Routing trong ASP.NET Core MVC

Routing trong ASP.NET Core là quá trình ánh xạ (mapping) *truy vấn HTTP* với một đối tượng xử lý truy vấn đó (*handler*). Trong MVC, handler chính là phương thức action của controller. Tức là, routing trong ASP.NET Core MVC là cơ chế ánh xạ giữa truy vấn HTTP và controller action.

Hiểu một cách đơn giản, khi truy vấn tới, cơ chế routing xác định xem có action nào tương ứng với nó hay không để thực thi. Khi xây dựng một controller và các action, routing cũng xác định một tập hợp các Url hợp lệ tương ứng cho controller và action đó.

> Routing thực ra không phải là một phần của kiến trúc MVC. Routing là một phần trong khối Asp.net Core platform. Nếu còn nhớ routing trong Razor Pages bạn hẳn sẽ thấy định nghĩa routing trong MVC có điểm khác biệt.

Với khái niệm như trên, routing trong ASP.NET Core MVC bắt buộc phải chỉ định rõ mỗi URL tương ứng với action nào. ASP.NET Core MVC sử dụng hai cách thức khác nhau để chỉ định: (1) sử dụng một template chung cho nhiều controller action, (2) chỉ định Url cụ thể cho từng controller action riêng biệt.

Phương pháp 1 được gọi là *conventional routing*. Phương pháp 2 được gọi là *attribute routing*. Bài học này sẽ giới thiệu cả hai loại hình routing trên.

NỘI DUNG CỦA BÀI [Ẩn]

Routing mặc định của dự án MVC

Trước mắt hãy cùng xem xét loại routing mặc định mà ASP.NET Core MVC sử dụng cho một project mới.

Khi cấu hình sử dụng Mvc Middleware, bạn bắt buộc phải chỉ định ít nhất một route template trong phương thức `Configure` của Startup.cs sử dụng phương thức `UseEndpoints` như sau:

```
1.  app.UseEndpoints(endpoints => {
2.      endpoints.MapControllerRoute(
3.          name: "default",
4.          pattern: "{controller=Home}/{action=Index}/{id?}");
5.  });
```

> Trong phương thức này bạn có thể định nghĩa thêm các template của mình. Chúng ta sẽ nói về vấn đề này ở phần sau của bài học.

`"{controller=Home}/{action=Index}/{id?}"` là **route template** mặc định ASP.NET Core sử dụng cho các dự án MVC. Đây là cơ chế **conventional routing**, áp dụng chung cho toàn bộ controller và action của ứng dụng.

Route template là chuỗi ký tự quy định hình thức của một Url hợp lệ.

Template mặc định của MVC quy định rằng phần path của một Url hợp lệ phải có dạng `{controller}/{action}` hoặc `{controller}/{action}/{id}`. Trong đó:

- `{controller}`, `{action}` và `{id}` được gọi là các *placeholder* hay *route parameter*. Placeholder {controller} và {action} được cơ chế routing của MVC sử dụng để xác định xem url này tương ứng với controller và action nào.
- Ký tự / phân chia chuỗi template thành các *segment*.
- Home và Index được gọi là giá trị mặc định tương ứng của controller và action.
- {id?} là không bắt buộc.

Đây là một template rất tổng quát.

Khi sử dụng template này, ASP.NET Core sẽ tách URL của truy vấn thành các segment. Hai segment đầu tiên cho phép rút ra thông tin về controller và action. Dựa trên thông tin này ASP.NET Core sẽ tìm kiếm controller và action phù hợp để thực thi.

Nếu có segment thứ 3, giá trị của segment này sẽ đưa vào một biến có tên id để truyền cho action. Segment thứ 3 không bắt buộc.

Lấy ví dụ truy vấn tới là `Book/Delete/3`. Cơ chế routing sẽ so khớp nó với mẫu {controller}/{action}/{id}. Quá trình so khớp này cho thấy controller là Book, action là Delete và id là 3. Từ thông tin về controlller và action, ASP.NET Core MVC sẽ kích hoạt đúng phương thức action phù hợp. Khi kích hoạt action này nó cũng đồng thời truyền giá trị id = 3 nếu action yêu cầu.

Khi bạn tạo một controller mới với các action trong đó, các action này tự động có URL phù hợp. Bạn không cần tự mình cấu hình router riêng. Khi nhìn thấy một Url, bạn có thể dự đoán được ngay action trong controller nào sẽ được gọi khi phát đi truy vấn.

Routing trong ASP.NET Core MVC

ASP.NET Core MVC cung cấp hai cơ chế routing: **conventional routing** và **attribute routing**.

Conventional routing là loại routing được định nghĩa chung cho ***nhiều*** controller action dựa trên một số quy ước xác định. Bất kỳ controller và action nào tuân theo quy ước đều tự

động tham gia vào routing tương ứng.

Trong cơ chế routing này, bạn xuất phát bằng cách định nghĩa một hoặc một vài route template trong phương thức `Configure` của lớp `Startup`.

Khi một truy vấn đến, ASP.NET Core MVC lần lượt so khớp Url của truy vấn với các template để xem Url đó phù hợp với template nào. Việc so sánh được thực hiện theo thứ tự khai báo của template: template nào khai báo trước sẽ so sánh trước. Nếu tìm thấy template phù hợp thì quá trình so khớp chấm dứt, kể cả khi vẫn còn template chưa so.

Conventional routing đưa ra khuôn mẫu Url và bạn xây dựng controller/action tuân thủ khuôn mẫu.

ASP.NET Core MVC bắt buộc phải xác định được tên controlller và action từ quá trình so khớp Url và route template.

Đây là cơ chế routing mặc định của ứng dụng ASP.NET Core MVC.

Conventional routing đơn giản, tiện lợi, dễ theo dõi và quản lý. Controller và action phải tuân theo quy ước thì mới áp dụng được cơ chế routing này.

Attribute routing là loại routing được định nghĩa riêng rẽ cho từng action bằng cách đặt attribute `[Route]` trước mỗi action.

Attribute routing có ý tưởng ngược lại so với conventional routing. Trong attribute routing bạn xuất phát từ action và định nghĩa Url tương ứng cho nó.

Khi bắt đầu, ứng dụng duyệt qua tất cả các action (trong tất cả các controller) để xem action nào được đánh dấu `[Route]` và xây dựng một từ điển url (khóa) => action (giá trị). Khi truy vấn tới, ASP.NET Core MVC trích url và tìm khóa (url) tương ứng trong từ điển. Nếu tìm thấy, action tương ứng sẽ được thực thi.

Loại routing này linh hoạt hơn nhiều so với conventional routing song lại khó theo dõi và quản lý hơn. Khi sử dụng attribute routing, bạn cần tự mình quản lý một danh sách các routing riêng rẽ.

Phương pháp routing này hữu dụng hơn khi xây dựng Web API.

Mvc Middleware cho phép cung cấp và sử dụng đồng thời cả hai phương pháp routing và nhiều cấu trúc ánh xạ khác nhau.

Mỗi cấu trúc ánh xạ được gọi là một *router* hoặc *route template*.

Route template là những chuỗi ký tự định nghĩa cấu trúc cho các URL hợp lệ của ứng dụng.

Dù sử dụng phương pháp nào, cuối cùng bạn cũng đều phải định nghĩa route template.

Chúng ta sẽ nói chi tiết về cấu trúc của route template ở phần sau của bài học.

Conventional routing trong ASP.NET Core MVC

Conventional routing được định nghĩa trong phương thức Configure của Startup.cs:

```
12.  public void Configure(IApplicationBuilder app, IWebHostEnvironment env) {
13.      if (env.IsDevelopment()) {
14.          app.UseDeveloperExceptionPage();
15.      }
16.
17.      app.UseRouting();
18.
19.      app.UseEndpoints(endpoints => {
20.          endpoints.MapControllerRoute("default", "{controller = Home}/{action = Index}/{i
21.      });
22.  }
```

Mỗi template được định nghĩa bởi một lần gọi phương thức MapControllerRoute. Phương thức này tối thiểu chứa tên template và chuỗi template.

> Tên template vừa dùng để phân biệt các template, vừa được dùng trong html helper hoặc tag helper sinh URL.

Mặc định ASP.NET Core MVC định nghĩa sẵn cho bạn "`default`" template. Chúng ta đã giải thích ý nghĩa của default template ở phần trên.

Bạn có thể định nghĩa bao nhiêu template tùy thích.

Quy trình hoạt động khi sử dụng conventional routing như sau:

1. Khi truy vấn tới, ASP.NET Core MVC sẽ trích ra url từ truy vấn.
2. Url sẽ được so khớp lần lượt với các template để xem nó phù hợp với template nào.
3. Nếu có nhiều route template, ASP.NET Core MVC sẽ so khớp lần lượt với từng template theo thứ tự khai báo cho đến khi tìm được route phù hợp.
4. Nếu tìm thấy template phù hợp, quá trình so khớp sẽ dừng lại, dù tiếp theo vẫn còn template nữa.
5. Dựa theo cấu trúc template, tên controller và action sẽ được trích ra từ Url.
6. Thực thi action tương ứng.

Bạn nên bố trí template cụ thể hơn ở trên, template tổng quát hơn ở dưới.

Giờ hãy điều chỉnh phương thức UseEnpoints như sau:

```
1.  app.UseEndpoints(endpoints => {
2.      endpoints.MapControllerRoute("my-route-1", "hi/{fname}-{lname}", new { controller =
3.      endpoints.MapControllerRoute("my-route-2", "web/{action}/{controller}");
4.      endpoints.MapControllerRoute("default", "{controller = Home}/{action = Index}/{id?}"
5.  });
```

Bạn đã xây dựng hai route template mới. "my-route-1" ánh xạ những Url như `hi/Donald-Trump`, `hi/Vladimir-Putin` vào phương thức action `Hi` của `DefaultController`. "my-route-2" cho phép gọi đến DefaultController.Hi() qua Url `web/hi/default`.

Trong hai template mới, my-route-2 tổng quát hơn vì nó áp dụng được với nhiều controller và action. my-route-1 cụ thể hơn vì nó chỉ áp dụng duy nhất cho DefaultController.Hi(). Mức độ tổng quát của template được xác định qua số controller/action tiềm năng.

> Bạn có thể tự xây dựng controller đặt tên là HomeController và tạo phương thức action Index như sau để thử nghiệm.

```
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers {
    public class DefaultController : Controller {
        public IActionResult Hi(string fname, string lname) {
            return View((fname, lname));
        }
    }
}
```

Trong template my-route-1 có 2 segment: hi và {fname}-{lname}. Segment thứ nhất (hi) được gọi là *literal segment*. Segment thứ hai là tổ hợp của hai *route parameter* `{fname}` và `{lname}` phân tách bởi ký tự `-`.

Do từ chuỗi template không xác định được sẽ gọi controller và action nào, bạn cung cấp thêm một object thuộc kiểu vô danh, trong đó chỉ rõ controller sẽ luôn nhận giá trị Default, và action luôn nhận giá trị Hi. Object này đảm bảo rằng nếu Url tới phù hợp, action Hi của DefaultController sẽ luôn được gọi.

Cơ chế routing của ASP.NET Core so khớp Url tới xem nó trùng khớp với khuôn mẫu nào. Nếu hình thức url phù hợp với mẫu my-route-1 (/hi/xxx-yyy), chuỗi ký tự ở vị trí tương ứng fname và lname sẽ được tách ra và đặt vào trong một object dạng từ điển để truyền vào action.

Để hiểu rõ hơn cách tạo route, chúng ta phải biết về cấu trúc của một route template.

Cấu trúc route template

Cấu trúc route template là chung cho cả attribute và conventional routing.

Segment và parameter

Route template được phân chia thành **segment**. Mỗi segment có thể là:

- Một chuỗi ký tự "tĩnh" gọi là literal segment. Như trong my-route-1, hi chính là một literal segment. Literal segment yêu cầu phải hoàn toàn trùng khớp nhưng không phân biệt hoa thường. Trong template "hi/{fname}-{lname}", segment đầu tiên bắt buộc phải là hi, Hi, HI, hay hI.
- Thành phần "động" gọi là **placeholder** hoặc **parameter**. Placeholder trong template được thể hiện trong cặp dấu {}. Đây là thành phần phức tạp và tạo ra sự linh hoạt của route template. Có hai loại placeholder: (1) loại do ASP.NET Core MVC định nghĩa sẵn và bắt buộc, bao gồm **{controller}** và **{action}**; (2) loại do bạn tự đặt ra (như {lname} và {fname} bên trên).
- Tổ hợp của cả literal và route parameter. Đây chính là cách chúng ta tạo ra segment thứ hai của template "hi/{fname}-{lname}".

Bạn bắt buộc phải có cách chỉ định {controller} và {action} khi thêm route template. Chỉ định này giúp cơ chế routing của ASP.NET Core MVC xác định controller và action xử lý truy vấn.

Nếu hai placeholder này không nằm sẵn trong chuỗi template, bạn phải cung cấp một object vô danh để chỉ định rõ giá trị của controller và action. Đây cũng là cách chúng ta sử dụng khi khai báo my-route-1.

Đối với parameter do bạn tự định nghĩa, bạn sử dụng nó như là cơ chế truyền dữ liệu từ Url vào action. Cơ chế routing có khả năng trích phần giá trị tương ứng của parameter và truyền vào làm tham số của action. Đây lại là một phần của cơ chế model binding mà chúng ta sẽ học ở một bài khác.

Như trong ví dụ trên, fname và lname cũng đồng thời là hai tham số của action Hi(string fname, string lname).

Số lượng segment và placeholder trong mỗi route hoặc template không hạn chế, có thể bắt buộc hoặc tùy chọn, có thể chịu các giới hạn nhất định.

Paramter tùy chọn và parameter bắt buộc

Trong route template, mặc định mỗi parameter đều là bắt buộc. Nghĩa là nếu bạn đã đặt một parameter vào template, trong Url bắt buộc phải có phần tương ứng với parameter. Nếu thiếu, Url không phù hợp với template.

Nếu bạn muốn một parameter trở thành không bắt buộc, nghĩa là ở vị trí đó có thể có hoặc có thể không đặt giá trị gì, bạn đặt thêm ký tự ? ở sau tên parameter.

Ví dụ, trong route template mặc định "{controller=Home}/{action=Index}/{id?}", {id?} là một parameter tùy chọn, trong khi {controller} và {action} là bắt buộc.

Khi này, cả hai Url Home/Index và Home/Index/3 đều phù hợp (với giả định rằng có HomeController và Index action).

Giá trị mặc định của parameter

Paramter có thể nhận giá trị mặc định bằng cách thêm =giá-trị vào sau tên paramter. Trong route mặc định "{controller=Home}/{action=Index}/{id?}", controller có giá trị mặc định là Home, và action có giá trị mặc định là Index.

Sử dụng giá trị mặc định cho phép bỏ qua paramter tương ứng.

Ví dụ: Url /, /home và /home/index đều ánh xạ sang Index action của HomeController.

Theo đó, / tương ứng với bỏ qua cả controller, action và id; /home tương ứng với bỏ qua action và id

Một số giới hạn khác:

ASP.NET Core MVC cho phép cung cấp thêm những giới hạn đối với parameter trong route template. Đây là những giới hạn về kiểu và phạm vi giá trị.

Bạn đặt thêm giới hạn vào sau tên parameter và phân tách bằng dấu hai chấm.

Bạn cũng có thể kết hợp nhiều giới hạn với nhau. Khi có nhiều giới hạn, bạn cũng phân tách chúng bằng dấu hai chấm.

Bảng sau đây giới thiệu một số loại giới hạn thường dùng:

Constraint	Example	Match examples
int	{qty:int}	123, -123, 0
Guid	{id:guid}	d071b70c-a812-4b54-87d2-7769528e2814
decimal	{cost:decimal}	29.99, 52, -1.01
min(value)	{age:min(18)}	18, 20
length(value)	{name:length(6)}	andrew,123456
optional int	{qty:int?}	123, -123, 0, null
optional int max(value)	{qty:int:max(10)?}	3, -123, 0, null

Attribute routing trong ASP.NET Core MVC

Hãy xây dựng GreetingController với các action như sau:

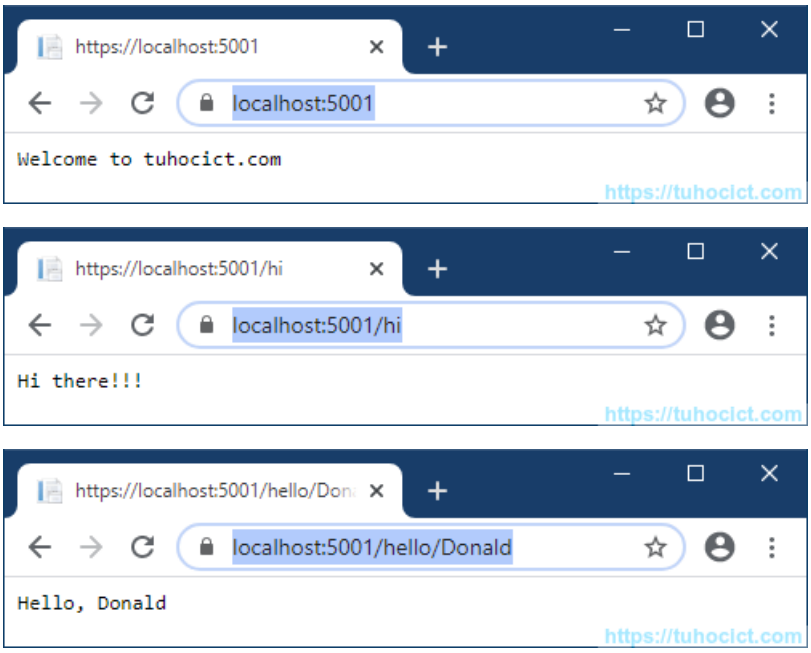
```
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers {
    public class GreetingController {
        [Route("")]
        public string Greeting() => $"Welcome to TuHocIct";

        [Route("hi")]
        public string Hi() => $"Hi there!!!";
```

```
10.
11.          [Route("hello/{name}")]
12.          public string Hello(string name) => $"Hello, {name}";
13.      }
14.  }
```

Chạy ứng dụng với các url `/`, `/hi`, `/hello/Donald` bạn sẽ thu được kết quả như sau:



Ba action của GreetingController đều sử dụng attribute routing:

1. Greeting tương ứng với root url `/`;
2. Hi() tương ứng với url `/hi`;
3. Hello(string name) tương ứng với url `/hello/{name}`.

> Lưu ý, `[Route("hi")]` và `[Route("/hi")]` trong trường hợp này là tương đương nhau và cùng tương ứng với url `/hi`. Tuy nhiên, không phải lúc nào chúng cũng tương đương nhau. Nếu sử dụng ***cơ chế ghép nối url*** của attribute routing, template bắt đầu với ký tự "/" có ý nghĩa hoàn toàn khác biệt.

Bạn cũng để ý thấy rằng khi chạy ứng dụng thì action Greeting sẽ được gọi đầu tiên vì nó ứng với root url "/". Kể cả bạn có cấu hình conventional routing "/" cho một action khác, Greeting sẽ được ưu tiên thực hiện.

Bạn có thể thấy việc định nghĩa attribute routing rất tự do. Tuy nhiên:

- bạn cần đảm bảo các ***template cho các action khác nhau không trùng lặp***;
- bạn có thể định nghĩa ***nhiều template cho cùng một action***.

Attribute routing cũng sử dụng placeholder/parameter trong template tương tự như conventional routing.

Cơ chế hoạt động của attribute routing khác biệt với conventional routing:

1. Khi ứng dụng chạy, ASP.NET Core MVC sẽ scan qua tất cả các action. Nếu action nào được đánh dấu bằng [Route(…)] thì template tương ứng sẽ được trích ra và đặt làm khóa cho một từ điển url, còn action sẽ được sử dụng làm giá trị của mục từ điển tương ứng. Kết thúc quá trình scan sẽ thu được một từ điển bao gồm các cặp url => action.
2. Khi truy vấn tới, url sẽ được trích ra và so sánh với các khóa có trong từ điển. Nếu trùng với khóa nào thì action tương ứng sẽ được thực thi.
3. Nếu một action đã được thực thi, việc xử lý url kết thúc. Tất cả các conventional template bị bỏ qua.
4. Nếu không tìm thấy khóa (url) tương ứng trong từ điển, ASP.NET Core MVC sẽ tiếp tục so khớp với các conventional template.

Qua đây bạn có thể thấy, attribute routing có độ ưu tiên cao hơn conventional routing.

Như vậy, mặc dù ASP.NET Core MVC cho phép sử dụng song song cả conventional và attribute routing, tuy nhiên nếu một action đã sử dụng attribute routing, nó sẽ không thể tham gia vào conventional routing được nữa.

Giờ hãy điều chỉnh GreetingController như sau:

```
1.  [Route("greeting")]
2.  public class GreetingController {
3.  ...
4.  }
```

Bạn chỉ thêm attribute [Route(“greeting”)] vào trước GreetingController class. Nếu chạy ứng dụng, bạn sẽ phải dùng các Url sau:

- `greeting/` cho `Greeting()` action
- `greeting/hi` cho `Hi()` action
- `greeting/hello/{name}` cho `Hello(string name)` action

Tức là url giờ được ghép từ template của controller và template của action. Đây là cơ chế **ghép template** của attribute routing.

Lưu ý về ký tự / đầu template: nếu một template bắt đầu bằng /, nó là một template tuyệt đối. Template tuyệt đối của action **không** ghép được với template của controller. Ví dụ [Route(“/hi”)] là một template tuyệt đối. Action với route này luôn luôn được truy xuất qua url `/hi`.

Một số vấn đề về URL

Qua nội dung của bài học bạn có thể thấy ASP.NET Core MVC rất linh hoạt trong định nghĩa khuôn mẫu URL. Cơ chế routing cho phép bạn định nghĩa ra gần như bất kỳ khuôn mẫu URL nào bạn muốn.

Khi xây dựng ứng dụng web nói chung, bạn nên lưu ý về việc lựa chọn cấu trúc URL. Đặc biệt nếu ứng dụng triển khai trên Internet và tham gia vào máy tìm kiếm.

Thứ nhất, hãy cố gắng tạo cấu trúc URL ngắn gọn và đơn giản.

Bạn hẳn sẽ dễ nhớ những URL như https://tuhocict.com/about, https://tuhocict.com/blog.

Cấu trúc này không khớp với template mặc định của dự án MVC.

Với những trường hợp như thế này bạn có thể sử dụng attribute routing `[Route("about")]` hoặc `[Route("blog")]` trước action tương ứng.

Nếu sử dụng conventional routing, bạn có thể sử dụng khai báo template sau:

```
endpoints.MapControllerRoute("about", "about", new { controller = "Home", action
= "About" });
```

Thứ hai, bạn nên sử dụng những cấu trúc Url thân thiện với máy tìm kiếm (nếu có thể). Điều này đặc biệt cần thiết nếu ứng dụng của bạn thiên về quản lý nội dung trên Internet như blog hoặc trang tin tức.

Ví dụ, Url thân thiện với máy tìm kiếm như https://tuhocict.com/blog/routing-trong-asp-net-core-mvc.

Giả sử nội dung mỗi bài viết được tạo ra từ BlogController.Article action, bạn có thể sử dụng conventional template:

```
endpoints.MapControllerRoute("article", "blog/{slug}", new { controller = "Blog",
action = "Article" });
```

hoặc attribute `[Route("blog/{slug}")]`.

Trong đó slug (mượn tạm thuật ngữ của wordpress) là một chuỗi ký tự tạo ra từ tên bài viết và loại bỏ đi các ký tự đặc biệt. Các từ trong tên gọi ghép với nhau bởi dấu gạch ngang.

Slug dùng làm tham số cho Article action. Template này cũng chỉ định luôn action tiếp nhận Url.

Tuy nhiên đối với site quản lý (hoặc phần backend), bạn không cần tạo các url đẹp như vậy do bạn cần cung cấp nhiều tham số cho action qua Url.

Thứ ba, nếu cần cung cấp tham số cho action thông qua Url, bạn có thể sử dụng *query string* (?fname=Donald&lname=Trump) hoặc *route parameter* ({fname} và {lname} trong route template).

Action xử lý Url phải có các tham số đầu vào tương ứng trùng tên.

Cơ chế model binding có khả năng tự động trích giá trị và truyền vào tham số tương ứng của action.

Khi truyền tham số qua Url, bạn nên kết hợp các giới hạn về kiểu hoặc khoảng giá trị trong route template.

Kết luận

Bài học đã giới thiệu chi tiết về cơ chế routing trong ASP.NET Core MVC. Khi hiểu cơ chế này bạn có thể dễ dàng xây dựng thêm các route template để ánh xạ với action mình xây dựng. ASP.NET Core MVC cho phép bạn xây dựng ra gần như bất kỳ cấu trúc URL nào.