Lập trình web

ASP.NET CORE MVC

GV: Nguyễn Huy Cường
Email: nh.cuong@hutech.edu.vn

Nội dung

1. Giới thiệu về ASP.NET Core MVC

- 📖 Tổng quan ASP.NET Core MVC
- 📖 Cấu trúc project
- 📖 Cơ chế hoạt động
- 📖 Định tuyến

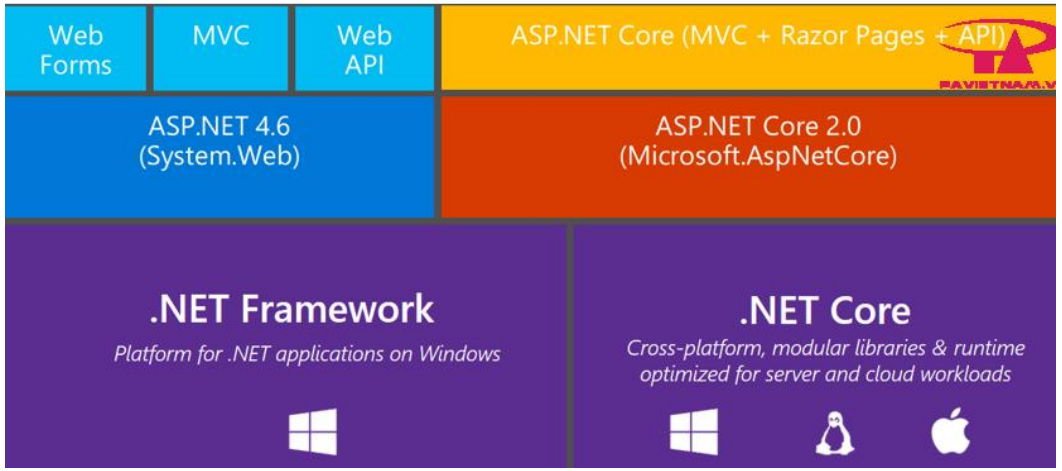
2. Lập trình với ASP.NET Core MVC

- 📖 Tạo controller
- 📖 Tạo Action trong controller
- 📖 Tạo View trong Action
- 📖 Truyền tham số
- 📖 Truyền dữ liệu từ Action sang View
- 📖 Layout, Sử dụng template

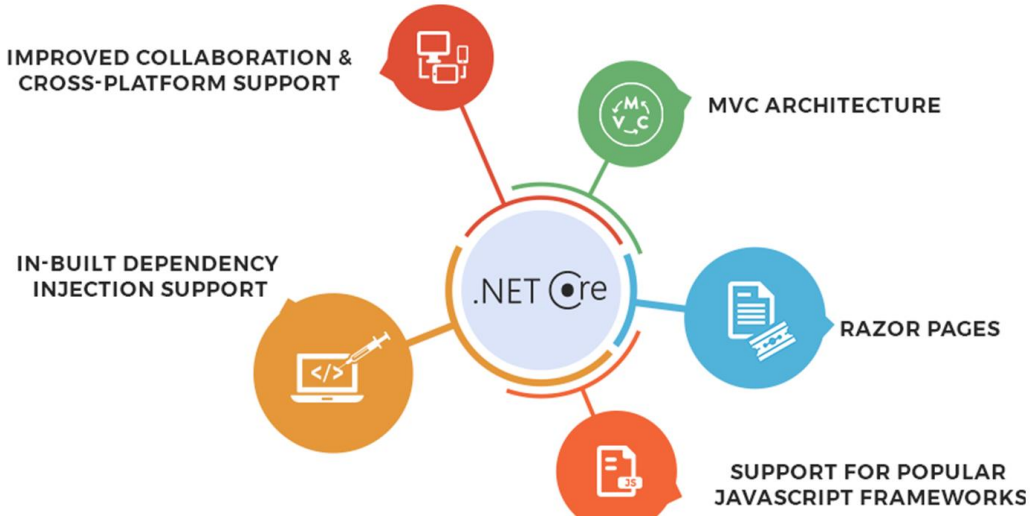
GIỚI THIỆU VỀ ASP.NET CORE MVC

ASP.NET Core MVC là gì?

- Là một phần của framework ASP.NET Core. MVC là viết tắt của 3 phần đó là **model, controller** và **view**.
  - Model: dữ liệu & logic, đối tượng, tập tin ..
  - Controller: xử lý trung gian giữa model và view.
  - View: Giao diện người dùng.

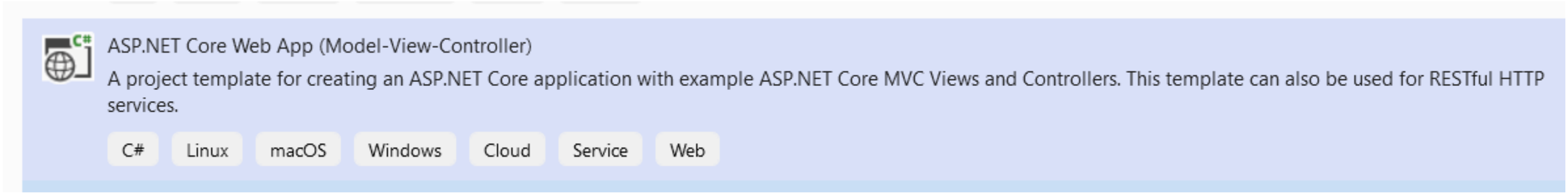


- Ưu điểm sử dụng ASP.NET Core MVC
  - Phát triển ứng dụng web đa nền tảng
  - Cung cấp tính năng: routing, dependency injection, middleware, cấu hình nhanh chóng...
  - Razor cung cấp ngôn ngữ tạo views, Tag Helper
  - Model Binding tự động ánh xạ dữ liệu từ HTTP request tới tham số của method action

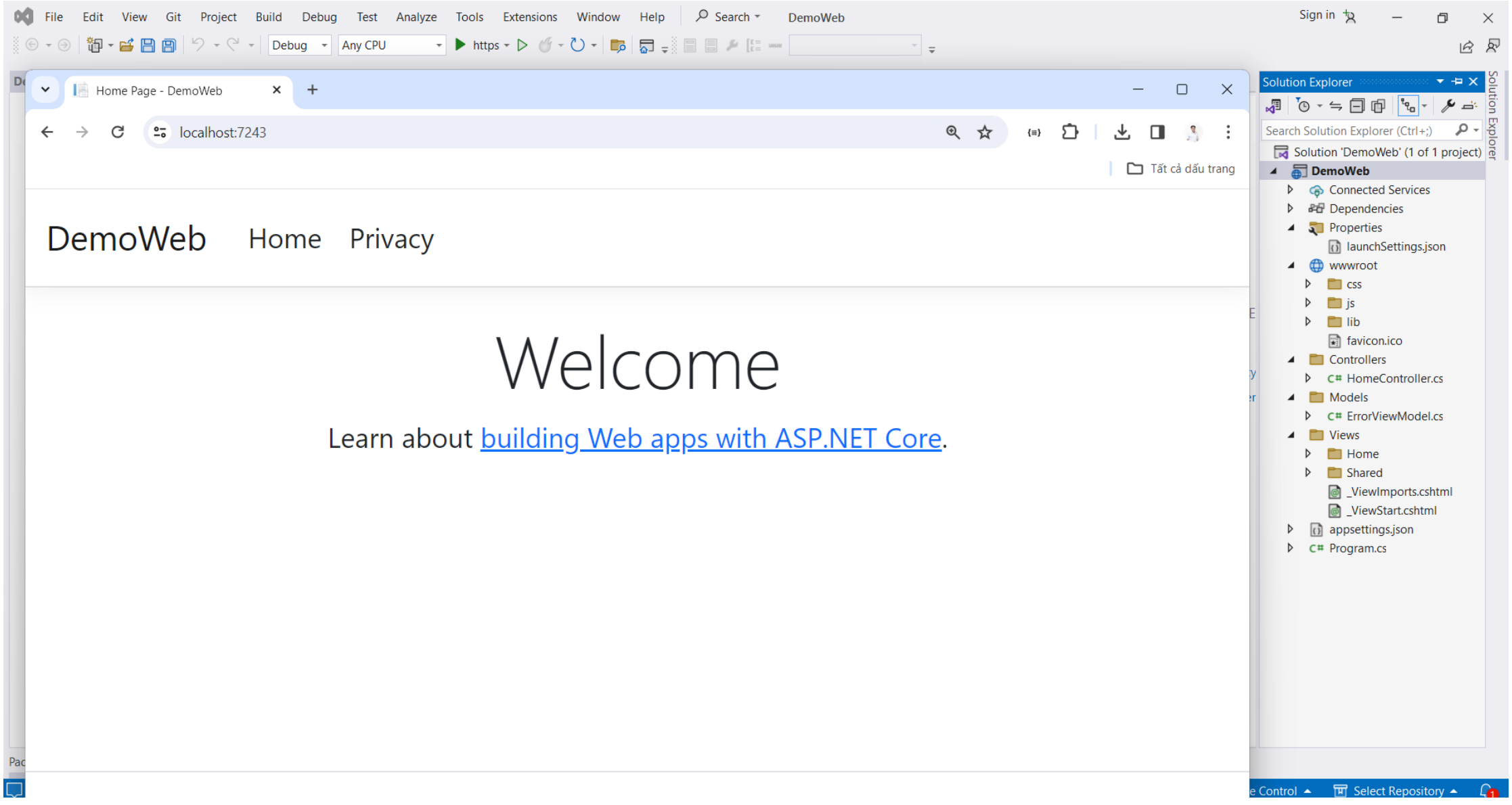


Tạo project ASP.NET Core MVC

Sử dụng VS2022 tạo project với template ASP.NET core Web App (Model-View-Controller)



DEMO Kết quả khi chạy project



Cấu trúc project

Cấu trúc file và thư mục của project ASP.NET Core MVC

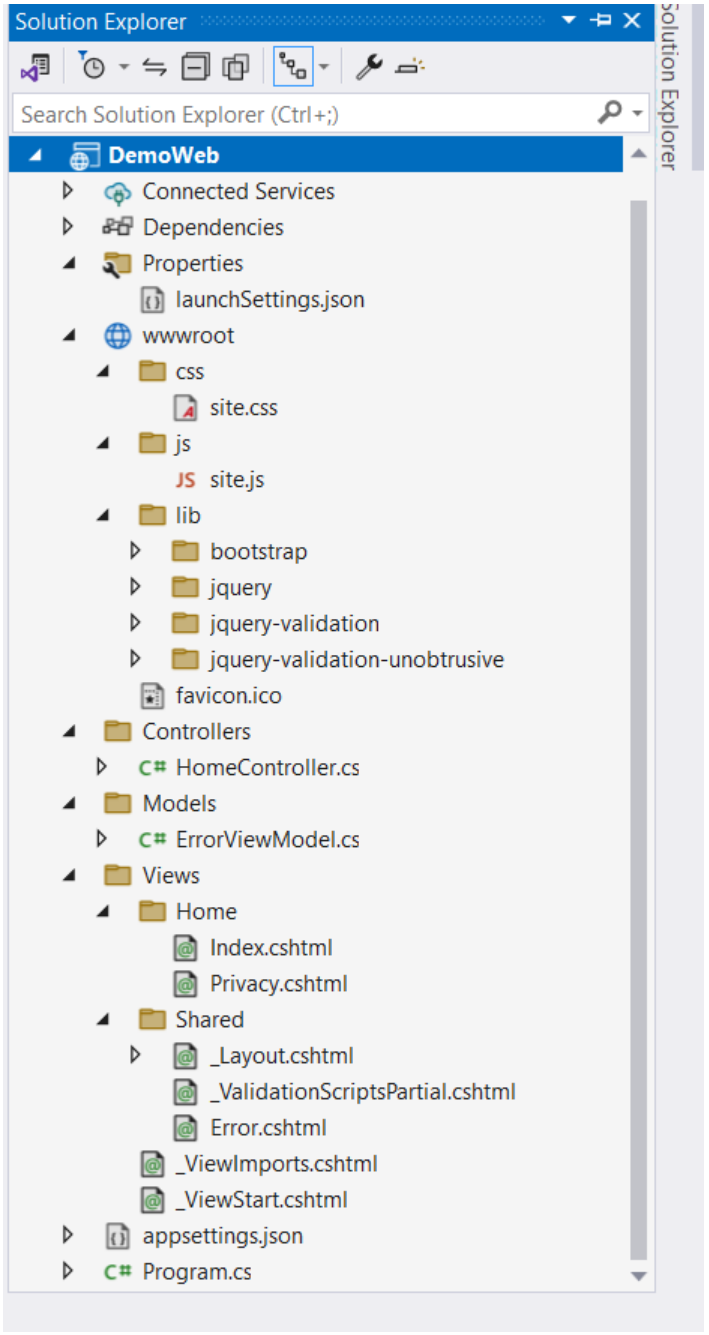
- **wwroot:** chứa những file

  tĩnh như *html, javascript, CSS, images, ...*

- **Dependencies:** chứa các thư viện cài đặt từ Nuget

- **Program.cs:** chứa các mã khởi tạo cho ứng dụng

- **Appsetting.json:** lưu trữ các cấu hình cho ứng dụng



Cơ chế hoạt động

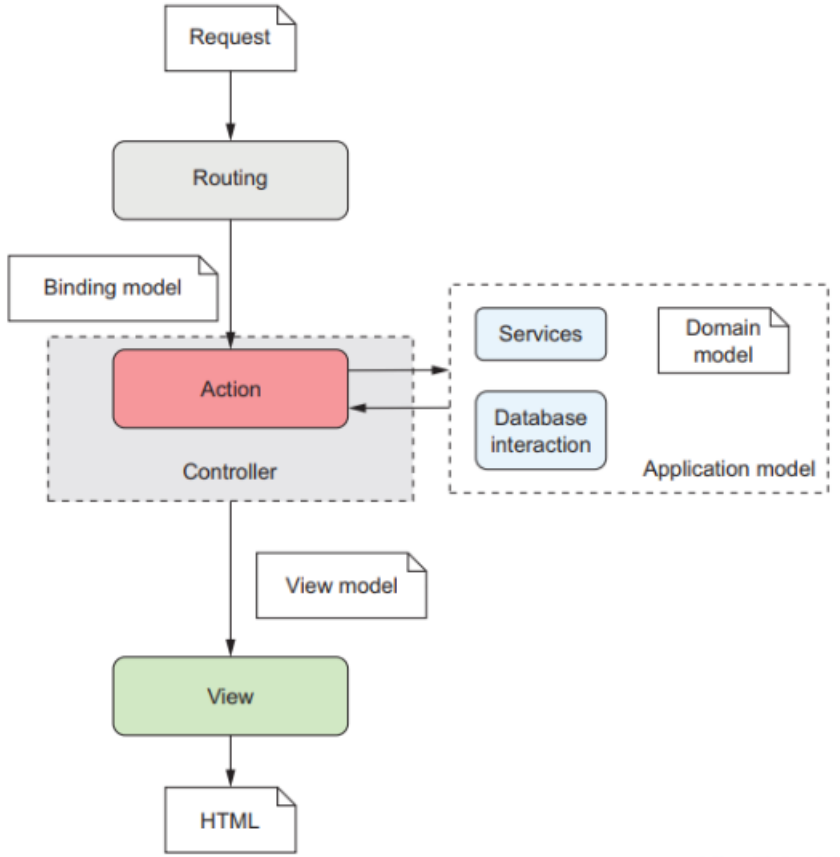
(1) User/Client gửi 1 yêu cầu tới server bằng URL từ browser

(2) Truy vấn HTTP được cơ chế **routing** ánh xạ sang 1 phương thức xác định là **Action**

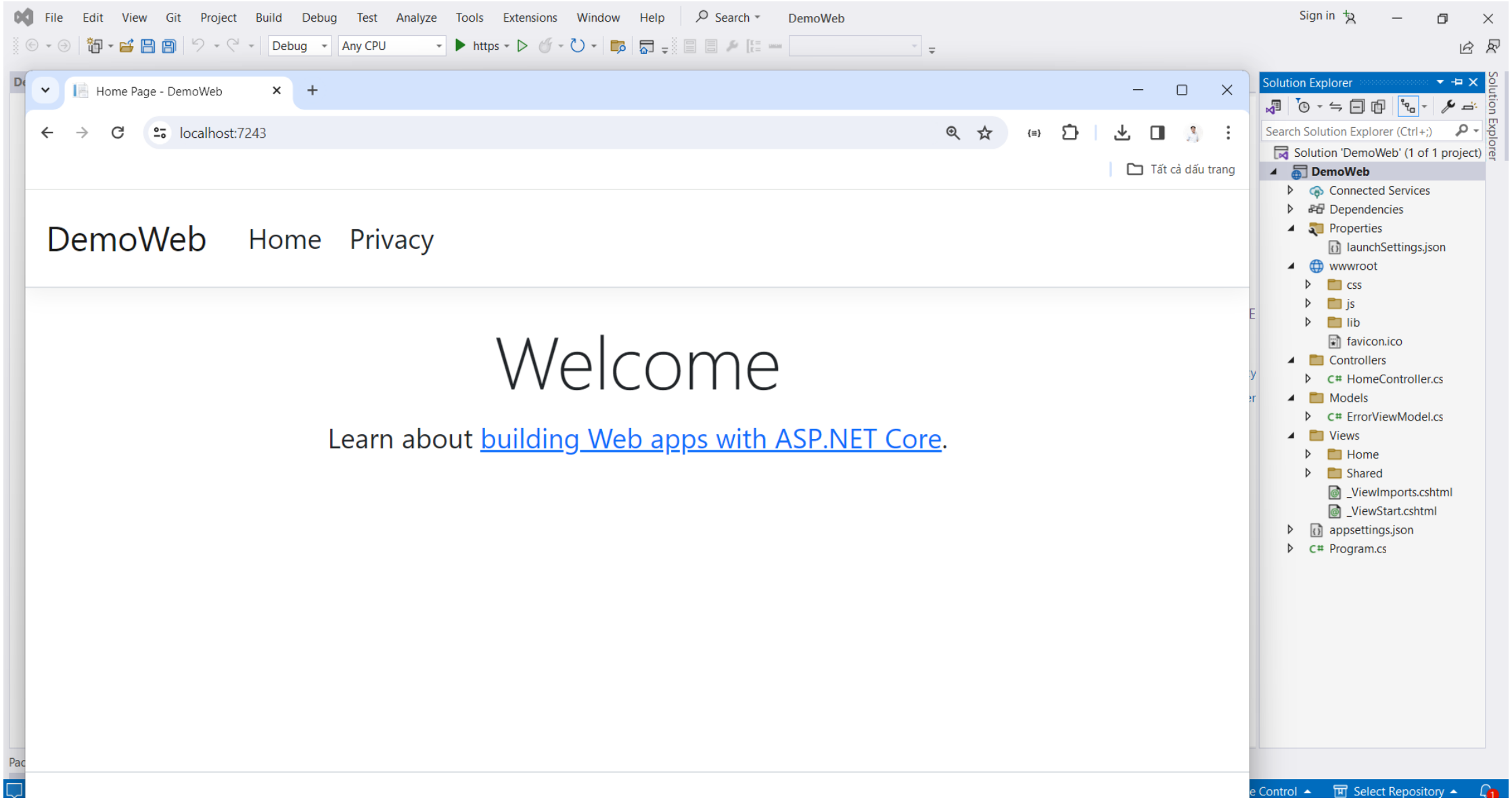
(3) Action được chọn xử lý:

- Nếu phải truy xuất dữ liệu thì controller sẽ tiến hành chuyển qua tầng model.
- Ở Model, dữ liệu sẽ được truy xuất từ CSDL và được truyền qua view nhờ controller.
- Dữ liệu sẽ được chuyển từ Model sang View nhờ Controller.

(4) View trả dữ liệu cho client.



DEMO Cơ chế hoạt động của ASP.NET Core MVC



Định tuyến (Routing) trong MVC

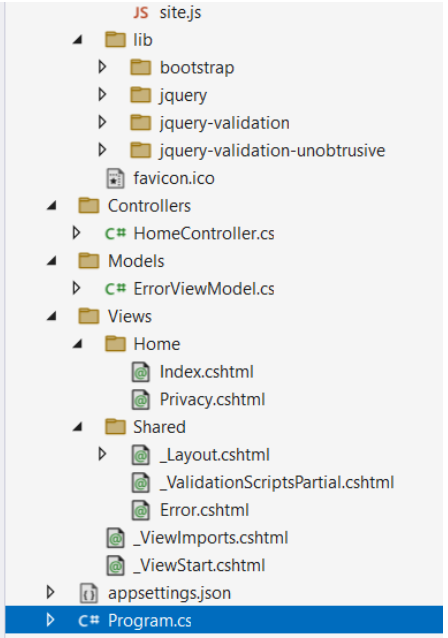
- Được đăng kí `MapControllerRoute` trong **Program.cs**
- Thông qua url (Liên kết) nó mô tả hành động của người dùng một cách chi tiết

```
app.MapControllerRoute(name: "test",
                pattern: "test",
                defaults: new { controller = "Home", action = "Index" });

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

lo issues found



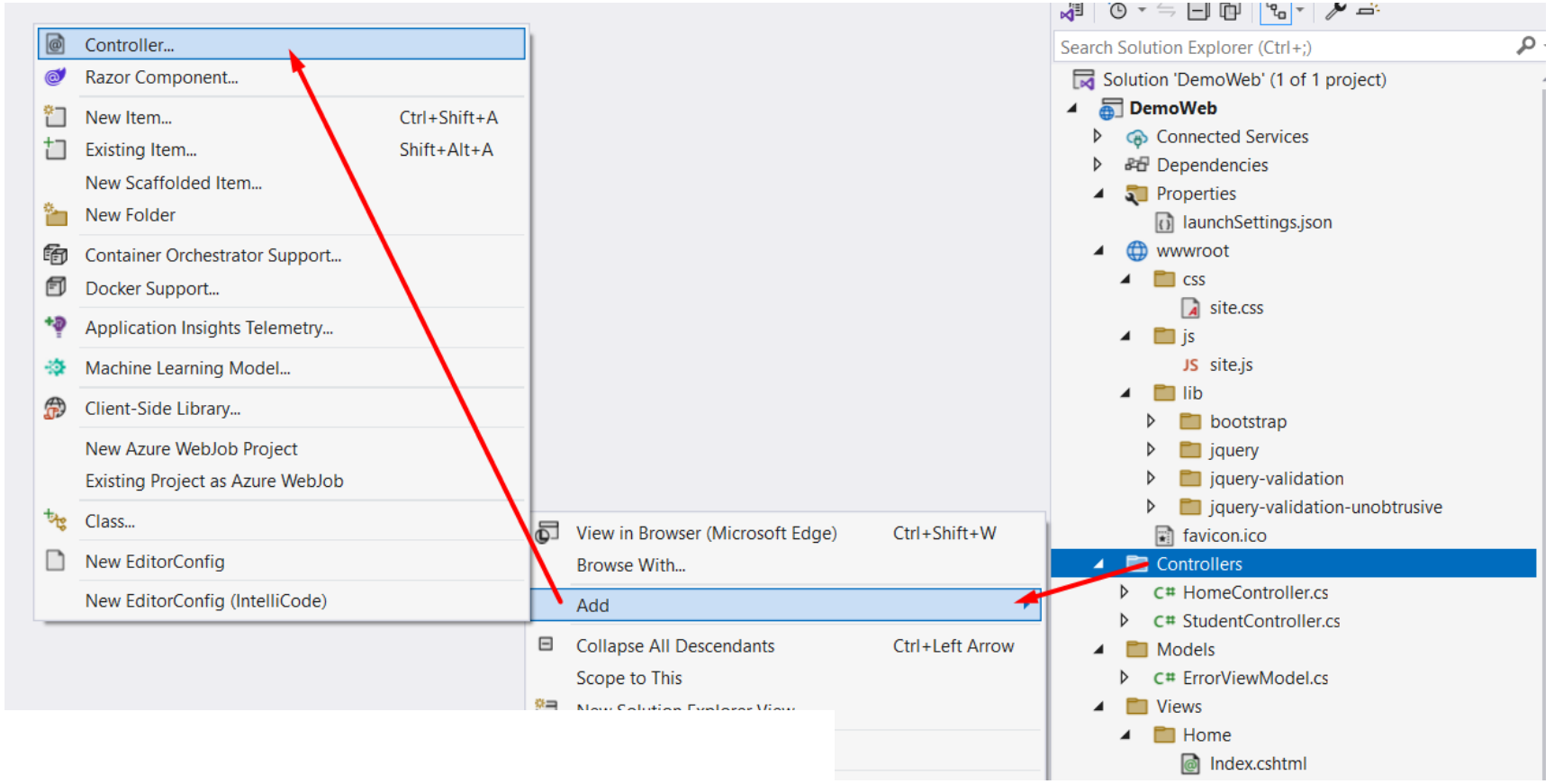
- Properties của Route (name, pattern, defaults)

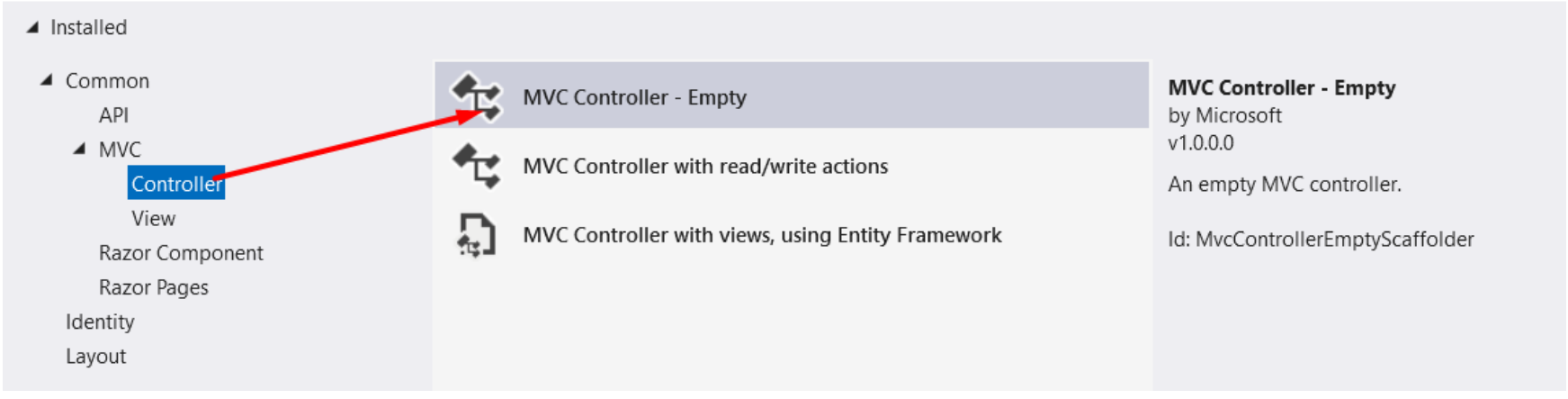
  pattern: phân cách bằng / và mẫu *{controller}/{action}/{id}*
- Default value sẽ được sử dụng khi trong URL không chứa params
- Có thể đăng kí nhiều routes, ưu tiên route theo thứ tự đăng kí

LẬP TRÌNH VỚI ASP.NET CORE MVC

Tạo controller

- Tạo controller:
  - Tạo **Action**
  - Tạo **View** cho Action





Tạo Action cho Controller

ActionResult

Action Result quy định nhiều loại kiểu dữ liệu trả về khác nhau sau khi thực hiện xong 1 Action trong Controller.

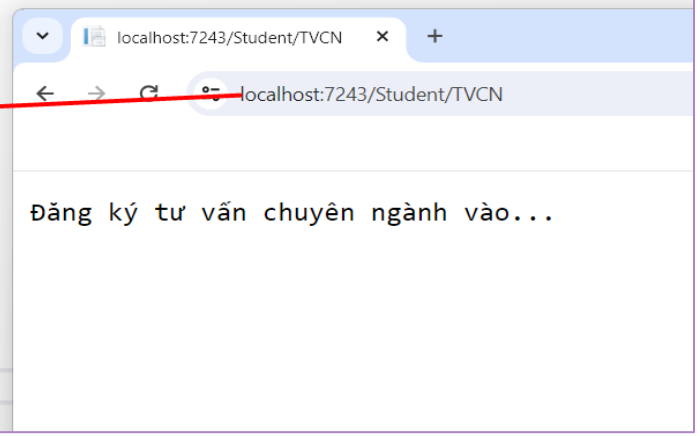
Tên Action Result	Mô tả	Hàm sử dụng
ContentResult	Trả về chuỗi	**Content**()
FileContentResult/ FilePathResult/ FileStreamResult	Trả về nội dung file	File()
JavaScriptResult	Trả về nội dung JavaScript	JavaScript()
JSonResult	Trả về dữ liệu dạng Json	Json()
RedirectResult	Chuyển sang URL mới	**Redirect**()
RedirectToRouteResult	Chuyển sang 1 Action hoặc 1 Action của controller khác	RedirectToRoute() **RedirectToAction**()
ViewResult	Chuyển sang View để hiển thị	**View**()
PartialViewResult	Chuyển sang View để hiển thị không layout	**PartialView**()

Tạo Action cho Controller

Action Selectors

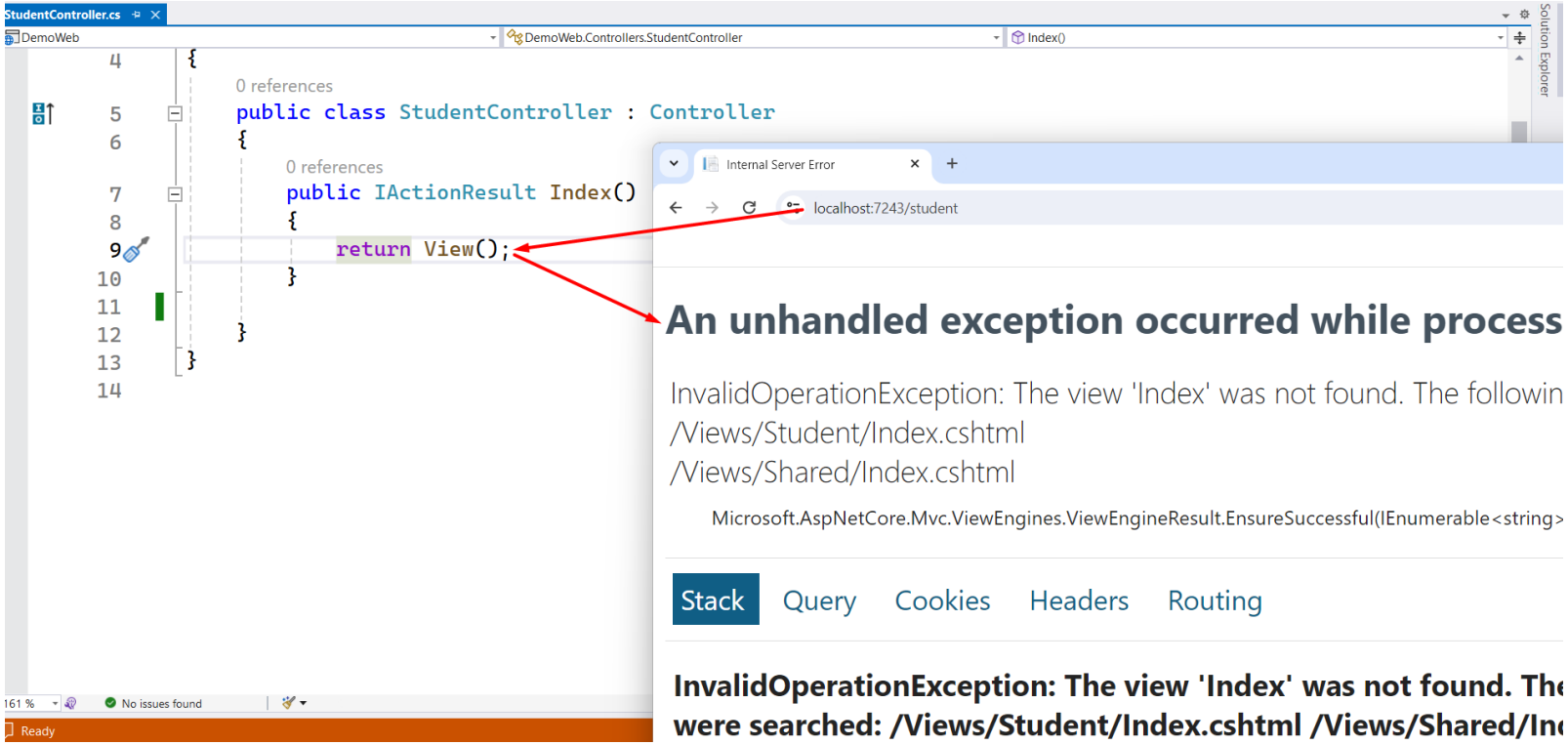
- Một Action sau khi được định nghĩa có thể được gọi theo cả **POST** và **GET**
- Nếu muốn chỉ gọi với **POST** / **GET** thì đánh dấu Action với [**HttpPost**] hay [**HttpGet**]
- Để đổi tên một Action sử dụng ***ActionName***

```
namespace DemoWeb.Controllers
{
    0 references
    public class StudentController : Controller
    {
        [ActionName("TVCN")]
        0 references
        public IActionResult Details()
        {
            return Content("Đăng ký tư vấn chuyên ngành vào...");
        }
    }
}
```

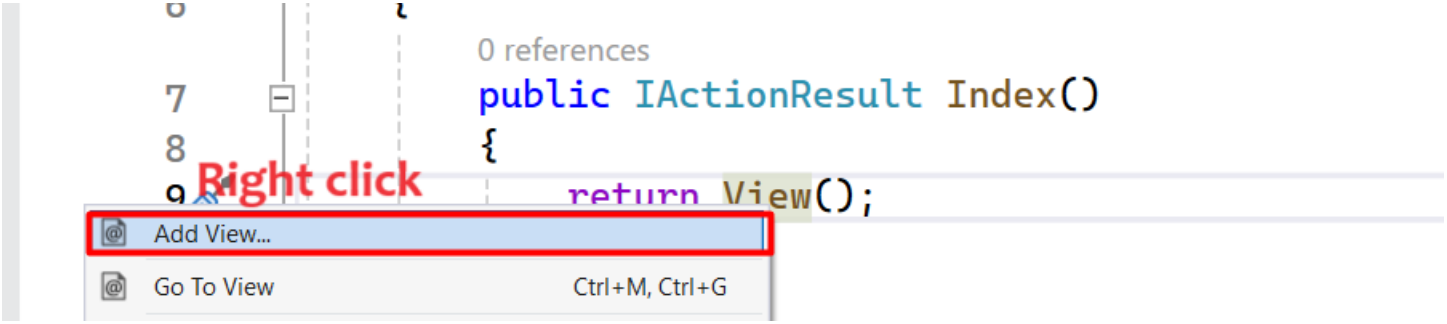


Tạo View cho Action

- Nếu view chưa được tạo sẽ gặp lỗi



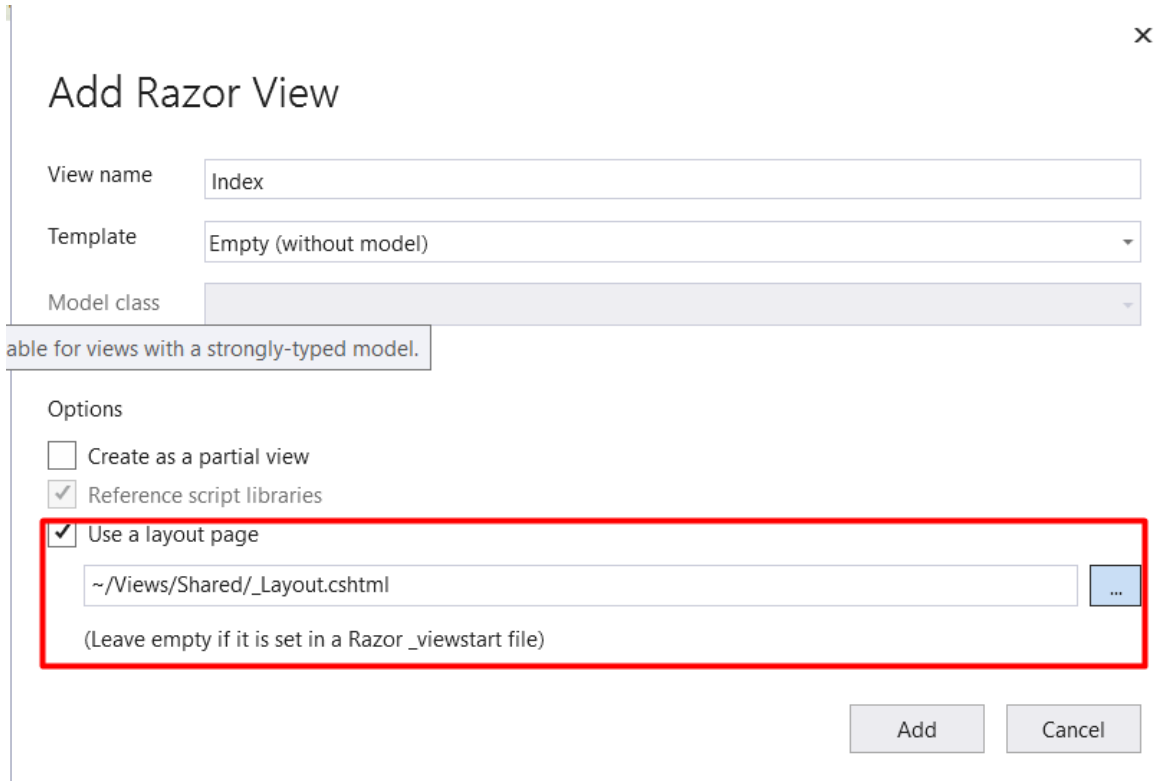
- Để tạo view trong action: Right click và tạo View, chọn **Razor View**

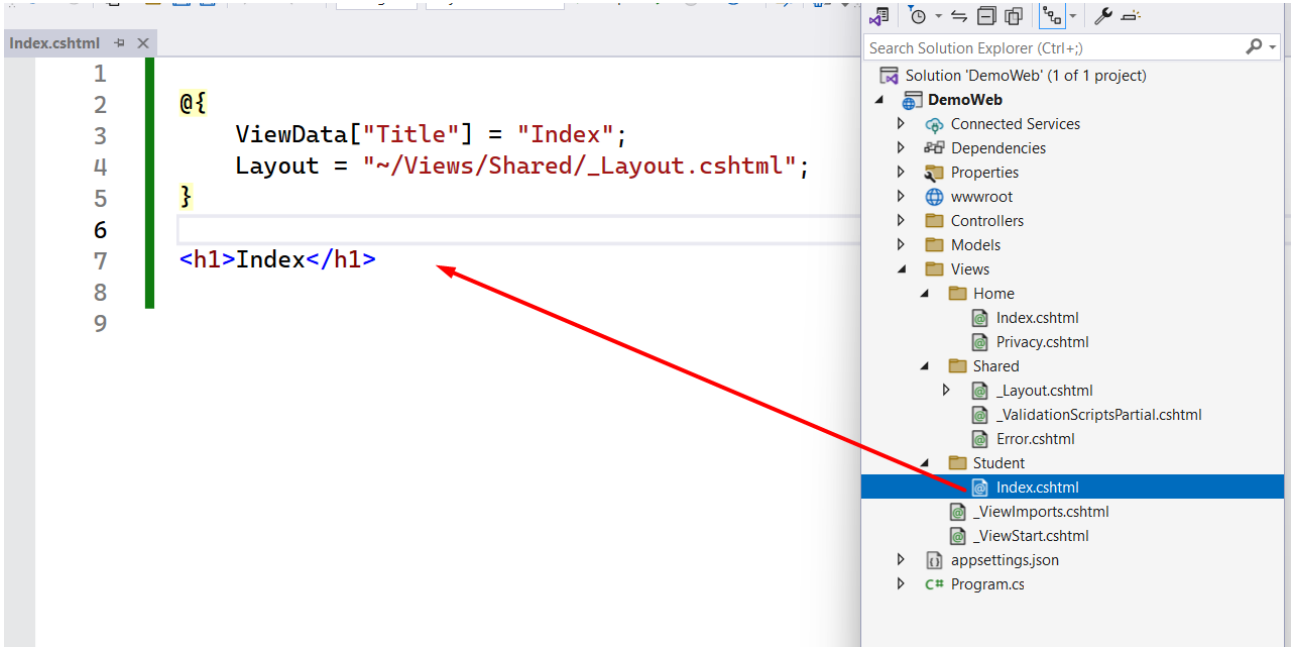


Tạo View cho Action

Cấu hình View

- ViewName / Template/ và các lựa chọn





- Một số template view mặc định cho CRUID

(chi tiết hơn ở các bài sau)

Truyền tham số

- Tham số yêu cầu từ người dùng được cung cấp dưới 2 dạng: Form field(lấy theo tên của html control) hoặc Query String(lấy theo link href)

```
<form action="/Student/ShowKQ" method="post">
    MSSV:<input name="id"/>
    HoTen:<input name="name" />
    Lop:<input name= "className" />
    <input type="submit" value= "DK"/>
</form>
```

dạng 2:

```
<a href="/Student/ShowKQ?id=1&name=Cuong&className=20DTHD2">Đăng ký hội thảo</a>
```

- Tiếp nhận tham số: Có 3 cách sử dụng: **Request**, Đối số của **Action**, và sử dụng viewmodel

truyền tham số

1. Sử dụng Request

Để lấy giá trị tham số truyền có thể viết:

- Nếu href hoặc method=**get** ➔**Request.Query**

```
int id = int.Parse(Request.Query["id"].ToString());
string name = Request.Query["name"].ToString();
string className = Request.Query["className"].ToString();
```

- Nếu method=**post** ➔**Request.Form**

```
int id = int.Parse(Request.Form["id"].ToString());
string name = Request.Form["name"].ToString();
string className = Request.Form["className"].ToString();
```

truyền tham số

2. Sử dụng đối số của Action

- ✓ Thay thế bằng các tên biến cụ thể (name của các html control trong form)..
- ✓ Định nghĩa tham số cho **Action** để ***nhận tham số cùng tên***.

```
0 references
public ActionResult ShowKQ(int id, string name, string className)
{
    return Content(string.Format("cam on ban {0}_{1} o lop {2} da dk tham du hoi thao", id, name, className));
}
```

```
<form action="/Student/ShowKQ" method="post">
    MSSV:<input name="id" />
    HoTen:<input name="name" />
    Lop:<input name="className" />
    <input type="submit" value="DK" />
</form>
```

truyền tham số

3. Sử dụng Model

- ✓ Tạo lớp data model chứa thuộc tính cùng tên với tham số.
- ✓ Sử dụng lớp này làm đối số cho Action để nhận tham số cùng tên với thuộc tính.

```
namespace DemoWeb.Models
{
    1 reference
    public class Student
    {
        1 reference
        public int Id { get; set; }
        1 reference
        public string? Name { get; set; }
        1 reference
        public string? ClassName { get; set; }
    }
}
```

```
public ActionResult ShowKQ(Student s)
{
    return Content(string.Format("cam on ban {0}_{1} o lop {2} da dk tham du hoi thao", s.Id, s.Name, s.ClassName));
}
```

Bài tập cộng điểm

1. Tạo 1 controller "**Student**", có Action **Index** với View là trang đăng kí chuyên ngành để khoa CNTT muốn biết trước số lượng
2. Trang View này (/View/Student/Index) cho phép nhập liệu các thông tin

   MSSV,

   Họ tên,

   Điểm TB ,

   Chuyên ngành (CNPM, HTTT, ANM, TTNT, MMT)

Khi submit "Đăng ký" -> sử dụng phương thức **POST** để tới trang hiển thị kết quả

3. Trang kết quả sau khi Đăng ký (/Student/**ShowKQ**)

   "Cảm ơn bạn {MSSV} - {Họ tên} đã đăng ký chuyên ngành {Chuyennganh}"

4. Cho biết số lượng SV đã Đăng ký cùng ngành học với bạn?

   "Số lượng sv đã đăng ký ngành giống bạn là : {số lượng}

Truyền dữ liệu

- Ở controller, Sử dụng **ViewBag**.Key = Value

```
public ActionResult Index()
{
    ViewBag.Message = "Đúng nhận sai cãi!";
    ViewBag.Date = DateTime.Now;
    return View();
}
```

Index.cshtml*

```
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/Layout.cshtml";
}
<h2>Index</h2>
ngày hiện tại: @ViewBag.Date.DayOfWeek
<p/>
Nội dung: @ViewBag.Message
```

- Truyền object model qua **View:** sử dụng @Model (Razor model keyword)

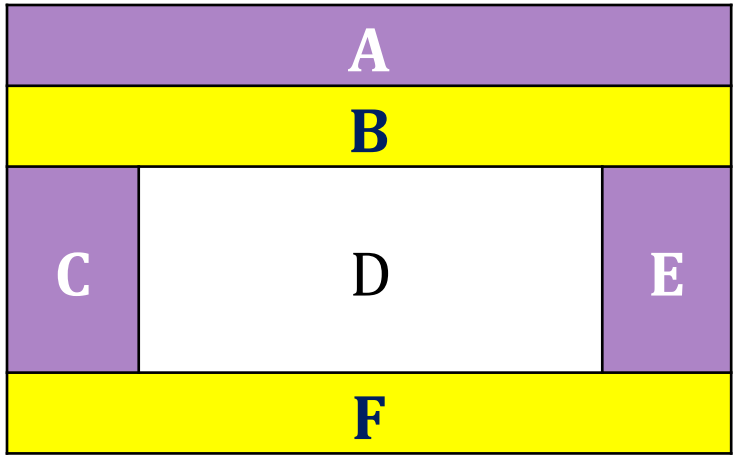
```
3 references | 0 changes | 0 authors, 0 changes
public class ObjData
{
    2 references | 0 changes | 0 authors, 0 changes
    public string Message { get; set; }
    2 references | 0 changes | 0 authors, 0 changes
    public DateTime Date { get; set; }
}
```

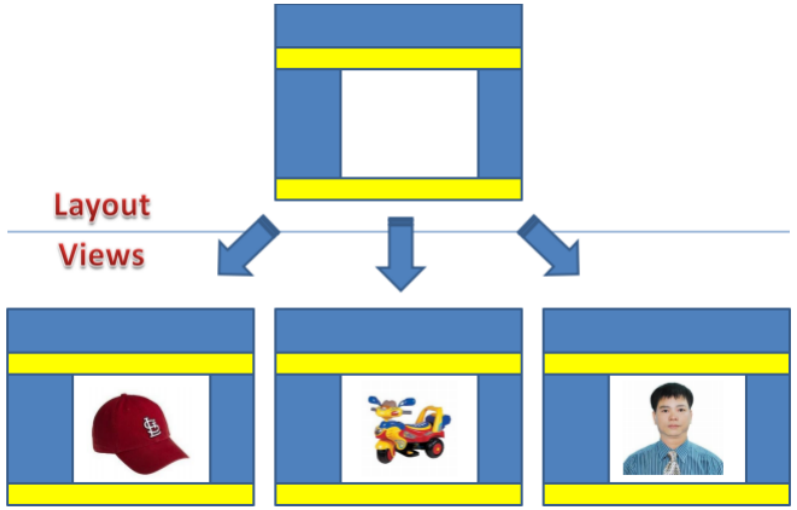
```
public ActionResult Index()
{
    var obj = new ObjData()
    { Message = "Đúng nhận sai cãi!",
      Date = DateTime.Now
    } ;
    return View(obj);
}
```

```
@model DemoWeb3.Models.ObjData
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/Layout.cshtml";
}
<h2>Index</h2>
ngày hiện tại: @Model.Date.DayOfWeek
<p />
Nội dung: @Model.Message
```

Layout

- Layout: là khung để định hình hiển thị nội dung, một trang Web sẽ được Layout ra nhiều khung, mỗi khung có thể được xem là một View hiển thị dữ liệu, các khung này có thể được tái sử dụng:





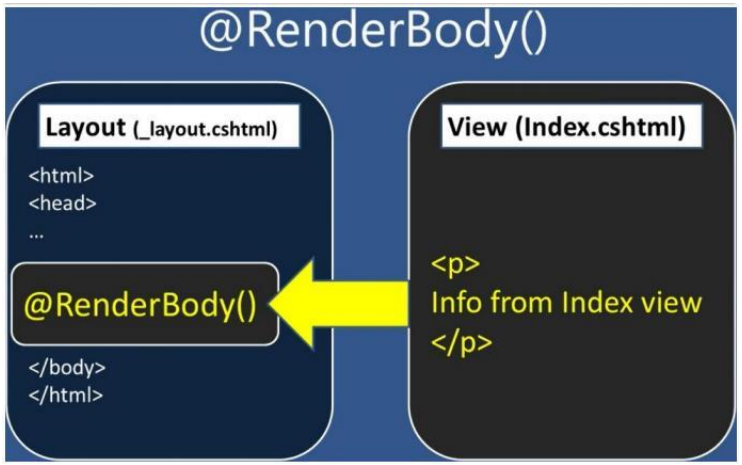
✓ Khi tạo ứng dụng ASP.NET Core, layout mặc định có tên **_Layout.cshtml** trong thư mục **Views/Shared**

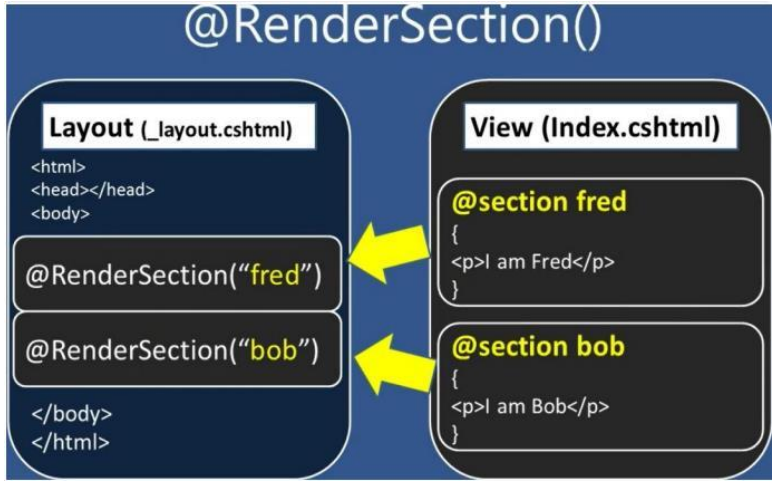
❑ Chỉ có 1 **@RenderBody()** để giữ chỗ cho nội dung trong View khi kế thừa

❑ Không hoặc nhiều @**RenderSection**(name, required:false/true) định nghĩa tên section để giữ chỗ cho các phần được đánh dấu **@section** trong view

Layout

- RenderBody() vs RenderSection()





_Layout.cshtml

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title>@ViewBag.Title - My ASP.NET Application</title>
5   </head>
6   <body>
7       <div class="navbar navbar-inverse navbar-fixed-top">…</div>
26      <div class="container body-content">
27          @RenderBody()
28          <hr />
29          <footer>
30              <p>&copy; @DateTime.Now.Year - My ASP.NE
31          </footer>
32      </div>
33  </body>
34  </html>
```

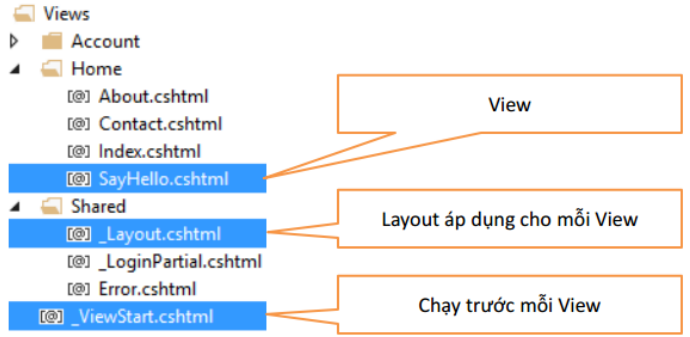
_ViewStart.cshtml

```
1   @{
2       Layout = "~/Views/Shared/_Layout.cshtml";
3   }
```

Index.cshtml

```
1   @{
2       ViewBag.Title = "Home Page";
3   }
4
5   <div class="jumbotron">…</div>
10
11  <div class="row">…</div>
```

- _ViewStart.cshtml chọn layout cho các view
  - Layout = **"~/Views/Shared/_Layout.cshtml"**
- View có thể chọn layout khác với mã tương tự



Sử dụng template với ASP.NET Core MVC

- Có thể chọn các template HTML, CSS phù hợp

Free-css: https://www.free-css.com/ themWagon https://themewagon.com/

W3Schools: https://www.w3schools.com/w3css/w3css_templates.asp

- Các bước để tích hợp template trên ASP.NET Core MVC

B1: Tìm mẫu website phù hợp

B2: Tạo thư mục ***images, css, js…*** từ template được đưa vào **wwwroot**.

B3: Thêm Razor Layout ***Layout.cshtml*** ở ***Shared*** Folder để làm master page.

- Lấy nội dung từ Index.html(Template File) vào ***Layout.cshtml***.
- Phần nội dung để `@RenderBody()`
- Điều chỉnh đường dẫn **css**, **images, js**.

B4: Thêm mới các View, cần sử dụng ***Layout.cshtml***.

DEMO Sử dụng template với MVC 5, chọn book-store template

Trang chủ

Chi tiết

ASM3: Bài tập về nhà

1. Tạo project Core MVC với tên "MSSV1_MSSV2_ASM3" để thiết kế Website bán sách như đã demo
2. Thiết kế, chỉnh sửa ở menu:

Trang chủ, Liên hệ, tìm kiếm

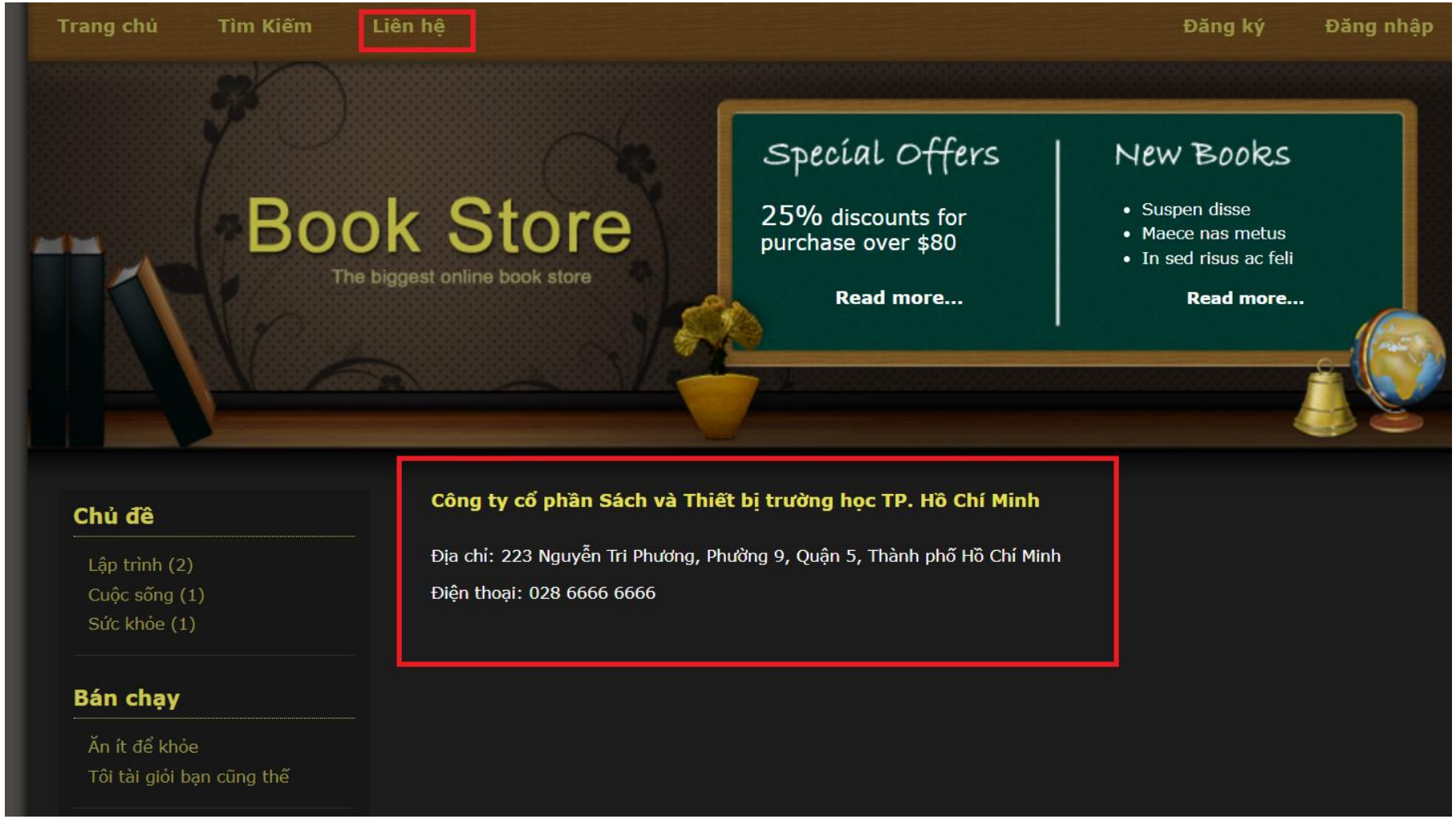
Đăng ký, Đăng nhập

ASM3:

3. **Chức năng liên hệ:** thể hiện thông tin nhà sách

/Home/Contact

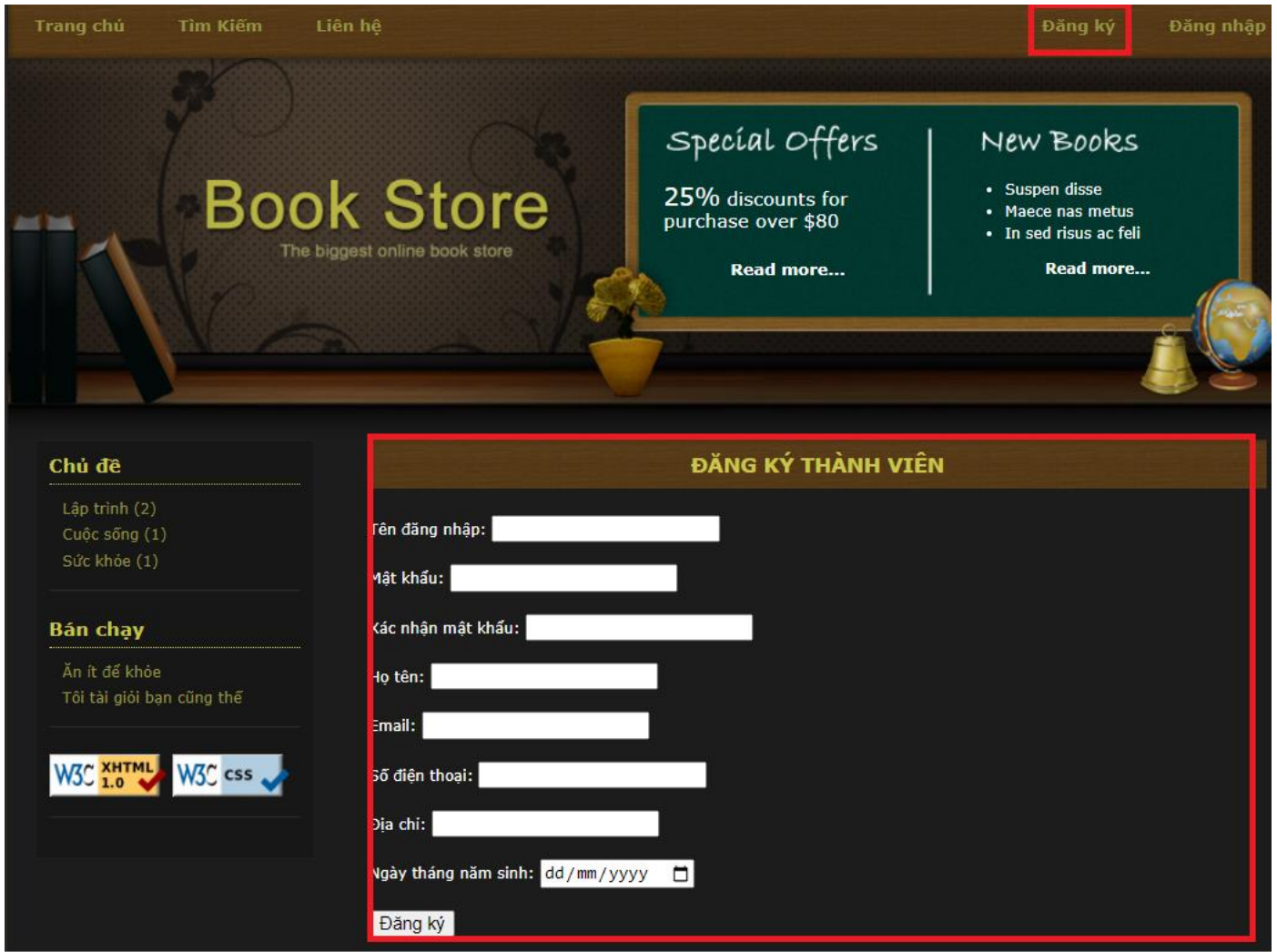
```
@Html.ActionLink("Contact", "Contact", "Home")
```



ASM3:

4. chức năng Đăng ký: thể hiện các trường cho phép đăng kí

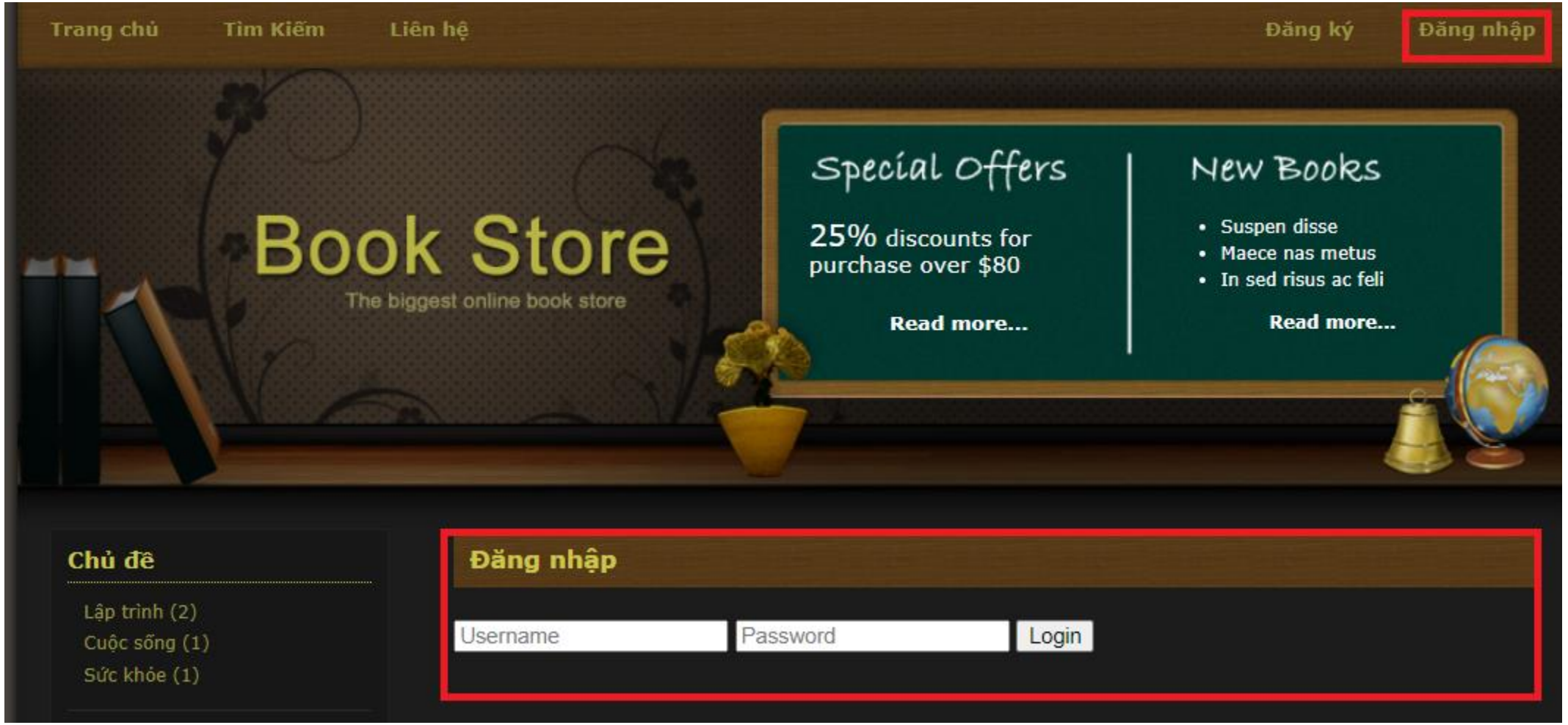
/Account/Register



ASM3:

5. chức năng Đăng Nhập:

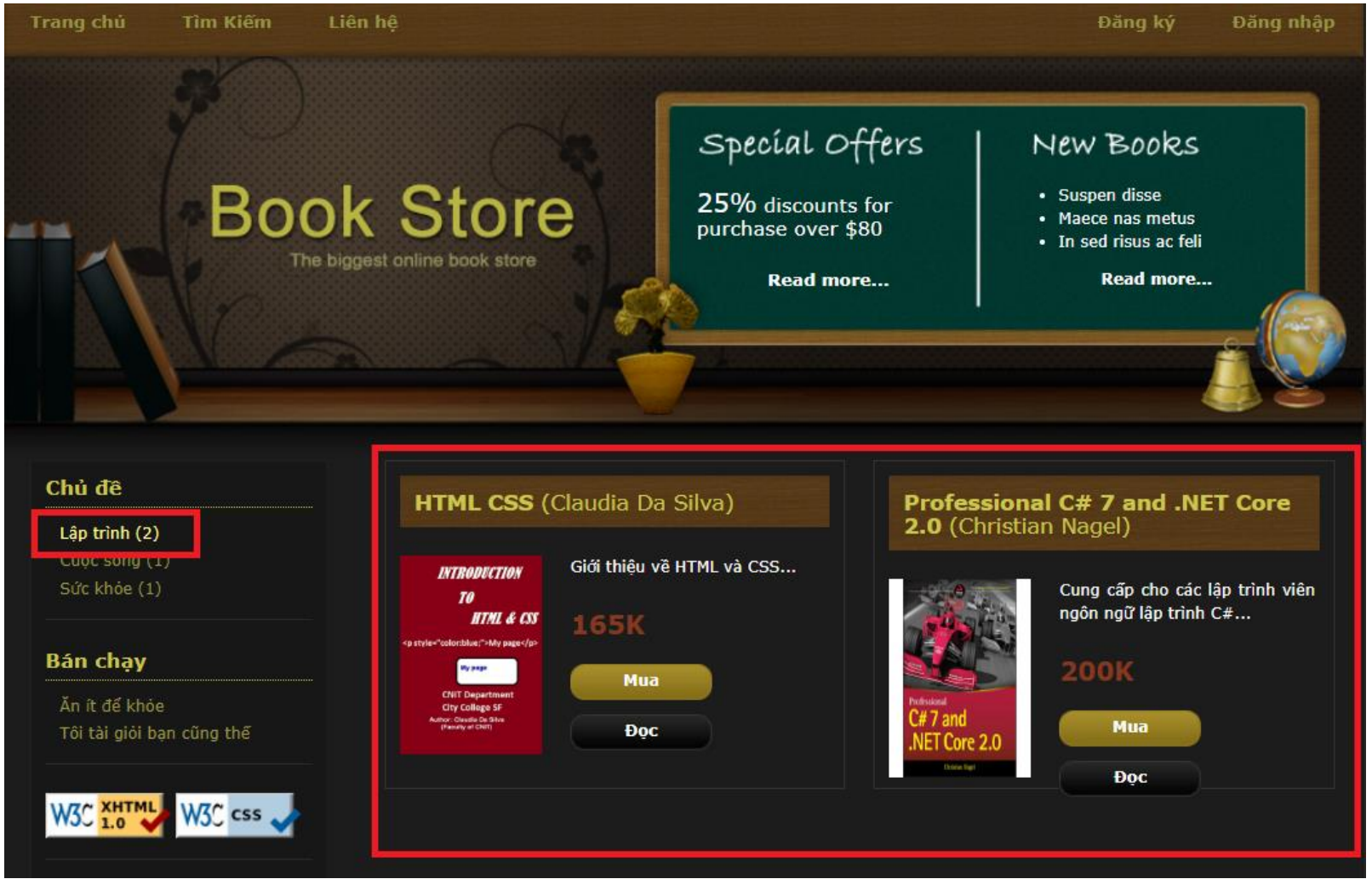
/Account/Login



ASM3:

6. chức năng Chủ đề:

VD: Lập trình thì có 2 cuốn sách, tương tự như vậy cho cuộc sống (1), sức khỏe (1)



ASM3:

7.chức năng Đọc (chi tiết) 1 cuốn sách:

8. Bán chạy (khi click vào Cũng thể hiện chi tiết như Đọc 1 cuốn)