

Упражнение №4 по ПС Entity Framework

Работа с база данни.

Целта на това упражнение

Да се запознаем с възможността за работа с база данни

Задачите в упражнението изграждат:

Малка студентска информационна система

В това упражнение:

Конзолно приложение, което работи с база данни.

- *Приложението записва и чете от база данни потребителите на системата.*

В края на упражнението:

Ще създадем възможност да записваме потребителите директно в базата данни, вместо статично в кода.

За домашно:

Да се промени кода така, че:

1. Да се добави Logger който да записва в базата данни.
2. Да се добави функционалност в която с меню да можем да избираме дали да вземем всички потребители, да добавим нов потребител или да изтрием съществуващ.
 - a. Добавянето да става чрез въвеждане през конзолата на име и парола
 - b. Изтриването да става чрез въвеждане на имената на потребителя

Важни знания от упражнението: EntityFramework

Зареждане на проект

1. Отворете **Visual Studio**
2. Заредете **Solution**-а създаден в предишните упражнения.
3. Създаваме нов проект, който кръщаваме **DataLayer**.
4. Добавяме референция към проекта **Welcome**
 - a. С десен бутон кликаме върху **Dependencies** на проекта **DataLayer** и избираме **Add Project Reference ...**
 - b. От отвореният прозорец избираме **Welcome** Проекта.

Създаване на папки

1. Създаваме следните папки: **Database, Model**

Добавяне на външни библиотеки

1. Кликваме с десен бутон върху Dependencies и избираме Manage NuGet Packages
2. От новоотвореният прозорец инсталираме следните пакети
 - a. Microsoft.EntityFrameworkCore
 - b. Microsoft.EntityFrameworkCore.Sqlite
 - c. Microsoft.EntityFrameworkCore.Tools

Създаване на клас DatabaseUser

1. Създаваме клас DatabaseUser в папката Model.
2. Новосъздаденият клас трябва да наследи класа User от проекта Welcome
3. Добавете ново свойство Id и го маркираме с override. (Съответно трябва да посочите на класа-родител, че свойството със същото име е virtual.)
4. Добавяме му следните два атрибута посочени на снимката.



```
1 [Key]
2 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
3 public override int Id { get; set; }
```

За да може да използваме тези атрибути ще се наложи да добавим и двата пакета от които идват, а именно System.ComponentModel.DataAnnotations и System.ComponentModel.DataAnnotations.Schema.

Първият атрибут, показва, че това ще бъде ключ в базата данни, а вторият, че стойността на полето ще бъде автоматично генерирано.

Създаване на Клас DatabaseContext

1. В папката **Database** създайте клас – **DatabaseContext**
2. Този клас трябва да наследява **DbContext**, който идва от пакета Microsoft.EntityFrameworkCore
3. Добавете следният код:



```
1 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
2 {
3     string solutionFolder = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
4     string databaseFile = "Welcome.db";
5     string databasePath = Path.Combine(solutionFolder, databaseFile);
6     optionsBuilder.UseSqlite($"Data Source={databasePath}");
7 }
```

В този метод задаваме път към *Sqlite* базата данни която ще използваме, а на последният ред създаваме конфигурацията на базата данни за работа с *Sqlite*.

4. Създаваме метод **OnModelCreating**

```
1 protected override void OnModelCreating(ModelBuilder modelBuilder)
2 {
3     modelBuilder.Entity<DatabaseUser>().Property(e => e.Id).ValueGeneratedOnAdd();
4
5     // Create a user
6     var user = new DatabaseUser()
7     {
8         Id = 1,
9         Name = "John Doe",
10        Password = "1234",
11        Role = UserRolesEnum.Administrator,
12        Expires = DateTime.Now.AddYears(10)
13    };
14
15    modelBuilder.Entity<DatabaseUser>()
16        .HasData(user);
17 }
```

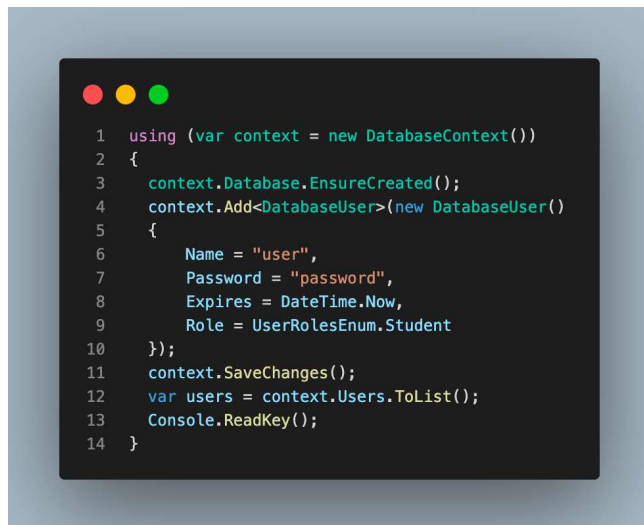
В този метод се описва какво искаме да направим при инициализация на контекста на базата данни, в този случай първо казваме, че полето *Id* на *Entity*-то *DatabaseUser* трябва да се генерира автоматично. След това създаваме потребител в базата като обект и го добавяме в базата ако не съществува.

5. По аналог на горният потребител, създайте няколко в различни роли и *Expire date*.
6. Добавянето на таблица към базата данни става като в този клас добавим свойство от тип ***DbSet<DatabaseUser>***

```
1 public DbSet<DatabaseUser> Users { get; set; }
```

Навързване на кода

1. Във файла Program.cs, премахнете съдържанието на Main метода и добавете следният код:

A screenshot of a code editor with a dark background and light-colored text. The code is in C# and is enclosed in a block with a light blue border. The code defines a DatabaseContext, ensures the database is created, adds a new user, saves changes, and lists the users.

```
1 using (var context = new DatabaseContext())
2 {
3     context.Database.EnsureCreated();
4     context.Add<DatabaseUser>(new DatabaseUser()
5     {
6         Name = "user",
7         Password = "password",
8         Expires = DateTime.Now,
9         Role = UserRolesEnum.Student
10    });
11    context.SaveChanges();
12    var users = context.Users.ToList();
13    Console.ReadKey();
14 }
```

using-a се използва за да гарантираме автоматично освобождаване на контекста след изпълнение на всичко в скобите. Методът EnsureCreated се използва за да се създаде базата данни и всички таблици в нея, ако все още не са създадени. След това добавяме потребител, изпълняваме метода SaveChanges, за да ги запазим в базата. На следващият ред, взимаме всички потребители от базата като List.

Тестване на въведените данни

1. В края на Main метода, добавете възможност за проверка за валиден потребител и парола с който чрез въвеждане последователно на потребител и парола да изкарва съобщение „Валиден потребител“ или „Невалидни данни“ в зависимост от това дали съществува потребителя. Използвайте Linq.