Readme

Các bạn muốn giải đề môn nào hoặc nội dung nào thì liên hệ mình qua fb hoặc email nhé

FB: https://www.facebook.com/uyledev/

Email: lequocuyit@gmail.com

Cấu trúc thư mục

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Razor Page -> Repo -> DAO -> BO

Tải Package

Install-Package Microsoft.EntityFrameworkCore.SqlServer -Version 6.0.23

Install-Package Microsoft.EntityFrameworkCore.Tools -Version 6.0.23

Install-Package Microsoft.EntityFrameworkCore.Design -Version 6.0.23

Install-Package Microsoft.Extensions.Configuration.Json -Version 8.0.0

Lệnh chuyển từ DB sang code

scaffold-dbcontext "Server=(local); Database=OilPaintingArt2024DB; Uid=sa; Pwd=12345" Microsoft.EntityFrameworkCore.SqlServer

Sửa trong DB Context

private string GetConnectionString()

{

IConfiguration configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", true, true).Build();

return configuration["ConnectionStrings:DefaultConnectionString"];

}

//Them trong app setting

"ConnectionStrings": {

"DefaultConnectionString": "Server=(local); Database=OilPaintingArt2024DB; Uid=sa; Pwd=12345;TrustServerCertificate=True"

}

Vô thi chú ý 7 method

**Login(email,password)** đăng nhập

**GetList()** hoặc **GetList(searchTerm, pageIndex**) lấy 1 list sản phẩm, làm được nâng cao thì làm luôn search

Get By Id

**Create(object)** tạo 1 sản phẩm

**Update(object)** update 1 sản phẩm

**Delete(id)** xóa 1 sản phẩm

5 cái trên chung 1 bảng

**GetList** của 1 bảng khác dùng để select khi tạo sản phẩm

A screenshot of a computer

Description automatically generated

Code tham khảo

Paging:

DAO

public async Task<OilPaintingArtResponse> GetOilPaintingArts(string searchTerm, int pageIndex)

{

var pageSize = 2;

//iqueryable

var query = \_context.OilPaintingArts.Include(x => x.Supplier).AsQueryable();

if (!string.IsNullOrEmpty(searchTerm))

{

query = query.Where(x => x.OilPaintingArtStyle.Contains(searchTerm) || x.Artist.Contains(searchTerm));

}

var count = await query.CountAsync();

var totalPages = (int)Math.Ceiling(count / (double)pageSize);

var result = new OilPaintingArtResponse

{

OilPaintingArts = await query.Skip((pageIndex - 1) \* pageSize).Take(pageSize).ToListAsync(),

TotalPages = totalPages,

PageIndex = pageIndex

};

return result;

}

public class OilPaintingArtResponse

{

public List<OilPaintingArt> OilPaintingArts { get; set; }

public int TotalPages { get; set; }

public int PageIndex { get; set; }

}

=======================

<nav>

<ul class="pagination">

@for(var i = 1; i<Model.TotalPages; i++)

{

<li class="page-item @(Model.PageIndex == i ? "active" : "")">

<**a** class="page-link" **asp-page**="./Index" **asp-route-pageIndex**="@i">@i</**a**>

</li>

}

</ul>

</nav>

//validate

using System.ComponentModel.DataAnnotations;

namespace OldPaintingArtBOs

{

public partial class OilPaintingArt

{

[Required(ErrorMessage = "Oil Painting Art Id is required")]

public int OilPaintingArtId { get; set; }

[Required(ErrorMessage = "Art Title is required")]

[MinLength(4, ErrorMessage = "Art Title must be more than 4 characters")]

// Each word starts with a capital letter and Not allow number

[RegularExpression(@"^[A-Z][a-z]\*$", ErrorMessage = "Each word starts with a capital letter and Not allow number")]

public string ArtTitle { get; set; } = null!;

[Required(ErrorMessage = "Is required")]

public string? OilPaintingArtLocation { get; set; }

[Required(ErrorMessage = "Is required")]

public string? OilPaintingArtStyle { get; set; }

[Required(ErrorMessage = "Is required")]

public string? Artist { get; set; }

[Required(ErrorMessage = "Is required")]

public string? NotablFeatures { get; set; }

[Required(ErrorMessage = "Is required")]

public decimal? PriceOfOilPaintingArt { get; set; }

[Required(ErrorMessage = "Is required")]

[Range(1, 50, ErrorMessage = "Store Quantity must be greater than 0 and below 50")]

public int? StoreQuantity { get; set; }

public DateTime? CreatedDate { get; set; } = DateTime.Now;

[Required(ErrorMessage = "Is required")]

public string? SupplierId { get; set; }

public virtual SupplierCompany? Supplier { get; set; }

}

}