Cú pháp Razor cơ bản, biểu thức và khối code

Razor là loại cú pháp kết hợp C# và HTML để sinh ra HTML động trong ứng dụng ASP.NET Core. Razor được sử dụng trong Razor Pages, MVC và Blazor. Đây là loại cú pháp rất quan trọng mà bạn bắt buộc phải biết khi học ASP.NET Core. Bài học sẽ giới thiệu chi tiết về loại cú pháp này.

> Mặc dù cú pháp Razor được giới thiệu chi tiết trong phần nội dung Razor Pages, bạn sẽ tiếp tục sử dụng nguyên vẹn các nội dung này khi học ASP.NET Core MVC và Blazor. Khi học ASP.NET Core MVC chúng ta sẽ không nhắc lại nội dung về Razor nữa.

NỘI DUNG CỦA BÀI [Ẩn]

Razor là gì?

Razor là tên gọi của view engine sử dụng bởi ASP.NET Core, đồng thời cũng là tên gọi của loại ngôn ngữ đánh dấu sử dụng trong view engine này.

Razor View Engine

Razor là tên gọi của view engine – cơ chế sinh ra mã HTML (từ nhiều nguồn gốc khác nhau) để trả lại cho trình duyệt – sử dụng trong ASP.NET Core. Razor view engine hoạt động dựa trên khả năng đọc loại mã kết hợp giữa C# và HTML.

Tại sao lại cần đến view engine?

Ngôn ngữ HTML chỉ có khả năng định dạng văn bản chứ không thể thực hiện các logic và điều khiển. Ở khía cạnh khác, các ngôn ngữ lập trình mặc dù có thể tạo ra HTML nhưng phải sử dụng cú pháp riêng của ngôn ngữ, vốn cồng kềnh và không tiện lợi.

Vì vậy, người ta tạo ra những view engine (cùng ngôn ngữ đánh dấu riêng của nó) với nhiệm vụ kết hợp ngôn ngữ lập trình với HTML để dễ dàng sinh ra HTML. Nói cách khác, view engine hoạt động như một chương trình dịch (compiler hoặc interpreter) để chuyển đổi loại ngôn ngữ lai này thành HTML.

Razor syntax/language

Tên gọi Razor cũng được dùng để chỉ loại cú pháp/ngôn ngữ đánh dấu (markup syntax/language) sử dụng trong Razor View Engine của ASP.NET Core. Cú pháp Razor kết hợp C# và HTML với nhiệm vụ "đánh dấu" xem đâu là C#, đâu là HTML để view engine xử lý cho phù hợp.

Từ một khía cạnh nào đó bạn cũng có thể hình dung Razor là một loại "ngôn ngữ lập trình" kiểu như PHP. Nếu bạn đã từng làm việc với PHP thì bạn sẽ hình dung ra ngay Razor. Nếu bạn thành thạo C# và HTML thì việc học Razor vô cùng dễ dàng. Bạn chỉ mất một buổi là có thể thành thạo Razor.

Razor là loại "ngôn ngữ" sử dụng trong các framework của ASP.NET Core như Razor Pages, MVC, Blazor. Cú pháp Razor sử dụng trong các framework này là như nhau. Do đó, nếu bạn học Razor ở đây, đến khi học MVC Framework hoặc Blazor, bạn sẽ không cần học lại Razor nữa.

File Razor có phần mở rộng là cshtml mà bạn đã làm quen từ những bài học trước.

> Cũng lưu ý rằng, Razor (với vai trò ngôn ngữ đánh dấu và view engine) đã xuất hiện từ ASP.NET MVC 3 (bên cạnh ASPX view engine đã có từ trước). Tuy nhiên Razor trong ASP.NET Core có nhiều khác biệt về tính năng/cấu trúc/cú pháp.

Giới thiệu chung về cú pháp Razor

Trong file cshtml (cũng gọi là razor file/page) có hai ngôn ngữ cùng tồn tại song song, C# và HTML. Vì vậy cú pháp Razor có nhiệm vụ phân định xem đâu là C# và đâu là HTML để View Engine có thể dịch và xử lý chính xác.

> Cũng vì lý do này bạn không cần học Razor như học một ngôn ngữ lập trình. Khi học Razor, bạn chỉ cần học cách diễn đạt (thông qua markup của Razor) đâu là C#, đâu là HTML.

Sau đây là ví dụ về một file razor đơn giản:

```
@page

@{ // đây là một code block
    // Tiêu đề trang
    ViewData["Title"] = "Razor syntax tutorial";

    var Name = "nCov"; // khai báo biến

    string SayHello(string name) => $"Hello in code block, {name}"; // khai báo phương t

    var countries = new[] { "USA", "Russia", "France", "GB" }; // khai báo mảng
```

```
12.
13.      void RenderName(string name) { // đây là một phương thức có cách chuyển đổi C# -> HT
14.          <p>Name: <strong>@name</strong></p>
15.      }
16.  }
17.
18.  <!-- Đây là thẻ h1 chứa biểu thức razor -->
19.  <h1>@SayHello(Name)</h1>
20.
21.  <!-- Đây là thẻ html <p> thông thường -->
22.  <p>Razor is a markup syntax for embedding server-based code into webpages. The Razor syn
23.
24.  <!-- Thẻ ul chứa cấu trúc điều khiển foreach sinh ra các thẻ li tương ứng -->
25.  <ul>
26.      @foreach (var c in countries) {
27.          <li>@c</li>
28.      }
29.  </ul>
```

Trong đó,

`@page` ở đầu file được gọi là **directive**.

```
1.  @{
2.      ViewData["Title"] = "Razor syntax tutorial";
3.      var Name = "nCov";
4.      string SayHello(string name) => $"Hello in code block, {name}";
5.  }
```

là một **code block** (khối code), chứa mã C# tiêu chuẩn.

`@SayHello(Name)` là một **biểu thức**, là một mã C# nhưng có trả lại kết quả. Biểu thức có thể nằm trong thẻ HTML.

<h1>, <p>, <ul>, <li> là các thẻ HTML thông thường.

```
1.  @foreach (var c in countries) {
2.      <li>@c</li>
3.  }
```

là **cấu trúc điều khiển** lặp foreach thông thường của C# kết hợp với thẻ html.

Bạn chưa cần hiểu và nhớ những khái niệm trên. Chúng ta đưa ra chỉ để bạn có hình dung đại khái về Razor. Chúng ta sẽ đi vào chi tiết trong các phần tiếp theo.

> Nhìn chung bạn cần nhớ các nguyên tắc cơ bản sau đây:
>
> (1) **Ngôn ngữ mặc định trong file cshtml là HTML**: Nghĩa là nếu không có đánh dấu gì đặc biệt thì Razor (view engine) sẽ hiểu nội dung bạn trình bày là ngôn ngữ HTML, và sẽ chuyển thẳng khối HTML này thành kết quả đầu ra.
>
> (2) **Ký tự @ yêu cầu Razor chuyển từ HTML sang C#**: Nghĩa là, nếu nhìn thấy ký tự @, Razor sẽ hiểu là bạn đang muốn viết code C# và chuyển sang chế độ dịch C#.
>
> (3) **Kết quả** dịch mã cuối cùng của file Razor (cshtml) là **mã HTML**. Tất cả những gì bạn viết trên page đều hướng tới sinh ra HTML (để trả lại cho trình duyệt).

> Ngoài ra bạn cũng ghi nhớ một số vấn đề sau:
> (1) **@ escape**: Viết hai lần ký tự @ (tức là **@@**) để diễn đạt cho chính chữ @ trong mã Razor (chứ không phải chuyển đổi sang C#);
> (2) Razor có khả năng tự phân biệt ký tự **@ nằm trong địa chỉ email**;
> (3) Cách ghi **chú thích phụ thuộc vào ngôn ngữ**: nếu trong vùng code C# thì ghi chú thích kiểu C#, nếu trong vùng mã HTML thì sử dụng cách ghi chú thích của HTML
> (4) Ký tự @ khi đi cùng một số từ đặc biệt (được Razor sử dụng cho mục đích riêng) gọi là các **directive**, ví dụ @page, @model, @using.

Nếu quen thuộc với PHP thì bạn có lẽ sẽ cảm thấy Razor hơi rắc rối hơi một chút khi chuyển đổi ngôn ngữ. PHP sử dụng cặp thẻ <?php ?> nên vùng mã PHP dễ phân tách khỏi HTML.

Tuy nhiên, khi nắm vững các nguyên tắc của Razor (sẽ học trong bài này và bài tiếp theo) bạn sẽ thấy rằng trong Razor, code C# và HTML kết hợp tự nhiên hơn trong PHP.

Biểu thức Razor

Khi học Razor, bạn cần nắm hai khái niệm quan trọng nhất: **Biểu thức** và **Khối code**. Chúng ta sẽ lần lượt xem xét chi tiết.

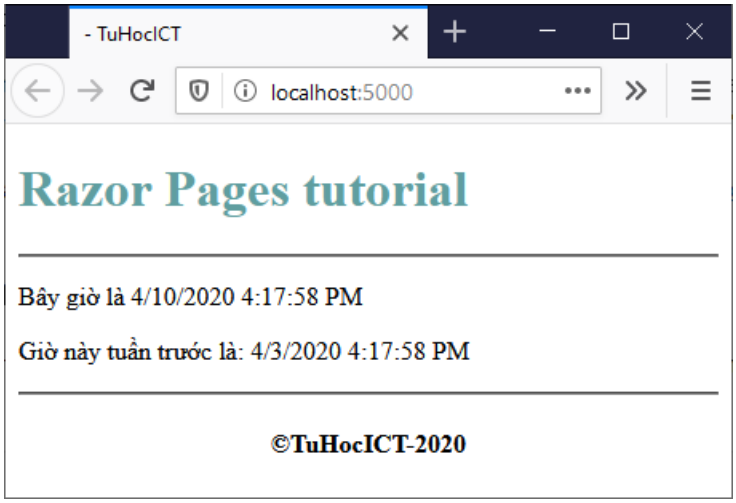
Khái niệm biểu thức Razor

Biểu thức Razor là *biểu thức của C#* được ghép trực tiếp cùng mã HTML thông qua ký tự @. Biểu thức Razor có tác dụng chèn giá trị (tính bằng C#) vào vị trí tương ứng của HTML.

Điều này có nghĩa là, bạn sử dụng biểu thức Razor khi đang code ở vùng HTML và cần chèn giá trị (tính bằng C#) vào một vị trí.

Ví dụ:

```
1.  <p>Bây giờ là @DateTime.Now</p>
2.  <p>Giờ này tuần trước là: @(DateTime.Now - TimeSpan.FromDays(7))</p>
```



> Để thử nghiệm code bạn có thể tự tạo ra một project Razor Pages trống. Link tải mã nguồn ví dụ ở cuối bài.

Nhắc lại: **biểu thức** (expression) trong C# là những lệnh có trả về kết quả. Đó có thể là lời gọi phương thức (method call) hoặc kết quả thực hiện các phép toán (operator). Những phương thức không trả về kết quả (còn gọi là các statement – lệnh) không thể sử dụng làm biểu thức Razor.

Đây là dạng thức đơn giản nhất trong lồng ghép C# với HTML: giá trị được tính ra bởi biểu thức C# và chèn vào vị trí tương ứng của mã HTML.

Biểu thức rõ và biểu thức ẩn

Biểu thức Razor có hai dạng: *biểu thức rõ* (explicit expression) và *biểu thức ẩn* (implicit expression).

Trong ví dụ trên, `@DateTime.Now` là biểu thức ẩn, `@(DateTime.Now - TimeSpan.FromDays(7))` là biểu thức rõ.

Sự khác biệt về mặt cú pháp giữa hai loại biểu thức nằm ở chỗ biểu thức rõ được đặt trong cặp dấu ngoặc ().

Biểu thức ẩn (implicit expression) không được phép chứa khoảng trống hoặc những ký tự đặc biệt gây nhầm lẫn với code HTML xung quanh.

Ví dụ biểu thức ẩn không thể chứa generics (vì cú pháp generics sử dụng cặp dấu <> với tham số kiểu, dẫn đến nhầm lẫn với thẻ HTML). Biểu thức ẩn cũng không thể là các phép toán thông thường (+, -, *, /).

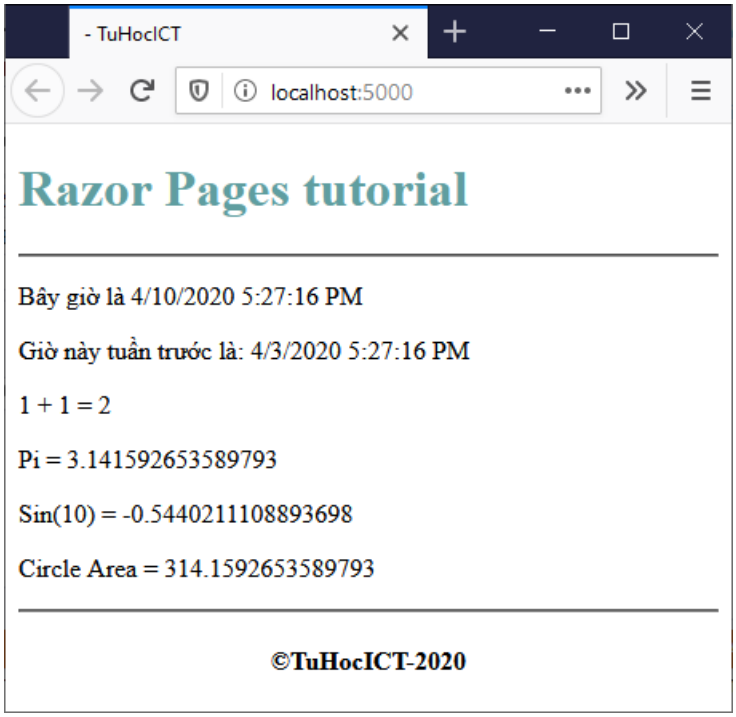
Nếu một biểu thức không thể biểu diễn ở dạng ẩn thì bạn bắt buộc phải dùng **biểu thức rõ**: biểu thức đặt trong cặp dấu ().

Như vậy, biểu thức ẩn thực ra là dạng đơn giản hóa giúp bạn tiện lợi hơn khi tích hợp biểu thức C# với HTML. Biểu thức rõ là dạng đầy đủ khi tích hợp biểu thức C# trong HTML.

> Lưu ý: Nếu viết biểu thức ở dạng ẩn mà bị lỗi, hãy chuyển nó về dạng rõ.

Dưới đây là một số ví dụ về biểu thức Razor:

```
1.  <p>1 + 1 = @(1 + 1)</p> <!-- explicit -->
2.  <p>Pi = @Math.PI</p> <!-- implicit -->
3.  <p>Sin(10) = @Math.Sin(10)</p> <!-- implicit -->
4.  <p>Circle Area = @(Math.PI * 10 * 10)</p> <!-- explicit -->
```



Kết quả thực hiện các biểu thức Razor bên trên

Khối code Razor

Khái niệm khối code trong Razor

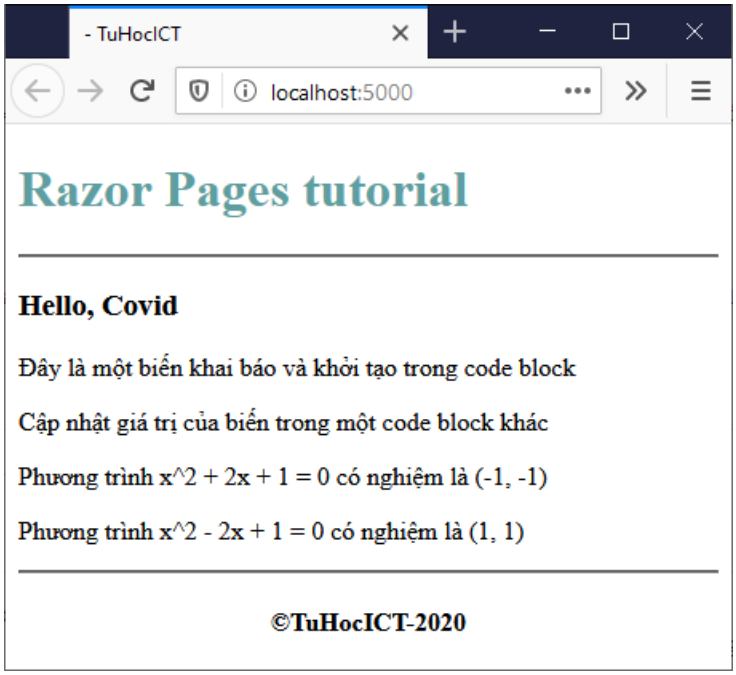
Khối code là một khái niệm cơ bản và rất quan trọng của Razor. **Khối code** Razor (Razor code block) là tất cả những code nằm trong một vùng @{ … }. Khối code là một vùng trong file/page Razor trong đó ngôn ngữ mặc định là C#.

> Các **cấu trúc điều khiển của Razor** bạn học trong bài tiếp theo cũng đều là những khối code.

Hãy cùng xem ví dụ sau:

```
@page
@{ // đây là một khối code
    // khai báo và khởi tạo biến
    var str = "Đây là một biến khai báo và khởi tạo trong code block";

    string SayHello(string name) {
        return $"Hello, {name}";
    }

    var sol = Solve(1, -2, 1);
}

<h3>@SayHello("Covid")</h3>

<p>@str</p>

@{ // đây là một khối code khác
    str = "Cập nhật giá trị của biến trong một code block khác";
}

<p>@str</p>

<p>Phương trình x^2 + 2x + 1 = 0 có nghiệm là @Solve(1, 2, 1)</p>

<p>Phương trình x^2 - 2x + 1 = 0 có nghiệm là @sol</p>
```

```
26.
27.  @{ // đây cũng là một khối code
28.      (double, double) Solve(double a, double b, double c) {
29.          var d = b * b - 4 * a * c;
30.          return
31.              ((-b + Math.Sqrt(d)) / (2 * a),
32.              (-b - Math.Sqrt(d)) / (2 * a));
33.      }
34.  }
```



So với biểu thức Razor, bên trong code block bạn có thể viết bất kỳ lệnh C# nào. Hiểu một cách đơn giản, code block là nơi bạn tự do viết code C# như trong một chương trình C# bình thường.

> Một giới hạn quan trọng trong code block là bạn không thể khai báo kiểu dữ liệu mới (như class, struct, enum, v.v.). Để giải quyết vấn đề này bạn phải sử dụng một mô hình tổ chức code khác của Razor mà chúng ta sẽ học trong bài sau.

Những đặc điểm của code block

(1) Code block là nơi bạn dùng để khai báo/khởi tạo biến và khai báo hàm cục bộ (local function). Trong ví dụ trên bạn đã khai báo và khởi tạo biến `str`, hàm `SayHello`, hàm `Solve`, biến `sol`.

> Lưu ý trong code block chúng ta không gọi là "phương thức" mà gọi là "hàm cục bộ".

```
1.  @page
2.  @{
3.      var str = "Đây là một biến khai báo và khởi tạo trong code block";
4.
5.      string SayHello(string name) {
6.          return $"Hello, {name}";
7.      }
8.
9.      var sol = Solve(1, -2, 1);
10. }
11.
```

```
@{
    (double, double) Solve(double a, double b, double c) {
        var d = b * b - 4 * a * c;
        return
            ((-b + Math.Sqrt(d)) / (2 * a),
            (-b - Math.Sqrt(d)) / (2 * a));
    }
}
```

(2) Các biến và hàm tạo ra trong code block có thể được sử dụng làm biểu thức Razor cũng như được sử dụng trong code block khác trên page đó.

Trong ví dụ trên, bạn đã sử dụng biến str, biến sol, gọi hàm SayHello và Solve để làm biểu thức Razor:

```
<h3>@SayHello("Covid")</h3>

<p>@str</p>

<p>@str</p>

<p>Phương trình x^2 + 2x + 1 = 0 có nghiệm là @Solve(1, 2, 1)</p>

<p>Phương trình x^2 - 2x + 1 = 0 có nghiệm là @sol</p>
```

(3) Không giới hạn số lượng và vị trí viết code block trên một page.

(4) Vị trí khai báo hàm không quan trọng, nghĩa là bạn có thể gọi hàm ở code block trước nơi hàm đó khai báo. Như bạn thấy trong ví dụ trên, Lời gọi hàm Solve được viết trước khi hàm này được khai báo.

(5) Biến chỉ có thể sử dụng sau vị trí khai báo. Đây là điều khác biệt với hàm. Nếu bạn sử dụng biến trước khi khai báo sẽ bị lỗi.

Chuyển đổi ngôn ngữ trong code block

Như đã nói ở trên, trong file Razor, ngôn ngữ mặc định là HTML. Riêng trong code block, ngôn ngữ mặc định lại là C#. Razor cho phép chuyển đổi ngôn ngữ trong code block từ C# sang HTML. Đây là đặc điểm rất mạnh nhưng cũng rối rắm của Razor.

Tự động chuyển đổi ngôn ngữ trong code block

Hãy cùng xem ví dụ sau:

```
@page
@{
    void SayHello(string name) {
        <p>
            <strong style="color:royalblue;">Hello, @name!</strong><br />
            <span>Welcome to Razor tutorial!</span>
        </p>
    }

    // Lời gọi hàm bình thường
    SayHello("H1N1");

    // Dưới đây lại là đoạn HTML kết hợp với lời gọi hàm
    <div style="background: #b6ff00;">@{SayHello("Covid"); }</div>

    // Viết code C# bình thường
}
```

```
<div style="background: #ffd800;">
    @{ SayHello("Sars"); }
</div>
```

Như bạn thấy, trong ví dụ trên, trong code block chúng ta đang vận dụng một tính năng đặc biệt của Razor: khả năng tự động chuyển đổi ngôn ngữ từ C# (mặc định) sang HTML.

Có những điểm sau cần lưu ý:

(1) Trong thân hàm SayHello chứa code HTML. Ở đây Razor tự động chuyển đổi từ C# sang HTML. Cơ chế này gọi là implicit transition (**chuyển đổi ngôn ngữ tự động**).

```
    void SayHello(string name) {
        <p>
            <strong style="color:royalblue;">Hello, @name!</strong><br />
            <span>Welcome to Razor tutorial!</span>
        </p>
    }
```

Razor sẽ tự động chuyển từ C# sang HTML nếu nó nhìn thấy thẻ HTML trong thân hàm. Bạn có thể yên tâm về sự ổn định khi chuyển đổi C# -> HTML tự động vì cú pháp HTML rất khác biệt với C#.

(2) Cơ chế chuyển đổi ngôn ngữ tự động hoạt động trong cả thân hàm và trong code block. Một khi đã chuyển đổi sang HTML, Razor sẽ chỉ chuyển đổi trở lại C# sau khi gặp thẻ đóng. Trong khối HTML bạn lại phải vận dụng kỹ thuật chuyển đổi từ HTML trở lại C# như đã biết.

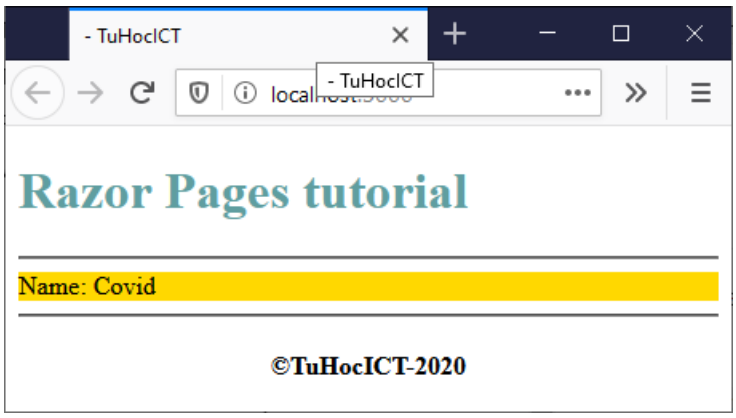
```
// Dưới đây lại là đoạn HTML kết hợp với lời gọi hàm
<div style="background: #b6ff00;">@{SayHello("Covid"); }</div>
```

Chuyển đổi ngôn ngữ chủ động

Trong một số trường hợp cơ chế chuyển đổi ngôn ngữ tự động có thể hoạt động không đúng ý bạn. Khi đó bạn có thể lựa chọn sử dụng cơ chế chuyển đổi ngôn ngữ chủ động của Razor. Đây là tình huống khi bạn **không muốn sử dụng thẻ HTML**. Ví dụ:

```
    void SayHello(string name) {
        <text>Name: @name</text>
    }
```

Do bạn không dùng thẻ HTML, Razor không hiểu được rằng bạn đang muốn xuất ra HTML. Trong trường hợp này bạn dùng cặp thẻ <text></text> bao quanh dữ liệu cần xuất ra.



Nếu sử dụng công cụ inspect của trình duyệt bạn sẽ thấy kết quả như sau:

```
<div style="background:#ffd800">Name: Covid</div>
```

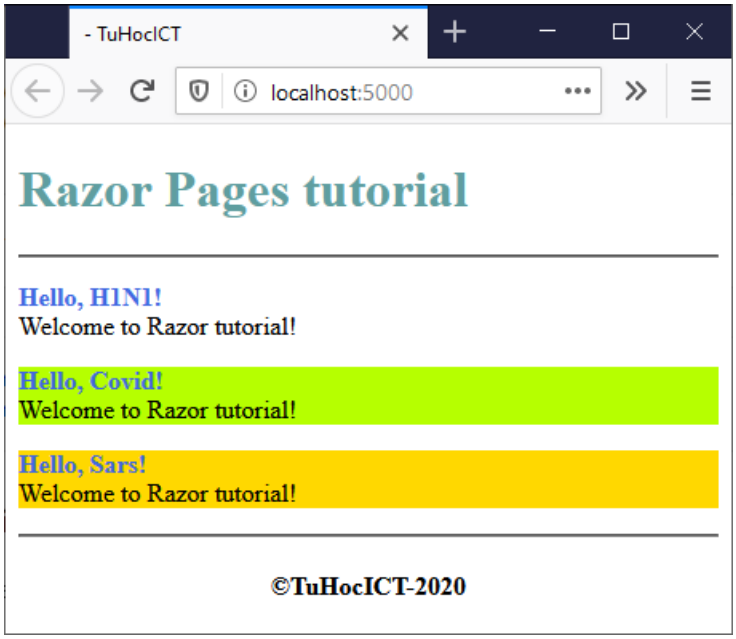
Rõ ràng kết quả xuất ra HTML của hàm không sử dụng thẻ nào. Tức là thẻ <text> chỉ có tác dụng thông báo cho Razor để chuyển sang chế độ render HTML chứ không phải là một thẻ HTML.

Hàm mẫu

Hàm mẫu (template function/method) là loại hàm cục bộ đặc biệt có khả năng xuất ra (render) HTML ở vị trí gọi hàm. Đặc điểm này giúp phân biệt hàm mẫu với hàm cục bộ thông thường (không render HTML).

Thực chất hàm mẫu là cách vận dụng tính năng chuyển đổi ngôn ngữ và xuất HTML của Razor trong code block.

Trong ví dụ trên bạn đã viết hàm mẫu SayHello và gọi nó ở 3 vị trí khác nhau. Kết quả thu được như sau:



Lời gọi hàm mẫu sẽ sinh ra HTML. Bạn không thể gọi hàm mẫu theo kiểu biểu thức Razor. Hàm mẫu phải được gọi trong code block `@{SayHello("Covid"); }`

```
<div style="background: #b6ff00;">@{SayHello("Covid"); }</div>

<div style="background: #ffd800;">
    @{ SayHello("Sars"); }
</div>
```

Kết luận

Trong bài học này bạn bắt đầu làm quen với một vấn đề quan trọng hàng đầu trong ASP.NET Core: cú pháp Razor. Cú pháp này được sử dụng trong một số framework bên trên Asp.net Core như MVC, Razor Pages, Blazor.

Tải mã nguồn solution S03_RazorSyntax

1 file(s) 0.00 KB

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!