Entity Frame Work(Repository,Dal,DBCotext) with stored procedures

Install packages:

Microsoft.Entityframework.Core

Microsoft.Entityframework.Core.Tools

Microsoft.Entityframework.Core.SqlServer

Step1: Create Models

namespace StudentRepositoryWebAPI.Models

{

public class Student

{

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime DateofBirth { get; set; }

public string RollNumber { get; set; }

public string Address { get; set; }

public string Marks { get; set; }

public string Grades { get; set; }

}

}

Step2: Create DBContext class

using Microsoft.Data.SqlClient;

using Microsoft.EntityFrameworkCore;

using StudentRepositoryWebAPI.Models;

using System;

namespace StudentRepositoryWebAPI.DataAccessLayer

{

public class StudentDBContext : DbContext

{

public StudentDBContext()

{

}

public StudentDBContext(DbContextOptions<StudentDBContext> options)

: base(options)

{

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("Server=DESKTOP-1JNARAI\\SAISQLSERVER;Initial Catalog=StudentDB;MultipleActiveResultSets=true;User ID=sa;Password=sai123");

}

}

public virtual DbSet<Student> Students { get; set; }

public async Task<List<Student>> GetStudents()

{

return await Students.FromSqlRaw("EXEC usp\_Get\_StudentList").ToListAsync();

}

public async Task<List<Student>> GetByStudentId(int id)

{

// using FromSqlInterpolated

// return await Students.FromSqlInterpolated($"EXEC usp\_GetById\_Student {id}").ToListAsync();

var param = new SqlParameter("@Id", id);

return await Students.FromSqlRaw($"EXEC usp\_GetById\_Student @Id",param).ToListAsync();

}

public async Task<Student> AddStudent(Student student)

{

using (var context = new StudentDBContext())

{

var parameters = new List<SqlParameter>();

parameters.Add(new SqlParameter("@FirstName", student.FirstName));

parameters.Add(new SqlParameter("@LastName", student.LastName));

parameters.Add(new SqlParameter("@DateOfBirth", student.DateofBirth));

parameters.Add(new SqlParameter("@RollNumber", student.RollNumber));

parameters.Add(new SqlParameter("@Address", student.Address));

parameters.Add(new SqlParameter("@Marks", student.Marks));

parameters.Add(new SqlParameter("@Grades", student.Grades));

await context.Database.ExecuteSqlRawAsync($"EXEC usp\_Insert\_checkRollId @FirstName, @LastName, @DateOfBirth, @RollNumber, @Address, @Marks, @Grades", parameters.ToArray());

return await context.Students.FirstOrDefaultAsync(s => s.RollNumber == student.RollNumber);

}

}

public async Task<Student> UpdateStudent(int id,Student student)

{

using(var context = new StudentDBContext())

{

var param = new SqlParameter("@Id",id);

var parameters = new List<SqlParameter>();

parameters.Add(new SqlParameter("@FirstName", student.FirstName));

parameters.Add(new SqlParameter("@LastName", student.LastName));

parameters.Add(new SqlParameter("@DateOfBirth", student.DateofBirth));

parameters.Add(new SqlParameter("@RollNumber", student.RollNumber));

parameters.Add(new SqlParameter("@Address", student.Address));

parameters.Add(new SqlParameter("@Marks", student.Marks));

parameters.Add(new SqlParameter("@Grades", student.Grades));

var result = await context.Database.ExecuteSqlRawAsync($"EXEC usp\_check\_rollId {id},@FirstName, @LastName, @DateOfBirth, @RollNumber, @Address, @Marks, @Grades", parameters.ToArray());

if(result == 1)

{

return await context.Students.FirstOrDefaultAsync(s => s.Id == id);

}

return null;

};

}

public async Task<Student> DeleteStudent(int id)

{

using( var context = new StudentDBContext()) {

await context.Database.ExecuteSqlInterpolatedAsync($"EXEC usp\_Delete\_Student {id}");

return await context.Students.FirstOrDefaultAsync(s => s.Id == id);

}

}

}

}

Step3: Create DAL class:

Create Base Data Access Layer:

using Microsoft.EntityFrameworkCore;

namespace StudentRepositoryWebAPI.DataAccessLayer

{

public class StudentBaseAD : IDisposable

{

public StudentDBContext dBContext = new StudentDBContext();

public StudentBaseAD()

{

dBContext = new StudentDBContext();

}

public void Dispose()

{

}

}

}

using Microsoft.EntityFrameworkCore;

using StudentRepositoryWebAPI.Models;

using StudentRepositoryWebAPI.Repository.Interfaces;

namespace StudentRepositoryWebAPI.DataAccessLayer

{

public class StudentDal: StudentBaseAD

{

//private readonly StudentDBContext \_dBContext;

//public StudentDal(StudentDBContext dBContext)

//{

// \_dBContext = dBContext;

//}

//public StudentDal()

//{

//}

public async Task<List<Student>> GetAllStudentList()

{

//return await \_dBContext.GetStudents();

return await dBContext.GetStudents();

}

public async Task<List<Student>> GetByStudentId(int id)

{

// return await \_dBContext.GetByStudentId(id);

return await dBContext.GetByStudentId(id);

}

public async Task<Student> AddStudent(Student student)

{

// return await \_dBContext.AddStudent(student);

return await dBContext.AddStudent(student);

}

public async Task<Student> UpdateStudent(int id, Student student)

{

//return await \_dBContext.UpdateStudent(id, student);

return await dBContext.UpdateStudent(id, student);

}

public async Task<Student> DeleteStudent(int id)

{

// return await \_dBContext.DeleteStudent(id);

return await dBContext.DeleteStudent(id);

}

}

}

Step4: Create InterfaceRepository class

using StudentRepositoryWebAPI.Models;

namespace StudentRepositoryWebAPI.Repository.Interfaces

{

public interface IStudentRepository

{

Task<List<Student>> GetAllStudents();

Task<List<Student>> GetByStudentId(int id);

Task<Student> AddStudent(Student student);

Task<Student> UpdateStudent(int id, Student student);

Task<Student> DeleteStudent(int id);

}

}

Step5: Create Repository class

using StudentRepositoryWebAPI.DataAccessLayer;

using StudentRepositoryWebAPI.Models;

using StudentRepositoryWebAPI.Repository.Interfaces;

using System.Runtime.CompilerServices;

namespace StudentRepositoryWebAPI.Repository.Services

{

public class StudentRepository : IStudentRepository

{

// private readonly StudentDal \_studentDal;

//public StudentRepository(

// StudentDal studentDal

// )

//{

// \_studentDal = studentDal;

// // Dal = studentDal;

//}

// public StudentDal Dal { get; }

public async Task<List<Student>> GetAllStudents()

{

//return await Dal.GetAllStudentList();

//return await \_studentDal.GetAllStudentList();

using (StudentDal dal = new StudentDal())

{

return await dal.GetAllStudentList();

}

}

public async Task<List<Student>> GetByStudentId(int id)

{

using(StudentDal dal = new StudentDal())

{

return await dal.GetByStudentId(id);

}

// return await \_studentDal.GetByStudentId(id);

}

public async Task<Student> AddStudent(Student student)

{

using (StudentDal dal = new StudentDal())

{

return await dal.AddStudent(student);

}

// return await \_studentDal.AddStudent(student);

}

public async Task<Student> UpdateStudent(int id, Student student)

{

using (StudentDal dal = new StudentDal())

{

return await dal.UpdateStudent(id, student);

}

// return await \_studentDal.UpdateStudent(id,student);

}

public async Task<Student> DeleteStudent(int id)

{

using (StudentDal dal = new StudentDal())

{

return await dal.DeleteStudent(id);

}

// return await \_studentDal.DeleteStudent(id);

}

}

}

Step6: Create Controller

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using StudentRepositoryWebAPI.Models;

using StudentRepositoryWebAPI.Repository.Interfaces;

namespace StudentRepositoryWebAPI.Controllers

{

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

[ApiController]

public class StudentController : ControllerBase

{

private readonly IStudentRepository \_studentRepository;

public StudentController(IStudentRepository studentRepository)

{

\_studentRepository = studentRepository;

}

[HttpGet]

[Route("GetAllStudentsList")]

public async Task<IActionResult> GetAllStudentsList()

{

var result = await \_studentRepository.GetAllStudents();

return Ok(result);

}

[HttpGet]

[Route("GetByStuId")]

public async Task<IActionResult> GetByStuId(int id)

{

var result = await \_studentRepository.GetByStudentId(id);

return Ok(result);

}

[HttpPost]

[Route("AddStudentDetails")]

public async Task<IActionResult> AddStudentDetails(Student student)

{

var result = await \_studentRepository.AddStudent(student);

return Ok(result);

}

[HttpPut]

[Route("UpdateStudentDetails")]

public async Task<IActionResult> UpdateStudentDetails(int id, Student student)

{

var result = await \_studentRepository.UpdateStudent(id, student);

return Ok(result);

}

[HttpDelete]

[Route("DeleteStudentDetails")]

public async Task<IActionResult> DeleteStudentDetails(int id)

{

var result = await \_studentRepository.DeleteStudent(id);

return Ok(result);

}

}

}

Step7: Program.cs:

builder.Services.AddDbContext<StudentDBContext>(options => options.UseSqlServer(builder.Configuration.GetConnectionString("StudentDBCon")));

builder.Services.AddScoped<IStudentRepository , StudentRepository>();

builder.Services.AddScoped<StudentDal>();

step8:appsettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"StudentDBCon": "Server=DESKTOP-1JNARAI\\SAISQLSERVER;Database=StudentDB;User Id=sa;Password=sai123;Integrated Security=False;MultipleActiveResultSets=true;TrustServerCertificate=True;Encrypt=False"

// "StudentDBCon": "Data Source=DESKTOP-1JNARAI\\SAISQLSERVER;Initial Catalog=StudentDB;User ID=sa;Password=sai123"

}

}