

# Layout, ViewStart, section trong ASP.NET Core

Hướng dẫn tự học lập trình ASP.NET Core toàn tập > Layout, ViewStart, section trong ASP.NET Core

Layout, ViewStart là những file đặc biệt của Razor. Việc sử dụng các file này giúp đơn giản hóa quá trình làm việc với Razor Pages. Trong bài học này chúng ta sẽ tìm hiểu chi tiết vai trò của các file này trong project Razor Pages.

## NỘI DUNG CỦA BÀI [ Ấn ]

1. Khái niệm layout trong ASP.NET Core
2. Thực hành 1: Sử dụng layout trong Razor Pages
3. Thực hành 2: Tự động chọn layout trong Razor: file \_ViewStart.cshtml
4. Thực hành 3: Tạo Razor page với layout bằng Visual studio
5. Sử dụng section trong layout
6. Kết luận
- 6.1. Tải mã nguồn solution S02\_RazorPages

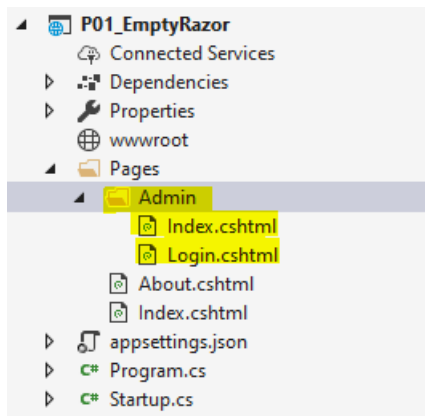
Trong bài học này chúng ta cũng tiếp cận vấn đề thông qua thực hành. Bạn tiếp tục sử dụng [project đã thực hiện](#) từ bài học trước.

## Khái niệm layout trong ASP.NET Core

Khi bạn duyệt các website hẳn đều để ý rằng các page trên cùng một site đều có một cấu trúc chung thống nhất. Về đại thể, cấu trúc này thường được chia thành các phần header, footer, navigation menu và phần nội dung chính. Đa số có thêm sidebar (phải | trái | hai bên). Cấu trúc chung thống nhất của các page trong cùng một site được gọi là **layout**.

Thêm vào đó, các file css và thư viện JavaScript thường được các page sử dụng chung. Ví dụ bộ thư viện css Bootstrap hay thư viện jQuery được rất nhiều site sử dụng và do đó được tham chiếu tới từ tất cả các page của một site.

Nhắc lại trong bài học trước bạn đã tạo project với các file và thư mục như sau:



Giờ hãy nhớ lại project chúng ta đã tạo ra trong bài học trước: hai page About.cshtml và Index.cshtml chỉ khác biệt nhau về một phần nội dung hiển thị. Những phần khác giống hệt nhau. Tuy nhiên chúng ta đang phải lặp lại code trong hai page này.

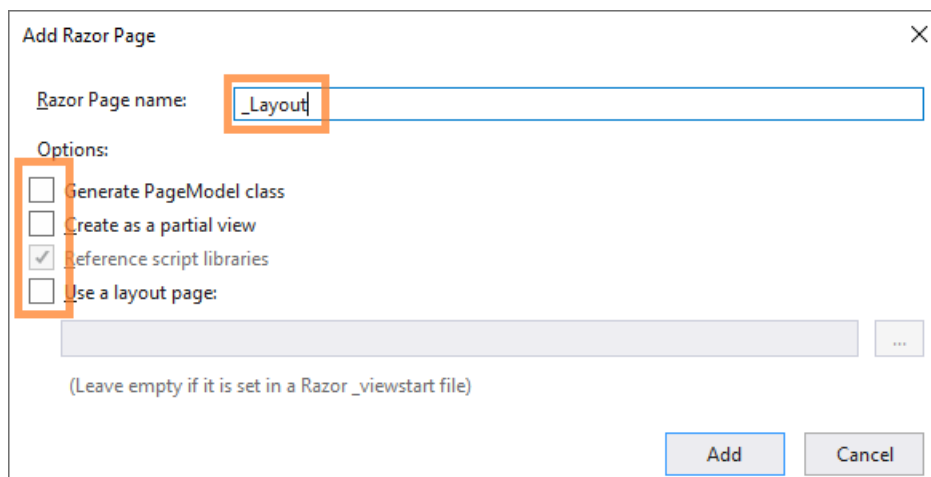
```
Index.cshtml About.cshtml
1. @page
2. @{
3.     Layout = null;
4.     ViewData["Title"] = "Index";
5. }
6.
7. <!DOCTYPE html>
8.
9. <html>
10. <head>
11.     <meta name="viewport" content="width=device-width" />
12.     <title>@ViewData["Title"] - Tự học ICT</title>
13.     <style>
14.         .header {
15.             color: cadetblue;
16.         }
17.
18.         .footer {
19.             text-align: center;
20.             font-size: small;
21.         }
22.     </style>
23. </head>
24. <body>
25.     <h1 class="header">Razor Pages tutorial</h1>
26.
27.     <hr />
28.     <p>This site is dedicated to helping developers who want to use the ASP.NET Core Raz
29.
30.     <hr />
31.     <h2 class="footer">@tuhocict.com - @DateTime.Now.Year</h2>
32. </body>
33. </html>
```

Từ đây phát sinh vấn đề phải tạo ra một cái khung chung thống nhất mà tất cả các page của một site đều sử dụng để tránh việc phải lặp đi lặp lại các đoạn code HTML, CSS và JavaScript.

Razor page cung cấp một cơ chế để thực hiện: Layout.

## Thực hành 1: Sử dụng layout trong Razor Pages

**Bước 1.** Tạo file \_Layout.cshtml trong thư mục Pages



Lưu ý dấu gạch chân trong tên gọi. Bạn có thể đặt tên là \_Layout hay bất kỳ tên nào cũng được.

### Quy ước đặt tên file

Dấu gạch chân trước tên gọi của file thường dùng cho một số loại file đặc biệt trong Razor Pages để phân biệt nó với trang nội dung bình thường.

Các trang đặc biệt như `_Layout`, `_ViewImports`, `_ViewStart` đều không chứa `@page`, và do đó trình duyệt sẽ không thể trực tiếp truy xuất được các file này.

Quy ước đặt tên này còn dùng cho một số loại trang đặc biệt nữa như [Partial Pages](#) và [View Component](#) (sẽ học trong một bài khác). Các loại trang đặc biệt này được sử dụng bởi các page khác chứ không được truy xuất trực tiếp từ trình duyệt.

### Bước 2. Nhập nội dung cho file `_Layout.cshtml` như sau:

```
1. <!DOCTYPE html>
2.
3. <html>
4. <head>
5.   <meta name="viewport" content="width=device-width" />
6.   <title>@ViewData["Title"] - Tự học ICT</title>
7.   <style>
8.     .header {
9.       color: cadetblue;
10.    }
11.
12.    .footer {
13.      text-align: center;
14.      font-size: small;
15.    }
16.  </style>
17. </head>
18. <body>
19.   <h1 class="header">Razor Pages tutorial</h1>
20.   <hr />
21.
22.   @RenderBody()
23.
24.   <hr />
25.   <h2 class="footer">@tuhocict.com - @DateTime.Now.Year</h2>
26. </body>
27. </html>
```

Hẳn bạn đã dễ dàng nhận ra, nội dung của `_Layout.cshtml` thực chất chính là nội dung chung của `Index.cshtml` và `About.cshtml`. Phần giữa hai thẻ `<ht/>` có lời gọi phương thức `@RenderBody()`.

### Phương thức `RenderBody()`

`RenderBody()` là phương thức đặc biệt trong Razor Pages có nhiệm vụ xuất ra trang nội dung ở vị trí tương ứng trong trang layout. Nói cách khác, nếu hiểu layout là một cái khung, các file còn lại chỉ cung cấp nội dung riêng, thì nội dung riêng của từng file sẽ xuất hiện ở đúng vị trí gọi phương thức `RenderBody()`.

Nếu thiếu lời gọi `RenderBody()`, trang layout sẽ trở thành vô dụng.

### Bước 3. Thay đổi nội dung các page để sử dụng layout vừa tạo

Index.cshtml

About.cshtml

Admin/Index.cshtml

Admin/Login.cshtml

```

1. @page
2. @{
3.     Layout = "_Layout";
4.     ViewData["Title"] = "Index";
5. }
6.
7. <p>This site is dedicated to helping developers who want to use the ASP.NET Core Razor P

```

Hẳn bạn dễ dàng nhận thấy, giờ đây một page chỉ còn chứa những nội dung riêng biệt của mình. Những phần chung nhất đã được đẩy sang file `_Layout.cshtml`.

### Lệnh Layout

Để chỉ định file layout để hiển thị nội dung của page, bạn gọi lệnh **Layout = "\_Layout"** trong khối code C# ở đầu page. Chú ý rằng bạn chỉ cần cung cấp tên file layout đặt trong dấu ngoặc kép nhưng không cần viết phần mở rộng cshtml.

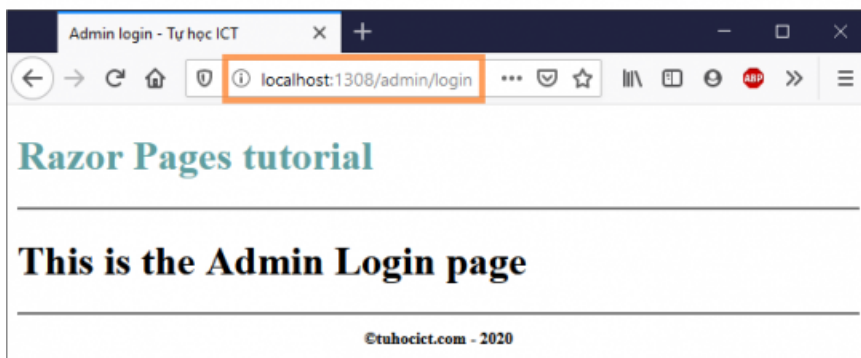
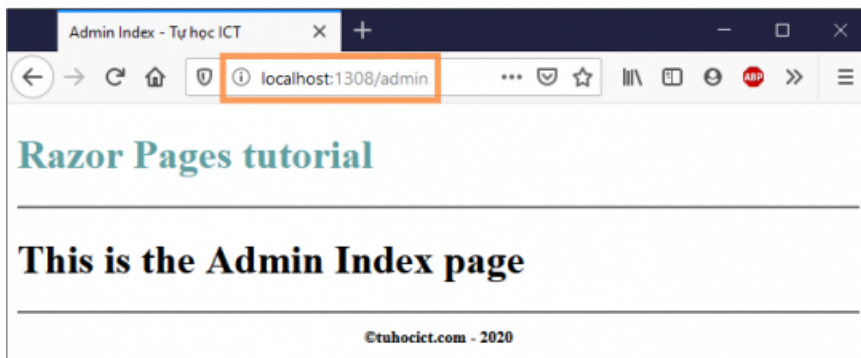
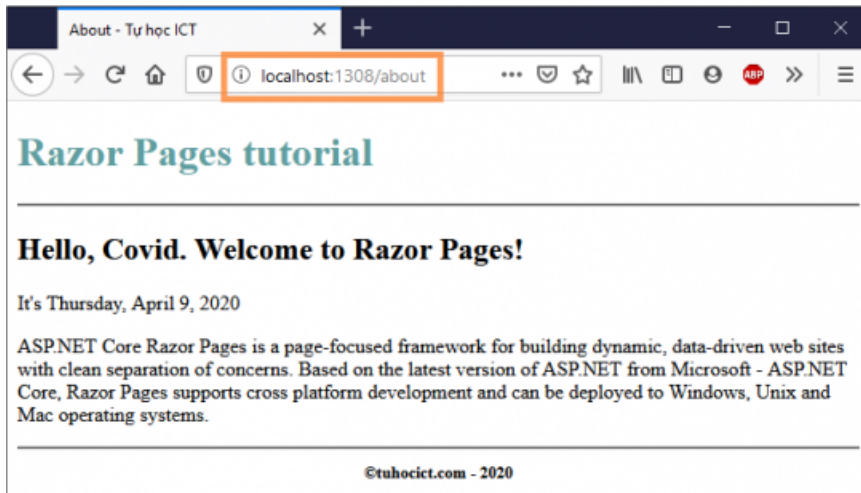
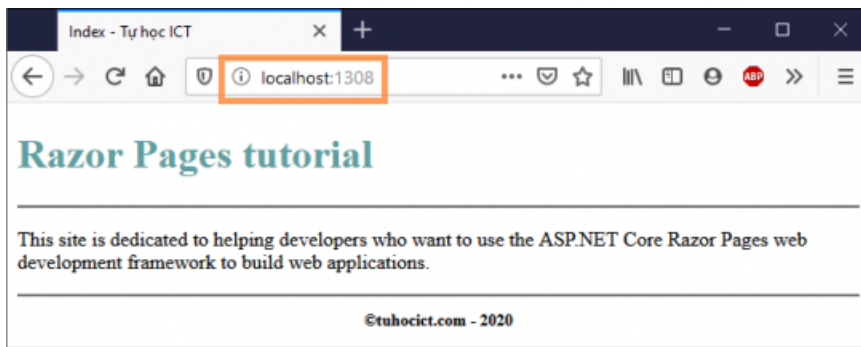
Nếu page nào đó bạn không muốn sử dụng layout thì chỉ cần viết `Layout = null` trong khối code đầu page. Nếu muốn sử dụng một layout khác, bạn có thể thay thế tên layout vào đây.

### ViewData object

**ViewData** là một object đặc biệt thuộc kiểu Dictionary được Razor Pages tạo sẵn và có thể truy cập từ bất kỳ đâu trong ứng dụng Razor Pages. Nhiệm vụ của nó là lưu trữ các cặp khóa / giá trị để sử dụng xuyên suốt trong tất cả các page của site. Trong trường hợp này ViewData được dùng để trao đổi giá trị dòng title của từng page với layout: Giá trị dòng title được gán ở từng page và hiển thị trên layout.

Ngoài ViewData, trong một số tài liệu bạn còn có thể gặp object **ViewBag** với cùng mục đích. ViewBag không được khuyến khích sử dụng do nó có kiểu dynamic – kiểu dữ liệu có hiệu suất thấp và làm mất tính chất strong typing của C#. Vì vậy trong bài giảng này chúng ta không đề cập đến ViewBag.

**Bước 4.** Chạy ứng dụng và thử truy cập vào các URL bạn thu được kết quả như sau:



Bạn có thể dễ dàng nhận thấy giờ tất cả các page đã có cùng cấu trúc (layout)

Chú ý phần port trong URL của bạn có thể không giống trong ví dụ này. Nếu chạy ở dạng console, port là 5000. Nếu chạy qua IIS, port sẽ được chọn bất kỳ.

### Thư mục Shared

Khi đọc các tài liệu khác bạn có thể gặp trường hợp file `_Layout.cshtml` được đặt trong thư mục con Shared của Pages. Khi này, chỉ định sử dụng layout trong mỗi page vẫn giữ nguyên là `Layout = "_Layout"`.

Lý do là Shared là một thư mục đặc biệt trong Razor Pages. Tất cả các file đặt trong thư mục này được truy xuất bởi bất kỳ page nào mà không cần chỉ rõ đường dẫn. Hay nói cách khác, bất kỳ page nào cũng có thể truy xuất các file đặt trong thư mục Shared mà không cần chỉ rõ đường dẫn.

Do đó, Shared thường dùng để chứa các file dùng chung của toàn site như `_Layout`.

## Thực hành 2: Tự động chọn layout trong Razor: file `_ViewStart.cshtml`

Trong phần thực hành trên bạn đã sử dụng layout cho các razor page. Tuy nhiên, vẫn có một vấn đề: mỗi page đều phải chỉ định rõ file layout cần sử dụng.

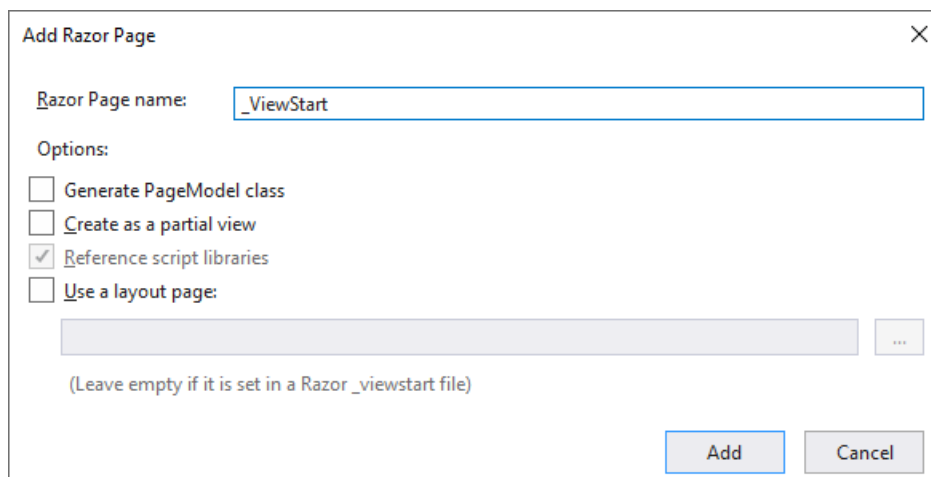
Hãy tưởng tượng tình huống site có hàng trăm page, và bạn muốn thay một file layout khác (có thể chỉ để thử nghiệm). Để thay đổi trên hàng trăm page, bạn phải mở từng page để chỉnh sửa dòng code `Layout = ...`.

Trong tình huống này Razor Pages cung cấp một công cụ: ViewStart.

View start là một file đặc biệt chứa những đoạn code Razor sẽ được thực hiện trước khi xử lý mỗi page.

Do vậy bạn có thể lợi dụng ViewStart để tự động đặt layout mặc định cho tất cả các page.

**Bước 1.** Tạo file `_ViewStart.cshtml` trong thư mục Pages:



Add Razor Page

Razor Page name:

Options:

- ☐ Generate PageModel class
- ☐ Create as a partial view
- ☒ Reference script libraries
- ☐ Use a layout page:

...

(Leave empty if it is set in a Razor \_viewstart file)

Add Cancel

Lưu ý rằng tên file bắt buộc phải là `_ViewStart.cshtml`. Đây là tên file đặc biệt do ASP.NET Core quy định. Nếu viết sai tên, ASP.NET Core sẽ không chấp nhận.

**Bước 2.** Viết code cho `_ViewStart` như sau:

```
1. @{  
2.     Layout = "_Layout";  
3. }
```

Bạn có thể thấy ngay nó chỉ là lệnh gán layout thông thường.

**Bước 3.** Xóa bỏ lệnh `Layout = "_Layout"` hoặc `Layout = null` trong tất cả các page.

**Bước 4.** Chạy thử chương trình, bạn sẽ thu được cùng một kết quả như phần thực hành 1. Tất cả các file đều sử dụng chung `_Layout.cshtml`.

Như vậy, nếu giờ đây bạn muốn thử nghiệm một layout mới thì chỉ cần thay thế nó trong `_ViewStart` chứ không cần thay trong từng page. `_ViewStart` là cơ chế chạy code chung trước khi xử lý page.

#### Phạm vi tác dụng của `_ViewStart`

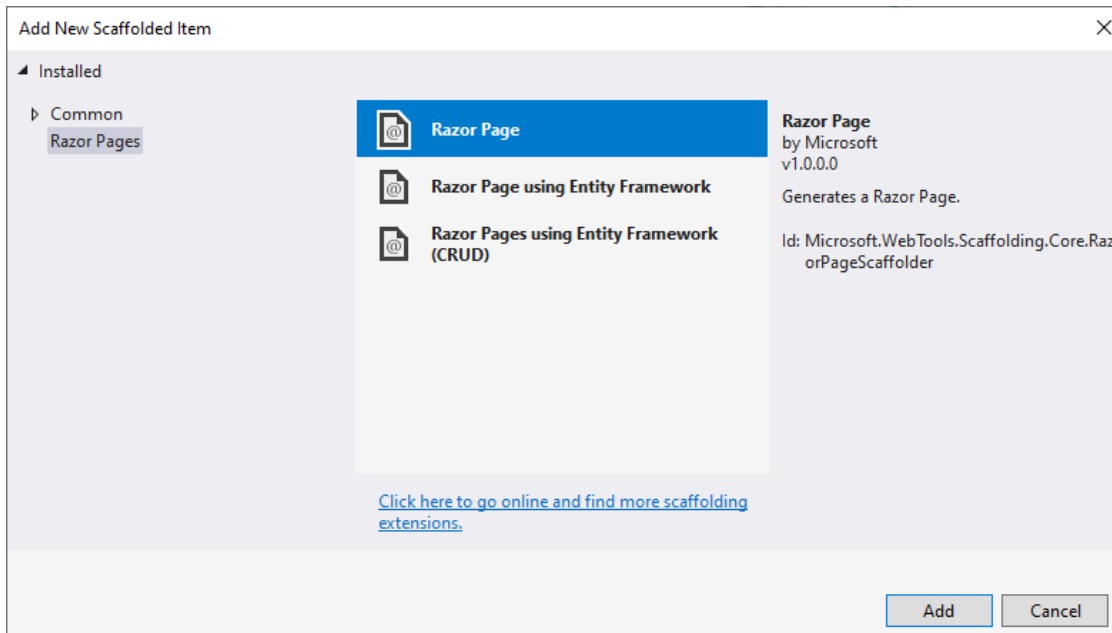
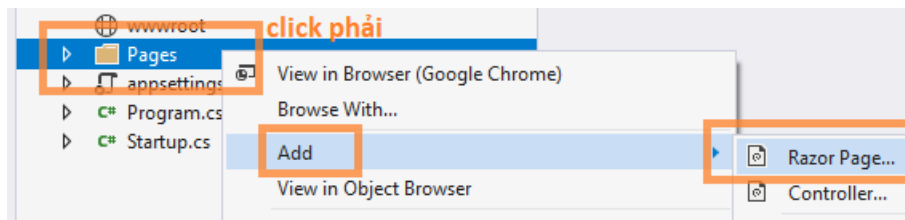
- (1) `_ViewStart` có tác dụng trong thư mục chứa nó và tất cả các thư mục con. Nếu bạn để `_ViewStart` trong Pages, nó có tác dụng trên toàn bộ site.
- (2) Nếu bạn đặt `_ViewStart` trong một thư mục con của Pages, ví dụ trong thư mục Admin, `_ViewStart` chỉ có tác dụng trong thư mục con này (và các thư mục con của thư mục đó). Khi này các page ở bên ngoài thư mục Admin sẽ không nhận layout tự động (do `_ViewStart` này không có tác dụng bên ngoài Admin).
- (3) `_ViewStart` trong thư mục con sẽ có tác dụng đè lên `_ViewStart` ở thư mục cha. Ví dụ, nếu thư mục Pages có thư mục con Admin và cả hai thư mục này đều có file `_ViewStart` thì các page trong Admin (và các thư mục con của Admin) sẽ chịu tác dụng của `_ViewStart` đặt trong Admin.

ASP.NET Core cũng có một cơ chế xử lý code chung nữa là `_ViewImports`. Tuy nhiên, `_ViewImports` chỉ chạy được các directive `@using` và `@addTagHelper` chứ không chạy được lệnh Razor như `_ViewStart`. Do đó, `_ViewImports` được sử dụng để chỉ định các thư viện C# dùng chung trong toàn project. Bạn sẽ học chi tiết về `_ViewImports` khi học [cấu trúc điều khiển của Razor](#).

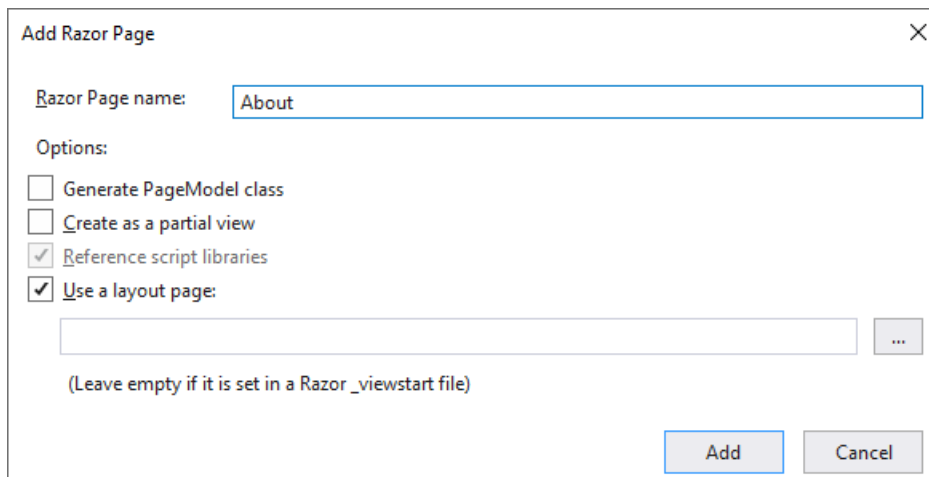
## Thực hành 3: Tạo Razor page với layout bằng Visual studio

Một khi đã tạo xong file `_Layout` và `_ViewStart`, khi sử dụng hộp thoại Add Razor Page của Visual Studio bạn có thể lựa chọn layout muốn sử dụng.

**Bước 1.** Mở hộp thoại tạo Add Razor Page



**Bước 2.** Trong hộp thoại Add Razor Page chọn tên cho page và đánh dấu chọn **"Use a layout page"**



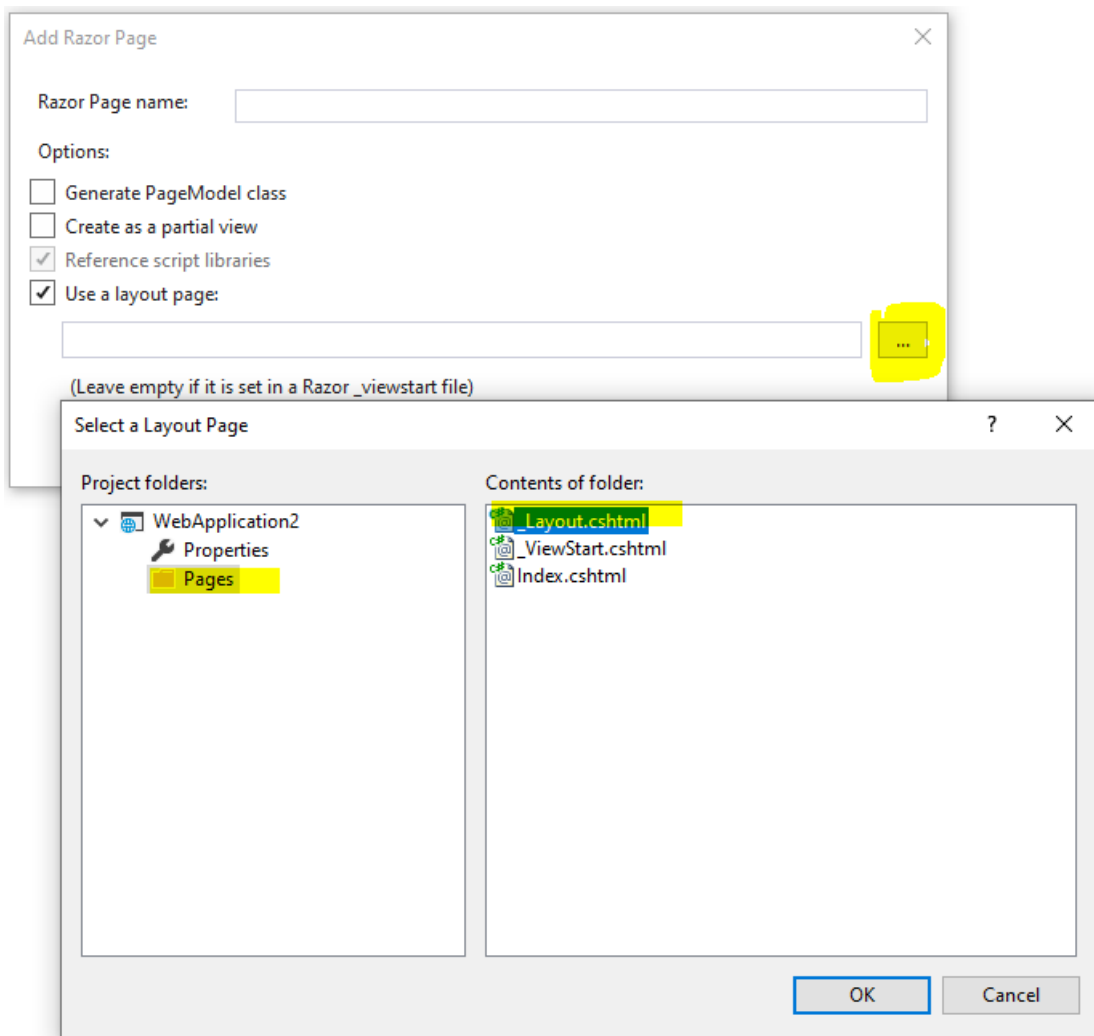
Đánh dấu chọn "Use a layout page"

Khi đánh dấu này bạn có hai lựa chọn:

**Option 1:** Nếu bạn để trống textbox, Razor sẽ tự động sử dụng layout bạn đã cấu hình trong `_ViewStart.cshtml` (dĩ nhiên với điều kiện là bạn đã cấu hình chọn layout tự động trong file `_ViewStart.cshtml`).

**Option 2:** Bạn có thể lựa chọn file layout nếu bạn có nhiều file layout khác nhau:





## Sử dụng section trong layout

Khi sử dụng layout, bạn dễ dàng đưa được nội dung của một trang vào layout nơi đặt `RenderBody()`.

Nhưng giờ hãy tưởng tượng tình huống bạn sử dụng một layout với sidebar bên tay phải. Bạn viết một số trang nội dung khác nhau. Mỗi trang nội dung sẽ có một phần xuất hiện ở layout trong vị trí `RenderBody()` và một phần nội dung phải xuất hiện ở sidebar.

Tức là, yêu cầu của bạn giờ đây là phải hiển thị (render) nội dung ở nhiều nơi khác nhau trong layout, chứ không phải chỉ render trang nội dung ở một nơi duy nhất trong layout.

Lấy một ví dụ khác. Các site thường phải sử dụng javascript. Với các thư viện mà tất cả các page sử dụng chung, bạn có thể đặt nó trong layout. Tuy nhiên, một vài page có thể sử dụng khối `<script></script>` riêng biệt. Có thể đó chỉ là vài dòng javascript đặc biệt cho giao diện.

HTML khuyến nghị rằng mã javascript nên đặt ở cuối thân, ngay trước thẻ `</body>`.

Vậy làm thế nào xây dựng một layout đảm bảo được cả hai điều kiện: có thể render khối `<script></script>` ngay trước thẻ đóng `</body>`; khối script này chỉ render đối với một số

page riêng (chứ không áp dụng cho tất cả page).

ASP.NET Core Razor Pages sử dụng cơ chế **section** để hiển thị nội dung ở nhiều vị trí khác nhau trong layout. Cơ chế section cho phép tạo ra nhiều vị trí để chèn nội dung trên layout. “Nội dung” ở đây có thể hiểu là HTML, CSS hay JavaScript.

Để sử dụng section cần có sự phối hợp giữa layout và trang nội dung.

Ở **layout**, bạn sử dụng lệnh sau ở vị trí bạn muốn chèn nội dung vào:

```
1. @RenderSection("Tên_section", true|false)
```

Ở **trang nội dung**, bạn khai báo section với cấu trúc sau:

```
1. @section Tên_section {  
2.     <!-- nội dung viết ở đây -->  
3. }
```

Trong đó Tên\_section ở nơi khai báo (trên trang nội dung) và nơi sử dụng (trên layout) phải trùng khớp nhau.

Tham số thứ hai của RenderSection là một biến kiểu bool. Nếu có giá trị true thì bất kỳ trang nội dung nào sử dụng layout này đều phải khai báo section có tên tương ứng. Nếu có giá trị false thì trang nội dung **không** bắt buộc phải khai báo section.

Một trong những trường hợp phải sử dụng section là khi cần render khối code JavaScript.

Nếu bạn sử dụng template WebApplication khi tạo dự án, Razor Pages tạo sẵn cho bạn một file layout. Hãy mở file này ra và tìm đến vị trí ngay trước thẻ kết thúc </body>, bạn sẽ thấy một ví dụ về cách sử dụng section:

```
1. @RenderSection("Scripts", false)
```

Nếu ở một trang nội dung nào đó bạn cần sử dụng JavaScript, bạn chỉ cần định nghĩa một Scripts section riêng:

```
1. @section Scripts {  
2.     <script>  
3.         // mã javascript viết ở đây  
4.     </script>  
5. }
```

Khi này, khối javascript của bạn sẽ được chèn vào đúng vị trí @RenderSection("Scripts", false) trên layout, ngay trước khi kết thúc <body> – đây cũng là vị trí khuyến nghị để chèn javascript cho trang.

Nếu trang của bạn không có nhu cầu sử dụng javascript thì cũng không ảnh hưởng gì đến layout vì tham số thứ hai của RenderSection là false. Nghĩa là bạn không bắt buộc phải khai báo Scripts section trên trang nội dung sử dụng layout này.

## Kết luận

---

Trong bài học này bạn đã học một số khái niệm và kỹ thuật cấu hình cơ bản của Razor Pages giúp đơn giản hóa quá trình làm việc sau này, bao gồm sử dụng Layout, ViewStart và section. Bạn cũng làm quen với project template WebApp của ASP.NET Core.

Lưu ý rằng, các khái niệm và kỹ thuật sử dụng layout, view start và section trong bài học này sẽ được vận dụng nguyên vẹn khi bạn chuyển sang học [ASP.NET Core MVC](#). ASP.NET Core Razor Pages và ASP.NET Core MVC cùng sử dụng Razor view engine nên tất cả các khái niệm trên trong hai loại dự án thực tế là một.



[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!