# Что такое Entity Framework Core

Entity Framework представляет ORM-технологиюот 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.

# Код и пояснение

## Класс User

public class User

{

public int Id { get; set; }

public string? Name { get; set; }

public string? Surname { get; set; }

public string? LivePlace { get; set; }

public string? email { get; set; }

public int Age { get; set; }

}

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

Надо отметить, что Entity Framework требует определения ключа элемента для создания первичного ключа в таблице в бд. По умолчанию при генерации бд EF в качестве первичных ключей будет рассматривать свойства с именами Id или [Имя\_класса]Id (то есть UserId).

## Класс ApplicationContext

{

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

{

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

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

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

{

//DbContextOptionsBuilder с помощью метода UseSqlite позволяет настроить строку подключения для соединения с базой данных SQLite.

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

}

}}

## Класс Program

using (ApplicationContext db = new ApplicationContext())

{

// создаем два объекта User

User artem = new User { Name = "Artem", Surname = "Stryzhakov", LivePlace = "Keila", email = "striz@gmail.com", Age = 18 };

User jaan = new User { Name = "Jaan", Surname = "Krohhin", LivePlace = "Tallinn", email = "krohha@gmail.com", Age = 20 };

// добавляем их в бд

db.Users.Add(artem);

db.Users.Add(jaan);

db.SaveChanges();

Console.WriteLine("Объекты успешно сохранены");

// получаем объекты из бд и выводим на консоль

var users = db.Users.ToList();

Console.WriteLine("Список объектов:");

foreach (User u in users)

{

Console.WriteLine($"{u.Id}.{u.Name},{u.Surname},{u.LivePlace},{u.email},{u.Age}");

}

**}**

## Результат

## 

# Подключение к существующей базе данных

## Класс helloappContext

public partial class helloappContext : DbContext

{

public helloappContext()

{

}

public helloappContext(DbContextOptions<helloappContext> options)

: base(options)

{

}

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlite("Data Source=C:\\Users\\opilane.TTHK\\source\\repos\\ArtemKabilov2\_TARpv20\\EntityFramework\\EntityFramework\\bin\\Debug\\net6.0\\helloapp.db");

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

}

## Переработанный класс Program

using (helloappContext db = new helloappContext())

{

// получаем объекты из бд и выводим на консоль

var users = db.Users.ToList();

Console.WriteLine("Список объектов:");

foreach (User u in users)

{

Console.WriteLine($"{u.Id}.{u.Name},{u.Surname},{u.LivePlace},{u.email},{u.Age}");

}

}

## Результат

## 

# Управление базой данны

Для управления базой данной в классе DbContext определено свойство Database, которое представляет тип Microsoft.EntityFrameworkCore.Infrastructure.DatabaseFacade и которое предоставляет некоторый функционал для управления базой данных.

## Database.EnsureCreated

Метод Database.EnsureCreated() и его асинхронная версия Database.EnsureCreatedAsync() гарантируют, что база данных будет создана.

## Database.EnsureDeleted

Метод Database.EnsureDeleted() и его асинхронная версия Database.EnsureDeletedAsync() гарантируют, что база данных будет удалена.

## Database.EnsureDeleted

Метод Database.EnsureDeleted() и его асинхронная версия Database.EnsureDeletedAsync() гарантируют, что база данных будет удалена.

## Database.CanConnect

Еще один метод, который стоит отметить, это Database.CanConnect() и его асинхронная версия Database.CanConnectAsync().

# Основные операции с данными. CRUD