Phần Category tag ở trang con đang lỗi dấu + , chưa có rating

Bài web nhóm 14 : chưa có wishlist , chưa có thanh toán

Thứ tự thực hiện và giải thích

I, các bước mở đầu trong 1 web

1, file Program

II,

1, Tạo folder Controller, tạo HomeController , rồi add 1 view là Index vào

Trong HomeController thì không trẻ về view Index ở ngoài vội , vì mình làm trong phần admin trước , nên là sẽ trả về Admin :

return Redirect("/Admin");

2( Bên trong Areas/Admin)

. Bước đầu để đi đến admin thì phải vào Program thêm dòng lệnh này

app.MapControllerRoute(

name: "admin",

pattern: "{area}/{controller=Home}/{action=Index}/{id?}");

* Tạo Areas/Admin gồm có Controllers và Views , chúng ta sẽ thực hiện phần Admin ở trong Areas/Admin
* Tạo HomeControllers ở trong Controller trong Admin , nhớ là trong tất cả các controller trong Admin đều phải có tag [Area("Admin")]
* Lấy giao diện Admin từ template
* Trong View thì tạo ra Shared / Layout.cshtml
* Tiếp theo tạo folder wwwroot trong đó gồm folder Admin và folder Frontend
* Taọ folder Document ở trong project để lưu template Admin với giao diện
* Coppy các file css của giao diện admin cho vào mục Admin trong wwwroot
* Coppy index.html trong dd admin cho vào Layout.cshtml
* Để load được các view thì cần phải có RenderBody() ở phần thân chính giữa , sửa lại các đường dẫn từ wwwroot/Admin đến các file css , img , js
* Add file layout vào Layout.cshtml trong View/Shared để lấy đc giao diện tĩnh , ở file Layout.cshtml thì dung tag @RenderBody() để xây dựng 1 trang web động và layout này sẽ được sử dụng chung . Ở mỗi trang con sẽ kế thừa layout chung ở đầu trang bằng cách sử dụng sau

@{

Layout = "\_Layout.cshtml"; // Đây là tên tệp layout chung

}

Khi bạn truy cập trang con này, @RenderBody() trong layout chung sẽ hiển thị nội dung của trang con. Điều này cho phép bạn duyệt qua nhiều trang con khác nhau sử dụng cùng một layout chung và chỉ cần cập nhật nội dung cụ thể của từng trang mà không cần thay đổi layout chung.

* Import file layout chung vào Home/Index.cshtml : để load layout chung và đồng thời chạy file con Index.cshtml
* Thêm cách danh mục cần thiết ở file layout.cshtml và đường link dẫn tới từng danh mục
* **Login Logout**
* Tạo ra database có tên là dotnet22\_project
* Tạo table Users gồm có Id , Name , Email , Password
* Add kết nối database vào appsetting.json
* Tải các nuget cần thiết
* Add thư viện mã hóa password using BCrypt.Net;
* Thực hiện mã hóa password 123 để đưa chuỗi mã hóa vào db
* Trong dt Users thì thêm các tk admin
* Tạo ra Model trong project để tương tác với các bảng db trong cơ sở dữ liệu, Vd Table Users
* Trong ItemUser , khai báo các thuộc tính cần lấy trong bảng Users trong db như Id , Name , Email , Password
* Tiếp đến tạo class MydbContext trong Models , ở đây sử dụng Entity Framework Core để làm việc với cơ sở dự liệu

public DbSet<ItemUser> Users { get; set; }

Dòng trên dùng để ánh xạ đối tượng ItemUser vào bảng Users trong db để thực hiện các thao tác CRUD dựa trên bảng Users

* Trong Admin tạo ra folder Attributes , trong Attributes thì tạo ra class CheckLogin.cs , ta sẽ kiểm tra xem người dung đã đăng nhập hay chưa , nếu chưa thì chuyển người dung đến ("/Admin/Account/Login")
* Tiếp đến tạo ra AccountController
* Add view của AccountController là Login.cshtml để hiển thị giao diện đăng nhập
* ở file AccountController , them thư viện mã hóa password , đặt tag [Area(“Admin”)]
* public MyDbContext db = new MyDbContext();: Đây là việc khởi tạo một đối tượng MyDbContext, mà sẽ được sử dụng để thao tác với cơ sở dữ liệu. MyDbContext là một lớp kế thừa từ DbContext và được sử dụng để tương tác với cơ sở dữ liệu.
* public IActionResult Login(): Phương thức này xử lý việc hiển thị trang đăng nhập cho người dùng và trả về một view cho action.
* [HttpPost]: Thuộc tính [HttpPost] áp dụng cho phương thức LoginPost, xác định rằng phương thức này chỉ được gọi khi có yêu cầu HTTP POST. Phương thức này xử lý việc đăng nhập và xác thực thông tin người dùng.
* public IActionResult LoginPost(IFormCollection fc): Phương thức này lấy dữ liệu từ biểu mẫu POST được gửi bởi người dùng. Nó trích xuất email và mật khẩu từ biểu mẫu, sau đó kiểm tra xem thông tin đăng nhập có hợp lệ hay không. Đầu tiên, nó kiểm tra xem email tồn tại trong cơ sở dữ liệu, sau đó so sánh mật khẩu đã nhập với mật khẩu đã mã hóa trong cơ sở dữ liệu bằng cách sử dụng thư viện BCrypt. Nếu xác thực thành công, nó tạo một session cho người dùng và chuyển hướng họ đến trang quản trị, còn không thành công thì nó sẽ chuyển về trangg login
* public IActionResult Logout(): Phương thức này xử lý việc đăng xuất người dùng. Nó hủy phiên làm việc (session) của người dùng và chuyển hướng họ đến trang đăng nhập.
* Thêm tag [CheckLogin] ở HomeController để kiểm tra đăng nhập , muốn nhìn thấy class CheckLogin.cs thì phải them thư viện sau: using project.Areas.Admin.Attributes;
* **User – Danh sách Admin ( sử dụng Linq)**
* Tạo UserController

+) them các thư viện , tag Admin

+) tạo đối tượng thao tác với csdl public MyDbContext db = new MyDbContext();

+) public IActionResult Index(int? page)

+) List<ItemUser> list\_users = db.Users.OrderByDescending(item => item.Id).ToList();: Đoạn này truy vấn dữ liệu từ cơ sở dữ liệu thông qua đối tượng db. Có một danh sách list\_users chứa các bản ghi từ bảng Users. Dữ liệu này được sắp xếp theo trường Id theo thứ tự giảm dần (từ lớn đến nhỏ) và sau đó chuyển thành một danh sách List.

+) return View(list\_users.ToPagedList(pageNumber, pageSize));: Đây là lệnh trả về một trang web cho người dùng. Trang web này sẽ hiển thị danh sách list\_users dưới dạng một trang có phân trang, sử dụng ToPagedList để chia thành các trang con dựa trên pageNumber và pageSize. Kết quả sẽ được trả về dưới dạng một đối tượng IActionResult, thường là một trang web (View) hiển thị danh sách người dùng.

* Tiếp đến tạo Index.cshtml , sửa các đường link cần thiết

+)

@foreach (var item in Model)

{

<tr>

<td>@item.Name</td>

<td>@item.Email</td>

<td style="text-align:center;">

<a href="/Admin/Users/Update/@item.Id">Update</a>&nbsp;

<a href="/Admin/Users/Delete/@item.Id" onclick="return window.confirm('Are you sure?');">Delete</a>

</td>

</tr>

}

@foreach (var item in Model): Đây là vòng lặp foreach được sử dụng để duyệt qua mỗi đối tượng trong biến Model. Mỗi đối tượng trong Model tương ứng với một hàng (row) trong bảng HTML.

<td>@item.Name</td> và <td>@item.Email</td>: Đây là cặp thẻ <td> và </td>, đại diện cho các ô (cell) trong bảng HTML. Trong các ô này, thông tin về tên (Name) và email (Email) của mỗi đối tượng sẽ được hiển thị.

<a href="/Admin/Users/Update/@item.Id">Update</a>: Đây là một liên kết (link) HTML (<a>) dùng để chuyển đến trang cập nhật thông tin người dùng. Liên kết này chứa địa chỉ URL dẫn đến trang cập nhật, và @item.Id sẽ thay thế bằng giá trị Id của đối tượng hiện tại.

<a href="/Admin/Users/Delete/@item.Id" onclick="return window.confirm('Are you sure?');">Delete</a>: Đây là một liên kết khác dùng để xóa người dùng. Khi liên kết này được nhấn, một hộp thoại xác nhận (window.confirm) sẽ xuất hiện để đảm bảo người dùng muốn xóa. Nếu người dùng xác nhận, họ sẽ bị chuyển đến trang xóa (Delete) với @item.Id thay thế bằng giá trị Id của đối tượng hiện tại.

+)Thực hiện phân trang bằng hàm PagedListPager: @Html.PagedListPager(Model,page => Url.Action("Index",new {page} ))

* Thực hiện Update public IActionResult Update(int? id)

+)int \_id = id ?? 0; đầu tiên là kiểm tra xem giá trị id có null hay không , nếu id là null thì \_id = 0

+) Khai báo form dung chung cho Update và Create qua View CreateUpdate.cshtml

+) Truy vấn dữ liệu của người dung , biến record sẽ chứa thông tin của người dung với id tương ứng ( tìm bằng cách so sánh với \_id ) nếu kh tìm thấy , record = null

+)return View("CreateUpdate", record); trả về view CreateUpdate và truyền dữ liệu từ record vào view để hiển thị lên

* Tiếp đến tạo view CreateUpdate.cshtml

+) import file layout vào và khởi tạo biến Name và Email

+) truyền @ViewBag.action vào action ở thẻ form

+) truyền dữ liệu vào value : @Name , @Email

* Tiếp đến là tạo hàm public IActionResult UpdatePost trong UserController , bằng cách sử dụng phương thức HttpPost để xử lí thông tin từ người dùng từ Update

+) Truy vấn cdsl từ bảng Users và kết quả đc gán cho biến record

+) Lấy dữ liệu từ biểu mẫu web và gán vào biến \_Name và \_Email , \_Password

+)Sau khi thu thập dữ liệu người dùng, mã kiểm tra xem người dùng có tồn tại trong cơ sở dữ liệu không. Nếu có, mã tiếp tục kiểm tra xem email đã tồn tại trong cơ sở dữ liệu chưa (với điều kiện email không trùng với bản ghi của chính người dùng đang cập nhật). Nếu email chưa tồn tại, thì mã sẽ cập nhật thông tin của người dùng và lưu vào cơ sở dữ liệu. Nếu email đã tồn tại, mã sẽ trả về một đường dẫn đến trang cập nhật và kèm theo thông báo "email-exists".

* Hàm public IActionResult Create :

+) Trả về biến action giống như Update là về view CreateUpdate.cshtml

* Hàm public IActionResult CreatePost xử lí thông tin người dung đang Create

+) Đầu vào của phương thức nhận đối tượng fc để lấy dữ liệu đc gửi từ biểu mẫu weeeb , rồi trích xuất giá trị từ fc cho các biến \_Name , \_Email , \_Password

+) Mã hóa \_Password bằng hàm BC.BCrypt.HashPassword

+) Tiếp đến là kiểm tra xem email đã tồn tại hay chưa , nếu chưa tồn tại thì mới cho insert vào bản ghi , nếu email đã tồn tại thì người dùng sẽ được chuyển hướng đến trang "/Admin/Users/Create?notify=email-exists" và hiện dòng chữ : Email đã tồn tại , xin mời chọn email khác

* Cuối cùng là public IActionResult Delete(int? id)

+) đầu tiên là kiểm tra giá trị của id , nếu id = null thì \_id = 0

+) ItemUser record = db.Users.Where(item => item.Id == id).FirstOrDefault(); Dòng này truy vấn csdl để lấy 1 bản ghi của người dùng , cụ thể nó tìm bản ghi với trường Id trùng với giá trị id , bản ghi này được gán cho record , với FirstOrDefault thì bạn chọn ra bản ghi đầu tiên hoặc null nếu không tìm ra bản ghi nào

+) db.Users.Remove(record); Dòng này thực hiện xóa bản ghi record của người dùng , nó sử dụng đối tượng db để truy cập csdl và phương thức Remove để xóa bản ghi này

+) Tiếp đến là lưu các thay đổi vào csdl bằng db.SaveChanges();

+) Sau khi xóa thành công thì chuyển người dùng về trang Index.cshtml

* **Categories – Danh mục sản phẩm**
* Add table Categories vào csdl
* Trong Models , tạo ra class ItemCategory.cs , trong ItemCategory khai báo các thư viện nhằm hỗ trợ việc ánh xạ các thuộc tính của lớp này vào csdl

+) [Table(“Category”)] : thuộc tính này xác định tên của bảng csdl tương ứng là Categories , điều này có nghĩa là các đối tượng của lớp ItemCategoy sẽ được lưu trong bẳng csdl có tên Categories

+) Trong ItemCategory thì khai báo các phương thức và thuộc tính đại diện cho các danh mục như Id , ParentId , Name , DisplayHomePage

Bằng cách này có thể sử dụng ItemCategory để ánh xạ vào csdl thông qua Entity Framework

* Ánh xạ đối tượng ItemCategory vào bảng Categories trong db

public DbSet<ItemCategory> Categories { get; set; } //<=> table Categories

* **Tạo CategoriesController** ( phần này sử dụng ADO )

+) Add các thư viện , tag Admin vào

+) Đầu tiên là khởi tạo biến lưu chuỗi kết nối public string strConnectionString;

Tiếp đến là khởi tạo hàm public CategoriesController() để thực hiện việc chuẩn bị chuỗi kết nối đến cơ sở dữ liệu . Như vậy đoạn code trong hàm public CategoriesController được sử dụng để đọc chuỗi kết nối từ tệp cấu hình appsettings.json và lưu trữ trong biến strConnectionString

+) **public IActionResult Index**(int? page): Đây là một phương thức có tên Index được định nghĩa trong Controller. Nó chấp nhận một tham số page kiểu int?, cho phép bạn truyền giá trị trang vào phương thức. Phương thức trả về một đối tượng IActionResult, thường là một trang web để hiển thị dữ liệu cho người dùng.

+)DataTable dt = new DataTable();: Đoạn mã này tạo một đối tượng DataTable để lưu trữ dữ liệu từ cơ sở dữ liệu. DataTable là một biểu đồ dữ liệu chứa dòng và cột dữ liệu.

+)using (SqlConnection conn = new SqlConnection(strConnectionString)): Đoạn mã này tạo một đối tượng kết nối SQL (SqlConnection) và sử dụng chuỗi kết nối strConnectionString để kết nối đến cơ sở dữ liệu. Dùng using đảm bảo rằng tài nguyên kết nối sẽ được giải phóng sau khi hoàn thành công việc.

+)SqlDataAdapter da = new SqlDataAdapter("select \* from Categories where ParentId = 0 order by Id desc", conn);: Đoạn mã này tạo một đối tượng SqlDataAdapter để thực hiện truy vấn SQL và lấy dữ liệu từ cơ sở dữ liệu. Truy vấn SQL lấy tất cả các bản ghi từ bảng "Categories" có "ParentId" bằng 0 và sắp xếp theo cột "Id" theo thứ tự giảm dần.

+)da.Fill(dt);: Đoạn mã này sử dụng SqlDataAdapter để lấy dữ liệu từ truy vấn SQL và đổ nó vào DataTable dt

+)List<ItemCategory> list\_categories = new List<ItemCategory>();: Đoạn mã này khởi tạo một danh sách (List) có tên list\_categories để lưu trữ dữ liệu từ DataTable và phục vụ cho việc hiển thị dữ liệu trên trang.

+)if (dt.Rows.Count > 0) { ... }: Kiểm tra xem DataTable có chứa bất kỳ dữ liệu nào hay không. Nếu có, thực hiện một vòng lặp để lấy từng dòng dữ liệu từ DataTable và thêm chúng vào danh sách list\_categories dưới dạng các đối tượng ItemCategory.

+) int pageSize = 4;: Đây là khai báo số lượng bản ghi trên mỗi trang khi thực hiện phân trang.

+) int pageNumber = page ?? 1;: Dòng này xác định trang hiện tại, sử dụng tham số page được truyền vào phương thức. Nếu page là null, nó sẽ được đặt thành 1 (mặc định hiển thị trang đầu tiên).

+) return View(list\_categories.ToPagedList(pageNumber, pageSize));: Phương thức trả về một trang web hiển thị danh sách list\_categories dưới dạng phân trang, sử dụng thư viện phân trang có thể được gọi bằng phương thức ToPagedList.

- Tiếp đến tạo view Index.cshtml để hiển thị dữ liệu ra

+) Ở trong Index.chsmt ta sử dụng hàm @function

public MyDbContext db = new MyDbContext();: Đoạn mã này tạo một đối tượng MyDbContext, một lớp DbContext của Entity Framework, để tương tác với cơ sở dữ liệu. Điều này giả định rằng bạn đã tạo một lớp MyDbContext để đại diện cho cơ sở dữ liệu của ứng dụng.

public List<ItemCategory> SubCategories(int \_CategoryId) { ... }: Đây là một phương thức có tên SubCategories được định nghĩa trong trang Razor. Phương thức này nhận một tham số \_CategoryId kiểu int, và nó trả về một danh sách (List) các đối tượng ItemCategory.

List<ItemCategory> list\_record = db.Categories.Where(item => item.ParentId == \_CategoryId).OrderByDescending(item => item.Id).ToList();: Đoạn mã này sử dụng Entity Framework để truy vấn cơ sở dữ liệu và lấy danh sách các bản ghi từ bảng "Categories." Nó lọc ra các danh mục con (các bản ghi có "ParentId" bằng \_CategoryId) và sắp xếp chúng theo cột "Id" theo thứ tự giảm dần. Kết quả được chuyển thành một danh sách (List) và lưu trong biến list\_record.

Phương thức SubCategories này được sử dụng để lấy các danh mục con tương ứng với một danh mục cha (dựa vào \_CategoryId) và sẽ được gọi trong trang Razor để hiển thị các danh mục con tương ứng trên giao diện người dùng.

+) Tiếp đến sửa các đường dẫn sao cho đúng

+) Trong vòng lặp foreach thì hiển thị thông tin của mỗi danh mục như là Name , DisplayHomePage , và các liên kêt Update , Delete , Create

- Tiếp đến là phần public IActionResult Update

+)ViewBag.action = "/Admin/Categories/UpdatePost/" + \_id;: Đoạn mã này đặt một giá trị vào ViewBag.action. Giá trị này được sử dụng để xác định hành động (action) của biểu mẫu (form) trong trang web. Trong trường hợp này, nó sẽ dẫn đến việc cập nhật danh mục với ID \_id thông qua hành động UpdatePost.

+)Tạo đối tượng DataTable và kết nối cơ sở dữ liệu: Mã code sử dụng SqlConnection để kết nối đến cơ sở dữ liệu, sau đó thực hiện truy vấn SQL để lấy dữ liệu về danh mục có ID tương ứng \_id. Kết quả truy vấn được đổ vào một đối tượng DataTable dt.

+)DataRow dr = dt.Rows.Count > 0 ? dt.Rows[0] : dt.NewRow();: Đoạn mã này kiểm tra xem có dữ liệu nào trong dt hay không. Nếu có, nó lấy dòng đầu tiên (dòng 0) của dt và lưu trữ nó trong dr. Nếu không có dữ liệu, nó tạo một dòng mới (new row) trong dt và lưu trữ nó trong dr. Điều này sẽ được sử dụng để hiển thị thông tin danh mục trong trang "CreateUpdate."

+)return View("CreateUpdate", dr);: Phương thức trả về một trang web có tên "CreateUpdate" và truyền dữ liệu dr (thông tin của danh mục) vào trang đó để hiển thị và cho phép người dùng cập nhật.

**- Tạo view CreateUpdate.cshtml :**

+) Import file Layout và khởi tạo biến Name , ParentId , DisplayHomePage

+) Trong hàm @function :

public MyDbContext db = new MyDbContext();: Đoạn mã này tạo một đối tượng MyDbContext, một lớp DbContext của Entity Framework, để tương tác với cơ sở dữ liệu. Điều này giả định rằng bạn đã tạo một lớp MyDbContext để đại diện cho cơ sở dữ liệu của ứng dụng.

public List<ItemCategory> Categories(int \_CurrentId) { ... }: Đây là một phương thức có tên Categories được định nghĩa trong trang Razor. Phương thức này nhận một tham số \_CurrentId kiểu int và trả về một danh sách (List) các đối tượng ItemCategory.

List<ItemCategory> list\_record = db.Categories.Where(item => item.ParentId == 0 && item.Id != \_CurrentId).OrderByDescending(item => item.Id).ToList();: Đoạn mã này sử dụng Entity Framework để truy vấn cơ sở dữ liệu và lấy danh sách các bản ghi từ bảng "Categories." Nó lọc ra các danh mục cha (các bản ghi có "ParentId" bằng 0) và đồng thời loại trừ danh mục hiện tại có ID bằng \_CurrentId. Kết quả được sắp xếp theo cột "Id" theo thứ tự giảm dần và chuyển thành một danh sách (List) rồi trả về.

Phương thức Categories này được sử dụng để lấy danh sách các danh mục cha (các danh mục không có danh mục cha) và loại trừ danh mục hiện tại có ID tương ứng. Điều này có thể được sử dụng để tạo danh sách các danh mục cha có thể gán cho danh mục hiện tại khi bạn đang chỉnh sửa hoặc tạo mới một danh mục.

+) Ở dưới thì tạo 1 biến \_CurrentId để lưu trữ giá trị Id từ Model hoặc 0 nếu Model không có dữ liệu.

Gọi phương thức Categories(\_CurrentId) để lấy danh sách các danh mục (có thể là danh sách con) dựa trên giá trị \_CurrentId và lưu trữ nó trong list\_categories.

Sau đó, có một vòng lặp @foreach để liệt kê các danh mục từ danh sách list\_categories. Mỗi danh mục sẽ được hiển thị trong một thẻ option, và nếu danh mục đó trùng với giá trị Model["ParentId"], nó sẽ được chọn (có thuộc tính "selected").

**- UpdatePost**

+) string \_Name = Request.Form["Name"].ToString().Trim();: Đoạn này lấy giá trị từ trường "Name" trong biểu mẫu gửi bằng phương thức POST và lưu vào biến "\_Name." Hàm .Trim() được sử dụng để loại bỏ các khoảng trắng xung quanh giá trị.

+) int \_ParentId = Convert.ToInt32(Request.Form["ParentId"]);: Tương tự, đoạn mã này lấy giá trị từ trường "ParentId" trong biểu mẫu và chuyển đổi thành một số nguyên (\_ParentId) bằng phương thức Convert.ToInt32.

+)int\_DisplayHomePage=!String.IsNullOrEmpty(Request.Form["DisplayHomePage"]) ? 1 : 0;: Đoạn này kiểm tra xem có một trường có tên "DisplayHomePage" trong biểu mẫu hay không. Nếu có, và nó không rỗng (không null hoặc chuỗi rỗng), thì "\_DisplayHomePage" được gán giá trị 1, nếu không, thì gán giá trị 0. Điều này thường sử dụng để kiểm tra xem một ô checkbox được chọn hay không

+)Tiếp theo, mã mở một kết nối đến cơ sở dữ liệu thông qua đối tượng SqlConnection sử dụng chuỗi kết nối được lưu trong biến "strConnectionString."

+) Sau đó, một đối tượng SqlCommand được tạo để thực hiện câu truy vấn SQL. Trong trường hợp này, câu truy vấn là một câu lệnh UPDATE để cập nhật thông tin của một danh mục trong bảng "Categories."

+)Các tham số của câu truy vấn được gán giá trị bằng các biến đã khai báo (\_Name, \_ParentId, \_DisplayHomePage, \_id).

+)cmd.ExecuteNonQuery();: Câu truy vấn được thực thi và dữ liệu trong cơ sở dữ liệu được cập nhật.

+)Cuối cùng, phương thức trả về một điều hướng đến trang "/Admin/Categories," nơi người dùng có thể xem danh sách các danh mục sau khi cập nhật.

**-** **Create** : Một ViewBag được sử dụng để truyền dữ liệu tới view, trong trường hợp này, biến "action" được sử dụng để xác định URL mà biểu mẫu tạo mới danh mục sẽ gửi đến. Sau đó, phương thức trả về view "CreateUpdate," nơi người dùng có thể điền thông tin cho danh mục mới.

**- CreatePost** : Đây là phương thức hành động POST để xử lý yêu cầu tạo mới danh mục. Nó nhận dữ liệu được gửi từ biểu mẫu và thực hiện việc chèn một bản ghi mới vào bảng "Categories" trong cơ sở dữ liệu. Đoạn mã này hoạt động như sau:

+) Lấy dữ liệu từ biểu mẫu, bao gồm tên danh mục (\_Name), danh mục cha (\_ParentId), và một ô kiểm tra (\_DisplayHomePage).

+) Kiểm tra xem ô kiểm tra "\_DisplayHomePage" có được chọn hay không và gán giá trị tương ứng vào "\_DisplayHomePage" (1 nếu được chọn, 0 nếu không).

+)Mở kết nối đến cơ sở dữ liệu và thực hiện câu lệnh SQL INSERT để thêm dữ liệu mới vào bảng "Categories."

Sau khi thêm thành công, chuyển hướng người dùng về trang danh sách danh mục.

**- Delete** : Đây là phương thức hành động để xử lý yêu cầu xóa danh mục. Nó nhận một tham số "id," mà có thể là giá trị null, và thực hiện việc xóa một danh mục dựa trên ID được chỉ định. Đoạn mã này hoạt động như sau:

+) Lấy giá trị của "id" hoặc gán giá trị mặc định là 0 nếu "id" là null.

Mở kết nối đến cơ sở dữ liệu và thực hiện câu lệnh SQL DELETE để xóa danh mục có ID bằng hoặc là danh mục cha của danh mục có ID tương ứng. Câu lệnh SQL này sẽ xóa danh mục và các danh mục con nếu có.

Sau khi xóa thành công, chuyển hướng người dùng về trang danh sách danh mục.

**\* News**

**- Tạo NewsController**

+) Thêm các thư viện , tag Admin, CheckLogin

+) Khởi tạo 1 đối tượng MydbContext để thao tác với cơ sở dữ liệu

+) **public IActionResult Index :**

Cũng khai báo số bản ghi trên 1 trang và tính số trang

List<ItemNews> list\_record = (from item in db.News orderby item.Id descending select item).ToList();: Đoạn mã này truy vấn cơ sở dữ liệu để lấy danh sách các bản ghi từ bảng "News" trong cơ sở dữ liệu. Danh sách này được sắp xếp theo trường "Id" theo thứ tự giảm dần và sau đó được chuyển thành một danh sách List<ItemNews>.

Cuối cùng là trả về 1 view với danh sách các bản ghi được hiển thị trên trạng với số bản ghi trên mỗi trang

- Tiếp tạo file **Index.cshtml** :

+) add các thư viện , import layout vào

+) ở trong hàm @foreach

@if (!String.IsNullOrEmpty(item.Photo)): Đây là một điều kiện kiểm tra xem trường Photo của mục hiện tại có giá trị không rỗng (không null hoặc không trống). Nếu điều kiện này đúng, một ảnh sẽ được hiển thị bằng cách sử dụng đường dẫn từ trường Photo.

<img src="~/Upload/News/@item.Photo">: Nếu điều kiện trong điều kiện if trên đúng, ảnh sẽ được hiển thị với đường dẫn được xây dựng từ thư mục "~/Upload/News/" và giá trị của trường Photo của mục hiện tạị

Thêm các trường gtri của Name và Hot

Thêm các đường dẫn tới Update , Delete , đường dẫn sẽ chứa Id của mục để xác định mục cần cập nhập

- **Tạo CreateUpdate.cshtml**

+) Import file layout vào , khởi tạo các biến Name , Description , Content

+) thêm dữ liệu vào các hàng như Name , Photo, Description , Content sao cho phù hợp

**- Update:**

+) Truyền vào 1 action trả về view CreateUpdate

+) Truy vấn csdl từ bản News . record được lựa chọn dựa trên điều kiện item.Id == \_id

+) Cuối cùng trả về 1 view có tên "CreateUpdate" và truyền dữ liệu bản ghi record vào view. Điều này cho phép hiển thị thông tin của bản ghi trong trang web khi trình duyệt hiển thị view "CreateUpdate

**- UpdatePost:**

+) Đầu tiên thì cũng kiểm tra giá trị của id và gán giá trị = 0 nếu id là null

+) Tiếp theo thì cũng truy vấn csdl từ bảng News

+) Gán giá trị các trường dữ liệu của bản ghi record từ dữ liệu gửi từ form (được trích xuất từ IFormCollection hoặc Request.Form). Các trường dữ liệu bao gồm Name, Description, Content, và Hot. Có một kiểm tra để xem xét giá trị của Hot có được chọn hay không.

+) Sau đó, có một phần xử lý về hình ảnh (\_Photo). Nếu có một hình ảnh được tải lên qua form, nó sẽ được lấy và tạo tên file mới dựa trên thời gian (để tránh trùng lặp tên file). Hình ảnh mới sau đó sẽ được tải lên và lưu trữ trong thư mục "wwwroot/Upload/News" trong dự án. Đường dẫn của hình ảnh mới sau đó được lưu vào trường Photo của bản ghi record.

+Cuối cùng, dữ liệu cập nhật của bản ghi record được lưu trong cơ sở dữ liệu thông qua db.SaveChanges() và sau đó chuyển hướng người dùng đến trang "Index" bằng cách sử dụng return RedirectToAction("Index");

**- Create:**

+)ViewBag.action được sử dụng để thiết lập URL mà form sẽ gửi dữ liệu đến khi người dùng thực hiện tạo mới bản ghi. Ở đây, nó đặt ViewBag.action thành "/Admin/News/CreatePost".

+)Cuối cùng, phương thức trả về một view có tên "CreateUpdate" để hiển thị trang tạo mới.

**- CreatePost:**

+)Phương thức CreatePost có HttpPost decorator, chỉ cho phép xử lý yêu cầu HTTP POST.

+)Đầu tiên, nó tạo một đối tượng ItemNews mới để đại diện cho bản ghi mới.

Sau đó, dữ liệu từ form (các trường Name, Description, Content, và Hot) được trích xuất và gán cho các trường tương ứng của đối tượng record.

+)Nếu có hình ảnh được tải lên, nó sẽ lưu trữ hình ảnh trong thư mục "wwwroot/Upload/News" với tên định dạng bao gồm thời gian và tên file ban đầu. Đường dẫn của hình ảnh sau đó được gán vào trường Photo của bản ghi record.

+)Bản ghi record được thêm vào cơ sở dữ liệu thông qua db.News.Add(record).

Cuối cùng, cơ sở dữ liệu được cập nhật bằng cách gọi db.SaveChanges() và sau đó chuyển hướng người dùng đến trang "Index".

**\*Products – Sản phẩm ( làm sau) là buổi 15 khóa 21**

**\* Adv, Slides,Tags tương tự**

**\* Phân quyền cũng làm sau**

**\* Tiếp đến là phần add frontend vào**

**-** Đầu tiên thì trong Views tạo Shared rồi tạo lấy 3 cái view dùng chung : Header.cshtml , LayoutTrangChu.cshtml , LayoutTrangTrong.cshtml

- Coppy css , fonts, images. Js từ template vào Frontend trong wwwroot

- Coppy Index trong template vào LayoutTrangChu rồi cắt phần thân gồm slider và main cho vào index.cshtml ở Views/Home

- Cắt cả header rồi cho vào Header.cshtml để dùng chung , trong Header.cshtml thì chỉnh các đừng dẫn tới Home , News, Cart , Contact sao cho đúng

**\* Bắt đầu đến phần đẩy dữ liệu lên**

**- Đầu tiên là trong View/Home/Index**

+) Truy xuất csdl từ các bảng đẩy lên web

+) Đầu tiên là sản phẩm giảm giá : lấy các sản phẩm có Discount > 0 từ bảng Products , gồm 9 sản phẩm được sắp xếp theo Id của các sản phẩm

+) Sản phẩm nổi bật : lấy các sản phẩm có Hot = 1 từ bảng Products

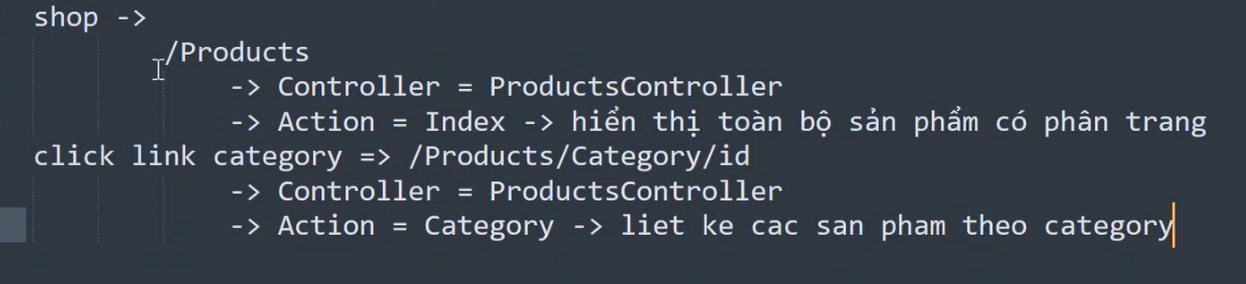
+) Tiếp theo là trả về các danh sách sản phẩm ( ItemCategory ) mà có DisplayHomePage = 1

+) Tiếp theo thì mỗi danh mục liệt kê 9 sản phẩm nổi bật có Id bằng \_CategoryId , dữ liệu được lấy từ 3 bảng : Categories , CategoriesProducts và Products

+) Cuối cùng là Tin tức nổi bật : trả về danh sách các tin tức nổi bật có Hot = 1

+) Sau đó tương ứng với các mục cần đẩy dữ liệu lên , hiển thị dữ liệu cần hiển thị như @item.Name , @item.Discount , @item.Id….

- Tiếp đến là phần Shop



+) Đầu tiên là tạo ProductsController , them các thư viện , rồi add 1 cái View là Index , view này sẽ lưu phần khung chính của phần Shop mà coppy từ template paste vào LayoutTrangTrong

+) Cắt phần chính giữa của LayoutTrangTrong ra rồi paste vào Index của Views/Products/Index.cshtml

+) Trong Views/Products/Index.cshtml thì cũng làm tương tự như phần Sale Item bên Views/Home/Index.cshtml

+) Tiếp đến là hiển thị các tag danh mục categori cha và con ở bên tay trái của LayoutTrangChu và LayoutTrangTrong ( Phần này đang lỗi )

+) Tiếp theo là phần sắp xếp:

.Ở trong ProductsControlles: Đầu tiên tạo 1 biến kiểu chuỗi là order có giá trị rỗng , Sau đó, code kiểm tra xem có tham số order được truyền vào thông qua URL không bằng cách sử dụng Request.Query["order"]. Nếu tham số này tồn tại và không rỗng, thì giá trị của tham số sẽ được gán vào biến order.

Sau đó gtri của biến order được đặt trong ViewBag với tên là Order , tiếp theo 1 câu lệnh switch được sử dụng để xác định giá trị của biến order vào thực hiện sắp xếp danh sách dựa trên 4 trường hợp : tang dần và giảm dần theo thuộc tính Id , theo giá từ thấp đến cao vào theo giá từ cao xuống thấp theo thuộc tính Price

Tiếp đến là Category.cshtml , <select id="orderby" onchange="location.href='/Products/Category/@ViewBag.CategoryId?order='+document.getElementById('orderby').value;" class="use-chosen">: Đây là một phần tử <select> (danh sách thả xuống) cho phép người dùng chọn cách sắp xếp danh sách sản phẩm.

Chỗ này tạo ra 1 giao diện cho việc sắp xếp danh sách sản phẩm dựa trên các tùy chọn được chọn từ danh sách thả xuống. Khi người dùng chọn một tùy chọn, trang web sẽ chuyển đổi URL để thực hiện sắp xếp dựa trên giá trị của biến order và thể hiện kết quả trong danh sách sản phẩm

- Tiếp đến là phần Detail từng sản phẩm

+) Trong Producs thì khai báo **public IActionResult Detail** rồi add 1 view là Detail.cshtml vào

+) Trong Detail thì Import LayoutTrangTrong vào

Trong hàm @function khai báo MyDbContext để thao tác với csdl

Trong GetCategoryName thì dùng Linq join giữa 3 bảng Categories, CategoriesProducts , Products để lấy thông tin về danh mục của sản phẩm cần tìm

Ở dưới phần view thì thêm các @Name , @Discount … và các đường dẫn tương ứng

- Phần RECOMMENDED ITEMS tương tự bên trang chủ

- Login , Logout , Register : Giống bên Admin

- Tiếp đến là phần Cart

+) Đầu tiên là ở tạo 1 view Cart/Index.cshtml ,

+) Ở trong CartController , them các thư viện cần thiết

+) Thêm sản phẩm vào giỏ hàng :

public IActionResult Buy(int id)