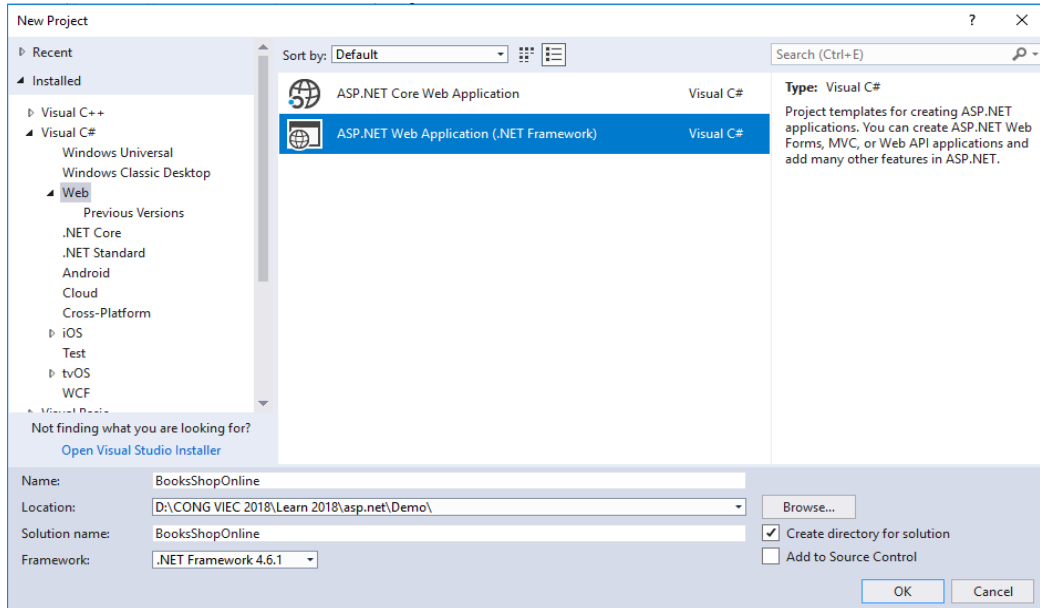


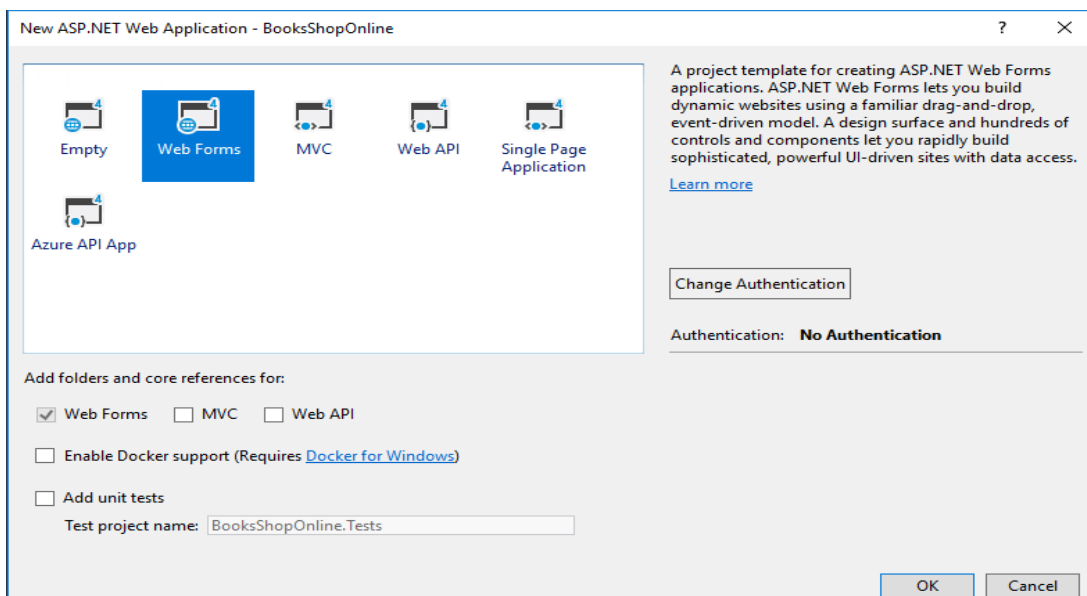
# Tạo một ứng dụng website thương mại dùng ASP.NET 4.X WebForm

## Tạo dự án

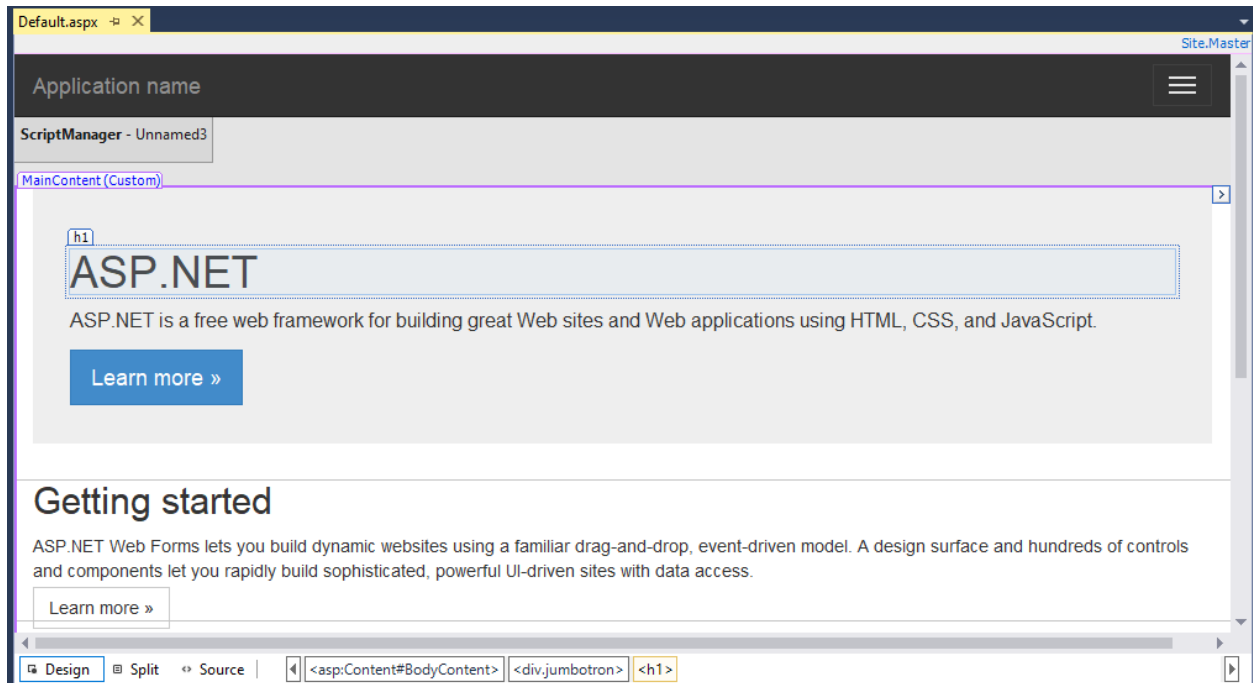
- Mở Visual Studio 2017 chọn File > New > Project
- Trong cửa sổ New Project, chọn Visual C# > Web > ASP.NET Web Application (.NET Framework)



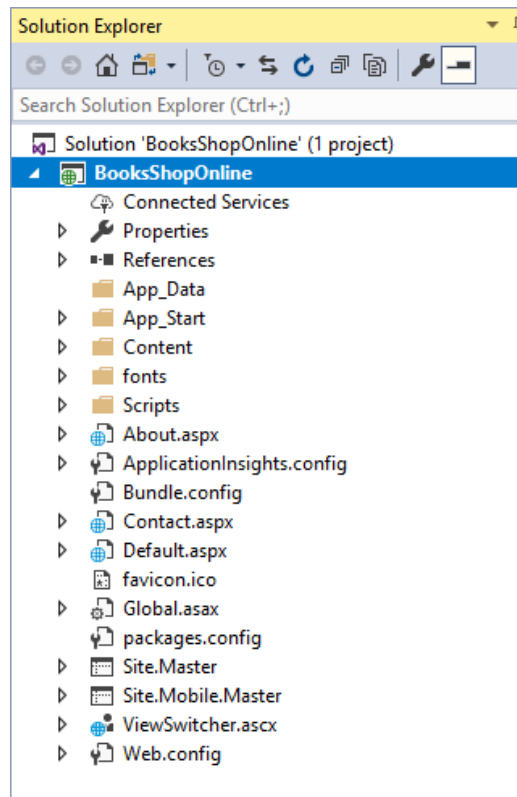
- Trong New Project, nhập tên dự án là BooksShopOnline trong mục Name, chọn đường dẫn lưu dự án trong Location và chọn phiên bản .NET Framework phù hợp (trong hình trên là 4.6.1). Nhấn OK sẽ đến hộp thoại New ASP.NET Web Application chọn Web Forms và nhấn OK.



- Chờ vài giây để dự án mới được tạo. Trong cửa sổ Solution Explorer, mở trang Default.aspx – là trang xuất hiện đầu tiên khi ứng dụng web được tải đến trình duyệt



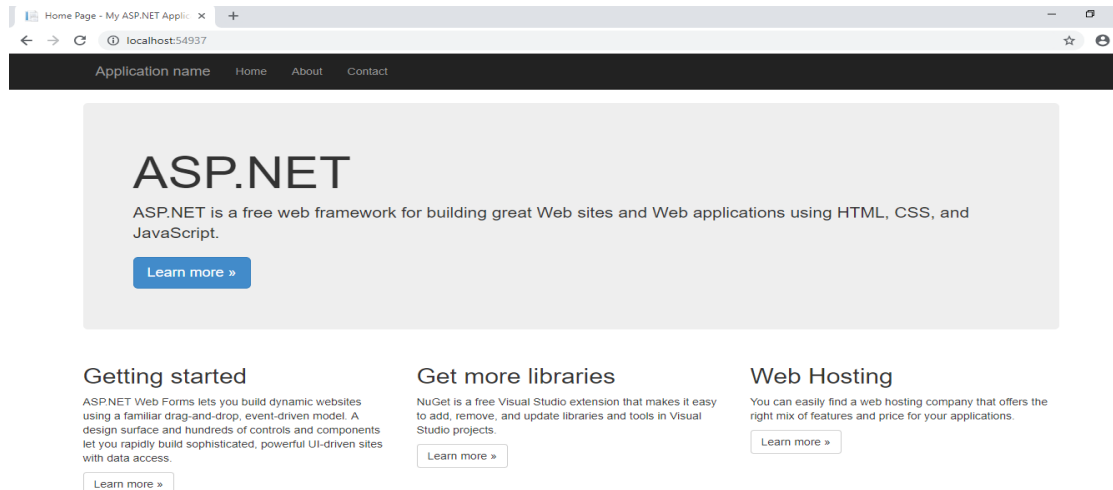
- Trang Default.aspx đang ở chế độ Design, có thể chuyển sang chế độ Source hay Split. Cấu trúc dự án mặc định trong cửa sổ Solution Explorer:



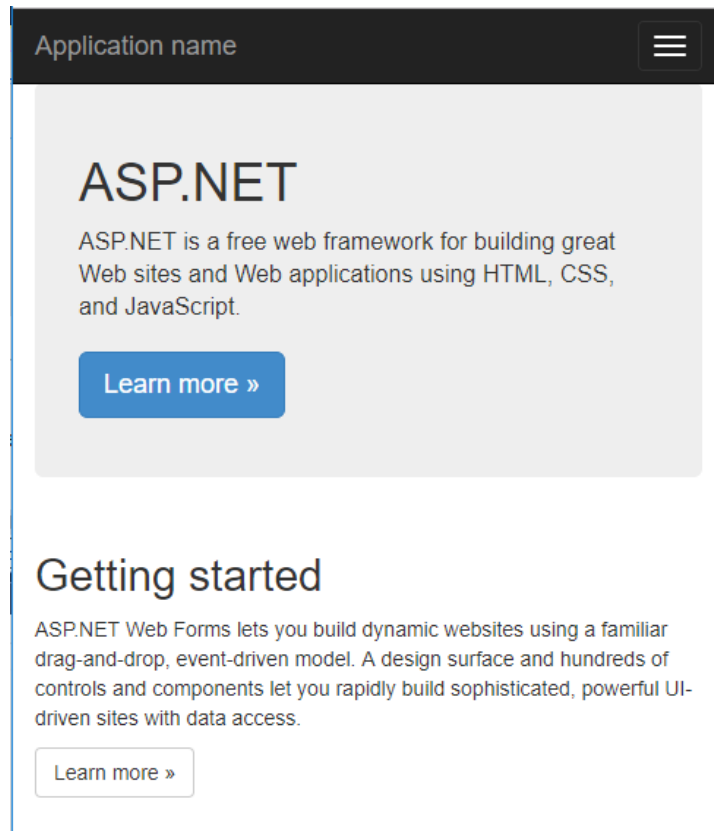
Có 3 trang mặc định khi dự án được tạo là Default.aspx, Contact.aspx và About.aspx. Trang Site.Master định nghĩa các định dạng và các ứng xử chung cho các trang khác trong ứng dụng, Site.Mobile.Master có vai trò giống Site.Master nhưng dùng trong các thiết bị di động. Global.ascx chứa các đoạn mã thực thi khi các sự kiện (mức Application hay Session) phát sinh bởi các mô-đun ASP.NET hay HTTP. Web.config chứa thông tin cấu hình ứng dụng.

Ngoài ra, VS còn phát sinh các thư mục chứa tài nguyên của dự án như kiểu chữ (thư mục fonts), các đoạn mã JavaScript hay jQuery (Scripts), nội dung (Content), v.v.

Thực thi ứng dụng lúc này, kết quả:



Trong giao diện thiết bị di động



## Mô hình dữ liệu

Trong phần này chúng ta dùng Entity Framework Code First (<https://ngocminhtran.com/entity-framework-code-first/>) và các lớp thực thể (Entity classes) – là các lớp định nghĩa lược đồ dữ liệu - để xây dựng mô hình dữ liệu cho dự án.

### Cơ sở dữ liệu BooksShopOnline

Bảng Book

	Column Name	Data Type	Allow Nulls
▶🔑	BookID	int	<input type="checkbox"/>
	BookName	nchar(100)	<input checked="" type="checkbox"/>
	Description	nchar(1000)	<input checked="" type="checkbox"/>
	Image	nchar(10)	<input checked="" type="checkbox"/>
	UnitPrice	float	<input checked="" type="checkbox"/>
	CategoryID	int	<input type="checkbox"/>

Bảng Category

	Column Name	Data Type	Allow Nulls
▶	CategoryID	int	<input type="checkbox"/>
	CategoryName	nchar(100)	<input checked="" type="checkbox"/>

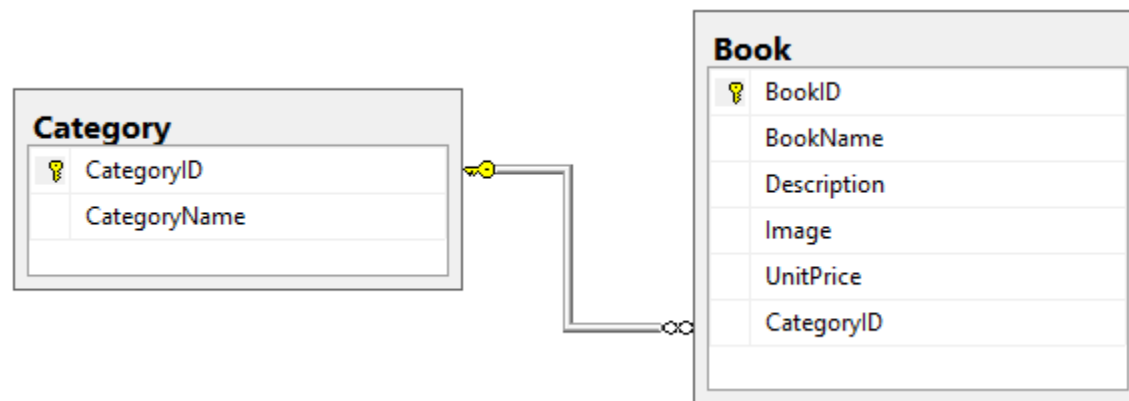
Dữ liệu bảng Book

BookID	BookName	Description	Image	UnitPrice	CategoryID
1	Fire & Blood ...	300 Years Before A Gam...	Pic1.png	16,04	1
2	Benjamin Franklin: An ...	In this authoritative and...	Pic2.png	19,6	2
3	Obama: An Intimate Po...	During Barack Obama's ...	Pic3.png	26,73	2
4	Sapiens: A Brief History ...	One hundred thousand ...	Pic4.png	23,79	3
5	The 7 Habits of Highly ...	Stephen Covey's cheris...	Pic5.png	16,04	4

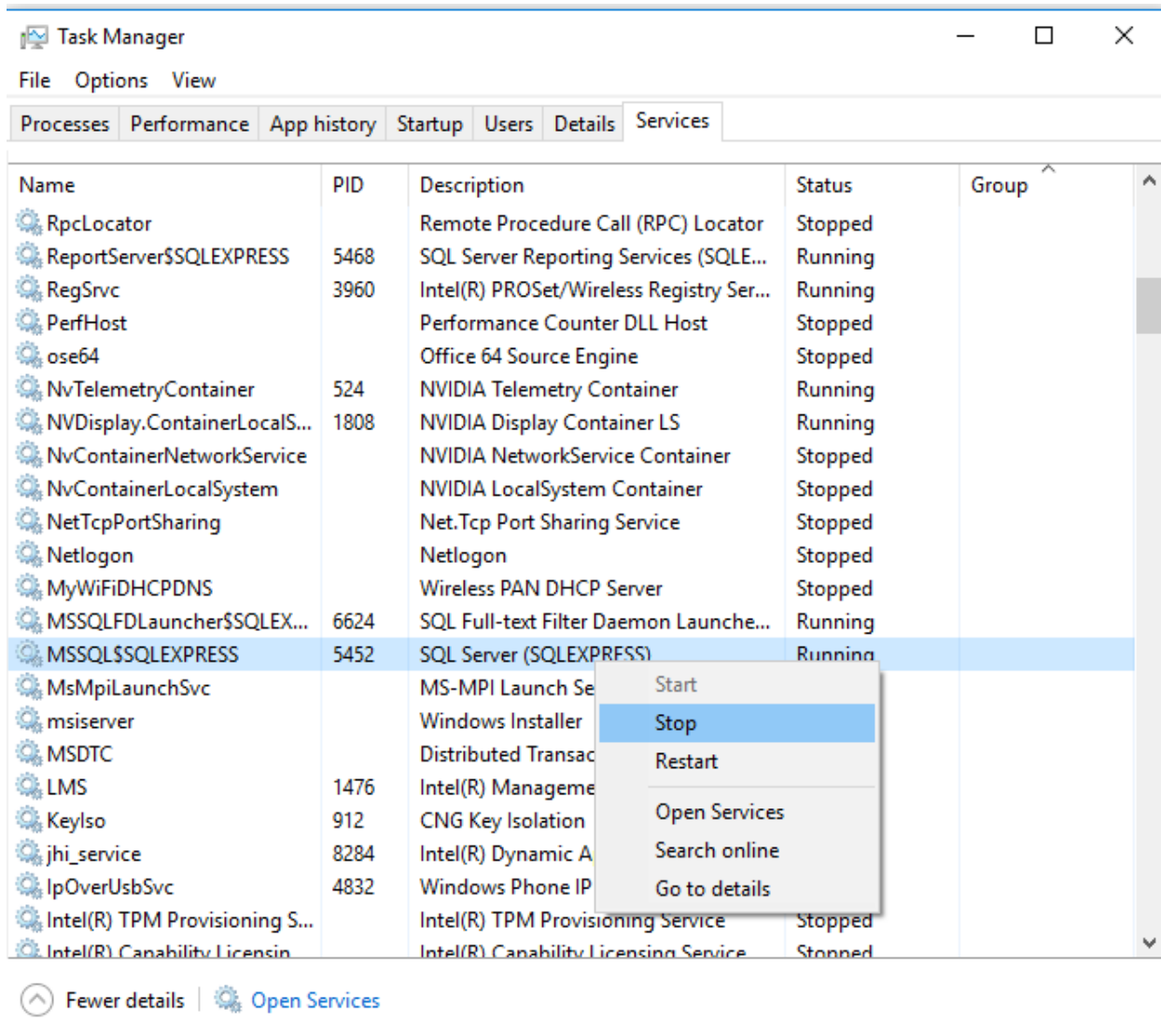
Dữ liệu bảng Category

CategoryID	CategoryName
1	Fiction
2	Biographies and Memoirs
3	Biological Sciences
4	Self-Help

Quan hệ



Tắt tất cả các dịch vụ SQL Server bằng cách mở Task Manager (Ctrl + Alt + Del)



Vào địa chỉ: C:\Program Files\Microsoft SQL Server\MSSQL11.SQLEXPRESS\MSSQL\DATA

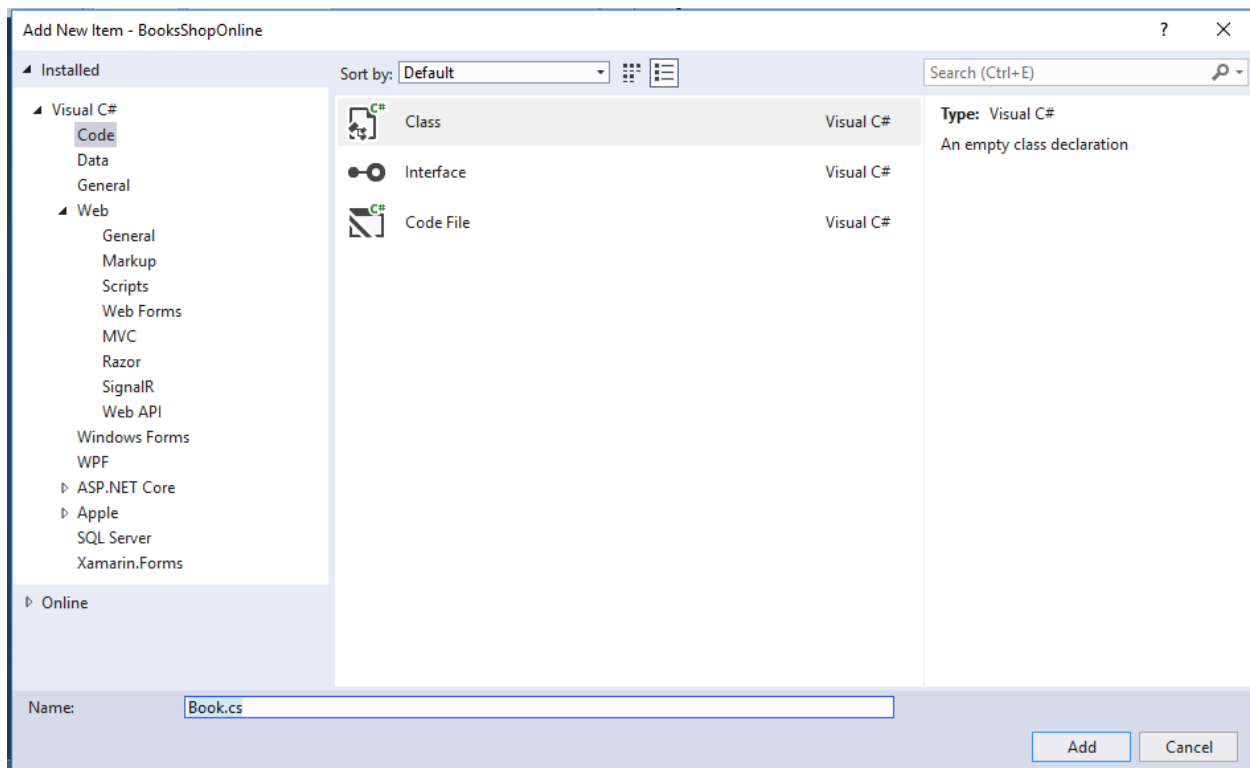
PC > OS (C:) > Program Files > Microsoft SQL Server > MSSQL11.SQLEXPRESS > MSSQL > DATA			
Name	Date modified	Type	Size
Album	27/11/2018 9:37 AM	SQL Server Databa...	3.072 KB
Album_log	27/11/2018 9:37 AM	SQL Server Databa...	1.024 KB
BooksShopOnline	27/11/2018 9:37 AM	SQL Server Databa...	3.072 KB
BooksShopOnline_log	27/11/2018 9:37 AM	SQL Server Databa...	1.024 KB
master	27/11/2018 9:37 AM	SQL Server Databa...	4.096 KB
mastlog	27/11/2018 9:37 AM	SQL Server Databa...	1.280 KB
model	27/11/2018 9:37 AM	SQL Server Databa...	2.112 KB
modellog	27/11/2018 9:37 AM	SQL Server Databa...	768 KB
MSDBData	27/11/2018 9:37 AM	SQL Server Databa...	14.080 KB
MSDBLog	27/11/2018 9:37 AM	SQL Server Databa...	768 KB
ReportServer\$SQLEXPRESS	27/11/2018 9:37 AM	SQL Server Databa...	5.184 KB
ReportServer\$SQLEXPRESS_log	27/11/2018 9:37 AM	SQL Server Databa...	5.824 KB
ReportServer\$SQLEXPRESSTempDB	27/11/2018 9:53 AM	SQL Server Databa...	4.160 KB
ReportServer\$SQLEXPRESSTempDB_log	27/11/2018 9:53 AM	SQL Server Databa...	1.088 KB
tempdb	27/11/2018 9:52 AM	SQL Server Databa...	3.136 KB
templog	27/11/2018 9:52 AM	SQL Server Databa...	512 KB
testDB	27/11/2018 9:37 AM	SQL Server Databa...	3.072 KB
testDB_log	27/11/2018 9:37 AM	SQL Server Databa...	1.024 KB

Tìm và sao chép tập tin BooksShopOnline.mdf đến vị trí mới.

## Tạo các lớp thực thể (Entity class)

### Tạo lớp Book (ánh xạ từ bảng Book)

- Tạo một thư mục tên Models trong dự án bằng cách nhấn chuột phải vào dự án BooksShopOnline trong cửa sổ Solution Explorer chọn Add > New Folder.
- Nhấn chuột phải vào thư mục Models chọn Add > New Item. Trong cửa sổ Add New Item chọn Visual C# > Code, chọn kiểu Class và gõ Book.cs trong Name và nhấn Add:



Trong lớp Book gõ nội dung sau:

```
public class Book
{
    [ScaffoldColumn(false)]
    public int BookID { get; set; }
    [Required, StringLength(100), Display(Name = "Name")]
    public string BookName { get; set; }
    [Required, StringLength(1000), Display(Name = "Book Description"),
    DataType(DataType.MultilineText)]
    public string Description { get; set; }
    public string ImagePath { get; set; }
    [Display(Name = "Price")]
    public float? UnitPrice { get; set; }
    public int? CategoryID { get; set; }
    public virtual Category Category { get; set; }
}
```

Ở đây chúng ta dùng các thuộc tính Data Annotations để mô tả các biến thành viên của lớp Book, ví dụ thuộc tính [ScaffoldColumn(false)] mô tả cho biến BookID. Khi vừa gõ các đoạn mã trên, có thể bị lỗi tại các thuộc tính Data Annotations như sau:



```

namespace BooksShopOnline.Models
{
    public class Book
    {
        [ScaffoldColumn(false)]
        public int BookID { get; set; }
        [Required, StringLength(100), Display(Name = "Name")]
        public string BookName { get; set; }
        [Required, StringLength(1000), Display(Name = "Book Description"),
        DataType(DataType.MultilineText)]
        public string Description { get; set; }
        public string ImagePath { get; set; }
        [Display(Name = "Price")]
        public float? UnitPrice { get; set; }
        public int? CategoryID { get; set; }
        public virtual Category Category { get; set; }
    }
}

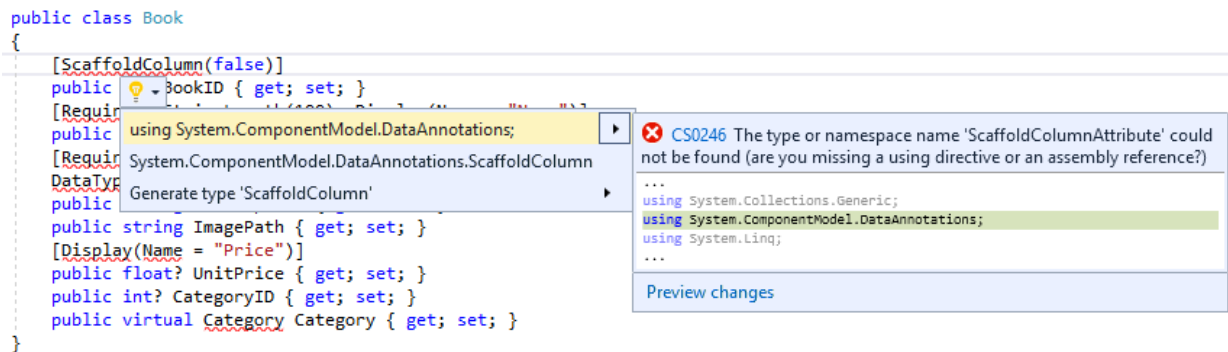
```

Lỗi này xảy ra do chúng ta chưa tham chiếu đến namespace

**System.ComponentModel.DataAnnotations**. Để tham chiếu đến namespace này chúng ta có thể dùng lệnh sau:

```
using System.ComponentModel.DataAnnotations;
```

Trong VS 2017, chúng ta có thể sửa lỗi bằng cách đưa con trỏ chuột vào dòng lệnh bị lỗi và nhấn tổ hợp Alt + Enter



```

public class Book
{
    [ScaffoldColumn(false)]
    public int BookID { get; set; }
    [Required, StringLength(100), Display(Name = "Name")]
    public string BookName { get; set; }
    [Required, StringLength(1000), Display(Name = "Book Description"),
    DataType(DataType.MultilineText)]
    public string Description { get; set; }
    public string ImagePath { get; set; }
    [Display(Name = "Price")]
    public float? UnitPrice { get; set; }
    public int? CategoryID { get; set; }
    public virtual Category Category { get; set; }
}

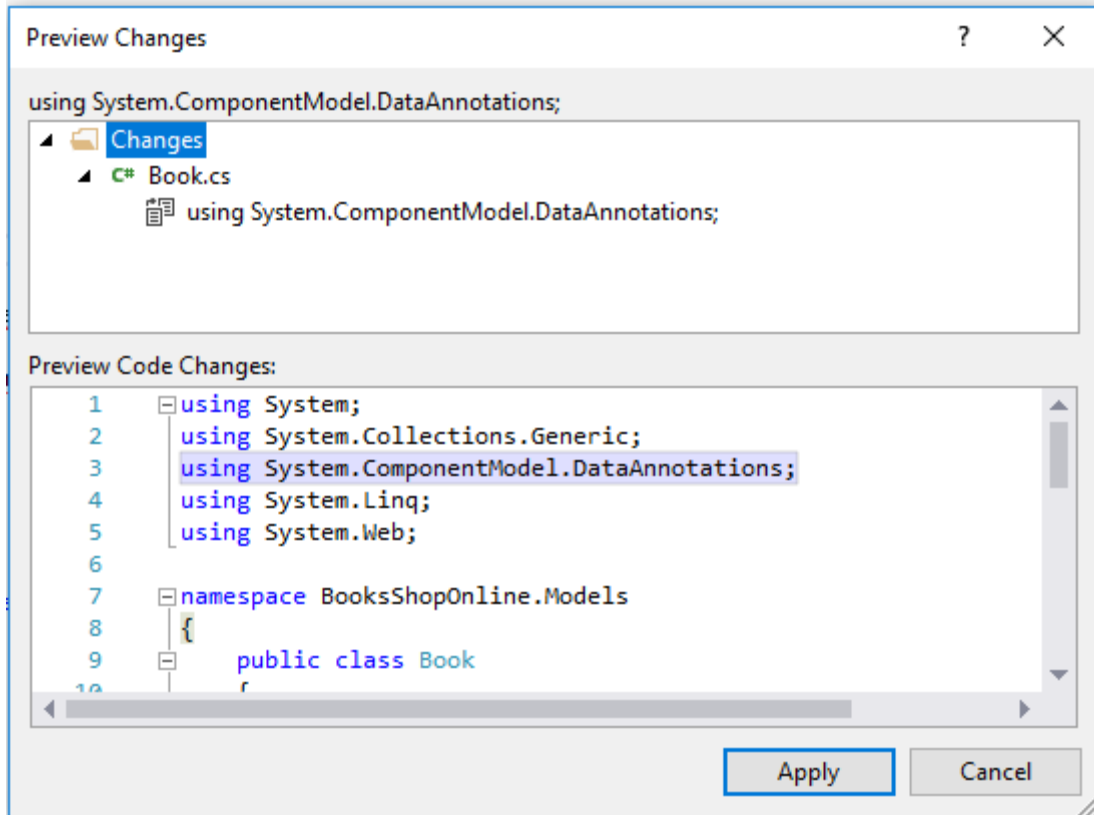
```

CS0246 The type or namespace name 'ScaffoldColumnAttribute' could not be found (are you missing a using directive or an assembly reference?)

- using System.Collections.Generic;
- using System.ComponentModel.DataAnnotations;
- using System.Linq;

Preview changes

Nhấn Preview changes



Nhấn Apply.

Lỗi xuất hiện tại dòng lệnh cuối do chúng ta chưa tạo lớp Category.

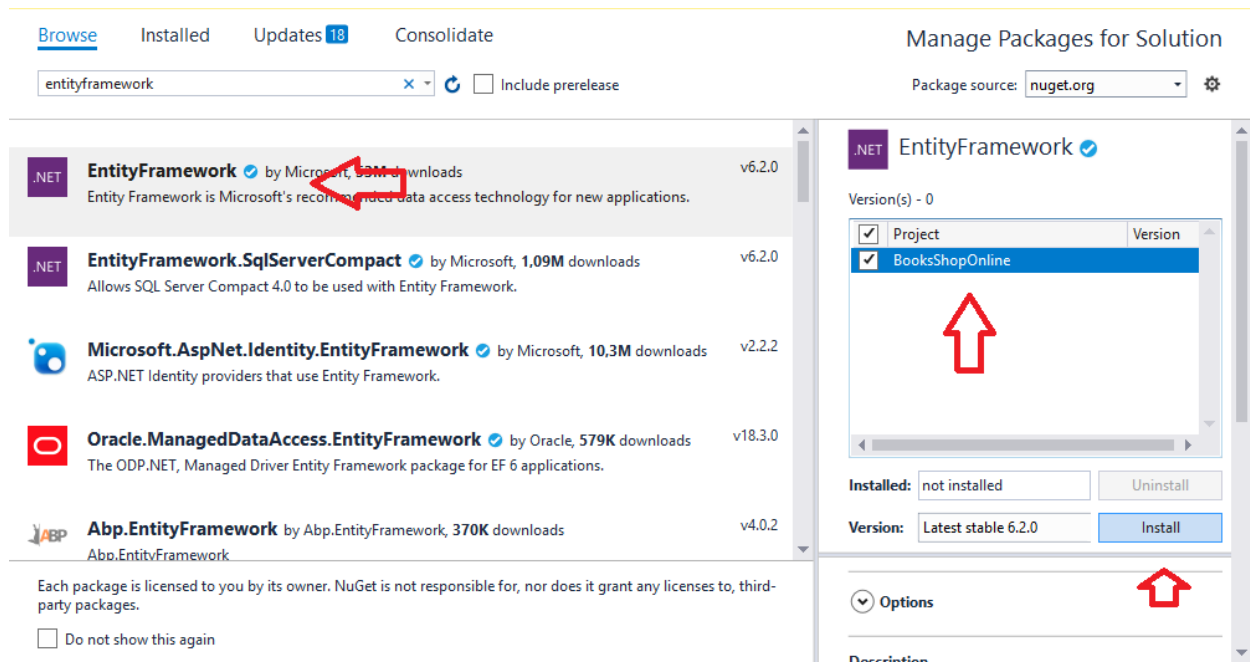
## Tạo lớp Category (ánh xạ từ bảng Category)

Tương tự lớp Book, chúng ta tạo lớp Category trong thư mục Models như sau:

```
public class Category
{
    [ScaffoldColumn(false)]
    public int CategoryID { get; set; }
    [Required, StringLength(100), Display(Name = "Name")]
    public string CategoryName { get; set; }
    public virtual ICollection<Book> Books { get; set; }
}
```

## Tạo lớp ngữ cảnh (context class)

Để có thể dùng các lớp truy cập dữ liệu, chúng ta cần định nghĩa các lớp ngữ cảnh. Lớp ngữ cảnh này quản lý hai lớp thực thể (Book và Category) và truy cập dữ liệu. Nhưng trước đó, chúng ta cần cài đặt Entity Framework bằng cách vào Tools > NuGet Package Manager > Manage NuGet Packages for Solution...(VS 2017). Chọn tab Browse và gõ 'entityframework' trong ô tìm kiếm, chọn EntityFramework, chọn BooksShopOnline và nhấn Install:



Nhấn OK trong Preview Changes và I Accept trong License Acceptance.

Trong thư mục Models tạo lớp tên BookContext tương tự như tạo các lớp Book và Category. Thay đổi nội dung của lớp BookContext như sau:

```
using System.Data.Entity;
namespace BooksShopOnline.Models
{
    public class BookContext : DbContext
    {
        public BookContext() : base("BooksShopOnline")
        { }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Book> Books { get; set; }
    }
}
```

Lớp ngữ cảnh BookContext, thừa kế từ lớp DbContext được cung cấp bởi Entity Framework, cho phép lưu trữ, cập nhật các thể hiện lớp Book trong cơ sở dữ liệu.

## Lớp khởi tạo (Initializer Class)

Vì chúng ta sử dụng phương thức Entity Framework Code First nên mọi thứ đều bắt đầu từ mã C#. Với lớp khởi tạo cho phép chúng ta khởi tạo cơ sở dữ liệu và các dữ liệu đầu tiên (còn gọi là seed data). Tạo lớp khởi tạo tên BookDatabaseInitializer trong thư mục Models

```
using System.Data.Entity;
namespace BooksShopOnline.Models
{
    public class BookDatabaseInitializer : DropCreateDatabaseAlways<BookContext>
    {
        protected override void Seed(BookContext context)
        {
        }
    }
}
```

```

{
    GetCategories().ForEach(c => context.Categories.Add(c));
    GetBooks().ForEach(p => context.Books.Add(p));
}
private static List<Category> GetCategories()
{
    var categories = new List<Category> {
        new Category
        {
            CategoryID = 1,
            CategoryName = "Fiction"
        },
        new Category
        {
            CategoryID = 2,
            CategoryName = "Biographies and Memoirs"
        },
        new Category{
            CategoryID = 3,
            CategoryName = "Biological Sciences"
        },
        new Category
        {
            CategoryID = 4,
            CategoryName = "Self-Help"
        }
    };
    return categories;
}
private static List<Book> GetBooks()
{
    var books = new List<Book> {
        //book 1
        new Book
        {
            BookID = 1,
            BookName = "Fire & Blood",
            Description = "300 Years Before A Game of Thrones (A Targaryen
History) (A Song of Ice and Fire).",
            ImagePath="Pic1.png",
            UnitPrice = 16.04f,
            CategoryID = 1
        },
        //book 2
        new Book
        {
            BookID = 2,
            BookName = "Benjamin Franklin: An American Life",
            Description = "In this authoritative and engrossing full-scale
biography, Walter Isaacson, " +
"bestselling author of Einstein and Steve Jobs, shows how the most
fascinating of " +
"America's founders helped define our national character.",
            ImagePath="Pic2.png",
            UnitPrice = 19.60f,
            CategoryID = 2
        },
        //book 3

```

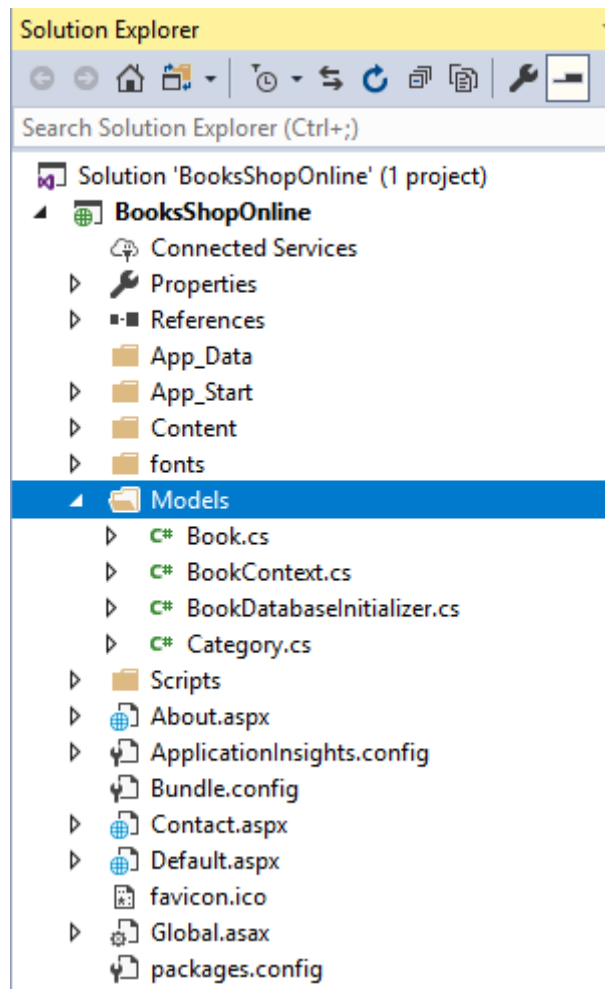
```

        new Book
        {
            BookID = 3,
            BookName = "Obama: An Intimate Portrait",
            Description = "During Barack Obama's two terms, Pete Souza was with
the President " +
            "during more crucial moments than anyone else--and he photographed
them all",
            ImagePath="Pic3.png",
            UnitPrice = 26.73f,
            CategoryID = 2
        },
        //book 4
        new Book
        {
            BookID = 4,
            BookName = "Sapiens: A Brief History of Humankind",
            Description = "One hundred thousand years ago, at least six different
species of " +
            "humans inhabited Earth. Yet today there is only one--homo sapiens. "
+
            "What happened to the others? And what may happen to us?",
            ImagePath="Pic4.png",
            UnitPrice = 23.79f,
            CategoryID = 3
        },
        //book 5
        new Book
        {
            BookID = 5,
            BookName = "The 7 Habits of Highly Effective People",
            Description = "Stephen Covey's cherished classic commemorates the
timeless wisdom and " +
            "power of the 7 Habits book, and does it in a highly readable and
understandable, infographics format.",
            ImagePath="Pic5.png",
            UnitPrice = 16.04f,
            CategoryID = 4
        },
    };
    return books;
}
}
}

```

Lớp khởi tạo BookDatabaseInitializer thực thi lớp DropCreateDatabaseAlways nên khi cơ sở dữ liệu được tạo nếu chúng ta cố gắng cập nhật dữ liệu đầu tiên (seed data) bằng cách điều chỉnh đoạn mã trên thì chúng ta sẽ không thấy bất kỳ sự thay đổi nào khi ứng dụng web đang thực thi. Nếu muốn cơ sở dữ liệu được tạo trở lại mỗi lần ứng dụng chạy, chúng ta có thể dùng lớp DropCreateDatabaseAlways.

Cho tới thời điểm này trong thư mục Models sẽ chứa các tập tin:



## Cấu hình ứng dụng để sử dụng mô hình dữ liệu

Sau khi tạo mô hình dữ liệu chúng ta sẽ cấu hình ứng dụng để sử dụng các lớp dữ liệu. Tập tin Global.asax được dùng để xử lý các sự kiện hay phương thức ứng dụng và là nơi chứa đoạn mã khởi tạo mô hình dữ liệu. Tập tin Web.config chứa các thông tin về cơ sở dữ liệu và ứng dụng web ASP.NET.

### Cập nhật tập tin Global.asax

Trong tập tin Global.asax.cs thêm các namespace và đoạn mã đến phương thức Application\_Start:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Optimization;
using System.Web.Routing;
using System.Web.Security;
using System.Web.SessionState;

using System.Data.Entity;
using BooksShopOnline.Models;
```

```

namespace BooksShopOnline
{
    public class Global : HttpApplication
    {
        void Application_Start(object sender, EventArgs e)
        {
            // Code that runs on application startup
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);

            // Initialize the book database.
            Database.SetInitializer(new BookDatabaseInitializer());
        }
    }
}

```

## Cập nhật tập tin Web.config

Trong tập tin Web.config chúng ta thêm chuỗi kết nối (connection string) để định vị cơ sở dữ liệu (BooksShopOnline.mdf) trong thư mục App\_Data như sau:

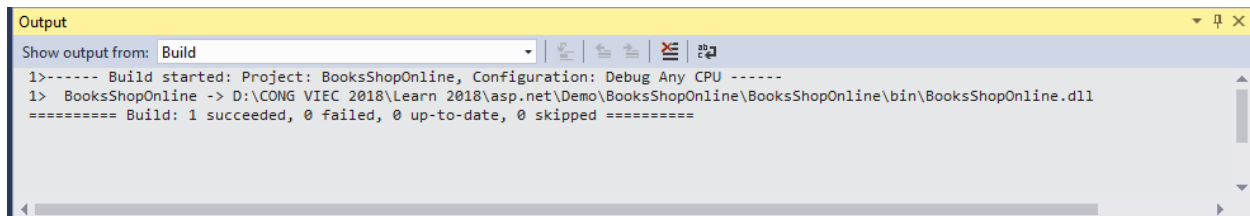
```

...
<connectionStrings>
    <add name="BooksShopOnline" connectionString="Data Source=.\SQLEXPRESS;
        AttachDbFilename=|DataDirectory|BooksShopOnline.mdf;
        Integrated Security=True;User Instance=true"
        providerName="System.Data.SqlClient" />
</connectionStrings>

</configuration>

```

Để đảm bảo các lớp làm việc, chúng ta khởi tạo ứng dụng bằng cách vào **Build > Build BooksShopOnline**



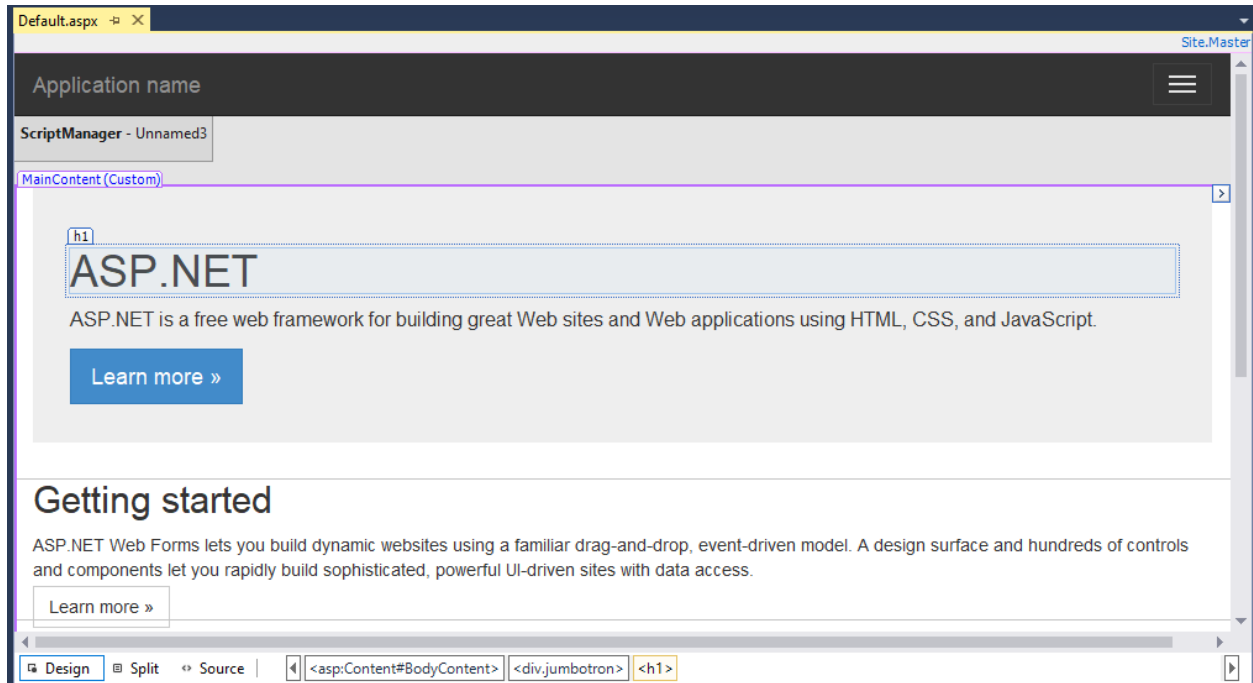
## Giao diện người dùng (UI) và điều hướng (Navigation)

ASP.NET cho phép tạo ra các nội dung động (dynamic content) cho ứng dụng Web. Một trang ASP.NET được tạo tương tự như một trang HTML tĩnh nhưng trong trang ASP.NET chứa các phần tử mở rộng để ASP.NET có thể nhận diện và xử lý để phát sinh mã HTML khi trang thực thi.

Khi có một yêu cầu đến trang HTML tĩnh (các tập tin .html hay .htm), máy chủ (server) sẽ đáp ứng yêu cầu bằng cách đọc tập tin và gửi nội dung đến trình duyệt (Web browser). Nếu có một yêu cầu đến trang ASP.NET (các tập tin .aspx), trang này sẽ thực thi như một chương trình trên Web server. Khi trang đang chạy, nó có thể thực thi bất cứ nhiệm vụ nào mà website yêu cầu như tính toán, đọc hay ghi cơ sở dữ liệu, gọi chương trình khác, v.v. Khi hoàn thành, trang sẽ tạo ra các mã HTML và gửi đến web browser.

## Thay đổi giao diện người dùng (UI)

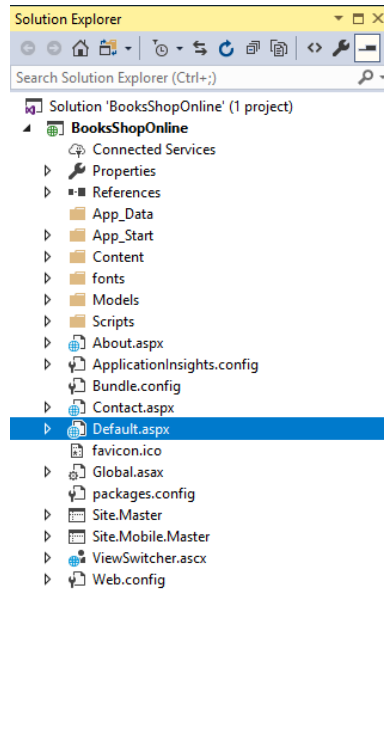
Khi chúng ta tạo một dự án ASP.NET WebForm, trang mặc định Default.aspx được tạo. Giao diện trông như sau (Tài liệu dùng Visual Studio 2017 Community):



Bây giờ chúng ta sẽ thực hiện thay đổi nội dung của trang **Default.aspx** phù hợp với mục đích của chúng ta là tạo một website **BooksShopOnline**. Các bước thực hiện:

1. Mở trang **Default.aspx** trong chế độ **Source**





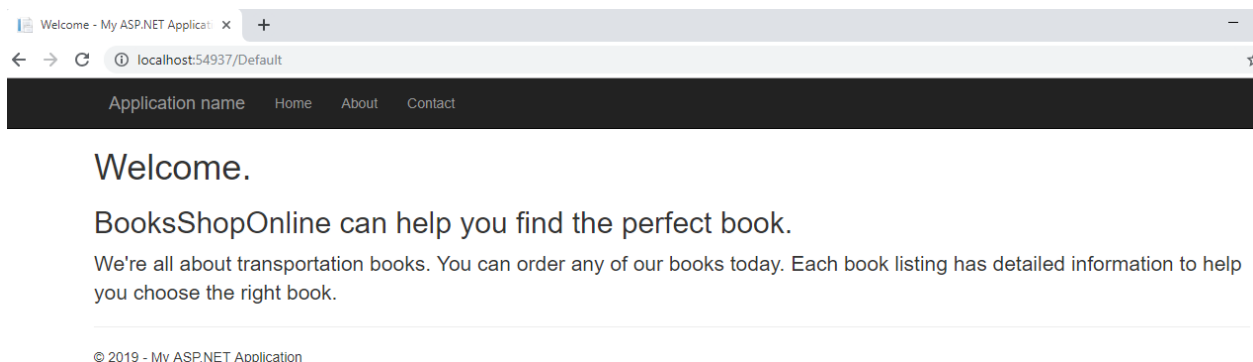
2. Tại đỉnh trang tìm đến chỉ thị @Page, thay đổi nội dung của thuộc tính Title đến Welcome:

```
<%@ Page Title="Welcome" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="BooksShopOnline._Default"
%>
```

3. Kế tiếp xóa toàn bộ nội dung trong cặp <asp:Content> và </asp:Content>, thay bằng nội dung sau:

```
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <h1><%: Title %>.</h1>
    <h2>BooksShopOnline can help you find the perfect book.</h2>
    <p class="lead">We're all about transportation books.
        You can order any of our books today.
        Each book listing has detailed information to help you choose the right book.</p>
</asp:Content>
```

4. Lưu trang Default.aspx và thực thi, kết quả:



Để ý rằng, khi nội dung được hiển thị trên trình duyệt, đoạn mã <%: Title %> sẽ lấy nội dung từ thuộc tính Title của chỉ thị @Page.

Trang Default.aspx chứa hầu hết các thành phần cấu tạo nên một trang ASP.NET. Các thành phần cơ bản bao gồm:

## Chỉ thị @Page

ASP.NET Web Forms chứa các chỉ thị (directives) cung cấp thông tin và các thuộc tính của trang ASP.NET. Các chỉ thị giống như các chỉ dẫn về cách xử lý các trang nhưng chúng không được chuyển thành các mã đánh dấu đến trình duyệt.

Chỉ thị dùng phổ biến nhất là @Page cung cấp một số thông tin cơ bản về trang gồm:

- Ngôn ngữ lập trình, ví dụ C#, qua thuộc tính Language
- Xác định trang chứa mã ngôn ngữ lập trình trực tiếp hay chứa trong một tập tin thông qua thuộc tính CodeBehind và thuộc tính Inherits xác định tên lớp trong tập tin mà trang kế thừa.
- Xác định liệu trang có kết nối với một trang master hay không thông qua thuộc tính MasterPageFile

Nếu chúng ta không dùng chỉ thị @Page trong trang hay không cung cấp thông tin cho chỉ thị thì các thông tin về trang sẽ được kế thừa từ tập tin Web.config hay từ Machine.config.

## Các điều khiển Web Server (Web Server controls)

Trong hầu hết các ứng dụng ASP.NET Web Forms, chúng ta luôn thêm vào các điều khiển cho phép người dùng tương tác với trang như textbox, button, checkbox, v.v. Các điều khiển Web Server tương tự các phần tử input hay button trong HTML. Tuy nhiên, các điều khiển Web Server được xử lý trên server và chúng ta có thể viết mã trên server để thiết lập thuộc tính cho các điều khiển này. Xử lý sự kiện cho các điều khiển Web Server cũng được thực hiện trên server.

Các thẻ cho điều khiển web server luôn bắt đầu bằng tiền tố **asp:** và chứa các thuộc tính **runat="server"** và thuộc tính **ID** dùng để tham chiếu đến điều khiển khi viết mã trên server. Ví dụ về một điều khiển button:

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

## Viết mã cho trang và các điều khiển trên server

ASP.NET Web Forms hỗ trợ các ngôn ngữ lập trình khác nhau như C#, Visual Basic, F#, v.v. và chúng ta có hai cách thức để viết mã cho trang.aspx. Cách thức thứ nhất là viết mã trực tiếp trong trang và cách thức thứ hai là viết mã trong một tập tin riêng biệt. Nếu chúng ta dùng cách thức thứ hai thì tập tin chứa mã được xác định trong thuộc tính CodeBehind và tên lớp mà trang kế thừa được xác định trong thuộc tính Inherits. Mã cho trang Default.aspx được chứa trong tập tin Default.aspx.cs:

```
<%@ Page Title="Welcome" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="BooksShopOnline._Default"
%>
```

## Các trang master (master pages)

Trong ASP.NET Web Forms, một trang master cho phép tạo sự nhất quán trong bố cục và cách ứng xử giữa các trang. Chúng ta có thể tạo ra các trang nội dung (content pages), ví dụ trang Default.aspx, thể hiện các nội dung khác nhau. Khi người dùng yêu cầu các trang nội dung, ASP.NET sẽ kết hợp chúng với trang master để tạo ra sự kết hợp giữa bố cục từ trang master và nội dung từ các trang content.

Một trang master là một tập tin ASP.NET có phần mở rộng là .master (trong ứng dụng BooksShopOnline của chúng ta là Site.Master) và sử dụng chỉ thị @Master thay vì @Page được dùng trong trang aspx.

Trong phần thực hành sau đây chúng ta sẽ thay đổi nội dung trang Site.Master của ứng dụng BooksShopOnline cho phù hợp với mục đích của chúng ta.

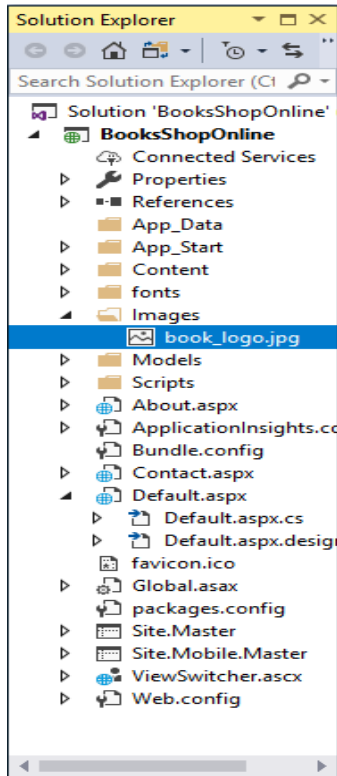
1. Trong cửa sổ Solution Explorer, tìm và mở tập tin Site.Master trong chế độ Source.
2. Thay thế các nội dung trong phần tử **title**, **a** và **footer** như sau:

```
...
<title><%: Page.Title %> - Books Shop Online</title>
...
<div class="navbar navbar-inverse navbar-fixed-top">
    ...
    <a class="navbar-brand" runat="server" href="~/>Books Shop Online</a>
    ...
</div>
...
<footer>
    <p>&copy; <%: DateTime.Now.Year %> - Books Shop Online</p>
</footer>
...
```

3. Thêm đoạn mã chèn logo đến website

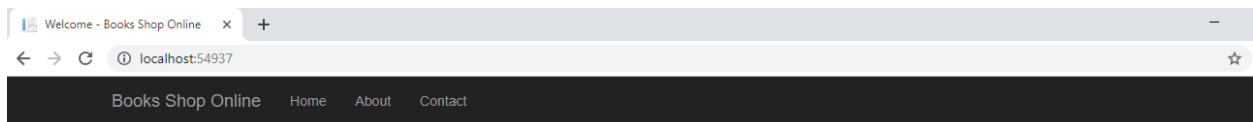
```
<div class="navbar navbar-inverse navbar-fixed-top">
    ...
</div>
<!--*****thêm logo*****-->
<div id="TitleContent" style="text-align: center">
    <a runat="server" href="~/>
    <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/book_logo.jpg"
    BorderStyle="None" />
    </a><br />
</div>
<!--*****-->
<div class="container body-content">
    ...
</div>
```

4. Thêm thư mục Images đến dự án bằng cách nhấn chuột phải vào BooksShopOnline chọn **Add > New Folder**. Kế tiếp là sao chép tập tin ảnh **book\_logo.jpg** đến thư mục **Images** vừa tạo



Lưu Tất cả những thay đổi trên **Site.Master**.

5. Mở lại trang Default.aspx và thực thi, kết quả:



*Books Shop Online*

Welcome.

BooksShopOnline can help you find the perfect book.

We're all about transportation books. You can order any of our books today. Each book listing has detailed information to help you choose the right book.

© 2019 - Books Shop Online

Dòng Books Shop Online màu đỏ chính là tập tin logo chúng ta vừa thêm vào.

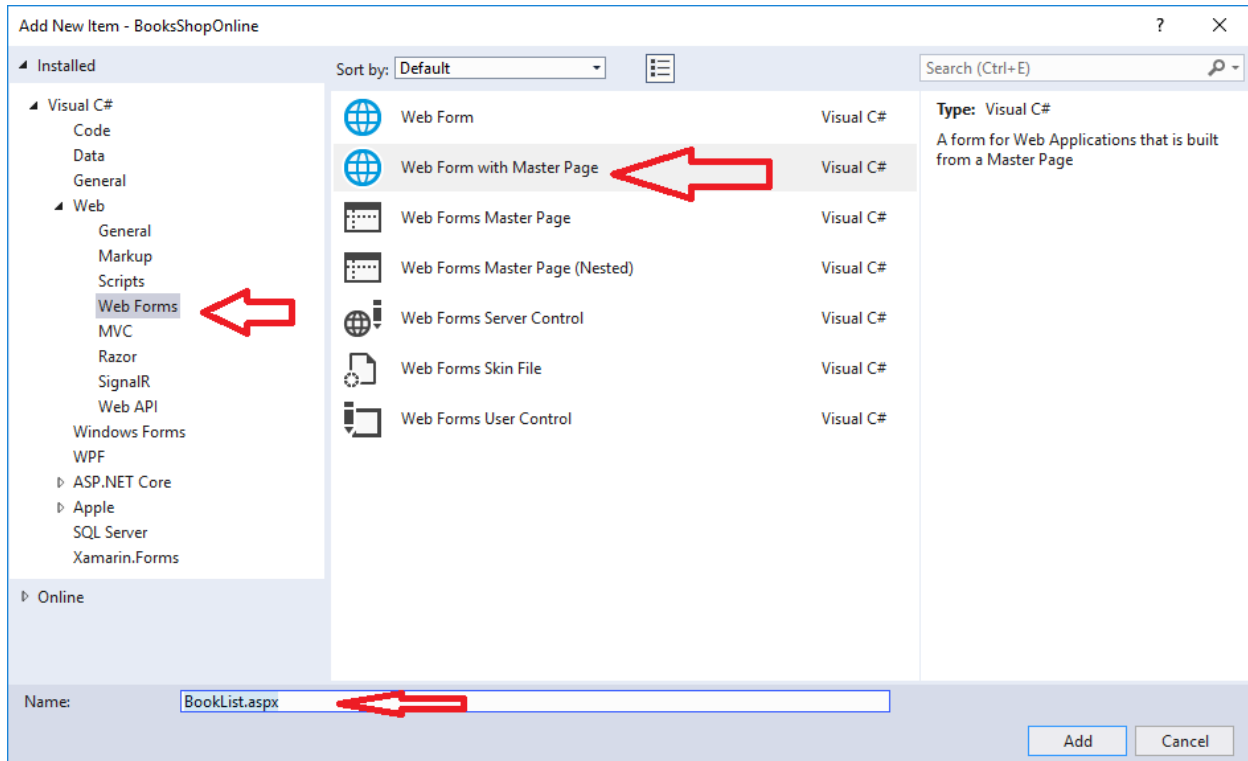
Thông qua tập tin **Site.Master** chúng ta thấy trang master chứa các phần tử HTML chủ yếu cho một tập tin HTML như <html>, <head>, <body>, <form>, <table>, <img> và các điều khiển server. Ngoài ra, trong trang master chứa một hay nhiều điều khiển **ContentPlaceHolder** dùng để chứa nội dung từ các trang content, ví dụ trang Default.aspx.

## Thêm các trang đến dự án

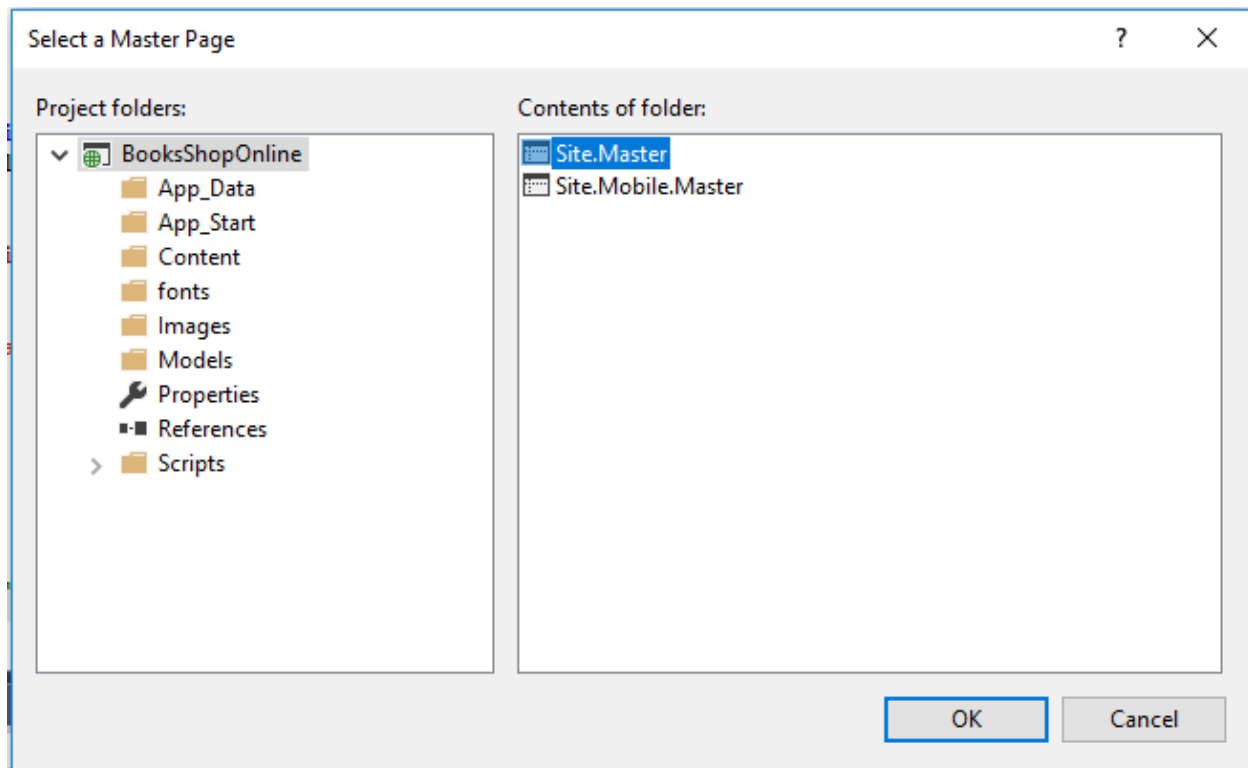
Bây giờ chúng ta thêm hai trang thể hiện các cuốn sách (BookList.aspx) và thông tin chi tiết về các cuốn sách (BookDetails.aspx). Cả hai trang này đều sử dụng bố cục từ trang master Site.Master.

Thêm trang BookList.aspx theo các bước sau:

1. Trong cửa sổ Solution Explorer nhấn chuột phải dự án BooksShopOnline chọn Add > New Item. Cửa sổ Add New Item xuất hiện, chọn các tùy chọn như trong hình và trong mục Name nhập tên trang là BookList.aspx:

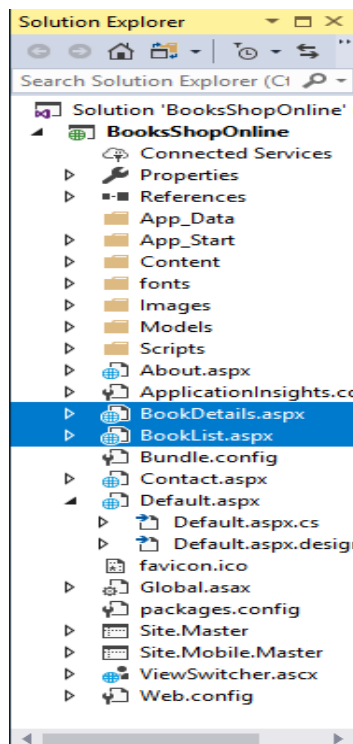


2. Nhấn nút Add. Trong cửa sổ Select a Master Page chọn Site.Master trong khung Contents of folder:



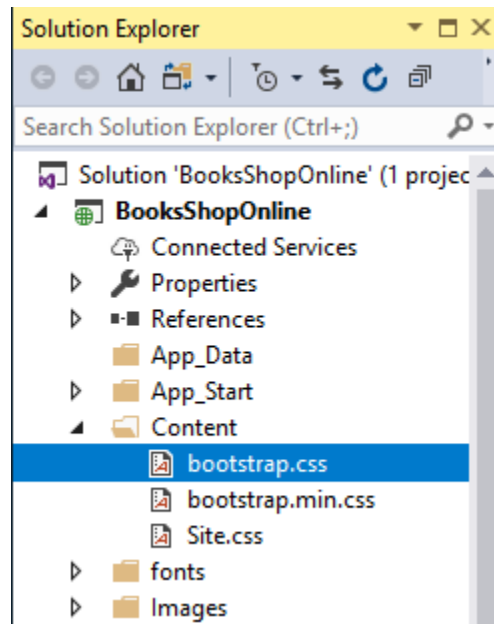
3. Nhấn OK.

Thêm trang BookDetails.aspx với các bước tương tự thêm trang BookList.aspx. Trong cửa sổ Solution Explorer lúc này:

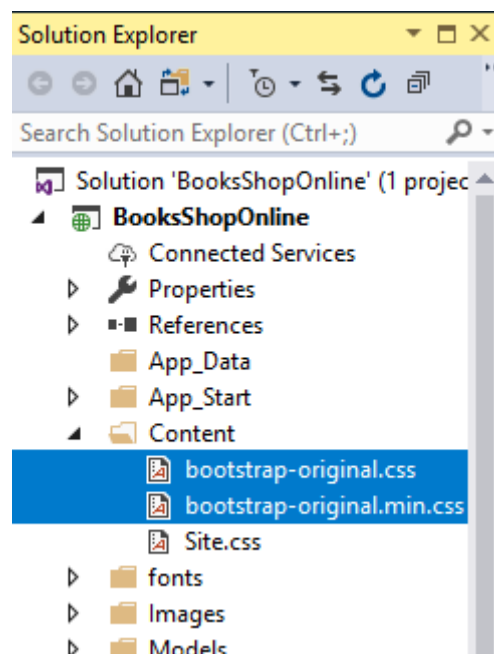


## Cập nhật Bootstrap

Mặc định, các ứng dụng ASP.NET sử dụng Bootstrap, một thư viện CSS được tạo bởi Twitter, để tạo bố cục hay giao diện cho các trang. Bootstrap được gộp trong Visual Studio như một gói Nuget. Các tập tin CSS của thư viện Bootstrap có thể được tìm thấy trong thư mục Content:

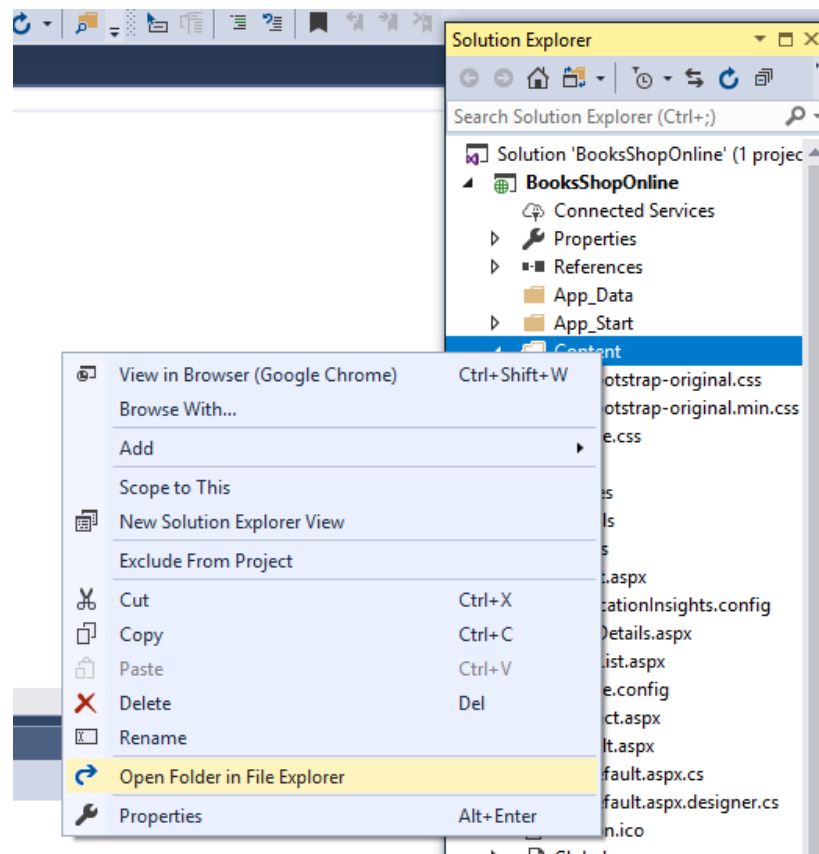


Trước khi cập nhật giao diện Bootstrap mới chúng ta sẽ thay đổi tên của tập tin bootstrap.css thành bootstrap-original.css và tên của bootstrap.min.css thành bootstrap-original.min.css:



Cập nhật thư viện Bootstrap theo các bước sau:

1. Trong cửa sổ Solution Explorer, nhấn chuột phải thư mục Content chọn Open Folder in File Explorer:

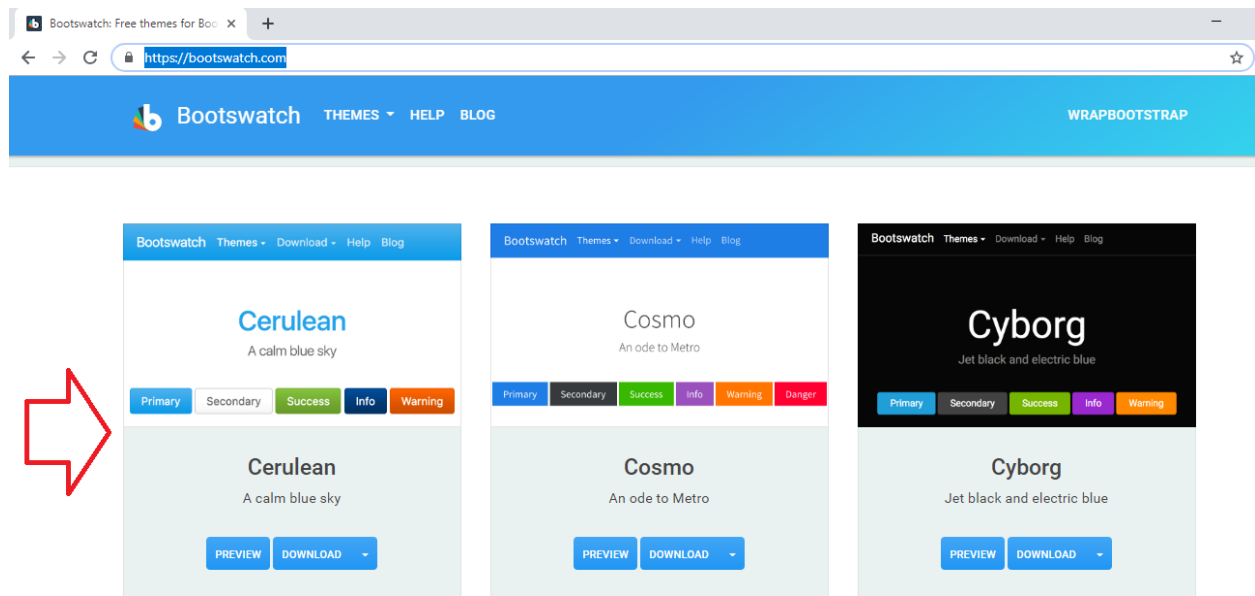


Cửa sổ File Explorer sẽ xuất hiện và chúng ta sẽ lưu các tập tin Bootstrap tại đây.

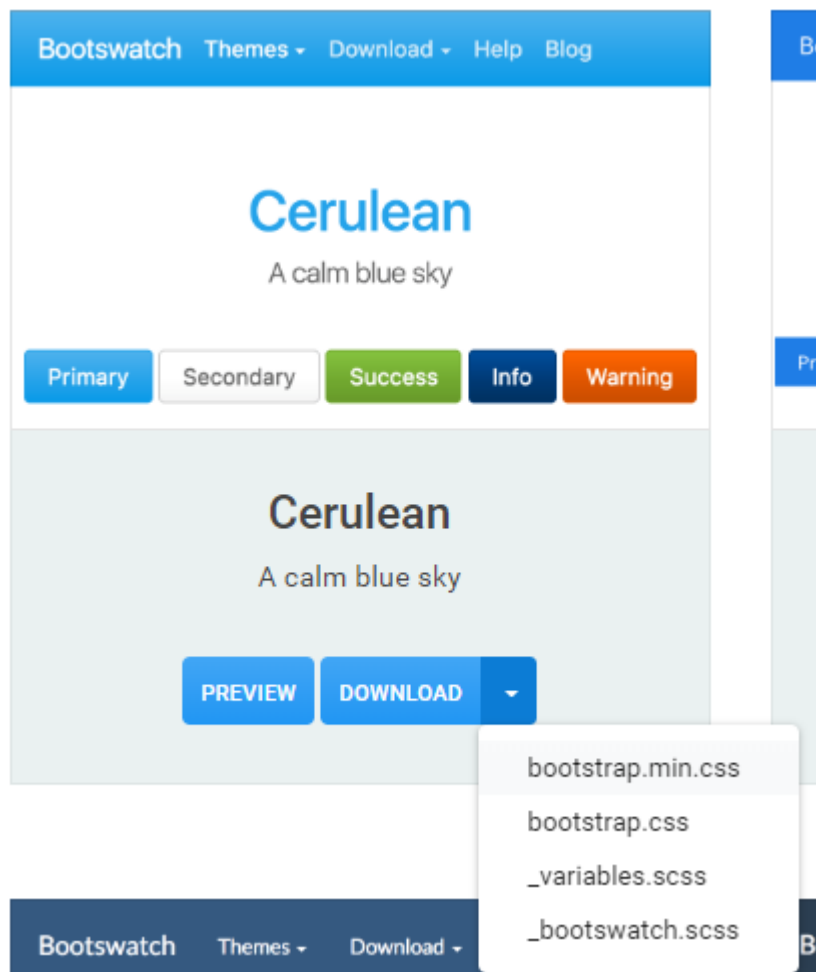
2. Trong trình duyệt web, mở trang <https://bootswatch.com/>

3. Cuộn cửa sổ trình duyệt tìm đến theme Cerulean:



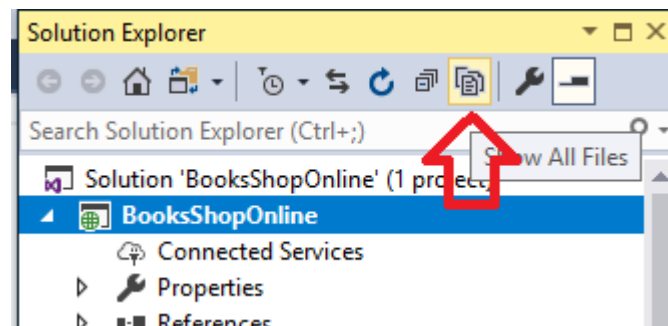


4. Nhấn nút mũi tên bên phải nút Download để tải các tập tin bootstrap.css và bootstrap.min.css:

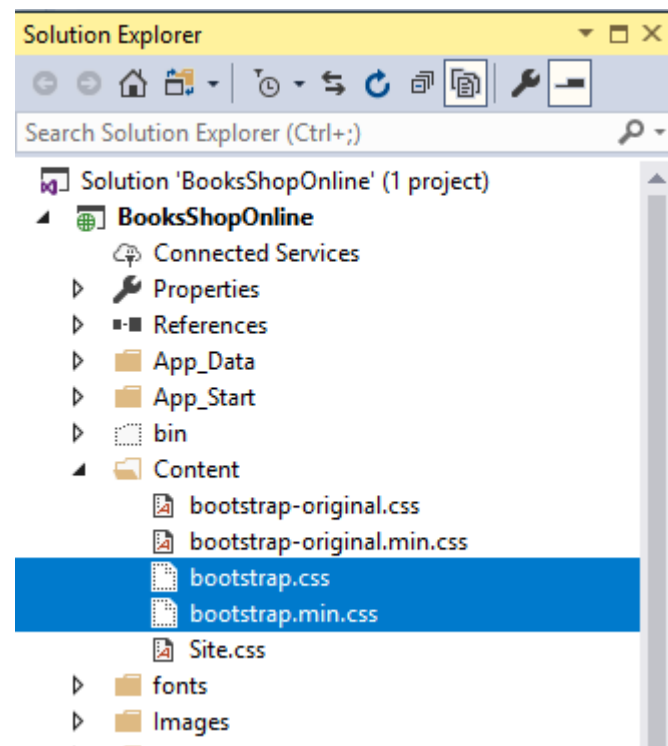


và lưu tại vị trí trong cửa sổ File Explorer (ở bước 1).

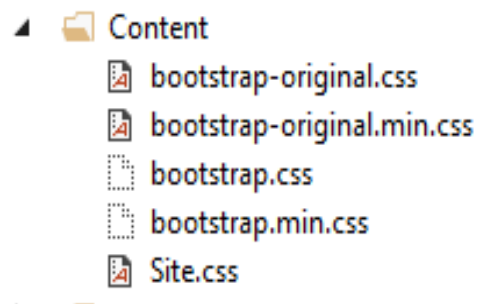
5. Trong cửa sổ Solution Explorer tìm và nhấn chuột vào biểu tượng Show All Files:



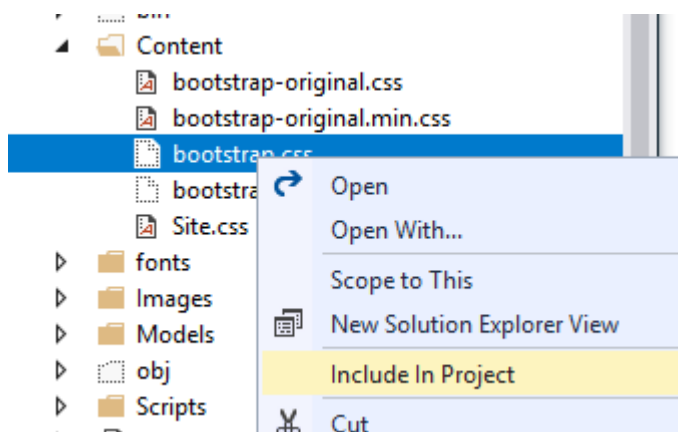
Lúc này các tập tin css chúng ta vừa tải đã xuất hiện trong thư mục Content:



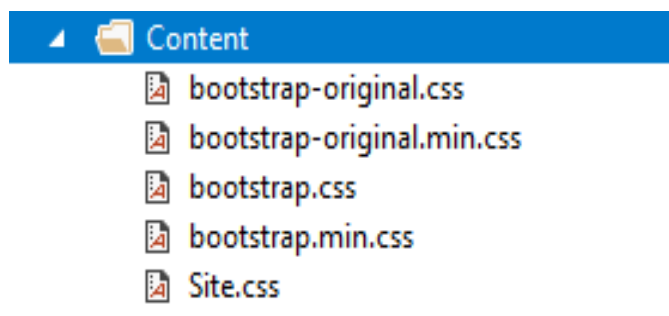
6. Tuy nhiên, chúng ta để ý rằng, hai biểu tượng của hai tập tin bootstrap.css và bootstrap.min.css có màu trắng có nghĩa rằng hai tập tin này chưa được gộp vào dự án:



Gộp hai tập tin css trên đến dự án bằng cách nhấn chuột phải vào tập tin bootstrap.css chọn Include In Project và nhấn chuột phải vào tập tin bootstrap.min.css chọn Include In Project:



Kết quả lúc này:



## Thay đổi thanh điều hướng mặc định (default navigation)

Thanh điều hướng mặc định của mỗi trang trong ứng dụng BooksShopOnline thừa kế mặc định từ trang Site.Master được tạo bởi phần tử ul, li và a. Đoạn mã có thể tìm thấy trong trang Site.Master ở chế độ Source như sau:

```
<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav">
    <li><a runat="server" href="~/>Home</a></li>
    <li><a runat="server" href="~/About">About</a></li>
    <li><a runat="server" href="~/Contact">Contact</a></li>
  </ul>
</div>
```

Biểu tượng ~ dùng để xác định đường dẫn từ thư mục gốc của dự án đến các trang About (About.aspx), trang Contact (Contact.aspx) và trang mặc định (Default.aspx). Trang mặc định không cần xác định tên trang.

Chúng ta sẽ thêm một mục mới đến thanh điều hướng tên Books liên kết đến trang BookList.aspx như sau:

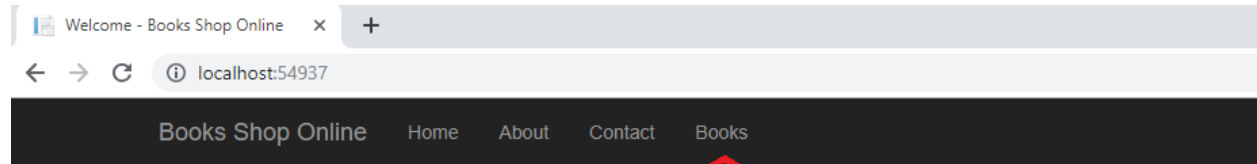
```
<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav">
    <li><a runat="server" href="~/>Home</a></li>
```

```

<li><a runat="server" href="~/About">About</a></li>
<li><a runat="server" href="~/Contact">Contact</a></li>
<li><a runat="server" href="~/BookList">Books</a></li>
</ul>
</div>

```

Nếu thực thi lại trang Default.aspx sẽ thấy mục Books trên thanh điều hướng:



*Books Shop Online*

Welcome.

BooksShopOnline can help you find the perfect book.

We're all about transportation books. You can order any of our books today. Each book list you choose the right book.

## Thêm điều khiển để hiển thị dữ liệu

Chúng ta sẽ thêm một điều khiển để hiển thị tất cả thể loại các cuốn sách từ cơ sở dữ liệu. Mỗi thể loại được xem như một liên kết đến trang BookList.aspx. Khi một người dùng chọn một thể loại bất kỳ, website sẽ định hướng người dùng đến trang hiển thị những cuốn sách thuộc thể loại đó.

Có nhiều điều khiển nhưng trong hướng dẫn này chúng ta sẽ dùng ListView để hiển thị các thể loại sách. Thêm một ListView đến trang master bằng cách thêm đoạn mã sau vào dưới khối <div id="TitleContent" ...>...</div>:

```

<div id="TitleContent" style="text-align: center">
    ...
</div>
<div id="CategoryMenu" style="text-align: center">
    <asp:ListView ID="categoryList" ItemType="BooksShopOnline.Models.Category"
    runat="server" SelectMethod="GetCategories" >
        <ItemTemplate>
            <b style="font-size: large; font-style: normal">
                <a href="/BookList.aspx?id=<#: Item.CategoryID %>">
                    <#: Item.CategoryName %>
                </a>
            </b>
        </ItemTemplate>
        <ItemSeparatorTemplate> | </ItemSeparatorTemplate>
    </asp:ListView>
</div>

```

Chúng ta sử dụng ListView với thuộc tính ItemType để kết nối cơ sở dữ liệu đến ListView và các thể loại sách từ cơ sở dữ liệu sẽ hiển thị ra ListView trong các ItemTemplate. Thông tin chi tiết mỗi thể loại (CategoryId và CategoryName) trong các ItemTemplate được thể hiện nhờ thuộc tính Item. Ký hiệu <%=...%> tương trưng cho biểu thức kết buộc dữ liệu (data-binding expression) với Item. Với dấu : sau <%, nội dung từ Item sẽ được mã hóa dưới dạng HTML.

## Liên kết điều khiển (ListView) đến cơ sở dữ liệu

Để liên kết điều khiển đến cơ sở dữ liệu, chúng ta thực hiện các bước:

- Trong cửa sổ Solution Explorer tìm đến trang Site.Master và nhấn chuột phải chọn View Code sẽ mở cửa sổ tập tin Site.Master.cs.

- Thêm phương thức GetCategory() (dưới phương thức mặc định Page\_Load) có nội dung sau:

```
protected void Page_Load(object sender, EventArgs e)
{
}

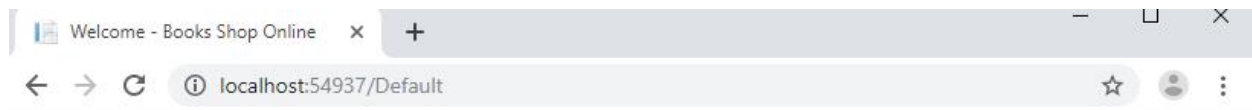
public IQueryable<Category> GetCategories()
{
    var _db = new BooksShopOnline.Models.BookContext();
    IQueryable<Category> query = _db.Categories;
    return query;
}
```

Thêm hai namespace sau đến đỉnh của tập tin Site.Master.cs:

```
using BooksShopOnline.Models;
using System.Linq;
```

Ở đây chúng ta dùng LINQ để truy vấn nội dung từ bảng Category của cơ sở dữ liệu.

Nếu lúc này chúng ta thực thi trang Default.aspx (trong Chrome) thì giao diện như sau:



Books Shop Online

## *Books Shop Online*

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

# Welcome.

## BooksShopOnline can help you find the perfect book.

We're all about transportation books. You can order any of our books today. Each book listing has detailed information to help you choose the right book.

---

© 2019 - Books Shop Online

Để ý chúng ta sẽ thấy một danh sách (chiều ngang) các thể loại sách như Fiction, Self\_Help,... Thanh thực đơn chính lúc này được thu gọn vào một biểu tượng bên trái tiêu đề Books Shop Online:



Books Shop Online

[Home](#)

[About](#)

[Contact](#)

[Books](#)

## Books Shop Online

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

# Welcome.

## Hiển thị chi tiết thông tin

Trong phần này chúng ta sẽ thể hiện chi tiết thông tin về các cuốn sách sử dụng Entity Framework Code First.

### Thêm một điều khiển để hiển thị các cuốn sách

Để kết buộc dữ liệu (data binding) đến một server control chúng ta dùng một trong 3 cách:

- Dùng các data source controls để kết buộc dữ liệu
- Viết mã truy cập cơ sở dữ liệu
- Dùng mô hình kết buộc dữ liệu và đây là cách chúng ta sẽ dùng để kết buộc dữ liệu cho website. Với cách này, chúng ta sẽ thiết lập giá trị thuộc tính SelectMethod đến tên phương thức hiển thị dữ liệu.

Thực hiện các bước sau để thay đổi nội dung trang BookList.aspx:

- Trong cửa sổ Solution Explorer, mở trang BookList.aspx
- Thay thế nội dung mặc định trong BookList.aspx bằng nội dung sau:

```
<%@ Page Title="Books" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true"
    CodeBehind="BookList.aspx.cs" Inherits="BooksShopOnline.BookList" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
    <section>
        <div>
            <hgroup>
                <h2><%: Page.Title %></h2>
            </hgroup>
```

```

        <asp:ListView ID="bookList" runat="server" DataKeyNames="BookID"
GroupItemCount="4"
        ItemType="BooksShopOnline.Models.Book" SelectMethod="GetBooks">
        <EmptyDataTemplate>
            <table >
                <tr>
                    <td>No data was returned.</td>
                </tr>
            </table>
        </EmptyDataTemplate>
        <EmptyItemTemplate>
            <td/>
        </EmptyItemTemplate>
        <GroupTemplate>
            <tr id="itemPlaceholderContainer" runat="server">
                <td id="itemPlaceholder" runat="server"></td>
            </tr>
        </GroupTemplate>
        <ItemTemplate>
            <td runat="server">
                <table>
                    <tr>
                        <td>
                            <a href="BookDetails.aspx?bookID=<%=Item.BookID%>">
                                <img src ="/Images/<%=Item.ImagePath%>"
                                    width="150" height="225"
style="border:solid" /></a>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <a href="BookDetails.aspx?bookID=<%=Item.BookID%>">
                                    <span>
                                        <%=Item.BookName%>
                                    </span>
                                </a>
                                <br />
                                <span>
                                    <b>Price:
</b><%=String.Format("{0:c}",Item.UnitPrice)%>
                                </span>
                                <br />
                            </td>
                        </tr>
                    </tr>
                </table>
            </td>
        </ItemTemplate>
        <LayoutTemplate>
            <table style="width:100%;">
                <tbody>
                    <tr>
                        <td>
                            <table id="groupPlaceholderContainer" runat="server"
style="width:100%">

```



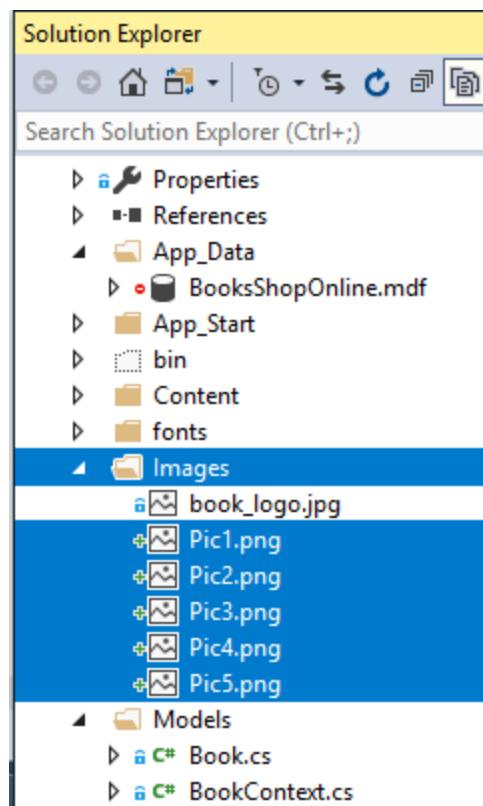
```

        <tr id="groupPlaceholder"></tr>
    </table>
</td>
</tr>
<tr>
    <td></td>
</tr>
<tr></tr>
</tbody>
</table>
</LayoutTemplate>
</asp:ListView>
</div>
</section>
</asp:Content>

```

Ở đây chúng ta đã thêm một ListView và thiết lập thuộc tính SelectMethod là GetBooks. Phương thức GetBooks sẽ được định nghĩa trong tập tin BookList.aspx.cs để lấy thông tin từ cơ sở dữ liệu.

Chúng ta cũng thêm một table để chứa hình ảnh và thông tin như tên, ID của các cuốn sách. Hình ảnh các cuốn sách được lưu trong thư mục Images trong cửa sổ Solution Explorer:



Để hiển thị cơ sở dữ liệu đến ListView, chúng ta cũng kết buộc ListView đến bảng tương ứng trong cơ sở dữ liệu bằng thuộc tính **ItemType**

```
ItemType="BooksShopOnline.Models.Book"
```

và hiển thị các cột của bảng thông qua thuộc tính **Item**

```

<td>
    <a href="BookDetails.aspx?bookID=<#:Item.BookID%>">
    <img src ="/Images/<#:Item.ImagePath%>"
        width="150" height="225" style="border:solid" /></a>
</td>

```

- Mở tập tin BookList.aspx.cs và định nghĩa phương thức GetBooks:

```

public partial class BookList : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    public IQueryable<Book> GetBooks([QueryString("id")] int?categoryId)
    {
        var _db = new BooksShopOnline.Models.BookContext();
        IQueryable<Book> query = _db.Books;
        if (categoryId.HasValue && categoryId > 0)
        {
            query = query.Where(p => p.CategoryID == categoryId);
        }
        return query;
    }
}

```

Để giới hạn kết quả theo một thể loại nhất định chúng ta thiết lập giá trị categoryId thông qua chuỗi truy vấn (query string) được chuyển đến trang BookList.aspx khi trang này được yêu cầu. Lớp QueryStringAttribute trong namespace System.Web.ModelBinding được dùng để nhận giá trị của biến chuỗi truy vấn (id). Câu truy vấn Linq được dùng để truy vấn các cuốn sách theo categoryId.

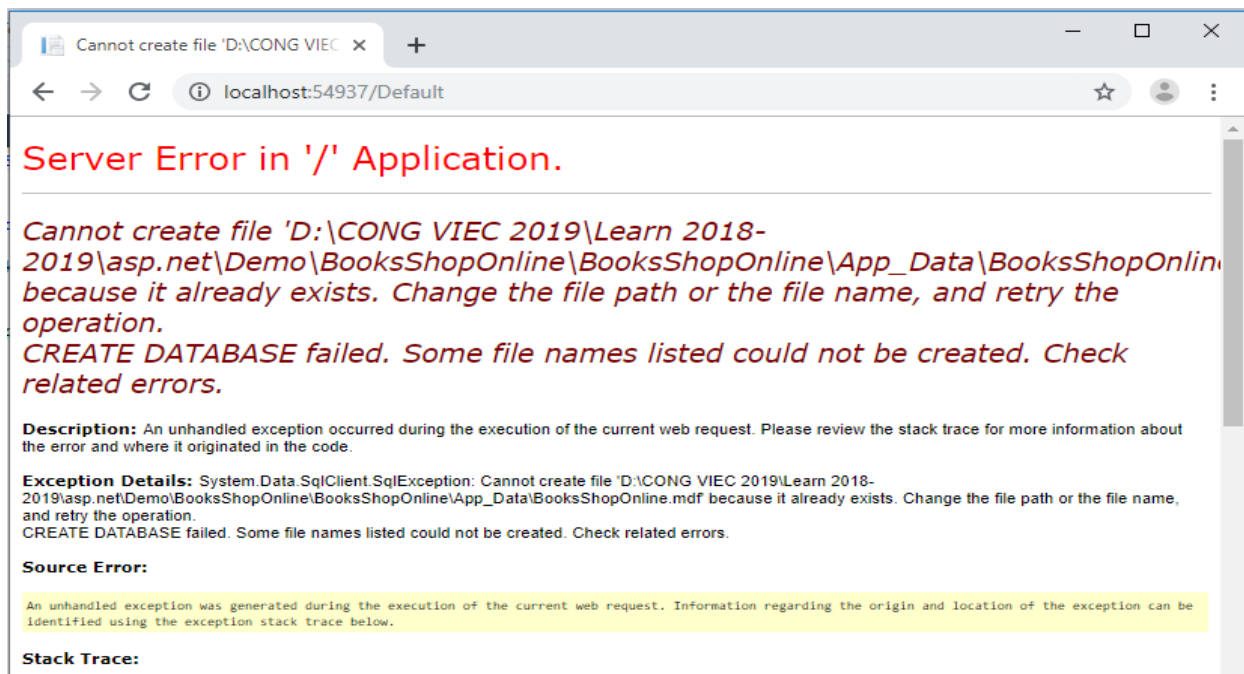
Trước khi chạy thử ứng dụng chúng ta cần thêm các khai báo các namespace:

```

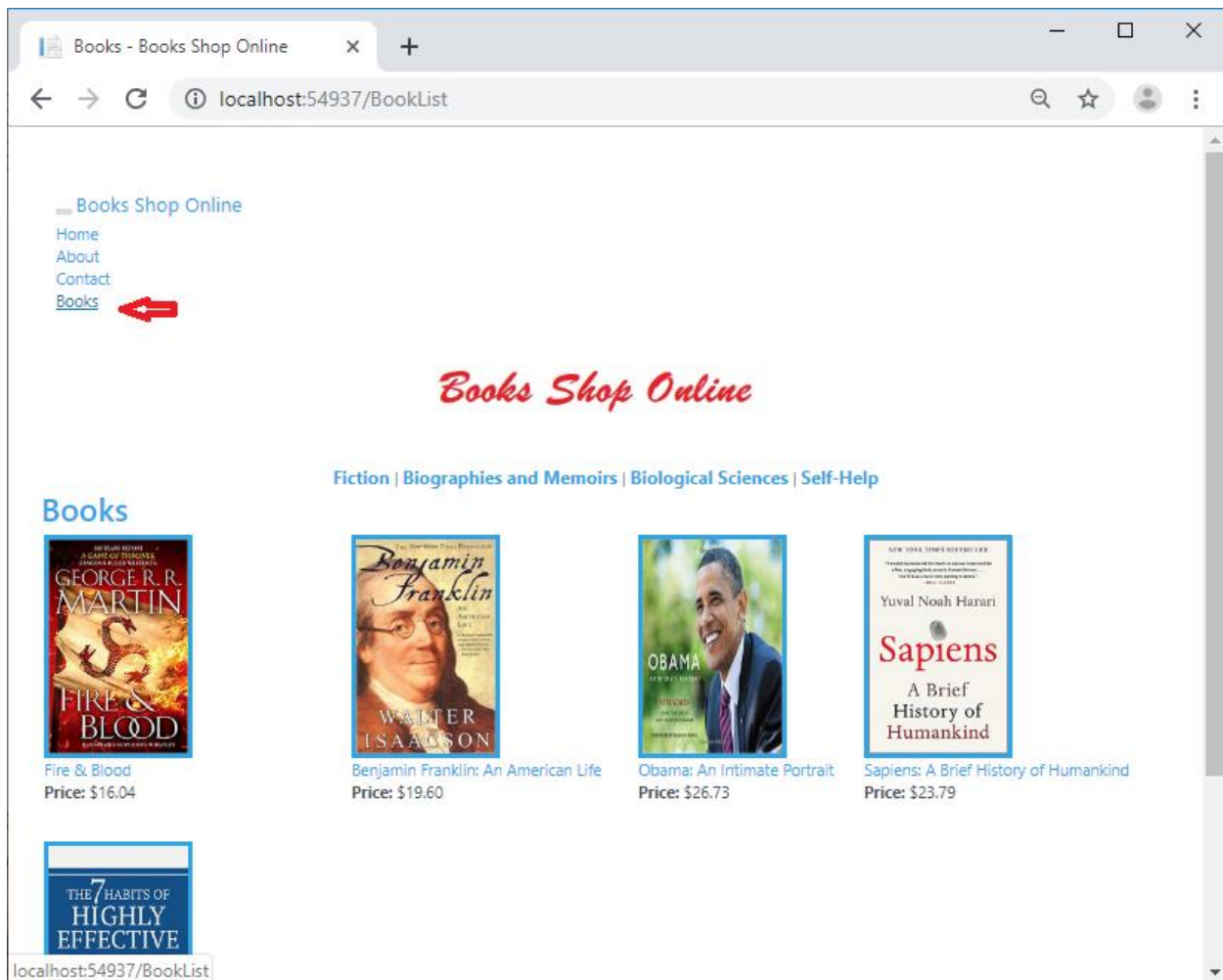
using System.Linq;
using BooksShopOnline.Models;
using System.Web.ModelBinding;

```

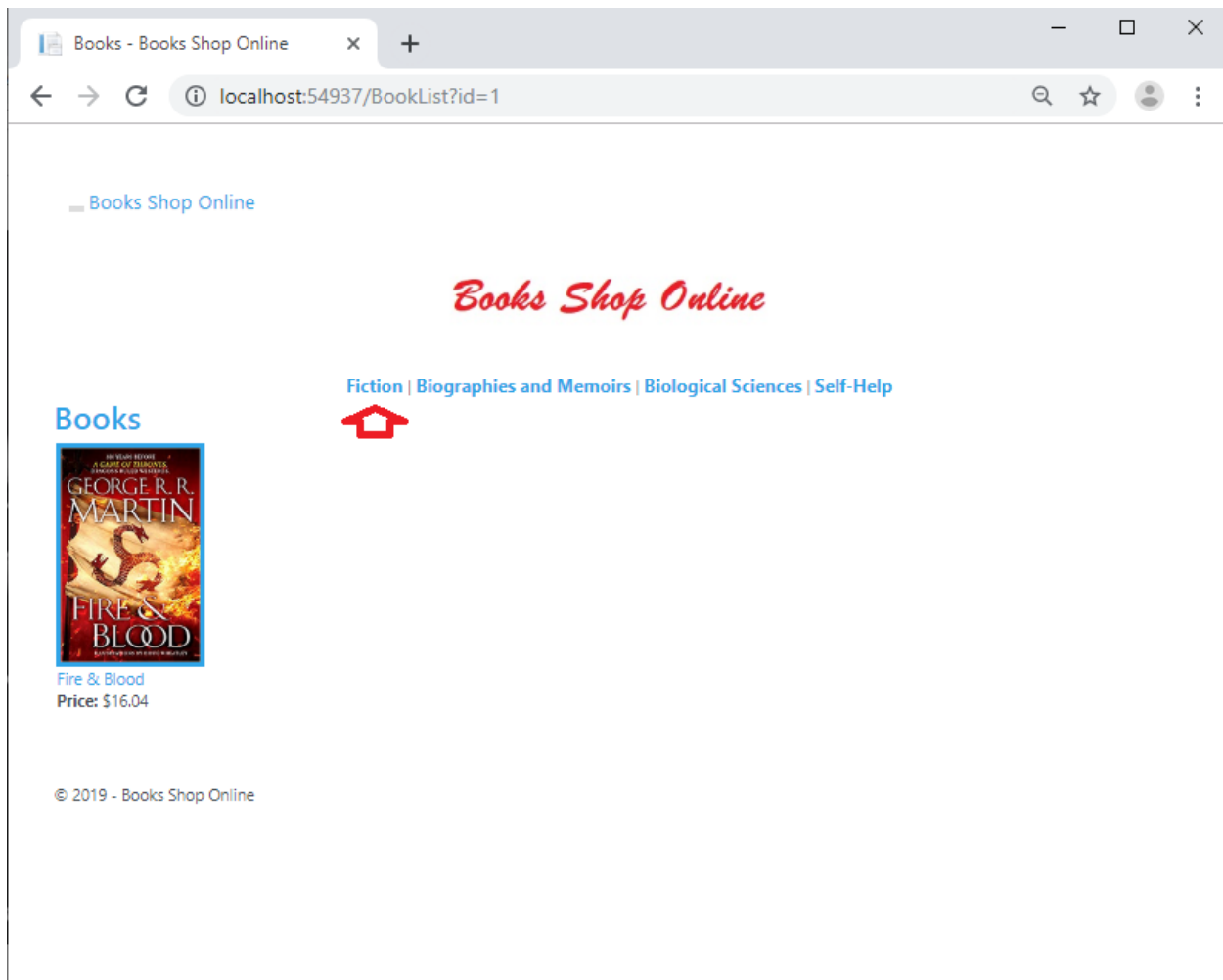
Lưu tất cả và thực thi lại trang Default.aspx, có thể xảy ra lỗi như sau:



Không cần lo lắng, chỉ việc tắt ứng dụng và chạy lại trang Default.aspx. Nhấn vào mục Books từ thực đơn sẽ xuất hiện các cuốn sách từ cơ sở dữ liệu:



Có thể xem các cuốn sách theo từng thể loại, ví dụ Fiction:



Tắt ứng dụng.

## Thêm một điều khiển để hiển thị chi tiết các cuốn sách

Sau khi đã hiển thị thành công các cuốn sách, bây giờ chúng ta sẽ hiển thị thông tin chi tiết cho mỗi cuốn sách trong trang BookDetails.aspx. Thay thế nội dung mặc định trong tập tin BookDetails.aspx bằng nội dung sau:

```
<%@ Page Title="Book Details" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="BookDetails.aspx.cs"
Inherits="BooksShopOnline.BookDetails" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
    <asp:FormView ID="bookDetail" runat="server" ItemType="BooksShopOnline.Models.Book"
    SelectMethod ="GetDetails"
        RenderOuterTable="false">
        <ItemTemplate>
            <div>
                <h1><%#:Item.BookName %></h1>
            </div>
            <br />
            <table>
                <tr>
```

```

        <td>
            "/>
        </td>
        <td>&nbsp;</td>
        <td style="vertical-align: top; text-align:left;">
            <b>Description:</b><br /><%#:Item.Description %>
            <br />
            <span><b>Price:</b>&nbsp;<%#:
String.Format("{0:c}",Item.UnitPrice) %></span>
            <br />
            <span><b>Book Number:</b>&nbsp;<%#:Item.BookID %></span>
            <br />
        </td>
    </tr>
</table>
</ItemTemplate>
</asp:FormView>
</asp:Content>

```

Lần này chúng ta dùng một điều khiển mới là FormView. Giống ListView, FormView cũng có phương thức SelectMethod tham chiếu đến phương thức hiển thị dữ liệu sẽ được định nghĩa trong tập tin BookDetails.aspx.cs (GetDetails). Chúng ta cũng sử dụng <table> kết hợp với Item để chứa thông tin các cuốn sách. Trong tập tin BookDetails.aspx.cs chúng ta định nghĩa phương thức GetDetails như sau:

```

public IQueryable<Book> GetDetails([QueryString("bookID")] int? bookId)
{
    var _db = new BooksShopOnline.Models.BookContext();
    IQueryable<Book> query = _db.Books;
    if (bookId.HasValue && bookId > 0)
    {
        query = query.Where(p => p.BookID == bookId);
    }
    else
    {
        query = null;
    }
    return query;
}

```

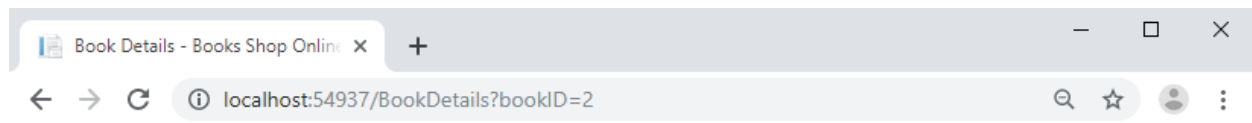
Để xem chi tiết mỗi cuốn sách theo BookID, chúng ta dùng lớp QueryStringAttribute trong namespace System.Web.ModelBinding và lệnh truy vấn Linq. Trước khi chạy lại trang Default.aspx chúng ta cần đảm bảo các namespace sau được khai báo:

```

using System.Linq;
using BooksShopOnline.Models;
using System.Web.ModelBinding;

```

Chạy trang Default.aspx. Nhấn chuột vào Books để xem tất cả các cuốn sách và nhấn chuột vào một cuốn sách bất kỳ để xem chi tiết:

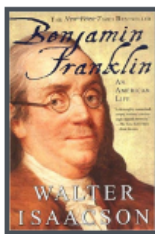


Books Shop Online

## Books Shop Online

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

### Benjamin Franklin: An American Life



**Description:**

In this authoritative and engrossing full-scale biography, Walter Isaacson, bestselling author of Einstein and Steve Jobs, shows how the most fascinating of America's founders helped define our national character.

**Price:** \$19.60

**Book Number:** 2

© 2019 - Books Shop Online

Tất ứng dụng. Thông tin về các cuốn sách đã được cung cấp đến người dùng. Bây giờ chúng ta sẽ xây dựng chức năng rất quan trọng cho một website thương mại – chức năng giỏ hàng (Shopping Cart).

## Tạo giỏ hàng (shopping cart)

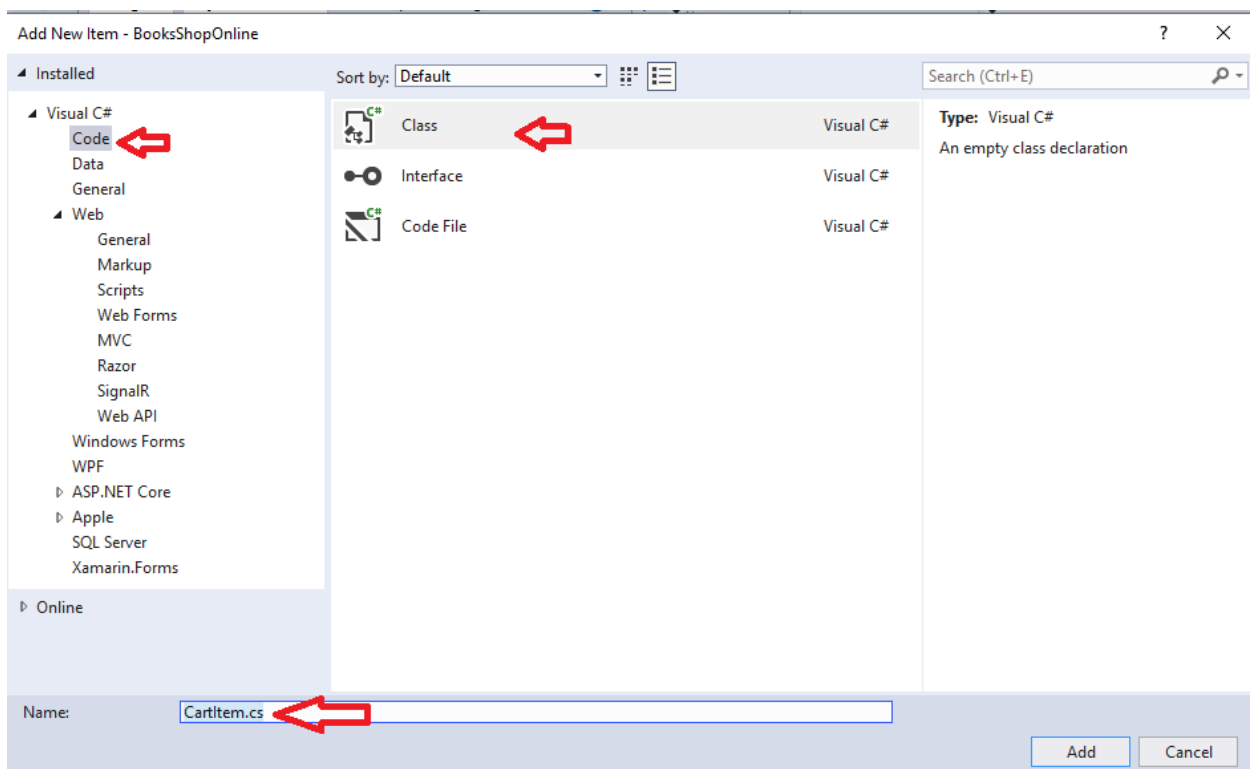
Trong phần này chúng ta sẽ tạo chức năng giỏ hàng giúp quản lý các cuốn sách mà người dùng thích mua. Người dùng có thể tìm và thêm các cuốn sách mình thích đến giỏ hàng ngay cả khi chưa là thành viên hay chưa đăng nhập vào website.

Để quản lý việc truy cập đến giỏ hàng, người dùng sẽ được gán một ID duy nhất còn gọi là GUID (Globally Unique Identifier) cho lần truy cập đầu tiên đến giỏ hàng. Các ID sẽ được lưu trữ nhờ ASP.NET Session.

### Thêm lớp mô hình - CartItem

Trong các phần trên chúng ta đã thêm các lớp mô hình Category và Book đến thư mục Models. Trong phần này chúng ta sẽ thêm một lớp mới gọi là CartItem để định nghĩa lược đồ cho giỏ hàng. Các bước thêm lớp CartItem đến thư mục Models:

- Trong cửa sổ Solution Explorer tìm đến thư mục Models
- Nhấn chuột phải vào thư mục Models chọn Add > New Item
- Trong hộp thoại Add New Item chọn Code, chọn Class và đổi tên Class là CartItem:



- Nhấn Add để thêm lớp CartItem đến thư mục Models. Nội dung của tập tin CartItem mặc định như sau:

```
using System;  
using System.Collections.Generic;
```



```
using System.Linq;
using System.Web;

namespace BooksShopOnline.Models
{
    public class CartItem
    {
    }
}
```

Thêm các thành viên đến lớp CartItem để định hình lược đồ cho giỏ hàng như sau:

```
public class CartItem
{
    [Key]
    public string ItemId { get; set; }
    public string CartId { get; set; }
    public int Quantity { get; set; }
    public System.DateTime DateCreated { get; set; }
    public int BookId { get; set; }
    public virtual Book Book { get; set; }
}
```

Chúng ta dùng thuộc tính [Key] để quy ước ItemId là khóa chính cho lược đồ CartItem. Đây là tính năng Data Annotations của EF Code First nên cần phải khai báo namespace sau:

```
using System.ComponentModel.DataAnnotations;
```

CartId là ID gán đến người dùng kết hợp với sản phẩm mà người dùng muốn mua. Chúng ta sẽ viết mã để tạo ra ID này khi người dùng truy cập đến giỏ hàng. ID này được lưu trữ trong biến ASP.NET Session. Quantity là số lượng cuốn sách người dùng thêm vào giỏ hàng, DateCreated là thời điểm người dùng truy cập giỏ hàng và BookId là thông tin về các cuốn sách có trong giỏ hàng.

## Tạo lớp ngữ cảnh

Giống các lớp Category và Book, chúng ta cần cập nhật lớp ngữ cảnh để quản lý các lớp thực thể và cung cấp cách thức truy cập dữ liệu đến cơ sở dữ liệu. Để làm điều này chúng ta cần thêm một lớp mới, gọi là ShoppingCartItems, đến lớp ngữ cảnh BookContext (tập tin BookContext.cs trong thư mục Models) như sau:

```
using System.Data.Entity;
namespace BooksShopOnline.Models
{
    public class BookContext : DbContext
    {
        public BookContext() : base("BooksShopOnline")
        { }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Book> Books { get; set; }
        public DbSet<CartItem> ShoppingCartItems { get; set; }
    }
}
```

## Xây dựng tính năng giỏ hàng

Giỏ hàng cần phải đảm bảo các tính năng:

- Thêm sản phẩm đến giỏ hàng
- Xóa sản phẩm khỏi giỏ hàng
- Nhận ID người dùng
- Nhận các sản phẩm từ giỏ hàng
- Thống kê tất cả các sản phẩm từ giỏ hàng
- Cập nhật thông tin giỏ hàng

Để giải quyết các vấn đề trên, chúng ta cần tạo một lớp mới gọi là ShoppingCart trong một thư mục mới gọi là Logic. Lớp này xử lý dữ liệu truy cập đến bảng CartItem. Chúng ta sẽ tạo một trang mới tên ShoppingCart.aspx để hiển thị tất cả các cuốn sách người dùng thêm vào giỏ hàng. Trang này và lớp ShoppingCart sẽ cùng truy cập đến dữ liệu giỏ hàng.

Chúng ta cũng tạo thêm một trang mới tên AddToCart.aspx để thêm các cuốn sách đến giỏ hàng. Để tiện lợi cho người dùng, chúng ta cũng cần thêm một liên kết (tên Add To Cart) đến các trang BookList.aspx và BookDetails.aspx để người dùng có thể thêm các cuốn sách đến giỏ hàng của mình. Khi người dùng nhấn chuột vào liên kết Add To Cart trên trang BookList.aspx hay trang BookDetails.aspx thì ứng dụng sẽ điều hướng đến trang AddToCart.aspx và sẽ tự động chuyển đến trang ShoppingCart.aspx. Trang AddToCart.aspx sẽ thêm các cuốn sách đến giỏ hàng bằng cách gọi một phương thức từ lớp ShoppingCart và trang ShoppingCart.aspx sẽ hiển thị các cuốn sách được thêm đến giỏ hàng:



## Tạo lớp ShoppingCart

Lớp ShoppingCart sẽ được thêm vào thư mục tên Logic như sau:

1. Trong cửa sổ Solution Explorer, nhấn chuột phải vào BooksShopOnline và chọn Add > New Folder và gõ tên thư mục là Logic.
2. Nhấn chuột phải vào thư mục Logic chọn Add > New Item và thêm một lớp mới tên ShoppingCartAction.cs (lưu ý chọn Visual C# > Code) và thay đổi nội dung mặc định bằng nội dung sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```

using BooksShopOnline.Models;

namespace BooksShopOnline.Logic
{
    public class ShoppingCartActions : IDisposable
    {
        public string ShoppingCartId { get; set; }
        private BookContext _db = new BookContext();
        public const string CartSessionKey = "CartId";
        public void AddToCart(int id)
        {
            // Retrieve the product from the database.
            ShoppingCartId = GetCartId();
            var cartItem = _db.ShoppingCartItems.SingleOrDefault(
                c => c.CartId == ShoppingCartId
                && c.BookId == id);
            if (cartItem == null)
            {
                // Create a new cart item if no cart item exists.
                cartItem = new CartItem
                {
                    ItemId = Guid.NewGuid().ToString(),
                    BookId = id,
                    CartId = ShoppingCartId,
                    Book = _db.Books.SingleOrDefault(
                        p => p.BookID == id),
                    Quantity = 1,
                    DateCreated = DateTime.Now
                };
                _db.ShoppingCartItems.Add(cartItem);
            }
            else
            {
                // If the item does exist in the cart,
                // then add one to the quantity.
                cartItem.Quantity++;
            }
            _db.SaveChanges();
        }
        public void Dispose()
        {
            if (_db != null)
            {
                _db.Dispose();
                _db = null;
            }
        }
        public string GetCartId()
        {
            if (HttpContext.Current.Session[CartSessionKey] == null)
            {
                if (!string.IsNullOrEmpty(HttpContext.Current.User.Identity.Name))
                {
                    HttpContext.Current.Session[CartSessionKey] =
                        HttpContext.Current.User.Identity.Name;
                }
            }
            else
            {
            }
        }
    }
}

```

```

        // Generate a new random GUID using System.Guid class.
        Guid tempCartId = Guid.NewGuid();
        HttpContext.Current.Session[CartSessionKey] = tempCartId.ToString();
    }
}
return HttpContext.Current.Session[CartSessionKey].ToString();
}
public List<CartItem> GetCartItems()
{
    ShoppingCartId = GetCartId();
    return _db.ShoppingCartItems.Where(
        c => c.CartId == ShoppingCartId).ToList();
}
}
}

```

Phương thức AddToCart cho phép thêm các cuốn sách đến giỏ hàng dựa theo ID của các cuốn sách. Nếu cuốn sách được thêm mà đã tồn tại trong giỏ hàng thì trường Quantity của CardItem sẽ được tăng lên một đơn vị.

Phương thức GetCardId trả về ID giỏ hàng cho người dùng. ID này dùng để theo dõi thông tin giỏ hàng của một người dùng nào đó. Nếu người dùng chưa có ID giỏ hàng thì một ID mới được tạo. Ở đây chúng ta cũng cần phân biệt người dùng đã đăng ký và chưa đăng ký vào hệ thống. Nếu người dùng đã đăng ký vào hệ thống thì ID giỏ hàng sẽ được thiết lập đến tên người dùng (user name), ngược lại, ID giỏ hàng sẽ được gán đến một giá trị duy nhất gọi là GUI dựa trên các phiên (session).

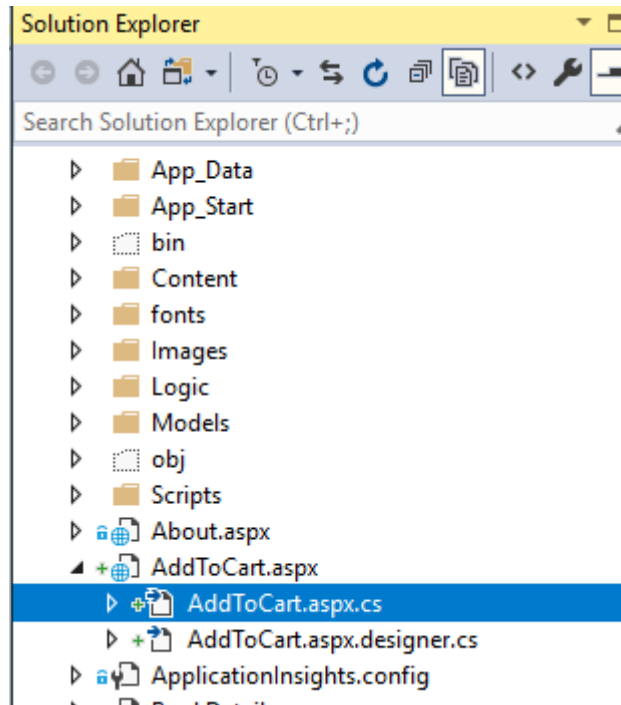
Phương thức GetCardItems trả về danh sách các cuốn sách trong giỏ hàng đến người dùng.

## Thêm cuốn sách đến giỏ hàng

Như trên đã đề cập, chúng ta sẽ tạo trang AddToCart.aspx để thêm các cuốn sách đến giỏ hàng của người dùng bằng cách gọi phương thức AddToCart từ lớp ShoppingCart dựa trên ID cuốn sách mà người dùng nhập từ trang AddToCart.aspx.

Cách thực hiện như sau:

1. Trong cửa sổ Solution Explorer, nhấn chuột phải vào BooksShopOnline chọn Add > New Item
2. Trong cửa sổ Add New Item chọn WebForm và gõ tên trang là AddToCart.aspx và nhấn Add
3. Trong cửa sổ Solution Explorer, tìm đến trang AddToCart.aspx, mở rộng và chọn AddToCart.aspx.cs:



Thay thế toàn bộ nội dung của tập tin AddToCart.aspx.cs bằng nội dung sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Diagnostics;
using BooksShopOnline.Logic;

namespace BooksShopOnline
{
    public partial class AddToCard : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string rawId = Request.QueryString["BookID"];
            int bookId;
            if (!String.IsNullOrEmpty(rawId) && int.TryParse(rawId, out bookId))
            {
                using (ShoppingCartActions usersShoppingCart = new
                    ShoppingCartActions())
                {
                    usersShoppingCart.AddToCart(Convert.ToInt16(rawId));
                }
            }
            else
            {
                Debug.Fail("ERROR : We should never get to AddToCart.aspx without a
BookId.");
                throw new Exception("ERROR : It is illegal to load AddToCart.aspx without
setting a BookId.");
            }
        }
    }
}
```

```

}
    Response.Redirect("ShoppingCart.aspx");
}
}
}

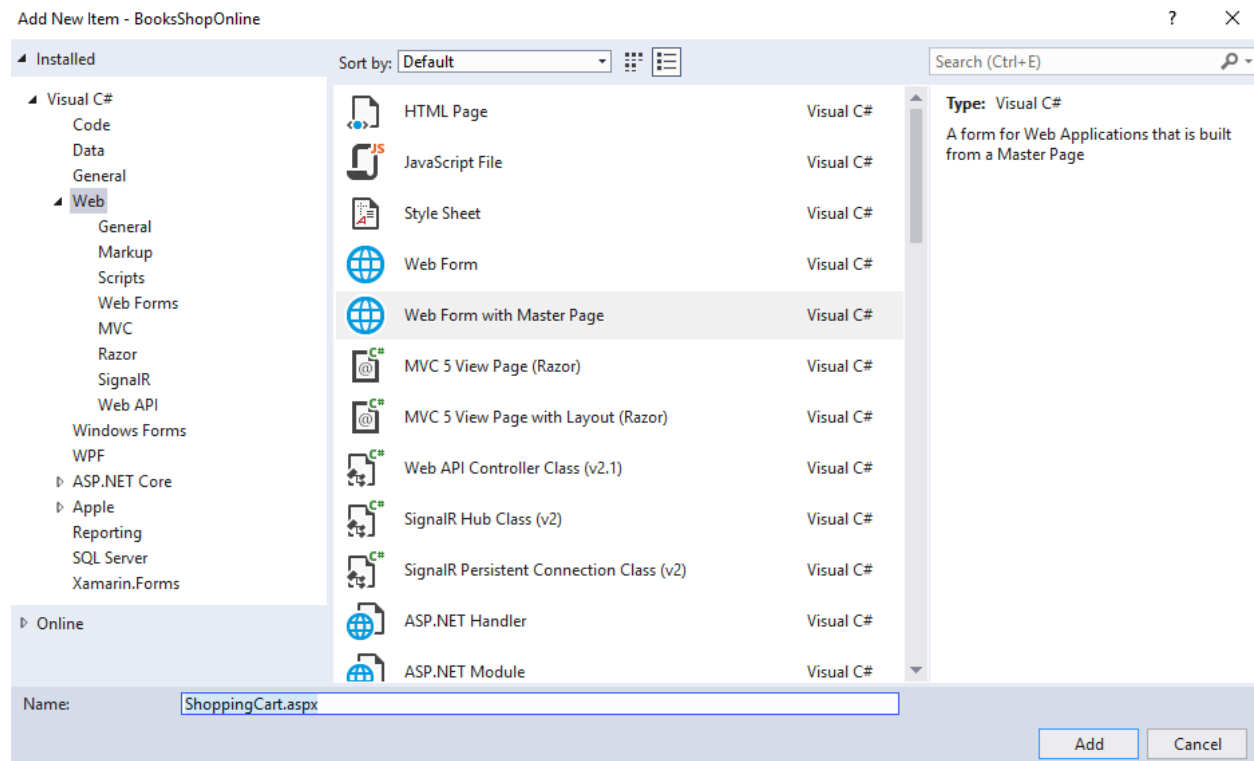
```

Khi trang AddToCart.aspx được tải, một ID của cuốn sách sẽ được nhận từ chuỗi truy vấn (query string). Kế tiếp, một đối tượng của lớp giỏ hàng (ShoppingCartActions) sẽ được tạo và gọi phương thức AddToCart. Cuối cùng, trang sẽ điều hướng đến trang ShoppingCart.aspx.

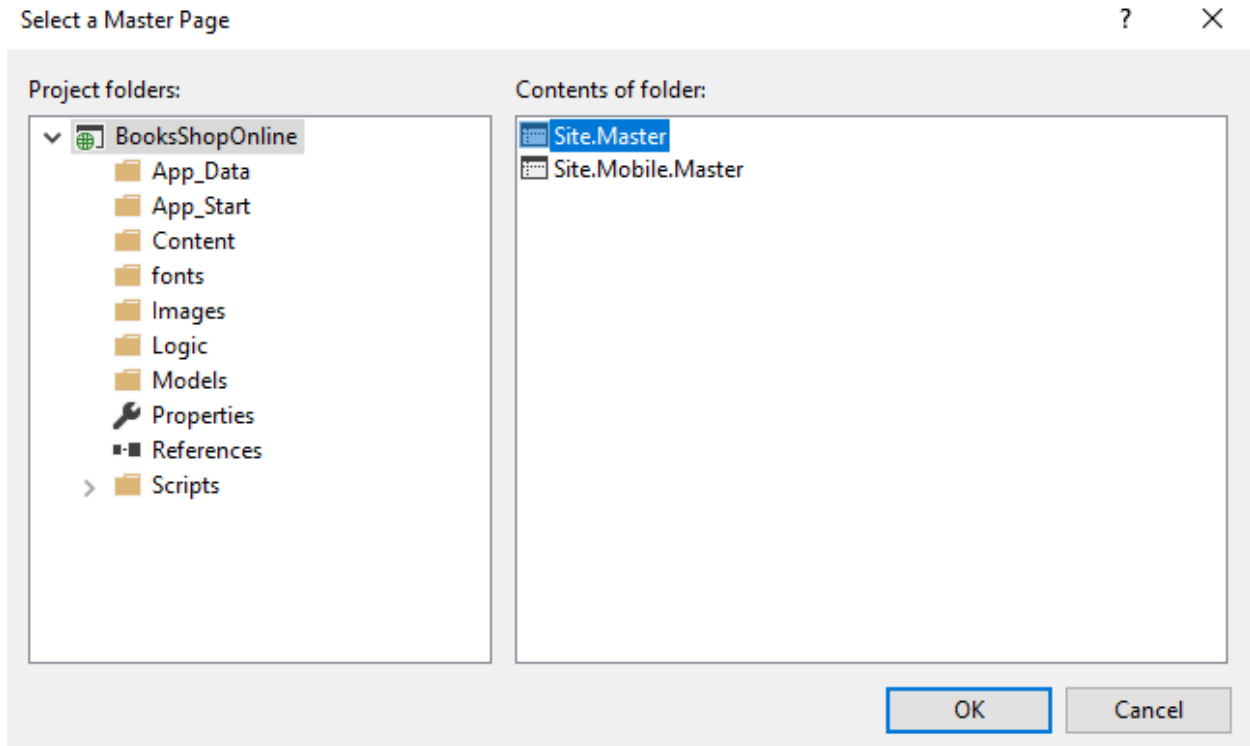
Như đã đề cập ở trên, một ID được sử dụng để kết nối một người dùng và các cuốn sách của anh ta. ID này sẽ được thêm đến một hàng trong bảng CartItem mỗi khi người dùng thêm một cuốn sách đến giỏ hàng. Bây giờ, chúng ta sẽ xây dựng giao diện trang ShoppingCart.aspx để hiển thị các cuốn sách mà người dùng đã thêm đến giỏ hàng. Với giao diện này, người dùng có thể thêm, xóa hay cập nhật các cuốn sách từ giỏ hàng.

## Xây dựng giao diện giỏ hàng

1. Trong cửa sổ Solution Explorer, nhấn chuột phải vào BooksShopOnline chọn Add > Add New Item và chọn Web Form with Master Page (hay Web Form using Master Page trong các phiên bản Visual Studio cũ hơn 2017) và gõ tên trang là ShoppingCart.aspx



Nhấn Add. Trong hộp thoại Select a Master Page chọn Site.Master và nhấn OK



2. Viết lại nội dung tập tin ShoppingCart.aspx như sau:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"
CodeBehind="ShoppingCart.aspx.cs" Inherits="BooksShopOnline.ShoppingCart" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
    <div id="ShoppingCartTitle" runat="server" class="ContentHead"><h1>Shopping
Cart</h1></div>
    <asp:GridView ID="CartList" runat="server" AutoGenerateColumns="False"
        ShowFooter="True" GridLines="Vertical" CellPadding="4"
        ItemType="BooksShopOnline.Models.CartItem"
        SelectMethod="GetShoppingCartItems"
        CssClass="table table-striped table-bordered" >
        <Columns><asp:BoundField DataField="BookID" HeaderText="ID"
            SortExpression="BookID" />
            <asp:BoundField DataField="Book.BookName" HeaderText="Name" />
            <asp:BoundField DataField="Book.UnitPrice" HeaderText="Price (each)"
                DataFormatString="{0:c}" />
            <asp:TemplateField HeaderText="Quantity">
                <ItemTemplate>
                    <asp:TextBox ID="PurchaseQuantity" Width="40"
                        runat="server" Text="<%#: Item.Quantity %>"></asp:TextBox>
                </ItemTemplate>
            </asp:TemplateField>
            <asp:TemplateField HeaderText="Item Total">
                <ItemTemplate>
                    <%#: String.Format("{0:c}",
                        ((Convert.ToDouble(Item.Quantity)) *
                        Convert.ToDouble(Item.Book.UnitPrice)))%>
                </ItemTemplate>
            </asp:TemplateField>
            <asp:TemplateField HeaderText="Remove Item">
```

```

                <ItemTemplate>
                    <asp:CheckBox id="Remove" runat="server"></asp:CheckBox>
                </ItemTemplate>
            </asp:TemplateField>
        </Columns>
    </asp:GridView>
</div>
    <p></p>
    <strong>
        <asp:Label ID="LabelTotalText" runat="server" Text="Order Total:"></asp:Label>
        <asp:Label ID="lblTotal" runat="server" EnableViewState="false"></asp:Label>
    </strong>
</div>
<br />
</asp:Content>

```

Chúng ta sử dụng GridView để nhận dữ liệu từ cơ sở dữ liệu CartItem thông qua thuộc tính ItemType và đồng thời chọn phương thức cho thuộc tính SelectMethod.

Sau khi xây dựng giao diện, kế tiếp sẽ là thêm một vài dòng mã đến tập tin ShoppingCart.aspx.cs để có thể nhận và thể hiện các sản phẩm (sách) từ giỏ hàng:

```

using BooksShopOnline.Models;
using BooksShopOnline.Logic;
...
public partial class ShoppingCart : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    public List<CartItem> GetShoppingCartItems()
    {
        ShoppingCartActions actions = new ShoppingCartActions();
        return actions.GetCartItems();
    }
}

```

Chúng ta đã thêm phương thức GetShoppingCartItems() đến lớp ShoppingCart và đây chính là phương thức chúng ta đã thiết lập đến thuộc tính SelectMethod của GridView. Cuối cùng, bổ sung liên kết Add To Cart để thêm một cuốn sách đến giỏ hàng bằng cách mở tập tin BookList.aspx và thêm một liên kết đến trang AddToCart.aspx theo giá trị ID của cuốn sách:

```

<tr>
    <td>
        <a href="BookDetails.aspx?bookID=<%=Item.BookID%>">
            <span>
                <%=Item.BookName%>
            </span>
        </a>
        <br />
        <span>
            <b>Price: </b><%=String.Format("{0:c}",Item.UnitPrice)%>
        </span>
        <br />
        <a href="AddToCart.aspx?bookID=<%=Item.BookID%>">
            <span>

```



```
<b>Add To Cart<b>  
</span>  
</a>  
</td>  
</tr>
```

Lưu và thực thi trang Default.aspx và chọn thể loại Fiction và lưu ý xuất hiện liên kết Add To Cart ở mỗi cuốn sách:

Books Shop Online

## Books Shop Online

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

### Books



Fire & Blood

Price: \$16.04

[Add To Cart](#)




© 2019 - Books Shop Online

Nhấn liên kết Add To Cart sẽ xuất hiện giỏ hàng với cuốn sách vừa chọn:

## Books Shop Online

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

### Shopping Cart

ID	Name	Price (each)	Quantity	Item Total	Remove Item
1	Fire & Blood	\$16.04	<input type="text" value="1"/>	\$16.04	



Order Total:

Tiếp tục chọn thể loại Self-Help và nhấn Add To Cart của cuốn sách xuất hiện, giỏ hàng lúc này:

## Books Shop Online

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

### Shopping Cart

ID	Name	Price (each)	Quantity	Item Total	Remove Item
5	The 7 Habits of Highly Effective People	\$16.04	<input type="text" value="1"/>	\$16.04	
1	Fire & Blood	\$16.04	<input type="text" value="1"/>	\$16.04	

Order Total:

Đóng trình duyệt. Như phần minh họa trên, giỏ hàng chúng ta vẫn chưa có tính năng tính tổng tiền các cuốn sách trong giỏ hàng và chúng ta sẽ thêm tính năng này trong bước kế tiếp.

### Tính toán và hiển thị tổng tiền thanh toán đến giỏ hàng

1. Mở tập tin ShoppingCartActions.cs trong thư mục Logic và thêm phương thức GetTotal() đến lớp ShoppingCartActions:

```
public decimal GetTotal()
{
    ShoppingCartId = GetCartId();
    // Tổng tiền mỗi cuốn sách (Item Total) = đơn giá (UnitPrice) nhân
    // số lượng (Quantity). Tổng của các tổng tiền chính là
    // số tiền mà người dùng phải trả (Order Total)
    decimal? total = decimal.Zero;
    total = (decimal?)(from cartItems in _db.ShoppingCartItems
                        where cartItems.CartId == ShoppingCartId
                        select (int?)cartItems.Quantity *
                               cartItems.Book.UnitPrice).Sum();
    return total ?? decimal.Zero;
}
```

2. Cho phép hiển thị tổng tiền các cuốn sách (Item Total và Order Total) bằng cách gọi phương thức GetTotal() trong phương thức xử lý sự kiện Load trong tập ShoppingCart.aspx.cs:

```
protected void Page_Load(object sender, EventArgs e)
{
    using (ShoppingCartActions usersShoppingCart = new ShoppingCartActions())
    {
        decimal cartTotal = 0;
        cartTotal = usersShoppingCart.GetTotal();
        if (cartTotal > 0)
        {
            // Display Total.
            lblTotal.Text = String.Format("{0:c}", cartTotal);
        }
        else
        {
            LabelTotalText.Text = "";
            lblTotal.Text = "";
            ShoppingCartTitle.InnerText = "Shopping Cart is Empty";
        }
    }
}
```

Lưu và thực thi ứng dụng như phần thực thi kiểm tra chức năng Add To Cart, kết quả xuất hiện giá trị tại Order Total:

## Books Shop Online

Fiction | Biographies and Memoirs | Biological Sciences | Self-Help

### Shopping Cart

ID	Name	Price (each)	Quantity	Item Total	Remove Item
5	The 7 Habits of Highly Effective People	\$16.04	<input type="text" value="1"/>	\$16.04	
1	Fire & Blood	\$16.04	<input type="text" value="1"/>	\$16.04	

Order Total: \$32.08

© 2019 - Books Shop Online

Đóng trình duyệt. Giỏ hàng của chúng ta phải cho phép người dùng chỉnh sửa (thay đổi, xóa) các thông tin và sẽ cập nhật lại các thông tin mới này thông qua nút Update.

Thêm nút Update đến giỏ hàng bằng cách thêm đoạn mã sau đến tập tin ShoppingCart.aspx:

```
<div>
    <p></p>
    <strong>
        <asp:Label ID="LabelTotalText" runat="server" Text="Order Total:"></asp:Label>
        <asp:Label ID="lblTotal" runat="server" EnableViewState="false"></asp:Label>
    </strong>
</div>
<br />
<table>
    <tr>
        <td>
            <asp:Button ID="UpdateBtn" runat="server" Text="Update"
                OnClick="UpdateBtn_Click" />
        </td>
    </tr>
</table>
</asp:Content>
```

Khi người dùng nhấn nút Update, phương thức UpdateBtn\_Click sẽ được gọi. Trong tập tin ShoppingCart.aspx.cs, thêm các dòng mã và phương thức sau (bôi vàng):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```

using BooksShopOnline.Models;
using BooksShopOnline.Logic;
using System.Collections.Specialized;
using System.Collections;
using System.Web.ModelBinding;
namespace BooksShopOnline
{
    public partial class ShoppingCart : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            using (ShoppingCartActions usersShoppingCart = new ShoppingCartActions())
            {
                decimal cartTotal = 0;
                cartTotal = usersShoppingCart.GetTotal();
                if (cartTotal > 0)
                {
                    // Display Total.
                    lblTotal.Text = String.Format("{0:c}", cartTotal);
                }
                else
                {
                    LabelTotalText.Text = "";
                    lblTotal.Text = "";
                    ShoppingCartTitle.InnerText = "Shopping Cart is Empty";
                    UpdateBtn.Visible = false;
                }
            }
        }
        public List<CartItem> GetShoppingCartItems()
        {
            ShoppingCartActions actions = new ShoppingCartActions();
            return actions.GetCartItems();
        }
        public List<CartItem> UpdateCartItems()
        {
            using (ShoppingCartActions usersShoppingCart = new ShoppingCartActions())
            {
                String cartId = usersShoppingCart.GetCartId();
                ShoppingCartActions.ShoppingCartUpdates[] cartUpdates = new
                ShoppingCartActions.ShoppingCartUpdates[CartList.Rows.Count];
                for (int i = 0; i < CartList.Rows.Count; i++)
                {
                    IDictionary rowValues = new OrderedDictionary();
                    rowValues = GetValues(CartList.Rows[i]);
                    cartUpdates[i].BookId = Convert.ToInt32(rowValues["BookID"]);
                    CheckBox cbRemove = new CheckBox();
                    cbRemove = (CheckBox)CartList.Rows[i].FindControl("Remove");
                    cartUpdates[i].RemoveItem = cbRemove.Checked;
                    TextBox quantityTextBox = new TextBox();
                    quantityTextBox =
                    (TextBox)CartList.Rows[i].FindControl("PurchaseQuantity");
                    cartUpdates[i].PurchaseQuantity =
                    Convert.ToInt16(quantityTextBox.Text.ToString());
                }
                usersShoppingCart.UpdateShoppingCartDatabase(cartId, cartUpdates);
                CartList.DataBind();
                lblTotal.Text = String.Format("{0:c}", usersShoppingCart.GetTotal());
            }
        }
    }
}

```

```

        return usersShoppingCart.GetCartItems();
    }
}
public static IOrderedDictionary GetValues(GridViewRow row)
{
    IOrderedDictionary values = new OrderedDictionary();
    foreach (DataControlFieldCell cell in row.Cells)
    {
        if (cell.Visible)
        {
            // Extract values from the cell.
            cell.ContainingField.ExtractValuesFromCell(values, cell,
                row.RowState, true);
        }
    }
    return values;
}
protected void UpdateBtn_Click(object sender, EventArgs e)
{
    UpdateCartItems();
}
}
}

```

Phương thức UpdateCartItems() sẽ được gọi khi người dùng nhấn nút Update. Phương thức này nhận các giá trị được cập nhật của mỗi cuốn sách từ giỏ hàng sau đó sẽ gọi phương thức UpdateShoppingCartDatabase() (sẽ được định nghĩa sau đây) để thêm hay xóa cuốn sách từ giỏ hàng. Các thông tin cập nhật sẽ được phản ánh qua trang giỏ hàng.

Bước kế tiếp sẽ là thêm các hàm chức năng cập nhật hay xóa thông tin từ giỏ hàng đến lớp ShoppingCartActions. Mở tập tin ShoppingCartActions.cs trong thư mục Logic và thêm các phương thức sau:

```

public ShoppingCartActions GetCart(HttpContext context)
{
    using (var cart = new ShoppingCartActions())
    {
        cart.ShoppingCartId = cart.GetCartId();
        return cart;
    }
}
public void UpdateShoppingCartDatabase(String cartId, ShoppingCartUpdates[]
CartItemUpdates)
{
    using (var db = new BooksShopOnline.Models.BookContext())
    {
        try
        {
            int CartItemCount = CartItemUpdates.Count();
            List<CartItem> myCart = GetCartItems();
            foreach (var cartItem in myCart)
            {
                // Lặp qua các hàng trong giỏ hàng
                for (int i = 0; i < CartItemCount; i++)
                {
                    if (cartItem.Book.BookID == CartItemUpdates[i].BookId)

```

```

        {
            if (CartItemUpdates[i].PurchaseQuantity < 1 ||
                CartItemUpdates[i].RemoveItem == true)
            {
                RemoveItem(cartId, cartItem.BookId);
            }
            else
            {
                UpdateItem(cartId, cartItem.BookId,
                    CartItemUpdates[i].PurchaseQuantity);
            }
        }
    }
}
catch (Exception exp)
{
    throw new Exception("ERROR: Unable to Update Cart Database - " +
        exp.Message.ToString(), exp);
}
}

public void RemoveItem(string removeCartID, int removeBookID)
{
    using (var _db = new BooksShopOnline.Models.BookContext())
    {
        try
        {
            var myItem = (from c in _db.ShoppingCartItems
                          where c.CartId == removeCartID && c.Book.BookID ==
removeBookID
                          select c).FirstOrDefault();
            if (myItem != null)
            {
                // Xóa
                _db.ShoppingCartItems.Remove(myItem);
                _db.SaveChanges();
            }
        }
        catch (Exception exp)
        {
            throw new Exception("ERROR: Unable to Remove Cart Item - " +
                exp.Message.ToString(), exp);
        }
    }
}

public void UpdateItem(string updateCartID, int updateBookID, int
quantity)
{
    using (var _db = new BooksShopOnline.Models.BookContext())
    {
        try
        {
            var myItem = (from c in _db.ShoppingCartItems
                          where c.CartId == updateCartID && c.Book.BookID ==
updateBookID
                          select c).FirstOrDefault();
            if (myItem != null)

```

```

        {
            myItem.Quantity = quantity;
            _db.SaveChanges();
        }
    }
    catch (Exception exp)
    {
        throw new Exception("ERROR: Unable to Update Cart Item - " +
            exp.Message.ToString(), exp);
    }
}

}

public void EmptyCart()
{
    ShoppingCartId = GetCartId();
    var cartItems = _db.ShoppingCartItems.Where(
        c => c.CartId == ShoppingCartId);
    foreach (var cartItem in cartItems)
    {
        _db.ShoppingCartItems.Remove(cartItem);
    }
    // cập nhật
    _db.SaveChanges();
}

public int GetCount()
{
    ShoppingCartId = GetCartId();
    // Đếm và tính tổng
    int? count = (from cartItems in _db.ShoppingCartItems
        where cartItems.CartId == ShoppingCartId
        select (int?)cartItems.Quantity).Sum();

    // Trả về 0 nếu rỗng
    return count ?? 0;
}

public struct ShoppingCartUpdates
{
    public int BookId;
    public int PurchaseQuantity;
    public bool RemoveItem;
}

```

Phương thức UpdateShoppingCartDatabase() lặp qua các hàng trong giỏ hàng. Nếu một cuốn sách nào đó được đánh dấu là xóa (Remove Item) hay mục Quantity < 1 thì phương thức RemoveItem() được gọi. Ngược lại, cuốn sách sẽ được đánh dấu cập nhật lại khi phương thức UpdateItem() được gọi. Sau khi giỏ hàng được cập nhật, cơ sở dữ liệu cũng sẽ được cập nhật sau đó.

Một cấu trúc (structure) ShoppingCartUpdates cũng được thêm vào để nắm giữ các thông tin về các cuốn sách trong giỏ hàng. Cấu trúc này sẽ được dùng nếu các cuốn sách trong giỏ hàng được cập nhật hay xóa.

Phương thức EmptyCart() sẽ được dùng để xóa giỏ hàng sau khi người dùng hoàn tất việc mua hàng của mình và phương thức này sẽ được dùng trong phần thanh toán sau này. Trước khi kết thúc phần giỏ hàng, chúng ta sẽ sử dụng phương thức GetCount() để xác định có bao nhiêu cuốn sách trong giỏ hàng.

Mở tập tin Site.Master và thêm một liên kết đến trang ShoppingCart để trình thực đơn website:



```
<ul class="nav navbar-nav">
  <li><a runat="server" href="~/>Home</a></li>
  <li><a runat="server" href="~/About">About</a></li>
  <li><a runat="server" href="~/Contact">Contact</a></li>
  <li><a runat="server" href="~/BookList">Books</a></li>
  <li><a runat="server" href="~/ShoppingCart" ID="cartCount">&nbsp;</a></li>
</ul>
```

Trong tập tin Site.Master.cs, thêm phương thức xử lý sự kiện PreRender của trang như sau:

```
protected void Page_Load(object sender, EventArgs e)
{
}

protected void Page_PreRender(object sender, EventArgs e)
{
    using (ShoppingCartActions usersShoppingCart = new
        ShoppingCartActions())
    {
        string cartStr = string.Format("Cart ({0})",
            usersShoppingCart.GetCount());
        cartCount.InnerText = cartStr;
    }
}
```

Sự kiện PreRender sẽ phát sinh trước khi nội dung trang được chuyển sang nội dung HTML để trình duyệt. Lúc này, toàn bộ các cuốn sách trong giỏ hàng sẽ được trả về.

Lưu và thực thi ứng dụng. Chọn cuốn sách từ mục Fiction và nhấn Add To Cart. Sau đó, chọn cuốn sách từ mục Self-Help và nhấn Add To Cart:

— Books Shop Online

## Books Shop Online

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

### Shopping Cart

ID	Name	Price (each)	Quantity	Item Total	Remove Item
1	Fire & Blood	\$16.04	<input type="text" value="1"/>	\$16.04	<input type="button" value="✖"/>
5	The 7 Habits of Highly Effective People	\$16.04	<input type="text" value="1"/>	\$16.04	<input type="button" value="✖"/>

Order Total: \$32.08

Thay đổi mục Quantity cuốn sách đầu tiên đến 3 và đánh dấu mục Remove Item cho cuốn sách thứ hai.  
Nhấn nút Update:

 Books Shop Online

## Books Shop Online

[Fiction](#) | [Biographies and Memoirs](#) | [Biological Sciences](#) | [Self-Help](#)

### Shopping Cart

ID	Name	Price (each)	Quantity	Item Total	Remove Item
1	Fire & Blood	\$16.04	<input type="text" value="3"/>	\$48.12	<input type="checkbox"/>

Order Total: \$48.12

© 2019 - Books Shop Online