Razor Pages: giới thiệu, cài đặt, project, page, URL

Razor Pages là một trong các framework dành cho phát triển ứng dụng web xây dựng trên ASP.NET Core. Razor Pages thể hiện cách tiếp cận và mô hình lập trình riêng trong phát triển ứng dụng web: lấy trang làm trung tâm (page centric), vận dụng mô hình MVVM trong web.

Microsoft cũng khuyến nghị người mới học ASP.NET Core nên bắt đầu với Razor Pages do tính đơn giản, dễ học và có nhiều khái niệm tương đồng sẽ được sử dụng tiếp trong các framework còn lại của ASP.NET Core (MVC, Web API).

Bài học này sẽ đưa ra những vấn đề cơ bản nhất giúp bạn hiểu rõ hơn về Razor Pages.

NỘI DUNG CỦA BÀI [Ẩn]

Razor Pages là gì?

Razor Pages là một trong các framework dành cho xây dựng ứng dụng web bên trên ASP.NET Core. Razor Pages cho phép trộn HTML và C# (gọi là cú pháp Razor) vào cùng một file (có đuôi cshtml) để dễ dàng tạo ra HTML theo logic của chương trình. Nhờ vậy dữ liệu HTML tạo ra "động" chứ không cố định như ở các trang web "tĩnh" thiết kế sẵn.

Razor pages được xây dựng dựa trên tư tưởng "Page centric" – lấy "trang" (page) làm trung tâm của việc tổ chức file và mã nguồn của dự án. Khi trình duyệt truy xuất trang (thực chất là file cshtml), mã C# tương ứng trên trang sẽ thực thi để tạo ra dữ liệu HTML trả về cho trình duyệt.

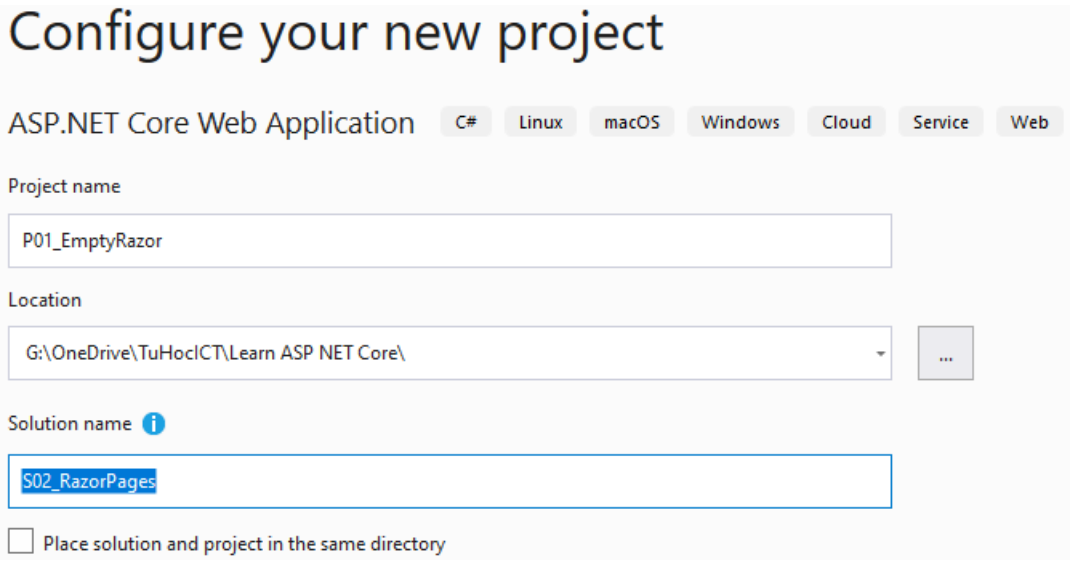
Nếu bạn đã từng quen thuộc với PHP hoặc ASP cổ điển thì bạn cũng có thể hình dung Razor Pages tương tự như vậy. Khác biệt là giờ đây bạn dùng ngôn ngữ lập trình C# (kết hợp với html) để tạo ra nội dung "động" cho trang web.

Để dễ dàng bắt đầu với Razor Pages, chúng ta lần lượt thực hiện các bài thực hành.

Thực hành 1: Cấu hình dự án Razor Pages

Trong bài thực hành này bạn sẽ tạo một project trống rồi lần lượt thực hiện các thao tác cấu hình để tạo ra một ứng dụng Razor Pages đơn giản. Bài thực hành này sẽ giúp bạn hiểu cấu trúc cơ bản của một ứng dụng Razor Pages.

Bước 1. Tạo một project trống



Bước 2. Cấu hình sử dụng service

Ứng dụng Razor Pages yêu cầu sử dụng service riêng bằng cách gọi phương thức `AddRazorPages` trong `ConfigureServices`. Mở file `Startup.cs`, tìm phương thức `ConfigureServices` và viết code lại như sau:

```
8.  public void ConfigureServices(IServiceCollection services) {
9.      services.AddRazorPages();
10. }
```

Bước 3. Cấu hình sử dụng middleware

Ứng dụng Razor Pages yêu cầu sử dụng middleware Routing và MapRazorPages endpoint. Mở file `Startup.cs`, tìm phương thức `Configure` và điều chỉnh code như sau:

```
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.MapRazorPages();
21.      });
22.  }
```

Bước 4. Tạo page cshtml

Bạn hãy tạo thư mục Pages trực thuộc project. Tất cả các trang về sau sẽ được tạo ra bên trong thư mục này. Nếu muốn bạn cũng có thể tạo luôn thư mục wwwroot.

Trong thư mục này tạo tiếp *file văn bản* About.cshtml như sau:

Ở bước này bạn lưu ý tạm thời không sử dụng các Razor template của Visual Studio mà chỉ sử dụng file text.

> **Lưu ý**:
> (1) File cshtml trong Razor Pages được gọi là trang nội dung (content page), thường gọi ngắn gọn là trang (page). Trong một số trường hợp, file cshtml không phải là trang nội dung. Chúng ta sẽ gặp trang riêng phần (partial page) hoặc view component – những file cshtml nhưng không phải là trang nội dung.
> (2) Tên trang nội dung cshtml không được chứa ký tự gạch chân (_) ở đầu. Ký tự _ chỉ dùng trong một số trường hợp đặc biệt (như _Layout, _ViewStart, _ViewImports) mà chúng ta sẽ xem xét sau.
> (3) Tất cả trang nội dung phải nằm trong thư mục Pages. Khi nằm ngoài thư mục này, file cshtml sẽ không được xử lý bởi Razor Pages.

Bước 5. Viết code cho About.cshtml

Mở file About.cshtml và viết code như sau:

```
@page
@{
    Layout = null;
    ViewData["Title"] = "About";
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewData["Title"] - Tự học ICT</title>
    <style>
        .header {
            color: cadetblue;
        }

        .footer {
            text-align: center;
            font-size: small;
        }
    </style>
</head>
<body>
    <h1 class="header">Razor Pages tutorial</h1>
    <hr />

    <h2>@SayHello("Covid")</h2>
    <p>It's @DateTime.Now.ToLongDateString()</p>

    <p>ASP.NET Core Razor Pages is a page-focused framework for building dynamic, dat

    <hr />
    <h2 class="footer">©tuhocict.com - @DateTime.Now.Year</h2>
</body>
</html>

@{
    string SayHello(string name) {
        return $"Hello, {name}. Welcome to Razor Pages!";
    }
}
```

Code C# ở đây đều rất đơn giản và cơ bản. Riêng object ViewData có thể hơi lạ một chút. Chúng ta sẽ quay lại giải thích chi tiết về object này trong một bài học sau.

> **Lưu ý**:
> (1) Trang nội dung bắt buộc phải có @page ở dòng đầu tiên. Thiếu dòng này page sẽ không hoạt động. @page được gọi là directive.
> (2) Bạn cũng có thể thấy, trong trang nội dung trộn lẫn code C# và markup của HTML. Trong đó, code C# được đặt bên trong khối @{ }. Ngoài khối này đều là mã HTML. Loại cú pháp sử dụng trong file này gọi là cú pháp Razor.

Bạn cũng để ý một số dòng nơi code C# và HTML trộn lẫn với nhau. Chúng ta sẽ dành hẳn một bài học riêng để tìm hiểu chi tiết về cú pháp Razor này.

Bước 6. Chạy thử ứng dụng

Hãy chạy thử ứng dụng trên, bạn sẽ thu được kết quả như sau:

Kết quả chạy chương trình Razor Pages đầu tiên

> Lưu ý bạn phải chỉ rõ page cần tải là **/about** (không cần đuôi cshtml).

Như vậy, qua bài thực hành đầu tiên bạn đã xây dựng được một ứng dụng Razor Pages đơn giản nhất từ một project trống. Qua đây bạn đã có được trải nghiệm đơn giản đầu tiên với Razor Pages.

Thực hành 2: Tạo page mới và trang mặc định

Trong bài thực hành này bạn sẽ tạo thêm một số trang cho ứng dụng đồng thời cấu hình trang mặc định cho site. Bạn sẽ tiếp tục sử dụng project đã tạo ở bài 1.

Bước 1. Thêm trang *Index.cshtml* vào thư mục Pages

Chọn "Razor Page"

Nhập tên page nhưng bỏ đánh dấu các ô bên cạnh

> Tạm thời bạn bỏ chọn các mục trong hộp thoại tạo page. Chúng ta sẽ giải thích chi tiết ý nghĩa các mục chọn này ở các bài học tương ứng.

Qua bước này bạn có thể thấy, template Razor Page của Visual Studio thực chất cũng chỉ là một file văn bản thông thường. Từ giờ bạn có thể sử dụng cách thức này để thêm page mới một cách nhanh chóng.

Bước 2. Điều chỉnh nội dung của Index.cshtml như sau:

```
@page
@{
    Layout = null;
    ViewData["Title"] = "Index";
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewData["Title"] - Tự học ICT</title>
    <style>
        .header {
            color: cadetblue;
        }

        .footer {
            text-align: center;
            font-size: small;
        }
    </style>
</head>
<body>
    <h1 class="header">Razor Pages tutorial</h1>

    <hr />
```

```
28.         <p>This site is dedicated to helping developers who want to use the ASP.NET Core
29.
30.         <hr />
31.         <h2 class="footer">©tuhocict.com - @DateTime.Now.Year</h2>
32.     </body>
33.     </html>
```

Bước 3. Chạy lại chương trình mà không chỉ định rõ trang cần tải bạn sẽ thấy trang Index.cshtml đã được tải vào trình duyệt.

> Razor Pages quy ước sử dụng Index.cshtml làm **trang mặc định**. Nghĩa là nếu trong thư mục Pages có file Index.cshtml thì khi bạn bỏ qua tên trang khi nhập địa chỉ trên trình duyệt, Razor Pages sẽ tự động tìm đến file Index.cshtml và xử lý trang này gửi cho trình duyệt.

Thực hành 3: Cấu trúc thư mục và URL

Bài thực hành này sẽ giúp bạn hiểu sự tương quan giữa cấu trúc thư mục của ứng dụng Razor Pages với cấu trúc URL của site.

Bước 1. Tạo thư mục con Admin của Pages

Bước 2. Tạo thêm hai file Index.cshtml và Login.cshtml trong thư mục Admin

Bạn thu được cấu trúc như sau

Bước 3. Lần lượt điều chỉnh nội dung của Login.cshtml và Index.cshtml như sau:

Login.cshtml | Index.cshtml

```
@page
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Login</title>
</head>
<body>
    <h1>This is the Admin Login page</h1>
</body>
</html>
```

Bước 3. Chạy chương trình với URL như sau:

> Như vậy, bạn có thể thấy sự tương quan giữa cấu trúc thư mục Pages và Url của các page như sau:
> (1) Pages là một thư mục đặc biệt. Tất cả các file cshtml phải nằm trong Pages hoặc thư mục con của Pages thì mới được Razor Pages xử lý.
> (2) Root Url **/** tương ứng với file **Index.cshtml** trong thư mục Pages.
> (3) Nếu trong thư mục con của Pages có chứa file Index.cshtml thì nó cũng sẽ được sử dụng làm file mặc định của thư mục đó. Trong ví dụ trên, URL **/admin** sẽ tương ứng với file **Pages/Admin/Index.cshtml**.
> (4) Tên file trong thư mục (không tính phần mở rộng cshtml) tương ứng với file của URL. Trong ví dụ trên, file **Pages/Admin/Login.cshtml** tương ứng với URL **/admin/login**.
> Sự tương quan giữa URL và cấu trúc file/thư mục này được gọi là **routing**.
>
> Sau này bạn sẽ học cách cấu hình chọn thư mục thay thế cho Pages, cũng như học về Area và cấu trúc URL tương ứng.

Thực hành 4: Sử dụng mẫu project Web Application

Tất cả những thao tác chúng ta đã thực hiện trong 3 bài thực hành trên gần như sẽ phải lặp lại mỗi khi tạo project mới.

Ngoài ra, hầu như mọi project cho ứng dụng web đều sử dụng một số thư viện css và javascript thông dụng như BootStrap, jQuery.

Để tiện lợi, ASP.NET Core tạo sẵn một template cho ứng dụng Razor Pages. Bạn có thể sử dụng template này trong Visual Studio hoặc từ giao diện dòng lệnh. Template này có tên gọi **Web Application** trong Visual Studio. Trong .NET Core CLI, template này có tên gọi là **webapp**. Đây cũng là template mặc định cho ứng dụng Asp.net Core.

Option1. Sử dụng Visual Studio

Bước 1. Thêm project Asp.net core vào solution

Bước 2. Lựa chọn template Web Application

Option 2. Sử dụng giao diện dòng lệnh

Bạn có thể sử dụng CLI từ Command Prompt hoặc PowerShell của windows:

1. Chuyển tới thư mục nơi sẽ tạo project;
2. Gõ lệnh `dotnet new webapp -o P02_RazorApp` và ấn Enter;
3. Chờ .NET Core hoàn thành tạo project.

Lệnh `dotnet new webapp -o <thư mục>` sẽ tạo ra một project thuộc loại **webapp** (tương đương template *Web Application* trong Visual Studio) bên trong thư mục được chỉ định. Tên thư mục cũng sẽ là tên của project.

Dù sử dụng cách nào bạn cũng sẽ thu được project với cấu trúc như sau:

Chạy thử ứng dụng bạn thu được kết quả như sau:

Lưu ý:

- (1) Project đã cài đặt thư viện front-end như BootStrap, jQuery, jQuery validation.
- (2) File site.css và site.js trống đã được tạo sẵn và tích hợp vào site. Bạn có thể trực tiếp viết css hoặc javascript vào các file này.
- (3) Một file layout cơ bản có đầy đủ header, footer, navigation menu đã được tạo ra và sử dụng mặc định cho tất cả các page thông qua _ViewStart.
- (4) File _ViewImports với một số cấu hình mặc định đã tạo sẵn.
- (5) File _Layout được đặt trong thư mục đặc biệt Shared để tất cả page có thể dễ dàng truy xuất.

Như vậy, khi sử dụng template WebApp (Web Application) của ASP.NET Core bạn sẽ nhanh chóng thu được một project Razor App với các thành phần cơ bản. Từ project này bạn chỉ việc tiếp tục phát triển tiếp các thành phần theo yêu cầu của bài toán. Những cấu hình và thiết kế cơ bản phổ biến nhất đã được thực hiện sẵn.

Kết luận

Trong bài học này bạn đã bước đầu làm quen với những vấn đề cơ bản của nhất của ASP.NET Core Razor Pages. Bạn đã học cách tạo dự án Razor Pages, cách tạo page, cấu trúc file/thư mục tương ứng với URL.

Tải mã nguồn solution
S02_RazorPages
1 file(s) 5Mb

TẢI MÃ NGUỒN SOLUTION

+ 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!