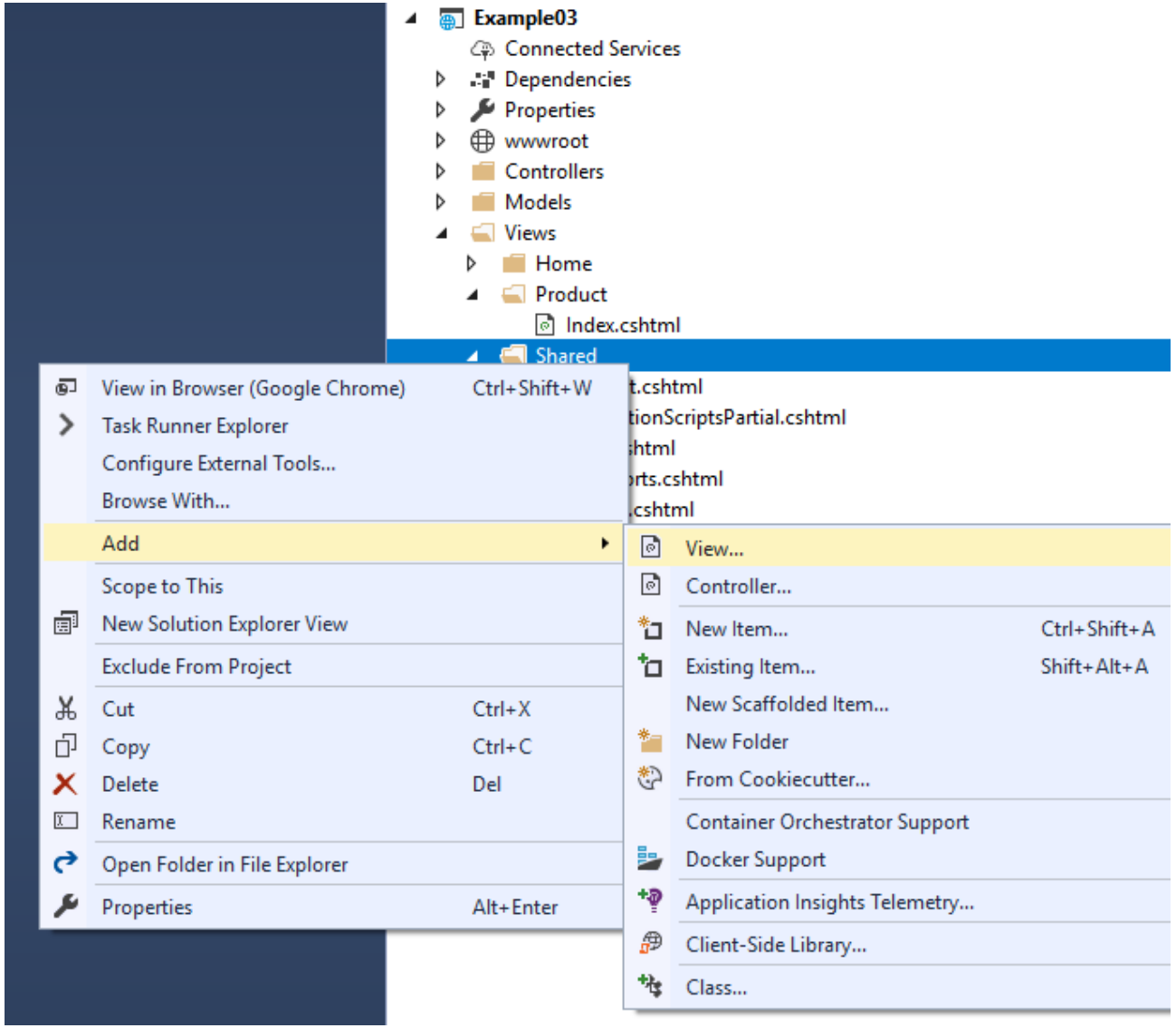
```
@section S2{
    <script>
        function checkStatus()
        {
                alert('OK');
        }
    </script>
}
```

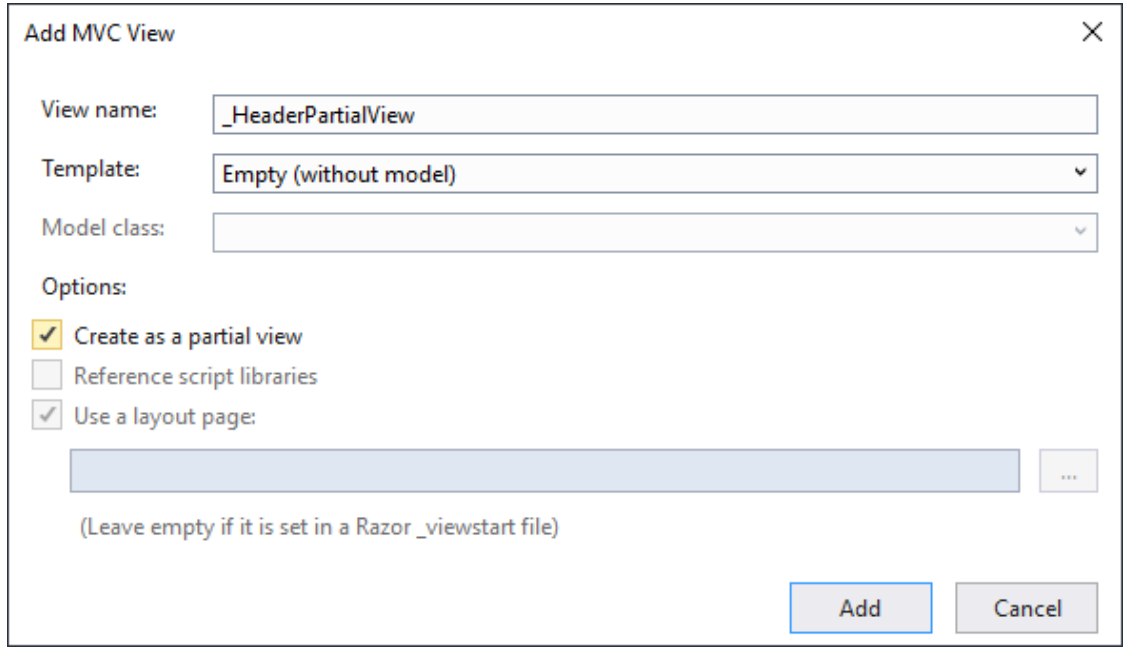
PartialView

- PartialView là một tệp .cshtml được render ra trong phần render của View khác.
- Nó giúp chia tệp .cshtml lớn thành các phần nhỏ
- Nó giảm sự trùng lặp những phần markup được xuất hiện ở nhiều View khác nhau.



Tạo PartialView





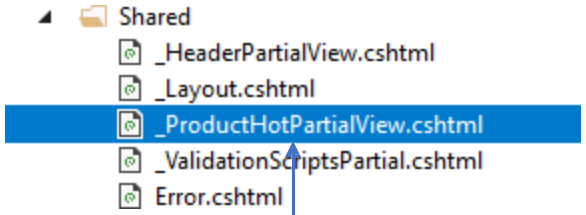
Right click vào thư mục Shared->Add->View -> Nhập tên PartialView. Lưu ý chọn Option "Create as a partial view"

Tạo PartialView

- Bạn cũng có thể tạo 1 Action để trả về PartialView phục vụ cho việc gọi về bằng Ajax

```
/*
 * Tạo action trả về danh sách các sản phẩm HOT
 * Hiển thị trể partial view _ProductHotPartialView
 */
public PartialViewResult GetProductHot()
{
    List<Product> productHot = new List<Product>();
    //lấy dữ liệu ra productHot
    //.........................
    //trả về partialview
    return PartialView("_ProductHotPartialView", productHot);
}
```

Code action



Tệp PartialView

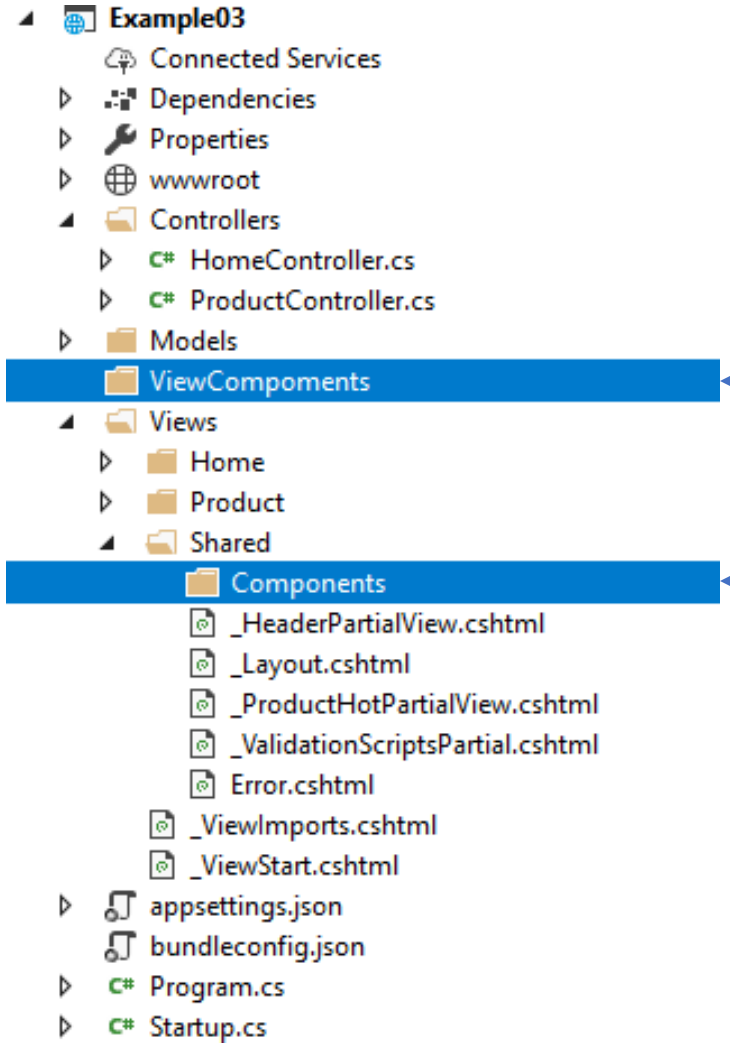
```
@model IEnumerable<Example03.Models.Product>
<h1>SẢN PHẨM HOT</h1>
<!--
    Đọc sản phẩm trong biến Model ra màn hình
-->
```

Code PartialView

View Component

- **View component** là 1 thành phần mới được giới thiệu trong **ASP.NET Core**, nó tương tự như **PartialView**.
- View component là đối tượng **riêng biệt** không phụ thuộc vào **controller**.
- View component là một **class C#** kế thừa từ lớp **ViewComponent**, có **Data Model** riêng biệt, có thể nhận các **tham số**, **đồng nhất** khi gọi ở các View.
- View component được gọi trên trang Razor và cũng render ra HTML.
- View component dễ dàng **test** và ít **bugs**.

Tạo View Component

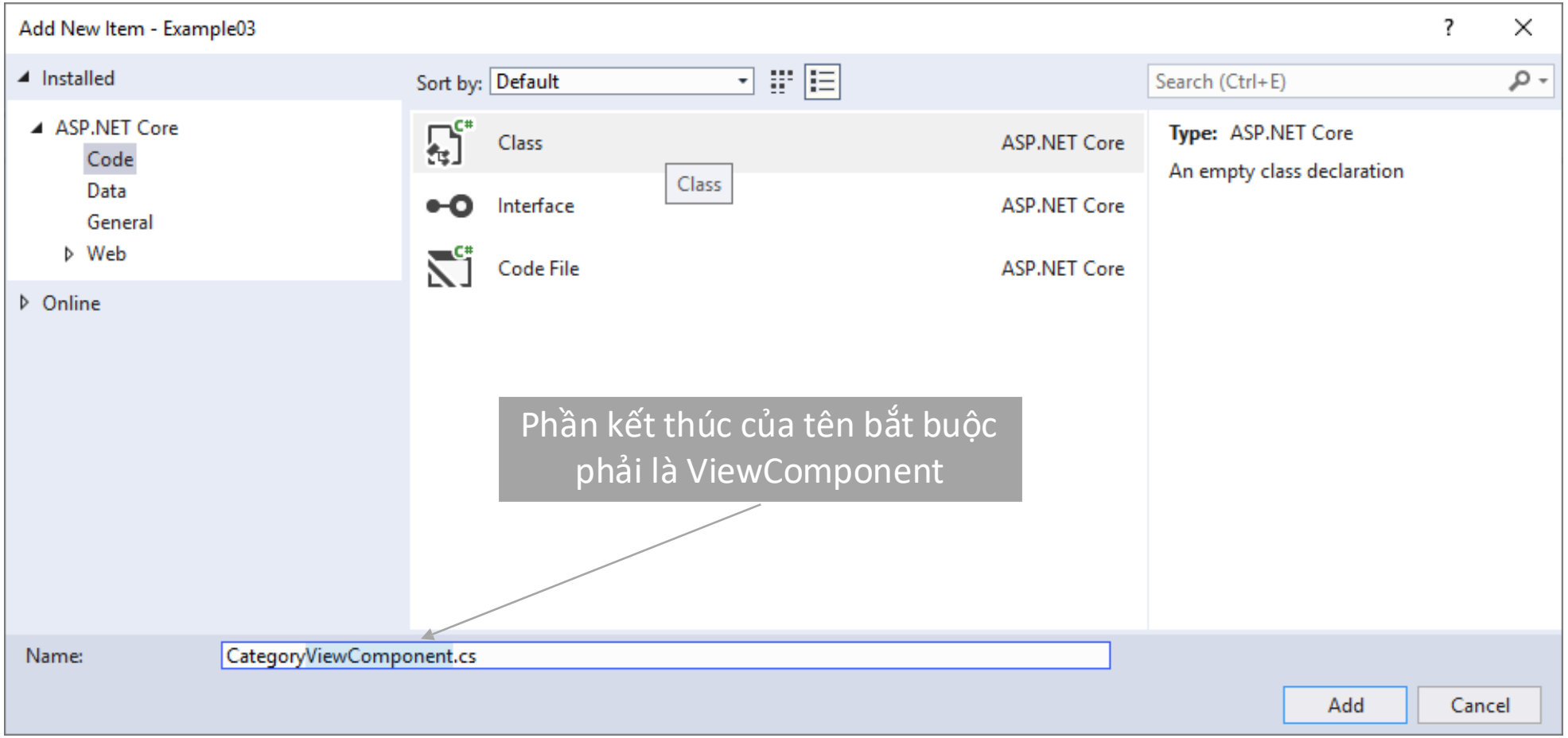


Tạo thư mục ViewComponents trong thư mục gốc của Projet để chứa các lớp ViewComponent logic. Các ViewComponent logic luôn có phần kết thúc của tên là ViewComponent. **Chúng ta có thể lưu trữ ở bất kỳ thư mục nào cũng được.**

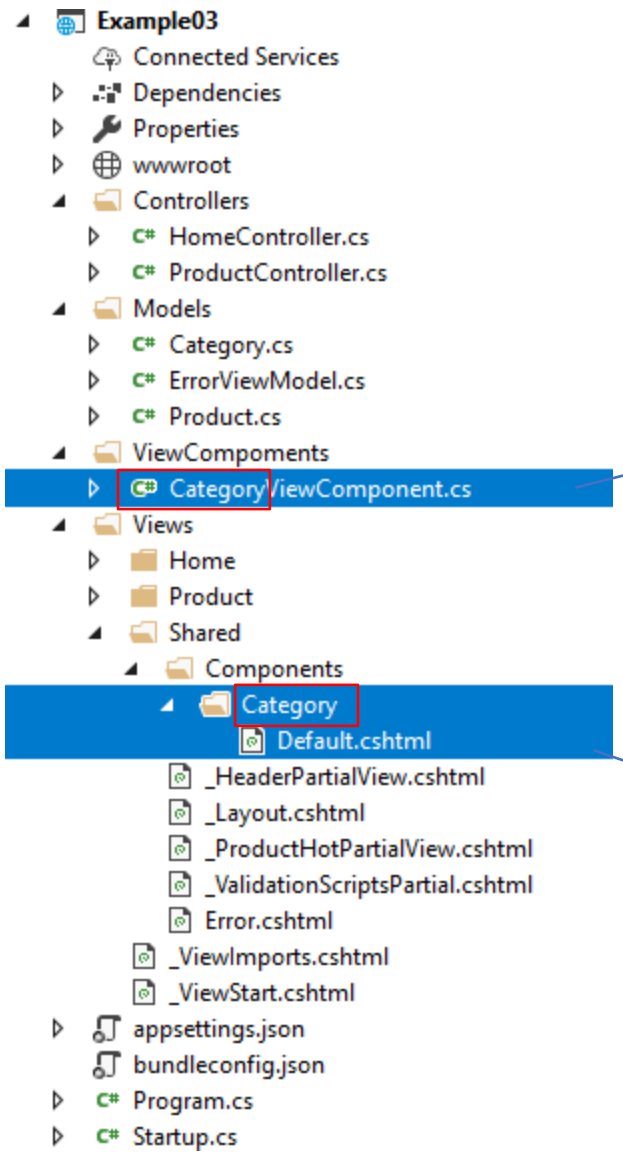
Tạo thư mục Components trong Views/Shared để chứa các ViewComponent Razor. Tên và vị trí thư mục Components là bắt buộc

Tạo View Component

- Chuột phải vào thư mục ViewComponents và thêm class mới



Tạo View Component



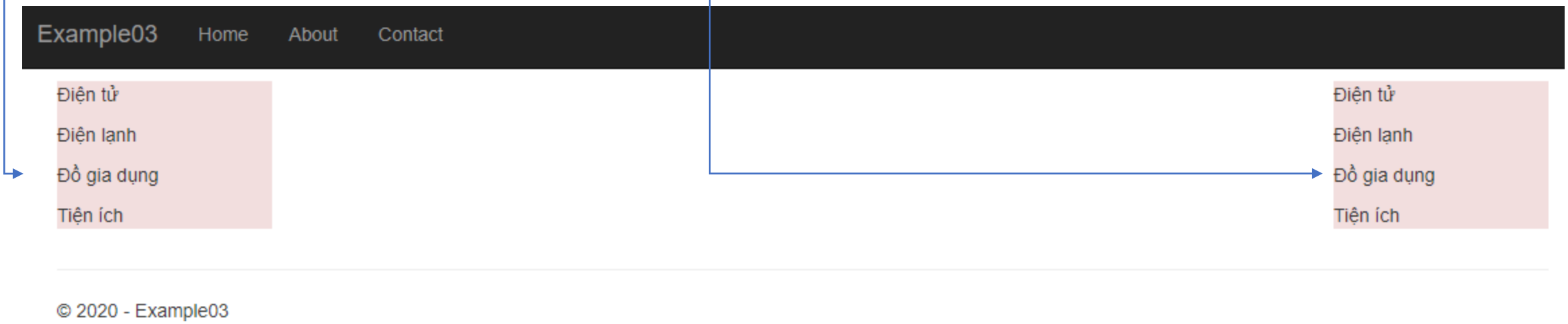

```
public class CategoryViewComponent:ViewComponent
{
    public IViewComponentResult Invoke()
    {
        List<Category> categories = new List<Category>()
        {
            new Category{CategoryId=1,CategoryName="Điện tử"},
            new Category{CategoryId=2,CategoryName="Điện lạnh"},
            new Category{CategoryId=3,CategoryName="Đồ gia dụng"},
            new Category{CategoryId=4,CategoryName="Tiện ích"}
        };
        return View(categories);
    }
}
```

```
@model IEnumerable<Example03.Models.Category>
<div class="bg-danger" style="margin-top:10px;">
    @foreach (var c in Model)
    {
        <p>@c.CategoryName</p>
    }
</div>
```

Tạo View Component

```
<div class="container body-content">
    <div class="col-md-2">
        @await Component.InvokeAsync("Category")
    </div>
    <div class="col-md-8">
    @RenderBody()
   </div>
    <div class="col-md-2">
        @await Component.InvokeAsync("Category")
    </div>
    <footer class="col-md-12">
        <hr />
        <p>&copy; 2020 - Example03</p>
    </footer>
</div>
```

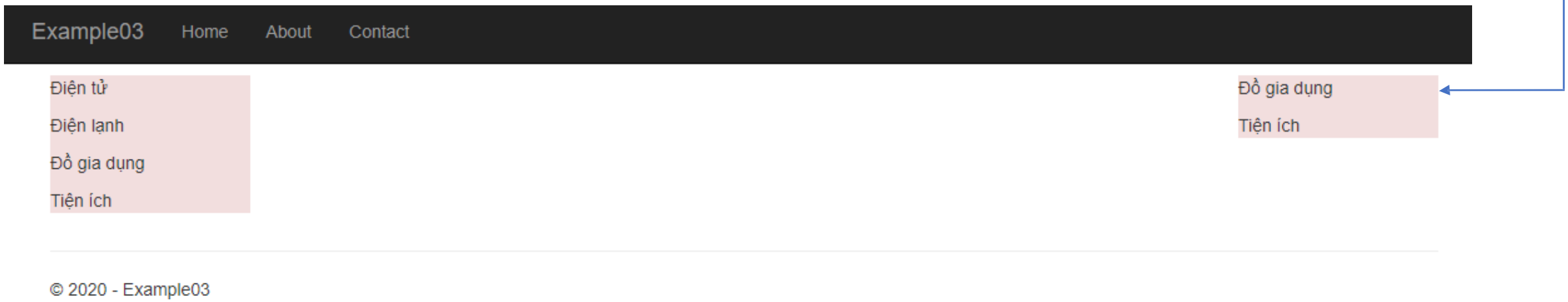
Gọi ViewComponent trong trang Razor "_Layout.cshtml". ViewComponent có thể được gọi lại nhiều lần và ở bất kỳ trang Razor nào.



Truyền tham số cho View Component

```
public class CategoryViewComponent:ViewComponent
{
    public IViewComponentResult Invoke(int? n)
    {
        List<Category> categories = new List<Category>()
        {
            new Category{CategoryId=1,CategoryName="Điện tử"},
            new Category{CategoryId=2,CategoryName="Điện lạnh"},
            new Category{CategoryId=3,CategoryName="Đồ gia dụng"},
            new Category{CategoryId=4,CategoryName="Tiện ích"}
        };
        n = n ?? 0;
        var search = categories.Where(x => x.CategoryId > n);
        return View(search);
    }
}
```

```
<div class="container body-content">
    <div class="col-md-2">
        @await Component.InvokeAsync("Category")
    </div>
    <div class="col-md-8">
    @RenderBody()
   </div>
    <div class="col-md-2">
        @await Component.InvokeAsync("Category", new { n = 2 })
    </div>
    <footer class="col-md-12">
        <hr />
        <p>&copy; 2020 - Example03</p>
    </footer>
</div>
```



HỎI ĐÁP