Hello App

Entity Framework представляет ORM-технологию (object-relational mapping - отображения данных на реальные объекты) от компании Microsoft для доступа к данным. Entity Framework Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными как с объектами классом независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

https://metanit.com/sharp/efcore/1.5.php

Итак, нам надо определить модель, которая будет описывать данные. Пусть наше приложение будет посвящено работе с пользователями. Поэтому добавим в проект новый класс ***User***:

**Код и обьяснение:**

public class User

{

public int Id { get; set; }

public string? Name { get; set; }

public int Age { get; set; }

public string? Gender { get; set; }

}

Это обычный класс, который содержит несколько свойств. Каждое свойство будет сопоставляться с отдельным столбцом в таблице из бд.

Взаимодействие с базой данных в Entity Framework Core происходит посредством специального класса - контекста данных. Поэтому добавим в наш проект новый класс, который назовем ApplicationContext и который будет иметь следующий код:

namespace HelloApp

{

public class ApplicationContext : DbContext // DbContext: определяет контекст данных, используемый для взаимодействия с базой данных

{

public DbSet<User> Users => Set<User>(); // DbSet/DbSet<TEntity>: представляет набор объектов, которые хранятся в базе данных

public ApplicationContext() => Database.EnsureCreated();

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) // DbContextOptionsBuilder: устанавливает параметры подключения

{

optionsBuilder.UseSqlite("Data Source=helloapp.db");

}

}

}

Теперь определим сам код программы, который будет взаимодействовать с созданной БД. Для этого изменим файл **Program.cs** следующим образом:

using HelloApp;

using (ApplicationContext db = new ApplicationContext())

{

// создаем два объекта User

User tom = new User { Name = "Tom", Age = 33, Gender = "Male" };

User alice = new User { Name = "Alice", Age = 26, Gender = "Female" };

// добавляем их в бд

db.Users.Add(tom);

db.Users.Add(alice);

db.SaveChanges();

Console.WriteLine("Elements were successfuly saved");

// получаем объекты из бд и выводим на консоль

var users = db.Users.ToList();

Console.WriteLine("Elements list:");

foreach (User u in users)

{

Console.WriteLine($"{u.Id}.{u.Name} - {u.Age} - {u.Gender}");

}

}