

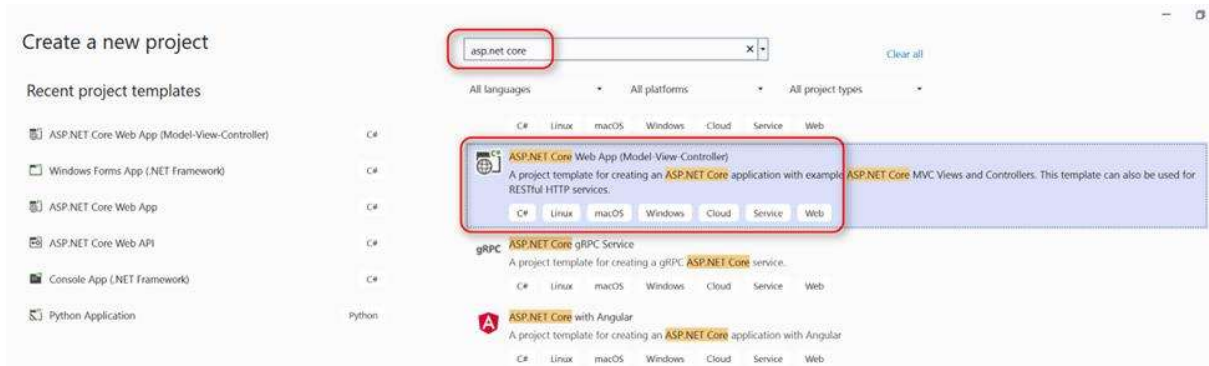
BÀI 2: XÂY DỰNG ỨNG DỤNG WEBSITE CƠ BẢN VỚI ASP.NET CORE MVC

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

Configure your new project

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows

Project name

WebsiteBanHang

Location

C:\Users\AnhNguyen\Documents\

Solution name ⓘ

WebsiteBanHang

☐ Place solution and project in the same directory

Project will be created in "C:\Users\AnhNguyen\Documents\WebsiteBanHang\WebsiteBanHang\"

Hình 2.2 Đặt tên cho Project

Additional information

ASP.NET Core Web App (Model-View-Controller)

C#

Linux

macOS

Framework ⓘ

.NET 6.0 (Long Term Support)

Authentication type ⓘ

None

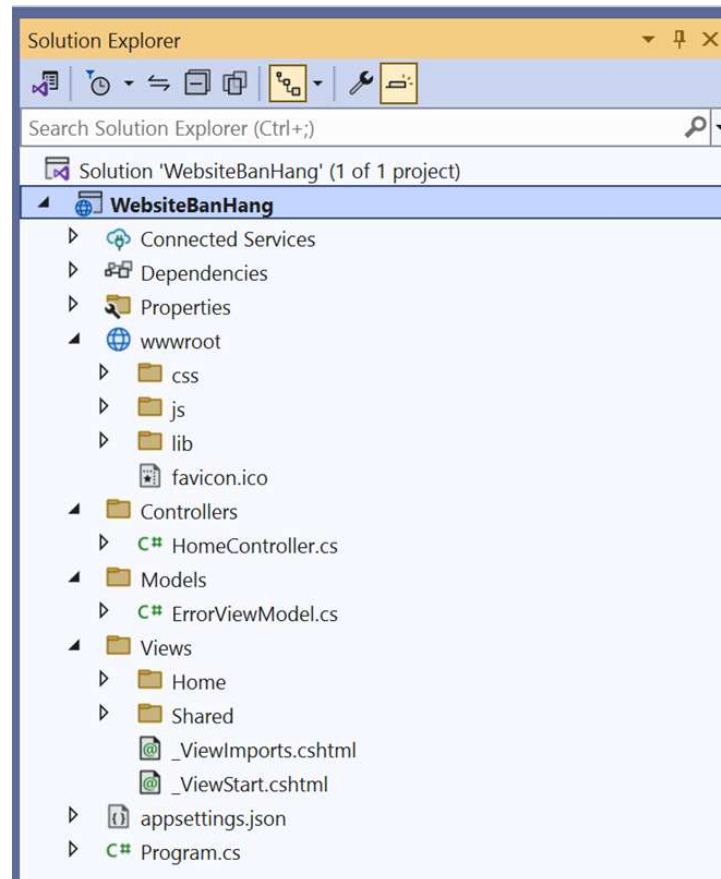
☐ Configure for HTTPS ⓘ☐ Enable Docker ⓘ

Docker OS ⓘ

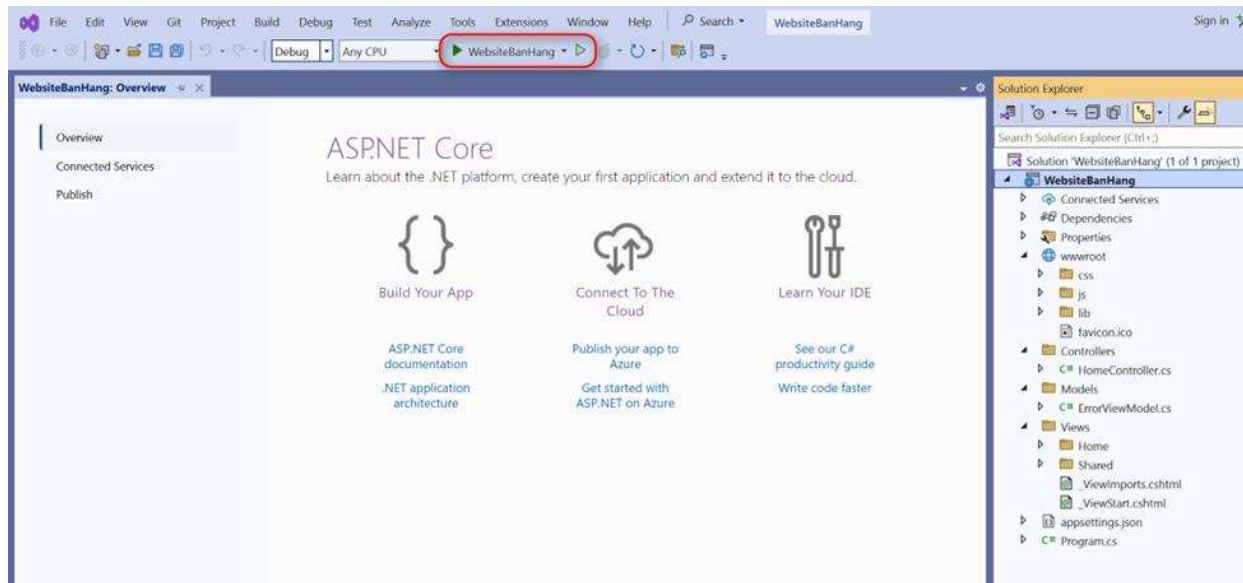
Linux

☐ Do not use top-level statements ⓘ

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 bố cục mặc định cho ứng dụng ASP.NET Core bao gồm các thành phần dùng chung của tất cả các trang con.

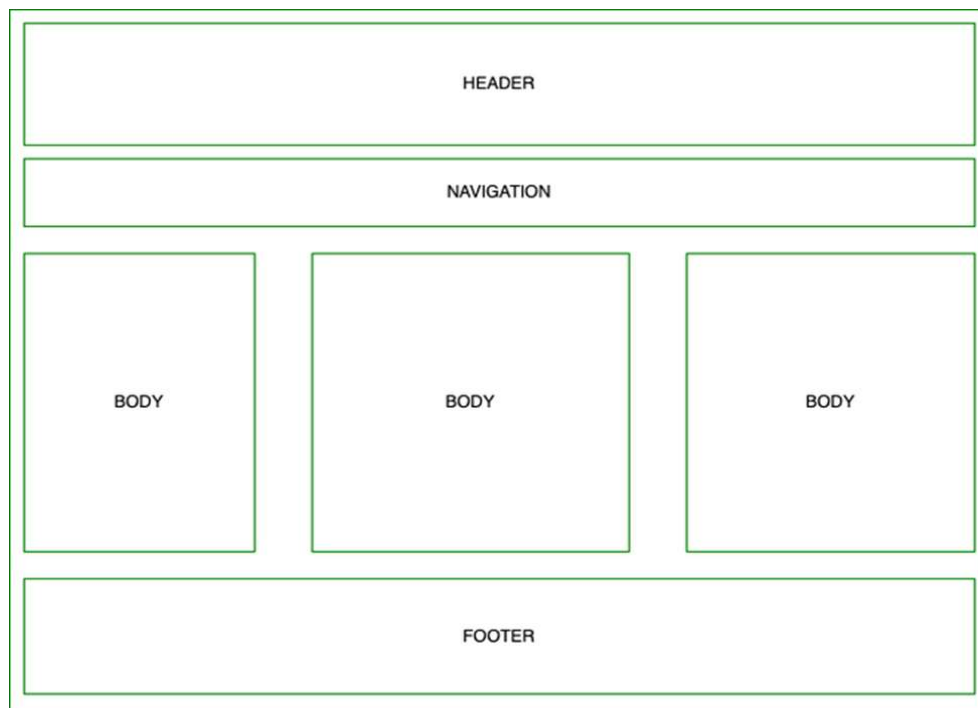
8. appsettings.json:

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:

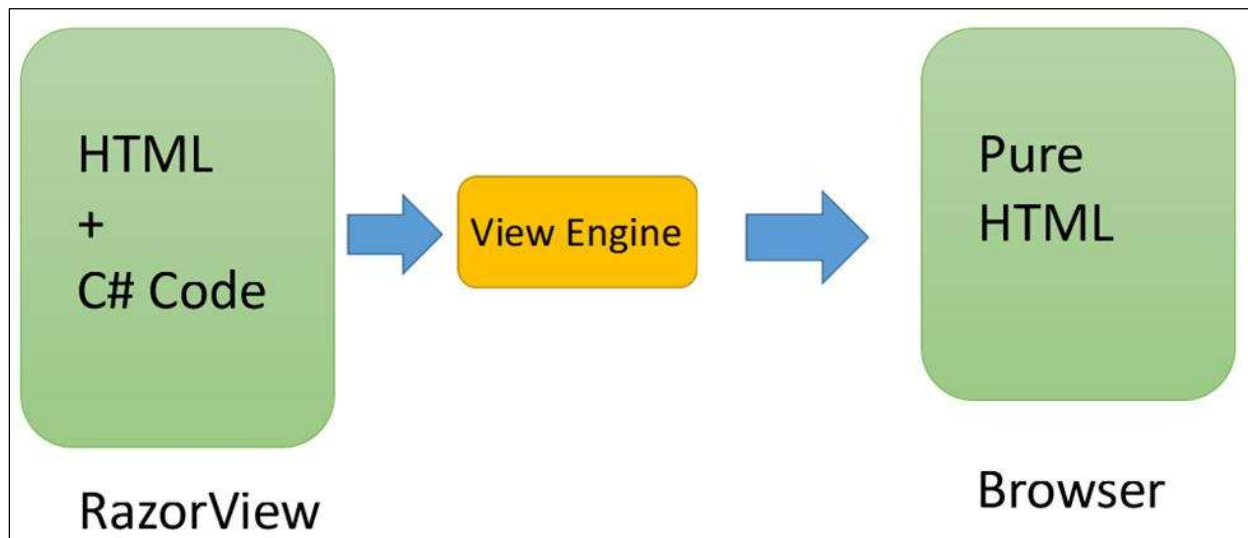
Lớp Program chứa phương thức Main. Phương thức Main chịu trách nhiệm thiết lập máy chủ web, cấu hình các dịch vụ, cấu hình các Thành phần Middleware và khởi động ứng dụng để ứng dụng có thể lắng nghe các yêu cầu từ client.

2.2.1 __Layout



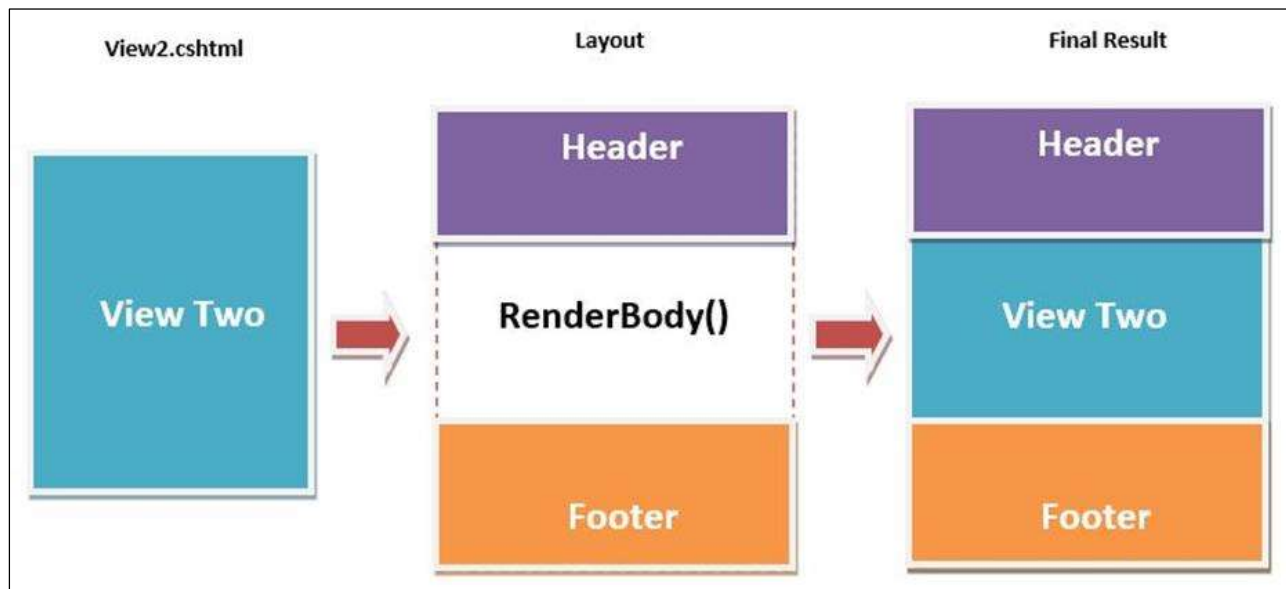
Hình 2.6 Cấu trúc của __Layout

2.2.2 Razor View Engine



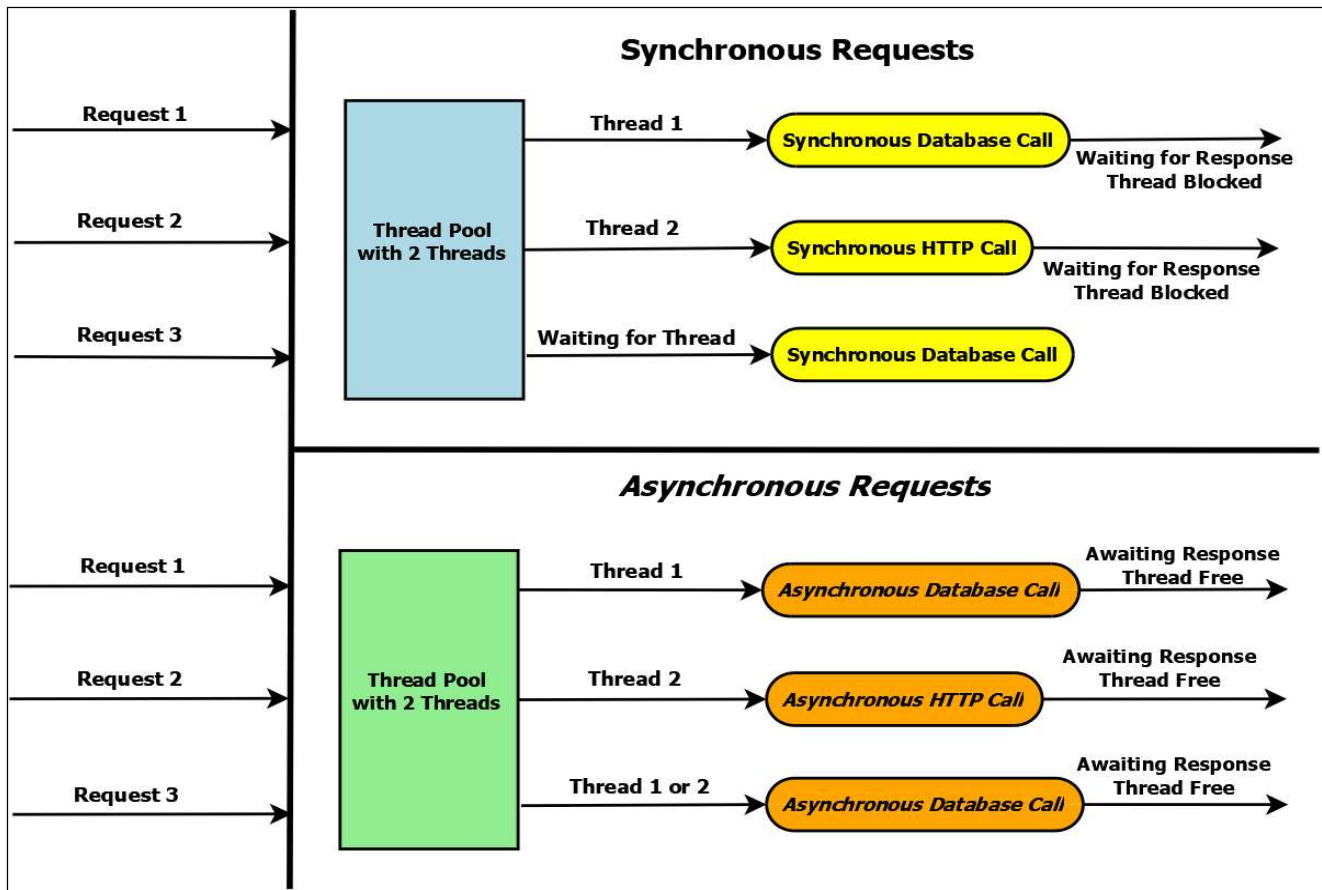
Hình 2.7 Mô hình code Razor View Engine

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; }
}
```

- *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("Delete")]
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();
```

- *AddProduct.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>
    <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>
```



```
        </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>
```

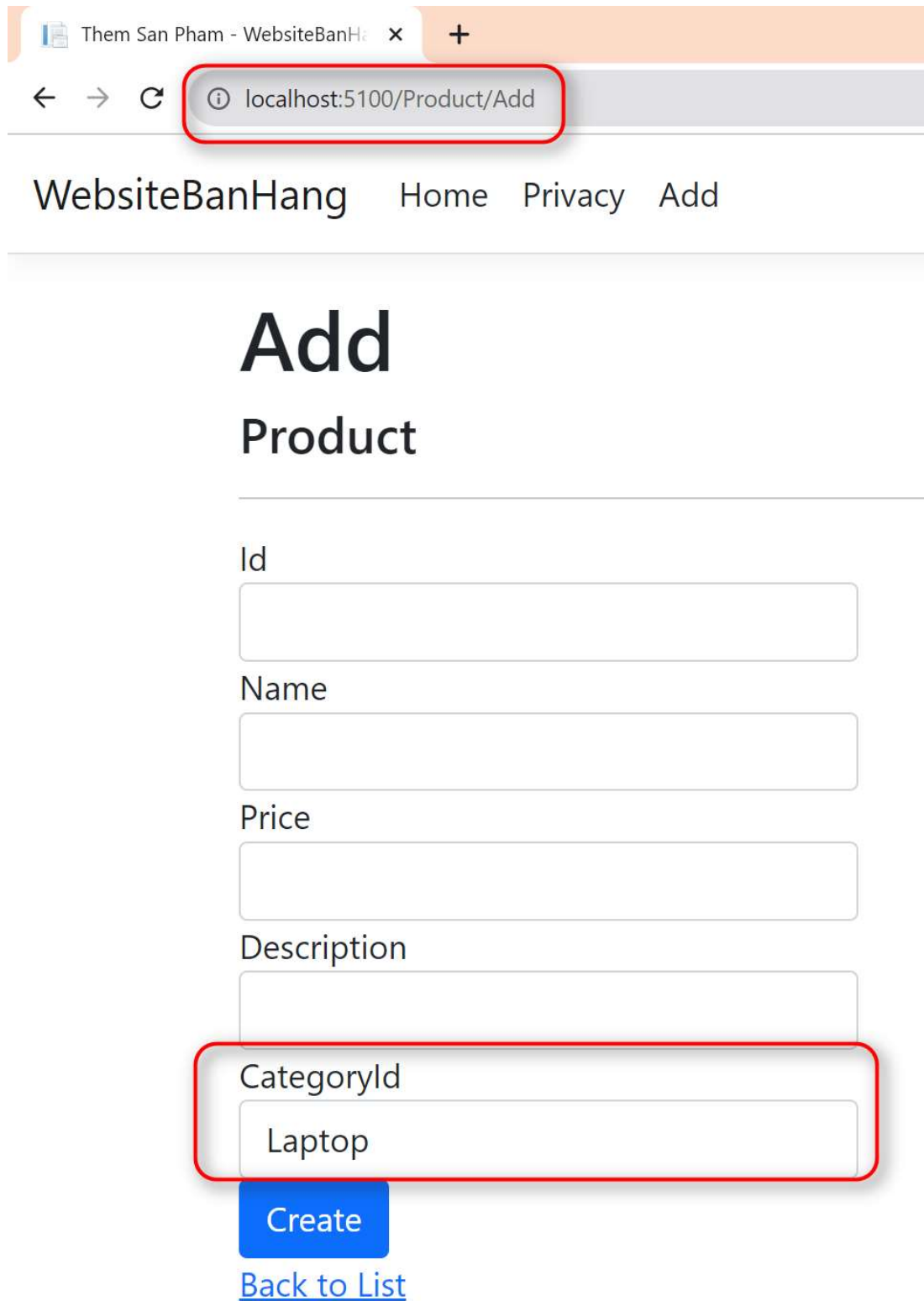
- *Update.cshtml (Cập Nhật Sản Phẩm)*

```
@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



Them San Pham - WebsiteBanHang x +

localhost:5100/Product/Add

WebsiteBanHang Home Privacy Add

Add Product

Id

Name

Price

Description

CategoryId

Laptop

Create

[Back to List](#)



Index - WebsiteBanHang

localhost:5100/Product

WebsiteBanHang Home Privacy Add

Index

[Create New](#)

Id	Name	Price	Description	CategoryId	
1	Iphone 15	35000000,00	Iphone 15 pro max	1	Edit Details Delete
2	iphone 16	50000000,00	iphone 16	1	Edit Details Delete

Edit - WebsiteBanHang x +

← → ↻ ⓘ localhost:5100/Product/Edit/1

WebsiteBanHang Home Privacy Add

Edit Product

Id

Name

Price

Description

CategoryId

[Save](#)

[Back to List](#)

2.3.5 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
}
```

- *Cập Nhật View `AddProduct.cshtml`*

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

```
@model YourNamespace.Models.Product

<!-- 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>

<div class="form-group">
    <label>Additional Images</label>
    <input type="file" asp-for="ImageUrls" multiple class="form-control" />
</div>

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

- *Cập nhật thêm vào form Add Product*

```
<form asp-action="Add" enctype="multipart/form-data"> ..... </form>
```

- *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 additionalImages)
            {
                // 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)
{
    var savePath = Path.Combine("wwwroot/images", image.FileName); // Thay
    đổi đường dẫn theo cấu hình của bạn
    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
```

- *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>@Model.Name</h2>


@foreach (var imageUrl in Model.ImageUrls)
{
    
}
```

Kết quả

Them San Pham - WebsiteBanHàng x +

localhost:5100/Product/Add

Add Product

Id

Name

Price

Description

CategoryId

Image

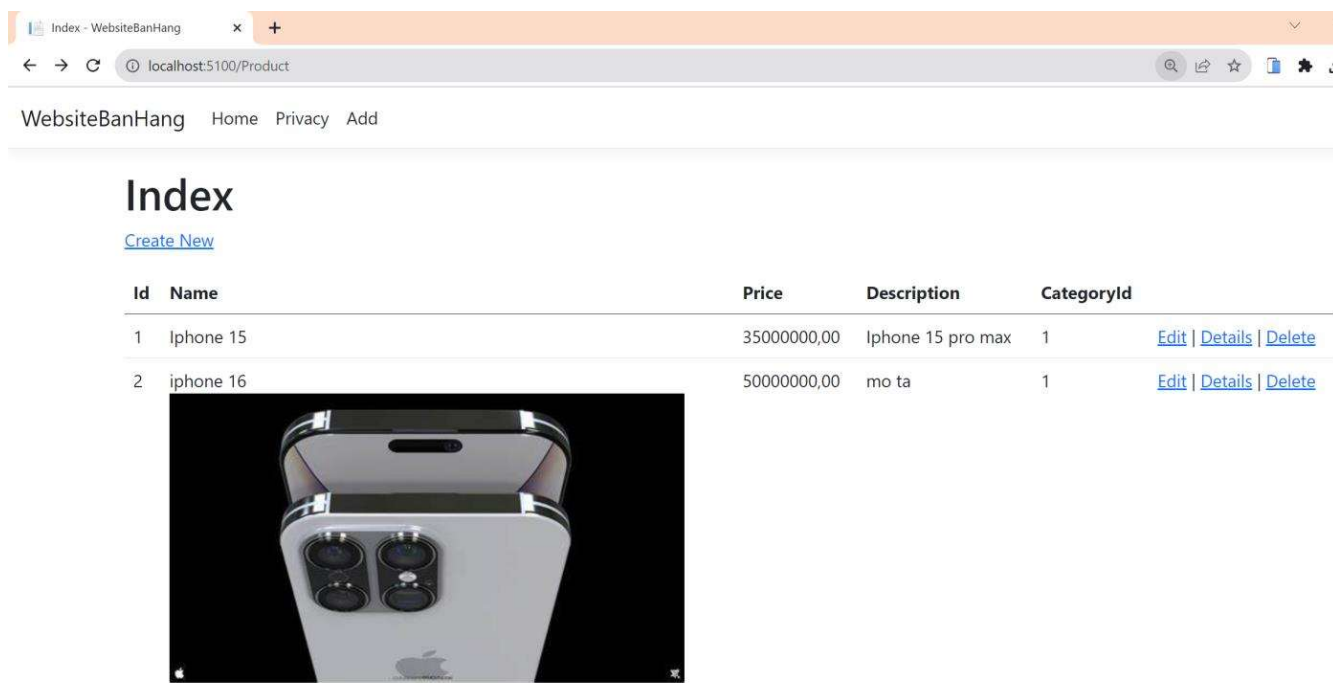
Choose File No file chosen

Additional Images

Choose Files No file chosen

Create

[Back to List](#)



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.

2.3.6 Yêu cầu bổ sung

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