Dự án ASP.NET Core MVC

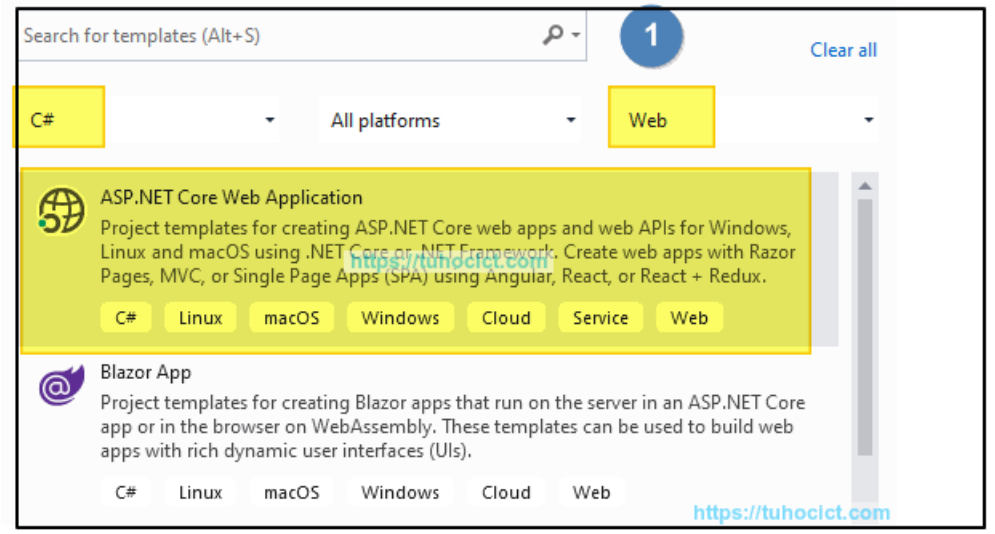
Dự án ASP.NET Core MVC có thể được tạo ra trong Visual Studio với template Web Application (Model – View – Controller), hoặc tạo ra từ giao diện dòng lệnh với template mvc. Bạn cũng có thể tạo một project rỗng và tự cài đặt những gì cần thiết mà không cần tới các template có sẵn.
Bài học này sẽ giới thiệu cách cài đặt một dự án ASP.NET Core MVC từ project rỗng để bạn nắm được các nguyên lý chung. Sau đó chúng ta sẽ làm quen với cách tạo dự án từ Visual Studio và giao diện dòng lệnh. Cuối cùng chúng ta sẽ học về cấu trúc của một dự án ASP.NET Core MVC.

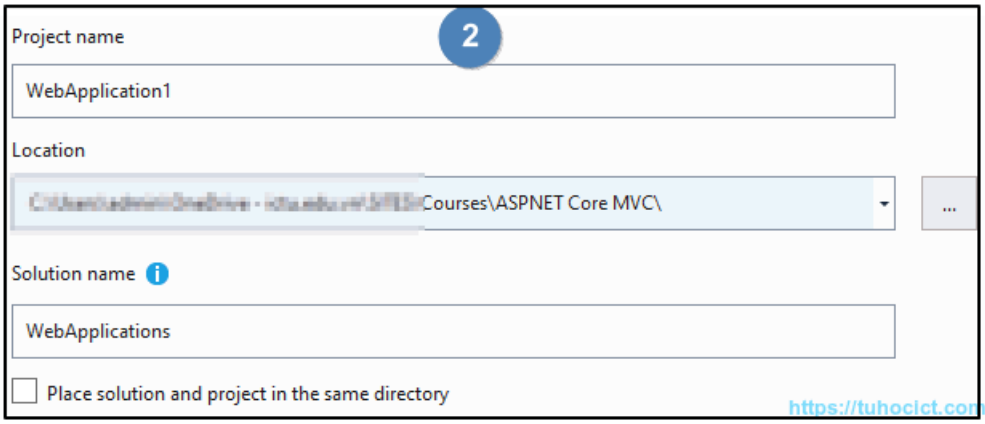
NỘI DUNG CỦA BÀI [Ẩn]

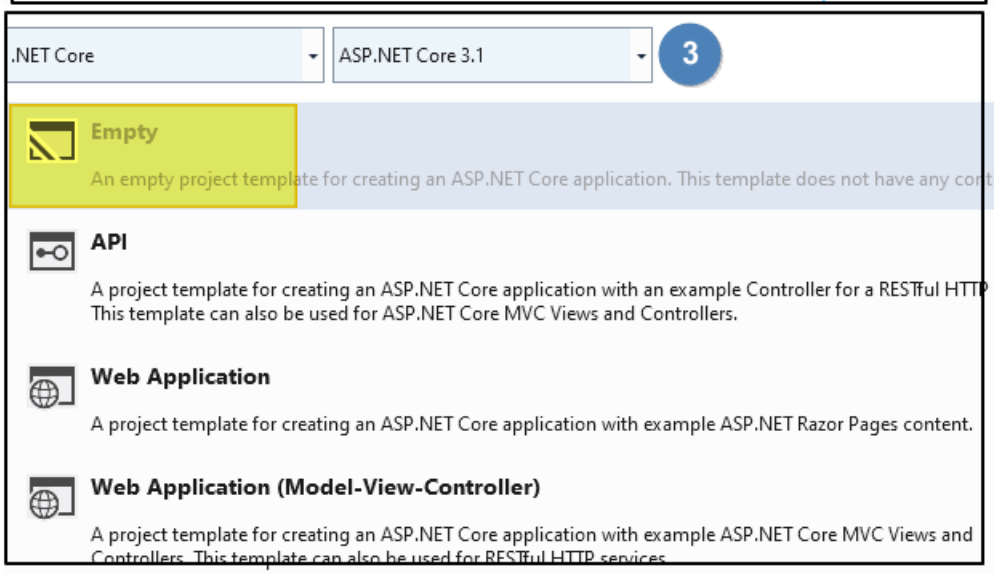
Thực hành 1 – Tự cấu hình dự án ASP.NET Core MVC

Trong phần này chúng ta sẽ tạo một dự án rỗng và cấu hình các thành phần cần thiết để tạo thành một dự án ASP.NET Core MVC cơ bản.

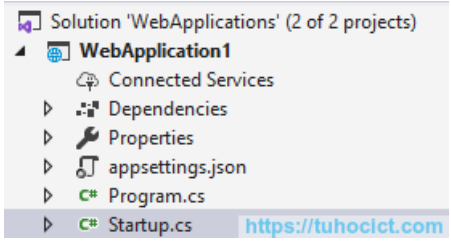
Bước 1. Tạo ra một dự án rỗng ASP.NET Core.







Bạn thu được một dự án rỗng ASP.NET Core với cấu trúc file như sau:



Bước 2. Mở file Startup.cs và viết code như sau:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace WebApplication1 {
    public class Startup {
        public void ConfigureServices(IServiceCollection services) {
            services.AddControllersWithViews();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env) {
            if (env.IsDevelopment()) {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseEndpoints(endpoints => {
                endpoints.MapControllerRoute("default", "{controller=Default}/{action=In
            });
        }
    }
}
```

Ở bước này chúng ta thực hiện hai thay đổi:

(1) Thêm lời gọi services.AddControllersWithViews(); vào phương thức ConfigureServices.

Nếu bạn đọc các tài liệu cũ hơn về ASP.NET Core MVC trước phiên bản 2.1 bạn có thể gặp phương thức AddMvc(). Từ ASP.NET Core MVC 3.0 chuyển sang sử dụng AddControllersWithViews().

(2) Điều chỉnh thân hàm lambda tham số của UseEndpoints thành endpoints.MapControllerRoute("default", "{controller=Default}/{action=Index}/{id?}");

Đây là cấu hình mẫu routing mặc định cho ứng dụng ASP.NET Core MVC. Mẫu này quy định rằng phần path của một Url hợp lệ phải có dạng controller/action/id. Trong đó:

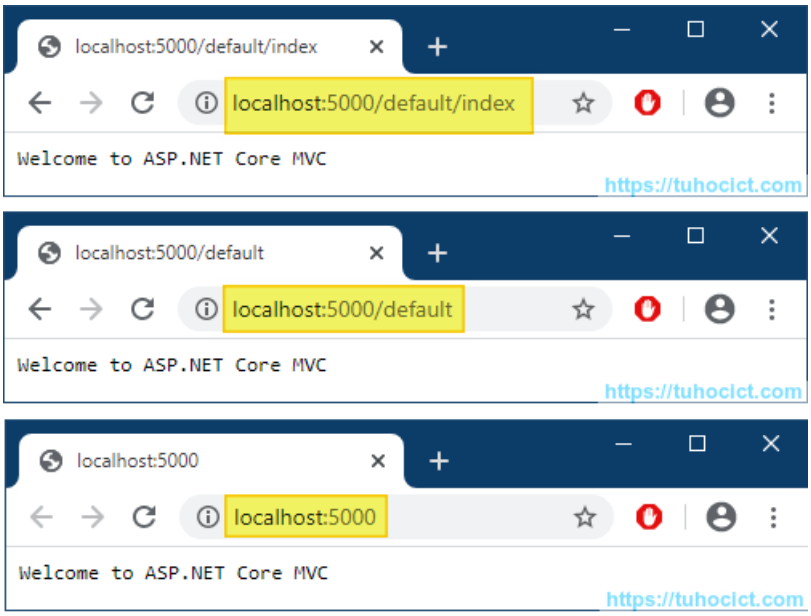
- controller là tên của lớp controller, action là tên của phương thức action trong controller, id là tham số của action.
- nếu không chỉ định controller thì mặc định sẽ gọi tới DefaultController; nếu không chỉ định action thì mặc định sẽ tìm tới phương thức Index trong DefaultController;
- id là không bắt buộc.

Bạn sẽ quay lại với nội dung routing trong một bài học riêng.

Bước 3. Tạo class DefaultController trong file cùng tên trực thuộc project và viết code như sau:

```
namespace WebApplication1 {
    public class DefaultController : Controller {
        public string Index(string id) {
            if (string.IsNullOrEmpty(id))
                return "Welcome to ASP.NET Core MVC";
            else
                return $"Hello, {id}! Welcome to ASP.NET Core MVC";
        }
    }
}
```

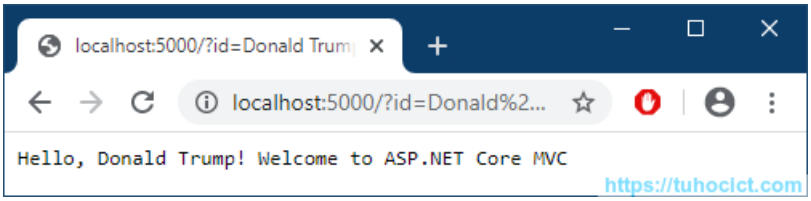
Khi chạy chương trình bạn sẽ thu được kết quả như sau (lưu ý cách viết Url):



Lý do liên quan đến mẫu routing chúng ta khai báo.

Nếu path có 2 segment, segment thứ nhất tương ứng với tên controller, segment thứ hai tương ứng với tên action trong controller đó. Như vậy /default/index tương ứng với gọi tới phương thức Index trong class DefaultController.

Nếu không nhìn thấy segment thứ hai (tương đương với tên action), giá trị "index" sẽ được sử dụng. Nếu không nhìn thấy segment đầu, giá trị "default" sẽ được sử dụng. Như vậy action Index trong DefaultController tương ứng với url /default/index hoặc /. Nếu thêm chuỗi truy vấn ?id=Donald Trump vào Url sẽ thu được kết quả như sau:



Đến đây bạn đã xây dựng được một ứng dụng ASP.NET Core MVC đơn giản nhất!

Trong phần này bạn đã cấu hình cho dự án ASP.NET Core hoạt động theo mô hình kiến trúc MVC. Bạn cũng đã xây dựng lớp controller đầu tiên với một action Index.

Thực hành 2 – Xây dựng View cho ASP.NET Core MVC

Trong phần thực hành 1 bạn đã xây dựng lớp Controller nhưng chưa thấy View và model đâu cả. Trong phần thực hành này chúng ta tiếp tục bổ sung View và sử dụng Model.

Bước 1. Xây dựng action List trong DefaultController như sau:

```
1.  public IActionResult List() {
2.      var model = new List<(string, string)> {
```

```
            ("Donald", "Trump"),
            ("Barack", "Obama"),
            ("George W.", "Bush")
        };

        var view = View("/DefaultList.cshtml");
        view.ViewData.Model = model;

        return view;
    }
```

Bước 2. Tạo file DefaultList.cshtml trực thuộc dự án.

Bạn có thể chọn tên bất kỳ cho file này. Ở đây chúng ta đặt tên DefaultList để dễ hình dung rằng đây là một view tương ứng với action List của DefaultController.

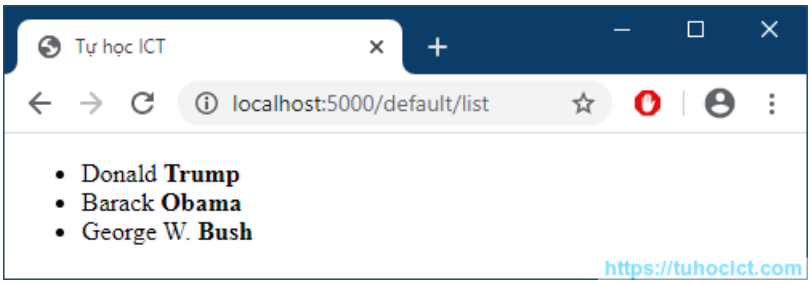
Bước 3. Viết code cho DefaultList.cshtml như sau:

```
@model List<(string firstName, string lastName)>

<!doctype html>

<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Tự học ICT</title>
</head>

<body>
    <div>
        <ul>
            @foreach (var i in Model) {
                <li>@i.firstName <b>@i.lastName</b></li>
            }
        </ul>
    </div>
</body>
</html>
```

Nếu bạn đã học qua Razor Pages thì hẳn sẽ rất quen thuộc với đoạn code Razor trên. Thực tế, view trong ASP.NET Core MVC chỉ là một file Razor thông thường. Directive @model có tác dụng báo cho Intellisense biết kiểu của model để hỗ trợ nhắc code.

Lưu ý: so với trang nội dung trong Razor Pages, view trong MVC không có directive @page (liên quan đến mẫu routing tới trang). Do vậy bạn chỉ có thể sử dụng view kết hợp với controller action chứ không thể trực tiếp truy xuất view từ trình duyệt.

Chạy thử với url /default/list, bạn thu được kết quả như sau:



Qua phần thực hành này bạn để ý mấy vấn đề sau:

(1) Model thực tế là bất kỳ dữ liệu nào cần hiển thị. Như trong ví dụ trên, model là một danh sách các cặp giá trị (string, string).

(2) View là một file Razor với directive @model có tác dụng chỉ định kiểu cụ thể của model mà view đang cần hiển thị.

(3) File view được chỉ định qua lời gọi phương thức `var view = View("/DefaultList.cshtml");` Lối viết này báo rằng cần tìm file DefaultList.cshtml trong thư mục dự án.

(4) Giá trị của model được action truyền sang cho view thông qua object của ViewData: `view.ViewData.Model = model;`

Bạn có thể viết gộp (3) và (4) thành một lệnh duy nhất `var view = View("/DefaultList.cshtml", model);`

Một số quy ước trong dự án ASP.NET Core MVC

Trong hai phần thực hành trên chúng ta tạo controller và view trực tiếp trong thư mục dự án. Nếu bạn chỉ có một vài file thì điều này không gây ra vấn đề gì.

Nếu số lượng file tăng lên, bạn cần tổ chức các file và thư mục trong dự án cho phù hợp.

Ngoài ra, ASP.NET Core MVC đưa ra một số yêu cầu về tổ chức file để đơn giản hóa việc cấu hình. Cấu hình dựa trên quy ước (configuration over convention) là một trong những cách cấu hình được nhiều framework lựa chọn để đơn giản hóa việc lập trình.

Đối với ASP.NET Core MVC cũng có nhiều quy ước cho cấu hình như vậy. Chúng ta sẽ lần lượt gặp trong những bài học cụ thể.

Trước mắt chúng ta làm quen với một số quy ước đơn giản nhất.

(1) Các lớp controller nên đặt trong thư mục Controllers trực thuộc dự án

Đây chỉ là một quy ước về tổ chức code.

Thực ra không có quy định nào bắt buộc về nơi đặt các lớp controller.

(2) Tên lớp controller cần có hậu tố "Controller"

Đây có thể xem vừa là một quy ước, vừa là một yêu cầu bắt buộc.

ASP.NET Core MVC yêu cầu các lớp controller phải thỏa mãn một trong hai yêu cầu: (1) Kế thừa từ lớp cha Controller hoặc ControllerBase; (2) Nếu không phải lớp dẫn xuất của Controller (hoặc ControllerBase) thì tên gọi phải có hậu tố Controller.

Bạn sẽ biết lý do của các yêu cầu này trong bài học riêng về controller trong ASP.NET Core MVC.

(3) Các file Razor view nên đặt trong thư mục Views trực thuộc dự án

Quy ước này liên quan đến cách ASP.NET Core MVC tìm file view cho action tương ứng.

Mặc định, ASP.NET Core MVC xác định file view như sau
/Views/{Controller}/{Action}.cshtml

Ví dụ, với action List trong controller DefaultController, ASP.NET Core MVC cho rằng file Razor view tương ứng sẽ là /Views/Default/List.cshtml. Nếu không chỉ định bất kỳ thông tin gì khác về view (như tên hay đường dẫn), ASP.NET Core MVC sẽ tự tìm file tương ứng.

Chúng ta sẽ quay lại với vấn đề này trong bài học riêng về view trong ASP.NET Core MVC.

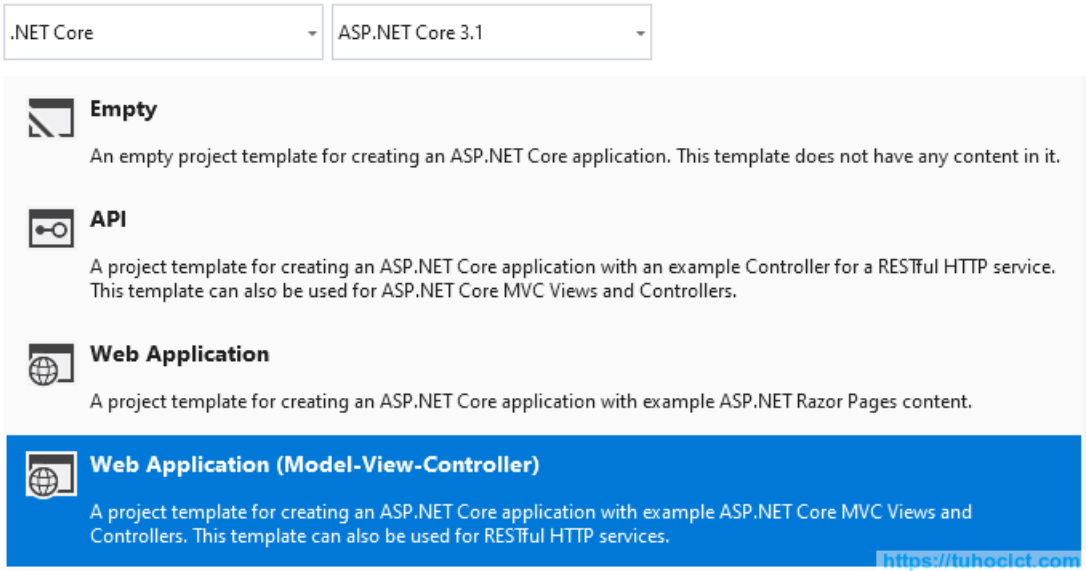
Template cho dự án ASP.NET Core MVC

Trong các phần trên chúng ta tự cấu hình dự án ASP.NET Core MVC từ một dự án rỗng. Mục đích là giúp bạn hiểu rõ hơn về một dự án ASP.NET Core MVC.

Để giúp bạn có một dự án với đầy đủ các thành phần cơ bản, ASP.NET Core cung cấp sẵn mẫu dự án MVC. Bạn có thể sử dụng Visual Studio hoặc giao diện dòng lệnh để tạo dự án theo mẫu.

(1) Sử dụng Visual Studio

Nếu sử dụng Visual Studio, mẫu dự án cho MVC được đặt tên là Web Application (Model-View-Controller).



Lưu ý, template Web Application là template cho ứng dụng Razor Pages mà bạn đã học trước đây.

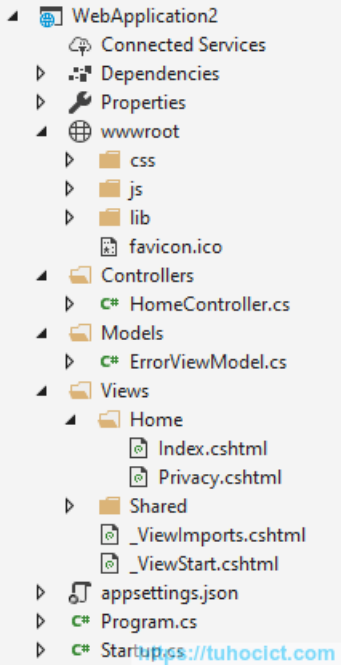
(2) Sử dụng giao diện dòng lệnh

Khi sử dụng giao diện dòng lệnh, tên mẫu tương ứng cho dự án ASP.NET Core MVC là mvc.

Bạn sử dụng cùng với lệnh dotnet new như sau:

```
dotnet new mvc -o WebApplication3
```

Khi sử dụng template này bạn thu được một dự án với cấu trúc như sau:



Dự án này tạo sẵn cho bạn controller đầu tiên HomeController, hai view tương ứng với hai action Index.cshtml và Privacy.cshtml.

Ngoài ra, mẫu dự án này cũng cài đặt sẵn các thư viện css/js (Bootstrap, jQuery), layout, view imports và view start. Các thành phần này giống như trong dự án Razor Pages.

Kết luận

Trong bài học này chúng ta bước đầu làm quen với dự án ASP.NET Core MVC, bao gồm cấu hình sử dụng MVC từ dự án rỗng và sử dụng template sẵn có cho dự án MVC.

> + Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
> + Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.
> + Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.
> Cảm ơn bạn!