Routing trong Razor Pages – ánh xạ URL sang page

Routing là một khái niệm rất quan trọng trong các web framework nói chung. Routing có nhiệm vụ ánh xạ (map) truy vấn của người dùng (dưới dạng URL – Uniform Resource Locator) sang trang web/phương thức trên server.

Trong Razor Pages, routing là cơ chế ánh xạ URL sang trang cshtml. Đồng thời routing trong Razor Pages còn có thêm những nhiệm vụ quan trọng khác nữa.

Bài học này sẽ giới thiệu chi tiết các vấn đề về routing trong Razor Pages.

NỘI DUNG CỦA BÀI [Ẩn]

Cấu trúc URL

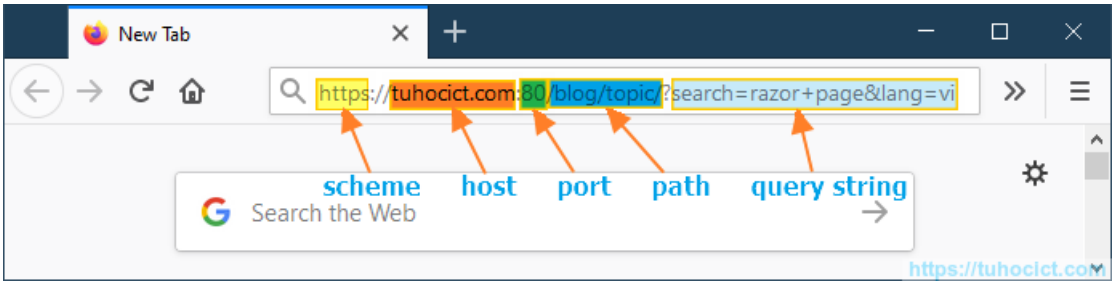
Bạn nào làm việc trên Internet hẳn cũng đã biết URL là gì. Tuy nhiên, chúng ta phải nhắc lại một số thuật ngữ liên quan để sử dụng thống nhất trong bài giảng.

URL là viết tắt của **Uniform Resource Locator**. Hiểu một cách nôm na nhất, mỗi URL là ***địa chỉ của một tài nguyên*** trên Internet. Còn tài nguyên là gì, chút nữa chúng ta sẽ trao đổi tiếp.

> Đôi khi bạn còn gặp thuật ngữ ***URI*** (Uniform Resource ***Identifier***) với ý nghĩa rất gần với URL. Nhiều người còn sử dụng lẫn lộn URI với URL. Chú ý rằng, URL là URI nhưng URI chưa chắc đã là URL. Nói cách khác, URL là một phần của URI. URI tương ứng với bất kỳ thứ gì ***có thể đặt tên*** trên Internet, bao gồm cả người, cả tài nguyên. Trong khi đó URL chỉ liên quan đến tài nguyên.

Cấu trúc tổng quát nhất của một URL như sau:

```
scheme://host:port/path?query-string#fragment-id
```



Scheme thường gặp là http, https hoặc một số giao thức khác (như ftp).

Trong cấu trúc trên, phần **port** ít gặp hơn. Port mặc định mà web server chiếm là 80. Các trình duyệt đều tự động thêm port 80 vào khi cần. Nếu sử dụng port khác 80, bạn phải tự mình chỉ định. Ví dụ một số site quản lý có thể dùng port 8080 thay cho 80.

Phần **path** có thể hiểu tương tự như đường dẫn tới một file (trên ổ cứng). Các web framework thường sẽ định nghĩa khuôn mẫu cho phần path trong một cơ chế gọi là routing (bạn sẽ học dưới đây).

Query string chứa các cặp khóa=giá trị. Các cặp phân tách bởi ký tự &. Query string phân tách với path bởi ký tự ?.

Fragment (hoặc anchor) thường chỉ gặp trong các tài nguyên nội dung (như blog post) dùng để đánh dấu các tiêu đề.

Trong cấu trúc trên, query string và fragment là không bắt buộc.

Routing là gì?

Khi bạn cần tải một nội dung gì đó, bạn nhập **URL** tương ứng vào thanh địa chỉ trình duyệt. URL thường được hiểu là "địa chỉ" của một *tài nguyên* nào nó trên Internet.

"Tài nguyên" thường được hiểu là các file trên mạng. Do đó, thông thường mỗi URL tương ứng với một file trên server, ví dụ file ảnh, file video, file audio.

Tuy nhiên, bạn cũng thường xuyên gặp những URL không có vẻ gì liên quan đến file. Lấy ví dụ về URL trỏ tới một trang bài học trên site Tự học ICT "https://tuhocict.com/event-su-kien-trong-c/". Tài nguyên này vốn là HTML do code PHP sinh ra.

Tức là, loại tài nguyên mà URL trỏ tới cũng có thể là việc chạy một đoạn code nào đó của chương trình web trên server.

Như vậy chương trình trên server phải có cách để xác định tài nguyên tương ứng với mỗi URL nó nhận được từ trình duyệt.

Giờ hãy nhìn nhận vấn đề theo một hướng khác.

Hãy hình dung khi viết một chương trình với giao diện dòng lệnh, bạn thường phải định nghĩa một tập lệnh cùng với các tham số cho từng lệnh. Người dùng chỉ có thể nhập các

lệnh hợp lệ chứ không thể nhập tùy ý.

Điều này cũng đúng với ứng dụng web. Khi viết một ứng dụng bạn cũng định nghĩa một tập "lệnh" hợp lệ dưới dạng tập hợp các Url mà ứng dụng có thể hiểu. Chỉ khi người dùng nhập Url hợp lệ, ứng dụng mới thực thi.

Như vậy, một ứng dụng web phải có khả năng xác định xem một Url nó nhận được có hợp lệ hay không. Nếu Url là hợp lệ thì nó tương ứng với tài nguyên nào của chương trình. Với "tài nguyên" thường được hiểu là file hoặc thực thi code.

Tóm lại, một ứng dụng web cần: (1) định nghĩa tập URL hợp lệ; (2) kiểm tra tính hợp lệ của một URL nó nhận được; (3) xác định xem URL (hợp lệ) tương ứng với tài nguyên nào.

Cơ chế thực hiện nhóm nhiệm vụ đó trong các ứng dụng web thường được gọi là **routing**.

Để thực hiện routing, các framework thường định nghĩa các **route template** – khuôn mẫu để tạo ra tập hợp các Url hợp lệ cho mỗi tài nguyên, đồng thời cũng dùng để kiểm tra tính hợp lệ của Url nhận được từ client. Mỗi cặp Url hợp lệ và tài nguyên tương ứng của nó tạo thành một **route**.

Như thế, khi hiểu route template, bạn có thể xác định được URL hợp lệ cho tài nguyên, đồng thời từ URL có thể xác định được tài nguyên tương ứng (nếu có) trên server.

> Lưu ý, đừng nhầm lẫn với khái niệm routing (định tuyến) trong mạng máy tính.

Routing mặc định trong Razor Pages

Cơ chế routing mặc định Razor Pages sử dụng khuôn mẫu là đường dẫn tương đối từ **thư mục Pages** đến các file cshtml (bỏ qua phần mở rộng cshtml).

Thư mục Pages được gọi là **thư mục gốc** (**root folder**).

> Bạn có thể cấu hình để Razor Pages nhận một thư mục khác là root folder.

Lấy ví dụ, nếu trong thư mục Pages có 3 file Index.cshtml, Privacy.cshtml, Error.cshtml, Razor Pages sẽ tạo ra một tập hợp 4 **route** sau:

1. `/` <-> file Pages\Index.cshtml
2. `/index` <-> file Pages\Index.cshtml
3. `/error` <-> file Pages\Error.cshtml
4. `/privacy` <-> file Pages\Privacy.cshtml

Tức là phần path của Url sẽ tương ứng với đường dẫn tới file cshtml (tính từ thư mục Pages). Nói chung, route template mặc định của mỗi trang đều rất đơn giản.

> Để tiện lợi trong trình bày, từ giờ về sau chúng ta sẽ chỉ viết từ phần path của URL chứ không viết cả phần scheme và host (như trong môi trường phát triển ứng dụng thì host đều là localhost).

Bạn có thể để ý thấy, riêng đối với Index.cshtml có hai route. Index.cshtml được gọi là **trang mặc định** của folder. Riêng trang Pages\Index.cshtml còn được gọi là **landing page** của site – trang được mở nếu không chỉ định trang nào trong URL.

Giả sử bạn tạo mới thư mục Pages\Admin. Trong thư mục Admin bạn tạo các file Index.cshtml, Login.cshtml.

Khi này Razor Pages sẽ tạo thêm các route sau:

1. `/admin/` <-> Pages\Admin\Index.cshtml
2. `/admin/index` <-> Pages\Admin\Index.cshtml
3. `/admin/login` <-> Pages\Admin\Login.cshtml

Người dùng có quyền nhập URL bất kỳ vào trình duyệt. Khi có một URL tới từ trình duyệt, Razor Pages so khớp URL đó với tập hợp route. Nếu tìm thấy route phù hợp, file cshtml tương ứng sẽ được Razor engine xử lý để tạo ra HTML tương ứng.

Giả sử bạn nhập Url `/member/login`, khuôn mẫu mặc định sẽ gợi ý rằng Url này có thể tương ứng với file `Pages/Member/Login.cshtml` hoặc `Pages/Member/Login/Index.cshtml`. Nếu không tìm thấy một trong file này, Razor sẽ báo lỗi.

Với cách hoạt động như vậy, nếu bạn tạo file Admin.cshtml trực tiếp trong thư mục Page và gọi tới url `/admin`, bạn sẽ gặp lỗi. Lý do là vì Url `/admin` thì tương ứng với cả file Admin.cshtml và Admin/Index.cshtml. Khi này Razor Pages không biết được file nào để trả lại truy vấn /admin.

Từ đây bạn có thể để ý, mỗi page cshtml có thể có nhiều URL tương ứng. Nhưng mỗi url chỉ có thể tương ứng 1 page.

> Các page mà tên bắt đầu bằng dấu gạch chân (như _Layout, _ViewStart, _ViewImports) sẽ không có route tương ứng. Nói cách khác, không có Url nào tương ứng với các page này và trình duyệt không thể truy xuất được chúng. Đây là các page có vai trò riêng đặc biệt trong Razor Pages.
>
> Sau này bạn sẽ gặp một số loại page khác cũng không có route tương ứng như partial page hoặc view component. Các loại page đặc biệt này không chứa directive @page ở đầu file.

Razor Pages cho phép chúng ta thay đổi các cấu hình liên quan đến routing, bao gồm thay đổi root folder (từ Pages sang một thư mục khác), thay đổi homepage – trang chủ mặc định của site, ghi đè route mặc định.

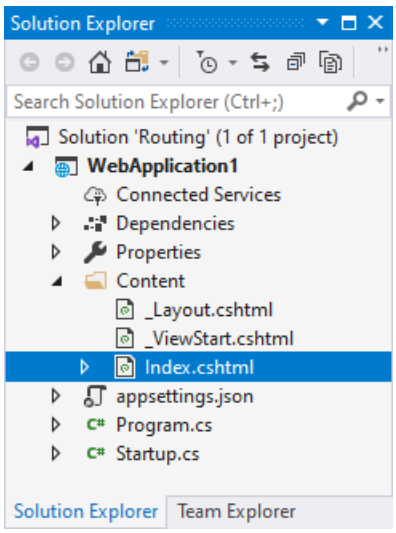
Khi học cách truyền dữ liệu qua URL bạn sẽ gặp lại cách ghi đè cơ chế routing mặc định sử dụng route template.

Thay root folder

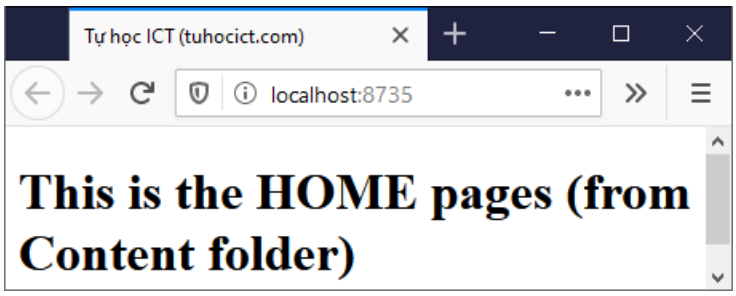
Mặc định Razor Pages sử dụng Pages làm root folder, nghĩa là URL được ánh xạ sang các page nằm trong thư mục Pages. Bạn có thể cấu hình, ví dụ, để **Content** trở thành root folder trong phương thức ConfigureServices (lớp Startup) như sau:

```
1.  public void ConfigureServices(IServiceCollection services) {
2.      services
3.      .AddRazorPages()
4.      .AddRazorPagesOptions(options => {
5.          options.RootDirectory = "/Content";
6.      });
7.  }
```

Khi này nếu thêm page Index.cshtml vào thư mục Content



sẽ thu được kết quả:



Thay đổi homepage

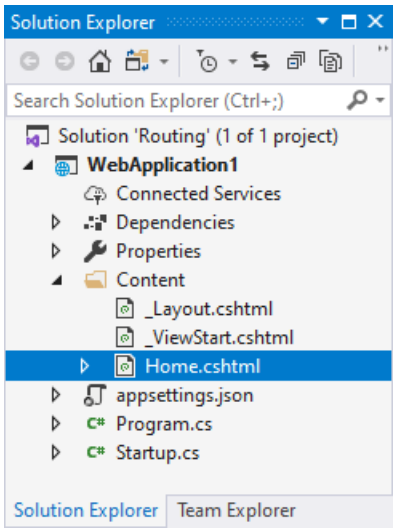
Homepage (trang chủ) là page tương ứng với URL /, là trang mặc định nếu người dùng không nhập gì thêm vào sau hostname trong URL.

Razor page sẽ ánh xạ sang trang chủ mặc định là <root folder>\Index.cshtml.

Bạn có thể thay đổi trong ConfigureServices như sau:

```
1.  public void ConfigureServices(IServiceCollection services) {
2.      services
3.      .AddRazorPages()
4.      .AddRazorPagesOptions(options => {
5.          options.RootDirectory = "/Content";
6.          options.Conventions.AddPageRoute("/Home", "/");
7.      });
8.  }
```

Lệnh `options.Conventions.AddPageRoute("/Home", "/");` sẽ cấu hình cho url `/` tương ứng với file `<root folder>\Home.cshtml`.



Lưu ý rằng, nếu trong root folder có file Index.cshtml thì cấu hình trên sẽ gây lỗi vì Razor Pages luôn coi Index làm page mặc định của folder. Nếu Index nằm trong <root folder> thì nó luôn làm homepage của site. Khi này một url sẽ tương ứng với 2 page, và điều này sẽ gây lỗi.

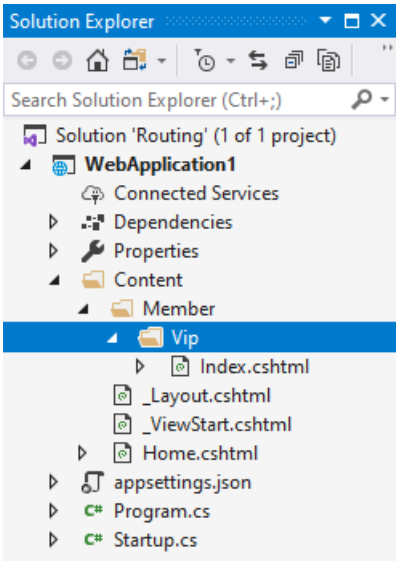
> Lệnh `AddPageRoute` còn nhiều tác dụng nữa mà bạn sẽ lần lượt làm quen về sau.

Ghi đè route mặc định

Đôi khi bạn muốn thay đổi route mặc định (ánh xạ url sang cấu trúc file). Đây là tình huống khi bạn không muốn Url ánh xạ cấu trúc thư mục – file, hoặc bạn cần cung cấp dữ liệu cho phương thức xử lý thông qua Url.

Phần này sẽ xem xét tình huống thứ nhất. Tình huống cung cấp dữ liệu qua route data sẽ xem xét trong bài học về xử lý truy vấn get.

Giả sử trong <root folder> bạn tạo thư mục con Member, trong Member tạo tiếp thư mục Vip, trong Vip tạo file Index.cshtml:



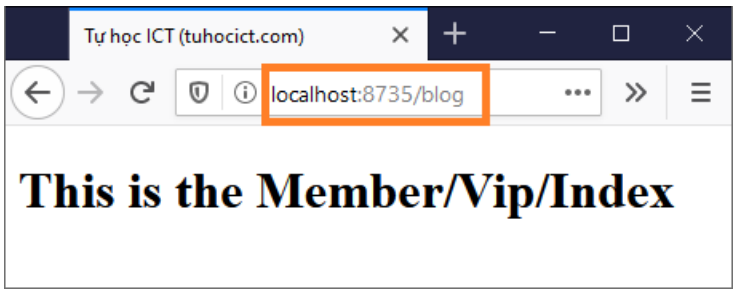
Khi này URL mặc định cho Index.cshtml là /member/vip hoặc /member/vip/index.

Giờ bạn muốn ghi đè route mặc định này, ví dụ, thay url tương ứng với file thành /blog. Bạn có hai cách thực hiện:

Option 1: Mở file /Member/Vip/Index.cshtml và thay đổi @page directive như sau:

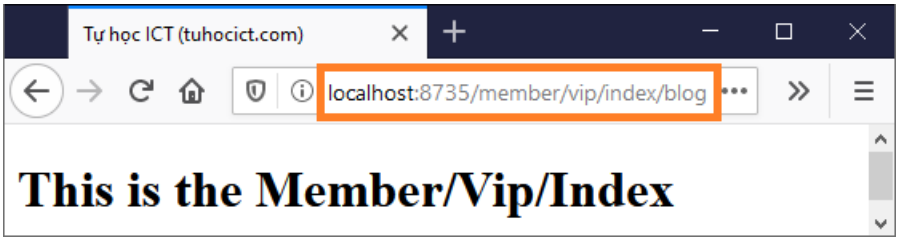
```
1.  @page "/blog"
2.  @model WebApplication1.Content.Member.Vip.IndexModel
3.  @{
4.  }
```

Lưu ý trong directive `@page "/blog"` phải có ký tự "/" hoặc "~/" để báo cho Razor Pages biết rằng bạn đang muốn ghi đè url trong route mặc định. Trong trường hợp này Url trong route tới page sẽ là **`/blog`**, thay vì url mặc định **`/member/vip/index`** hoặc **`/member/vip/`**.



> Hiểu một cách đơn giản nhất, @page không có tham số gì đứng sau chỉ định rằng sẽ sử dụng route mặc định (theo cấu trúc thư mục) cho page. Nếu có thêm tham số sau @page sẽ ghi đè route mặc định theo mẫu quy định bởi tham số đó.
> Bạn sẽ học chi tiết hơn về cách cấu hình routing sử dụng @page directive trong bài học về route template và truyền dữ liệu qua Url.

Chú ý: Nếu thiếu ký tự **/**"Razor Pages sẽ hiểu rằng đây là một *đoạn* (segment) của một url chứ không phải là url hoàn chỉnh. Cùng ví dụ trên, nếu bạn viết **`@page "blog"`** thì url tương ứng với page sẽ là **`/member/vip/index/blog`** hoặc **`/member/vip/blog`**. Nghĩa là **`blog`** khi đó trở thành segment cuối cùng của url mặc định tương ứng:



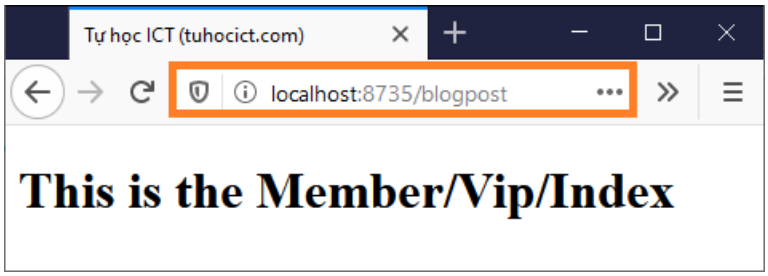
Option 2: Sử dụng phương thức `AddPageRoute` trong phương thức ConfigureServices lớp Startup:

```
1.  public void ConfigureServices(IServiceCollection services) {
2.      services
3.      .AddRazorPages()
4.      .AddRazorPagesOptions(options => {
5.          options.RootDirectory = "/Content";
6.          options.Conventions.AddPageRoute("/Home", "/");
7.          options.Conventions.AddPageRoute("/Member/Vip/Index", "/blogpost");
8.      });
9.  }
```

Phương thức AddPageRoute nhận hai tham số: tham số thứ nhất là đường dẫn tương đối từ root folder tới page (bỏ đuôi cshtml); tham số thứ hai là url cần ánh xạ tới page này.

Trong lệnh trên bạn cấu hình một route mới ánh xạ url **`/blogpost`** sang page `/Member/Vip/Index`.

Bằng cách này bạn có thể định nghĩa nhiều route tới cùng một page.



Kết luận

Trong bài học này bạn đã làm quen với cơ chế routing của Razor Pages. Do là một framework dựa trên page, cơ chế routing của Razor Pages tương đối dễ hiểu. Đồng thời, Razor Pages cũng cho phép thực hiện các thay đổi với mẫu routing mặc định.

Bài học này mới chỉ cung cấp phần đầu của vấn đề routing. Chúng ta sẽ quay trở lại vấn đề routing khi xem xét cách truyền dữ liệu qua Url.