

# MỤC LỤC

<b>MỤC LỤC .....</b>	<b>I</b>
<b>HƯỚNG DẪN .....</b>	<b>III</b>
<b>BÀI 1: XÂY DỰNG LAYOUT CHO TRANG WEB BẰNG HTML &amp; CSS .....</b>	<b>1</b>
<b>    1.1 Yêu cầu chung của bài thực hành.....</b>	<b>1</b>
1.1.1 <i>Bố cục trang web .....</i>	1
1.1.2 <i>Menu điều hướng .....</i>	1
1.1.3 <i>Danh sách sản phẩm .....</i>	1
1.1.4 <i>Thanh bên (Sidebar).....</i>	2
1.1.5 <i>Chân trang (Footer).....</i>	2
1.1.6 <i>CSS Grid .....</i>	2
1.1.7 <i>Độ phản hồi.....</i>	2
<b>    1.2 Nội dung bổ sung.....</b>	<b>2</b>
<b>    1.3 Chú ý .....</b>	<b>2</b>
<b>    1.4 Hướng dẫn .....</b>	<b>4</b>
1.4.1 <i>Tập tin Index.html.....</i>	4
1.4.2 <i>Tập tin Style.css .....</i>	7
<b>    1.5 Yêu cầu bổ sung.....</b>	<b>11</b>
1.5.1 <i>Xem chi tiết sản phẩm .....</i>	11
1.5.2 <i>Sử dụng Bootstrap .....</i>	11
1.5.3 <i>Triển khai yêu cầu bổ sung.....</i>	12
<b>BÀI 2: XÂY DỰNG ỨNG DỤNG WEBSITE CƠ BẢN VỚI ASP.NET CORE MVC .....</b>	<b>17</b>
<b>    2.1 Khởi tạo ứng dụng ASP.NET Core .....</b>	<b>18</b>
<b>    2.2 Cấu trúc dự án ASP.NET Core MVC .....</b>	<b>20</b>
2.2.1 <i>_Layout.....</i>	21
2.2.2 <i>Razor View Engine.....</i>	22
2.2.3 <i>RenderBody.....</i>	22
2.2.4 <i>RenderSectionAsync .....</i>	23
<b>    2.3 Bài tập .....</b>	<b>23</b>
2.3.1 <i>Yêu Cầu Bài Thực Hành: Ứng Dụng Thêm/Đọc/Xóa/Sửa trên ASP.NET Core MVC... </i>	23
2.3.2 <i>Code Mẫu Chi Tiết .....</i>	24
2.3.3 <i>Kết quả .....</i>	35
2.3.4 <i>Bổ sung tính năng upload file cho ứng dụng .....</i>	36
2.3.5 <i>Yêu cầu bổ sung.....</i>	40
<b>BÀI 3: XÂY DỰNG ỨNG DỤNG WEBSITE BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 1)</b>	<b>41</b>
<b>    3.1 Bài tập .....</b>	<b>41</b>
3.1.1 <i>Yêu cầu.....</i>	41
3.1.2 <i>Hướng dẫn thực hiện .....</i>	41
3.1.3 <i>Kết quả .....</i>	55

3.1.4 Yêu cầu bổ sung .....	56
<b>BÀI 4: XÂY DỰNG ỨNG DỤNG WEB BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 2) ....</b>	<b>60</b>
<b>4.1 Bài thực hành .....</b>	<b>60</b>
<b>4.2 Hướng dẫn thực hiện.....</b>	<b>60</b>
4.2.1 Cấu Hình ASP.NET Core Identity .....	60
4.2.2 Scaffolding ASP.NET Core Identity.....	63
4.2.3 Thêm Liên Kết Đăng Ký và Đăng Nhập.....	65
4.2.4 Phần vùng Area .....	74
<b>4.3 Yêu cầu bổ sung .....</b>	<b>80</b>
<b>BÀI 5: XÂY DỰNG ỨNG DỤNG WEBSITE BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 3)</b>	<b>81</b>
<b>5.1 Mục tiêu của bài thực hành .....</b>	<b>81</b>
5.1.1 Yêu cầu.....	81
5.1.2 Xây dựng giao diện hiển thị Danh sách sản phẩm người dùng .....	81
<b>5.2 Hướng dẫn thực hiện.....</b>	<b>84</b>
5.2.1 Code mẫu thực hiện chức năng giờ hàng .....	84
5.2.2 Hướng dẫn thực hiện chức năng đặt hàng.....	88
5.2.3 Yêu cầu bổ sung .....	91
<b>BÀI 6: RESTFUL API .....</b>	<b>92</b>
<b>6.1 Mục tiêu của bài thực hành .....</b>	<b>92</b>
6.1.1 Giới thiệu .....	92
6.1.2 Yêu cầu.....	93
<b>6.2 Hướng dẫn thực hiện.....</b>	<b>93</b>
<b>6.3 Yêu cầu bổ sung .....</b>	<b>106</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>107</b>

# HƯỚNG DẪN

## MÔ TẢ MÔN HỌC

Học phần trang bị cho sinh viên những kiến thức và kỹ năng cơ bản về ASP.NET Core MVC để có thể xây dựng các ứng dụng web từ đơn giản đến nâng cao. Sinh viên sẽ nắm được các khái niệm cơ bản về ASP.NET Core MVC như: cấu trúc chung của một ứng dụng MVC, vai trò và mối quan hệ của các thành phần chính như Controller, Action, View, Model. Sinh viên sẽ hiểu được sự khác biệt giữa ASP.NET Core MVC với các framework khác như ASP.NET Web Form, ASP.NET MVC cũ. Điều này giúp sinh viên nắm được lợi thế của ASP.NET Core MVC để có thể lựa chọn và áp dụng phù hợp. Cụ thể, Sinh viên sẽ biết cách tạo project ASP.NET Core MVC, tạo controller và action, tạo view và layout, tạo model và sử dụng Entity Framework Core, xử lý form và validate dữ liệu, cấu hình routing và xây dựng hệ thống authentication. Những kiến thức và kỹ năng này sẽ giúp sinh viên có thể tự tin xây dựng các ứng dụng web bằng ASP.NET Core MVC.

## NỘI DUNG MÔN HỌC

- Bài 1. Xây dựng layout cho trang web bán hàng bằng HTML & CSS
- Bài 2. Xây dựng ứng dụng cơ bản với ASP.NET Core MVC
- Bài 3: Xây dựng ứng dụng website bán hàng với ASP.NET Core MVC (Phần 1)
- Bài 4: Xây dựng ứng dụng website bán hàng với APS.NET Core MVC (Phần 2)
- Bài 5: Xây dựng ứng dụng website bán hàng với APS.NET Core MVC (Phần 3)
- Bài 6: Xây dựng ứng dụng website bán hàng với APS.NET Core MVC (Phần 4)

## KIẾN THỨC TIỀN ĐỀ

Kiến thức cơ bản về lập trình hướng đối tượng bằng C# hoặc VB.NET. Điều này bao gồm các khái niệm như lớp, đối tượng, kế thừa, đa hình,...

Kiến thức cơ bản về cơ sở dữ liệu, đặc biệt là cách thao tác với cơ sở dữ liệu SQL Server.

## YÊU CẦU MÔN HỌC

Người học phải dự học đầy đủ các buổi lên lớp và làm bài tập đầy đủ ở nhà.

## CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần ôn tập các bài đã học, trả lời các câu hỏi và làm đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

Đối với mỗi bài học, người học đọc trước mục tiêu và tóm tắt bài học, sau đó đọc nội dung bài học. Kết thúc mỗi ý của bài học, người học cần thực hành đầy đủ các hướng dẫn trong phần bài tập.

## PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Môn học được đánh giá gồm:

- Điểm chuyên cần (30%): Hình thức và cách đánh giá do giảng viên dạy thực hành quyết định được phê duyệt của bộ môn.
- Điểm bài tập (70%): Hình thức làm bài tập trong các buổi học thực hành và giảng viên đánh giá chấm điểm. Danh sách bài tập thực hành được bộ môn kiểm duyệt và cung cấp vào đầu khóa học.



# BÀI 1: XÂY DỰNG LAYOUT CHO TRANG WEB BẰNG HTML & CSS

Sau khi học xong bài này, sinh viên có thể:

- Hiểu và biết cách sử dụng HTML & CSS trong việc thiết kế các trang web tĩnh.
- Có khả năng thiết kế được các trang web với tính linh hoạt cao và đẹp mắt.
- Biết cách thiết kế trang web đảm bảo được tính nhất quán và trải nghiệm người dùng tốt trên cả máy tính và thiết bị di động.

## **1.1 Yêu cầu chung của bài thực hành**

### **1.1.1 Bố cục trang web**

- Chia trang thành hai cột trên máy tính. Cột trái sẽ hiển thị sản phẩm và cột phải sẽ hiển thị thông tin mạng xã hội.
- Trên thiết bị di động, hiển thị các nội dung thành 1 cột.

### **1.1.2 Menu điều hướng**

- Thêm một menu điều hướng ở phía trên trang web với ít nhất ba liên kết: Trang chủ, Sản phẩm và Liên hệ.

### **1.1.3 Danh sách sản phẩm**

- Tạo một danh sách sản phẩm trong cột trái.
- Hiển thị mỗi sản phẩm trong một thẻ sản phẩm với hình ảnh, tên sản phẩm, mô tả và giá.

### 1.1.4 Thanh bên (Sidebar)

- Tạo một phần sidebar trong cột phải
- Thêm ít nhất bốn liên kết mạng xã hội vào sidebar (ví dụ: Facebook, Twitter, Instagram, LinkedIn).
- Sử dụng font-awesome để hiển thị các icon cho mạng xã hội tương ứng.

### 1.1.5 Chân trang (Footer)

- Tạo một phần footer ở dưới cùng của trang web.
- Trên máy tính, hiển thị ba cột trong footer. Trên thiết bị di động, hiển thị thành 1 cột, các cột xếp chồng lên nhau.

### 1.1.6 CSS Grid

- Sử dụng CSS Grid để xây dựng bố cục của trang web.

### 1.1.7 Độ phản hồi

- Đảm bảo trang web đáp ứng tốt trên các kích thước màn hình khác nhau.

## 1.2 Nội dung bổ sung

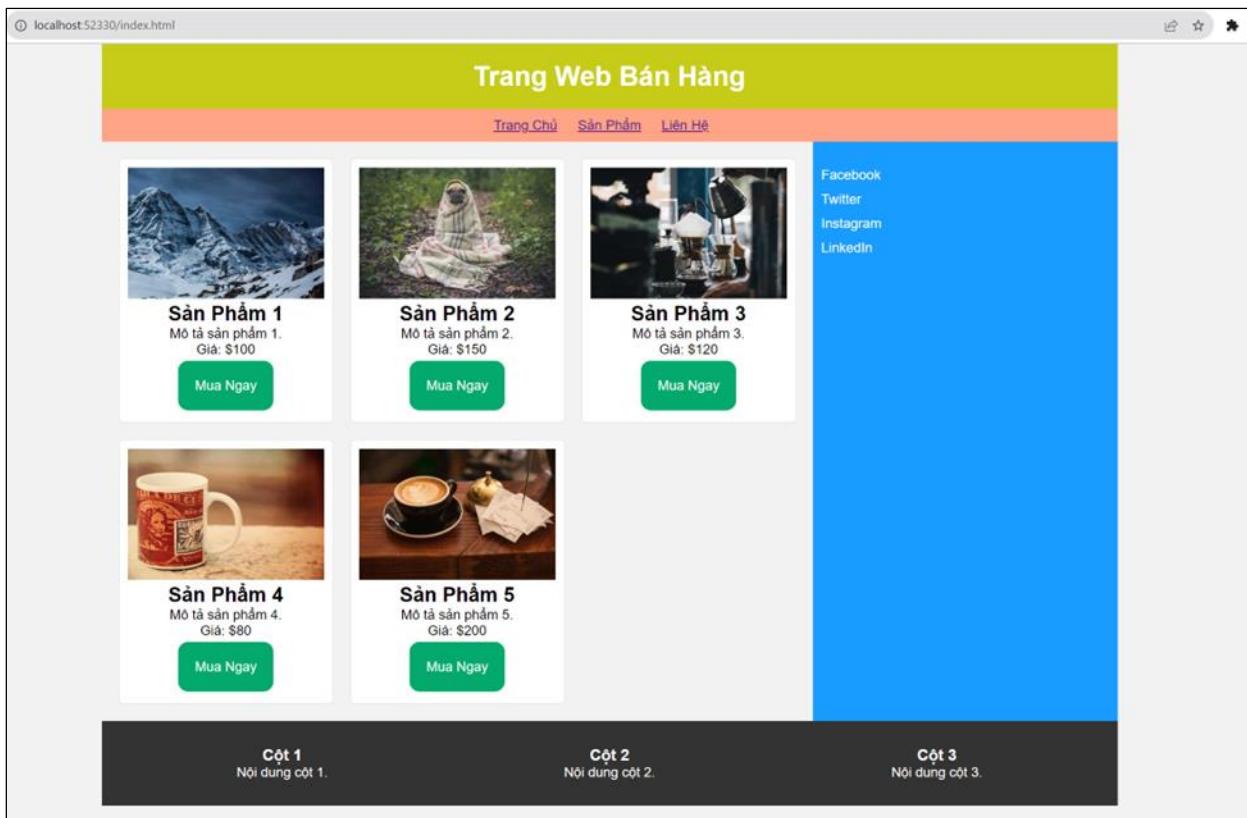
---

- Tùy chỉnh màu sắc, kiểu chữ và biểu đồ màu sắc để làm cho trang web thẩm mỹ hơn.
- Thêm hiệu ứng hover cho sản phẩm và các liên kết.
- Sử dụng hình ảnh thay thế cho các sản phẩm.

## 1.3 Chú ý

---

- Sử dụng HTML và CSS để thực hiện bài tập này.
- Tuân theo các nguyên tắc CSS tốt như sử dụng các **class** và **id** có ý nghĩa.



**Hình 1.1 Thực hành HTML và CSS**

## 1.4 Hướng dẫn

---

### 1.4.1 Tập tin Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Trang Web Bán Hàng</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <header>
        <h1>Trang Web Bán Hàng</h1>
    </header>

    <!-- Menu điều hướng -->
    <nav>
        <ul>
            <li><a href="#">Trang Chủ</a></li>
            <li><a href="#">Sản Phẩm</a></li>
            <li><a href="#">Liên Hệ</a></li>
        </ul>
    </nav>

```

```
<div class="product-list">
    <!-- Hiển thị duy nhất một sản phẩm trên mỗi hàng -->
    <div class="product-card">
        
        <h2>Sản Phẩm 1</h2>
        <p>Mô tả sản phẩm 1.</p>
        <p>Giá: $100</p>
        <button class="button button4">Mua Ngay</button>
    </div>

    <!-- Bổ sung thêm sản phẩm -->
    <div class="product-card">
        
        <h2>Sản Phẩm 2</h2>
        <p>Mô tả sản phẩm 2.</p>
        <p>Giá: $150</p>
        <button class="button button4">Mua Ngay</button>
    </div>

    <div class="product-card">
        
        <h2>Sản Phẩm 3</h2>
        <p>Mô tả sản phẩm 3.</p>
        <p>Giá: $120</p>
        <button class="button button4">Mua Ngay</button>
    </div>

    <div class="product-card">
        
        <h2>Sản Phẩm 4</h2>
        <p>Mô tả sản phẩm 4.</p>
        <p>Giá: $80</p>
        <button class="button button4">Mua Ngay</button>
    </div>

    <div class="product-card">
        
        <h2>Sản Phẩm 5</h2>
        <p>Mô tả sản phẩm 5.</p>
        <p>Giá: $200</p>
        <button class="button button4">Mua Ngay</button>
    </div>
</div>
```

```
<!-- Sidebar -->
<div class="sidebar">
    <!-- Hiển thị thanh sidebar ở phía dưới sản phẩm trên di động -->
    <div class="social-icons">
        <a href="#">Facebook</a>
        <a href="#">Twitter</a>
        <a href="#">Instagram</a>
        <a href="#">LinkedIn</a>
    </div>
</div>

<!-- Phần footer -->
<div class="footer">
    <div class="footer-item">
        <h3>Cột 1</h3>
        <p>Nội dung cột 1.</p>
    </div>
    <div class="footer-item">
        <h3>Cột 2</h3>
        <p>Nội dung cột 2.</p>
    </div>
    <div class="footer-item">
        <h3>Cột 3</h3>
        <p>Nội dung cột 3.</p>
    </div>
</div>

</body>

</html>
```

### 1.4.2 Tập tin Style.css

```
index.html    style.css  x
style.css > .product-list

1  /* Reset CSS để loại bỏ các mặc định của trình duyệt */
2  * {
3      margin: 0;
4      padding: 0;
5      box-sizing: border-box;
6  }
7
8  /* Thiết lập font chung cho trang web */
9  body {
10     font-family: Arial, sans-serif;
11     background-color: #f2f2f2;
12     display: grid;
13     grid-template-columns: 70% 30%;
14     /* Chia trang thành hai cột */
15     max-width: 1200px;
16     /* Độ rộng tối đa của trang web */
17     margin: 0 auto;
18     /* Canh giữa trang web trên màn hình */
19 }
20
21 /* Phần tiêu đề trang web */
22 header {
23     background-color: #c7cc19;
24     color: #fff;
25     padding: 20px;
26     text-align: center;
27     grid-column: 1 / span 2;
28     /* Trải đều qua cả hai cột */
29 }
30
```

```
31 /* Phần menu điều hướng */
32 nav {
33     background-color: #ffa486;
34     color: white;
35     text-align: center;
36     padding: 10px;
37     grid-column: 1 / span 2;
38     /* Trải đều qua cả hai cột */
39 }
40
41 nav ul {
42     list-style: none;
43 }
44
45 nav ul li {
46     display: inline;
47     margin-right: 20px;
48 }
49
50 /* Phần danh sách sản phẩm */
51 .product-list {
52     display: grid;
53     grid-template-columns: repeat(3, 1fr);
54     /* Hiển thị 3 sản phẩm trên mỗi hàng */
55     gap: 20px;
56     /* Khoảng cách giữa các sản phẩm */
57     padding: 20px;
58     grid-column: 1;
59     /* Chỉ nằm ở cột đầu tiên */
60 }
61
62 .product-card {
63     background-color: white;
64     border: 1px solid #ddd;
65     border-radius: 5px;
66     padding: 10px;
67     width: 100%;
68     text-align: center; /* Căn giữa nội dung trong sản phẩm */
69 }
70
71 .product-card img {
72     max-width: 100%;
73     height: auto;
74 }
75
```

```
76  /* Phần sidebar */
77  .sidebar {
78      background-color: #199cff;
79      color: #fff;
80      padding: 10px;
81      grid-column: 2;
82      /* Chỉ nằm ở cột thứ hai */
83  }
84
85  .social-icons {
86      margin-top: 20px;
87  }
88
89  .social-icons a {
90      color: #fff;
91      text-decoration: none;
92      display: block;
93      margin-bottom: 10px;
94  }
95
96  /* Chính lại bố cục cho màn hình thiết bị di động */
97  @media (max-width: 576px) {
98      .product-list {
99          grid-template-columns: 1fr;
100         /* Hiển thị một sản phẩm trên mỗi hàng */
101     }
102
103     .product-card {
104         width: 100%;
105         margin-bottom: 20px;
106     }
107
108     .sidebar {
109         grid-column: 1 / span 2;
110         /* Trải đều qua cả hai cột */
111     }
112
113     /* Chính lại bố cục cho phần footer */
114     .footer {
115         grid-template-columns: 1fr;
116         /* 1 cột xếp chồng lên nhau trên màn hình di động */
117     }
118 }
```

```
119
120 /* Phần footer */
121 .footer {
122     background-color: ■#333;
123     color: □#fff;
124     text-align: center;
125     padding: 20px;
126     grid-column: 1 / span 2;
127     /* Trải đều qua cả hai cột */
128     display: grid;
129 }
130
131 /* Cột trong phần footer */
132 .footer-item {
133     padding: 10px;
134 }
135
136 /* Chính layout cho máy tính */
137 @media (min-width: 768px) {
138     .footer {
139         grid-template-columns: repeat(3, 1fr);
140         /* 3 cột trên máy tính */
141     }
142 }
143
144 .button {
145     background-color: ■#04AA6D;
146     border: none;
147     color: □white;
148     padding: 20px;
149     text-align: center;
150     text-decoration: none;
151     display: inline-block;
152     font-size: 16px;
153     margin: 4px 2px;
154 }
```

```
156 .button1 {  
157     border-radius: 2px;  
158 }  
159  
164 .button3 {  
165     border-radius: 8px;  
166 }  
167  
168 .button4 {  
169     border-radius: 12px;  
170 }  
171  
172 .button5 {  
173     border-radius: 50%;  
174 }
```

## 1.5 Yêu cầu bổ sung

### 1.5.1 Xem chi tiết sản phẩm

- Khi người dùng nhấp vào một sản phẩm trong danh sách sản phẩm, họ sẽ được chuyển hướng đến một trang xem chi tiết sản phẩm.
- Trang xem sản phẩm cần hiển thị thông tin chi tiết về sản phẩm, bao gồm hình ảnh lớn, tên sản phẩm, mô tả chi tiết và giá cả.
- Thêm một nút "Quay lại" hoặc "Tiếp tục mua sắm" để người dùng có thể trở lại danh sách sản phẩm.

### 1.5.2 Sử dụng Bootstrap

- Thay vì sử dụng CSS tùy chỉnh, bạn có thể sử dụng Bootstrap để xây dựng giao diện tương tự.
- Thêm tập tin của Bootstrap CSS và JavaScript trong mã HTML (có thể sử dụng các phiên bản Bootstrap từ trang chính thức của Bootstrap).
- Sử dụng các lớp và thành phần của Bootstrap để thiết kế trang web

## 1.5.3 Triển khai yêu cầu bổ sung

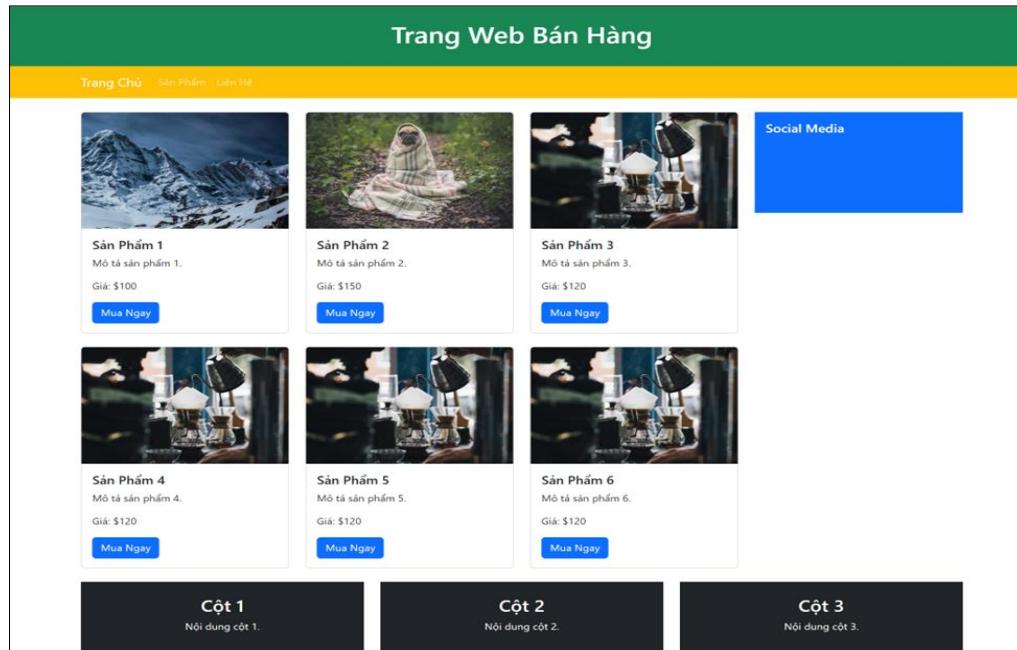
### 1.5.3.1 Xem chi tiết sản phẩm

- Tạo một trang mới trong dự án của bạn để hiển thị thông tin chi tiết về sản phẩm.
- Trong danh sách sản phẩm, sử dụng một thẻ `<a>` để bao quanh hình ảnh và tên sản phẩm. Đặt href của thẻ `<a>` thành URL của trang xem chi tiết sản phẩm.
- Sử dụng một nút hoặc liên kết để cho phép người dùng quay lại trang danh sách sản phẩm.

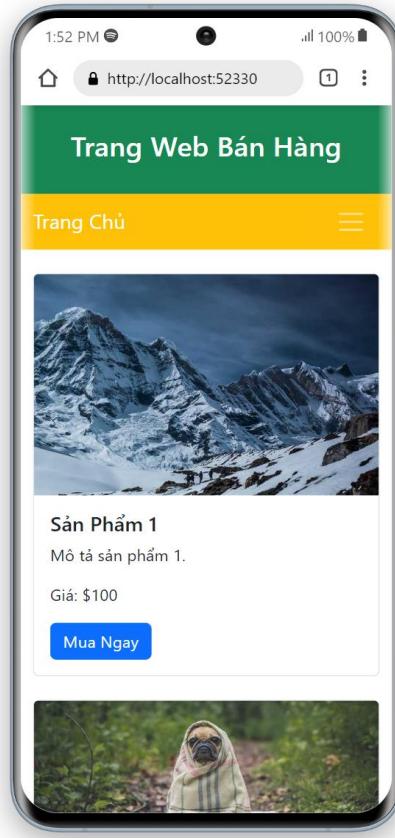
### 1.5.3.2 Sử dụng Bootstrap

- Tải Bootstrap CSS và JavaScript từ trang chính thức của Bootstrap (<https://getbootstrap.com/>).
- Bao gồm tệp Bootstrap CSS và JavaScript vào mã HTML của bạn, thường được đặt trong phần `<head>` của tệp HTML.
- Sử dụng lớp và các thành phần Bootstrap để thiết kế giao diện của trang web.

### 1.5.3.3 Kết quả thực hiện bằng Bootstrap



Hình 1.2. Kết quả thực hiện bằng Bootstrap



Hình 1.3 Kết quả hiển thị trên thiết bị di động

index.html > html

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Trang Web Bán Hàng</title>
8
9      <!-- Sử dụng Bootstrap CSS -->
10     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
11 </head>
12
13 <body>
14     <header class="bg-success text-white text-center py-4">
15         <h1>Trang Web Bán Hàng</h1>
16     </header>
17
18     <!-- Menu điều hướng sử dụng Bootstrap Navbar -->
19     <nav class="navbar navbar-expand-lg navbar-dark bg-warning">
20         <div class="container">
21             <a class="navbar-brand" href="#">Trang Chủ</a>
22             <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
23                 aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
24                 <span class="navbar-toggler-icon"></span>
25             </button>
26             <div class="collapse navbar-collapse" id="navbarNav">
27                 <ul class="navbar-nav">
28                     <li class="nav-item">
29                         <a class="nav-link" href="#">Sản Phẩm</a>
30                     </li>
31                     <li class="nav-item">
32                         <a class="nav-link" href="#">Liên Hệ</a>
33                     </li>
34                 </ul>
35             </div>
36         </div>
37     </nav>
38 
```

```
39   <div class="container mt-4">
40     <div class="row">
41       <!-- Danh sách sản phẩm -->
42       <div class="col-md-9">
43         <!-- Hiển thị sản phẩm bằng dạng dòng sản phẩm sử dụng Bootstrap Row và Col -->
44         <div class="row row-cols-1 row-cols-md-3 g-4">
45           <!-- Sản phẩm 1 -->
46           <div class="col">
47             <div class="card">
48               
49               <div class="card-body">
50                 <h5 class="card-title">Sản Phẩm 1</h5>
51                 <p class="card-text">Mô tả sản phẩm 1.</p>
52                 <p class="card-text">Giá: $100</p>
53                 <a href="#" class="btn btn-primary">Mua Ngay</a>
54             </div>
55           </div>
56         </div>
57
58         <!-- Sản phẩm 2 -->
59         <div class="col">...
60       </div>
61
62
63
64         <!-- Sản phẩm 3 -->
65         <div class="col">...
66       </div>
67
68
69         <!-- Sản phẩm 4 -->
70         <div class="col">...
71       </div>
72     </div>
73
74
75         <!-- Sidebar -->
76         <div class="col-md-3">
77           <div class="bg-primary text-white p-3">
78             <h5>Social Media</h5>
79             <ul class="list-unstyled">
80               <li><a href="#">Facebook</a></li>
81               <li><a href="#">Twitter</a></li>
82               <li><a href="#">Instagram</a></li>
83               <li><a href="#">LinkedIn</a></li>
84             </ul>
85           </div>
86         </div>
87       </div>
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136   </div>
```

```
137
138     <!-- Phần footer -->
139     <div class="container mt-4">
140         <div class="row">
141             <div class="col-md-4">
142                 <div class="bg-dark text-white text-center py-4">
143                     <h3>Cột 1</h3>
144                     <p>Nội dung cột 1.</p>
145                 </div>
146             </div>
147             <div class="col-md-4">
148                 <div class="bg-dark text-white text-center py-4">
149                     <h3>Cột 2</h3>
150                     <p>Nội dung cột 2.</p>
151                 </div>
152             </div>
153             <div class="col-md-4">
154                 <div class="bg-dark text-white text-center py-4">
155                     <h3>Cột 3</h3>
156                     <p>Nội dung cột 3.</p>
157                 </div>
158             </div>
159         </div>
160     </div>
161
162     <!-- Sử dụng Bootstrap JS và Popper.js (cần cài đặt trước khi sử dụng Bootstrap JS) -->
163     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
164 </body>
165
166 </html>
```

---Hết Lab TH 01---

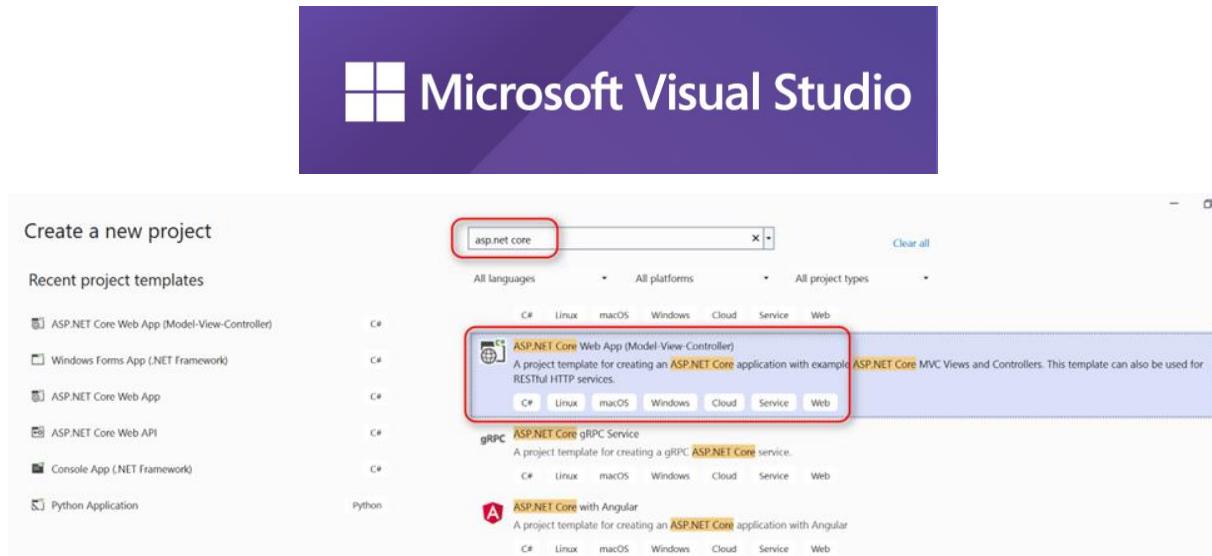
# BÀI 2: XÂY DỰNG ỨNG DỤNG WEBSITE CƠ BẢN VỚI ASP.NET CORE MVC

Sau khi học xong bài này, sinh viên có thể:

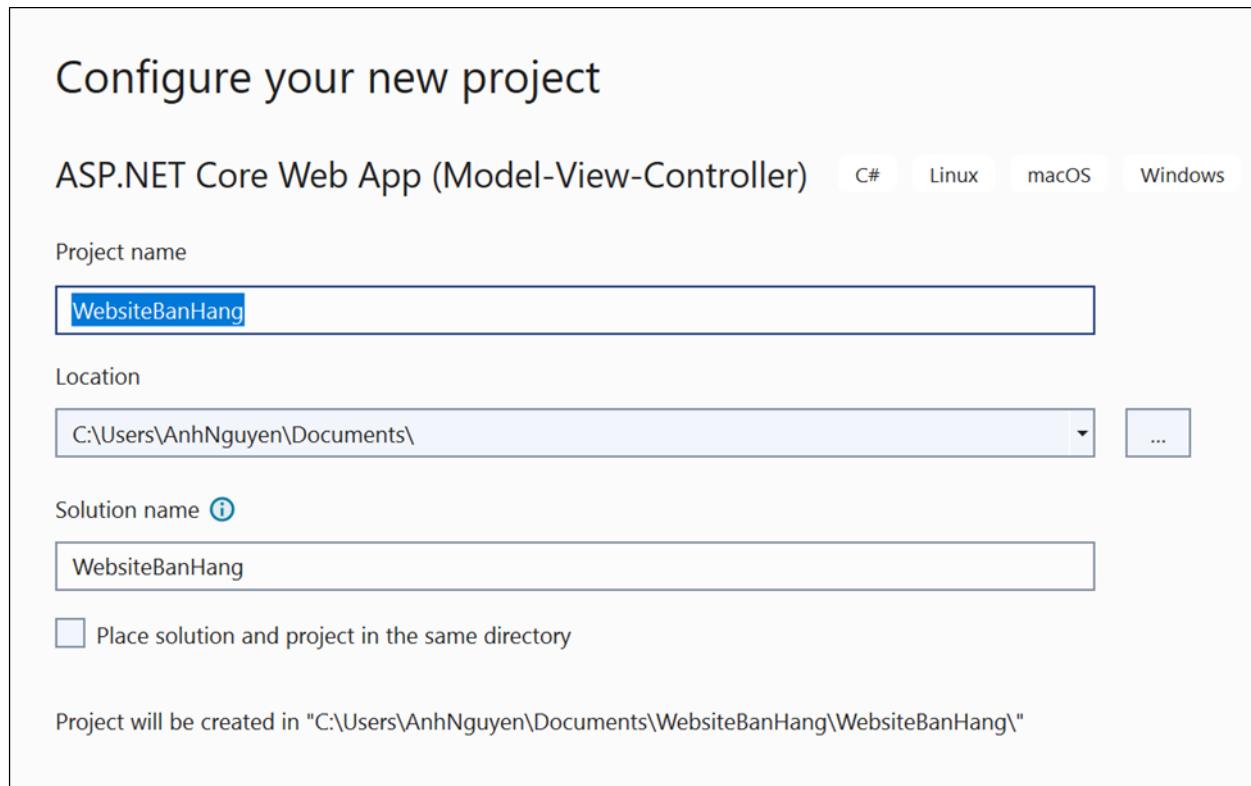
- Hiểu rõ cấu trúc nội bộ của ứng dụng ASP.NET Core MVC, bao gồm sự phân chia và tương tác giữa các lớp và modules.
- Hiểu cơ chế truyền tải và nhận dữ liệu trong mô hình MVC, bao gồm cách thức hoạt động của binding và routing
- Nắm vững khái niệm và quản lý của 'view', cũng như sử dụng Razor syntax
- Hiểu sâu về cấu trúc và chức năng của 'model' trong MVC, cũng như cách thức tương tác và xử lý dữ liệu.
- Hiểu rõ vai trò của 'controller' trong MVC, bao gồm cách quản lý luồng dữ liệu, xử lý yêu cầu và phản hồi.
- Hiểu rõ cách truyền 'model' giữa các thành phần trong MVC.
- Thực hành kỹ thuật xử lý và lưu trữ hình ảnh.

## 2.1 Khởi tạo ứng dụng ASP.NET Core

Khởi động phần mềm **Visual Studio 2022**

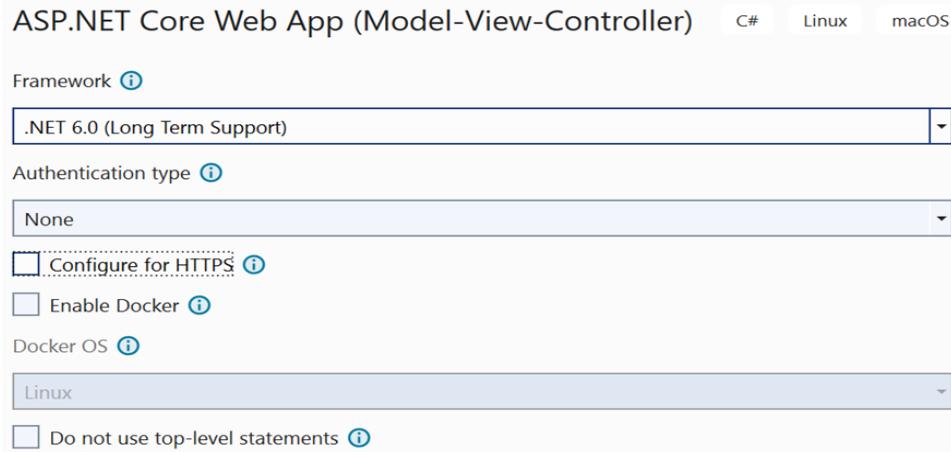


**Hình 2.1 Tạo project ASP.NET CORE mới**

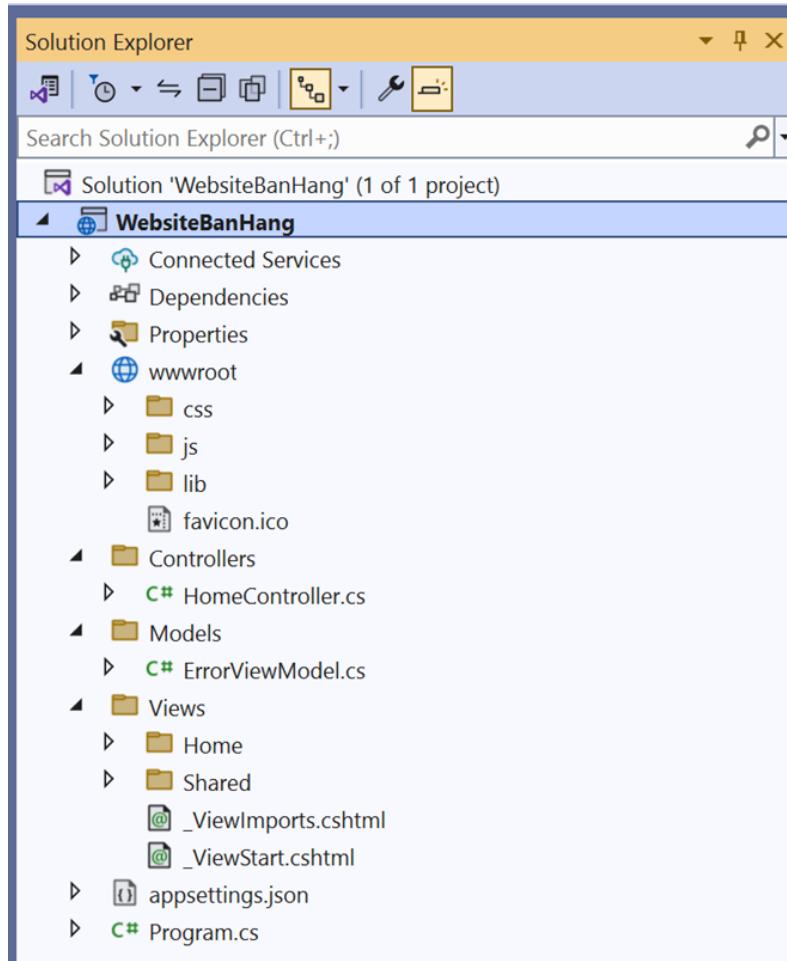


**Hình 2.2 Đặt tên cho Project**

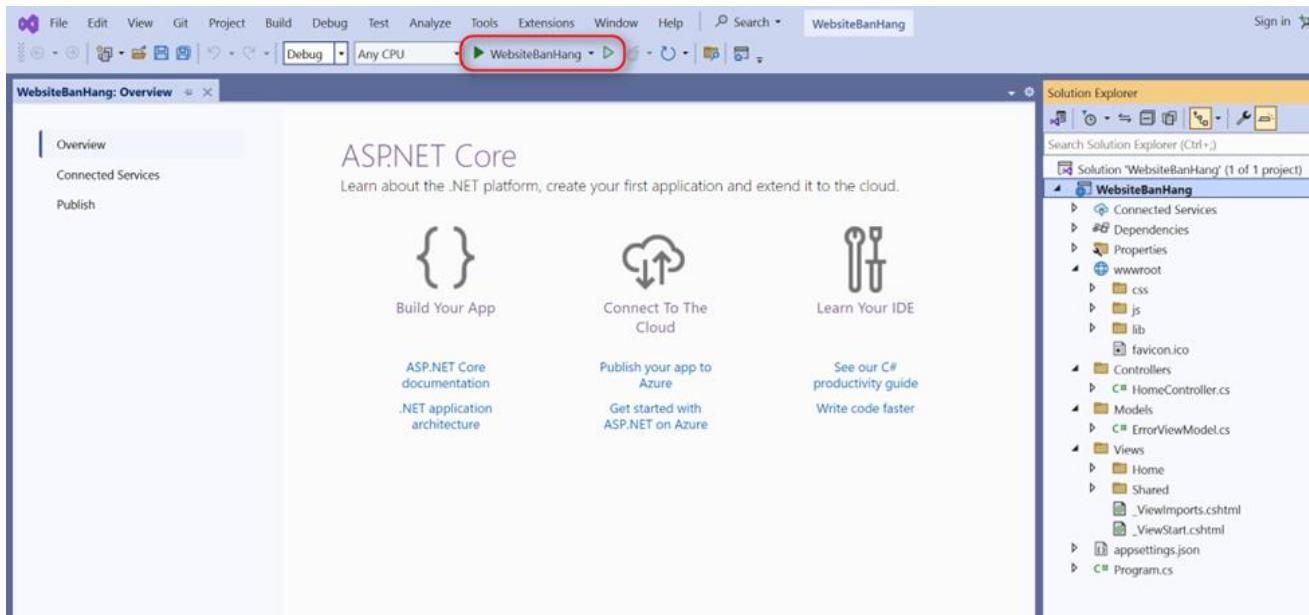
## Additional information



**Hình 2.3 Thiết lập Framework cho project**



**Hình 2.4 Cấu trúc thư mục cho project**



**Hình 2.5 Tiến hành Run project**

## 2.2 Cấu trúc dự án ASP.NET Core MVC

### 1. Dependencies:

Thư mục này bao gồm các phụ thuộc cần thiết, tức là các gói cần thiết để chạy ứng dụng ASP.NET Core.

### 2. Properties:

Thư mục này chứa tệp launchsettings.json và chỉ được sử dụng trong môi trường phát triển.

### 3. wwwroot:

Đây là thư mục gốc web của dự án. Thư mục wwwroot sẽ chứa tất cả các tệp tĩnh như .css, .js, và các tệp bootstrap, v.v.

### 4. Controllers:

Chứa các lớp điều khiển để xử lý nghiệp vụ trong ứng dụng.

### 5. Models:

Chứa các lớp Model để thao tác với dữ liệu hoặc trong ứng dụng web.

## 6. Views:

Chứa các tệp giao diện Razor (.cshtml) để hiển thị nội dung HTML. Razor là một công cụ giao diện được sử dụng trong ASP.NET để tạo HTML động

## 7. Shared:

Thư mục chứa \_Layout.cshtml. Đó là bối cảnh mặc định cho ứng dụng ASP.NET Core bao gồm các thành phần dùng chung của tất cả các trang con.

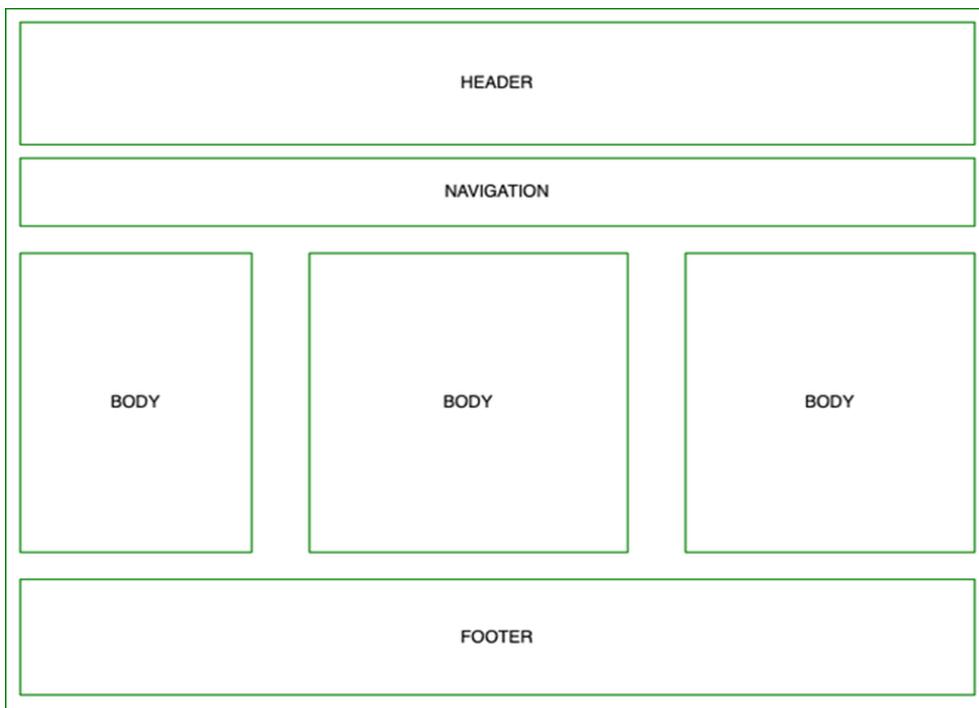
## 8. appsettings.json:

Tệp này sẽ chứa các cài đặt chung trên toàn bộ ứng dụng, như chuỗi kết nối, biến toàn cục phạm vi ứng dụng, v.v..

## 9. Program.cs:

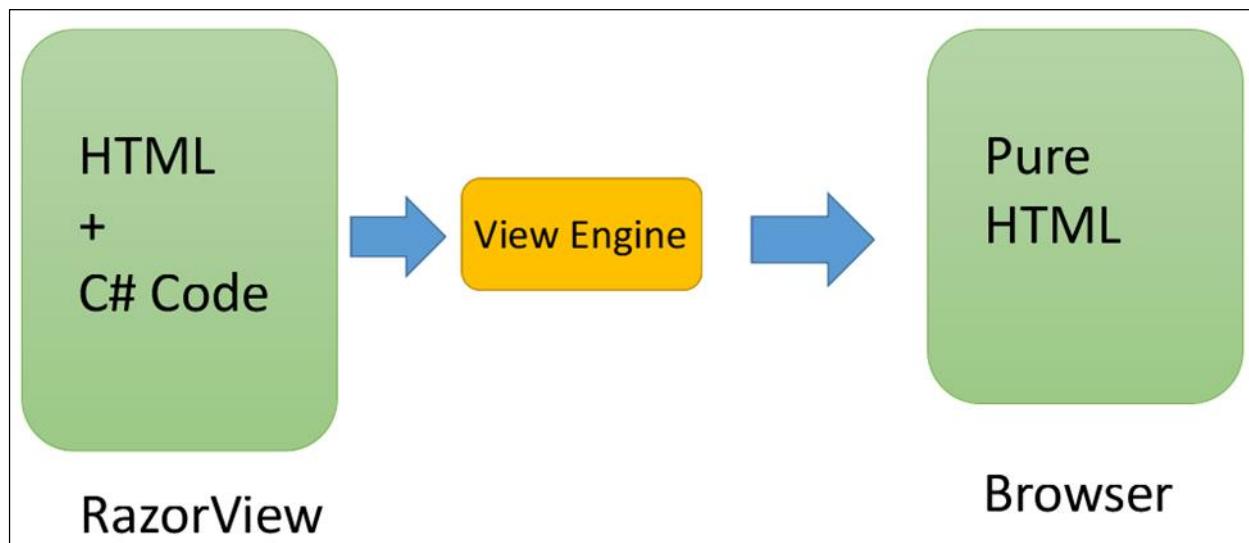
Lớp Program chứa phương thức Main. Phương thức Main chịu trách nhiệm thiết lập máy chủ web, cấu hình các dịch vụ, cấu hình các Thành phần Middleware và khởi động ứng dụng để ứng dụng có thể lắng nghe các yêu cầu từ client.

### 2.2.1 \_\_Layout



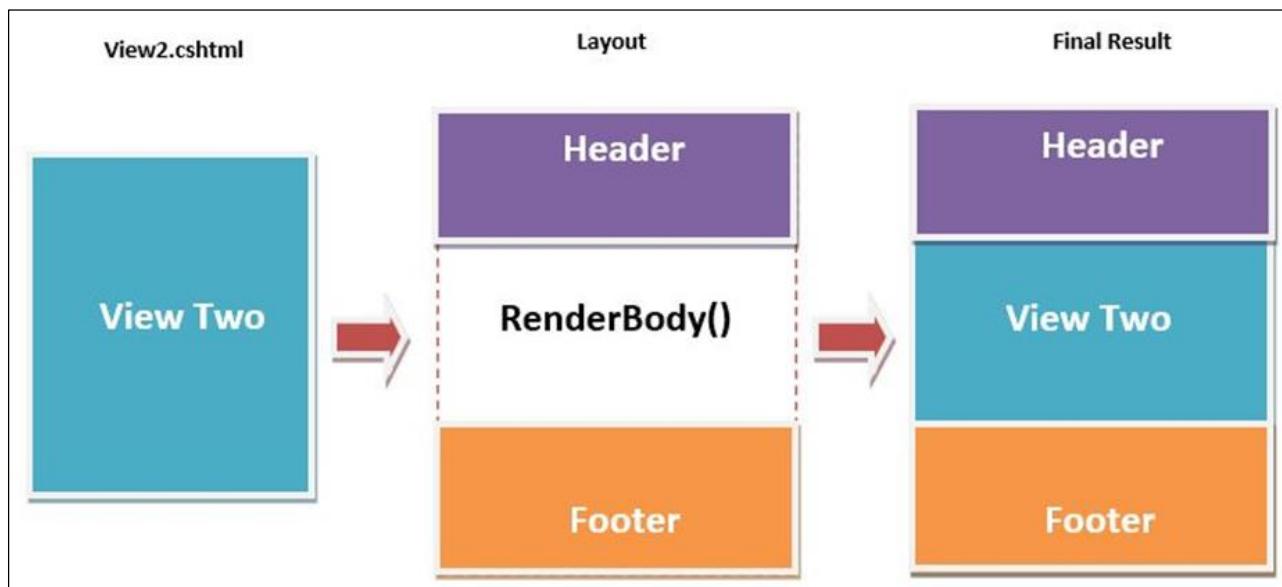
Hình 2.6 Cấu trúc của \_\_Layout

## 2.2.2 Razor View Engine



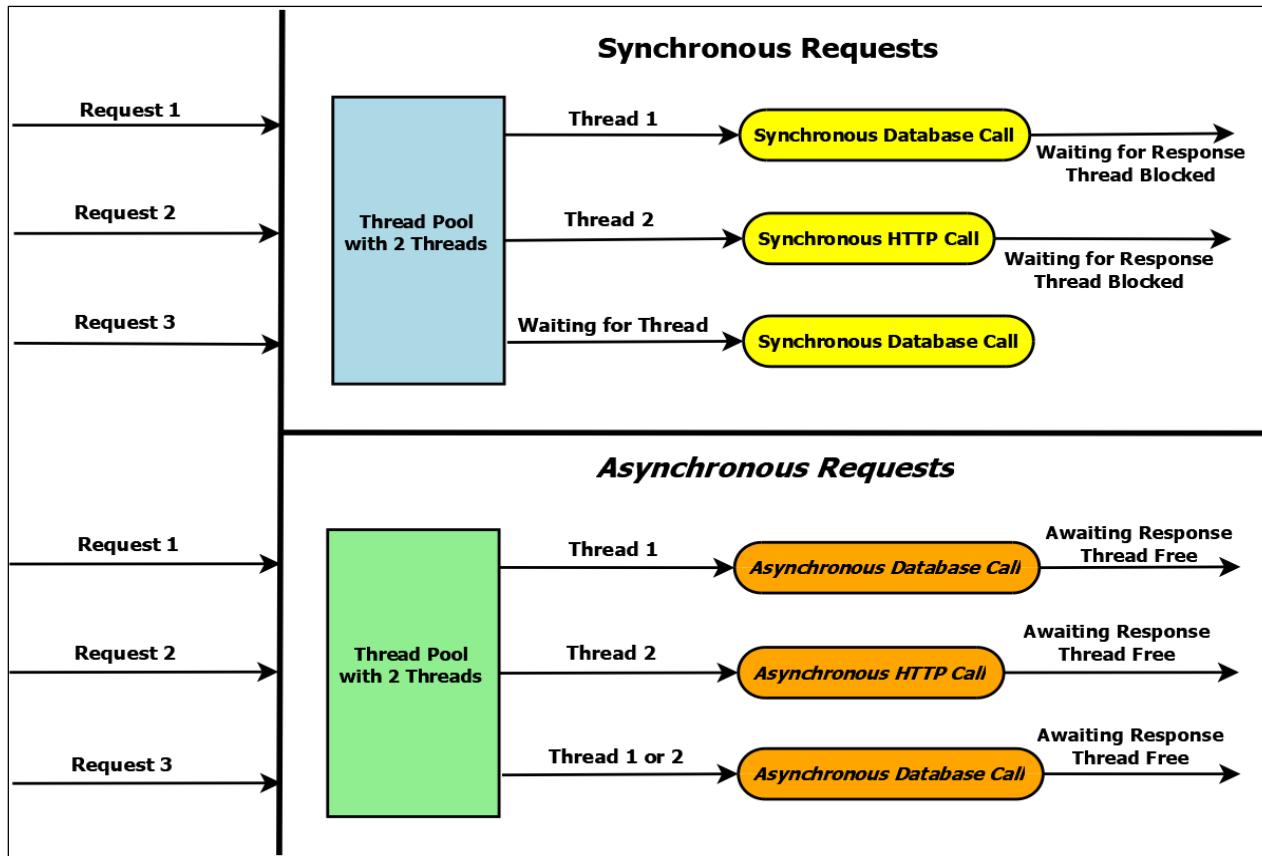
Hình 2.7 Mô hình code Razor View Engine

## 2.2.3 RenderBody



Hình 2.8 Mô hình RenderBody

## 2.2.4 RenderSectionAsync



Hình 2.9 Mô tả RenderSectionAsync

Khai báo tại trang \_\_layout

```
@await RenderSectionAsync("Scripts", required: true)
```

Khai báo tại trang View

```
@section Scripts{  
}
```

## 2.3 Bài tập

### 2.3.1 Yêu Cầu Bài Thực Hành: Ứng Dụng

Thêm/Đọc/Xóa/Sửa trên ASP.NET Core MVC

#### 2.3.1.1 Mục Đích:

Xây dựng ứng dụng Web với các thao tác CRUD sử dụng ASP.NET Core MVC, áp dụng mô hình **MVC, Model Binding, Data Annotations và sử dụng Repository Pattern** với **Mock Data**.

### 2.3.1.2 Yêu Cầu Cụ Thể:

#### 1. Cấu Hình Dự Án và Môi Trường

- Sử dụng ASP.NET Core MVC.
- Khởi tạo dự án mới với ASP.NET Core Web App (**MVC, .NET 6**).

#### 2. Tạo Models

- Tạo `Product` và `Category` models với các thuộc tính thích hợp.
- Sử dụng Data Annotations cho validation.

#### 3. Repository Pattern

- Tạo `IProductRepository` và `ICategoryRepository` interfaces.
- Implement các interfaces này với Mock Data.

#### 4. Controllers

- Tạo `ProductController` và `CategoryController` .
- Implement các actions: `Add`, `Display`, `Delete`, `Update` cho sản phẩm.

#### 5. Views

- Tạo các views tương ứng cho các chức năng trên.

#### 6. Routing và Navigation

- Cấu hình routing và tạo menu điều hướng.

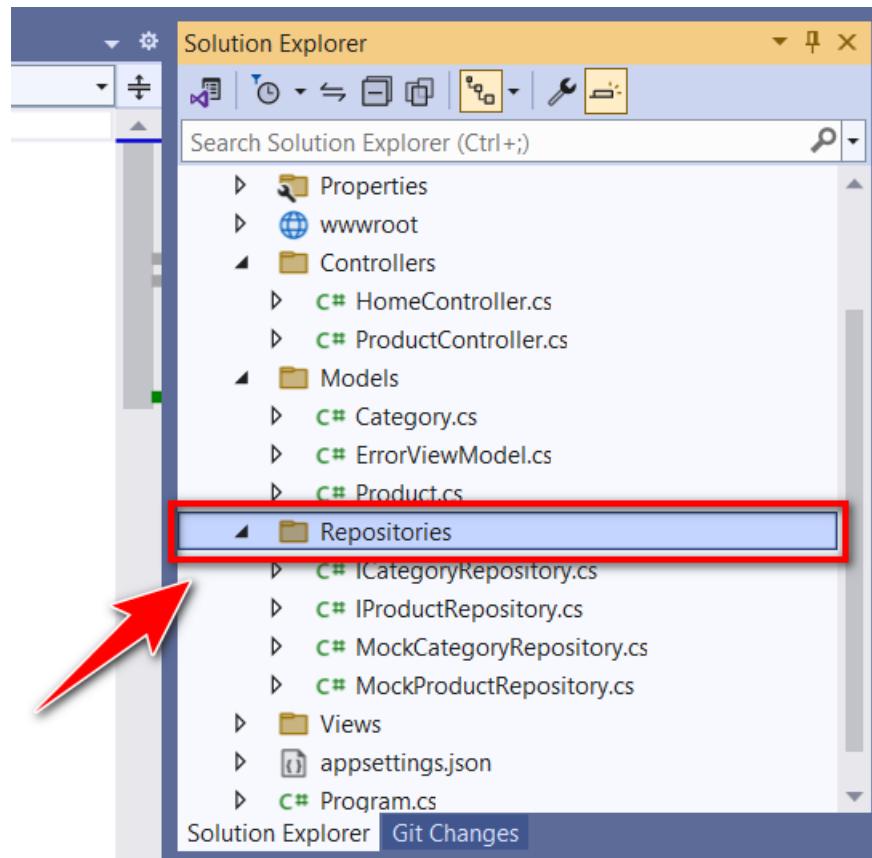
### 2.3.2 Code Mẫu Chi Tiết

- *Models*

```
// Product.cs
public class Product
{
    public int Id { get; set; }
    [Required, StringLength(100)]
    public string Name { get; set; }
    [Range(0.01, 10000.00)]
    public decimal Price { get; set; }
    public string Description { get; set; }
    public int CategoryId { get; set; }
}

// Category.cs
public class Category
{
    public int Id { get; set; }
    [Required, StringLength(50)]
    public string Name { get; set; }
}
```

- *Tiến hành tạo thư mục **Repositories**:*



- *IProductRepository.cs*

```
using System.Collections.Generic;
using YourNamespace.Models; // Thay thế bằng namespace thực tế của bạn

public interface IProductRepository
{
    IEnumerable<Product> GetAll();
    Product GetById(int id);
    void Add(Product product);
    void Update(Product product);
    void Delete(int id);
}
```

- *MockProductRepository.cs*

```
using System.Collections.Generic;
using System.Linq;
using YourNamespace.Models; // Thay thế bằng namespace thực tế của bạn

public class MockProductRepository : IProductRepository
{
    private readonly List<Product> _products;

    public MockProductRepository()
    {
        // Tạo một số dữ liệu mẫu
        _products = new List<Product>
        {
            new Product { Id = 1, Name = "Laptop", Price = 1000,
Description = "A high-end laptop",
                // Thêm các sản phẩm khác
            };
        }

        public IEnumerable<Product> GetAll()
        {
            return _products;
        }

        public Product GetById(int id)
        {
            return _products.FirstOrDefault(p => p.Id == id);
        }

        public void Add(Product product)
        {
            product.Id = _products.Max(p => p.Id) + 1;
        }
    }
}
```

```

        _products.Add(product);
    }

    public void Update(Product product)
    {
        var index = _products.FindIndex(p => p.Id == product.Id);
        if (index != -1)
        {
            _products[index] = product;
        }
    }

    public void Delete(int id)
    {
        var product = _products.FirstOrDefault(p => p.Id == id);
        if (product != null)
        {
            _products.Remove(product);
        }
    }
}

```

- *ICategoryRepository*

```

public interface ICategoryRepository
{
    IEnumerable<Category> GetAllCategories();
}

```

- *MockCategoryRepository.cs*

```

public class MockCategoryRepository : ICategoryRepository
{
    private List<Category> _categoryList;

    public MockCategoryRepository()
    {
        _categoryList = new List<Category>
        {
            new Category { Id = 1, Name = "Laptop" },
            new Category { Id = 2, Name = "Desktop" },
            // Thêm các category khác
        };
    }

    public IEnumerable<Category> GetAllCategories()
    {
        return _categoryList;
    }
}

```

- *ProductController.cs*

```
using Microsoft.AspNetCore.Mvc;
using YourNamespace.Models; // Thay thế bằng namespace thực tế của bạn
using YourNamespace.Repositories; // Thay thế bằng namespace thực tế
                                của bạn

public class ProductController : Controller
{
    private readonly IProductRepository _productRepository;
    private readonly ICategoryRepository _categoryRepository;

    public ProductController(IProductRepository productRepository,
    ICategoryRepository categoryRepository)
    {
        _productRepository = productRepository;
        _categoryRepository = categoryRepository;
    }

    public IActionResult Add()
    {
        var categories = _categoryRepository.GetAllCategories();
        ViewBag.Categories = new SelectList(categories, "Id", "Name");
        return View();
    }

    [HttpPost]
    public IActionResult Add(Product product)
    {
        if (ModelState.IsValid)
        {
            _productRepository.Add(product);
            return RedirectToAction("Index"); // Chuyển hướng tới trang
danh sách sản phẩm
        }
        return View(product);
    }

    // Các actions khác như Display, Update, Delete

    // Display a list of products
    public IActionResult Index()
    {
        var products = _productRepository.GetAll();
        return View(products);
    }
}
```

```
// Display a single product
public IActionResult Display(int id)
{
    var product = _productRepository.GetById(id);
    if (product == null)
    {
        return NotFound();
    }
    return View(product);
}

// Show the product update form
public IActionResult Update(int id)
{
    var product = _productRepository.GetById(id);
    if (product == null)
    {
        return NotFound();
    }
    return View(product);
}

// Process the product update
[HttpPost]
public IActionResult Update(Product product)
{
    if (ModelState.IsValid)
    {
        _productRepository.Update(product);
        return RedirectToAction("Index");
    }
    return View(product);
}

// Show the product delete confirmation
public IActionResult Delete(int id)
{
    var product = _productRepository.GetById(id);
    if (product == null)
    {
        return NotFound();
    }
    return View(product);
}

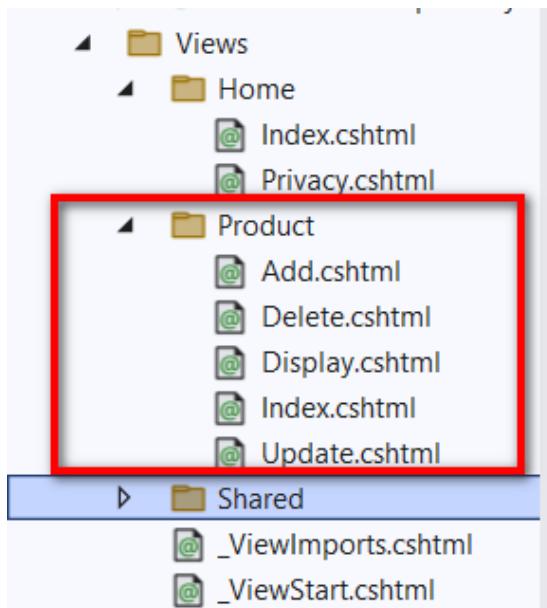
// Process the product deletion
[HttpPost, ActionName("DeleteConfirmed")]
public IActionResult DeleteConfirmed(int id)
```

```
{  
    _productRepository.Delete(id);  
    return RedirectToAction("Index");  
}  
}
```

- *Program.cs*

```
var builder = WebApplication.CreateBuilder(args);  
  
// Add services to the container.  
builder.Services.AddControllersWithViews();  
  
builder.Services.AddSingleton<IProductRepository,  
MockProductRepository>();  
builder.Services.AddScoped<ICategoryRepository,  
MockCategoryRepository>();  
  
var app = builder.Build();  
  
// Configure the HTTP request pipeline.  
if (!app.Environment.IsDevelopment())  
{  
    app.UseExceptionHandler("/Home/Error");  
    app.UseHsts();  
}  
  
app.UseHttpsRedirection();  
app.UseStaticFiles();  
app.UseRouting();  
  
app.UseAuthorization();  
  
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}");  
  
app.Run();
```

- Tạo folder **Product** trong folder **Views** chứa các file bên dưới:



- *Add.cshtml*

```
@model YourNamespace.Models.Product
@using Microsoft.AspNetCore.Mvc.Rendering
 @{
     ViewData["Title"] = "Add Product";
 }

<h1>Add Product</h1>

<form asp-action="Add">
    <div asp-validation-summary="All" class="text-danger"></div>
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Price"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Description"></label>
        <textarea asp-for="Description" class="form-control"></textarea>
        <span asp-validation-for="Description" class="text-danger"></span>
    </div>
</form>
```

```
<div class="form-group">
    <label asp-for="CategoryId">Category</label>
    <select asp-for="CategoryId" asp-items="ViewBag.Categories"
    class="form-control"></select>
</div>
<button type="submit" class="btn btn-primary">Add</button>
</form>
```

### Giải Thích

- **Index**: Hiển thị danh sách tất cả sản phẩm.
  - **Display**: Hiển thị thông tin chi tiết của một sản phẩm cụ thể.
  - **Update** (GET): Hiển thị form để cập nhật thông tin sản phẩm.
  - **Update** (POST): Xử lý việc cập nhật sản phẩm.
  - **Delete** (GET): Hiển thị xác nhận xóa sản phẩm.
  - **DeleteConfirmed**: Xử lý việc xóa sản phẩm từ database.
- Đối với mỗi action, bạn sẽ cần tạo các views tương ứng trong thư mục **Views/Product**, ví dụ **Index.cshtml**, **Display.cshtml**, **Update.cshtml**, và **Delete.cshtml**.
- Đảm bảo rằng bạn đã đăng ký **IProductRepository** và **implementation** của nó (ví dụ: **MockProductRepository**) trong **Program.cs** của ứng dụng để **Dependency Injection** hoạt động đúng cách.
- *Index.cshtml (Hiển Thị Danh Sách Sản Phẩm)*

```
@model IEnumerable<YourNamespace.Models.Product>

<h2>Products</h2>

<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Description</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var product in Model)
        {
            <tr>
```

```

<td>@product.Name</td>
<td>@product.Price</td>
<td>@product.Description</td>
<td>
    <a asp-action="Display" asp-route-id="@product.Id">View</a> |
    <a asp-action="Update" asp-route-id="@product.Id">Edit</a> |
    <a asp-action="Delete" asp-route-id="@product.Id">Delete</a>
</td>
</tr>
}
</tbody>
</table>

```

- *Display.cshtml (Hiển Thị Thông Tin Chi Tiết Sản Phẩm)*

```

@model YourNamespace.Models.Product

<h2>Product Details</h2>

<div>
    <h4>Name: @Model.Name</h4>
    <h4>Price: @Model.Price</h4>
    <h4>Description: @Model.Description</h4>
</div>

<a asp-action="Index">Back to List</a>

```

- *Delete.cshtml (Xác Nhận Xóa Sản Phẩm)*

```

@model YourNamespace.Models.Product

<h2>Are you sure you want to delete this?</h2>

<div>
    <h4>Name: @Model.Name</h4>
    <h4>Price: @Model.Price</h4>
    <h4>Description: @Model.Description</h4>
</div>

<form asp-action="DeleteConfirmed" method="post">
    <input type="hidden" asp-for="Id" />
    <input type="submit" value="Delete" class="btn btn-danger" /> |
    <a asp-action="Index">Cancel</a>
</form>

```

- *Update.cshtml (Cập Nhật Sản Phẩm)*

```
@model YourNamespace.Models.Product

<h2>Edit Product</h2>

<form asp-action="Update">
    <input type="hidden" asp-for="Id" />
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Price"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Description"></label>
        <textarea asp-for="Description" class="form-control"></textarea>
        <span asp-validation-for="Description" class="text-danger"></span>
    </div>
    <button type="submit" class="btn btn-primary">Update</button>
</form>
```

### 2.3.3 Kết quả

The screenshot shows a browser window titled "Add Product - WebBanHang\_Lab03". The URL in the address bar is "localhost:5082/Product/Add". The page content is titled "Add Product" and contains four input fields: "Name", "Price", "Description", and "Category". The "Category" field has the value "Hoa". A blue "Add" button is at the bottom.

Name

Price

Description

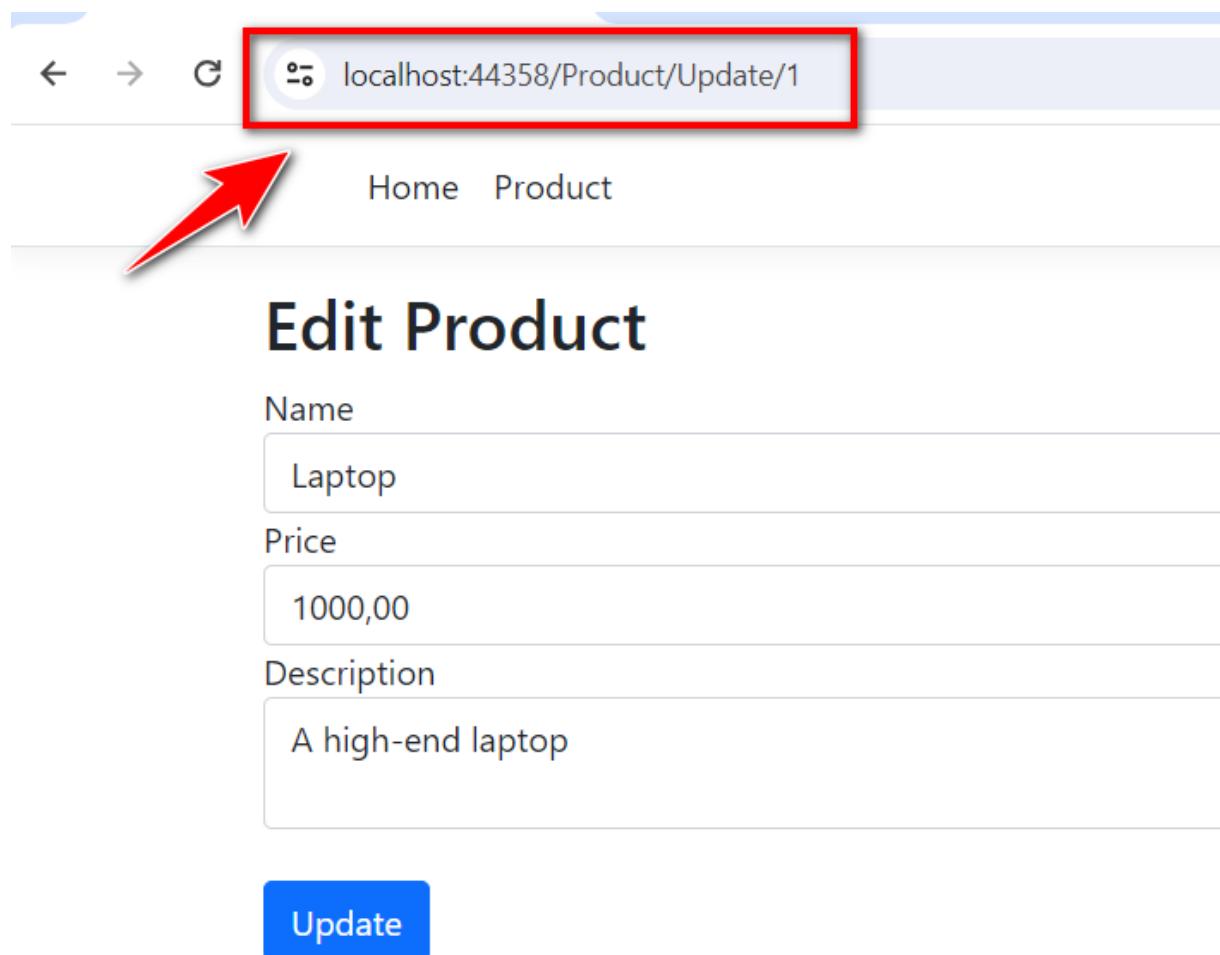
Category

Hoa

Add

The screenshot shows a browser window titled "Index - WebsiteBanHang". The URL in the address bar is "localhost:5100/Product". The page content is titled "Index" and includes a "Create New" link. Below it is a table with columns: Id, Name, Price, Description, and CategoryId. Two rows of data are shown: one for "Iphone 15" and one for "iphone 16". The row for "iphone 16" is highlighted with a red border.

Id	Name	Price	Description	CategoryId	
1	Iphone 15	35000000,00	Iphone 15 pro max	1	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
2	iphone 16	50000000,00	iphone 16	1	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>



## 2.3.4 Bổ sung tính năng upload file cho ứng dụng

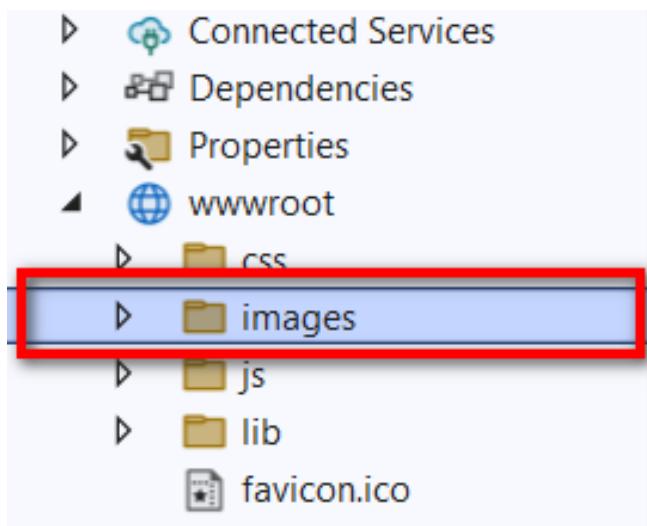
Để bổ sung tính năng thêm hình ảnh đại diện và nhiều hình ảnh cho sản phẩm trong ứng dụng trên, bạn cần mở rộng model `Product`, cập nhật views và controllers để xử lý việc upload hình ảnh. Dưới đây là các bước cơ bản để thực hiện điều này:

- *Mở Rộng Model `Product`*

Mở rộng model `Product` để bao gồm các thuộc tính cho hình ảnh đại diện và danh sách các hình ảnh:

```
public class Product
{
    // Các thuộc tính hiện có
    public string? ImageUrl { get; set; } // Đường dẫn đến hình ảnh đại diện
    public List<string>? ImageUrls { get; set; } // Danh sách các hình ảnh khác
}
```

- Tạo folder **images** trong **wwwroot** để upload hình ảnh



- Cập Nhật View 'Add.cshtml'

Cập nhật view `Add.cshtml` để thêm fields cho việc upload hình ảnh:

```
@model YourNamespace.Models.Product //thay thế Project Name của bạn

<!-- Phần còn lại của form -->

<div class="form-group">
    <label asp-for="ImageUrl">Image</label>
    <input type="file" asp-for="ImageUrl" class="form-control" />
</div>

<br>

<button type="submit" class="btn btn-primary">Add</button>
```

- Cập nhập thêm vào form **Add.cshtml** trong **Folder Product**

```
<h1>Add Product</h1>
<form asp-action="Add" enctype="multipart/form-data">
    <div asp-validation-summary="All" class="text-danger"></div>
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
```

- Xử Lý Upload Trong Controller

Cập nhật `ProductController` để xử lý việc upload hình ảnh:

```
[HttpPost]
public async Task<IActionResult> Add(Product product, IFormFile imageUrl, List<IFormFile> imageUrls)
{
    if (ModelState.IsValid)
    {
        if (imageUrl != null)
        {
            // Lưu hình ảnh đại diện
            product.ImageUrl = await SaveImage(imageUrl);
        }

        if (imageUrls != null)
        {
            product.ImageUrls = new List<string>();
            foreach (var file in imageUrls)
            {
                // Lưu các hình ảnh khác
                product.ImageUrls.Add(await SaveImage(file));
            }
        }
    }

    _productRepository.Add(product);
    return RedirectToAction("Index");
}

return View(product);
}

private async Task<string> SaveImage(IFormFile image)
{
    // Thay đổi đường dẫn theo cấu hình của bạn
    var savePath = Path.Combine("wwwroot/images", image.FileName);
    using (var fileStream = new FileStream(savePath, FileMode.Create))
    {
        await image.CopyToAsync(fileStream);
    }
    return "/images/" + image.FileName; // Trả về đường dẫn tương đối
}
```

- *Cấu Hình 'Program.cs'*

Đảm bảo rằng ứng dụng của bạn cấu hình đúng cách để phục vụ các tệp tĩnh:

```
app.UseStaticFiles(); // Cho phép ứng dụng phục vụ các tệp tĩnh từ thư mục wwwroot
```

- Kết quả:

Add Product - WebBanHang\_Lab03

localhost:5082/Product/Add

WebBanHang\_Lab03 Home Privacy Product Category

## Add Product

Name

Price

Description

Category

Hoa

Product Image

Choose File No file chosen

Additional Images

Choose File No file chosen

Add

- Hiển Thị Hình Ảnh

Cập nhật các views cần thiết để hiển thị hình ảnh của sản phẩm. Ví dụ, trong `Display.cshtml` :

```
@model YourNamespace.Models.Product
<h2>Product Details</h2>

<div>
    <h4>Name: @Model.Name</h4>
    <h4>Price: @Model.Price</h4>
    <h4>Description: @Model.Description</h4>
    

    @if (Model.ImageUrls != null)
    {
        foreach (var imageUrl in Model.ImageUrls)
        {
            img src="@imageUrl" alt="Product Image" style="width: 300px; height: auto;" />
        }
    }
</div>
```

### Kết quả:

The screenshot shows a browser window with the title "Index - WebsiteBanHang". The address bar shows "localhost:5100/Product". The page content is titled "Index" and contains a table with two rows of product data. The first row has Id 1, Name "Iphone 15", Price 35000000,00, Description "Iphone 15 pro max", and CategoryId 1. The second row has Id 2, Name "iphone 16", Price 50000000,00, Description "mo ta", and CategoryId 1. Below the table is a thumbnail image of an iPhone 15.

ID	Name	Price	Description	CategoryId
1	Iphone 15	35000000,00	Iphone 15 pro max	1
2	iphone 16	50000000,00	mo ta	1

**Lưu ý:** Thêm kiểm tra và xử lý lỗi cho các tình huống như tệp không phải hình ảnh được tải lên hoặc kích thước tệp quá lớn.

### 2.3.5 Yêu cầu bổ sung

Áp dụng giao diện đã thiết kế ở LAB 1 cho ứng dụng website bán hàng

# BÀI 3: XÂY DỰNG ỨNG DỤNG WEBSITE BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 1)

Sau khi học xong bài này, sinh viên có thể nắm được:

- Học cách sử dụng Entity Framework Core để tương tác với CSDL MS SQL Server.
- Xây dựng ứng dụng web bán hàng dựa trên ASP.NET Core MVC, kết hợp với cơ sở dữ liệu và Entity Framework Core.

## 3.1 Bài tập

### 3.1.1 Yêu cầu

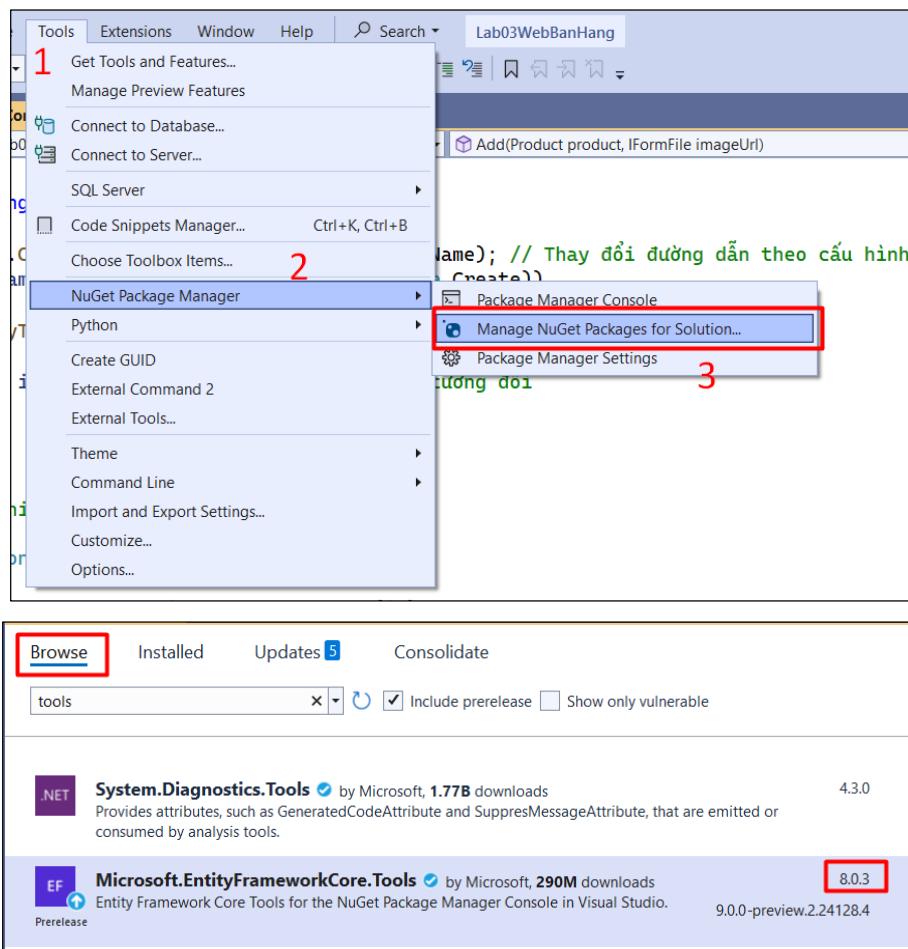
Xây dựng web bán hàng có các chức năng Hiển thị/ Thêm/ Xóa/ Sửa với Entity Framework Core và Cơ sở dữ liệu MS SQL Server

### 3.1.2 Hướng dẫn thực hiện

- *Cài Đặt Gói NuGet Cần Thiết*

Cài đặt các gói sau bằng NuGet:

- `Microsoft.EntityFrameworkCore` → phiên bản 8.0.3
- `Microsoft.EntityFrameworkCore.SqlServer` → phiên bản 8.0.3 (hoặc provider cơ sở dữ liệu khác)
- `Microsoft.EntityFrameworkCore.Tools` → phiên bản 8.0.3



- Cài đặt Tương tự cho các Nuget khác.
- Thiết Kế Models

Trong thư Models Tạo các models 'Product' và 'Category':

```
public class Product
{
    public int Id { get; set; }
    [Required, StringLength(100)]
    public string Name { get; set; }
    [Range(0.01, 10000.00)]
    public decimal Price { get; set; }
    public string Description { get; set; }
    public string? ImageUrl { get; set; }
    public List<ProductImage>? Images { get; set; }
    public int CategoryId { get; set; }
    public Category? Category { get; set; }
}

public class Category
```

```

public int Id { get; set; }
[Required, StringLength(50)]
public string Name { get; set; }
public List<Product>? Products { get; set; }
}

public class ProductImage
{
    public int Id { get; set; }
    public string Url { get; set; }
    public int ProductId { get; set; }
    public Product? Product { get; set; }
}

```

- *Cấu Hình Entity Framework Core*

Tạo một lớp `ApplicationDbContext` trong thư mục Models và cấu hình:

```

using Microsoft.EntityFrameworkCore;

public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
    }

    public DbSet<Product> Products { get; set; }
    public DbSet<Category> Categories { get; set; }
    public DbSet<ProductImage> ProductImages { get; set; }
}

```

- *Cấu hình EF Core trong file `Program.cs`:*

```

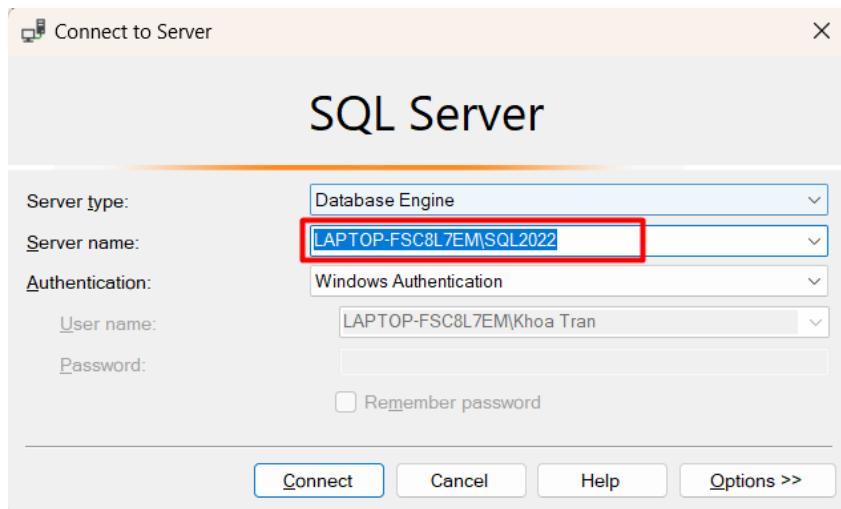
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("Default
Connection")));

// Các cấu hình khác

```

- *Cấu hình connection string trong `appsettings.json`.*
- *Xem tên Server name trong SQL Server*

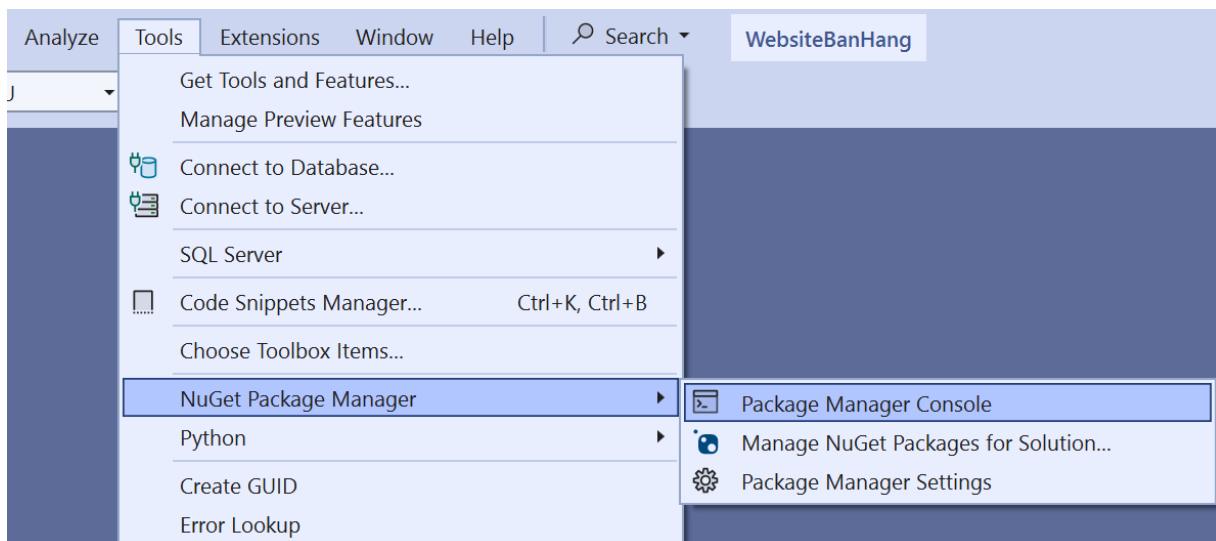


```
".ConnectionStrings": {
  "DefaultConnection": {
    "Server=Servername;Database=DbName;Trusted_Connection=True;TrustServerCertificate=True"
  }
},
```

Lưu ý: Thay bằng thông tin **Server name** và **Database** của mình

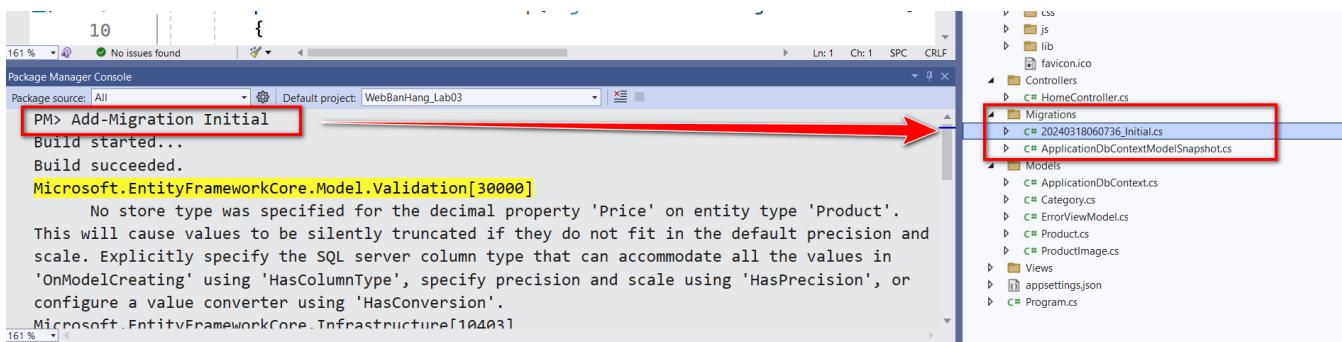


- *Tạo Migrations*



- Tạo migrations để tạo cơ sở dữ liệu:

### Add-Migration Initial

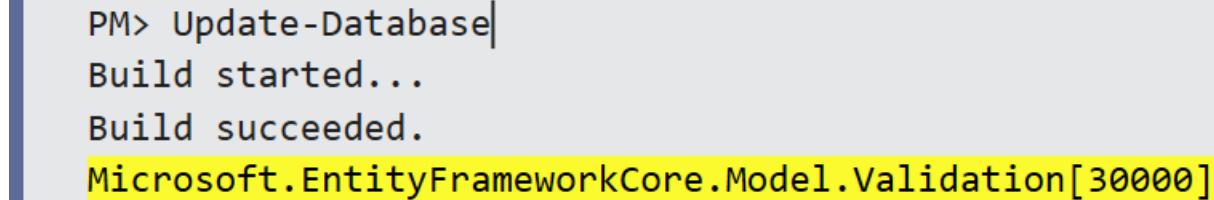


```
PM> Add-Migration Initial
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Model.Validation[30000]
No store type was specified for the decimal property 'Price' on entity type 'Product'.
This will cause values to be silently truncated if they do not fit in the default precision and
scale. Explicitly specify the SQL server column type that can accommodate all the values in
'OnModelCreating' using 'HasColumnType', specify precision and scale using 'HasPrecision', or
configure a value converter using 'HasConversion'.
Microsoft.EntityFrameworkCore.Infrastructure[10403]
```

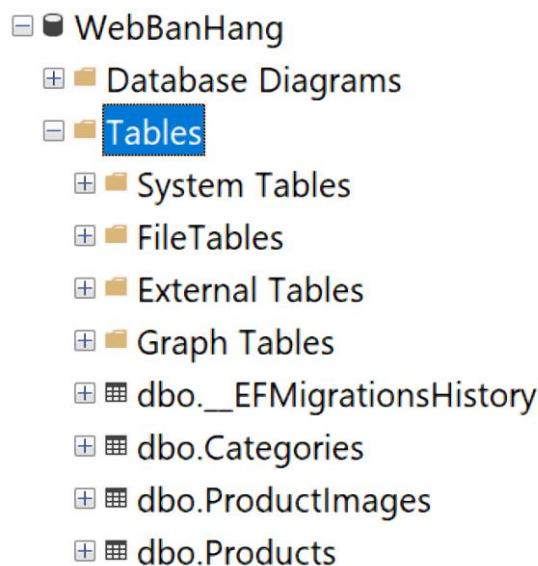
The file tree on the right shows the project structure with a red box highlighting the 'Migrations' folder containing the generated migration file '20240318060736\_Initial.cs'.

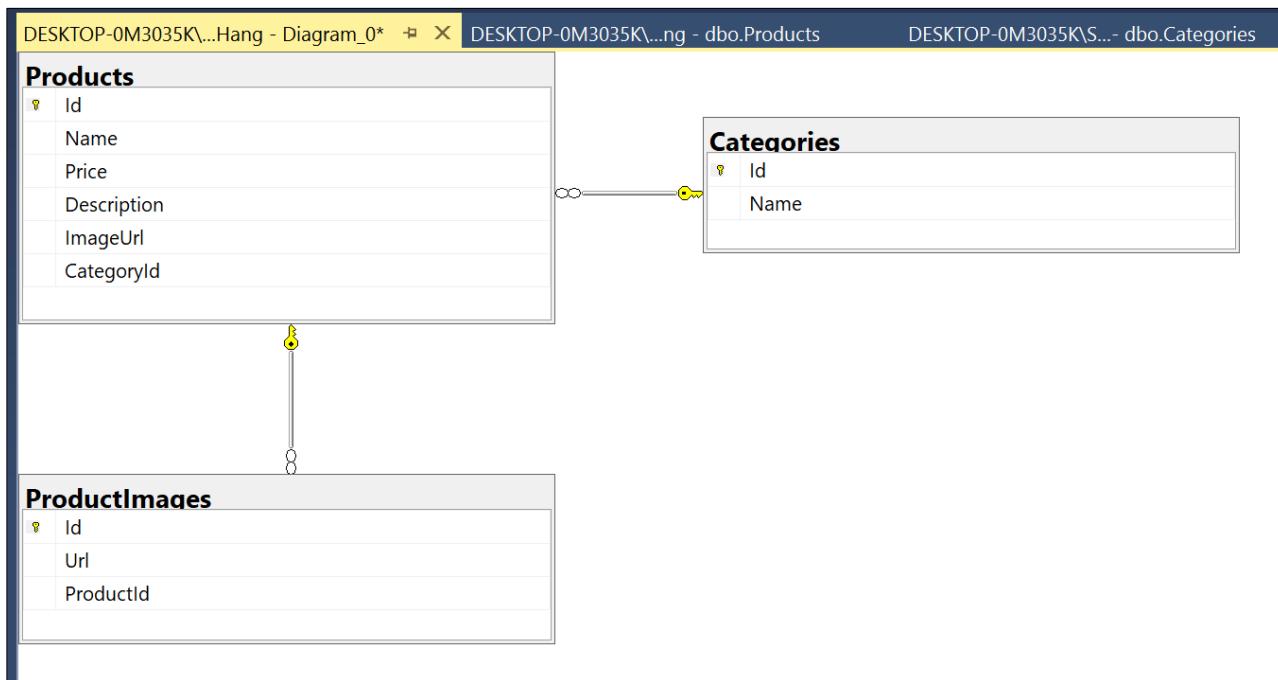
- Update Database

### Update-Database



```
PM> Update-Database
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Model.Validation[30000]
```





Tạo thư mục **Repositories** và thêm các file sau:

- **IProductRepository.cs** và **ICategoryRepository.cs**

Đây là các interface định nghĩa các phương thức cần thiết để tương tác với cơ sở dữ liệu cho Product và Category.

```

public interface IProductRepository
{
    Task<IEnumerable<Product>> GetAllAsync();
    Task<Product> GetByIdAsync(int id);
    Task AddAsync(Product product);
    Task UpdateAsync(Product product);
    Task DeleteAsync(int id);
}

public interface ICategoryRepository
{
    Task<IEnumerable<Category>> GetAllAsync();
    Task<Category> GetByIdAsync(int id);
    Task AddAsync(Category category);
    Task UpdateAsync(Category category);
    Task DeleteAsync(int id);
}
  
```

- *EFProductRepository* và *EFCategoryRepository*

```
public class EFProductRepository : IProductRepository
{
    private readonly ApplicationDbContext _context;

    public EFProductRepository(ApplicationDbContext context)
    {
        _context = context;
    }

    public async Task<IEnumerable<Product>> GetAllAsync()
    {
        // return await _context.Products.ToListAsync();
        return await _context.Products
            .Include(p => p.Category) // Include thông tin về category
            .ToListAsync();
    }

    public async Task<Product> GetByIdAsync(int id)
    {
        // return await _context.Products.FindAsync(id);
        // lấy thông tin kèm theo category
        return await _context.Products.Include(p =>
p.Category).FirstOrDefaultAsync(p => p.Id == id);
    }

    public async Task AddAsync(Product product)
    {
        _context.Products.Add(product);
        await _context.SaveChangesAsync();
    }

    public async Task UpdateAsync(Product product)
    {
        _context.Products.Update(product);
        await _context.SaveChangesAsync();
    }

    public async Task DeleteAsync(int id)
    {
        var product = await _context.Products.FindAsync(id);
        _context.Products.Remove(product);
        await _context.SaveChangesAsync();
    }
}
```

```

public class EFCategoryRepository : ICategoryRepository
{
    private readonly ApplicationDbContext _context;

    public EFCategoryRepository(ApplicationDbContext context)
    {
        _context = context;
    }

    // Tương tự như EFProductRepository, nhưng cho Category
}

```

- Đăng Ký trong Program.cs

```

builder.Services.AddScoped<IPrductRepository, EFProductRepository>();
builder.Services.AddScoped<ICategoryRepository, EFCategoryRepository>();

```

```

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))); ①

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddScoped<IPrductRepository, EFProductRepository>(); ②
builder.Services.AddScoped<ICategoryRepository, EFCategoryRepository>();

```

```
var app = builder.Build();
```

- Tạo và Cập Nhật Controllers

Sử Dụng Repository trong Controllers

```

using ProjectName.Models;
using ProjectName.Repositories;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace ProjectName.Controllers
{
    public class ProductController : Controller
    {
        private readonly IPrductRepository _productRepository;
        private readonly ICategoryRepository _categoryRepository;
        public ProductController(IPrductRepository productRepository,
        ICategoryRepository categoryRepository)
        {
            _productRepository = productRepository;
            _categoryRepository = categoryRepository;
        }
    }
}

```

```

    }

    // Hiển thị danh sách sản phẩm
    public async Task<IActionResult> Index()
    {
        var products = await _productRepository.GetAllAsync();
        return View(products);
    }
    // Hiển thị form thêm sản phẩm mới
    public async Task<IActionResult> Add()
    {
        var categories = await _categoryRepository.GetAllAsync();
        ViewBag.Categories = new SelectList(categories, "Id", "Name");

        return View();
    }

    // Xử lý thêm sản phẩm mới
    [HttpPost]
    public async Task<IActionResult> Add(Product product, IFormFile
imageUrl)
    {
        if (ModelState.IsValid)
        {
            if (imageUrl != null)
            {
                // Lưu hình ảnh đại diện tham khảo bài 02 hàm SaveImage
                product.ImageUrl = await SaveImage(imageUrl);
            }

            await _productRepository.AddAsync(product);
            return RedirectToAction(nameof(Index));
        }
        // Nếu ModelState không hợp lệ, hiển thị form với dữ liệu đã nhập
        var categories = await _categoryRepository.GetAllAsync();
        ViewBag.Categories = new SelectList(categories, "Id", "Name");
        return View(product);
    }

    // Viết thêm hàm SaveImage (tham khảo bài 02)

    private async Task<string> SaveImage(IFormFile image)
    {
        //Thay đổi đường dẫn theo cấu hình của bạn
        var savePath = Path.Combine("wwwroot/images", image.FileName);
        using (var fileStream = new FileStream(savePath, FileMode.Create))
        {
            await image.CopyToAsync(fileStream);
        }
        return "/images/" + image.FileName; // Trả về đường dẫn tương đối
    }

    //Nhớ tạo folder images trong wwwroot
}

```

```
// Hiển thị thông tin chi tiết sản phẩm
public async Task<IActionResult> Display(int id)
{
    var product = await _productRepository.GetByIdAsync(id);
    if (product == null)
    {
        return NotFound();
    }
    return View(product);
}
// Hiển thị form cập nhật sản phẩm
public async Task<IActionResult> Update(int id)
{
    var product = await _productRepository.GetByIdAsync(id);
    if (product == null)
    {
        return NotFound();
    }

    var categories = await _categoryRepository.GetAllAsync();
    ViewBag.Categories = new SelectList(categories, "Id", "Name",
product.CategoryId);
    return View(product);
}
// Xử lý cập nhật sản phẩm
[HttpPost]
public async Task<IActionResult> Update(int id, Product product,
IFormFile imageUrl)
{
    ModelState.Remove("ImageUrl"); // Loại bỏ xác thực ModelState cho
ImageUrl
    if (id != product.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {

        var existingProduct = await
_productRepository.GetByIdAsync(id); // Giả định có phương thức GetByIdAsync
tải lên
        // Giữ nguyên thông tin hình ảnh nếu không có hình mới được
tải lên
        if (imageUrl == null)
        {
            product.ImageUrl = existingProduct.ImageUrl;
        }
        else
        {
            // Lưu hình ảnh mới
            product.ImageUrl = await SaveImage(imageUrl);
        }
    }
}
```

```

        // Cập nhật các thông tin khác của sản phẩm
        existingProduct.Name = product.Name;
        existingProduct.Price = product.Price;
        existingProduct.Description = product.Description;
        existingProduct.CategoryId = product.CategoryId;
        existingProduct.ImageUrl = product.ImageUrl;

        await _productRepository.UpdateAsync(existingProduct);

        return RedirectToAction(nameof(Index));
    }
    var categories = await _categoryRepository.GetAllAsync();
    ViewBag.Categories = new SelectList(categories, "Id", "Name");
    return View(product);
}

// Hiển thị form xác nhận xóa sản phẩm
public async Task<IActionResult> Delete(int id)
{
    var product = await _productRepository.GetByIdAsync(id);
    if (product == null)
    {
        return NotFound();
    }
    return View(product);
}

// Xử lý xóa sản phẩm
[HttpPost, ActionName("DeleteConfirmed")]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    await _productRepository.DeleteAsync(id);
    return RedirectToAction(nameof(Index));
}
}
}

```

- *Tạo và Cập nhật Views*

Xây dựng các view của `ProductController` trong ASP.NET Core MVC, bao gồm các trang hiển thị danh sách sản phẩm, thêm sản phẩm mới, cập nhật thông tin sản phẩm, và xóa sản phẩm.

- *Index.cshtml (Danh sách sản phẩm)*

```

@model IEnumerable<YourNamespace.Models.Product>



## Products






```

```

<tr>
    <th>Name</th>
    <th>Price</th>
    <th>Description</th>
    <th>Category</th>
    <th>Actions</th>
</tr>
</thead>
<tbody>
@foreach (var product in Model)
{
    <tr>
        <td>@product.Name</td>
        <td>@product.Price</td>
        <td>@product.Description</td>
        <td>@product.Category?.Name</td>
        <td>
            <a asp-action="Display" asp-route-id="@product.Id">View</a> |
            <a asp-action="Update" asp-route-id="@product.Id">Edit</a> |
            <a asp-action="Delete" asp-route-id="@product.Id">Delete</a>
        </td>
    </tr>
}
</tbody>
</table>

```

- *Add.cshtml (Thêm sản phẩm mới)*

```

@model YourNamespace.Models.Product
@using Microsoft.AspNetCore.Mvc.Rendering
 @{
     ViewData["Title"] = "Add Product";
 }

<h2>Add Product</h2>

<form asp-action="Add" method="post" enctype="multipart/form-data">
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Price"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>
    <div class="form-group">

```

```

        <label asp-for="Description"></label>
        <textarea asp-for="Description" class="form-control"></textarea>
        <span asp-validation-for="Description" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="CategoryId">Category</label>
        <select asp-for="CategoryId" asp-items="ViewBag.Categories"
class="form-control"></select>
    </div>
    <div class="form-group">
        <label asp-for="ImageUrl">Product Image</label>
        <input type="file" asp-for="ImageUrl" class="form-control" />
    </div>
    <button type="submit" class="btn btn-primary">Add</button>
</form>

```

- *Display.cshtml* (Hiển thị thông tin chi tiết sản phẩm)

```

@model YourNamespace.Models.Product

<h2>@Model.Name</h2>

<div>
    <h3>Price: @Model.Price</h3>
    <p>@Model.Description</p>
    @if (!string.IsNullOrEmpty(Model.ImageUrl))
    {
        
    }
</div>

<a asp-action="Index">Back to List</a>

```

- *Update.cshtml* (Cập nhật thông tin sản phẩm)

```

@model YourNamespace.Models.Product
@using Microsoft.AspNetCore.Mvc.Rendering
 @{
     ViewData["Title"] = "Edit Product";
 }

<h2>Edit Product</h2>

<form asp-action="Update" method="post" enctype="multipart/form-data">
    <input type="hidden" asp-for="Id" />

```

```

<div class="form-group">
    <label asp-for="Name"></label>
    <input asp-for="Name" class="form-control" />
    <span asp-validation-for="Name" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="Price"></label>
    <input asp-for="Price" class="form-control" />
    <span asp-validation-for="Price" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="Description"></label>
    <textarea asp-for="Description" class="form-control"></textarea>
    <span asp-validation-for="Description" class="text-
danger"></span>
</div>
<div class="form-group">
    <label asp-for="CategoryId">Category</label>
    <select asp-for="CategoryId" asp-items="@ViewBag.Categories"
class="form-control"></select>
</div>
<div class="form-group">
    <label asp-for="ImageUrl">Product Image</label>
    <input type="file" asp-for="ImageUrl" class="form-control" />
    
</div>
    <button type="submit" class="btn btn-primary">Update</button>
</form>

```

- *Delete.cshtml (Xác nhận xóa sản phẩm)*

```

@model YourNamespace.Models.Product

<h2>Delete Product</h2>

<form asp-action="DeleteConfirmed" method="post">
    <input type="hidden" asp-for="Id" />
    <p>Are you sure you want to delete @Model.Name?</p>
    <button type="submit" class="btn btn-danger">Delete</button>
</form>

```

### 3.1.3 Kết quả

Kết quả, hướng dẫn trên chỉ cơ bản. Sinh viên hoàn thiện thêm để được kết qua bên dưới. Áp dụng bài số 2 (thiết kế giao diện) để hoàn thiện thêm.

Trang chủ website.

Products					
Tên	Giá	Mô tả	Ảnh sản phẩm	Loại	Chức năng
Lập trình C	234.000,00 đ	Sách bộ môn CNPM		Lập trình	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Delete</a>
Tôi tài giỏi bạn cũng thế	129.900,00 đ	Sách văn hoá		Văn hoá	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Delete</a>
Nhà Giả Kim	123.000,00 đ	123		Văn hoá	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Delete</a>

Trang chi tiết sản phẩm.

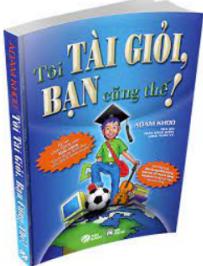
Chi Tiết Sản Phẩm - Web bán hàng

localhost:5198/Product/Display/11

Web bán hàng Home Thêm sản phẩm

## Chi Tiết Sản Phẩm

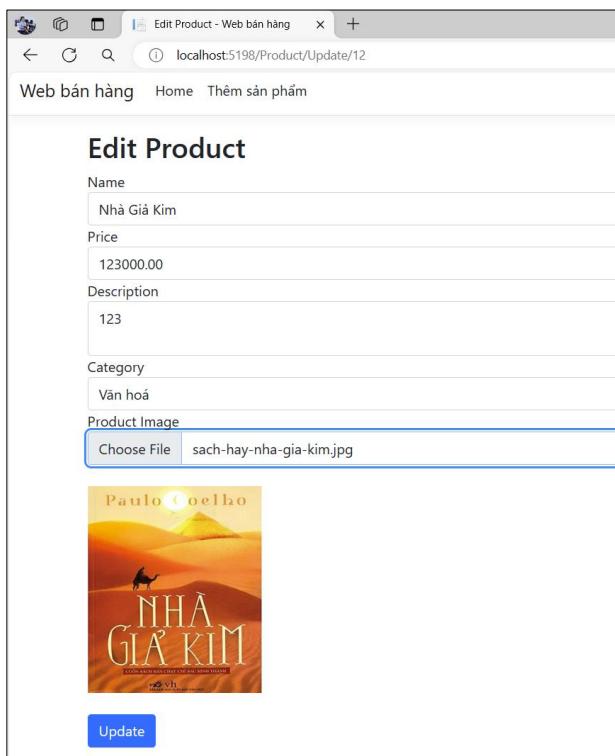
### Tôi tài giỏi bạn cũng thế



**Giá:** 129900.00đ  
**Mô tả:** Sách văn hoá  
**Loại:** Văn hoá

[Chỉnh Sửa](#) [Quay lại Danh sách](#)

Trang Edit sản phẩm.



### 3.1.4 Yêu cầu bổ sung

#### Yêu cầu:

Ở trang chỉnh sửa sản phẩm **Update.cshtml**, khi thay đổi ảnh khác thì ảnh sẽ cập nhật hình ảnh hiện ngay trên trang.

Gợi ý dùng:

```
<div class="form-group">
    <label asp-for="ImageUrl">Product Image</label>
    <input type="file" asp-for="ImageUrl" class="form-control"
id="imageInput" />
    <div class="form-group">
        <input type="submit" value="Save" class="btn btn-primary"
/>
    </div>
</form>
</div>
<div class="col-8">
    
</div>
</div>
```

```

<script>
    document.querySelectorAll('input[type="file"]').forEach(input => {
        input.addEventListener('change', function (event) {
            const file = event.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onload = function (e) {
                    const img = document.getElementById("previewImage")
                    img.src = e.target.result
                };
                reader.readAsDataURL(file);
            }
        });
    });
</script>

```

Thực hiện cho các chức năng **CRUD** tương tự cho **Category**

- *CategoriesController*

```

public class CategoriesController : Controller
{
    private readonly IProductRepository _productRepository;
    private readonly ICategoryRepository _categoryRepository;

    public CategoriesController(IProductRepository productRepository,
        ICategoryRepository categoryRepository)
    {
        _productRepository = productRepository;
        _categoryRepository = categoryRepository;
    }

    public async Task<IActionResult> Index()
    {
        var category = await _categoryRepository.GetAllAsync();
        return View(category);
    }

    public async Task<IActionResult> Display(int id)
    {

        var category = await _categoryRepository.GetByIdAsync(id);
        if (category == null)
        {
            return NotFound();
        }

        return View(category);
    }
}

```

```
public IActionResult Add()
{
    return View();
}

[HttpPost]
public async Task<IActionResult> Add( Category category)
{
    if (ModelState.IsValid)
    {
        _categoryRepository.AddAsync(category);
        return RedirectToAction(nameof(Index));
    }
    return View(category);
}

public async Task<IActionResult> Update(int id)
{
    var category = await _categoryRepository.GetByIdAsync(id);
    if (category == null)
    {
        return NotFound();
    }
    return View(category);
}

[HttpPost]
public async Task<IActionResult> Update(int id, Category category)
{
    if (id != category.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        _categoryRepository.UpdateAsync(category);
        return RedirectToAction(nameof(Index));
    }
    return View(category);
}

public async Task<IActionResult> Delete(int id)
{
    var category = await _categoryRepository.GetByIdAsync(id);
    if (category == null)
    {
        return NotFound();
    }
}
```

```
        }

        return View(category);
    }

[HttpPost, ActionName("DeleteConfirmed")]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var category = await _categoryRepository.GetByIdAsync(id);
    if (category != null)
    {
        _categoryRepository.DeleteAsync(id);
    }
    return RedirectToAction(nameof(Index));
}
```

#### Lưu ý:

- Thay thế 'YourNamespace.Models.Product` với namespace thực tế của model `Product` trong dự án của bạn.
- Đảm bảo rằng `ViewBag.Categories` được đúng cách gán giá trị trong controller trước khi gửi đến `Add.cshtml` và `Update.cshtml` .
- Thêm logic để xử lý đường dẫn hình ảnh trong `Display.cshtml` và `Update.cshtml` tùy theo cách bạn lưu trữ và truy xuất hình ảnh.

# BÀI 4: XÂY DỰNG ỨNG DỤNG WEB BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 2)

Sau khi học xong bài này, sinh viên có thể nắm được:

- Sử dụng Identity Core để thực hiện các chức năng quản lý người dùng trong ứng dụng
- Kỹ năng sử dụng Area trong ASP.NET Core MVC để phân tách và tổ chức các chức năng quản trị ra khỏi phần còn lại của ứng dụng web. Điều này giúp tăng cường bảo mật và rõ ràng trong cấu trúc ứng dụng.
- Tạo khu vực Admin để chứa các chức năng như quản lý người dùng, cài đặt hệ thống, báo cáo, và các công cụ quản trị khác.

## 4.1 Bài thực hành

- **Yêu cầu**

Tích hợp Identity Core vào ứng dụng web bán hàng ở bài thực hành LAB 2 để quản lý người dùng.

Thiết lập một Area Admin trong ứng dụng web bán hàng để tách biệt chức năng dành cho người quản trị.

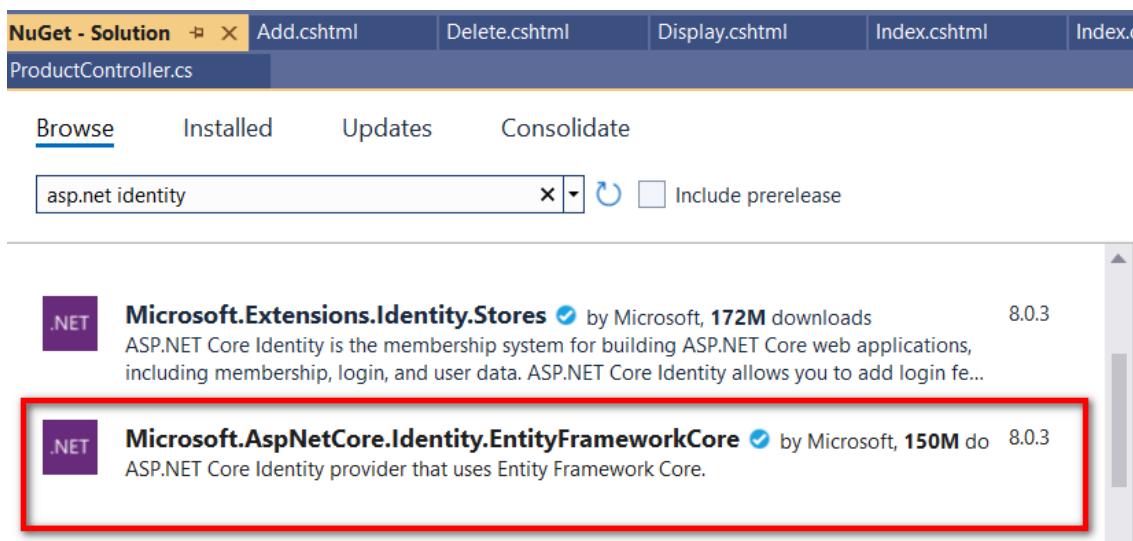
## 4.2 Hướng dẫn thực hiện

### 4.2.1 Cấu Hình ASP.NET Core Identity

- *Cài Đặt Gói NuGet Cần Thiết*

Cài đặt các gói sau bằng NuGet:

- `Microsoft.AspNetCore.Identity.EntityFrameworkCore` → phiên bản 8.0.3



Cấu hình ASP.NET Core Identity trong `ApplicationDbContext` và `Program.cs` :

- *ApplicationDbContext.cs*

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using YourNamespace.Models; // Thay thế bằng namespace thực tế của bạn

public class ApplicationDbContext : IdentityDbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }

    public DbSet<Product> Products { get; set; }
    public DbSet<Category> Categories { get; set; }
    // Các DbSet khác nếu cần
}
```

- *Program.cs*

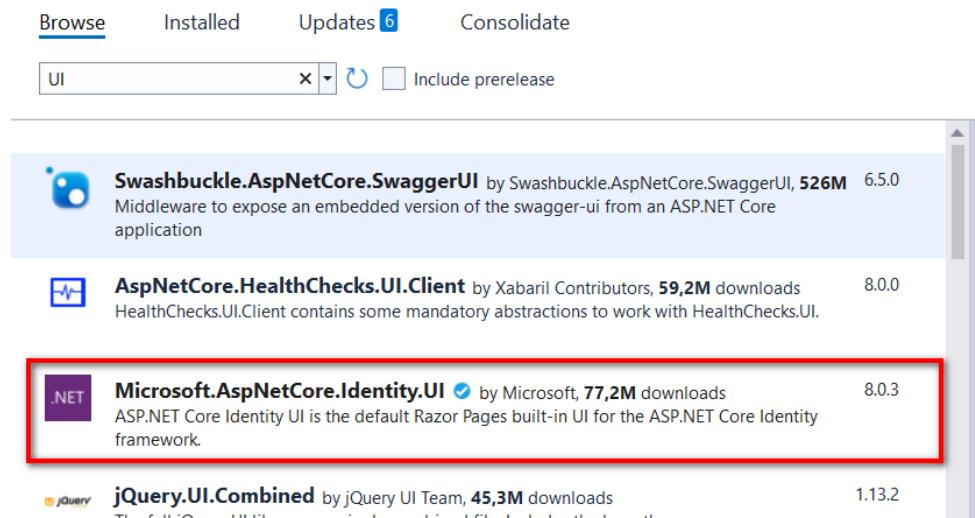
```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using WebsiteBanHang.DataAccess;
using WebsiteBanHang.Repositories;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();
```

```
builder.Services.AddDbContext<ApplicationContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddDefaultTokenProviders()
    .AddDefaultUI()
    .AddEntityFrameworkStores<ApplicationContext>();
builder.Services.AddRazorPages();
builder.Services.AddScoped<IProductRepository,
EFProductRepository>();
builder.Services.AddScoped<ICategoryRepository,
EFCategoryRepository>();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}
app.UseStaticFiles();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.MapRazorPages();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

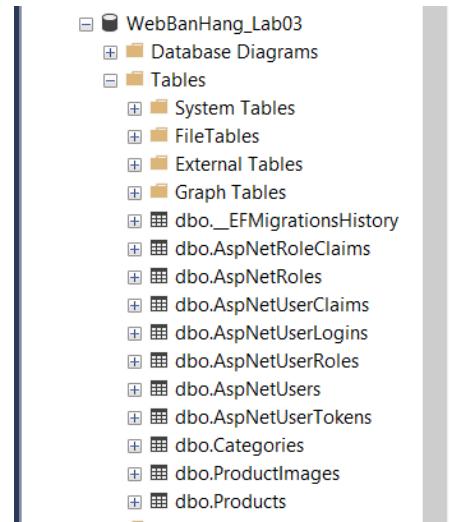
- Add thêm Nuget: Microsoft.AspNetCore.Identity.UI



- Cập nhật Database

### Add-Migration AddIdentity

### Update-Database



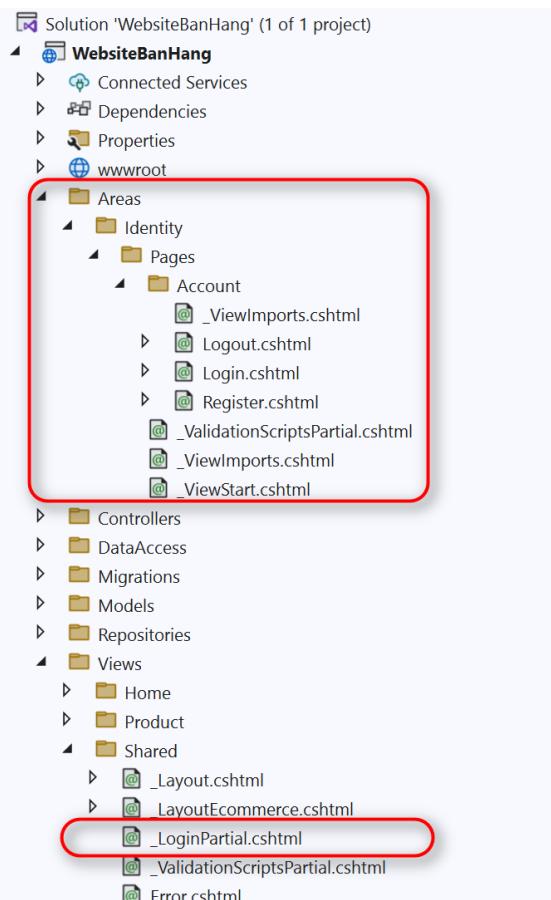
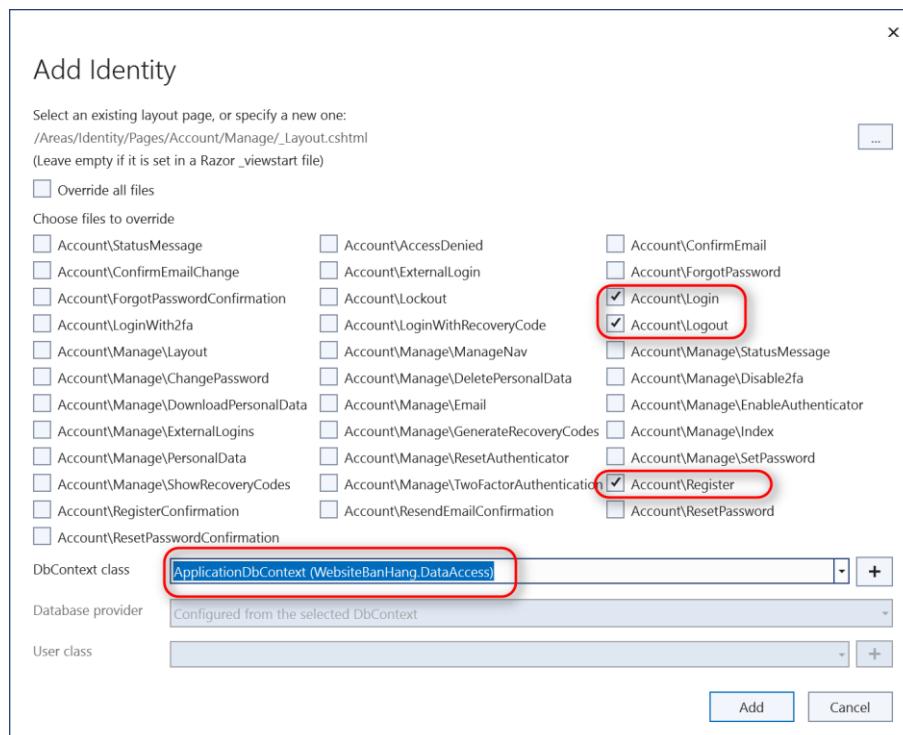
## 4.2.2 Scaffolding ASP.NET Core Identity

Sử dụng ASP.NET Core scaffolder để thêm các view đăng nhập và đăng ký:

1. Trong Visual Studio, click chuột phải vào project, chọn **Add -> New Scaffolded Item**.
2. Chọn **Identity** và click **Add**.

**3. Chọn các **pages** bạn muốn thêm, ví dụ: **Login**, **Logout** và **Register**.**

**4. Chọn **Add**.**



- Quay về *Program.cs* xoá dòng bên dưới

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<ApplicationContext>();

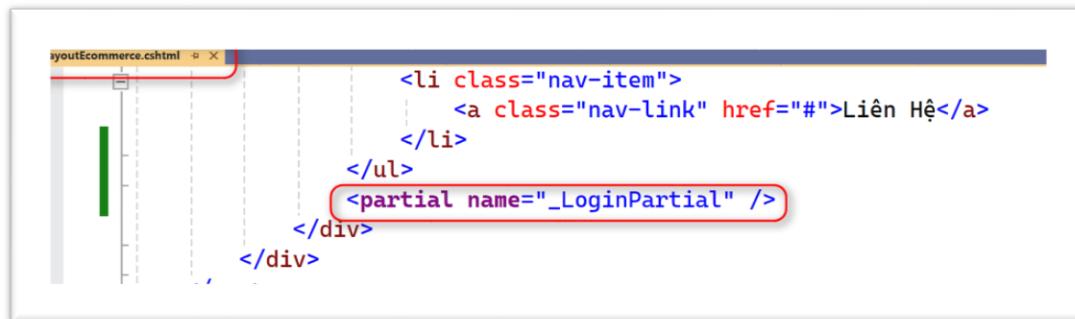
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddDefaultTokenProviders()
    .AddDefaultUI()
    .AddEntityFrameworkStores<ApplicationContext>();

builder.Services.AddRazorPages();
```

**Nhớ xoá dòng này**

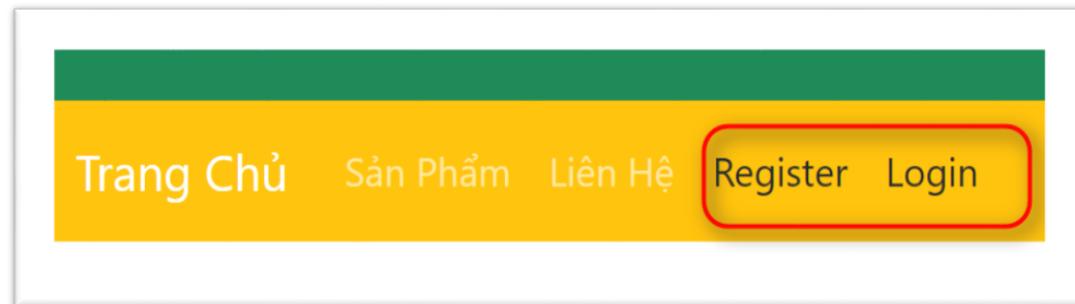
### 4.2.3 Thêm Liên Kết Đăng Ký và Đăng Nhập

- Cập nhật `'\_Layout.cshtml` để thêm các liên kết đăng ký và đăng nhập:



```
<li class="nav-item">
    <a class="nav-link" href="#">Liên Hệ</a>
</li>
</ul>
<div>
    <partial name=_LoginPartial />
</div>
</div>
```

Kết quả



Use a local account to log in.

---

Remember me?

[Log in](#)

## Register

---

[Register](#)

Use another service to register.

---

There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services](#).

Thay đổi giao diện trang đăng ký và đăng nhập cho phù hợp với trang \_Layout mặc định.

Sau khi tiến hành đăng ký, các bạn kiểm tra dữ liệu có lưu vào CSDL hay chưa??

	Id	UserName	NormalizedUserName	Email	NormalizedEmail	EmailConfirmed	PasswordHash	SecurityStamp
*	e7286464-8...	tinhocsaovi...	TINHOCSA...	tinhocsa...@gmail.com	TINHOCSA...	False	AQAAAAEA...	2K
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Thêm các thông tin bổ sung cho trang đăng ký như: **FullName, Address, Age,...vvv**

Ta thêm một số thuộc tính về thông tin người dùng ta tạo một class **ApplicationUser** trong **Models**.

- ApplicationUser.cs

```
public class ApplicationUser : IdentityUser {
    [Required]
    public string FullName { get; set; }
    public string? Address { get; set; }
    public string? Age { get; set; }
}
```

Trong ApplicationDbContext ta tiến hành thay đổi như sau:

- **ApplicationDbContext.cs**

```

ApplicationDbContext.cs  ✘ ×
WebBanHang_Lab03  WebBanHang_Lab03.Models.ApplicationDbContext  Products
1  using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
2  using Microsoft.EntityFrameworkCore;
3
4  namespace WebBanHang_Lab03.Models
5  {
6      public class ApplicationDbContext : IdentityDbContext<ApplicationUser> ①
7      {
8          public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) ②
9          {
10         }
11
12         public DbSet<Product> Products { get; set; }
13         public DbSet<Category> Categories { get; set; }
14         public DbSet<ProductImage> ProductImages { get; set; }
15     }
16 }
17

```

- *Bây giờ ta sẽ Add Migration để cập nhật các thông tin người dùng. Sử dụng câu lệnh:*
  - ❖ **Add-Migration ExtendIdentityUser**
  - ❖ *Tiếp theo ta sẽ update database bằng câu lệnh: **Update-Database***

Sau khi Add xong các thông tin bổ xung của người dùng được cập nhật thêm.

```

namespace WebBanHang_Lab03.Migrations
{
    public partial class ExtendIdentityUser : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AddColumn<string>(
                name: "Address",
                table: "AspNetUsers",
                type: "nvarchar(max)",
                nullable: true); 1

            migrationBuilder.AddColumn<string>(
                name: "Age",
                table: "AspNetUsers",
                type: "nvarchar(max)",
                nullable: true); 2

            migrationBuilder.AddColumn<string>(
                name: "FullName",
                table: "AspNetUsers",
                type: "nvarchar(max)",
                nullable: false,
                defaultValue: ""); 3
        }
    }
}

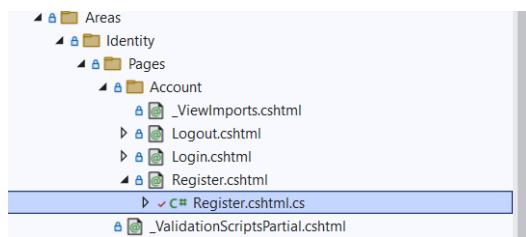
```

*Sau khi Update Database thành công -> Refresh lại CSDL*

+	dbo.AspNetUserLogins
+	dbo.AspNetUserRoles
-	dbo.AspNetUsers
	Columns
	Id (PK, nvarchar(450), not null)
	UserName (nvarchar(256), null)
	NormalizedUserName (nvarchar(256), null)
	Email (nvarchar(256), null)
	NormalizedEmail (nvarchar(256), null)
	EmailConfirmed (bit, not null)
	PasswordHash (nvarchar(max), null)
	SecurityStamp (nvarchar(max), null)
	ConcurrencyStamp (nvarchar(max), null)
	PhoneNumber (nvarchar(max), null)
	PhoneNumberConfirmed (bit, not null)
	TwoFactorEnabled (bit, not null)
	LockoutEnd (datetimeoffset(7), null)
	LockoutEnabled (bit, not null)
	AccessFailedCount (int, not null)
	Address (nvarchar(max), null) <span style="border: 2px solid red; padding: 2px;">1</span>
	Age (nvarchar(max), null) <span style="border: 2px solid red; padding: 2px;">2</span>
	FullName (nvarchar(max), not null) <span style="border: 2px solid red; padding: 2px;">3</span>

Tiếp theo ta mở class **Register.cshtml.cs** trong phần CreateUser ta sẽ đổi từ **IdentityUser** thành **ApplicationUser**

- Register.cshtml.cs



```

private ApplicationUser CreateUser() ①
{
    try
    {
        return Activator.CreateInstance<ApplicationUser>(); ②
    }
    catch
    {
        throw new InvalidOperationException($"Can't create an instance of '{nameof(ApplicationUser)}'. " +
            $"Ensure that '{nameof(ApplicationUser)}' is not an abstract class and has a parameterless constructor, or alternatively " +
            $"override the register page in /Areas/Identity/Pages/Account/Register.cshtml");
    }
}

```

- Để đăng ký tài khoản chứa trường FullName ta tiến hành:

- Tại Program.cs yêu cầu chỉnh sửa như sau:

```

1  using Microsoft.AspNetCore.Identity;
2  using Microsoft.EntityFrameworkCore;
3  using WebBanHang_Lab03.Models;
4  using WebBanHang_Lab03.Repositories;
5
6  var builder = WebApplication.CreateBuilder(args);
7
8  builder.Services.AddDbContext<ApplicationContext>(options =>
9      options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
10
11 builder.Services.AddIdentity<ApplicationUser, IdentityRole>()
12     .AddDefaultTokenProviders()
13     .AddDefaultUI()
14     .AddEntityFrameworkStores<ApplicationContext>(); ①
15
16 builder.Services.AddRazorPages(); ②
17
18 // Add services to the container.
19 builder.Services.AddControllersWithViews();
20
21 builder.Services.AddScoped<IPrductRepository, EFProductRepository>();
22 builder.Services.AddScoped<ICategoryRepository, EFCategoryRepository>(); ③
23
24
25 var app = builder.Build();
26
27 // Configure the HTTP request pipeline.
28 if (!app.Environment.IsDevelopment())
29 {
30     app.UseExceptionHandler("/Home/Error");
31 }
32 app.UseStaticFiles();
33
34 app.UseRouting();
35 app.UseAuthentication();
36 app.UseAuthorization();
37
38 app.MapControllerRoute(
39     name: "default",
40     pattern: "{controller=Home}/{action=Index}/{id?}");
41
42 app.MapRazorPages(); ④
43
44 app.Run();
45

```

- Trong **Register.cshtml.cs** thêm thuộc tính **FullName** tại **InputModel**

```

Register.cshtml.cs
WebBanHang_Lab03
WebBanHang_Lab03.Areas.Identity.Pages.Account.Register.cshtml.cs
70     /// This API supports the ASP.NET Core Identity
71     /// directly from your code. This API may change
72     /// </summary>
73     public class InputModel
74     {
75         [Required]
76         public string FullName { get; set; }
77         /// This API supports the ASP.NET Core Identity
78         /// directly from your code. This API may change
79         /// </summary>
80         [Required]
81         [EmailAddress]
82         [Display(Name = "Email")]
83         public string Email { get; set; }
84

```

- Quay lại **Register.cshtml** tạo **Form Input** cho trường **FullName**

```

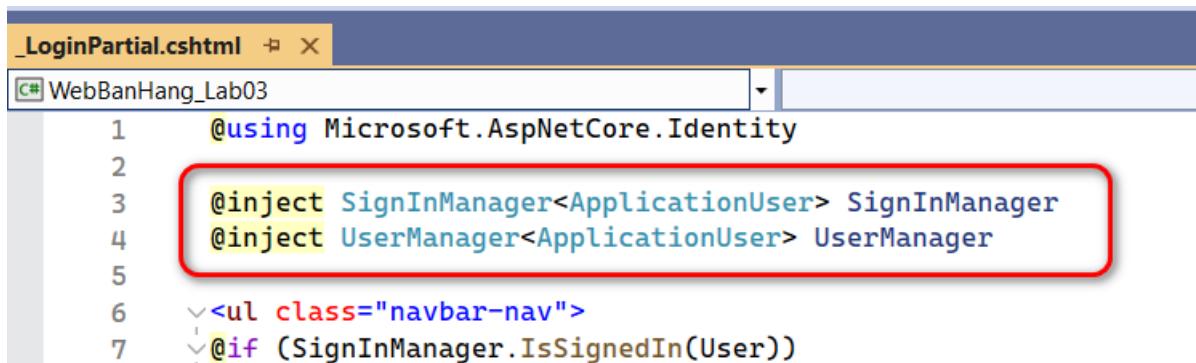
@{
    ViewData["Title"] = "Register";
}

<h1>@ViewData["Title"]</h1>

<div class="row">
    <div class="col-md-4">
        <form id="registerForm" asp-route-returnUrl="@Model.ReturnUrl" method="post">
            <h2>Create a new account.</h2>
            <hr />
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-floating">
                <input asp-for="Input.FullName" class="form-control" autocomplete="fullname" aria-required="true" />
                <label asp-for="Input.FullName"></label>
                <span asp-validation-for="Input.FullName" class="text-danger"></span>
            </div>
            <div class="form-floating">
                <input asp-for="Input.Email" class="form-control" autocomplete="username" aria-required="true" />
                <label asp-for="Input.Email"></label>
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
        </form>
    </div>
</div>

```

- Vào phần **\_LoginPartial.cshtml** thay thế **IdentityUser** → **ApplicationUser**



```

_LoginPartial.cshtml ✎ ×
C# WebBanHang_Lab03
1  @using Microsoft.AspNetCore.Identity
2
3  @inject SignInManager< ApplicationUser > SignInManager
4  @inject UserManager< ApplicationUser > UserManager
5
6  <ul class="navbar-nav">
7  @if (SignInManager.IsSignedIn(User))

```

- Thực hiện Add lại Scaffolded Item.**

Sử dụng ASP.NET Core scaffolder để thêm các view đăng nhập và đăng ký:

- Trong Visual Studio, click chuột phải vào project, chọn Add -> New Scaffolded Item.
- Chọn Identity và click Add.
- Chọn các pages bạn muốn thêm, ví dụ: Login, Logout và Register.
- Chọn Add.

**Sau khi Add lại thì sẽ sinh ra dòng bên dưới, các bạn xoá đi là được**

```

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext< ApplicationDbContext >(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddDefaultIdentity< ApplicationUser >(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores< ApplicationDbContext >();

builder.Services.AddIdentity< ApplicationUser, IdentityRole >()
    .AddDefaultTokenProviders()
    .AddDefaultUI()
    .AddEntityFrameworkStores< ApplicationDbContext >();

builder.Services.AddRazorPages();

```

**Xoá dòng này**

- Thêm lại thuộc tính **FullName** tại **InputModel** (Vì add lại bị xoá đi)

```

Register.cshtml.cs
WebBanHang_Lab03.Areas.Identity.Pages.Account.Register.cshtml.cs
...
public class InputModel
{
    [Required]
    public string FullName { get; set; }
    ...
}

```

- Vào **Register.cshtml.cs** bổ sung dưới mục **OnPostAsync**

```

0 references | phamvankhai1000, 23 hours ago | 1 author, 1 change
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl ??= Url.Content("~/");
    ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
    if (ModelState.IsValid)
    {
        var user = CreateUser();

        user.FullName = Input.FullName;
        await _userStore.SetUserNameAsync(user, Input.Email, CancellationToken.None);
        await _emailStore.SetEmailAsync(user, Input.Email, CancellationToken.None);
        var result = await _userManager.CreateAsync(user, Input.Password);
    }
}

```

- Quay lại **Register.cshtml** tạo **Form Input** cho trường **FullName**

```

@{
    ViewData["Title"] = "Register";
}

<h1>@ViewData["Title"]</h1>

<div class="row">
    <div class="col-md-4">
        <form id="registerForm" asp-route-returnUrl="@Model.ReturnUrl" method="post">
            <h2>Create a new account.</h2>
            <hr />
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-floating">
                <input asp-for="Input.FullName" class="form-control" autocomplete="fullname" aria-required="true" />
                <label asp-for="Input.FullName"></label>
                <span asp-validation-for="Input.FullName" class="text-danger"></span>
            </div>
            <div class="form-floating">
                <input asp-for="Input.Email" class="form-control" autocomplete="username" aria-required="true" />
                <label asp-for="Input.Email"></label>
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
        </form>
    </div>
</div>

```

#### 4.2.4 Phân vùng Area

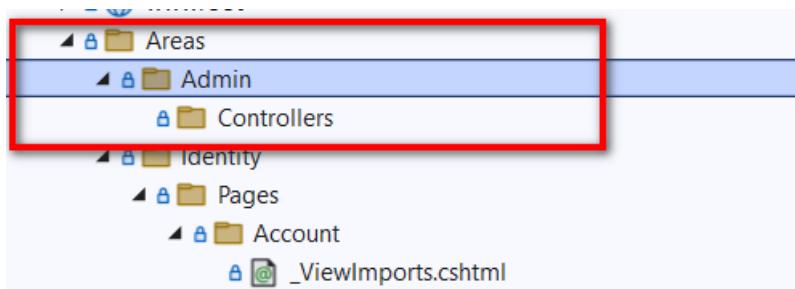
Để tạo và thiết lập một Area Admin trong ứng dụng web bán hàng với ASP.NET Core MVC, bạn có thể làm theo các bước chi tiết sau:

1. Tạo Area Admin

- Trong Visual Studio, Chuột phải vào Area, đặt tên là `Admin` , và tạo area.

2. Tạo Controllers và Views trong Admin Area:

- Trong `Areas/Admin` , tạo các thư mục `Controllers` và `Views` .



Tiếp theo ta sẽ tạo một Class SD và để tạo các vai trò cho người dùng.

- SD.cs tạo vào Folder **Models**

```
namespace YourNameSpace.Models
{
    public static class SD {
        public const string Role_Customer = "Customer";
        public const string Role_Company = "Company";
        public const string Role_Admin = "Admin";
        public const string Role_Employee = "Employee";
    }
}
```

- Ta sẽ thêm phần `private readonly RoleManager<IdentityRole> _roleManager;` vào trong public class **RegisterModel : PageModel** và `_roleManager = roleManager` vào trong public **RegisterModel** trong **Register.cshtml.cs**

```
public class RegisterModel : PageModel
{
    private readonly SignInManager< ApplicationUser > _signInManager;
    private readonly RoleManager< IdentityRole > _roleManager; 1
    private readonly UserManager< ApplicationUser > _userManager;
    private readonly IUserStore< ApplicationUser > _userStore;
    private readonly IUserEmailStore< ApplicationUser > _emailStore;
    private readonly ILogger< RegisterModel > _logger;
    private readonly IEmailSender _emailSender;

    0 references | 0 changes | 0 authors, 0 changes
    public RegisterModel(
        UserManager< ApplicationUser > userManager,
        RoleManager< IdentityRole > roleManager, 2
        IUserStore< ApplicationUser > userStore,
        SignInManager< ApplicationUser > signInManager,
        ILogger< RegisterModel > logger,
        IEmailSender emailSender)
    {
        _roleManager = roleManager; 3
        _userManager = userManager;
        _userStore = userStore;
        _emailStore = GetEmailStore();
        _signInManager = signInManager;
        _logger = logger;
        _emailSender = emailSender;
    }
}
```

Tiếp theo để gọi danh sách quyền cho người dùng ta cần thêm RoleList như sau:

```
public class InputModel
{
    [Required]
    4 references | 0 changes | 0 authors, 0 changes
    public string FullName { get; set; }
    /// </summary>
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    7 references | phamvankhai1000, 1 day ago | 1 author, 1 change
    public string Email { get; set; }

    /// </summary>
    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    4 references | phamvankhai1000, 23 hours ago | 1 author, 1 change
    public string Password { get; set; }

    /// </summary>
    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    3 references | phamvankhai1000, 23 hours ago | 1 author, 1 change
    public string ConfirmPassword { get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    public string? Role { get; set; }
    [ValidateNever]
    0 references | 0 changes | 0 authors, 0 changes
    public IEnumerable<SelectListItem> RoleList { get; set; }
}
```



Trong phần **OnGetAsync** ta sẽ tạo như sau:

```
0 references | phamvankhai1000, 1 day ago | 1 author, 1 change
public async Task OnGetAsync(string returnUrl = null)
{
    if (!_roleManager.RoleExistsAsync(SD.Role_Customer).GetAwaiter().GetResult())
    {
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Customer)).GetAwaiter().GetResult();
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Employee)).GetAwaiter().GetResult();
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Admin)).GetAwaiter().GetResult();
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Company)).GetAwaiter().GetResult();
    }
    Input = new()
    {
        RoleList = _roleManager.Roles.Select(x => x.Name).Select(i => new SelectListItem
        {
            Text = i,
            Value = i
        });
    };

    ReturnUrl = returnUrl;
    ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
}
```

Tiếp theo trong phần **OnPostAsync** ta sẽ cho xử lý nếu người dùng đăng ký thành công.

```
0 references | phamvankhai1000, 1 day ago | 1 author, 1 change
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl ??= Url.Content("~/");
    ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
    if (ModelState.IsValid)
    {
        var user = CreateUser();

        user.FullName = Input.FullName;
        await _userStore.SetUserNameAsync(user, Input.Email, CancellationToken.None);
        await _emailStore.SetEmailAsync(user, Input.Email, CancellationToken.None);
        var result = await _userManager.CreateAsync(user, Input.Password);

        if (result.Succeeded)
        {
            _logger.LogInformation("User created a new account with password.");
            if (!String.IsNullOrEmpty(Input.Role))
            {
                await _userManager.AddToRoleAsync(user, Input.Role);
            }
            else
            {
                await _userManager.AddToRoleAsync(user, SD.Role_Customer);
            }
        }
    }
}

var userId = await _userManager.GetUserIdAsync(user);
var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
```



- Để hiển thị được danh sách Role vừa thêm. Thêm đoạn bên dưới: Register.cshtml

```
<div class="form-floating">
    <input asp-for="Input.ConfirmPassword" class="form-control" autocomplete="new-password" aria-required="true" type="password" />
    <label asp-for="Input.ConfirmPassword"></label>
    <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
</div>

<div class="form-floating">
    <select asp-for="Input.Role" asp-items="@Model.Input.RoleList" class="form-control">
        <option disabled selected>-Select Role-</option>
    </select>
</div>

<button id="registerSubmit" type="submit" class="w-100 btn btn-lg btn-primary">Register</button>
</form>
```

Sau khi tiến hành khởi chạy xong. Vào CSDL kiểm tra **AspNetRoles** đã lưu vào hay chưa

QuocNam\NAMDEV...dbo.AspNetRoles				
Id	Name	Normalized Name	Concurrency Stamp	
4b-a7a8-6b0a76d168b4	Admin	ADMIN	5c107dc7-4ae5-464b-86...	
7e0ab9f2-9508-48c3...	Employee	EMPLOYEE	cad331ce-3bd9-4df3-9d...	
8313d33c-0d50-461...	Customer	CUSTOMER	9f5feecf-d2b0-4a32-bb6...	
83f84cb9-b9c4-4600...	Company	COMPANY	015acae7-9833-4d02-a3...	
NULL	NULL	NULL	NULL	

- Để phân quyền Admin trong ProductController của Admin ta sử dụng:

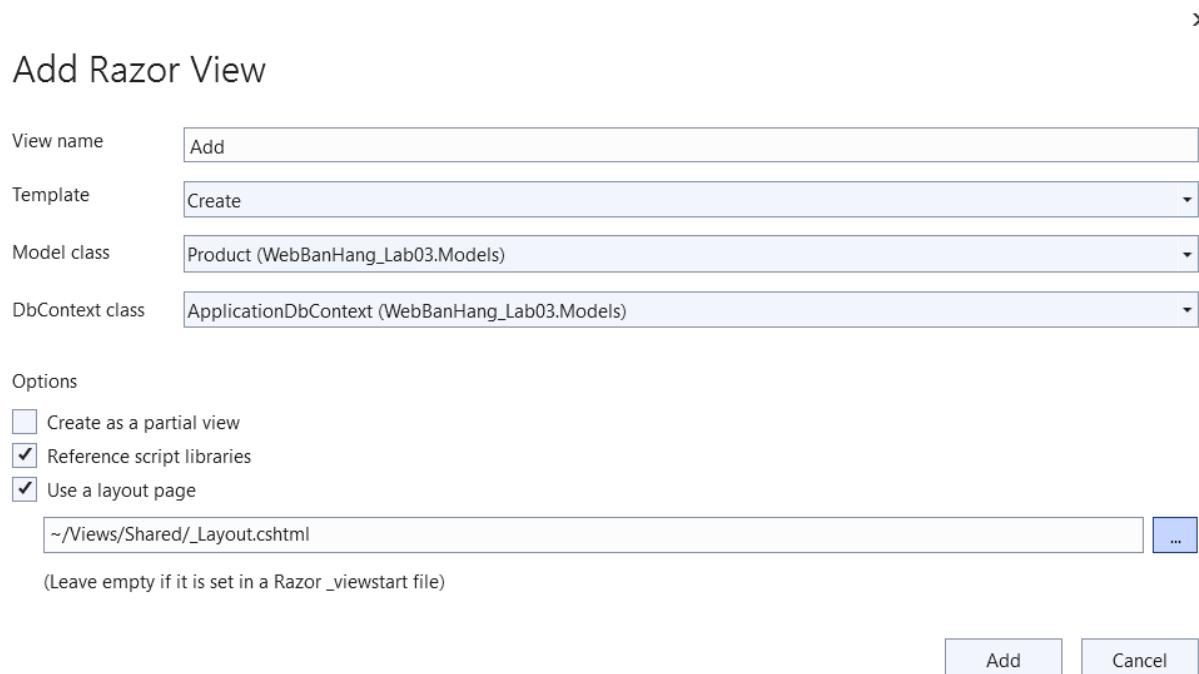
```

namespace WebBanHang_Lab03.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = SD.Role_Admin)]
    1 reference | 0 changes | 0 authors, 0 changes
    public class ProductController : Controller
    {
        private readonly IProductRepository _productRepository;
        private readonly ICategoryRepository _categoryRepository;
    }
}

```

- Tạo các view tương ứng trong `Areas/Admin/Views/Admin`.

**Ví dụ minh họa:**



### 3. Cấu hình Routing cho Admin Area:

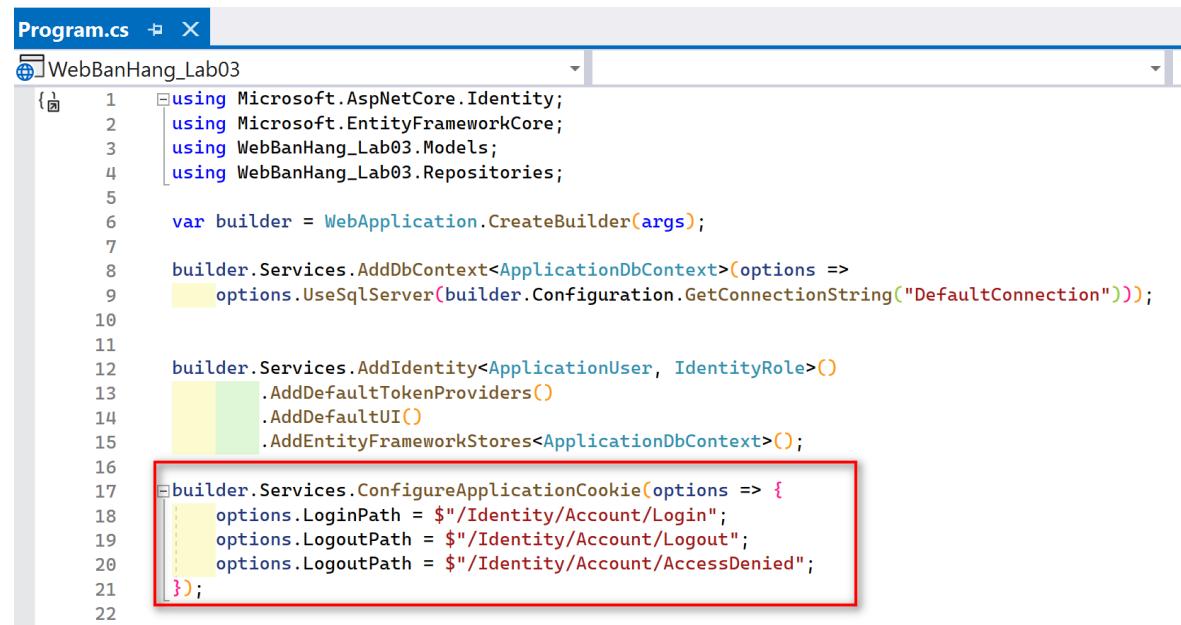
- Trong `Program.cs`, thêm cấu hình routing cho admin area:

```

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "Admin",
        pattern: "{area:exists}/{controller=Home}/{action=Index}/{id?}");
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});

```

- Hiển thị thông báo lỗi về việc hạn chế quyền truy cập và chỉ cho phép người dùng có quyền Admin:



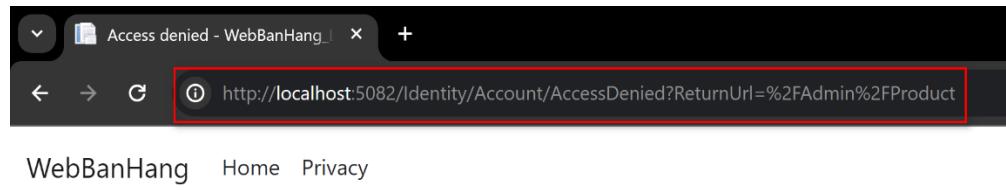
```

Program.cs  X
WebBanHang_Lab03

1  using Microsoft.AspNetCore.Identity;
2  using Microsoft.EntityFrameworkCore;
3  using WebBanHang_Lab03.Models;
4  using WebBanHang_Lab03.Repositories;
5
6  var builder = WebApplication.CreateBuilder(args);
7
8  builder.Services.AddDbContext<ApplicationContext>(options =>
9      options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
10
11
12  builder.Services.AddIdentity< ApplicationUser, IdentityRole>()
13      .AddDefaultTokenProviders()
14      .AddDefaultUI()
15      .AddEntityFrameworkStores< ApplicationContext >();
16
17  builder.Services.ConfigureApplicationCookie(options => {
18      options.LoginPath = $"/Identity/Account/Login";
19      options.LogoutPath = $"/Identity/Account/Logout";
20      options.LogoutPath = $"/Identity/Account/AccessDenied";
21  });
22

```

- Người dùng là User muốn truy cập vào đường dẫn Admin thì sẽ hiện thông báo dưới đây:



#### 4. Thêm Chức Năng Quản Lý Sản Phẩm:

- Tạo `ProductController` trong `Areas/Admin/Controllers` .
- Thêm các action `Index` , `Add` , `Edit` , `Delete` cho việc quản lý sản phẩm.
- Tạo các view tương ứng trong `Areas/Admin/Views/Product` .

#### 5. Quản Lý Đơn Hàng:

- Tương tự, tạo `OrderController` trong `Areas/Admin` và các view tương ứng.

## 4.3 Yêu cầu bổ sung

- Xây dựng lại giao diện cho các trang đăng nhập, đăng ký
- Xây dựng một trang layout mới với mẫu tùy chỉnh phù hợp với một trang web bán hàng, áp dụng trang layout mới này cho tất cả các trang con.
- Triển khai hoàn chỉnh các chức năng cho người quản trị như Thêm/Xóa/Sửa Sản phẩm, danh mục, đơn hàng, ...vvv

# BÀI 5: XÂY DỰNG ỨNG DỤNG WEBSITE BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 3)

Sau khi học xong bài này, sinh viên có thể nắm được:

- Xây dựng chức năng giỏ hàng và đặt hàng trong ứng dụng ASP.NET Core MVC

## 5.1 Mục tiêu của bài thực hành

### 5.1.1 Yêu cầu

Xây dựng chức năng giỏ hàng và đặt hàng trong ứng dụng ASP.NET Core MVC

### 5.1.2 Xây dựng giao diện hiển thị Danh sách sản phẩm người dùng

Để xây dựng trang Home hiển thị danh sách sản phẩm và giỏ hàng, bạn thực hiện các bước sau:

- Tại “\_Layout.cshtml” nhúng bootstrap icon

```
<!-- Bootstrap icons-->
<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css"
      rel="stylesheet" />
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ ViewData["Title"] - WebBanHang_Lab03</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <!-- Bootstrap icons-->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css" rel="stylesheet" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
```

- Thêm button giỏ hàng

```
<partial name="_LoginPartial" />

<form class="d-flex">
    <button class="btn btn-outline-dark" type="submit">
        <i class="bi-cart-fill me-1"></i>
        Cart
    </button>
</form>
```

- Xây dựng giao diện danh sách sản phẩm:

- Tại HomeController.cs

```
namespace WebBanHang_Lab03.Controllers
{
    1 reference | phamvankhai1000, 6 days ago | 1 author, 1 change
    public class HomeController : Controller
    {
        private readonly IProductRepository _productRepository;

        0 references | 0 changes | 0 authors, 0 changes
        public HomeController(IProductRepository productRepository)
        {
            _productRepository = productRepository;
        }

        // Hiển thị danh sách sản phẩm
        0 references | phamvankhai1000, 6 days ago | 1 author, 1 change
        public async Task<IActionResult> Index()
        {
            var products = await _productRepository.GetAllAsync();
            return View(products);
        }
    }
}
```

- Hiển thị giao diện sản phẩm: Vào Views -> Home -> Index.cshtml

```
@model IEnumerable<YourNameSpace.Models.Product>

 @{
    ViewData["Title"] = "Home Page";
}

<section class="py-2">
    <div class="container px-4 px-lg-5 mt-5">
```

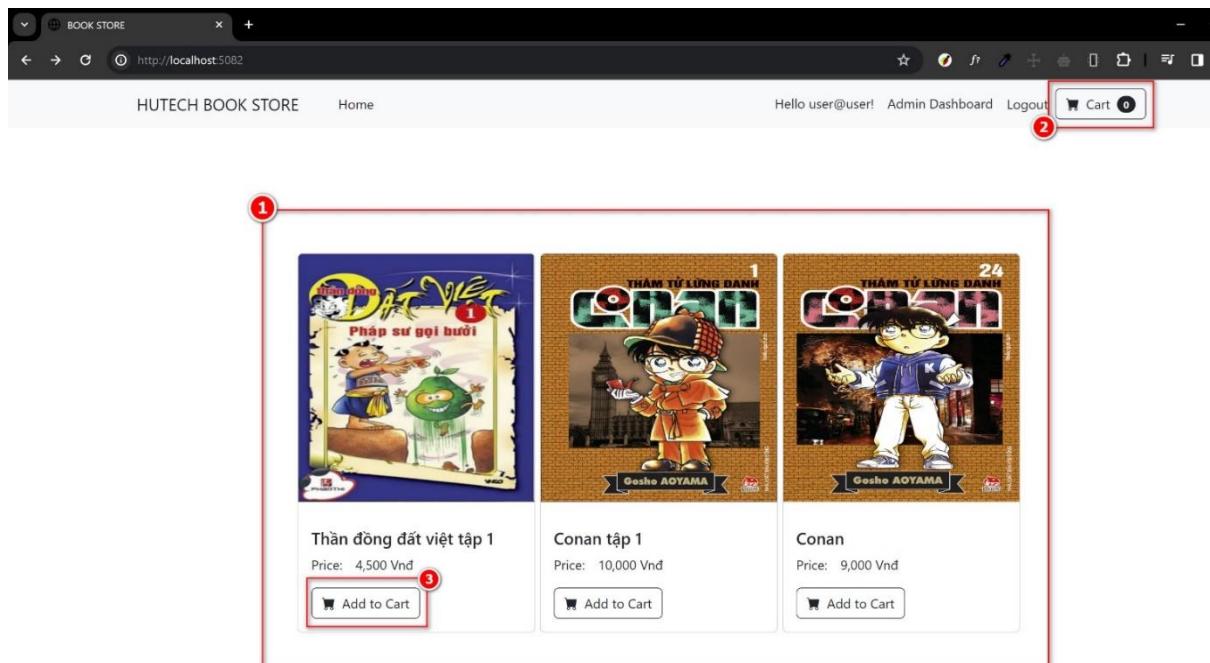
```

<div class="row gx-4 gx-lg-5 row-cols-2 row-cols-md-3 row-cols-xl-4 justify-content-center">

    @foreach (var item in Model)
    {
        <div class="col mb-5">
            <div class="card gap-3" style="width: 18rem">
                
                <div class="card-body">
                    <h5 class="card-title">@Html.DisplayFor(modelItem => item.Name)</h5>
                    <div class="d-flex">
                        <span>Price: </span>
                        <p class="mx-3">@item.Price.ToString("#,##0") Vnd</p>
                    </div>
                    <a ><button class="btn btn-outline-dark">
                        <i class="bi-cart-fill me-1"></i> Add to Cart </button>
                    </a>
                </div>
            </div>
        </div>
    </div>
</section>

```

Giao diện sau khi hoàn thành:



## 5.2 Hướng dẫn thực hiện

### 5.2.1 Code mẫu thực hiện chức năng giỏ hàng

Để xây dựng một chức năng giỏ hàng sử dụng session trong ASP.NET Core, bạn cần thực hiện các bước sau:

- *Cấu Hình Session trong 'Program.cs'*

```
var builder = WebApplication.CreateBuilder(args);

// Đặt trước AddControllersWithViews();
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Đặt trước UseRouting
app.UseSession();

// Các middleware khác...
app.UseRouting();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "Admin",
        pattern: "{area:exists}/{controller=Home}/{action=Index}/{id?}");
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
app.MapRazorPages();
app.Run();
```

- *Tạo Folder **Extensions**. Tạo file SessionExtensions.cs để làm việc với Session.*

```
using System.Text.Json;

namespace YourNameSpace.Extensions
{
    public static class SessionExtensions
    {
        public static void SetObjectAsJson(this ISession session, string key, object value)
        {
            session.SetString(key, JsonSerializer.Serialize(value));
        }

        public static T GetObjectFromJson<T>(this ISession session, string key)
        {
            var value = session.GetString(key);
            return value == null ? default :
JsonSerializer.Deserialize<T>(value);
        }
    }
}
```

- *Tạo CartItem.cs trong **Models***

```
public class CartItem
{
    public int ProductId { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int Quantity { get; set; }
}
```

- *ShoppingCart.cs*

```
public class ShoppingCart
{
    public List<CartItem> Items { get; set; } = new
List<CartItem>();

    public void AddItem(CartItem item)
    {
        var existingItem = Items.FirstOrDefault(i => i.ProductId ==
item.ProductId);
        if (existingItem != null)
        {
```

```

        existingItem.Quantity += item.Quantity;
    }
    else
    {
        Items.Add(item);
    }
}

public void RemoveItem(int productId)
{
    Items.RemoveAll(i => i.ProductId == productId);
}

// Các phương thức khác...
}

```

- Tạo `ShoppingCartController`

```

public class ShoppingCartController : Controller
{
    private readonly IProductRepository _productRepository;

    public ShoppingCartController(IProductRepository productRepository)
    {
        _productRepository = productRepository;
    }

    public async Task<IActionResult> AddToCart(int productId, int quantity)
    {
        // Giả sử bạn có phương thức lấy thông tin sản phẩm từ productId
        var product = await GetProductFromDatabase(productId);

        var cartItem = new CartItem
        {
            ProductId = productId,
            Name = product.Name,
            Price = product.Price,
            Quantity = quantity
        };

        var cart = HttpContext.Session.GetObjectFromJson<ShoppingCart>("Cart") ??
new ShoppingCart();
        cart.AddItem(cartItem);

        HttpContext.Session.SetObjectAsJson("Cart", cart);

        return RedirectToAction("Index");
    }

    public IActionResult Index()
    {

```

```

    var cart = HttpContext.Session.GetObjectFromJson<ShoppingCart>("Cart") ??
new ShoppingCart();
    return View(cart);
}
// Các actions khác...
private async Task<Product> GetProductFromDatabase(int productId)
{
    // Truy vấn cơ sở dữ liệu để lấy thông tin sản phẩm
    var product = await _productRepository.GetByIdAsync(productId);
    return product;
}

public IActionResult RemoveFromCart(int productId)
{
    var cart = HttpContext.Session.GetObjectFromJson<ShoppingCart>("Cart");

    if (cart is not null)
    {
        cart.RemoveItem(productId);

        // Lưu lại giỏ hàng vào Session sau khi đã xóa mục
        HttpContext.Session.SetObjectAsJson("Cart", cart);
    }

    return RedirectToAction("Index");
}

```

- Quay về Home -> Index.cshtml để xử lý thêm sản phẩm vào giỏ hàng

```

@{
    ViewData["Title"] = "Home Page";
    int numOfQuantity = 1; ①
}

<section class="py-2">
    <div class="container px-4 px-lg-5 mt-5">
        <div class="row gx-4 gx-lg-5 row-cols-2 row-cols-md-3 row-cols-xl-4 justify-content-center">

            @foreach (var item in Model)
            {
                <div class="col mb-5">
                    <div class="card gap-3" style="width: 18rem">
                        
                        <div class="card-body">
                            <h5 class="card-title">@Html.DisplayFor(modelItem => item.Name)</h5>
                            <div class="d-flex">
                                <span>Price: </span>
                                <p class="mx-3">@item.Price.ToString("#,##0") VNđ</p>
                            </div>
                            <a asp-controller="ShoppingCart" asp-action="AddToCart" asp-route-productId="@item.Id"
                               asp-route-quantity="@numOfQuantity" >
                                <button class="btn btn-outline-dark" type="button"><i class="bi-cart-fill me-1"></i> Add to Cart </button>
                            </a>
                        </div>
                    </div>
                </div>
            }
        </div>
    </div>

```

- *Index.cshtml (Trong Views/ShoppingCart)*

```
@model ShoppingCart

<h2>Your Cart</h2>



| Product | Quantity | Price | Total |
|---------|----------|-------|-------|
|---------|----------|-------|-------|


```

## 5.2.2 Hướng dẫn thực hiện chức năng đặt hàng

- *Model 'Order'*

```
public class Order
{
    public int Id { get; set; }
    public string UserId { get; set; }
    public DateTime OrderDate { get; set; }
    public decimal TotalPrice { get; set; }

    public string ShippingAddress { get; set; }
    public string Notes { get; set; }

    [ForeignKey("UserId")]
}
```

```
[ValidateNever]
public ApplicationUser ApplicationUser { get; set; }
public List<OrderDetail> OrderDetails { get; set; }

}
```

- *OrderDetail (Chi Tiết Đơn Hàng): Lưu thông tin chi tiết cho mỗi mặt hàng trong đơn.*

```
public class OrderDetail
{
    public int Id { get; set; }
    public int OrderId { get; set; }
    public int ProductId { get; set; }
    public int Quantity { get; set; }
    public decimal Price { get; set; }

    public Order Order { get; set; }
    public Product Product { get; set; }
}
```

- *Cập Nhật ApplicationDbContext*

Thêm DbSet cho các lớp mới vào ApplicationDbContext.

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    // ... Các DbSet hiện có ...

    public DbSet<Order> Orders { get; set; }
    public DbSet<OrderDetail> OrderDetails { get; set; }
}
```

- **Add Migration và update database**

```
Add-Migration initialOrder
```

```
Update-Database
```

- `ShoppingCartController` để Xử Lý Đặt Hàng

Trong `ShoppingCartController` , cập nhật action `Checkout` để xử lý thông tin địa chỉ giao hàng và ghi chú.

```
[Authorize]
public class ShoppingCartController : Controller
{
    private readonly IProductRepository _productRepository;
```

```
private readonly ApplicationDbContext _context;
private readonly UserManager< ApplicationUser > _userManager;
public ShoppingCartController(ApplicationDbContext context,
UserManager< ApplicationUser > userManager, IProductRepository
productRepository)
{
    _productRepository = productRepository;
    _context = context;
    _userManager = userManager;
}
public IActionResult Checkout()
{
    return View(new Order());
}

[HttpPost]
public async Task< IActionResult > Checkout(Order order)
{
    var cart =
HttpContext.Session.GetObjectFromJson< ShoppingCart > ("Cart");
    if (cart == null || !cart.Items.Any())
    {
        // Xử lý giỏ hàng trống...
        return RedirectToAction("Index");
    }

    var user = await _userManager.GetUserAsync(User);
    order.UserId = user.Id;
    order.OrderDate = DateTime.UtcNow;
    order.TotalPrice = cart.Items.Sum(i => i.Price *
i.Quantity);
    order.OrderDetails = cart.Items.Select(i => new OrderDetail
{
    ProductId = i.ProductId,
    Quantity = i.Quantity,
    Price = i.Price
}).ToList();

    _context.Orders.Add(order);
    await _context.SaveChangesAsync();

    HttpContext.Session.Remove("Cart");

    return View("OrderCompleted", order.Id); // Trang xác nhận
    hoàn thành đơn hàng
}
```

{}

- Tạo View `Checkout` `Checkout.cshtml`

Tạo một view mới để nhập thông tin đặt hàng, bao gồm địa chỉ giao hàng và ghi chú.

```
@model Order

<h2>Checkout</h2>

<form asp-action="Checkout" method="post">
    <div class="form-group">
        <label asp-for="ShippingAddress">Shipping Address</label>
        <input asp-for="ShippingAddress" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="Notes">Notes</label>
        <textarea asp-for="Notes" class="form-control"></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Place Order</button>
</form>
```

- Cập Nhật View `OrderCompleted` `OrderCompleted.cshtml`

Cập nhật view `OrderCompleted` để hiển thị thông tin xác nhận đơn hàng.

```
@model int

<h2>Order Completed</h2>
<p>Your order with ID @Model has been placed successfully.</p>
```

### 5.2.3 Yêu cầu bổ sung

- Xây dựng giao diện thân thiện cho giỏ hàng.

# BÀI 6: RESTful API

Sau khi học xong bài này, sinh viên có thể nắm được:

- Hiểu Về RESTful API: Nắm vững các khái niệm cơ bản của RESTful API, bao gồm các HTTP methods (GET, POST, PUT, DELETE) và cách tương tác với tài nguyên.
- Thiết Kế API Chuẩn RESTful: Biết cách thiết kế API theo chuẩn RESTful, bao gồm cách đặt endpoint, sử dụng HTTP methods, và quản lý tài nguyên.
- Sử Dụng ControllerBase: Hiểu cách sử dụng ControllerBase để xây dựng API Controllers và ánh xạ chúng với các endpoint.

## **6.1 Mục tiêu của bài thực hành**

### **6.1.1 Giới thiệu**

RESTful API (Representational State Transfer) là một kiến trúc thiết kế cho các dịch vụ web, dựa trên các nguyên tắc cơ bản như sự độc lập giữa client và server, tương tác thể hiện qua trạng thái biểu diễn, và sử dụng các phương thức HTTP để thực hiện các thao tác. Đây là một số nguyên tắc quan trọng của RESTful API:

- Stateless (Không Lưu Trạng Thái): Mỗi request từ client đều chứa đủ thông tin để server hiểu và xử lý. Server không lưu giữ trạng thái của client giữa các requests.
- Resource-Based (Dựa Trên Tài Nguyên): Mọi thứ trong hệ thống được xem như một tài nguyên (resource) và được xác định bởi URI (Uniform Resource Identifier).
- Representation (Biểu Diễn): Dữ liệu của tài nguyên được truyền tải giữa client và server dưới dạng biểu diễn, thường là JSON hoặc XML.
- CRUD Operations (Create, Read, Update, Delete): Sử dụng các phương thức HTTP tương ứng để thực hiện các thao tác CRUD trên tài nguyên.

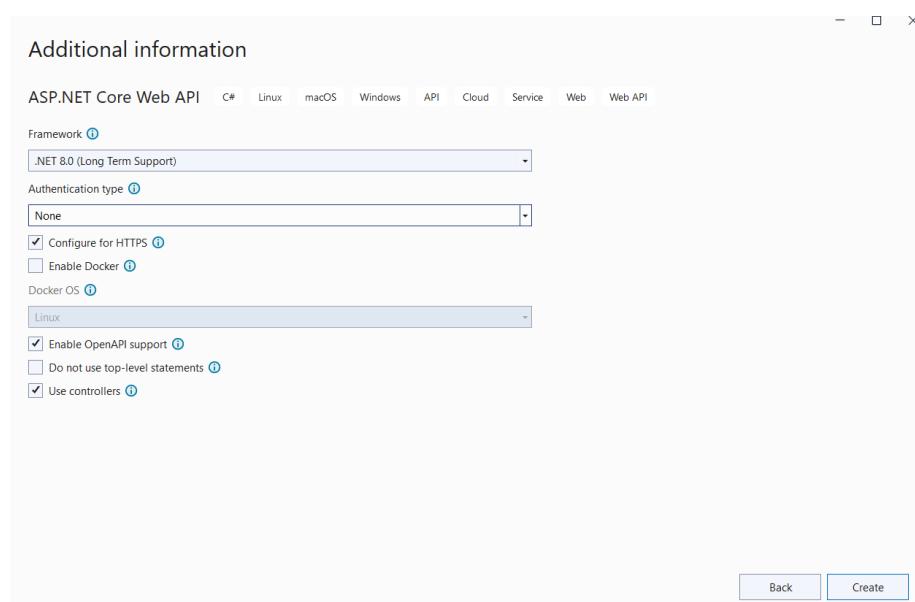
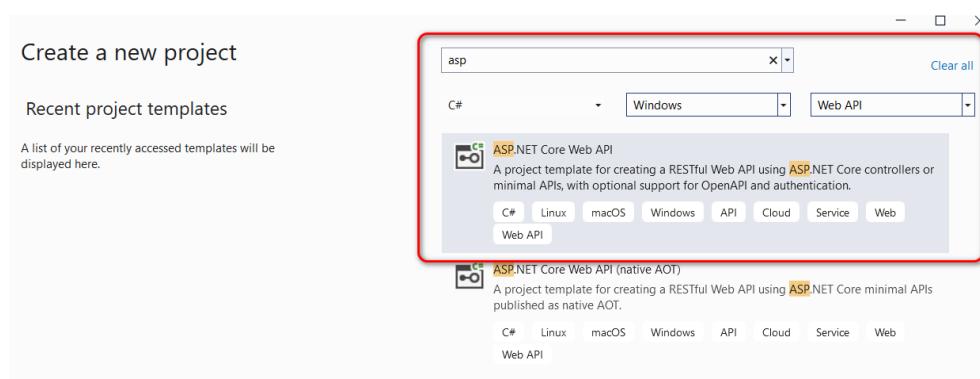
## 6.1.2 Yêu cầu

Thiết Kế RESTful API cho CRUD Operations trên Product

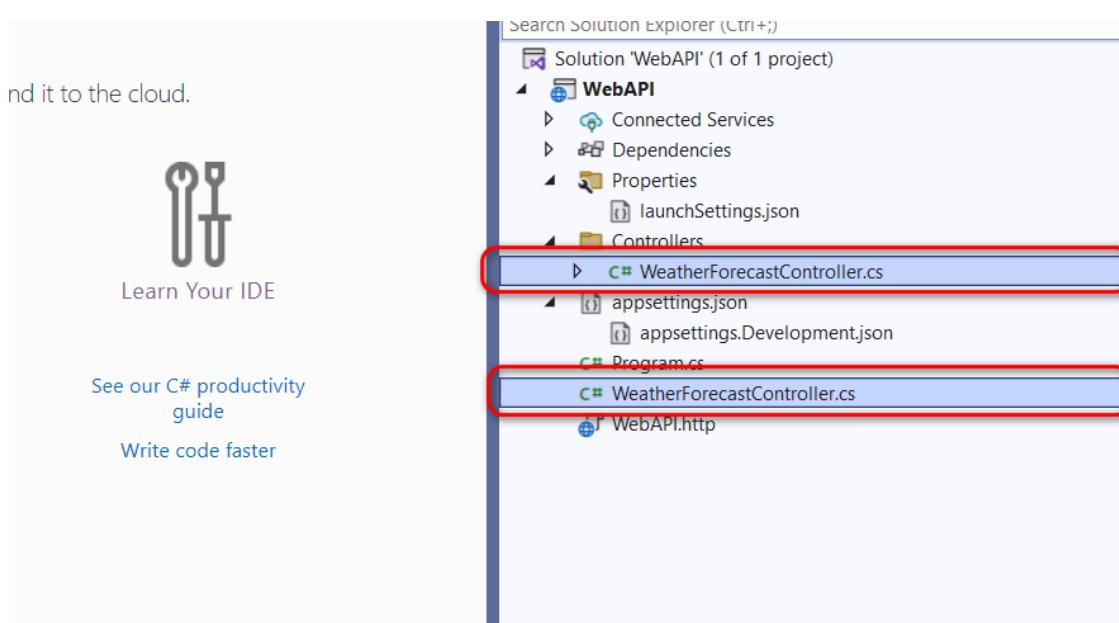
- GET /api/products: Lấy danh sách tất cả sản phẩm.
- GET /api/products/{id}: Lấy thông tin chi tiết của một sản phẩm theo ID.
- POST /api/products: Tạo một sản phẩm mới.
- PUT /api/products/{id}: Cập nhật thông tin của sản phẩm theo ID.
- DELETE /api/products/{id}: Xóa một sản phẩm theo ID.

## 6.2 Hướng dẫn thực hiện

Khởi động phần mềm **Visual Studio 2022**



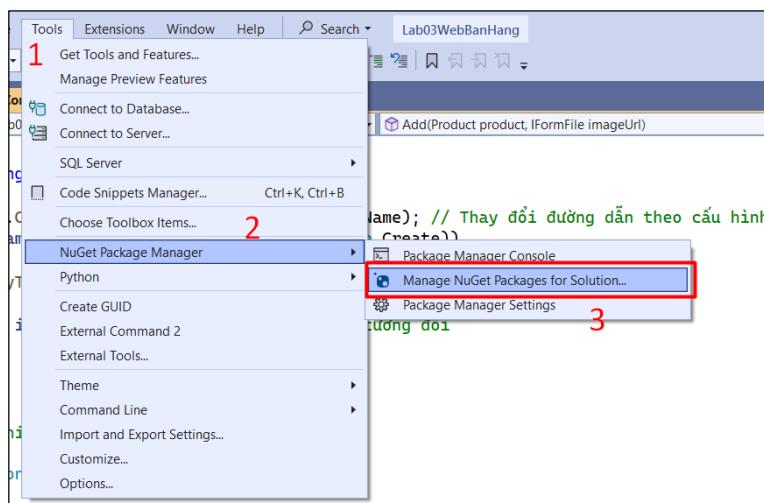
Sau khi khởi tạo xong, tái cấu trúc lại thư mục bằng cách xoá 2 file bên dưới:



- *Cài Đặt Gói NuGet Cần Thiết*

Cài đặt các gói sau bằng NuGet:

- `Microsoft.EntityFrameworkCore` → phiên bản 8.0.3
- `Microsoft.EntityFrameworkCore.SqlServer` → phiên bản 8.0.3 (hoặc provider cơ sở dữ liệu khác)
- `Microsoft.EntityFrameworkCore.Tools` → phiên bản 8.0.3

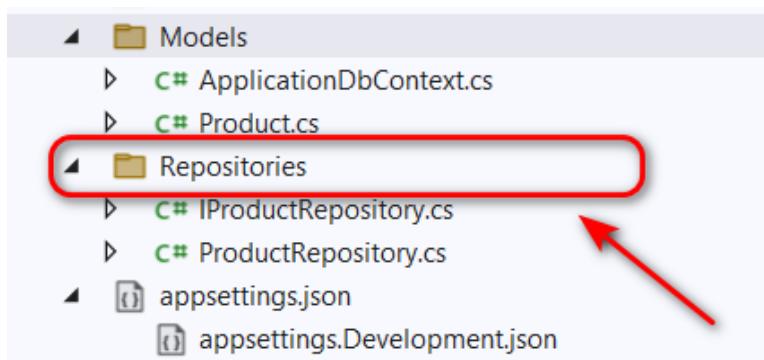


Dưới đây là một ví dụ đơn giản về cách xây dựng một **RESTful API** cho **Product** trong **ASP.NET Core**, sử dụng **Controller và Repository Pattern**.

- Product.cs (Model)

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public string Description { get; set; }
    // Other properties
}
```

- Tiến hành tạo thư mục **Repositories**:



- IProductRepository.cs

```
public interface IProductRepository
{
    Task<IEnumerable<Product>> GetProductsAsync();
    Task<Product> GetProductByIdAsync(int id);
    Task AddProductAsync(Product product);
    Task UpdateProductAsync(Product product);
    Task DeleteProductAsync(int id);
}
```

- Cấu Hình Entity Framework Core

Tạo một lớp `ApplicationDbContext` trong thư mục Models và cấu hình:

```
using Microsoft.EntityFrameworkCore;

public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options) : base(options)
    {
    }
    public DbSet<Product> Products { get; set; }
}
```

- Cấu hình EF Core trong file `Program.cs`:

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Các cấu hình khác
```

- Cấu hình connection string trong `appsettings.json`.
- Xem tên Server name trong SQL Server

```
"ConnectionStrings": {
  "DefaultConnection": 
    "Server=Servername;Database=DbName;Trusted_Connection=True;TrustServerCertificate=True"
},
```

Add-Migration Initial

Update-Database

- ProductRepository.cs

```
public class ProductRepository : IProductRepository
{
    private readonly ApplicationContext _context;

    public ProductRepository(ApplicationContext context)
    {
        _context = context;
    }

    public async Task<IEnumerable<Product>> GetProductsAsync()
    {
        return await _context.Products.ToListAsync();
    }

    public async Task<Product> GetProductByIdAsync(int id)
    {
        return await _context.Products.FindAsync(id);
    }

    public async Task AddProductAsync(Product product)
```

```

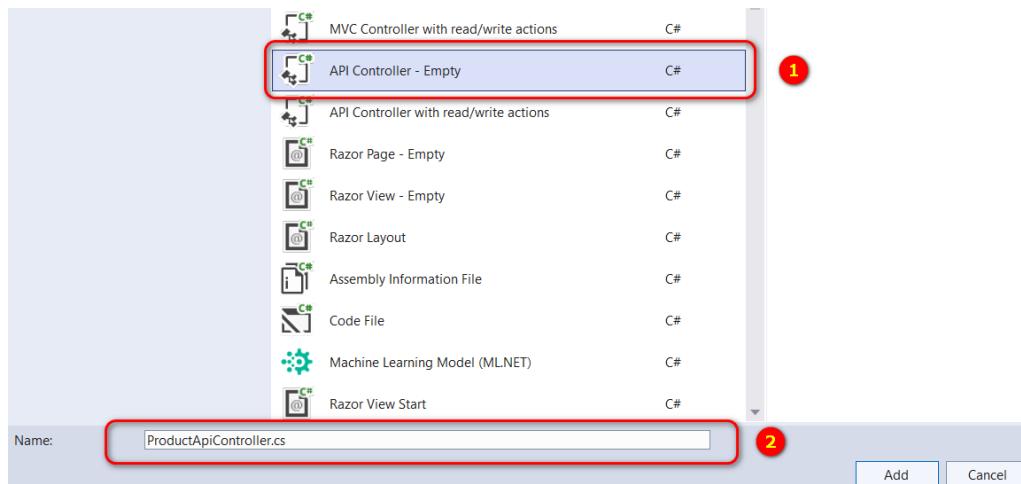
    {
        _context.Products.Add(product);
        await _context.SaveChangesAsync();
    }

    public async Task UpdateProductAsync(Product product)
    {
        _context.Entry(product).State = EntityState.Modified;
        await _context.SaveChangesAsync();
    }

    public async Task DeleteProductAsync(int id)
    {
        var product = await _context.Products.FindAsync(id);
        if (product != null)
        {
            _context.Products.Remove(product);
            await _context.SaveChangesAsync();
        }
    }
}

```

- Tiến hành khởi tạo “ProductApiController.cs” trong Controller.



```

[ApiController]
[Route("api/products")]
public class ProductApiController : ControllerBase
{
    private readonly IProductRepository _productRepository;

    public ProductApiController(IProductRepository productRepository)
    {
        _productRepository = productRepository;
    }

    [HttpGet]

```

```
public async Task<IActionResult> GetProducts()
{
    try
    {
        var products = await _productRepository.GetProductsAsync();
        return Ok(products);
    }
    catch (Exception ex)
    {
        // Handle exception
        return StatusCode(500, "Internal server error");
    }
}

[HttpGet("{id}")]
public async Task<IActionResult> GetProductById(int id)
{
    try
    {
        var product = await _productRepository.GetProductByIdAsync(id);
        if (product == null)
            return NotFound();

        return Ok(product);
    }
    catch (Exception ex)
    {
        // Handle exception
        return StatusCode(500, "Internal server error");
    }
}

[HttpPost]
public async Task<IActionResult> AddProduct([FromBody] Product product)
{
    try
    {
        await _productRepository.AddProductAsync(product);
        return CreatedAtAction(nameof(GetProductById), new { id =
product.Id }, product);
    }
    catch (Exception ex)
    {
        // Handle exception
        return StatusCode(500, "Internal server error");
    }
}

[HttpPut("{id}")]
public async Task<IActionResult> UpdateProduct(int id, [FromBody]
Product product)
```

```

{
    try
    {
        if (id != product.Id)
            return BadRequest();

        await _productRepository.UpdateProductAsync(product);
        return NoContent();
    }
    catch (Exception ex)
    {
        // Handle exception
        return StatusCode(500, "Internal server error");
    }
}

[HttpDelete("{id}")]
public async Task<IActionResult> DeleteProduct(int id)
{
    try
    {
        await _productRepository.DeleteProductAsync(id);
        return NoContent();
    }
    catch (Exception ex)
    {
        // Handle exception
        return StatusCode(500, "Internal server error");
    }
}
}

```

- Quay về *Program.cs* bổ sung thêm

```

using Microsoft.EntityFrameworkCore;
using WebAPI.Models;
using WebAPI.Repositories;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
}

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddScoped<IProductRepository, ProductRepository>();

```

```
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddCors(options =>
{
    options.AddPolicy(name: "MyAllowOrigins", policy =>
    {
        //Thay bằng địa chỉ localhost khi khởi chạy bên frontend (VSCode)
        policy.WithOrigins("http://127.0.0.1:5500", "http://localhost:5500")
            .AllowAnyHeader()
            .AllowAnyMethod();
    });
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

//Đặt trên UseAuthorization
app.UseCors("MyAllowOrigins");

app.UseAuthorization();

app.MapControllers();

app.Run();
```

- Giao diện sau khi thực hiện xong

The screenshot shows the Swagger UI interface for a WebAPI. At the top, it displays the URL `localhost:5030/swagger/index.html`. Below the header, the Swagger logo is shown with the text "Supported by SMARTBEAR". The main title is "WebAPI 1.0 OAS3" with the URL `http://localhost:5030/swagger/v1/swagger.json` underneath. The section title is "ProductApi". Below the title, there are five API endpoints listed: "GET /api/ProductApi" (blue button), "POST /api/ProductApi" (green button), "GET /api/ProductApi/{id}" (blue button), "PUT /api/ProductApi/{id}" (orange button), and "DELETE /api/ProductApi/{id}" (red button).

- Thêm dữ liệu vào Database

The screenshot shows the configuration for the "POST /api/ProductApi" endpoint. At the top, there is a "Parameters" section which says "No parameters". To the right, there is a "Try it out" button with a red circle containing the number "2". Below this, there is a "Request body" section with a dropdown menu set to "application/json". A JSON object is displayed in the body editor:

```
{
  "id": 0,
  "name": "Hoa Hồng",
  "price": 1000,
  "description": "Hoa Hồng nhập từ Đà Lạt"
}
```

A red arrow points to the "id" field with the text "ID tự tăng". At the bottom, there is a blue "Execute" button.

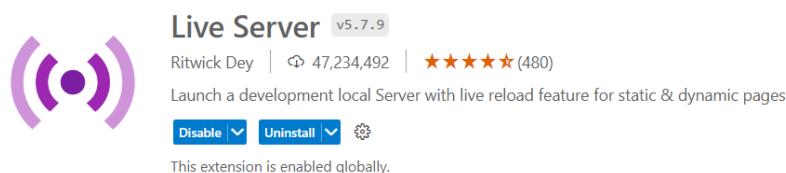
Để xây dựng front end để gọi các API của `ProductApiController`, bạn có thể sử dụng JavaScript hoặc một framework JavaScript như Angular, React, hoặc Vue.js.

- Các em có thể tham khảo Source Code theo link sau:

<https://drive.google.com/file/d/1xqZSMjcqh1xuCKw4XgHzqrFy1C355Q52/view?usp=sharing>

- **Mở VSCode -> Open Folder Source Code**

**(Tải LiveServer để chạy localhost)**



- Thực hiện lấy danh sách sản phẩm từ Database:

The image contains two screenshots of the Postman application. The top screenshot shows the 'ProductApi' collection with a single GET request to the endpoint '/api/ProductApi'. The request has a red circled '1' next to its URL. The parameters section shows 'No parameters'. The bottom screenshot shows the same request being executed, with a red circled '3' above the progress bar. The 'Cancel' button is visible at the top right of the execution area.

Sau khi Execute xong sẽ hiển thị đường dẫn để lấy danh sách sản phẩm:

Curl

```
curl -X 'GET' \
  'http://localhost:5030/api/ProductApi' \
  -H 'accept: */*'
```

Request URL

<http://localhost:5030/api/ProductApi>

Copy

Server response

- Dán đường dẫn localhost của Server vào **main.js** trong folder **js** sau khi mở project trong VScode và thực hiện thêm và lấy danh sách sản phẩm.

```
JS main.js • 1
QLS (done) > js > JS main.js > handleResponse
14
15 // Fetch products from the server and display them
16 function fetchProducts() {
17   const apiUrl = `http://localhost:5030/api/ProductApi`; 2
18   fetch(apiUrl)
19     .then(handleResponse)
20     .then(data => displayProducts(data))
21     .catch(error => console.error('Fetch error:', error.message));
22 }
23
```

- Thêm và lấy danh sách Sản Phẩm (POST Create Product):

```
document.addEventListener('DOMContentLoaded', function () {
  fetchProducts();
  document.getElementById('btnAdd').addEventListener('click', addProduct);
});

function fetchProducts() {
  const apiUrl = 'http://localhost:5030/api/ProductApi';
  fetch(apiUrl)
    .then(handleResponse)
    .then(data => displayProducts(data))
    .catch(error => console.error('Fetch error:', error.message));
}

// Handle fetch response, check for error, and parse JSON
function handleResponse(response) {
```

```
        if (!response.ok) throw new Error('Network response was not
ok');
        return response.json();
}

// Display products in the HTML table
function displayProducts(products) {
    const bookList = document.getElementById('bookList');
    bookList.innerHTML = ''; // Clear existing products
    products.forEach(product => {
        bookList.innerHTML += createProductRow(product);
    });
}

// Create HTML table row for a product
function createProductRow(product) {
    return `
        <tr>
            <td>${product.id}</td>
            <td>${product.name}</td>
            <td>${product.price}</td>
            <td>${product.description}</td>
            <td>
                <button class="btn btn-danger delete-btn" data-
id="${product.id}">Delete</button>
                <button class="btn btn-warning edit-btn" data-
id="${product.id}">Edit</button>
                <button class="btn btn-primary view-btn" data-
id="${product.id}">View</button>
            </td>
        </tr>
    `;
}

// Add a new product
function addProduct() {
    const productData = {
        name: document.getElementById('bookName').value,
        price: document.getElementById('price').value,
        description: document.getElementById('description').value,
    };

    fetch('http://localhost:5030/api/ProductApi', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(productData),
    })
}
```

```

        .then(handleResponse)
        .then(data => {
            console.log('Product added:', data);
            fetchProducts(); // Refresh the product list
        })
        .catch(error => console.error('Error:', error));
    }
}

```

**(LƯU Ý PHẢI CHẠY SERVER LÊN TRƯỚC MỚI CHẠY ĐƯỢC GIAO DIỆN  
VS CODE)**

- Kết quả thực hiện:

Thông tin sách

Mô tả

Eg. Sách quá hay!

+ Thêm Sản Phẩm Cập nhật Reset

Danh sách

ID	Tên sách	Giá	Mô tả	
4	Hoa Hồng	1000	Hoa Hồng nhập từ Đà Lạt	<span style="background-color: red; color: white; border: 1px solid red; padding: 2px 5px;">Delete</span> <span style="background-color: yellow; border: 1px solid yellow; padding: 2px 5px;">Edit</span> <span style="background-color: blue; color: white; border: 1px solid blue; padding: 2px 5px;">View</span>

- Lấy Thông Tin Chi Tiết Sản Phẩm (GET Product by ID):

```

// Thay {id} bằng ID cụ thể của sản phẩm
const productId = 1;

fetch(`https://your-api-url/api/products/${productId}`)
    .then(response => response.json())
    .then(product => {
        // Xử lý thông tin chi tiết sản phẩm
        console.log(product);
    })
    .catch(error => console.error('Error:', error));

```

- Cập Nhật Thông Tin Sản Phẩm (PUT Update Product):

```

// Thay {id} và cập nhật thông tin sản phẩm
const productIdToUpdate = 1;

const updatedProduct = {

```

```

id: productIdToUpdate,
name: 'Updated Product',
price: 150,
description: 'An updated product',
// Thêm các thông tin khác
};

fetch(`https://your-api-url/api/products/${productIdToUpdate}`, {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify(updatedProduct),
})
.then(response => {
  if (response.status === 204) {
    console.log('Product updated successfully.');
  } else {
    console.error('Failed to update product.');
  }
})
.catch(error => console.error('Error:', error));

```

- Xóa Sản Phẩm (DELETE Product):

```

// Thay {id} bằng ID cụ thể của sản phẩm cần xóa
const productIdToDelete = 1;

fetch(`https://your-api-url/api/products/${productIdToDelete}`, {
  method: 'DELETE',
})
.then(response => {
  if (response.status === 204) {
    console.log('Product deleted successfully.');
  } else {
    console.error('Failed to delete product.');
  }
})
.catch(error => console.error('Error:', error));

```

## 6.3 Yêu cầu bổ sung

Hoàn thiện giao diện cho tất cả các trang Thêm/Xóa/Sửa/Đọc với API của Product ở phần trên.

# TÀI LIỆU THAM KHẢO

1. Bài giảng Lập trình C# trên Windows, ThS. Nguyễn Hà Giang, 2010.
2. C# and .NET programing, msdn.microsoft.com, 2012.
3. Pro C# 2005 and the .NET 2.0 Platform, Andrew Troelsen, Apress, 2005.
4. C# 2.0 Practical Guide for Programmers, Michel de Champlain, Brian G. Patrick, Morgan Kaufmann publishers. 2005.
5. Windows Forms Programming with C#, Erik Brown, Manning Publications, 2008.
6. Microsoft Visual C# 2010 Step by Step, Microsoft Press, 2010.
7. Windows Forms 2.0 Programming, Chris Sells, Michael Weinhardt, Additon Wesley Professional, 2003.
8. Teach yourself .NET Windows Forms in 21 Days, Chris Payne, SAMS, 2003.
9. Source code tham khảo ở <http://www.wrox.com>.
10. Các topic lập trình ở [www.codeguru.com](http://www.codeguru.com), [www.codeproject.com](http://www.codeproject.com).
11. <https://dotnettutorials.net/>