Step:1 create model classes

Model class:

Department table:

namespace Employees.model

{

public class departments

{

public int DeptNo { get; set; }

public string DeptName { get; set; }

// Navigation Properties

public ICollection<deptemp> DeptEmp { get; set; }

public ICollection<deptmanager> DeptManager { get; set; }

}

}

Department employee: namespace Employees.model

{

public class deptemp

{

public int EmpNo { get; set; }

public string DeptNo { get; set; }

public DateTime FromDate { get; set; }

public DateTime ToDate { get; set; }

// Navigation Properties

public employees Employee { get; set; }

public departments Department { get; set; }

}

}

Department manager:

namespace Employees.model

{

public class deptmanager

{

public int DeptNo { get; set; }

public int EmpNo { get; set; }

public DateTime FromDate { get; set; }

public DateTime ToDate { get; set; }

// Navigation Properties

public employees Employees { get; set; }

public departments Department { get; set; }

}

}

Employees : namespace Employees.model

{

public class employees

{

public int EmpNo { get; set; }

public DateTime BirthDate { get; set; }

public string First\_Name { get; set; }

public string Last\_Name { get; set; }

public string Gender { get; set; }

public DateTime Hire\_Date { get; set; }

//navigations properties

public ICollection<deptemp> Deptemp { get; set; }

public ICollection<deptmanager> Deptmanager { get; set; }

public ICollection<titles> Title { get; set; }

public ICollection<salaries> Salarie { get; set; }

}

}

Salaries:

namespace Employees.model

{

public class salaries

{

public int EmpNo { get; set; }

public int SalaryAmount { get; set; }

public DateTime FromDate { get; set; }

public DateTime ToDate { get; set; }

// Navigation Property

public employees Employees { get; set; }

}

}

Titles:

namespace Employees.model

{

public class titles

{

public int EmpNo { get; set; }

public string TitleName { get; set; }

public DateTime FromDate { get; set; }

public DateTime ToDate { get; set; }

// Navigation Properties

public employees Employees { get; set; }

}

}

Step:2 create employees controller

Employees controller:

using System;

using Employees.data;

using Employees.model;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace EmployeeAPI.Controllers

{

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

[ApiController]

public class EmployeesController : ControllerBase

{

private readonly EmployeesDbContext dbcontext;

public EmployeesController(EmployeesDbContext context)

{

dbcontext = context;

}

[HttpGet]

public async Task<IActionResult> GetAllEmployees()

{

return Ok(await dbcontext.employees.ToListAsync());

}

[HttpGet("{id}")]

public async Task<IActionResult> GetEmployee(int id)

{

var employee = await dbcontext.employees.FindAsync(id);

if (employee == null) return NotFound();

return Ok(employee);

}

[HttpPost]

public async Task<IActionResult> CreateEmployee(employees employee)

{

dbcontext.employees.Add(employee);

await dbcontext.SaveChangesAsync();

return CreatedAtAction(nameof(GetEmployee), new { id = employee.EmpNo }, employee);

}

[HttpPut("{id}")]

public async Task<IActionResult> UpdateEmployee(int id, employees employee)

{

if (id != employee.EmpNo) return BadRequest();

dbcontext.Entry(employee).State = EntityState.Modified;

await dbcontext.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteEmployee(int id)

{

var employee = await dbcontext.employees.FindAsync(id);

if (employee == null) return NotFound();

dbcontext.employees.Remove(employee);

await dbcontext.SaveChangesAsync();

return NoContent();

}

}

}

Step:3 add dbcontext

Employees dbcontext:

using Employees.model;

using Microsoft.EntityFrameworkCore;

namespace Employees.data

{

public class EmployeesDbContext:DbContext

{

public EmployeesDbContext(DbContextOptions dbContextOptions):base(dbContextOptions)

{

}

public DbSet<departments> departments { get; set; }

public DbSet<deptemp> deptemps { get; set; }

public DbSet<deptmanager> deptmanagers { get; set; }

public DbSet<employees> employees { get; set; }

public DbSet<salaries> salaries { get; set; }

public DbSet<titles> titles { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<employees>().HasKey(e => e.EmpNo);

modelBuilder.Entity<deptemp>().HasKey(de => new { de.EmpNo, de.DeptNo });

modelBuilder.Entity<deptmanager>().HasKey(dm => new { dm.DeptNo, dm.EmpNo });

modelBuilder.Entity<departments>().HasKey(d => d.DeptNo);

modelBuilder.Entity<titles>().HasKey(t => new { t.EmpNo, t.TitleName, t.FromDate });

modelBuilder.Entity<salaries>().HasKey(s => new { s.EmpNo, s.FromDate });

}

}

}

Step:4 add connection string on appsetting.json

"AllowedHosts": "\*",

"ConnectionStrings": {

"EmployeesConnectionString": "Server=DESKTOP-SJG8UOR;database= EmployeesDbContext;trusted\_connection=true;TrustServerCertificate=true"

}

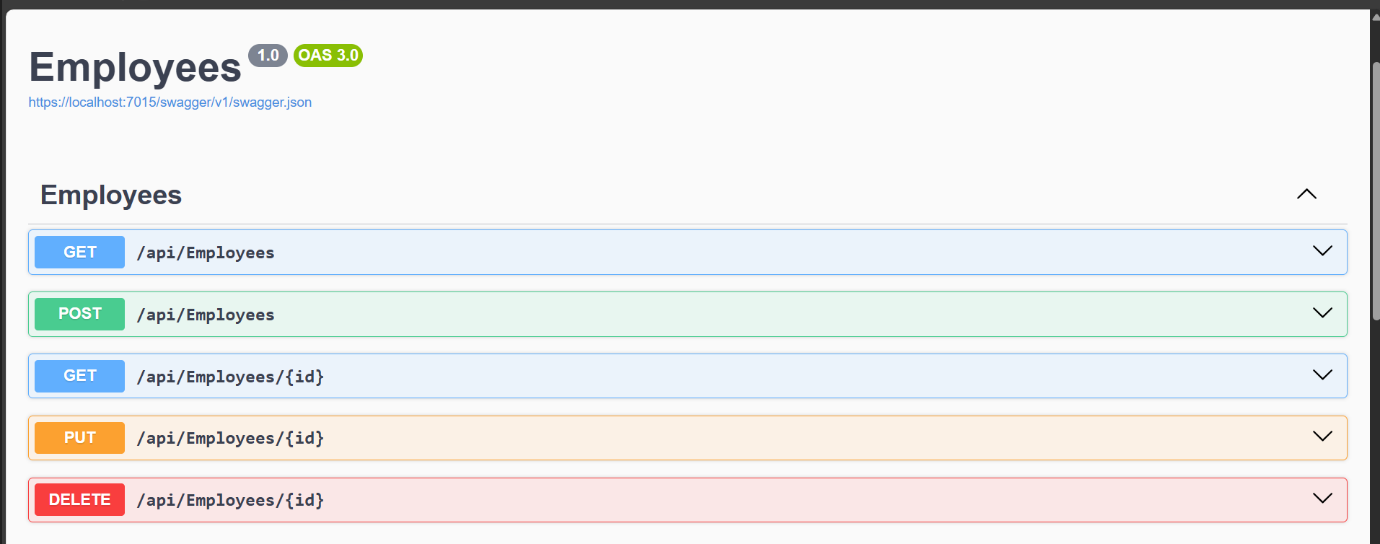
Step:5

Connection to main program.cs

builder.Services.AddDbContext<EmployeesDbContext>(Options =>

Options.UseSqlServer(builder.Configuration.GetConnectionString("EmployeesConnectionString")));

Output:



using Employees.data;

using Employees.model;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

namespace EmployeeAPI.Controllers

{

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

[ApiController]

public class EmployeesController : ControllerBase

{

private readonly IEmployeeRepository \_employeeRepository;

public EmployeesController(IEmployeeRepository employeeRepository)

{

\_employeeRepository = employeeRepository;

}

[HttpGet]

public async Task<IActionResult> GetAllEmployees()

{

var employees = await \_employeeRepository.GetAllEmployees();

return Ok(employees);

}

[HttpGet("{id}")]

public async Task<IActionResult> GetEmployee(int id)

{

var employee = await \_employeeRepository.GetEmployeeById(id);

if (employee == null) return NotFound();

return Ok(employee);

}

[HttpPost]

public async Task<IActionResult> CreateEmployee(employees employee)

{

var createdEmployee = await \_employeeRepository.CreateEmployee(employee);

return CreatedAtAction(nameof(GetEmployee), new { id = createdEmployee.EmpNo }, createdEmployee);

}

[HttpPut("{id}")]

public async Task<IActionResult> UpdateEmployee(int id, employees employee)

{

if (id != employee.EmpNo) return BadRequest();

var success = await \_employeeRepository.UpdateEmployee(employee);

if (!success) return NotFound();

return Ok();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteEmployee(int id)

{

var success = await \_employeeRepository.DeleteEmployee(id);

if (!success) return NotFound();

return Ok();

}

}

}