**TẠO WEBSITE SỬ DỤNG ASP NET CORE 6.0 & SQL SERVER**

1. **Tạo cấu trúc project website**

* src
  + application
    - CafeDevCode.Website (ASP Net Core MVC)
  + database
    - CafeDevCode.Database (Class Library)
    - CafeDevCode.Database.Shared (Class Library)
  + business
    - CafeDevCode.Logic (Class Library)
    - CafeDevCode.Logic.Shared (Class Library)
  + common
    - CafeDevCode.Utils (Class Library)
    - CafeDevCode.Common.Shared (Class Library)

1. **Khởi tạo Entities**

* Xây dựng phần database cho website
* Khởi tạo BaseEntity chứa các thời gian chỉnh sửa và cập nhật.
  + CafeDevCode.Common.Shared
    - Model
      * BaseEntity.cs
* public class BaseEntity
* {
* public string? CreateBy { get; set; }
* public DateTime? CreateAt { get; set; }
* public string? LastUpdateBy { get; set; }
* public DateTime? LastUpdateAt { get; set; }
* public bool IsDeleted { get; set; }
* public BaseEntity SetCreateInfo(string user, DateTime date)
* {
* this.CreateBy = user;
* this.CreateAt = date;
* return this;
* }
* public BaseEntity SetUpdateInfo(string user, DateTime date)
* {
* this.LastUpdateBy = user;
* this.LastUpdateAt = date;
* return this;
* }
* }
* Khai báo, khởi tạo các Entities và Configuration
  + CafeDevCode.Database
    - Configuration
    - Entities
* Thêm các nuget liên quan (Entity Framework Core & AspNetCore Identity)
  + Microsoft.EntityFrameworkCore
  + Microsoft.EntityFrameworkCore.Tools
  + Microsoft.EntityFrameworkCore.Design
  + Microsoft.EntityFrameworkCore.SqlServer
  + Microsoft.AspNetCore.Identity
  + Microsoft.AspNetCore.Identity.Stores
  + Microsoft.AspNetCore.Identity.EntityFrameworkCore

1. **Khởi tạo AppDatabase DbContext và Migrate**

* Thêm AppDatabase.cs trong CafeDevCode.Database
* public class AppDatabase : IdentityDbContext<User>
* {
* public AppDatabase()
* {
* }
* public AppDatabase(DbContextOptions<AppDatabase> options)
* : base(options) { }
* protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
* {
* optionsBuilder.UseSqlServer();
* }
* public virtual DbSet<Author> Authors => Set<Author>();
* public virtual DbSet<Category> Categories => Set<Category>();
* public virtual DbSet<Comment> Comments => Set<Comment>();
* public virtual DbSet<PlayList> PlayLists => Set<PlayList>();
* public virtual DbSet<PlayListVideo> PlayListVideos => Set<PlayListVideo>();
* public virtual DbSet<Post> Posts => Set<Post>();
* public virtual DbSet<PostCategory> PostCategories => Set<PostCategory>();
* public virtual DbSet<PostRelated> PostRelateds => Set<PostRelated>();
* public virtual DbSet<PostTag> PostTags => Set<PostTag>();
* public virtual DbSet<PostType> PostTypes => Set<PostType>();
* public virtual DbSet<SitePage> SitePages => Set<SitePage>();
* public virtual DbSet<Tag> Tags => Set<Tag>();
* public virtual DbSet<Video> Videos => Set<Video>();
* }
* Migrate database
  + Add-Migration InitialDatabase
* Khởi tạo dữ liệu mặc định Seeder
  + CafeDevCode.Database
    - Seeder
      * AppSeeder.cs
* public class ApppSeeder
* {
* public static async Task InitializeAsync(AppDatabase database,
* UserManager<User> user, RoleManager<IdentityRole> role)
* {
* database.Database.EnsureCreated();
* if (!role.Roles.Any())
* {
* await role.CreateAsync(new IdentityRole
* {
* Id = Guid.NewGuid().ToString(),
* Name = "Admin",
* NormalizedName = "ADMIN",
* });
* await role.CreateAsync(new IdentityRole
* {
* Id = Guid.NewGuid().ToString(),
* Name = "user",
* NormalizedName = "USER",
* });
* }
* if (!user.Users.Any())
* {
* var result = await user.CreateAsync(new User
* {
* Id = Guid.NewGuid().ToString(),
* UserName = "admin",
* NormalizedUserName = "Admin",
* Email = "admin@gmail.com",
* LockoutEnabled = false,
* }, "Admin@123");
* if (result.Succeeded)
* {
* var userResult = await user.FindByNameAsync("admin");
* await user.AddToRoleAsync(userResult, "Admin");
* }
* }
* }
* }

1. **Xây dựng cấu trúc tầng business**

* CafeDevCode.Logic
  + Commands
    - Data
    - Handler
  + Event
    - Data
    - Handler
  + Queries
    - Implement
    - Interface

1. **Cắt layout Admin (Sb Admin)**

* Sử dụng layout admin mẫu từ SbAdmin để làm giao diện quản trị.
  + Tải layout mẫu, thêm các resource vào project (css, js, assets…)
  + Khởi tạo layout Admin từ index.html của SbAdmin
    - CafeDevCode.Website/Views/Shared/\_AdminLayout.cshtml
    - CafeDevCode.Website/Views/Shared/\_AdminAuthenticateLayout.cshtml
    - CafeDevCode.Website/Controllers/HomeController.cs
    - CafeDevCode.Website/Views/Shared/Home/Index.cshtml
    - CafeDevCode.Website/Views/Shared/Partial/AdminLeftMenu.cshtml

1. **Xây dựng tầng business**

* Khởi tạo common sử dụng chung
  + CafeDevCode.Common.Shared/Model/BaseQuery.cs
* public class BaseQuery
* {
* public string? Keywords { get; set; } = string.Empty;
* public int? PageSize { get; set; }
* public int? PageIndex { get; set; }
* }
  + CafeDevCode.Common.Shared/Model/BasePagingData.cs
* public class BasePagingData<T>
* {
* public List<T> Items { get; set; } = new List<T>();
* public int PageIndex { get; set; }
* public int PageSize { get; set; }
* public int TotalItem { get; set; }
* public int TotalPage { get; set; }
* }
  + CafeDevCode.Common.Shared/Model/BaseCommandResult.cs
* public class BaseCommandResult
* {
* public bool Success { get; set; }
* public string Messages { get; set; } = string.Empty;
* public List<BaseError> Errors { get; set; } = new List<BaseError>();
* }
  + CafeDevCode.Common.Shared/Model/BaseCommandResultWithData.cs
* public class BaseCommandResultWithData<T> : BaseCommandResult
* {
* public T? Data { get; set; }
* }
  + CafeDevCode.Common.Shared/Model/BaseError.cs
* public class BaseError
* {
* public string? Code { get; set; } = string.Empty;
* public string? { get; set; } = string.Empty;
* public string? Message { get; set; } = string.Empty;
* }
* Thêm thư viện:
  + AutoMapper
  + AutoMapper.Extensions.Microsoft.DependencyInjection
  + MediatR
  + MediatR.Extensions.Microsoft.DependencyInjection
* Xây dựng models tương ứng trong CafeDevCode.Logic.Shared
  + Detail Model
  + Summary Model
* Xây dựng Interface và Implement (business/CafeDevCode.Logic/Queries)
* Xây dựng commands request và handler tương ứng
  + Create command request
* public class CreateAuthor : Author,
* IIdentifiedCommand,
* IRequest<BaseCommandResultWithData<Author>>
* {
* public string? RequestId { get; set; }
* public string? IpAddress { get; set; }
* public string? UserName { get; set; }
* }
  + Create command handler
* public class CreateAuthorHandler : IRequestHandler<CreateAuthor, BaseCommandResultWithData<Author>>
* {
* private readonly IMapper mapper;
* private readonly AppDatabase database;
* public CreateAuthorHandler(IMapper mapper,
* AppDatabase database)
* {
* this.mapper = mapper;
* this.database = database;
* }
* public Task<BaseCommandResultWithData<Author>> Handle(CreateAuthor request,
* CancellationToken cancellationToken)
* {
* var result = new BaseCommandResultWithData<Author>();
* try
* {
* var author = mapper.Map<Author>(request);
* author.SetCreateInfo(request.UserName ?? string.Empty, AppGlobal.SysDateTime);
* database.Authors.Add(author);
* database.SaveChanges();
* result.Success = true;
* result.Data = author;
* }
* catch (Exception ex)
* {
* result.Messages = ex.Message;
* }
* return Task.FromResult(result);
* }
* }
  + Update command request
* public class UpdateAuthor : Author,
* IIdentifiedCommand,
* IRequest<BaseCommandResultWithData<Author>>
* {
* public string? RequestId { get; set; }
* public string? IpAddress { get; set; }
* public string? UserName { get; set; }
* }
  + Update command handler
* public class UpdateAuthorHandler : IRequestHandler<UpdateAuthor, BaseCommandResultWithData<Author>>
* {
* private readonly AppDatabase database;
* private readonly IMapper mapper;
* public UpdateAuthorHandler(AppDatabase database,
* IMapper mapper)
* {
* this.database = database;
* this.mapper = mapper;
* }
* public Task<BaseCommandResultWithData<Author>> Handle(UpdateAuthor request,
* CancellationToken cancellationToken)
* {
* var result = new BaseCommandResultWithData<Author>();
* try
* {
* var author = database.Authors.FirstOrDefault(x => x.Id == request.Id);
* if (author != null)
* {
* mapper.Map(request, author);
* author.SetUpdateInfo(request.UserName ?? string.Empty, AppGlobal.SysDateTime);
* database.Authors.Update(author);
* database.SaveChanges();
* result.Success = true;
* result.Data = author;
* }
* else
* {
* result.Messages = $"Can't find author ID is {request.Id}";
* }
* }
* catch (Exception ex)
* {
* result.Messages = ex.Message;
* }
* return Task.FromResult(result);
* }
* }
  + Delete command request
* public class DeleteAuthor :
* IIdentifiedCommand,
* IRequest<BaseCommandResult>
* {
* public int Id { get; set; }
* public string? RequestId { get; set; }
* public string? IpAddress { get; set; }
* public string? UserName { get; set; }
* }
  + Delete command handler
* public class DeleteAuthorHandler : IRequestHandler<DeleteAuthor, BaseCommandResult>
* {
* private readonly AppDatabase database;
* private readonly IMapper mapper;
* public DeleteAuthorHandler(AppDatabase database,
* IMapper mapper)
* {
* this.database = database;
* this.mapper = mapper;
* }
* public Task<BaseCommandResult> Handle(DeleteAuthor request,
* CancellationToken cancellationToken)
* {
* var result = new BaseCommandResult();
* try
* {
* var author = database.Authors.FirstOrDefault(x => x.Id == request.Id);
* if (author != null)
* {
* author.MaskDelete(request.UserName ?? string.Empty, AppGlobal.SysDateTime);
* database.Authors.Update(author);
* database.SaveChanges();
* result.Success = true;
* }
* else
* {
* result.Messages = $"Can't find author ID is {request.Id}";
* }
* }
* catch (Exception ex)
* {
* result.Messages = ex.Message;
* }
* return Task.FromResult(result);
* }
* }

*(Tương tự với các object khác)*

* Khởi tạo MappingProfile (CafeDevCode.Logic/MappingProfile)
* public AuthorMappingProfile()
* {
* CreateMap<CreateAuthor, Author>();
* CreateMap<UpdateAuthor, Author>();
* CreateMap<Author, AuthorSummaryModel>()
* .ReverseMap();
* CreateMap<Author, AuthorDetailModel>()
* .ReverseMap();
* }
* Khởi tạo Global (CafeDevCode.Utils/Global/AppGlobal.cs)
  + Get DateTime
* public class AppGlobal
* {
* public static DateTime SystemDateTime => DateTime.Now;
* }

1. **Xây dựng login handler, xây dựng trang đăng nhập**

* Bắt đầu xây dựng login
* Login Command Request
  + CafeDevCode.Logic/Commands/Request/User/Login.cs
* public class Login : IRequest<BaseCommandResultWithData<User>>
* {
* public string? UserName { get; set; }
* public string? Password { get; set; }
* public bool? RememberPassword { get; set; }
* }
* Login Command Handler
  + CafeDevCode.Logic/Commands/Handler/User/LoginHandler.cs
* public class LoginHandler : IRequestHandler<Login, BaseCommandResultWithData<User>>
* {
* private readonly AppDatabase database;
* private readonly IMapper mapper;
* private readonly UserManager<User> userManager;
* private readonly SignInManager<User> signInManager;
* public LoginHandler(AppDatabase database,
* IMapper mapper,
* UserManager<User> userManager,
* SignInManager<User> signInManager)
* {
* this.database = database;
* this.mapper = mapper;
* this.userManager = userManager;
* this.signInManager = signInManager;
* }
* public Task<BaseCommandResultWithData<User>> Handle(Login request, CancellationToken cancellationToken)
* {
* var result = new BaseCommandResultWithData<User>();
* try
* {
* var user = userManager.FindByNameAsync(request.UserName).Result;
* if (user != null)
* {
* var isPasswordValid = userManager.CheckPasswordAsync(user, request.Password).Result;
* if (isPasswordValid)
* {
* result.Success = true;
* result.Data = user;
* }
* else
* {
* result.Messages = AppGlobal.InvalidPassword;
* }
* }
* else
* {
* result.Messages = AppGlobal.InvalidUserName;
* }
* }
* catch (Exception ex)
* {
* result.Messages = ex.Message;
* }
* return Task.FromResult(result);
* }
* }
* Tạo Controllers User
  + CafeDevCode.Website/Controllers/UserController.cs
    - [HttpGet] AdminLogin
    - [HttpPost] AdminLoginSubmit
* Tạo Views User
  + CafeDevCode.Website/Views/User/Index.cshtml
  + CafeDevCode.Website/Views/User/AdminLogin.cshtml
    - Sử dụng Sb Admin login.html để custom lại view
    - Chỉnh sửa file \_AdminAuthenticateLayout.cshtml trong thư mục Shared, @RenderBody = AdminLogin.cshtml
* Tạo Models User
  + CafeDevCode.Website/Models/User/LoginViewModel.cs
* public class LoginViewModel
* {
* [Required(ErrorMessage = "Tên đăng nhập không thể bỏ trống") ]
* public string? UserName { get; set; }
* [Required(ErrorMessage = "Mật khẩu không thể bỏ trống") ]
* public string? Password { get; set; }
* public bool RememberPassword { get; set; }
* }
* Tạo Utils Extension
  + Add Microsoft.Extensions.DependencyInjection.Abstractions
  + Add Microsoft.EntityFramworkCore
  + Add Microsoft.EntityFramworkCore.SqlServer
  + Add Microsoft.AspNetCore.Authentication.Cookies
  + Add Microsoft.AspNetCore.Authentication.Facebook
  + Add Microsoft.AspNetCore.Authentication.Google
  + Add Microsoft.AspNetCore.Identity.EntityFrameworkCore
  + CafeDevCode.Utils/Extensions/ServiceCollectionExtension.cs
* public static class ServiceCollectionExtension
* {
* public static IServiceCollection AddSqlServerDatabase<TContext>(this IServiceCollection services,
* string connectionString)
* where TContext: DbContext
* {
* services.AddDbContext<TContext>(options =>
* options.UseSqlServer(connectionString, c =>
* {
* c.CommandTimeout(1000);
* c.EnableRetryOnFailure(5);
* }));
* return services;
* }
* public static IServiceCollection AddCookiesAuthenticate(this IServiceCollection services,
* IConfiguration configuration)
* {
* services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
* .AddCookie(options =>
* {
* options.ExpireTimeSpan = TimeSpan.FromMinutes(double.Parse(configuration.GetSection("Cookies: TimeOut").Value));
* options.SlidingExpiration = true;
* options.AccessDeniedPath = "/Forbidden";
* options.LoginPath = configuration.GetSection("Cookies: LoginPath").Value;
* });
* return services;
* }
* public static IServiceCollection AddExternalAuth(this IServiceCollection services,
* IConfiguration configuration)
* {
* services.AddAuthentication()
* .AddFacebook(facebookOptions =>
* {
* facebookOptions.AppId = configuration["Authentication:Facebook:AppId"];
* facebookOptions.AppSecret = configuration["Authentication:Facebook:AppSecret"];
* facebookOptions.AccessDeniedPath = configuration["Authentication:Facebook:AccessDeniedPath"];
* })
* .AddGoogle(googleOptions =>
* {
* googleOptions.ClientId = configuration["Authentication:Google:ClientId"];
* googleOptions.ClientSecret = configuration["Authentication:Google:ClientSecret"];
* });
* return services;
* }
* public static IServiceCollection AddIdentityConfig<TUser, TRole, TDbContext>(this IServiceCollection services)
* where TUser : IdentityUser
* where TRole : IdentityRole
* where TDbContext : DbContext
* {
* services.AddIdentity<TUser, TRole>()
* .AddEntityFrameworkStores<TDbContext>()
* .AddDefaultTokenProviders()
* .AddSignInManager();
* services.Configure<IdentityOptions>(options =>
* {
* options.Password.RequireDigit = false;
* options.Password.RequiredLength = 1;
* options.Password.RequireNonAlphanumeric = false;
* options.Password.RequireLowercase = false;
* options.Password.RequireUppercase = false;
* });
* return services;
* }
* }
  + CafeDevCode.Website/Program.cs
    - Add Microsoft.AspNetCore.Mvc.NewtonsoftJson
* public static async Task Main(string[] args)
* {
* var builder = WebApplication.CreateBuilder(args);
* builder.Services.AddControllersWithViews()
* .AddJsonOptions(options => options.JsonSerializerOptions.PropertyNamingPolicy = null)
* .AddNewtonsoftJson(options => options.SerializerSettings.ReferenceLoopHandling =
* Newtonsoft.Json.ReferenceLoopHandling.Ignore);
* //Service Collection Extension
* builder.Services.AddCookiesAuthenticate(builder.Configuration);
* builder.Services.AddSqlServerDatabase<AppDatabase>(builder.Configuration
* .GetConnectionString("Database"));
* builder.Services.AddIdentityConfig<User, IdentityRole, AppDatabase>();
* // Add AutoMapper & MediatR
* builder.Services.AddMediatR(typeof(Login).Assembly);
* builder.Services.AddAutoMapper(typeof(AuthorMappingProfile).Assembly);
* var app = builder.Build();
* if (!app.Environment.IsDevelopment())
* {
* app.UseExceptionHandler("/Home/Error");
* app.UseHsts();
* }
* //Create Scope
* using (var scope = app.Services.CreateScope())
* {
* var userManager = scope.ServiceProvider.GetRequiredService<UserManager<User>>();
* var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();
* var database = scope.ServiceProvider.GetRequiredService<AppDatabase>();
* await database.Database.MigrateAsync();
* await ApppSeeder.InitializeAsync(database, userManager, roleManager);
* }
* app.UseHttpsRedirection();
* app.UseStaticFiles();
* app.UseRouting();
* app.UseAuthorization();
* app.MapControllerRoute(
* name: "default",
* pattern: "{controller=Home}/{action=Index}/{id?}");
* app.Run();
* }
  + CafeDevCode.Utils/Extensions/ModelStateDictionaryExtension.cs
* public static string GetError(this ModelStateDictionary modelState)
* {
* string result = string.Empty;
* modelState.Values.ToList().ForEach(value =>
* {
* result += "\r\n";
* result += string.Join("\r\n", value.Errors.Select(x => x.ErrorMessage));
* });
* return result;
* }
  + CafeDevCode.Website/appsetting.Development.json
* "ConnectionStrings": {
* "Database": "Server=.\\; Database=Website; Trusted\_Connection=True;"
* },
* "Authenticate": {
* },
* "AuthenCookies": {
* "LoginPath": "User/AdminLogin",
* "TimeOut": 30
* }
  + CafeDevCode.Website/Controllers/UserController.cs
    - Func Login
* public UserController(IMediator mediator)
* {
* this.mediator = mediator;
* }
* public async Task<ActionResult> AdminLoginSubmit(LoginViewModel model)
* {
* if (ModelState.IsValid)
* {
* var command = model.ToCommand();
* var result = await mediator.Send(command);
* if (result.Success)
* {
* var user = result.Data!;
* var claims = new List<Claim>()
* {
* new Claim("UserName", user.UserName),
* new Claim("AuthorId", user.AuthorId ?? string.Empty),
* };
* var claimIdentities = new List<ClaimsIdentity>()
* {
* new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme),
* };
* var claimPrincipal = new ClaimsPrincipal(claimIdentities);
* await HttpContext.SignInAsync(claimPrincipal);
* if (string.IsNullOrEmpty(model.ReturnUrl))
* {
* return RedirectToAction("Index", "Home");
* }
* else
* {
* return Redirect(model.ReturnUrl);
* }
* }
* else
* {
* model.ErrorMessage = result.Messages;
* return RedirectToAction("AdminLogin", model);
* }
* }
* else
* {
* model.ErrorMessage = ModelState.GetError();
* return RedirectToAction("AdminLogin", model);
* }
* }
  + CafeDevCode.Website/Views/User/AdminLogin.cshtml
    - Show error when login
* @{
* if (!string.IsNullOrEmpty(Model.ErrorMessage))
* {
* var messages = Model.ErrorMessage.Split("\r\n");
* <ul>
* @foreach (var message in messages)
* {
* if (!string.IsNullOrEmpty(message))
* {
* <li class="text-danger">@message</li>
* }
* }
* </ul>
* }
* }
  + CafeDevCode.Website/Models/User/LoginViewModel.cs
    - Update LoginViewModel
* public string? ReturnUrl { get; set; }
* public string? ErrorMessage { get; set; }
* public Login ToCommand()
* {
* return new Login()
* {
* UserName = UserName,
* Password = Password,
* RememberPassword = RememberPassword,
* };
* }

1. **Tạo view và thêm xóa sửa**

* Tạo các controller cần thiết
  + CafeDevCode.Website/Controllers/
* Tạo các View tương ứng Controller.
  + CafeDevCode.Website/Views/
    - Index.cshtml
    - Detail.cshtml
* Thêm sửa xóa Author.
  + CafeDevCode.Logic
    - Register.cs
* public static class Register
* {
* public static IServiceCollection AddQueries(this IServiceCollection services)
* {
* services.AddScoped<IAuthorQueries, AuthorQueries>();
* return services;
* }
* }
  + CafeDevCode.Website/Program.cs
    - builder.Services.AddQueries();
    - CafeDevCode.Website/Views/Author/Index.cshtml
    - CafeDevCode.Website/Views/Author/Detail.cshtml
* Thêm Base: Models/BaseViewModel.cs
* public class BaseViewModel
* {
* public string? UserName { get; set; } = string.Empty;
* public string? RequestId { get; set; } = string.Empty;
* public string? IpAddress { get; set; } = string.Empty;
* public void SetBaseFromContext(HttpContext context)
* {
* this.IpAddress = context.Connection.RemoteIpAddress?.ToString();
* this.UserName = context.User.Claims.FirstOrDefault(x => x.Type == nameof(UserName))?.Value;
* this.RequestId = context.Connection.Id;
* }
* }
* Thêm Models/Author/AuthorDetailViewModel.cs (Website)
* public class AuthorDetailViewModel : BaseViewModel
* {
* public int Id { get; set; }
* public string FullName { get; set; } = string.Empty;
* [DataType(DataType.PhoneNumber, ErrorMessage = "Số điện thoại không hợp lệ")]
* public string Phone { get; set; } = string.Empty;
* [DataType(DataType.EmailAddress, ErrorMessage = "Email không hợp lệ")]
* public string Email { get; set; } = string.Empty;
* public string ShortName { get; set; } = string.Empty;
* public CreateAuthor ToCreateCommand()
* {
* return new CreateAuthor()
* {
* IpAddress = this.IpAddress,
* FullName = this.FullName,
* Phone = this.Phone,
* Email = this.Email,
* ShortName = this.ShortName,
* };
* }
* public UpdateAuthor ToUpdateCommand()
* {
* return new UpdateAuthor()
* {
* Id = this.Id,
* IpAddress = this.IpAddress,
* FullName = this.FullName,
* Phone = this.Phone,
* Email = this.Email,
* ShortName = this.ShortName,
* };
* }
* }
* Thêm Task<ActionResult> SaveChange thực hiện lưu dữ liệu. (Trong Author controller).
* public async Task<ActionResult> SaveChange(AuthorDetailViewModel model)
* {
* if (ModelState.IsValid)
* {
* model.SetBaseFromContext(HttpContext);
* var commandResult = new BaseCommandResultWithData<Author>();
* if (model.Id == 0)
* {
* var createCommand = model.ToCreateCommand();
* commandResult = await mediator.Send(createCommand);
* }
* else
* {
* var updateCommand = model.ToUpdateCommand();
* commandResult = await mediator.Send(updateCommand);
* }
* if (commandResult.Success)
* {
* return Json(new { success = true,
* message = AppGlobal.DefaultSuccessMessage,
* data = commandResult.Data });
* }
* else
* {
* ModelState.AddModelError("", commandResult.Messages);
* return Json(new { success = false, message = commandResult.Messages });
* }
* }
* else
* {
* return Json(new { success = false, message = ModelState.GetError() });
* }
* }
* Thêm thư viện sweetAlert (wwwroot)
* Add path jquery trong \_AdminLayout (Views/Shared)
* <**script** src="~/sweetalert/sweetalert.min.js"></**script**>
* <**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>
* Trong Views/Author/Detail.cshtml thêm jquery thực hiện lưu dữ liệu.
* <script>
* onSaveChange = () => {
* $.ajax({
* type: 'POST',
* url: '@Url.Action("SaveChange")',
* data: {
* Id: $('#Id').val(),
* FullName: $('#FullName').val(),
* Email: $('#Email').val(),
* Phone: $('#Phone').val(),
* ShortName: $('#ShortName').val(),
* },
* success: (res) => {
* console.log(res);
* if(res.success === true){
* swal({
* title: 'Thành công!',
* text: res.message,
* icon: 'success',
* })
* }
* },
* error:(error) => {
* swal({
* title: 'Có lỗi!',
* text: res.message,
* icon: 'error',
* })
* }
* })
* }
* </script>
* Thêm Action List, Deletetrong Controller.
* public IActionResult List()
* {
* var model = new List<AuthorSummaryModel>();
* model = authorQueries.GetAll();
* return PartialView(model);
* }
* public async Task<ActionResult> Delete(int id)
* {
* var command = new DeleteAuthor()
* {
* Id = id,
* RequestId = HttpContext.Connection.Id,
* IpAddress = HttpContext.Connection.RemoteIpAddress?.ToString(),
* UserName = HttpContext.User.Claims.FirstOrDefault(x => x.Type == "UserName")?.ToString(),
* };
* var result = await mediator.Send(command);
* return Json(new { success = result.Success, message = result.Messages });
* }
* Trong Views/Author/Index.cshtml thêm jquery thực hiện load, xóa dữ liệu.
* $(window).on("load", function () {
* onLoad();
* })
* onLoad = () => {
* $.ajax({
* type: 'GET',
* url: '@Url.Action("List")',
* success: (res) => {
* $('#author-list-body').empty();
* $('#author-list-body').html(res);
* }
* })
* }
* onDelete = (id) => {
* swal({
* title: 'Xác nhận',
* text: 'Bạn có chắc chắn muốn xóa?',
* icon: 'warning',
* buttons: true,
* dangerMode: true,
* }).then((willDelete) => {
* if (willDelete) {
* $.ajax({
* type: 'POST',
* url: '@Url.Action("Delete")',
* data: {
* id: id,
* },
* success: (res) => {
* if (res.success === true) {
* swal({
* title: 'Thành công!',
* text: res.message,
* icon: 'success',
* });
* onload();
* }
* else {
* swal({
* title: 'Lỗi',
* text: res.message,
* icon: 'error',
* })
* }
* }
* })
* }
* })
* }