

appsettings.json:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Host=localhost;Database=lab\_asp\_db;Username=postgres;Password=root"

}

}

Program.cs:

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Lab2.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

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

builder.Services.AddScoped<IRepository<Museum>, MuseumRepository>();

builder.Services.AddScoped<IRepository<Exhibit>, ExhibitRepository>();

var app = builder.Build();

using (var scope = app.Services.CreateScope())

{

var services = scope.ServiceProvider;

var context = services.GetRequiredService<ApplicationDbContext>();

context.Database.Migrate();

}

if (app.Environment.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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();

Repository/IRepository.cs:

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Linq;

using Lab2.Models;

using static Lab2.Models.Exhibit;

public interface IRepository<T> where T : class

{

IEnumerable<T> GetAll();

T Get(int id);

void Add(T entity);

void Update(T entity);

void Delete(int id);

}

public class MuseumRepository : IRepository<Museum>

{

private readonly ApplicationDbContext \_context;

public MuseumRepository(ApplicationDbContext context)

{

\_context = context;

}

public IEnumerable<Museum> GetAll()

{

return \_context.Museums.Include(h => h.Exhibits);

}

public Museum Get(int id)

{

return \_context.Museums.Include(h => h.Exhibits).FirstOrDefault(h => h.Id == id);

}

public void Add(Museum museum)

{

\_context.Museums.Add(museum);

\_context.SaveChanges();

}

public void Update(Museum museum)

{

\_context.Museums.Update(museum);

\_context.SaveChanges();

}

public void Delete(int id)

{

var museum = Get(id);

if (museum != null)

{

\_context.Museums.Remove(museum);

\_context.SaveChanges();

}

}

}

public class ExhibitRepository : IRepository<Exhibit>

{

private readonly ApplicationDbContext \_context;

public ExhibitRepository(ApplicationDbContext context)

{

\_context = context;

}

public IEnumerable<Exhibit> GetAll()

{

return \_context.Exhibits.Include(p => p.Museum);

}

public Exhibit Get(int id)

{

return \_context.Exhibits.Include(p => p.Museum).FirstOrDefault(p => p.Id == id);

}

public void Add(Exhibit exhibit)

{

\_context.Exhibits.Add(exhibit);

\_context.SaveChanges();

}

public void Update(Exhibit exhibit)

{

\_context.Exhibits.Update(exhibit);

\_context.SaveChanges();

}

public void Delete(int id)

{

var exhibit = Get(id);

if (exhibit != null)

{

\_context.Exhibits.Remove(exhibit);

\_context.SaveChanges();

}

}

}

Models/Exhibit.cs:

using System.ComponentModel.DataAnnotations;

namespace Lab2.Models

{

public class Exhibit

{

/// id

public int Id { get; set; }

/// название

[Required(ErrorMessage = "Поле обязательно для заполнения.")]

public string Name { get; set; }

///описание

[Required(ErrorMessage = "Поле обязательно для заполнения.")]

public string Description { get; set; }

/// эпоха

[Required(ErrorMessage = "Поле обязательно для заполнения.")]

public string Era { get; set; }

/// цена

[Required(ErrorMessage = "Поле обязательно для заполнения.")]

public int Price { get; set; }

public int MuseumId { get; set; }

public Museum Museum { get; set; }

}

}

Data/ApplicationDbContext.cs

using Microsoft.EntityFrameworkCore;

using Lab2.Models;

using static Lab2.Models.Exhibit;

using System.Collections.Generic;

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Museum> Museums { get; set; }

public DbSet<Exhibit> Exhibits { get; set; }

}

Controllers/ExhibitController.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Lab2.Models;

namespace Lab2.Controllers

{

public class ExhibitController : Controller

{

private readonly IRepository<Exhibit> \_exhibitRepository;

private readonly IRepository<Museum> \_museumRepository;

public ExhibitController(IRepository<Exhibit> exhibitRepository, IRepository<Museum> museumRepository)

{

\_exhibitRepository = exhibitRepository;

\_museumRepository = museumRepository;

}

// GET

public ActionResult Index()

{

var exhibits = \_exhibitRepository.GetAll();

return View(exhibits);

}

// GET

public ActionResult Details(int id)

{

var exhibit = \_exhibitRepository.Get(id);

return View(exhibit);

}

// GET

public ActionResult Create()

{

var museums = \_museumRepository.GetAll();

ViewBag.Museums = new SelectList(museums, "Id", "Name");

return View();

}

// POST

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Exhibit exhibit)

{

try

{

\_exhibitRepository.Add(exhibit);

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

// GET

public ActionResult Edit(int id)

{

var exhibit = \_exhibitRepository.Get(id);

var museums = \_museumRepository.GetAll();

ViewBag.Museums = new SelectList(museums, "Id", "Name");

return View();

}

// POST

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(int id, Exhibit exhibit)

{

try

{

\_exhibitRepository.Update(exhibit);

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

// GET

public ActionResult Delete(int id)

{

var exhibit = \_exhibitRepository.Get(id);

return View(exhibit);

}

// POST

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(int id, IFormCollection collection)

{

try

{

\_exhibitRepository.Delete(id);

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

}

}