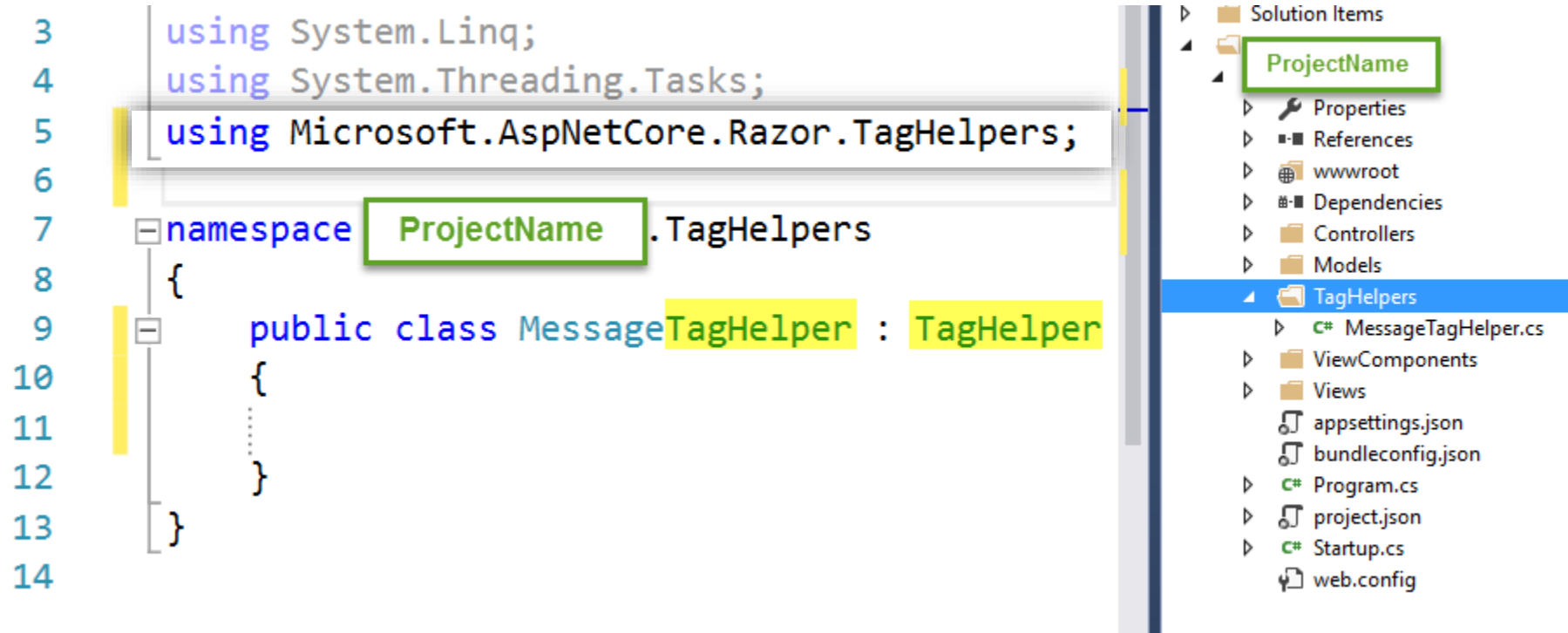


# HƯỚNG DẪN TẠO MỘT TAG HELPER TRONG MVC .NET CORE (MVC 6 +)

1. Tạo thư mục TagHelpers đặt tại thư mục gốc của dự án
2. Bên trong tạo một class có tên gọi tùy ý, lưu ý có đuôi TagHelper

```
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Razor.TagHelpers;
6
7  namespace ProjectName.TagHelpers
8  {
9      public class MessageTagHelper : TagHelper
10     {
11         ...
12     }
13 }
14
```



The screenshot displays the Visual Studio IDE. On the left, the code editor shows the implementation of a TagHelper class named `MessageTagHelper` within the `ProjectName.TagHelpers` namespace. The code includes using statements for `System.Linq`, `System.Threading.Tasks`, and `Microsoft.AspNetCore.Razor.TagHelpers`. The class `MessageTagHelper` inherits from `TagHelper`. On the right, the Solution Explorer shows the project structure. The `TagHelpers` folder is highlighted, and the file `MessageTagHelper.cs` is listed within it.

### 3. Override hàm Process của base class

```
public class MessageTagHelper : TagHelper
{
    public override void Process(TagHelperContext context, TagHelperOutput output)
    {
        base.Process(context, output);
    }
}
```

## 4. Code

Demo sau tạo một TagHelper message với 2 thuộc tính như sau:

```
<message type="success" content="Đã gửi thư liên hệ"></message>  
<message type="danger" content="Rất tiếc, không thể gửi thư liên hệ"></message>
```

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Razor.TagHelpers;  
  
namespace ProjectName.TagHelpers  
{  
    [HtmlTargetElement("message")]  
    public class MessageTagHelper : TagHelper  
    {  
        [HtmlAttributeName("type")]  
        public string Type { get; set; } = "info";  
  
        [HtmlAttributeName("content")]  
        public string Content { get; set; }  
  
        public override void Process(TagHelperContext context, TagHelperOutput output)  
        {  
            string type = this.Type.ToString().ToLower();  
            string content = this.Content;  
  
            string template = @$@"<div class='alert alert-{type}'>  
                                    <button type='button' class='close'  
                                        data-dismiss='alert'>&times;</button>  
                                    <span>{content}</span>  
                                </div>";  
  
            output.TagName = string.Empty;  
            output.Content.SetHtmlContent(template);  
        }  
    }  
}
```

## 5. ProcessAsync TagHeper

Để load nội dung bên trong TagHelper, cần **override async** hàm **ProcessAsync** như sau:

```
<message type="success">
```

```
    Đã gửi thư liên hệ <a href="/">Về trang chủ</a>
```

```
</message>
```

```
<message type="danger">
```

```
    Rất tiếc, không thể gửi thư liên hệ <a href="/">Về trang chủ</a>
```

```
</message>
```

```
public override async Task ProcessAsync(TagHelperContext context, TagHelperOutput output)
{
    string type = this.Type.ToString().ToLower();
    string content = this.Content;

    #region Only For ProcessAsync
    if (string.IsNullOrEmpty(content))
    {
        var elemContent = await output.GetChildContentAsync();
        content = elemContent.GetContent();
        this.Content = content;
    }
    #endregion

    string template = @$"<div class='alert alert-{type}'>
        <button type='button' class='close' data-dismiss='alert'>&times;</button>
        <span>{content}</span>
    </div>";

    output.TagName = string.Empty;
    output.Content.SetHtmlContent(template);
}
```

## 6. Khai báo sử dụng TagHelper tại file `_ViewImports.cshtml`

`@addTagHelper` "\*", `ProjectName`"

Với `ProjectName` là tên dự án của bạn

Tại View, để đổ dữ liệu từ ViewBag vào TagHelper đã tạo ra ở các bước trước, có thể làm theo **cách 1**:

```
<message type="@ViewBag.MessageType" content="@ViewBag.Message">
</message>
```

Nếu nội dung của thuộc tính `content` dài, phức tạp, hoặc có chứa mã HTML, thì hãy dùng **cách 2**:

```
<message type="@ViewBag.MessageType">
    @ViewBag.Message
</message>
```

Trường hợp vừa dùng thuộc tính, vừa dùng nội dung, thì message tagHelper này sẽ ưu tiên thuộc tính.

```
<message type="@ViewBag.MessageType" content="@ViewBag.Message">
    @ViewBag.Message
</message>
```

Không quan tâm đến giá trị này

Ưu tiên dùng giá trị này

Nếu muốn ưu tiên theo hướng ngược lại, hãy chỉnh sửa lại hàm **ProcessAsync**. Vd:

```
public override async Task ProcessAsync(TagHelperContext context, TagHelperOutput output)
{
    string type = this.Type.ToString().ToLower();
    string content = string.Empty;

    var elemContent = await output.GetChildContentAsync();
    content = elemContent.GetContent();

    if (string.IsNullOrEmpty(content))
        content = this.Content;

    string template = @"<div class='alert alert-{type}'>
                        <button type='button' class='close' data-dismiss='alert'>&times;</button>
                        <span>{content}</span>
                    </div>";

    output.TagName = string.Empty;
    output.Content.SetHtmlContent(template);
}
```

Code vui 😊

Thầy Hiếu – mrhieuit@gmail.com