-AppSetting.json

{

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-MvcExample-014b4166-fb86-417c-b509-4313b550e82d;Trusted\_Connection=True;MultipleActiveResultSets=true",

"LibraryDb": "Server=DESKTOP-H233I23\\SQLEXPRESS01;Database=MvcDatabase;Trusted\_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

-Models:

1.)Author.cs

namespace LibraryApp.Models

{

public class Author

{

public int AuthorId { get; set; }

public string Name { get; set; } = "";

public string Bio { get; set; } = "";

public ICollection<Book> Books { get; set; } = new List<Book>();

}

}

2.)Genre.cs

namespace LibraryApp.Models

{

public class Genre

{

public int GenreId { get; set; }

public string Name { get; set; } = "";

public ICollection<Book> Books { get; set; } = new List<Book>();

}

}

3.)Book.cs

namespace LibraryApp.Models

{

public class Book

{

public int BookId { get; set; }

public string Title { get; set; } = "";

public int AuthorId { get; set; }

public Author? Author { get; set; }

public ICollection<Genre> Genres { get; set; } = new List<Genre>();

}

}

-Data

1.)LibraryContext.cs

using Microsoft.EntityFrameworkCore;

using LibraryApp.Models;

namespace LibraryApp.Data

{

public class LibraryContext : DbContext

{

public LibraryContext(DbContextOptions<LibraryContext> options) : base(options) { }

public DbSet<Book> Books { get; set; }

public DbSet<Author> Authors { get; set; }

public DbSet<Genre> Genres { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// Many-to-Many

modelBuilder.Entity<Book>()

.HasMany(b => b.Genres)

.WithMany(g => g.Books);

// One-to-Many

modelBuilder.Entity<Author>()

.HasMany(a => a.Books)

.WithOne(b => b.Author)

.HasForeignKey(b => b.AuthorId);

}

}

}

-Repositories

1.)IGenericRepository.cs

using System.Linq.Expressions;

namespace LibraryApp.Repositories

{

public interface IGenericRepository<T> where T : class

{

Task<IEnumerable<T>> GetAllAsync();

Task<T?> GetByIdAsync(int id);

Task AddAsync(T entity);

Task UpdateAsync(T entity);

Task DeleteAsync(int id);

}

}

2.) Repositories/GenericRepository.cs

using LibraryApp.Data;

using Microsoft.EntityFrameworkCore;

namespace LibraryApp.Repositories

{

public class GenericRepository<T> : IGenericRepository<T> where T : class

{

private readonly LibraryContext \_context;

private readonly DbSet<T> \_dbSet;

public GenericRepository(LibraryContext context)

{

\_context = context;

\_dbSet = \_context.Set<T>();

}

public async Task<IEnumerable<T>> GetAllAsync() => await \_dbSet.ToListAsync();

public async Task<T?> GetByIdAsync(int id) => await \_dbSet.FindAsync(id);

public async Task AddAsync(T entity)

{

await \_dbSet.AddAsync(entity);

await \_context.SaveChangesAsync();

}

public async Task UpdateAsync(T entity)

{

\_dbSet.Update(entity);

await \_context.SaveChangesAsync();

}

public async Task DeleteAsync(int id)

{

var entity = await GetByIdAsync(id);

if (entity != null)

{

\_dbSet.Remove(entity);

await \_context.SaveChangesAsync();

}

}

}

}

Controllers

1.)BookController.cs

using Microsoft.AspNetCore.Mvc;

using LibraryApp.Models;

using LibraryApp.Repositories;

namespace LibraryApp.Controllers

{

public class BooksController : Controller

{

private readonly IGenericRepository<Book> \_repo;

public BooksController(IGenericRepository<Book> repo)

{

\_repo = repo;

}

public async Task<IActionResult> Index()

{

var books = await \_repo.GetAllAsync();

return View(books);

}

[HttpPost]

public async Task<IActionResult> Create([FromBody] Book book)

{

if (ModelState.IsValid)

{

await \_repo.AddAsync(book);

return Json(new { success = true, message = "Book added!" });

}

return Json(new { success = false, message = "Invalid data" });

}

}

}

Views:

Views/Book/Index.cshtml

<h2>Books</h2>

<div id="book-list"></div>

<form id="book-form">

<input type="text" id="title" placeholder="Book Title" />

<button type="submit">Add</button>

</form>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script>

$(function() {

// Load Books

$.get("/Books/Index", function(data) {

$("#book-list").html($(data).find("#book-list").html());

});

// Add Book via AJAX

$("#book-form").submit(function(e) {

e.preventDefault();

$.ajax({

url: "/Books/Create",

type: "POST",

contentType: "application/json",

data: JSON.stringify({ title: $("#title").val(), authorId: 1 }),

success: function(response) {

alert(response.message);

location.reload();

}

});

});

});

</script>

Program.cs:

using LibraryApp.Data;

using LibraryApp.Repositories;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(connectionString));

builder.Services.AddDbContext<LibraryContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("LibraryDB")));

builder.Services.AddScoped(typeof(IGenericRepository<>), typeof(GenericRepository<>));

var app = builder.Build();

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseRouting();

app.MapControllerRoute(

name: "default",

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

app.Run();

