Phần 4: Thêm mới View

**View** là một thành phần trong mô hình [MVC](https://www.dammio.com/glossary/mvc) (Model-View-Controller), chịu trách nhiệm chủ yếu về việc hiển thị nội dung trên giao diện Web (bằng mã HTML) cũng như tiếp nhận các tương tác từ người dùng (client) để gửi đến server.

Đầu tiên, bạn sẽ chỉnh sửa các phương thức trong lớp **DammioController** để trả về các tập tin View. Sau đó, bạn phải tạo các tập tin View (đuôi mở rộng là **.cshtml**) và sử dụng Razor để viết mã. **Cơ chế Razor** giảm thiểu số lượng ký tự và phím nhấn cần để tạo một tập tin View, do đó lập trình viên có thể làm việc nhanh và trôi chảy hơn.

Trong lớp DammioController.cs, chỉnh sửa phương thức Index như sau:

// GET: /Dammio/

public ActionResult Index()

{

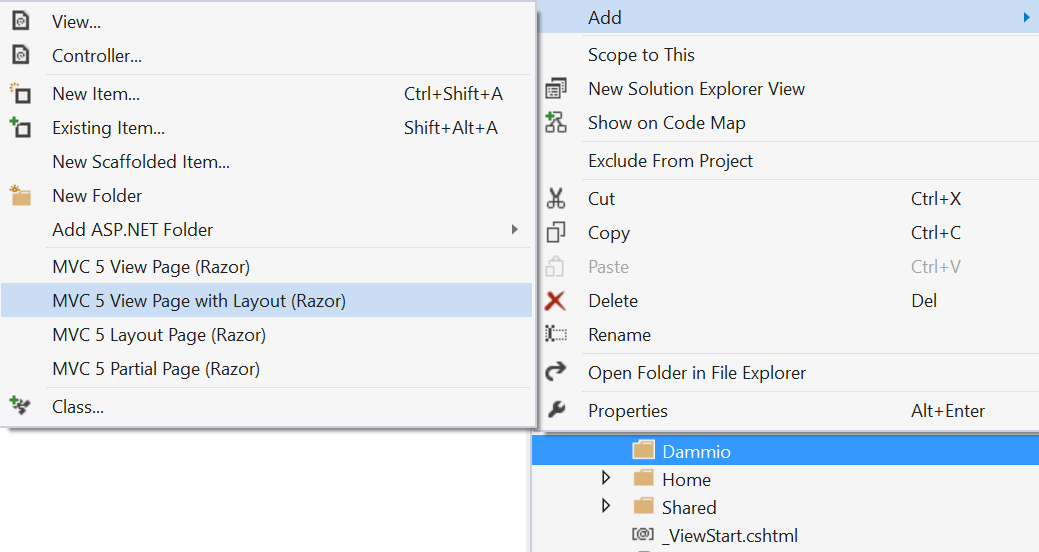
      return View();

}

Trong ví dụ trên, phương thức Index sẽ trả về một ActionResult (thể hiện kết quả của một phương thức hành động) hay một lớp kế thừa từ lớp ActionResult bằng câu lệnh return View();. View() được hiểu là ám chỉ một View tương ứng với Controller.

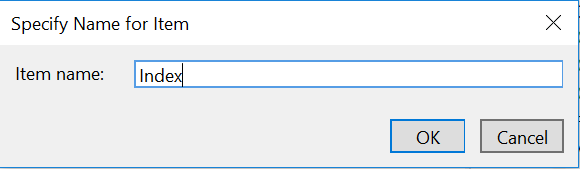
# Tạo một View cho Controller

Để tạo một View mới, bạn chọn thư mục Views/Dammio trong Solution Explorers, sau đó chuột phải chọn Add -> MVC5 View Page with Layout (Razor). Vì bạn mới tìm hiểu về View cho nên bạn nên dùng kiểu tạo mới bằng Layout, khi nào bạn rành bạn có thể tạo mới bằng Add -> View nếu muốn.

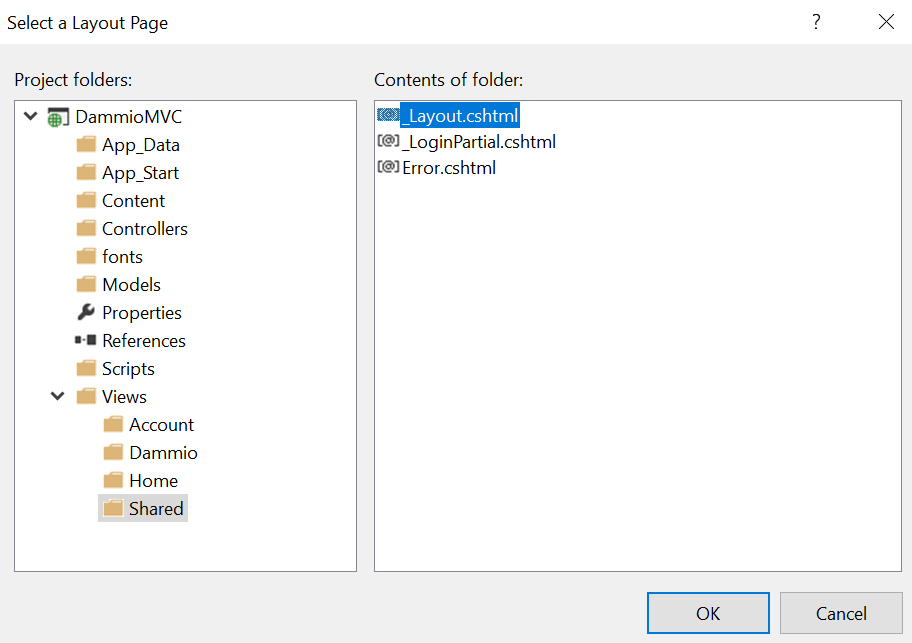


Lưu ý, bạn phải thêm View tương ứng với Controller. Ví dụ, bạn có Controller tên là ABCController, thì lúc thêm View bạn phải thêm vào thư mục Views/ABC.

Sau đó, bạn phải đặt tên View vào hộp thoại, ví dụ Index và nhấn OK.



Tiếp theo, bạn chọn Layout (bố cục) cho View. Khi tạo dự án [MVC](https://www.dammio.com/glossary/mvc) thì Visual Studio để tạo cho bạn View mặc định là \_Layout.cshtml, bạn có thể tạo bất cứ layout nào nếu muốn.



Sau khi bạn tạo xong, một tập tin **Index.cshtml** sẽ được tạo trong thư mục Views/Dammio vói nội dung mã nguồn như sau.

@{

    Layout = "~/Views/Shared/\_Layout.cshtml";

}

Tiếp theo bạn chỉnh sửa tập tin **Index.cshtml** với nội dung mã nguồn sau:

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@{

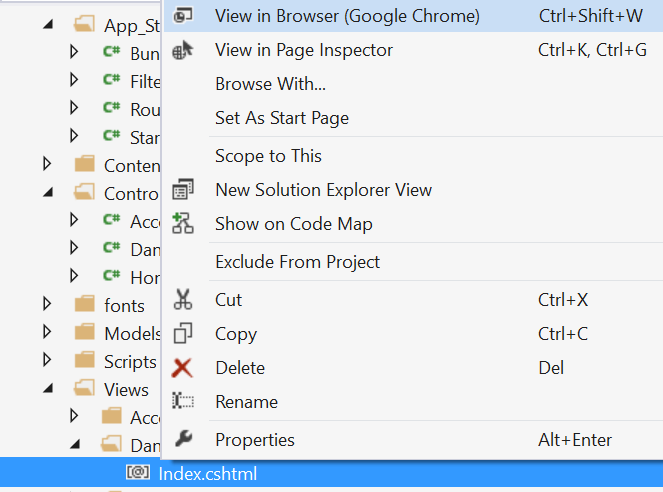
ViewBag.Title = "Index";

}

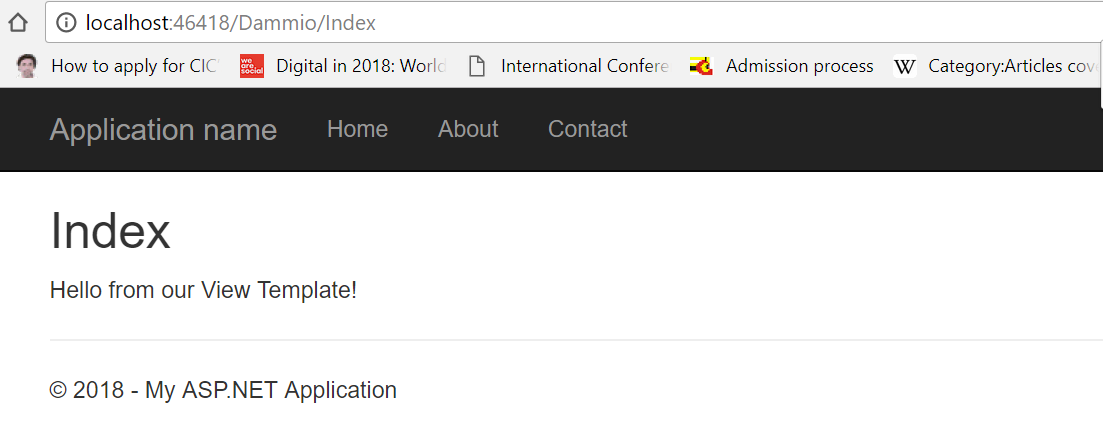
<h2>Index</h2>

<p>Hello from our View Template!</p>

Sau đó, bạn chọn chuột phải lên tập tin Index.cshtml ở Solution Expoler và chọn chế độ **View in Browser** để xem nội dung View Index vừa tạo ở trình duyệt. Cách thứ hai, bạn có thể chạy đường dẫn **http://localhost:xxxx/Dammio/Index** sau khi nhấn F5 hoặc Ctrl + F5 ở dự án.



Cuối cùng, đây là **giao diện trang Index.cshtml** bạn có ở trình duyệt như sau:



Đến bước này, tôi sẽ trình bày tóm tắt lại để bạn hiểu rõ quy trình hiển thị dữ liệu trên giao diện View như sau, mong bạn đọc và chú ý kỹ, đây là phần quan trọng.

Đầu tiên, một người dùng sẽ chạy đường dẫn http://localhost:xxxx/Dammio/Index, server sẽ dò tìm và biết được đường dẫn này thuộc về tập tin DammioController.cs với phương thức Index(). Tiếp tục, server sẽ thực thi nội dung trong phương thức Index() và trả về dòng lệnh return View(). Vì đây là phương thức Index trả về View, server sẽ dò tìm đúng tập tin có tên Index.cshtml nằm trong thư mục Views/Dammio. Tiếp theo, server đọc nội dung Index.cshtml và hiển thị mã HTML trên màn hình.

Nếu bạn nắm được cách hoạt động trên, thì bạn cũng sẽ hiểu được tất cả các View trong ASP.NET [MVC](https://www.dammio.com/glossary/mvc) đều hoạt động như vậy.

Tiếp theo, chúng ta thử tạo 1 View tên là **Hello.cshtml** nằm trong thư mục **Views/Dammio**. Để tạo mới View, bạn phải tạo phương thức tương ứng trong lớp DammioController trước. Bạn tạo phương thức Hello với nội dung như sau ở tập tin DammioController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace DammioMVC.Controllers

{

public class DammioController : Controller

{

//

// GET: /Dammio/

public ActionResult Index()

{

return View();

}

// Phương thức Hello

public ActionResult Hello()

{

return View();

}

//

// GET: /Dammio/ChaoMung/

public string ChaoMung(string ten, int tuoi = 1)

{

return HttpUtility.HtmlEncode("Xin chào " + ten + ". Tuổi của bạn là: " + tuoi);

}

}

}

Sau đó tương tự bạn cũng tạo 1 View tên là Hello.cshtml nằm trong thư mục **Views/Dammio** với nội dung:

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

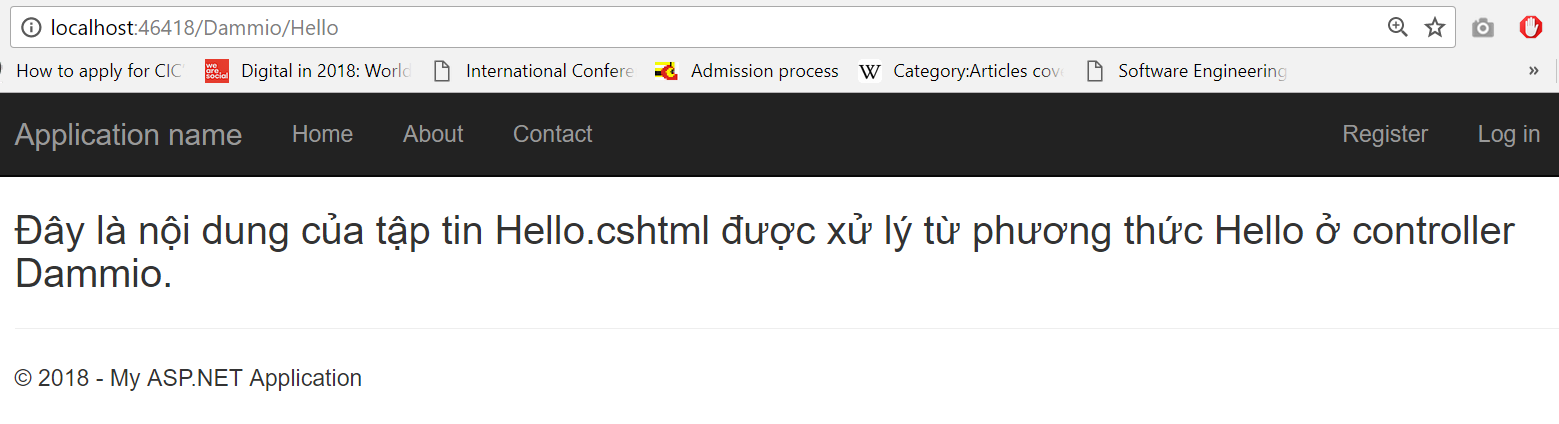
@{

ViewBag.Title = "Hello";

}

<h3>Đây là nội dung của tập tin Hello.cshtml được xử lý từ phương thức Hello ở controller Dammio (DammioController).</h3>

Tiếp theo, bạn nhấn chuột phải lên tập tin Hello.cshtml và View in Browser để xem giao diện ở trình duyệt hoặc chạy đường dẫn **http://localhost:xxxx/Dammio/Hello** sau khi nhấn F5 hoặc Ctrl + F5 ở dự án.



**Thay đổi nội dung trang bố cục (Layout) và tập tin View**  
Sau khi khởi tạo các trang view theo các phương thức khác nhau ở controller, bạn tiếp tục tìm hiểu cách thay đổi nội dung các trang View cũng như bố cục trang bố cục.

Đầu tiên, bạn mở tập tin bố cục **\_Layout.cshtml** ở thư mục /Views/Shared trong cửa sổ Solution Explorer và thấy mã nguồn như sau:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title - My ASP.NET Application</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

@Html.ActionLink("Application name", "Index", "Home", null, new { @class = "navbar-brand" })

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Home", "Index", "Home")</li>

<li>@Html.ActionLink("About", "About", "Home")</li>

<li>@Html.ActionLink("Contact", "Contact", "Home")</li>

</ul>

@Html.Partial("\_LoginPartial")

</div>

</div>

</div>

<div class="container body-content">

@RenderBody()

<hr />

<footer>

<p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>

</footer>

</div>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

@RenderSection("scripts", required: false)

</body>

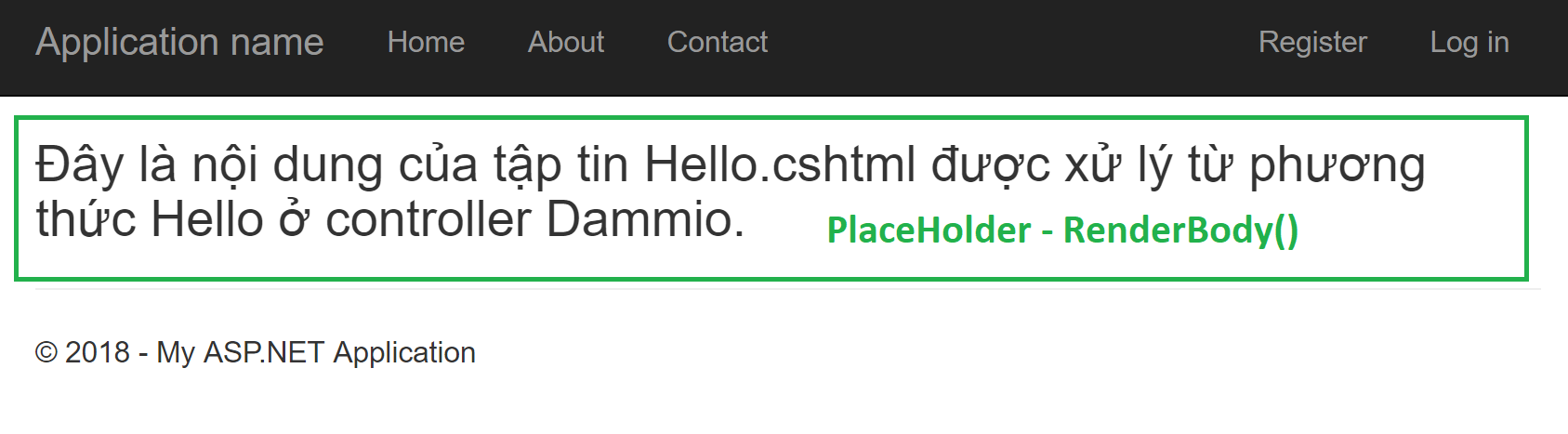
</html>

Trong tập tin này, ở dòng <title>@ViewBag.Title – My ASP.NET Application</title> bạn có thể đổi tên tiêu đề cho các trang Web tùy ý. @ViewBag.Title là một biến Title thuộc đối tượng chứa biến ViewBag và có thể được thiết lập giá trị ở mỗi trang View khác nhau. Tiếp theo bạn thay dòng này bằng đoạn mã

<title>@ViewBag.Title - Dammio Website</title>



Tiếp theo, bạn dò tìm dòng **@RenderBody()**. RenderBody chính là 1 placeholder (nơi chứa) giúp bạn đặt nội dung khác nhau ở mỗi trang khác nhau, bao quanh chính là bố cục trang. Để hình dụng điều này, bạn chạy lại trang http://localhost:xxxx/Dammio/Hello ở trình duyệt:



Trong hình trên, bạn có thể thấy trang Hello.cshtml sử dụng tập tin \_Layout.cshtml làm tập tin bố cục, vì vậy menu màu đen ở đầu và dòng chữ “© 2018 – My ASP.NET Application” đều lấy từ tập tin \_Layout.cshtml, chỉ có nội dung ở giữa (khung xanh lá) là nội dung của chính trang Hello.cshtml.

Trong phần Menu của tập tin \_Layout.cshtml, bạn có thể chỉnh sửa ActionLink (một siêu liên kết, giống như thẻ a href) như sau:

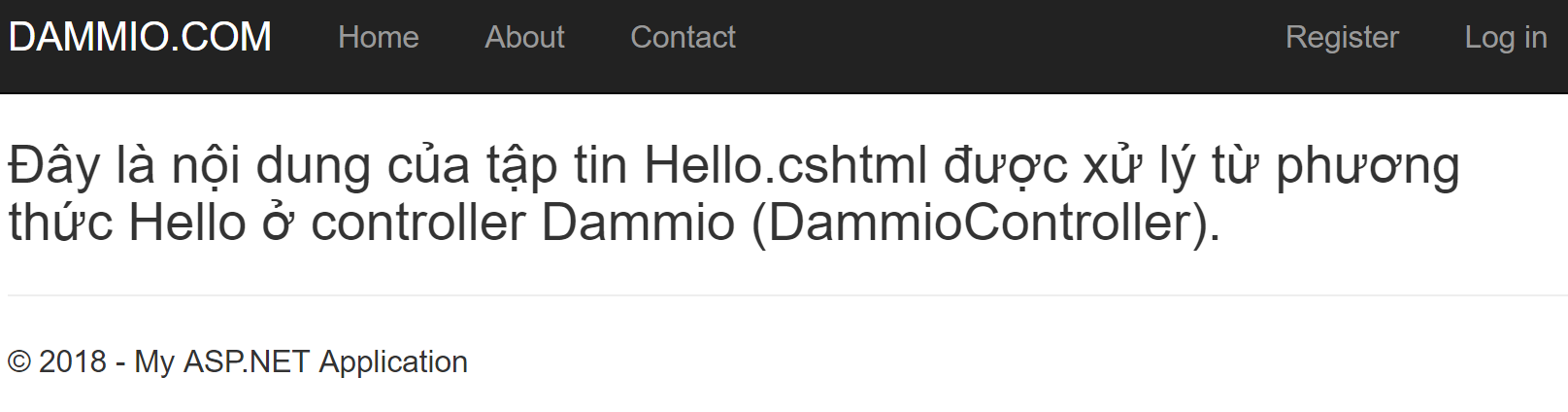
Thay dòng:

@Html.ActionLink("Application name", "Index", "Home", null, new { @class = "navbar-brand" })

bằng:

@Html.ActionLink("DAMMIO.COM", "Index", "Home", null, new { @class = "navbar-brand" })

Bạn có thể chạy lại trang http://localhost:xxxx/Dammio/Hello để xem kết quả thay đổi tên Menu.



Ở đây, xin giải thích về **phương thức ActionLink**, phương thức này có cú pháp:

ActionLink(string **linkText**, string **actionName**, string **controllerName**)

* **linkText:** là văn bản hiển thị trên đường dẫn
* **actionName:** là tên phương thức
* **controllerName:** là tên controller

Cách dùng ActionLink như sau:

@Html.ActionLink("Liên kết", "Hello","Dammio")

thì sẽ phân giải thành mã HTML như sau:

<a href="/Dammio/Hello">Liên kết</a>

Kế đến, bạn có thay đổi bất kỳ nội dung nào trong tập tin \_Layout.cshtml bạn muốn. Chúng ta quay trở lại trang Hello.cshtml ở thư mục Views/Dammio, chúng ta mở tập tin này lên lại một lần nữa:

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

Trong tập tin này, bạn thấy đoạn sau chính là nơi nhúng tập tin bố cục \_Layout.cshtml vào trang web của bạn.

@{

    Layout = "~/Views/Shared/\_Layout.cshtml";

}

Nếu bạn không muốn nhúng tập tin bố cục \_Layout.cshtml vào trang Hello.cshtml, bạn có thể xóa hoặc che đoạn này bằng cách thêm chú thích:

@\*@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}\*@

Tiếp đến, bạn có thể chỉnh sửa nội dung trang Hello.cshtml tùy ý bằng mã HTML hay bằng cú pháp Razor. Bạn sẽ tìm hiểu về cú pháp Razor trong các bài sau.

# Đẩy dữ liệu từ Controller sang View

Trước khi thảo luận đến database và các mô hình, chúng ta hãy tìm hiểu cách dữ liệu được chuyển/đẩy từ controller sang view. Như giải thích ở phần trước, người dùng sẽ gõ địa chỉ web, địa chỉ này sẽ phân giải sang Controller cần đọc cũng như thuộc phương thức nào (ví dụ http://localhost:xxxx/Dammio/Hello thì sẽ thuộc về phương thức Hello thuộc tập tin DammioController.cs). Tiếp đến từ phương thức, chúng ta sẽ có đoạn mã return View(), trong trường hợp này từ Controller sang View không đẩy bất cứ dữ liệu nào. Tuy nhiên, bạn vẫn có thể đẩy dữ liệu bạn muốn.

Mở tập tin DammioController.cs và thay đổi phương thức Hello() thành nội dung:

public ActionResult Hello(string name, int numTimes = 1)

{

ViewBag.Message = "Hello " + name;

ViewBag.NumTimes = numTimes;

return View();

}

Lúc này, phương thức sẽ dùng thêm 2 tham số name và numTimes và hai biến dữ liệu Message và NumTimes sẽ được đẩy về View. **Mấu chốt cơ chế đẩy dữ liệu nằm ở biến ViewBag**, bạn có thể hình dùng ViewBag là 1 thuộc tính động (dynamic), nó lưu trữ tất cả biến cần đẩy về. Ngoài cách dùng ViewBag, bạn còn có thể dùng ViewData (kiểu từ điện đối tượng) và TempData. Bạn sẽ tìm hiểu thêm về cách dùng ViewBag, ViewData và TempDate cũng như ưu, nhược điểm của 3 cách đẩy dữ liệu này ở các bài viết chuyên biệt.

Sau đó ở trang View tương ứng với phương thức Hello() là **Hello.cshtml** (ở thư mục Views/Dammio), bạn chỉnh sửa thành đoạn mã:

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@{

ViewBag.Title = "Hello";

}

<h3>Đây là nội dung của tập tin Hello.cshtml được xử lý từ phương thức Hello ở controller Dammio (DammioController).</h3>

<ul>

@for (int i = 0; i < ViewBag.NumTimes; i++)

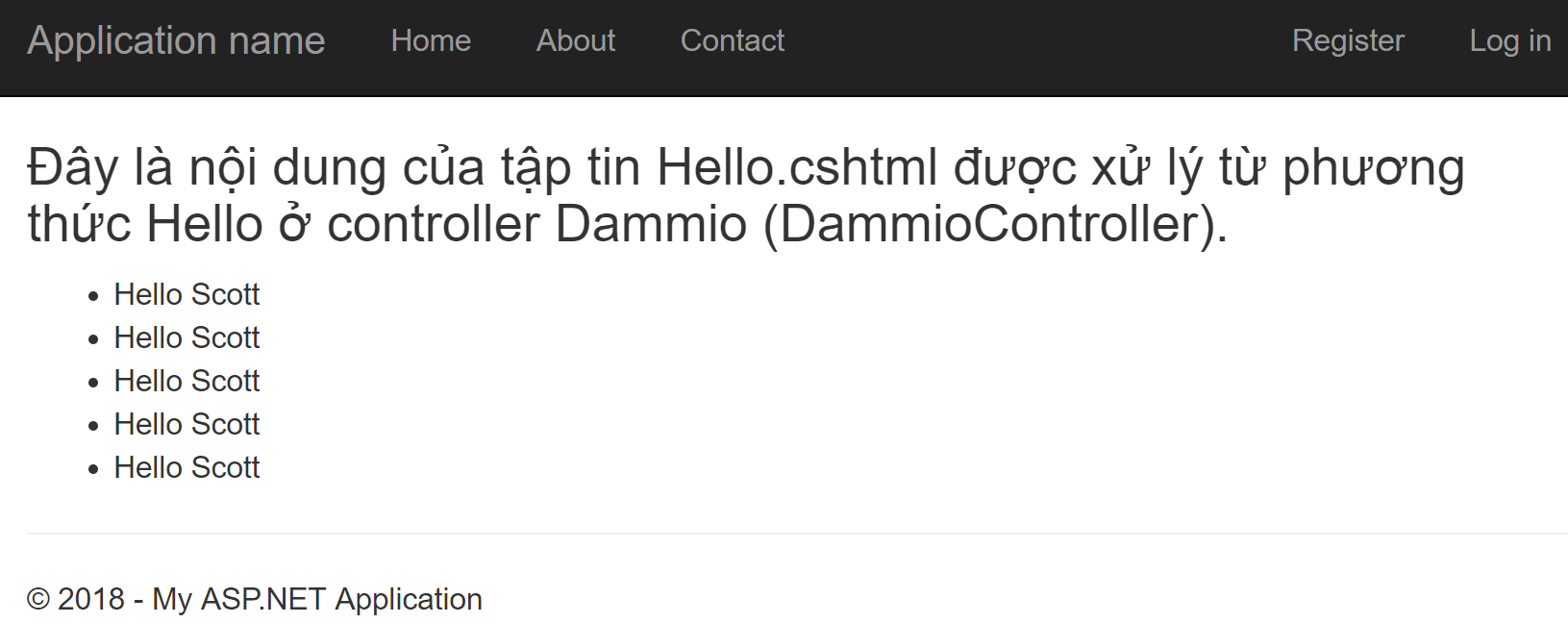
{

<li>@ViewBag.Message</li>

}

</ul>

Nhớ Build (Ctrl + Shift + B) lại toàn bộ dự án và chạy đường dẫn **http://localhost:46418/Dammio/Hello?name=Scott&numtimes=5**, bạn sẽ có kết quả ở giao diện:



Như vậy, 2 biến query name=Scott và numtimes=5 được lấy từ đầu vào và chuyển thành 2 biến trong ViewBag để đẩy dữ liệu từ Controller sang View. Cách thức tương tự như bạn đẩy dữ liệu database (nằm trong Model) đến Controller, sau đó là sang View để hiển thị dữ liệu từ database.

**Kết luận**: Bài học này giúp bạn nắm cách tạo View mới, cách hoạt động giữa View và Controller, cách thay đổi tập tin bố cục cũng như cách đẩy dữ liệu từ Controller sang View. Những kiến thức ngoài lề bạn phải tìm hiểu thêm và mời bạn tiếp tục theo dõi bài viết tiếp theo.