Week 9 - Assignment

Activity 1 – ConAppEFCode1

**Program.cs**

using ConAppEFCode1.Models;

using ConAppEFCode1.Repositories;

namespace ConAppEFCode1

{

public class Program

{

public static void Main(string[] args)

{

DepartmentBO dbo = new DepartmentBO(new Models.OrganizationContext());

bool ans = true;

Console.WriteLine("Add new record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

while (ans)

{

Console.WriteLine("Enter Department Name and Location");

string deptName = Console.ReadLine();

string location = Console.ReadLine();

Department dept = new Department()

{

DeptName = deptName,

Location = location

};

bool b = dbo.AddDept(dept);

if (b)

{

Console.WriteLine("Department record created !");

}

Console.WriteLine("Create another record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

}

Console.WriteLine("Departments are ....");

List<Department> deptList = dbo.GetDepartments();

foreach (Department dept in deptList)

{

Console.WriteLine($"{dept.DeptId,5}{dept.DeptName,30}{dept.Location,30}");

}

Console.WriteLine("Want to edit record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

while (ans)

{

Console.WriteLine("Enter Department Id");

int deptId4 = Convert.ToInt32(Console.ReadLine());

Department dept4 = dbo.GetDepartmentById(deptId4);

Console.WriteLine("Enter Department Name and Location");

dept4.DeptName = Console.ReadLine();

dept4.Location = Console.ReadLine();

bool b = dbo.ModifyDept(dept4);

if (b)

{

Console.WriteLine("Department Modified");

}

Console.WriteLine("Create another record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

}

Console.WriteLine("Search record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

while (ans)

{

Console.WriteLine("Enter Department Id");

int deptId = Convert.ToInt32(Console.ReadLine());

Department dept2 = dbo.GetDepartmentById(deptId);

if (dept2 != null)

{

Console.WriteLine($"{dept2.DeptId,5}{dept2.DeptName,30}{dept2.Location,30}");

}

else

{

Console.WriteLine("Department is not available");

}

Console.WriteLine("Search another record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

}

Console.WriteLine("Delete a record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

while (ans)

{

Console.WriteLine("Enter Department Id for remove...");

int deptId2 = Convert.ToInt32(Console.ReadLine());

Department dept3 = dbo.GetDepartmentById(deptId2);

if (dbo.RemoveDept(dept3))

{

Console.WriteLine("Department Deleted");

}

Console.WriteLine("Delete another record? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

}

Console.WriteLine("Re-run Application? Press Y");

ans = Console.ReadLine().ToUpper()[0] == 'Y' ? true : false;

if (ans)

{

Main(args);

}

}

}

}

**appconfig.json**

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="sqlcon" providerName="System.Data.SqlClient"

connectionString="Data Source=LTIN438659\SQLEXPRESS;Initial catalog=TestNewDb;Integrated Security=True;MultipleActiveResultSets=True;TrustServerCertificate=True"/>

</connectionStrings>

</configuration>

**DepartmentBO.cs**

using ConAppEFCode1.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConAppEFCode1.Repositories

{

public class DepartmentBO

{

OrganizationContext context;

public DepartmentBO(OrganizationContext context)

{

this.context = context;

}

public bool AddDept(Department department)

{

bool b=false;

DbSet<Department> departments=context.Departments; //calling the method departments we declared

departments.Add(department);

int r=context.SaveChanges();

if (r > 0)

{

b = true;

}

return b;

}

public List<Department> GetDepartments()

{

DbSet<Department> deptList =context.Departments;

return deptList.ToList();

}

public Department GetDepartmentById(int id)

{

DbSet<Department> departments = context.Departments;

Department department=departments.Find(id); //Find() to find only by primary key.. if find by name/loaction -- do it by Where extension method

return department;

}

public bool RemoveDept(Department department)

{

DbSet<Department> departments = context.Departments;

departments.Remove(department);

int r = context.SaveChanges();

if (r > 0)

{

return true;

}

else

{

return false;

}

}

public bool ModifyDept(Department updateDept)

{

DbSet<Department> departments = context.Departments;

Department department = departments.Find(updateDept.DeptId);

department.DeptName = updateDept.DeptName;

department.Location=updateDept.Location;

int r=context.SaveChanges();

if (r > 0)

{

return true;

}

return false;

}

}

}

**OrganizationContext.cs**

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConAppEFCode1.Models

{

public class OrganizationContext:DbContext

{

public OrganizationContext()

{

//no argument constructor

}

protected override void OnConfiguring(DbContextOptionsBuilder dbco)

{

base.OnConfiguring(dbco); //takes care of building connections

string constring = ConfigurationManager.ConnectionStrings["sqlcon"].ConnectionString;

dbco.UseSqlServer(constring);

}

public virtual DbSet<Department> Departments { get; set; }

public virtual DbSet<Employee> Employees { get; set; }

}

}

**Employee.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConAppEFCode1.Models

{

public class Employee

{

[Key]

public string EmpId { get; set; } //scalar property

[Required, MaxLength(50)]

public string Name { get; set; }

[ForeignKey("Department")]

public int WorkDeptId { get; set; } // set navigation property as we have declared it as foreign key

[Required]

public long Salary { get; set; }

//reference navigation property for work dept id

public Department Department { get; set; }

public ICollection<Employee> Employees { get; set; } //collection navigation property

//because one dept can have many employees, this is an one to many kind of relationship

}

}

A screenshot of a computer program

Description automatically generated

Activity 2 – EFPractice1

**Program.cs**

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace EFPractice1

{

public class Program

{

public static void Main(string[] args)

{

SQLCON2 context = new SQLCON2();

DbSet<User> users=context.Users;

foreach (User user in users.ToList())

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("User Id : {0}",user.UserID);

Console.WriteLine("Full Name : {0}", user.FullName);

Console.WriteLine("Email : {0}", user.Email);

Console.WriteLine("Password : {0}", user.PasswordHash);

Console.WriteLine("Phone : {0}", user.Phone);

Console.WriteLine("Role : {0}", user.Role);

Console.WriteLine("Created At : {0}", user.CreatedAt);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine();

Thread.Sleep(1000);

Console.ReadKey();

}

}

}

}

**App.config**

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

<connectionStrings>

<add name="sqlcon" providerName="System.Data.SqlClient" connectionString="Data Source=LTIN438659\SQLEXPRESS;initial Cataog=BankingDB;integrated security=true" />

<add name="SQLCON2" connectionString="metadata=res://\*/BankEDM.csdl|res://\*/BankEDM.ssdl|res://\*/BankEDM.msl;provider=System.Data.SqlClient;provider connection string=&quot;

data source=LTIN488135\SQLEXPRESS;initial catalog=BankingDB;integrated security=True;trustservercertificate=True;MultipleActiveResultSets=True;App=EntityFramework&quot;"

providerName="System.Data.EntityClient" /></connectionStrings>

</configuration>

**User.cs**

namespace EFPractice1

{

using System;

using System.Collections.Generic;

public partial class User

{

public int UserID { get; set; }

public string FullName { get; set; }

public string Email { get; set; }

public string PasswordHash { get; set; }

public string Phone { get; set; }

public string Role { get; set; }

public Nullable<System.DateTime> CreatedAt { get; set; }

}

}

**BankEDM.Context.cs**

namespace EFPractice1

{

using System;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

public partial class SQLCON2 : DbContext

{

public SQLCON2()

: base("name=SQLCON2")

{

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

throw new UnintentionalCodeFirstException();

}

public virtual DbSet<User> Users { get; set; }

}

}

A screenshot of a computer

Description automatically generated

ACTIVITY 3 – MVCApp1

**Program.cs**

using Microsoft.EntityFrameworkCore;

using MVCApp1.Models;

using System.Configuration;

namespace MVCApp1

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

string conString = builder.Configuration.GetConnectionString("sqlcon");

builder.Services.AddDbContextPool<SportsDbContext>(options=>options.UseSqlServer(conString));

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

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

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.Run();

}

}

}

**Appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"ConnectionStrings": { "sqlcon": "Data Source=LTIN438659\\SQLEXPRESS;Initial Catalog=BankingDB;Integrated Security=True;Trust Server Certificate=True;MultipleActiveResultSets=True" },

"AllowedHosts": "\*"

}

**Game.cs**

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace MVCApp1.Models

{

public class GameEvent

{

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required, MaxLength(30)]

public string TournamentName { get; set; }

[Required, MaxLength(30)]

public string EventName { get; set; }

[Required, MaxLength(30)]

public string Venue { get; set; }

[Required]

public DateTime EventDate { get; }

public ICollection<Performance> Performances { get; set; }

//ICollection for the property which is used for many type of relationship

}

}

**GameEvent.cs**

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace MVCApp1.Models

{

public class GameEvent

{

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required, MaxLength(30)]

public string TournamentName { get; set; }

[Required, MaxLength(30)]

public string EventName { get; set; }

[Required, MaxLength(30)]

public string Venue { get; set; }

[Required]

public DateTime EventDate { get; }

public ICollection<Performance> Performances { get; set; }

//ICollection for the property which is used for many type of relationship

}

}

**Performance.cs**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace MVCApp1.Models

{

public class Performance

{

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int SerialId { get; set; }

[ForeignKey("Player")]

public int PlayerId { get; set; }

[ForeignKey("GameEvent")]

public int EventId { get; set; }

[Required,MaxLength(200)]

public string PerformanceRecord { get; set; }

public int Rank { get; set; }

//one to one relationship..

//player and his performance are 1 to 1

//therefore no ICollections which is used for many to one or vice versa

public Player Player { get; set; }

public GameEvent GameEvent { get; set; }

}

}

**Player.cs**

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace MVCApp1.Models

{

enum Gender

{

Male,Female

}

public class Player

{

[DatabaseGenerated(DatabaseGeneratedOption.None)]

public int PlayerId { get; set; }

//[Required,MaxLength(30)]

public string PlayerName { get; set; }

//[Required, MaxLength(30)]

public string Country { get; set; }

[Required,Range(15,65)]

public int Age { get; set; }

[Required,EnumDataType(typeof(Gender))]

public string Gender { get; set; }

//[ForeignKey("Game")]

public int GameId { get; set; }

//reference navigation property for FK (one to many relationship)

public Game Game { get; set; }

}

}

**SportsDbContext.cs**

using Microsoft.EntityFrameworkCore;

namespace MVCApp1.Models

{

public class SportsDbContext:DbContext

{

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

{

//no argument constructor

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

base.OnConfiguring(optionsBuilder);

}

//fluent api

protected override void OnModelCreating(ModelBuilder builder)

{

base.OnModelCreating(builder);

builder.Entity<Player>().HasKey(p=>p.PlayerId); //configure player id in player instance as key instead of data anotation using fluent api extention method

builder.Entity<Player>().Property(p => p.PlayerName).IsRequired().HasMaxLength(30);

builder.Entity<Player>().Property(p=>p.Country).IsRequired().HasMaxLength(30);

builder.Entity<Player>().Property(p => p.Age).IsRequired();

//builder.Entity<Player>().HasData(

// new Models.Player[]

// {

// new Player(){PlayerId=716,Country="India",Gender=Gender.Male.ToString(),Age=24,PlayerName="Neeraj Chopra",GameId=1},

// new Player(){PlayerId=716,Country="India",Gender=Gender.Female.ToString(),Age=24,PlayerName="PT Usha",GameId=2}

// });

builder.Entity<Game>().HasKey(g => g.GameId);

builder.Entity<Game>().HasMany(g => g.Players).WithOne().HasForeignKey(p => p.GameId);

builder.Entity<Game>().HasData(new Game[]

{

new Game(){GameId=1, Name="Javelin Throw",GameType="Individual"},

new Game(){GameId=2,Name="Women's 400 M Race",GameType="Individual"},

new Game(){GameId=3,Name="Long Jump",GameType="Individual"}

});

builder.Entity<GameEvent>().HasKey(ge=>ge.Id);

builder.Entity<Performance>().HasKey(pf=>pf.SerialId);

}

public virtual DbSet<Player> Players { get; set; }

public virtual DbSet<Game> Game { get;set; }

public virtual DbSet<GameEvent> GameEvent { get; set; }

public virtual DbSet<Performance> Performance { get; set; }

}

}

**IRepository.cs**

namespace MVCApp1.Repositories

{

public interface IRepository<T> where T : class

{

bool Add(T entity);

bool Remove(T entity);

bool Modify (T entity);

List<T> GetAllDetails();

T Search(object Id);

}

}

**GamesBO.cs**

using MVCApp1.Models;

using System.Diagnostics;

namespace MVCApp1.Repositories

{

public class GameBO : IRepository<Game>

{

SportsDbContext context;

public GameBO(SportsDbContext context) //Dependency injection through constructor --> when GameBO is called, SportsDbContext db is invoked automatically

//using controller reference

{

this.context = context;

}

public bool Add(Game entity)

{

try

{

context.Game.Add(entity);

int r = context.SaveChanges();

if (r > 0)

{

return true;

}

return false;

}

catch (Exception ex)

{

Debug.WriteLine("################" + ex.Message);

return false;

}

}

public List<Game> GetAllDetails()

{

return context.Game.ToList();

}

public bool Modify(Game entity)

{

Game game=context.Game.Find(entity.GameId);

game.Name=entity.Name;

game.GameType=entity.GameType;

int r = context.SaveChanges();

if (r > 0)

{

return true;

}

return false;

}

public bool Remove(Game entity)

{

context.Game.Remove(entity);

int r=context.SaveChanges();

if (r > 0)

{

return true;

}

return false;

}

public Game Search(object Id)

{

int gameId = (int)Id;

Game game=context.Game.Find(gameId);

return game;

}

}

}

**PlayerBO.cs**

using MVCApp1.Models;

using System.Diagnostics;

namespace MVCApp1.Repositories

{

public class PlayerBO : IRepository<Player>

{

public SportsDbContext context;

public PlayerBO(SportsDbContext context)

{

this.context = context; //dependency injection

}

public bool Add(Player entity)

{

try

{

context.Players.Add(entity);

int r = context.SaveChanges();

if (r > 0)

{

return true;

}

else

{

return false;

}

}

catch (Exception ex)

{

Debug.WriteLine("################" + ex.Message);

return false;

}

}

public List<Player> GetAllDetails()

{

return context.Players.ToList();

}

public bool Modify(Player entity)

{

Player player = context.Players.Find(entity.PlayerId);

player.PlayerName = entity.PlayerName;

player.Country = entity.Country;

player.Gender = entity.Gender;

player.GameId = entity.GameId;

player.Age = entity.Age;

int r = context.SaveChanges();

if (r > 0)

{

return true;

}

else

{

return false;

}

}

public bool Remove(Player entity)

{

context.Players.Remove(entity);

int r = context.SaveChanges();

if (r > 0)

{

return true;

}

else

{

return false;

}

}

public Player Search(object Id)

{

int playerId = (int)Id;

Player player = context.Players.Find(playerId);

return player;

}

}

}

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a chat

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a chat

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

Activity 4

**Program.cs**

namespace Activity4

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

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

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.Run();

}

}

}

**Appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**MyWebController.cs**

using Microsoft.AspNetCore.Mvc;

namespace Activity4.Controllers

{

public class MyWebController : Controller

{

public IActionResult Greetings(string id)

{

if (string.IsNullOrEmpty(id))

{

return Content("Hello World");

}

else

{

return Content($"Hello {id}");

}

}

}

}

**HomeController.cs**

using Activity4.Models;

using Microsoft.AspNetCore.Mvc;

using System.Diagnostics;

namespace Activity4.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated