Упражнение №3 по ПС

Слой на логиката и данните. Предаване на данни между тях. Работа с бази данни.

Целта на упражнението е студентите да се запознаят с базите данни и създаването на връзка с тях в една десктоп система. Студентите трябва да могат да създават собствена валидация и да я прилагат. Студентите ще се запознаят със създаването на MS SQL база данни и таблици в нея и LINQ-to-SQL, за да реализират потребителски достъп до разработваната информационна система.

Get и Set на свойствата.

В упражнение 2 се наложи да напишем една собствена член променлива на класа MainForm, която да запазва статуса на потребителя, ползващ нашата програма. Езикът C# предоставя съкратен запис на Get и Set методите за достъп до това поле, а именно:

```
public UserStatus userStatus {get; private set;}
```

Дефинираните от програмиста свойства се използват по същия начин, както и всички стандартни свойства на контролите. Името на свойството може да стои както от ляво така и от дясно на оператор за присвояване (=).

Валидация в слоя на логиката.

1. Отворете проекта си от предния час.
2. Добавете валидация на въведените от потребителя име и парола. Валидацията да поддържа проверка за празни полета и дължина на паролата поне 6 символа.
   a. В папката Logic добавете нов клас с име **LoginValidation** (Десен бутон → Add Class...)
   b. Валидацията ще се извършва върху два стринга, затова добавяме 2 член променливи, които ще инициализираме в конструктура:

```
private string _username;
private string _password;

public LoginValidation(string username, string pass)
{
    _username = username;
    _password = pass;
    errText = string.Empty;
}
```

   c. Добавете поле за запаметяване на възникналите грешки. То трябва да

има възможност да се променя само в този клас, а да се чете и отвън.

```
public string errText { get; private set; }
```

d. Напишете private методи за проверка дали поле (стринговете на класа) е празно и дали дължината на въведеното е поне 6 символа.
e. Напишете общ публичен метод за валидация, към който да се обръща логин формата. Той трябва да извиква методите от точка d. и в зависимост от резултатите трябва да задава текст на възникналите грешки, подходящ за извеждане на потребителя. За унифициране на имената кръстете метода `public bool ValidateUserInput()`.

3. Направете необходимите промени в класа LoginForm.cs, за да проведете валидацията при натискане на бутона Login.
   a. От Properties прозореца задайте на свойството DialogResult на бутона Login стойност **None**.
   b. Добавете метод прихващащ събитието Click на бутона Login.
   c. В новосъздадения метод добавете следния код:

```
LoginValidation lv =
 new LoginValidation(txtUserName.Text, txtPassword.Text);
if (lv.ValidateUserInput())
{
     this.DialogResult = DialogResult.OK;
     this.Close();
}
else
{
    MessageBox.Show(lv.errText);
    errText = String.Empty;
}
```

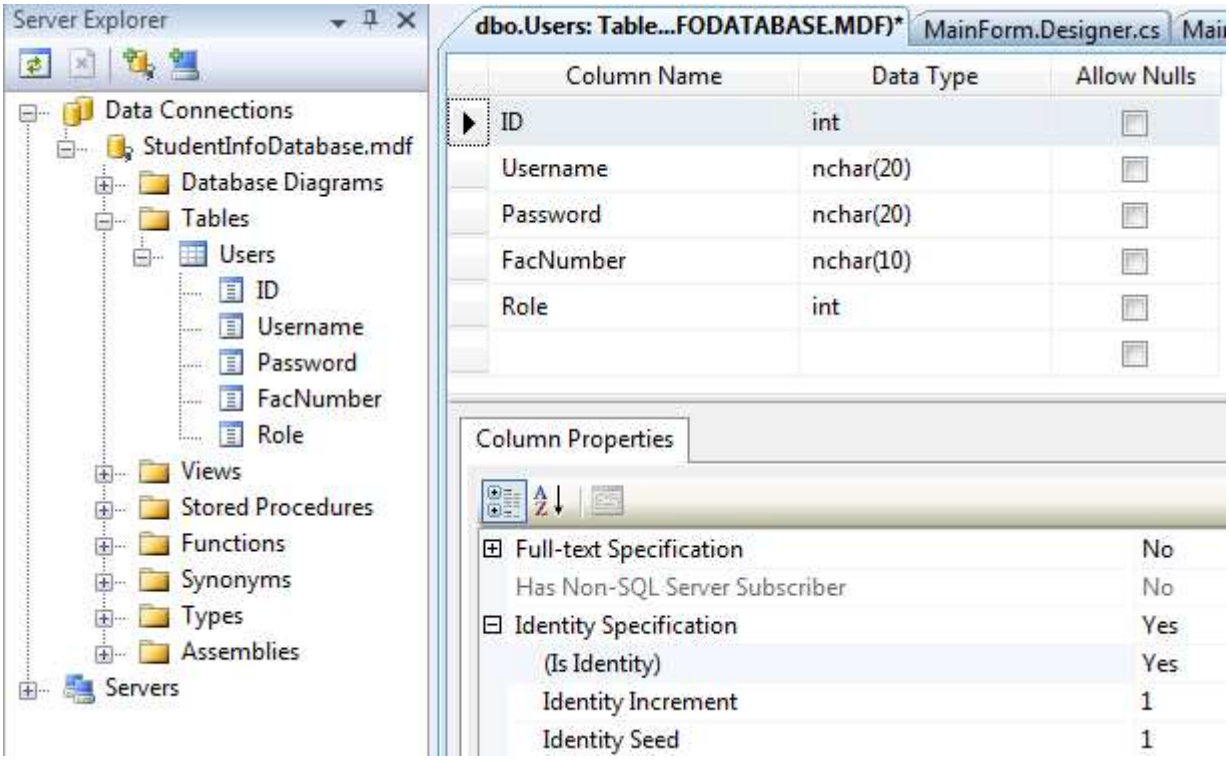
**този метод реализира изчакване на потребителя да въведе име и парола, отговарящи на изискванията. Единствено след като валидацията е премината успешно, формата се затваря.*

4. Компилирайте приложението. Ако има грешки – оправете ги преди да се продължи нататък. Стартирайте и разгледайте дали работи правилно.

Създаване на база данни с потребители.

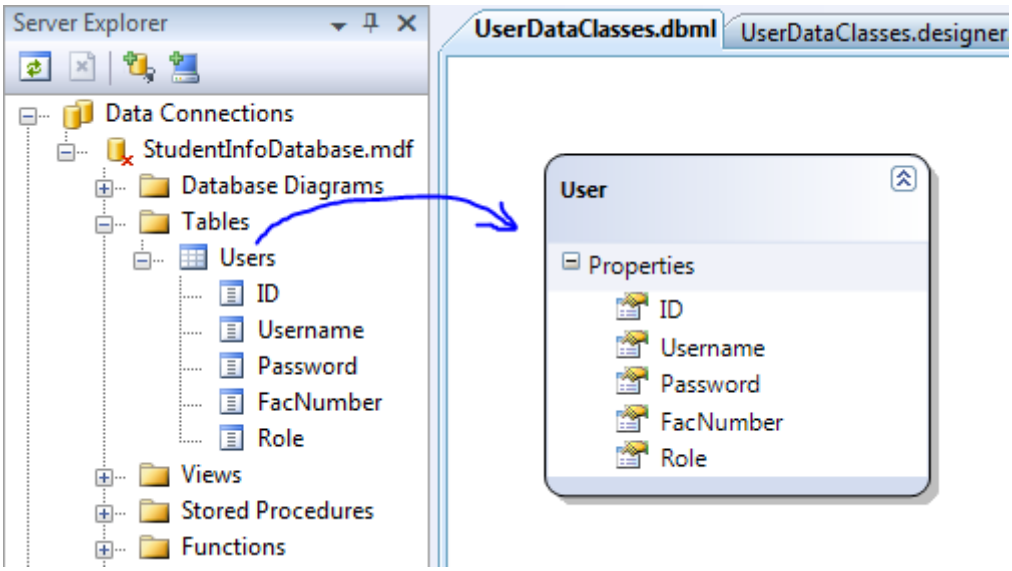
5. Към проекта добавете база данни. Кръстете я StudentInfoDatabase.mdf.
   **Добавянето става с десен бутон върху името на проекта в Solution Explorer → Add New Item, изберете от категория Data – Service-based Database.*
6. Базата си данни можете да управлявате през Server Explorer. Добавете таблица към базата данни и изадайте колони като показаните на Фиг. 1.
   **Добавянето става с десен бутон върху папка Tables → Add New Table.*

**В колоната Role ще записваме 1 за Студент, 2 за Преподавател, 3 – Администратор.*



Фиг. 1. *Колони на таблица Users.*

7. Запазете таблицата под името Users.
8. Добавете 1-2 потребителя към вашата база данни. С десен бутон върху името на Users на таблицата изберете Show Table Data. Тук можете ръчно да добавяте нови записи във вашата база.



Фиг. 2. *Автоматично генериран LINQ to SQL клас User.*

9. В папка Data добавете нов елемент: Add New Item → LINQ to SQL Classes, задайте му име: **UserDataClasses.dbml**
10. От Server Explorer провлачете таблицата **Users**. Така автоматично ще ви се създаде клас **User**, описващ данните във ващата таблица (виж. Фиг. 2).
11. Създайте клас **UserData.cs**, който ще съдържа методи за проверка на потребителите в базата данни (Десен бутон върху папката Data → Add Class...).
12. Създайте двойка публичен и private класове за проверка дали съществува потребител с дадено име и парола:

```
public static int IsUserPassCorrect(string username, string password)
{
    return UserData._IsUserPassCorrect(username, password);
}

private static int _IsUserPassCorrect(string username,string password)
{
    UserDataClassesDataContext dc = new UserDataClassesDataContext();
    var queryResult = (from users in dc.GetTable<User>()
                       where (users.Username == username )&&
                                                (users.Password == password)
                       select users).ToArray<User>();
    if(queryResult.Count<User>() > 0 )
        return queryResult.ElementAt<User>(0).Role;
    return -1;
}
```

13. В метода ValidateUserInput() добавете обръщение към горната проверка.

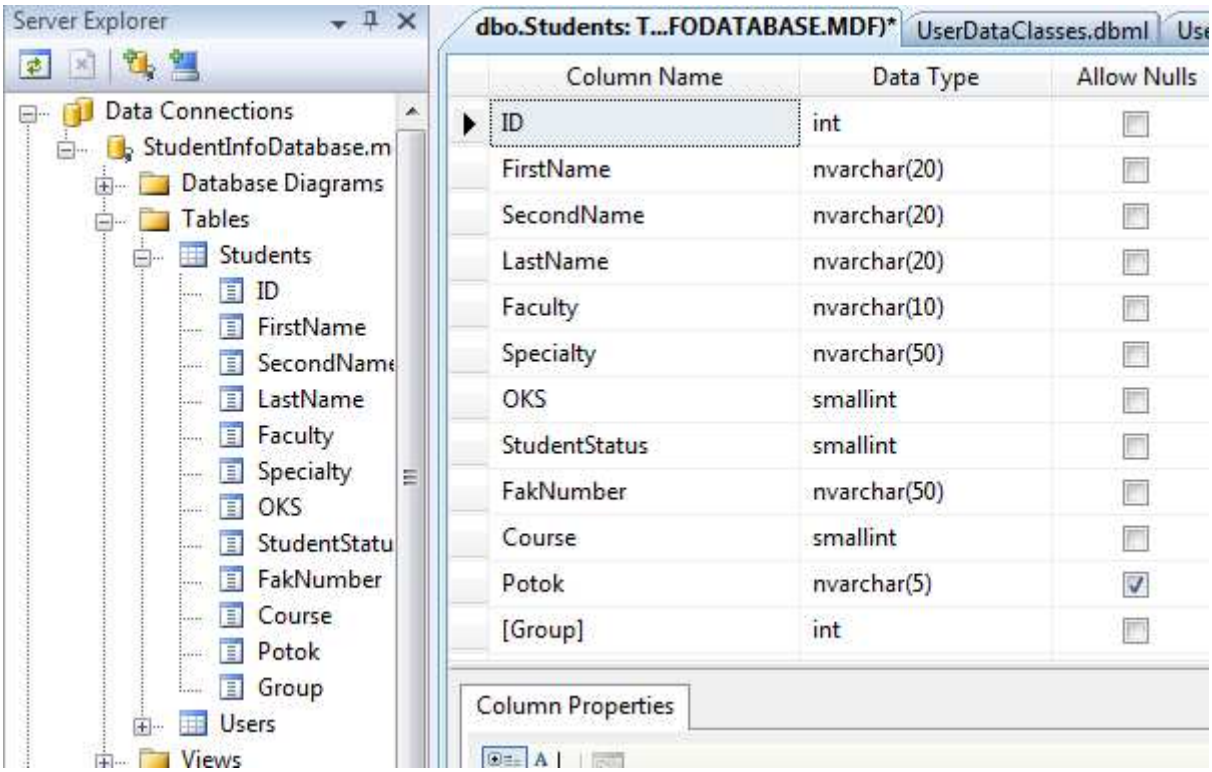
```
...
int queryResult = UserData.IsUserPassCorrect(_username, _password);
// returns the user role
if (queryResult < 1)
{
     errText += "Въвели сте грешно потребителско име или парола!";
     return false;
}
```

14. Компилирайте и тествайте.

За домашно:

В класа UserData създайте метод, който да проверява дали дадено потребителско име е заето.

Добавете нова таблица към базата ви данни с име Students, съдържаща полета като на фиг. 3.



Фиг. 3. *Колони на таблица Students.*

Важно: Запишете си направеното в часа на USB флаш памет и на следващото упражнение си я носете, за да продължите проекта.