**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ВТ**

**отчет**

**по лабораторной работе №2**

**по дисциплине «Программирование в среде .Net»**

**Тема: РАЗРАБОТКА СЛОЯ ДОСТУПА К ДАННЫМ ПРИЛОЖЕНИЯ**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6306 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Бохан Ю.А. |
|  |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Пешехонов К. А. |

Санкт-Петербург

2020

**Цель работы**

Реализовать слой доступа к данным приложения.

**Задание**

Написать слой доступа к данным: Entity Framework Code First + MS SQL Server Developer Edition

**Код программы**

Примеры кода программы даны для сущности «гонщики», другая сущность («команды») реализована похожим образом.

1) Контекст для работы с базой данных

using Formula1.DataAccess.Entities;  
using Microsoft.EntityFrameworkCore;  
using Microsoft.EntityFrameworkCore.Metadata;  
  
namespace Formula1.DataAccess.Context  
{  
 public partial class F1TeamContext : DbContext  
 {  
 public F1TeamContext()  
 {  
 }  
  
 public F1TeamContext(DbContextOptions<F1TeamContext> options) : base(options)  
 {  
 }  
  
 public virtual DbSet<F1Team> F1Team { get; set; }  
 public virtual DbSet<Rider> Rider { get; set; }  
  
 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
 {  
   
 }  
  
 protected override void OnModelCreating(ModelBuilder modelBuilder)  
 {  
 modelBuilder.Entity<F1Team>(entity =>  
 {  
 entity.Property(c => c.Id).UseIdentityColumn().Metadata.SetBeforeSaveBehavior(PropertySaveBehavior.Ignore);  
 entity.Property(c => c.TeamName).IsRequired();  
 entity.Property(c => c.LeaderName).IsRequired();  
 entity.Property(c => c.TeamCountry).IsRequired();  
 });  
  
 modelBuilder.Entity<Rider>(entity =>  
 {  
 entity.Property(s=>s.Id).UseIdentityColumn().Metadata.SetBeforeSaveBehavior(PropertySaveBehavior.Ignore);  
 entity.Property(s => s.Name).IsRequired();  
 entity.Property(s => s.Country).IsRequired();  
 entity.Property(s => s.Seasons).IsRequired();  
 entity.Property(s => s.Champion).IsRequired();  
 entity.Property(s => s.Starts).IsRequired();  
 entity.Property(s => s.Wins).IsRequired();  
 entity.Property(s => s.Poles).IsRequired();  
 entity.HasOne(s => s.F1Team)  
 .WithMany(c => c.Rider)  
 .HasForeignKey(s => s.F1TeamId)  
 .HasConstraintName("FK\_Rider\_F1Team");  
 });  
   
 this.OnModelCreatingPartial(modelBuilder);  
 }  
  
 partial void OnModelCreatingPartial(ModelBuilder modelBuilder);  
  
 }  
}

2) Пример сущности (Rider)

using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace Formula1.DataAccess.Entities  
{  
 public class Rider  
 {  
 [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]  
   
 //Идентификатор  
 public int Id { get; set; }  
   
 //Имя гощника  
 public string Name { get; set; }  
   
 //Национальность  
 public string Country { get; set; }  
   
 //Сезоны  
 public int Seasons { get; set; }  
   
 //Чемпион мира  
 public int Champion { get; set; }  
   
 //Гран-при(старты)  
 public int Starts { get; set; }  
   
 //Победы  
 public int Wins { get; set; }  
  
 //Поулы  
 public int Poles { get; set; }  
   
 public int? F1TeamId { get; set; }  
  
 public virtual F1Team F1Team { get; set; }  
 }  
}

3) Пример интерфейса с декларацией методов для доступа к данным (IRiderDataAccess)

using System.Collections.Generic;  
using System.Threading.Tasks;  
using Formula1.Domain;  
using Formula1.Domain.Contracts;  
using Formula1.Domain.Models;  
namespace Formula1.DataAccess.Contracts  
{  
 public interface IRiderDataAccess  
 {  
 Task<Rider> InsertAsync(RiderUpdateModel rider);  
 Task<IEnumerable<Rider>> GetAsync();  
 Task<Rider> GetAsync(IRiderIdentity riderId);  
 Task<Rider> UpdateAsync(RiderUpdateModel rider);  
 Task<Rider> GetByAsync(IRiderContainer rider);  
  
 }  
}

4) Пример класса с реализацией методов для доступа к данным (RiderDataAccess)

using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
using Formula1.DataAccess.Context;  
using Formula1.DataAccess.Contracts;  
using Formula1.Domain;  
using Formula1.Domain.Contracts;  
using Formula1.Domain.Models;  
using Microsoft.EntityFrameworkCore;  
using AutoMapper;  
namespace Formula1.DataAccess.Implementations  
{  
 public class RiderDataAccess : IRiderDataAccess  
 {  
 private F1TeamContext Context { get; }  
 private IMapper Mapper { get; }  
  
 public RiderDataAccess(F1TeamContext context, IMapper mapper)  
 {  
 this.Context = context;  
 Mapper = mapper;  
 }  
  
 public async Task<Rider> InsertAsync(RiderUpdateModel rider)  
 {  
 var result = await this.Context.AddAsync(this.Mapper.Map<DataAccess.Entities.Rider>(rider));  
 await this.Context.SaveChangesAsync();  
 return this.Mapper.Map<Rider>(result.Entity);  
 }  
  
 public async Task<IEnumerable<Rider>> GetAsync()  
 {  
 return this.Mapper.Map<IEnumerable<Rider>>(await this.Context.Rider.Include(x => x.F1Team).ToListAsync());  
 }  
  
 public async Task<Rider> GetAsync(IRiderIdentity riderId)  
 {  
  
 var result = await this.Get(riderId);  
 return this.Mapper.Map<Rider>(result);  
 }  
   
 private async Task<Formula1.DataAccess.Entities.Rider> Get(IRiderIdentity riderId)  
 {  
 if (riderId == null)  
 throw new ArgumentNullException(nameof(riderId));  
 return await this.Context.Rider.Include(x => x.F1Team).FirstOrDefaultAsync(x => x.Id == riderId.Id);  
   
 }  
  
 public async Task<Rider> UpdateAsync(RiderUpdateModel rider)  
 {  
 var existing = await this.Get(rider);  
  
 var result = this.Mapper.Map(rider, existing);  
  
 this.Context.Update(result);  
  
 await this.Context.SaveChangesAsync();  
  
 return this.Mapper.Map<Rider>(result);  
 }  
  
 public async Task<Rider> GetByAsync(IRiderContainer rider)  
 {  
 return rider.RiderId.HasValue   
 ? this.Mapper.Map<Rider>(await this.Context.Rider.FirstOrDefaultAsync(x => x.Id == rider.RiderId))   
 : null;  
 }  
 }  
}

**Выводы:**

В процессе выполнения лабораторной работы был реализован доступа к данным приложения в среде .NET. Были получены навыки по разработке организации доступа приложения к внешней базе данных с помощью EntityFrameworkCore и MS SQL Server.