Model binding cơ bản trong Razor Pages

Model binding là một loại kỹ thuật đặc trưng và rất quan trọng trong Razor Pages giúp bạn dễ dàng làm việc với dữ liệu người dùng cung cấp qua truy vấn. Nhờ Model binding bạn không cần phải trích dữ liệu từ truy vấn một cách thủ công. Thay vào đó, model binding sẽ trích dữ liệu tự động giúp quá trình đọc dữ liệu từ truy vấn nhanh và an toàn hơn.

Bài học này sẽ cung cấp những nội dung quan trọng nhất để bạn hiểu và áp dụng được kỹ thuật model binding trong Razor Pages.

NỘI DUNG CỦA BÀI [Ẩn]

Model binding là gì?

Khi học về cách truy xuất dữ liệu từ URL hoặc truy xuất dữ liệu form, bạn đã biết rằng Razor Pages phân tách các thành phần của truy vấn HTTP và đưa vào object Request. Bạn có thể truy xuất dữ liệu của truy vấn thông qua property Query hoặc Form.

Ví dụ, nếu page nhận truy vấn GET với URL như sau:

```
/?username=Donald%20Trump&email=trump@gmail.com
```

Bạn có thể truy xuất giá trị của username và email từ server code như sau:

```
var username = Request.Query["username"];
var email = Request.Query["email"];
```

Hoặc nếu bạn sử dụng form:

```
<form method="post">
    <input name="username" type="text">
    <input name="email" type="text">
```

```
    <button type="submit">POST Subscribe</button>
</form>
```

Bạn sẽ truy xuất giá trị của các textbox username và email như sau:

```
var username = Request.Form["username"];
var email = Request.Form["email"];
```

Đây là cách thức truy xuất dữ liệu từ truy vấn cơ bản nhất và thường gặp trong các web framework. Phương pháp này đơn giản nhưng chỉ phù hợp với chuỗi truy vấn hoặc form đơn giản.

Khi bạn phải xử lý những form phức tạp với hàng chục thông tin, việc truy xuất dữ liệu như trên có thể tạo ra hàng loạt vấn đề.

Vấn đề dễ thấy nhất là cơ chế nhắc code của Intellisense không hoạt động được khiến bạn phải ghi nhớ các tên gọi. Từ đây dẫn tới khả năng viết lẫn. Ví dụ khi truy xuất email, bạn dễ dàng viết Request.Form[“emial”] thay cho Request.Form[“email”]. Những lỗi như vậy rất khó theo dõi.

Để giải quyết những vấn tương tự, Razor Pages đưa ra cơ chế model binding.

Model binding là cơ chế tự động phân tách giá trị từ truy vấn HTTP và ánh xạ chúng vào tham số của phương thức xử lý truy vấn (handler method) hoặc vào thuộc tính của model class tương ứng.

Cơ chế model binding giải phóng lập trình viên khỏi việc trích xuất dữ liệu thủ công khỏi truy vấn rồi tự gán cho từng biến cục bộ. Qua đó, model binding giúp tránh lỗi và tăng hiệu quả công việc.

Parameter và Property binding

Model binding trong Razor Pages chia làm hai loại: parameter binding và property binding.

Parameter binding là ánh xạ dữ liệu của truy vấn vào tham số của phương thức xử lý (handler). Trong parameter binding, bạn đặt tham số vào phương thức xử lý truy vấn. Tên của tham số cần trùng với tên của điều khiển (form)/tên tham số của chuỗi truy vấn/tên placeholder của route.

Property binding là ánh xạ dữ liệu của truy vấn vào property của model class. Trong property binding, bạn khai báo thêm các property trong model class sao cho tên property trùng với tên điều khiển/tham số/placeholder, đồng thời đặt thêm attribute [BindProperty] phía trước.

Khi này cơ chế model binding sẽ tự trích dữ liệu từ truy vấn, chuyển đổi thành kiểu phù hợp và truyền vào tham số và thuộc tính tương ứng.

> Trên thực tế khi học về handler, URL và Form, bạn đã sử dụng parameter binding để ánh xạ dữ liệu của truy vấn vào tham số của handler.

Sự tương quan giữa tên tham số/property và tên điều khiển/tham số/placeholder **không phân biệt chữ hoa/thường**. Nghĩa là bạn có thể đặt tên viết thường cho điều khiển nhưng vẫn có thể dùng tên viết hoa chữ cái đầu cho property. Như vậy bạn vẫn giữ được quy ước đặt tên ở HTML và C#.

Do việc chuyển đổi kiểu là tự động, bạn cần đặt **kiểu tham số/ property phù hợp với dữ liệu** bạn chờ đợi. Ví dụ, nếu trên form bạn đặt input để người dùng nhập số thì bạn cần đặt kiểu cho tham số / property tương ứng là một trong các kiểu số.

Trong một số trường hợp bạn phải dùng kiểu đặc biệt, thay cho các kiểu cơ sở quen thuộc. Ví dụ, khi tải file, kiểu của tham số / property tương ứng với file tải lên là IFormFile.

Để dễ hiểu, chúng ta sẽ đi vào từng trường hợp cụ thể.

Sử dụng Model Binding với form

Tạo page mới đặt tên là FormBinding. Viết code cho Razor view như sau:

```
@page
@model ModelBinding.Pages.FormBindingModel
@{
    ViewData["Title"] = "FormBinding";
    var method = Request.Method.ToUpper();
}

@if (method == "GET") {
    <form class="center border border-light p-3 w-50" method="post">
        <div class="text-center">
            <p class="h4 mb-4">Subscribe</p>
            <p>Join our mailing list. We write rarely, but only the best content.</p>
        </div>

        <input name="fullname" type="text" placeholder="Name" class="form-control mb-
        <input name="email" type="email" placeholder="E-mail" class="form-control mb-
        <input name="age" type="number" placeholder="Age" class="form-control mb-2">
        <button type="submit" class="btn btn-info btn-block">Register</button>
    </form>
}
else if (method == "POST") {
    <div class="center text-center border border-success p-3 w-50">
        <p class="h5">@Model.Message</p>
    </div>
}
```

Chú ý tên các điều khiển lần lượt là **fullname**, **age** và **email**.

Form này cho kết quả như sau:

Parameter binding

Nếu sử dụng parameter binding, model class sẽ có dạng như sau:

```
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace ModelBinding.Pages {
    public class FormBindingModel : PageModel {
        public string Message { get; private set; }
        public void OnPost(string fullName, string eMail, int age) => Message = age <
                $"Sorry, {fullName}. You cannot subscribe for our site." :
                $"Hello, {fullName}. Thank you for your subscription. We will send em
    }
}
```

Để ý tên tham số của OnGet lần lượt là **fullName**, **eMail** và **age**. Riêng age có kiểu là int.

> Hãy lưu ý rằng,
> (1) tên tham số của OnPost và tên điều khiển tương ứng có sự khác biệt về chữ hoa/thường. Razor Pages vẫn có khả năng ánh xạ chính xác sang tham số tương ứng của OnGet;
> (2) mặc dù trong truy vấn HTTP, mọi thứ đều là văn bản (từ góc nhìn của C#) nhưng Razor Pages vẫn chuyển đổi được giá trị từ điều khiển age (form) về giá trị số nguyên cho tham số age của OnPost.

Cơ chế parameter binding này hoạt động rất tốt với phương thức xử lý truy vấn. Tuy nhiên, trong nhiều trường hợp bạn muốn sử dụng các giá trị eMail, fullName và age trong toàn class. Khi này bạn nên sử dụng cơ chế property binding.

Property binding

Thay thế code của FormBindingModel như sau:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace ModelBinding.Pages {
    public class FormBindingModel : PageModel {
        [BindProperty]
        public int Age { get; set; }
        [BindProperty]
        public string FullName { get; set; }
        [BindProperty]
        public string Email { get; set; }

        public string Message { get; private set; }

        public void OnPost() => Message = Age < 18 ?
               $"Sorry, {FullName}. You cannot subscribe for our site." :
               $"Hello, {FullName}. Thank you for your subscription. We will send ema
    }
}
```

Bạn có thể thấy trong đoạn code trên chúng ta bỏ tham số trong OnPost. Thay vào đó chúng ta khai báo thêm 3 property: **Age**, **FullName** và **Email**. Các property này đều được đánh dấu bởi attribute **[BindProperty]**. Bạn có thể thấy tên property không hoàn toàn trùng khớp (hoa/thường) với tên điều khiển.

Trong trường hợp này Razor Pages sẽ gắn dữ liệu từ truy vấn vào các property có tên tương ứng. Bạn có thể truy xuất các property ở bất kỳ đâu trong model class cũng như trên Razor view.

Property binding cho toàn thể model class

Nếu bạn có nhiều property hơn nữa, việc đánh dấu [BindProperty] cho từng property làm code nhìn rối hơn nhiều. Bắt đầu từ ASP.NET Core 2.1 bạn có thể **sử dụng attribute [BindProperties] cho model class** như sau:

```
namespace ModelBinding.Pages {
    [BindProperties]
    public class FormBindingModel : PageModel {
        public int Age { get; set; }
        public string FullName { get; set; }
        public string Email { get; set; }

        public string Message { get; private set; }

        public void OnPost() => Message = Age < 18 ?
               $"Sorry, {FullName}. You cannot subscribe for our site." :
               $"Hello, {FullName}. Thank you for your subscription. We will send ema
    }
}
```

Để ý rằng tất cả attribute [BindProperty] đã bị xóa bỏ. Bạn không cần đến nó nữa.

[BindProperties] tự động áp dụng property binding cho bất kỳ **public property** nào của class. Theo đó, Age, FullName, Email đều tham gia vào property binding.

Tuy nhiên cách này có một vấn đề nhỏ: Message cũng tham gia vào property binding vì nó cũng là một public property. Vì vậy, khi sử dụng [BindProperties] cần lưu ý đến đặc điểm của class để hạn chế những property không cần thiết tham gia vào quá trình binding.

Sử dụng Model Binding với chuỗi truy vấn

Giả sử chúng ta muốn xử lý chuỗi truy vấn có dạng: ?fullname=xyz&email=abc@gmail.com&age=18.

Tạo page mới UrlBinding và viết code cho Razor view như sau:

```
@page
@model ModelBinding.Pages.UrlBindingModel
@{
    ViewData["Title"] = "UrlBinding";
}

<strong>@Model.Message</strong>
```

Sử dụng parameter binding

Nếu sử dụng **parameter binding**, bạn viết code của model class như sau:

```
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace ModelBinding.Pages {
    public class UrlBindingModel : PageModel {
        public string Message { get; set; }
        public void OnGet(string fullName, string eMail, int age) => Message = age <
            $"Sorry, {fullName}. You cannot subscribe for our site." :
            $"Hello, {fullName}. Thank you for your subscription. We will send email
    }
}
```

Bạn có thể thấy, cách sử dụng parameter binding với chuỗi truy vấn và phương thức GET không có gì khác biệt với xử lý form. Thực tế cách thức này bạn đã sử dụng nhiều lần trong bài học về truy xuất dữ liệu từ Url.

Sử dụng property binding

Mặc định property binding chỉ hỗ trợ truy vấn POST. Nếu muốn sử dụng với truy vấn GET để trích dữ liệu từ chuỗi truy vấn vào property, bạn cần điều chỉnh attribute [BindProperty] thành `[BindProperty(SupportsGet = true)]`:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace ModelBinding.Pages {
    public class UrlBindingModel : PageModel {
        [BindProperty(SupportsGet =true)]
        public int Age { get; set; }

        [BindProperty(SupportsGet = true)]
        public string FullName { get; set; }

        [BindProperty(SupportsGet = true)]
        public string Email { get; set; }

        public string Message { get; private set; }

        public void OnGet() => Message = Age < 18 ?
               $"Sorry, {FullName}. You cannot subscribe for our site." :
               $"Hello, {FullName}. Thank you for your subscription. We will send ema
    }
}
```

Như vậy, ngoài trừ yêu cầu thêm Supportsget = true, property binding cho truy vấn GET với query string không có gì khác biệt so với truy vấn POST và form.

Tương tự, bạn cũng có thể sử dụng attribute `[BindProperties(SupportsGet = true)]` cho model class.

Sử dụng Model Binding với route data

Route data là loại dữ liệu được đặt trong phần path của URL thay vì trong query string. Như bạn đã học, Razor Pages cho phép định nghĩa route (mẫu URL) của mỗi page sử dụng cụm đánh dấu (placeholder). Cụm đánh dấu được sử dụng như một biến trong quá trình trích xuất dữ liệu từ URL.

Tạo page mới RouteBinding. Viết code như sau cho Razor view:

```
@page "/register/{fullname?}/{email?}/{age:int?}"
@model ModelBinding.Pages.RouteBindingModel
@{
    ViewData["Title"] = "RouteBinding";
}
<a href="/register/Donald%20Trump/trump@gmail.com/18">Get binding</a>
<strong>@Model.Message</strong>
```

Ở dòng đầu tiên chúng ta sử dụng directive @page để ghi đè mẫu url mặc định tới page:

```
@page "/register/{fullname?}/{email?}/{age:int?}"
```

Mẫu này chỉ định tất cả các url có phần path cần có dạng như `/register/donald%20trump/trump@gmail.com/18`. Trong đó, phần cuối cùng phải có dạng số. Trong Url có thể vắng mặt cả ba thành phần này (do có dấu ? đứng sau tên placeholder).

> Lưu ý `%20` là dấu cách. Dấu cách không được phép xuất hiện trong URL. Nó là ký tự đặc biệt dùng để phân tách các phần của HTTP Request Line. Nếu cần biểu diễn dấu cách trong URL, bạn thay nó bằng cụm **%20**.

Khi này bạn có thể sử dụng **parameter binding** để trích xuất dữ liệu từ Url path như sau:

```
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace ModelBinding.Pages {
    public class RouteBindingModel : PageModel {
        public string Message { get; private set; }
        public void OnGet(string FullName, string Email, int Age) => Message = Age <
                $"Sorry, {FullName}. You cannot subscribe for our site." :
                $"Hello, {FullName}. Thank you for your subscription. We will send ema
    }
}
```

Cách sử dụng parameter binding giống hệt như trong trường hợp xử lý chuỗi truy vấn.

Do đây cũng là một truy vấn GET, bạn có thể tiếp tục sử dụng kỹ thuật property binding cho route data, giống hệt như đối với query string:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace ModelBinding.Pages {
    // [BindProperties(SupportsGet = true)]
    public class RouteBindingModel : PageModel {

        [BindProperty(SupportsGet =true)]
        public int Age { get; set; }

        [BindProperty(SupportsGet = true)]
```

```
        public string FullName { get; set; }

        [BindProperty(SupportsGet = true)]
        public string Email { get; set; }

        public string Message { get; private set; }

        public void OnGet() => Message = Age < 18 ?
               $"Sorry, {FullName}. You cannot subscribe for our site." :
               $"Hello, {FullName}. Thank you for your subscription. We will send ema
    }
}
```

Ở đoạn code trên bạn tiếp tục sử dụng attribute [BindProperty (SupportsGet = true)] đối với property hoặc [BindProperties(SupportsGet = true)] đối với model class.

Sử dụng Model Binding với object

Từ đầu đến giờ chúng ta đều chỉ vận dụng model binding với các kiểu dữ liệu đơn giản như string, int. Nhìn chung, model binding hoạt động tương tự nhau với hầu hết các kiểu dữ liệu cơ sở của C#.

Tuy nhiên, trong các dự án bạn chắc chắn sẽ phải xây dựng các domain class riêng với rất nhiều property. Theo như kỹ thuật đã học, bạn có thể khai báo các property của domain class ngay trong model class để sử dụng property binding.

Tuy nhiên, việc trộn lẫn domain class với model class là rất khó chấp nhận vì hai loại class này có mục đích khác biệt.

Razor Pages cho phép bạn thực hiện property binding với cả các class phức tạp do bạn tự xây dựng.

Sử dụng property binding

Hãy cùng thực hiện ví dụ sau:

Bước 1. Tạo thư mục Models trực thuộc dự án.

Bước 2. Tạo class Subscription trong thư mục Models và viết code như sau:

```
namespace ModelBinding.Models {
    public class Subscription {
        public int Id { get; set; }
        public string FullName { get; set; }
        public string Email { get; set; }
        public int Age { get; set; }
    }
}
```

Đây là một POCO class được sử dụng làm domain class. Bạn cũng phải thường xuyên tạo ra các class tương tự khi phát triển ứng dụng quản lý (LOB – Line of Business) sử dụng Entity Framework.

Bước 3. Tạo trang mới trong thư mục Pages đặt tên là ObjectBinding.

Bước 4. Viết code như sau cho model class (file ObjectBinding.cshtml.cs):

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
```

```
3.
4.  using ModelBinding.Models;
5.
6.  namespace ModelBinding.Pages {
7.      public class ObjectBindingModel : PageModel {
8.          public string Message { get; private set; }
9.          private string ResponseMessage(Subscription subscription) => Message = subscr
10.             $"Sorry, {subscription.FullName}. You cannot subscribe for our site." :
11.             $"Hello, {subscription.FullName}. Thank you for your subscription. We will
12.
13.         [BindProperty]
14.         public Subscription Subscription { get; set; }
15.         public void OnPost() => Message = ResponseMessage(Subscription);
16.
17.         //public void OnPost(Subscription subscription) => Message = ResponseMessage(
18.     }
19. }
```

Để ý dòng attribute [BindProperty] giờ áp dụng cho property Subscription thuộc kiểu phức tạp Subscription do chúng ta vừa xây dựng. Như vậy [BindProperty] có thể áp dụng với property có kiểu phức tạp.

Bước 5. Viết code như sau cho file ObjectBinding.cshtml:

```
1.  @page
2.  @model ModelBinding.Pages.ObjectBindingModel
3.  @{
4.      ViewData["Title"] = "ObjectBinding";
5.      var method = Request.Method.ToUpper();
6.  }
7.
8.
9.  @if (method == "GET") {
10.     <form class="center border border-light p-3 w-50" method="post">
11.         <div class="text-center">
12.             <p class="h4 mb-4">Subscribe</p>
13.             <p>Join our mailing list. We write rarely, but only the best content.</p>
14.         </div>
15.
16.         <input name="Subscription.Fullname" type="text" placeholder="Name" class="for
17.         <input name="Subscription.Email" type="email" placeholder="E-mail" class="for
18.         <input name="Subscription.Age" type="number" placeholder="Age" class="form-co
19.         <button type="submit" class="btn btn-info btn-block">Register</button>
20.     </form>
21. }
22. else if (method == "POST") {
23.     <div class="center text-center border border-success p-3 w-50">
24.         <p class="h5">@Model.Message</p>
25.     </div>
26. }
```

Thực tế bạn sử dụng lại form đã có từ ví dụ sử dụng model binding với form.

Trong form này để ý cách đặt tên cho điều khiển: `name="Subscription.Fullname"`, `name="Subscription.Age"`, `name="Subscription.Email"`. Nghĩa là bây giờ điều khiển được đặt tên theo tên đầy đủ của property trong model class.

Chạy chương trình và bạn thu được cùng kết quả như các ví dụ khác.

> Bạn hoàn toàn có thể sử dụng [BindProperties] đối với model class.

Cơ chế model binding của Razor Pages có khả năng tự trích dữ liệu từ form và dùng chúng để khởi tạo object.

Sử dụng parameter binding

Nếu không muốn sử dụng property binding, bạn cũng có thể áp dụng parameter binding cho phương thức OnPost như sau:

```
using Microsoft.AspNetCore.Mvc.RazorPages;

using ModelBinding.Models;

namespace ModelBinding.Pages {
    public class ObjectBindingModel : PageModel {
        public string Message { get; private set; }
        private string ResponseMessage(Subscription subscription) => Message = subscr
            $"Sorry, {subscription.FullName}. You cannot subscribe for our site." :
            $"Hello, {subscription.FullName}. Thank you for your subscription. We will

        public void OnPost(Subscription subscription) => Message = ResponseMessage(su
    }
}
```

Trong trường hợp này, biến subscription thuộc kiểu Subscription được sử dụng làm tham số của OnPost. Cơ chế model binding của Razor Pages có khả năng tự trích dữ liệu từ form và dùng chúng để khởi tạo object.

Object binding với truy vấn GET

Mặc định việc binding với object được sử dụng với truy vấn POST. Tuy nhiên bạn cũng có thể sử dụng kỹ thuật này với truy vấn GET. Tuy nhiên cần điều chỉnh một chút.

Nếu sử dụng **parameter binding**, bạn cần thêm attribute [FromQuery] vào trước tham số của handler. Attribute này báo cho Razor Pages biết cần trích dữ liệu từ chuỗi truy vấn.

```
public void OnGet([FromQuery] Subscription subscription) {
    Message = ResponseMessage(subscription);
}
```

Phương thức OnGet như trên có thể trích dữ liệu từ query string và khởi tạo object subscription dựa trên dữ liệu thu được. Bạn có thể nhập các truy vấn với query string như bình thường:

```
?fullname=xyz&email=abc@gmail.com&age=18
```

Sở dĩ phải có attribute `[FromQuery]` là vì Razor Pages mặc định hiểu rằng parameter binding cho object sẽ luôn lấy dữ liệu từ form. Tức là, nếu không chỉ định rõ nguồn thì Razor Pages sẽ luôn hiểu object tham số của handler sẽ được áp dụng attribute `[FromForm]`. Do vậy, khi đọc dữ liệu từ form bạn không cần viết [FromForm] trước tham số nữa.

Như bạn thấy, tên mỗi khóa trong query string chỉ cần tương đồng với tên property tương ứng của class Subscription. Bạn không cần đặt tên đặc biệt như đối với form control.

Nếu sử dụng **property binding**, bạn cần đánh dấu như sau:

```
[BindProperty(SupportsGet=true)]
public Subscription Subscription {get; set;}
```

Object binding với route data

Object binding có thể hoạt động cả với route data.

Giả sử bạn định nghĩa route cho trang bằng directive `@page "/register/{fullname?}/{email?}/{age:int?}"` .

Nếu sử dụng **parameter binding**, bạn cần chỉ định attribute [FromRoute] trước tên tham số để báo cho Razor Pages biết rằng cần trích dữ liệu từ route data để khởi tạo object tương ứng của tham số.

```
1.  public void OnGet([FromRoute] Subscription subscription) {
2.      Message = ResponseMessage(subscription);
3.  }
```

Nếu sử dụng **property binding**, bạn áp dụng [BindProperty(SupportsGet=true)] giống hệt như khi sử dụng chuỗi truy vấn:

```
1.  [BindProperty(SupportsGet=true)]
2.  public Subscription Subscription {get; set;}
```

Kết luận

Bài học này đã hướng dẫn bạn kỹ thuật làm việc với model binding cơ bản trong Razor Pages. Các kỹ thuật này giúp bạn dễ dàng trích xuất dữ liệu từ các truy vấn và gán vào property hoặc biến cục bộ.

Model binding trong Razor Pages còn có nhiều ứng dụng mạnh mẽ (và phức tạp hơn). Chúng ta sẽ còn quay trở lại với đề tài model binding nâng cao trong một bài học riêng.

Tải mã nguồn Model Binding

1 file(s) 0.00 KB

TẢI MÃ NGUỒN

+ 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!