Entity Framework core

ORM 🡺 Object Relational Mapper

It act as bridge b/w C# Class and Database (MS SQL Server )

Cross-platform

Database provider sipport 🡺 Ms SQL Server, My Sql, Postgre SQL, Oracle,Sqlite,Cosmos DB

Different types Approach in EF Core

1. Code First Approach 🡺 When we don’t have db design we want to create everything from C# Class.
2. Database First Approach 🡺 already if you have database design just want to communicate with API,MVC

Steps to perform Code First approach

1. Create a model (C# Class) POCO plain old CLR Object

Appsettings.json

"ConnectionStrings": {

"myconnection": "Data Source=LAPTOP-0TBPBTEL\\SQLEXPRESS;Initial Catalog=Hexa\_21\_Aug\_DB;Integrated Security=True;TrustServerCertificate=True"

}

1. Install packages Microsoft.EntityFrameworkCore.SqlServer
2. Microsoft.EntityFrameworkCore.tools
3. Create Another class within model or Context folder 🡺ApplcationDbContext :DbContext

ApplicationDbContext.cs

public class ApplicationDbContext:DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options):base(options)

{

}

public DbSet<Department> Departments { get; set; } //Table

}

Program.cs

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

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

build your solution

Goto Nuget Package Manager Console

PM> Add-Migration "department table creation"

PM> update-database

Repositories Folder

Add new Interface IDepartmentCRUD

public interface IDepartmentCRUD

{

Task<IEnumerable<Department>> GetAllDepartments();

Task<Department?> GetDepartmentById(int id);

Task<Department> AddNewDepartement(Department department );

Task UpdateDepartement(Department department);

Task DeleteDepartement(int id);

}

Add new class in Repositories folder DepartmentCRUD

using DemoAPI1.Models;

using Microsoft.EntityFrameworkCore;

namespace DemoAPI1.Repositories

{

public class DepartmentCRUD : IDepartmentCRUD

{

private readonly ApplicationDbContext \_context;

public DepartmentCRUD(ApplicationDbContext context)

{

\_context = context;

}

public async Task<IEnumerable<Department>> GetAllDepartments()

{

try

{

return await \_context.Departments.ToListAsync();

}

catch (Exception e)

{

throw new Exception("Error while fetching all Departments"+e.Message);

}

}

public async Task<Department?> GetDepartmentById(int id)

{

try

{

return await \_context.Departments.FirstOrDefaultAsync(d => d.Id == id);

}

catch(Exception e)

{

throw new Exception("Error while fetching Department by Id"+e.Message);

}

}

public async Task<Department> AddNewDepartement(Department department)

{

try

{

if (department == null)

throw new ArgumentNullException("Department is null");

\_context.Departments.Add(department); //insert query

await \_context.SaveChangesAsync(); //commit transaction (or) execute the insert the query

return department;

}

catch(Exception e)

{

throw new Exception("Error while adding new Department"+e.Message);

}

}

public async Task UpdateDepartement(Department department)

{

try

{

var dept = await \_context.Departments.FirstOrDefaultAsync(d => d.Id == department.Id);

if (dept == null)

throw new Exception($"Department with Id {department.Id} not found");

dept.DepartmentName = department.DepartmentName;

dept.Location = department.Location;

await \_context.SaveChangesAsync();

}

catch (Exception e)

{

throw new Exception("Error while updating Department" + e.Message);

}

}

public async Task DeleteDepartement(int id)

{

try

{

var dept = await \_context.Departments.FirstOrDefaultAsync(d => d.Id == id);

if (dept == null)

throw new Exception($"Department with Id {id} not found");

\_context.Departments.Remove(dept); //delete query

await \_context.SaveChangesAsync(); //commit transaction (or) execute the delete the query

}

catch (Exception e)

{

throw new Exception("Error while deleting Department" + e.Message);

}

}

}

}

Program.cs

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

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

builder.Services.AddScoped<IDepartmentCRUD, DepartmentCRUD>();

Create New controller 🡺 DepartmentCRUDController.cs

using DemoAPI1.Repositories;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace DemoAPI1.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class DepartmentCRUDController : ControllerBase

{

private readonly IDepartmentCRUD \_departmentCRUD;

public DepartmentCRUDController(IDepartmentCRUD departmentCRUD )

{

\_departmentCRUD = departmentCRUD;

}

[HttpGet]

public async Task<IActionResult> GetAllDepartments()

{

var departments = await \_departmentCRUD.GetAllDepartments();

return Ok(departments);

}

[HttpGet("{id}")]

public async Task<IActionResult> GetDepartmentById(int id)

{

var department = await \_departmentCRUD.GetDepartmentById(id);

if (department == null)

return NotFound($"Department with Id {id} not found");

return Ok(department);

}

[HttpPost]

public async Task<IActionResult> AddNewDepartment([FromBody] Models.Department department)

{

var newDepartment = await \_departmentCRUD.AddNewDepartement(department);

return CreatedAtAction(nameof(GetDepartmentById), new { id = newDepartment.Id }, newDepartment);

}

[HttpPut]

public async Task<IActionResult> UpdateDepartment([FromBody] Models.Department department)

{

await \_departmentCRUD.UpdateDepartement(department);

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

await \_departmentCRUD.DeleteDepartement(id);

return NoContent();

}

}

}

Build your solution and Execute