Model binding trong ASP.NET Core MVC

Model binding là quá trình tự động trích dữ liệu từ truy vấn HTTP và biến đổi thành object .NET cung cấp cho action. Model binding diễn ra ngay sau khi routing lựa chọn được action để thực thi.

Model binding không thuộc về kiến trúc MVC nhưng lại đóng góp phần quan trọng trong hoạt động của ứng dụng. Model binding giải phóng lập trình viên ASP.NET Core khỏi các thao tác xử lý truy vấn cấp thấp để tập trung cho logic riêng của ứng dụng.

Model binding của ASP.NET Core MVC hoạt động hoàn toàn tương tự parameter binding của Razor Pages.

NỘI DUNG CỦA BÀI [Ẩn]

Truyền dữ liệu cho controller action

Chúng ta bắt đầu bằng một ví dụ. Hãy tạo một dự án rỗng và cấu hình sử dụng Mvc Middleware với Startup class sau:

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

namespace WebApplication1 {
    public class Startup {
        public void ConfigureServices(IServiceCollection services) => services.AddContro

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env) {
            app.UseRouting();

            app.UseEndpoints(endpoints => {
                endpoints.MapControllerRoute("power", "power/{number?}/{power?}", new {
                endpoints.MapControllerRoute("default", "{controller}/{action}/{id?}");
            });
        }
    }
}
```

Chú ý chúng ta thêm một route template mới: `endpoints.MapControllerRoute("power", "power/{number?}/{power?}", new { controller = "App", action = "Power" });` Route này chấp nhận các Url như `power/`, `power/2/4` và ánh xạ sang action `Power` của `AppController`.

Xây dựng AppController với action Power như sau:

```
using System;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1 {
    public class AppController : Controller {
        public IActionResult Power(int number, int power) {
            var res = Math.Pow(number, power);
            return Content($"{number} powered by {power} is {res}");
        }
    }
}
```

Để đơn giản, chúng ta không xây dựng view. Kết quả trả về chỉ là một chuỗi ký tự.

> Bạn cũng có thể sử dụng attribute routing `[Route("power/{number?}/{power?}")]` cho Power.

Giờ hãy chạy ứng dụng và thử nghiệm với các URL sau:

1. `/power` (kết quả là 0 powered by 0 is 1)
2. `/power/2/4` (2 powered by 4 is 16)
3. `/power?number=3&power=2` (3 powered by 2 is 9)
4. `/power/2/4?number=3&power=2` (2 powered by 4 is 16)

Trong ví dụ trên chúng ta đã minh họa việc truyền dữ liệu cho action bằng cách sử dụng query string và route parameter.

- Trong trường hợp 1, number và power không nhận được giá trị từ ngoài nên chúng có giá trị mặc định 0 của kiểu số trong C#.
- Trường hợp 2, các giá trị được truyền qua route parameter. Theo thứ tự quy định trong route template, number = 2, power = 4.
- Trường hợp 3, các giá trị được truyền qua query string. Các khóa trùng tên với tham số của Power.
- Trường hợp 4 kết hợp 2 và 3. Bạn có thể thấy, giá trị trong route parameter được sử dụng. Giá trị từ query string bị bỏ qua.

Bạn có thể để ý thấy rằng, action Power không thực hiện bất kỳ lệnh đặc biệt nào để lấy dữ liệu từ URL. Nó chỉ là một phương thức C# cơ bản thông thường nhất. Tuy vậy, khi hoạt động, dữ liệu từ query string hoặc từ route parameter đều được trích và truyền sang cho action Power một cách tự động.

Khái niệm model binding trong ASP.NET Core MVC

Khi học về truy vấn HTTP ở những bài học đầu tiên về ASP.NET Core bạn đã biết rằng truy vấn HTTP thực sự chỉ là một chuỗi văn bản có định dạng.

Bạn cũng đã học rằng, action trong ASP.NET Core MVC là những phương thức C# thông thường. Như trong ví dụ trên, action là phương thức `public IActionResult Power(int number, int power) { ... }`.

Điểm lưu ý ở action này là tham số đầu vào của nó là hai biến có kiểu `int`. Bạn không cần thực hiện bất kỳ thao tác gì liên quan đến truy vấn HTTP nhưng bạn vẫn có hai giá trị int để sử dụng trong action!

Ở đây sẽ phát sinh câu hỏi: ASP.NET Core MVC đã làm gì để action có được các giá trị tham số phù hợp?

ASP.NET Core MVC sử dụng một cơ chế có tên gọi là *model binding* để thực hiện công việc này. **Model binding** là cơ chế trích giá trị từ truy vấn HTTP và biến đổi thành các object .NET cung cấp cho action.

> Object được tạo ra trong quá trình model binding được gọi là **binding model**. Binding model là một trong nhiều loại model được phân biệt rõ ràng trong ASP.NET Core MVC, cùng với application model, domain model, view model. Chúng ta đã phân biệt các loại model này trong bài học giới thiệu về mẫu kiến trúc MVC.

Model binding cũng là điểm làm cho ASP.NET Core MVC trở nên thân thiện với lập trình viên. Bạn không cần quan tâm về truy vấn HTTP. ASP.NET Core MVC thông qua model binding tự động trích dữ liệu phù hợp từ truy vấn HTTP để tạo object phù hợp cho action. Bạn chỉ cần tập trung vào xử lý logic của ứng dụng.

> Model binding trong ASP.NET Core MVC tương tự như parameter binding trong Razor Pages.

Khi truy vấn HTTP tới, cơ chế routing sẽ xác định action cần thực thi. Dựa vào danh sách tham số của action, model binding tìm kiếm giá trị phù hợp trong truy vấn HTTP. Khi tìm kiếm được giá trị có tên trùng với tham số, model binding sẽ biến đổi giá trị (convert) từ string về kiểu của tham số.

Sở dĩ model binding có thể tìm kiếm giá trị trong truy vấn là vì HTTP thường chứa dữ liệu ở dạng các cặp `biến=giá trị`. Ví dụ bạn thường xuyên gặp lối viết như `?id=5&category=book`. Đây là lối viết của query string. Trong phần thân của truy POST cũng chứa dữ liệu theo cách tương tự. Trường hợp dữ liệu nằm trong segment của URL, cơ chế routing sẽ làm nhiệm vụ so khớp URL với route template. Từ kết quả so khớp này sẽ rút ra giá trị tương ứng với placeholder (có vai trò tương tự tên biến).

Binding source

Các vị trí có thể chứa giá trị trong một truy vấn HTTP được gọi là **binding source**.

Thông thường, trong một truy vấn HTTP có thể có 3 binding source: phần thân truy vấn POST, URL, query string.

Do dữ liệu tới có thể chứa ở nhiều binding source, model binding tìm kiếm theo thứ tự sau:

1. Dữ liệu chứa trong thân của truy vấn POST: những giá trị này được gọi là **form value** do nó được thu thập từ HTML form.
2. Dữ liệu chứa trong URL: ví dụ trong URL `/product/get/3`, giá trị 3 là id của sản phẩm cần lấy. Giá trị gửi kèm URL như thế này có tên gọi là **route data/value**. Để URL có thể mang route data, khi định nghĩa route template bạn cần sử dụng placeholder/route parameter.
3. Dữ liệu chứa trong query string của URL: ví dụ trong Url `/product/get?id=3`, id=3 là phần query string.

Nếu một giá trị trùng tên và nằm ở nhiều vị trí, cơ chế model binding sẽ lấy nó theo thứ tự ưu tiên: form value > route value > query string value. Tức là, form value có độ ưu tiên cao nhất, query string value có độ ưu tiên thấp nhất.

Bạn có thể hiểu thế này, giả sử biến name xuất hiện cả trong form, route và query string thì model binding sẽ sử dụng giá trị trong form. Giá trị biến name trong route và query string bị bỏ qua.

Đây là ví dụ về việc kết hợp nguồn dữ liệu từ URL và body. Model binding có thể trích và biến đổi dữ liệu thành tham số truyền cho action.

Do sự phân biệt trên, bạn hoàn toàn có thể kết hợp cung cấp dữ liệu qua nhiều binding source. Ví dụ, có thể đồng thời cung cấp dữ liệu qua route data và query string.

Từ khía cạnh xây dựng action, bạn không cần quan tâm hay phân biệt dữ liệu đến từ đâu. Bạn chỉ cần đặt tham số cho action.

Binding với kiểu class

Model binding hoạt động với cả kiểu cơ sở của C# cũng như kiểu do người dùng xây dựng (class, struct).

Tiếp tục với project thử nghiệm. Hãy tạo một class Book mới như sau:

```
1.  namespace WebApplication1 {
2.      public class Book {
3.          public int Id { get; set; }
4.          public string Title { get; set; }
5.          public string Authors { get; set; }
6.          public string Publisher { get; set; }
7.          public int Year { get; set; }
8.      }
9.  }
```

Bạn có thể để file mã nguồn của class này ở đâu tùy ý. Để đơn giản, có thể đặt thẳng trong thư mục project.

Thêm action sau vào AppController:

```
1.  public IActionResult CreateNewBook(Book book) {
2.      return Content($"{book.Title} by {book.Authors} ({book.Publisher}, {book.Year})");
3.  }
```

Chạy thử với URL sau: /app/createnewbook?title=C%20programming&authors=Donald%20Trump&publisher=ICT&year=2020

Bạn thu được kết quả

```
C programming by Donald Trump (ICT, 2020)
```

Ví dụ trên cho thấy rằng, cơ chế model binding không chỉ hoạt động với các kiểu cơ sở mà còn hoạt động với cả các kiểu class phức tạp hơn do người dùng định nghĩa.

Trong tình huống này bạn để ý rằng ở binding source (ở đây là query string) bạn phải sử dụng biến trùng tên với các public property của class. Trong query string, tên biến không phân biệt hoa/thường.

Ở đây tình hình diễn ra phức tạp hơn. Khi đã xác định được action cần thực thi, cơ chế binding thử tạo object của class Book trước (vì action đòi hỏi tham số kiểu Book).

Khi khởi tạo xong object, cơ chế binding sẽ tìm qua tất cả các public property của object. Ứng với mỗi public property, nó sẽ tìm kiếm tham số có tên tương ứng trong các binding

source. Nếu tìm thấy tham số phù hợp với property, nó sẽ trích giá trị tương ứng và gán vào cho property.

Quá trình tìm kiếm này tương tự như với tình huống binding với kiểu cơ sở.

Khi quá trình tìm kiếm kết thúc, object sẽ truyền vào cho action như thường lệ.

Khi bạn hiểu cơ chế này bạn hoàn toàn có thể hình dung được nếu một property nào đó không có giá trị tương ứng trong truy vấn, nó sẽ nhận giá trị mặc định. Thêm vào đó, object (trong trường hợp này là book) sẽ không bao giờ nhận giá trị null (vì nó luôn được khởi tạo).

Model binding với form

Để kết thúc bài học này chúng ta xem xét tình huống rất thường gặp trong phát triển ứng dụng – thực hiện model binding với dữ liệu đến từ form.

Form có thể trả dữ liệu về server bằng truy vấn GET (mặc định) hoặc truy vấn POST.

Trong trường hợp trả về bằng truy vấn GET, giá trị từ tất cả các điều khiển trên form mà được đặt tên (thiết lập qua attribute name=…) sẽ được tập hợp thành các cặp tên_điều_khiển=giá_trị trong query string của URL.

Trong trường hợp trả về bằng truy vấn POST, các cặp tên_điều_khiển=giá_trị được đặt trong thân truy vấn.

Dù là trả về theo truy vấn nào, cơ chế model binding đều có thể trích được dữ liệu tự động.

Hãy cùng thực hiện một ví dụ (thực hiện tiếp).

Tạo mới file Form.cshtml trong dự án (cho tiện lợi) và viết code như sau:

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Form binding</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css
</head>
<body>
    <form action="CreateNewBook" method="post" class="p-3 w-50 shadow">
        <label for="title" class="">Tiêu đề</label>
        <input name="title" type="text" class="form-control mb-2" id="title" />
        <label for="authors" class="">Tác giả</label>
        <input name="authors" type="text" class="form-control mb-2" id="authors" />
        <label for="publisher" class="">Nhà xuất bản</label>
        <input name="publisher" type="text" class="form-control mb-2" id="publisher" />
        <label for="year" class="">Năm xuất bản</label>
        <input name="year" type="number" class="form-control mb-2" id="year" />
        <input type="submit" value="Submit" class="btn btn-primary btn-block" />
    </form>
</body>
</html>
```

Để ý thẻ <form action="CreateNewBook" method="post">. Khi người dùng ấn nút submit, dữ liệu trả về cho action CreateNewBook mà chúng ta đã xây dựng từ ví dụ trước đó theo truy vấn POST.

Các điều khiển trên form đều phải thiết lập attribute **name** với giá trị trùng với các public property của lớp Book (đã tạo trong ví dụ trước đó). Không có name, giá trị sẽ không được đóng gói để gửi về server.

> Đọc lại bài học về cơ chế hoạt động của HTML form và cách xử lý điều khiển tren form nếu bạn không nhớ. Các cơ chế này trong Razor Pages và MVC là giống hệt nhau. Nếu đã thành thạo cách sử dụng trên Razor Pages, bạn áp dụng nguyên vẹn với MVC.

Chạy ứng dụng với Url `/app/createnewbookform`:

Khi ấn submit, dữ liệu trả về action CreateNewBook xử lý như sau:

Kết quả này không có gì khác biệt với trường hợp binding với query string mà chúng ta đã làm trước đó.

Sự khác biệt duy nhất là giờ đây binding source là thân (body) của truy vấn HTTP với dữ liệu thu thập từ điều khiển trên form. Như đã nói từ trước, cơ chế model binding ưu tiên dữ liệu đến từ form.

Nếu trong form trên bạn thay `method="get"`, kết quả không khác biệt, ngoại trừ dữ liệu sẽ xuất hiện trên thanh địa chỉ, thay vì đóng gói trong thân truy vấn.

Kết luận

Bài học đã hướng dẫn cách làm việc cơ bản với cơ chế model binding trong ASP.NET Core MVC.

Cơ chế này giúp tự động trích và biến đổi dữ liệu từ truy vấn HTTP cho phù hợp với yêu cầu về tham số đầu vào của action.

Cơ chế model binding có khả năng làm việc với cả các kiểu cơ sở của C# cũng như các kiểu object phức tạp do người dùng tự xây dựng.