

Упражнение №3 по ПС LINQ

Работа със списъци.

Целта на това упражнение

Работа с колекции, заявки с LINQ и Helper функции

Задачите в упражнението изграждат:

Малка студентска информационна система

В това упражнение:

Конзолно приложение, което надгражда Упражнение №2.

- Създаване на колекция *UserData*, имплементиране на методи за работа с колекцията
- Използване на Хелпър функции

В края на упражнението:

Ще създадем възможност за потребител да влезе в системата с потребител и парола, съществуващ вече в колекция и ще покажем неговите данни в конзолата.

За домашно:

Да се промени кода така, че:

1. Да се направи логър който да записва във файл всеки успешен вход в системата, записа трябва да съдържа, кой потребител кога се е е логнал.
2. Да се записва във файл всеки опит всеки грешен опит за вход със съответната грешка, отново да се използва логър.

Важни знания от упражнението: **Linq, Collections, Lists, Helpers**

Зареждане на проект

1. Отворете **Visual Studio 2022** (или по-ново)
2. Заредете **Solution**-а създаден в предишното упражнение.

Допълване на проект Welcome

1. Добавете следните частни променливи и публични свойства в класа **User**:
 - a. Променлива **id** и свойство **Id** от тип **int**
 - b. Променлива **expires** и свойство **Expires** от тип **DateTime**

Разширяване на проекта WelcomeExtended

Оттук надолу целият код който ще добавяме ще бъде в **WelcomeExtended** проекта.

Създаване на папки

1. От **Solution Explorer** избираме проекта **WelcomeExtended**
2. Кликаме с десен бутон върху **WelcomeExtended**, след което избираме **Add → New Folder**
3. Създаваме следните папки: **Data**

Създаване на клас UserData

1. Кликнете с десен бутон мишката на папката **Data**
2. От контекстното меню изберете **Add → New Item...**
3. От новоотвореният прозорец изберете **Class**
4. В полето **Name**, въведете за име **UserData.cs**
5. Добавяме частно поле **_users** от тип **List<User>**

List в C# е динамичен списък от елементи от определен тип, който може да се променя по време на изпълнение на програмата. Това означава, че може да добавяме, премахваме и да променяме елементите в списъка, без да е необходимо да дефинираме размера му предварително. List-ът е много полезен, когато трябва да се работи с големи колекции от данни и когато не е известен броят на елементите предварително. Като част от стандартната библиотека на C#, той предоставя множество методи за манипулиране на списъка, като например добавяне на елементи, търсене на елементи по индекс, премахане на елементи и много други.

6. Добавяме частно поле **_nextId** от тип **int**

Тъй като идеята на упражнението е да симулираме данни които се запамятават в база данни, трябва да симулираме автоматичната инкрементация на Primary Key-а при базата данни. Това поле ще съдържа следващият свободен индекс.

7. Създаваме конструктор в който задаваме първоначалните данни с които започваме, а именно за **_nextId** стойността 0 и **_users** трябва да бъде нов списък.
8. Създаваме метод **AddUser**, чийто входящ параметър е **User user**, а в тялото на метода задаваме **user.Id** да е равно на **_nextId++**, както и да добавим потребителя в колекцията **_users**
9. Създаваме метод **DeleteUser**, който приема като входящ параметър **User user**, и след това премахваме даденият потребител от колекцията **_users**
10. Създаваме следните три метода:



```
1 public bool ValidateUser(string name, string password)
2 {
3     foreach (var user in _users)
4     {
5         if (user.Name == name && user.Password == password)
6         {
7             return true;
8         }
9     }
10    return false;
11 }
12
13 public bool ValidateUserLambda(string name, string password)
14 {
15     return _users.Where(x => x.Name == name && x.Password == password)
16                   .FirstOrDefault() != null ? true : false;
17 }
18
19 public bool ValidateUserLinq(string name, string password)
20 {
21     var ret = from user in _users
22               where user.Name == name && user.Password == password
23               select user.Id;
24     return ret != null ? true : false;
25 }
```

Тези три метода илюстрират една и съща логика имплементирани с обикновен **foreach**, използвайки **Lambda** синтаксиса и **Linq**

11. Използвайки познанията до тук, трябва да направите следните методи

- a. GetUser – който ще връща даден потребител по зададени **потребителско име** и **парола** – с LINQ
- b. SetActive – който по входящи параметри **потребителско име** и **валидна дата**, задава датата като Expires стойност – с LINQ
- c. AssignUserRole – по входящи параметри **потребителско име** и **Роля**, да зададе на даденият потребител дадената роля. – с LINQ

Създаване на Клас UserHelper

1. В папката **Helpers** създайте още един клас – **UserHelper**
2. Променете класа да бъде **static**
3. Добавете нов статичен метод **ToString**, който приема като входящ параметър **this User user**.

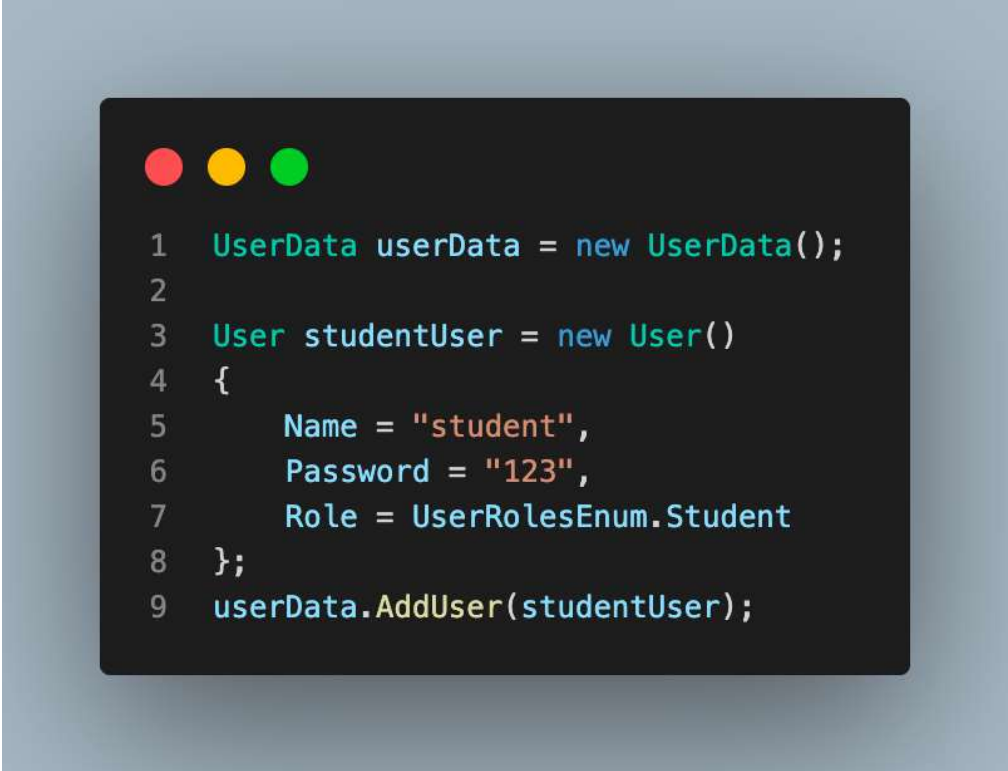
По този начин правим Extension метод на класа **User**.

*Идеята на **this User user** е това, че когато имаме един обект от тип **User**, ние ще добавим нов негов метод, който ще се казва **ToString()**. Това често се използва за да пренапишем стандартни методи включени в **.NET**.*

4. По аналог да се добавят два нови Extension хелпър метода на класа **UserData**. Тези два метода приемат като параметри: **this обекта** към който ще бъде приложен метода, **string name** и **string password**.
 - a. ValidateCredentials – в този метод ще проверяваме дали имаме въведени name и password и ако са празни трябва да върнем грешка **“The {field} cannot be empty”**, след което извикваме **ValidateUser** метода на колекцията която надграждаме.
 - b. GetUser – В този метод използваме **GetUser** метода на колекцията която надграждаме с параметрите и го връщаме като стойност

Навързване на кода

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



```
1  UserData userData = new UserData();
2
3  User studentUser = new User()
4  {
5      Name = "student",
6      Password = "123",
7      Role = UserRolesEnum.Student
8  };
9  userData.AddUser(studentUser);
```

2. По аналог добавете още 3 нови потребителя, със следните данни:
 - a. Student2 / 123 / Student
 - b. Teacher / 1234 / Professor
 - c. Admin / 12345 / Administrator
3. Изведете съобщение за подкана на потребителя да въведе име и парола след което използвайте метода **ValidateCredentials** на колекцията с потребителите за да проверите дали потребителя съществува и ако съществува вземете потребителя с **GetUser** и визуализирайте данни за потребителя посредством **ToString** Helper метода, ако потребителя не съществува хвърлете грешка „Потребителят не е намерен“