Упражнение №2 по ПС

Трислойна архитектура на десктоп приложение. Слой на изгледа. Създаване и отваряне на форми.

Целта на това упражнение е студентите да придобият основни познания за многослойния модел на програмиране, да упражнят различните видове контроли, техните свойства и да се научат да отделят логически отделните части на подготвяното приложение. В упражнението се затвърждават знанията по създаване и извикване на методи.

Какво е многослойна архитектура и има ли почва у нас?

Многослойната архитектура цели разделяне кода на приложението на отделни части и за да направи по-лесна разработката, тестването и разширяването му. Използването на слоеве може да намали зависимостта между обектите, тъй като основното изискване при построяването на такава архитектура, е един слой да се свързва само с себе си и този под него. Тук ще разглеждаме главно трислойния модел състоящ се от слоевете “View”, “Logic” и “Data” (показани са на фигура 1). На български език слоевете се наричат още: „Презентационен”, „Бизнес логика” и „Слой на данните”.

Фиг. 1. *Схема на трислойна архитектура*.

Правила

- Всеки слой (n) може да достъпва (да създава обекти, да изпраща стойности) само към слоя директно под него (n-1).
- Всеки слой може да получава отговор (например true/false) от слоят под него (n-1), така че текущият слой (n) може да изпълни задачите си и при наличие на грешки.
- Един слой може да създава обекти от долния си слой и да записва информация

получена от негови методи, naprimer чрез return.

Какво съдържа всеки един от слоевете?

View

Всички форми заедно с техните GUI контроли се асоциират към този слой. В него се включва всичко свързано с презентационната информация за клиента, включително звук и видео.

Logic

Този слой е най-малко преносимият в системата. Причината е, че обикновено той е силно свързан със горния и долния слой. Той играе ролята на преход между презентационния слой и слоя на данните, затова се състои от процеси и операции.

Data

Записът, зареждането и всички подобни операции се носят от този слой. Този слой не е нужно да подлага на валидация данните подадени му от Бизнес логиката, но може да го уведомява за възникнали грешки при четене/запис. Този слой работи само с данните.

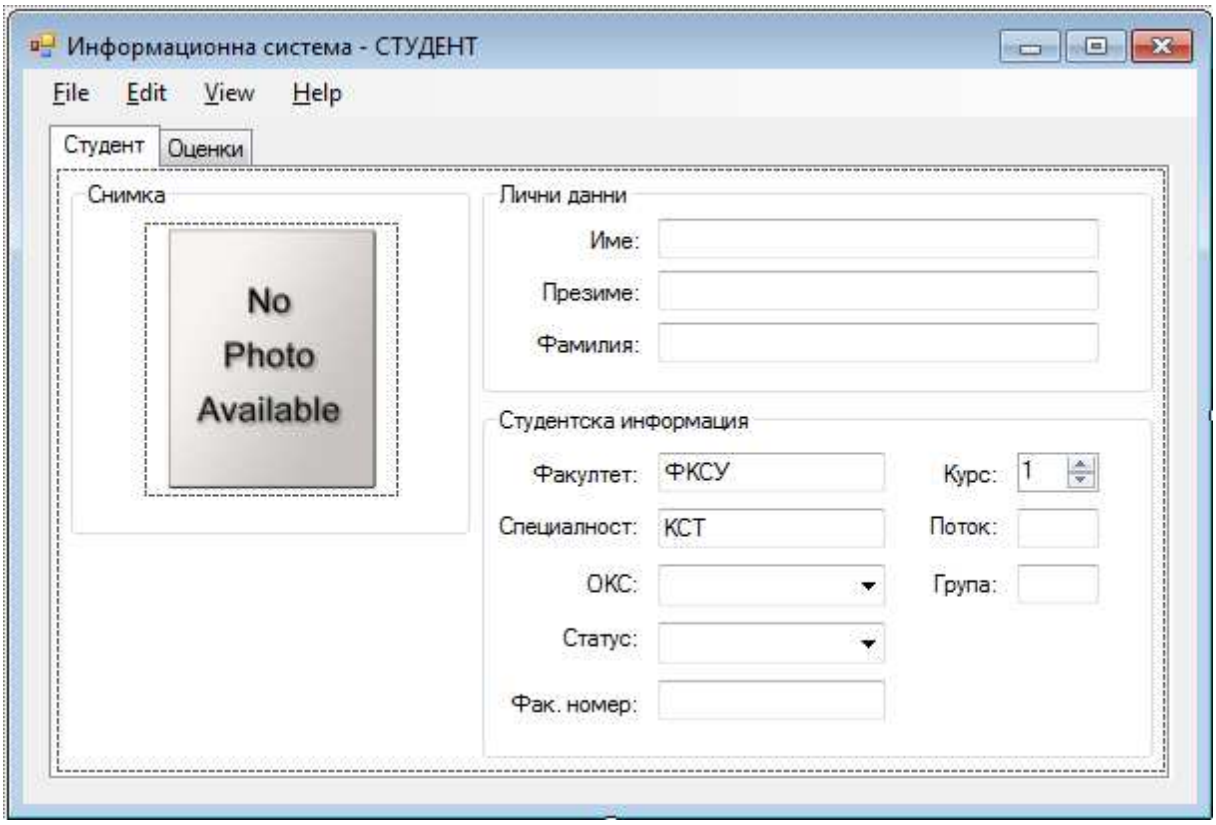
Пример

Представете си, че имате приложение, което позволява на потребител да се логне в система. Клиента въвежда своето потребителско име и парола в две текстови полета на логин формата и ги изпраща чрез натискане на бутон. Тогава следват следните примерни стъпки:

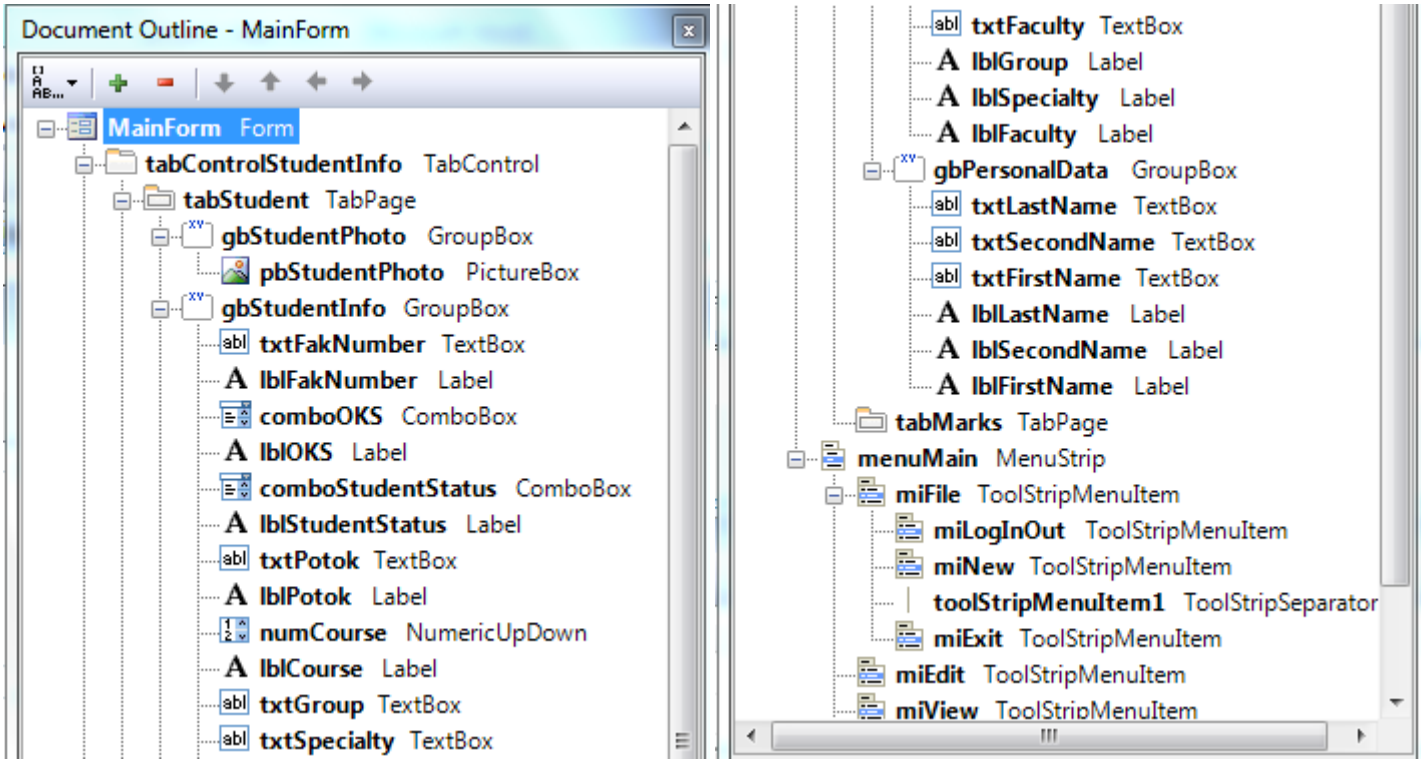
1. Слоят View предава стойностите от текстовите полета към Logic слоя за да бъдат валидирани (т.е. дали имат попълнен текст).
2. Обектът Logic може да създаде Data обект и да подаде, вече валидираните данни, за проверка в базата данни.
3. Когато слоя на Данните завърши всички свои процеси ще върне като резултат прочетените данни на Logic.
4. Логиката продължава работа, като довършва проверките за правилността на потребителския вход.
5. Ако обекта Logic е върнал съобщение „потребителят може да влезе”, презентационния слой позволява на потребителя да продължи и да достъпи функционалности, които преди са били забранени.

Създаване на слоя на изгледа на вашата програма.

1. Създайте нов проект с име StudentInfoSystem (Solution – PS_<gr>_<ime>). **Ако искате да работите в стария си Solution, отворете го и в Solution Explorer, натиснете десен бутон върху името на вашия Solution, от контекстното меню изберете Add →New Project... След като сте добавили проекта, трябва да му зададете следното свойство: отново десен бутон, но върху името му – Set as StartUp Project. В резултат трябва името му да е изписано удебелено.*
2. Изтрийте Form1.cs от проекта.
3. Добавете нова папка към проекта си с име View (Днес ще работим главно с нея). Добавете и още две папки за другите слоеве – Logic и Data.
4. В новата папка добавете нова форма – Add New Item → Windows Form. Дайте ѝ име MainForm.
5. Компилирайте (Build Solution). Ще ви се появи една грешка, тъй като изтрихме Form1, трябва да заменим извикването му с новата ни основна форма, а именно MainForm. MainForm се намира в друг файл и за да може да бъде намерен, трябва да се добави референция: `using StudentInfoSystem.View;`
6. Добавете меню към формата (MenuStrip). Добавете елементи на менюто и ги именувайте. За днешното упражнение менюто трябва да съдържа Help → About и Login/Logout (Изписаното име ще се сменя в зависимост от това дали има вписан потребител или не.)
7. Добавете TabControl. Първият таб ще съдържа основната информация за студента. Добавете необходимите контроли за въвеждане/извеждане на тази информация. Организирайте ги в GroupBox контроли. Може да взаимствате от показаната на Фиг. 2. форма. Спазвайте принципите за именуване на контролите. За по-лесно, след като поставите всяка една от тях, може да ги преименувате от прозореца Document Outline (отваря се на View →Other Windows → Document Outline). Списъците на контролите са показани на фигура на 3.



Фиг. 2. *Разположение на контролите върху MainForm.*



Фиг. 3. *Имена на използваните контроли.*

8. Създайте метод за :
   - изтриване на въведената информация в контролите (или връщане в началните им стойности).
     *Идеята на този метод е, че при извършване на операция LogOut трябва лесно да можем да „зачистим” всичката изведена от преди това информация.
   - Извеждане на информация за студент. Тази информация ще бъде, четена от база данни по-нататък и предавана като параметър на метода, затова напишете си коментар за това че има да се дописва четенето и задайте някакви статични стойности – например вашите данни. Този метод ще се извиква след успешен Login.
   - Направете метод, който прави контролите на формата неактивни. Идеята – когато потребителят е анонимен, той не може да работи с контролите. Този метод трябва да се грижи в менюто на формата да излиза като опция Login.
   - Направете и противоположния на горния метод – който активира всички контроли и в менюто се изписва възможността за LogOut.

До този момент не трябва да имате нито един метод прихващащ събитие. Ако имате такива, изтрийте ги и компилирайте за да изчистите грешките.

9. В папка Data добавете нов елемент – Add New Item → Code File. Дайте му име UserStatus.cs. В него сложете долния код. Това ще представляват видовете потребители на нашата програма. ADMIN е поставен, за да може при разрастване на програмата да се включи възможност само администратор (Преподавател) да може да променя данни. За момента ще третираме ADMIN и STUDENT по един и същи начин.

```
namespace StudentInfoSystem
{
    public enum UserStatus
    {
        ANONYMOUS, STUDENT, ADMIN
    }
}
```

10. Създайте метод, който да се извиква при избор на потребителя на бутона Login/Logout от менюто. Ако потребителят е анонимен да се променя състоянието му на студент (после ще добавим код за извикване на логин форма) и да се извикат метода за извеждане на информацията за студента.
    Ако потребителят е анонимен, да се извършат необходимите процедури по LogOut и промяна на потребителския статус.
    **Нещо тарикатско – ако напишете Switch и натиснете табулация ще ви се*

напише конструкцията на оператора. Заменете написаното в скобите с името на вашата променлива за статуса на потребителя и ще ви се опишат всички възможни клаузи case:, така че само трябва да добавите останалия си код.

Допълнителни форми. Извикване.

11. В папката View добавете нов файл, представляващ Windows Form и го именувайте LoginForm.cs. Намерете свойството ѝ, което определя позицията при отваряне. Задайте тя да се показва в центъра на извикващата я форма.
12. Добавете по две Label и TextBox контроли, съответно за потребителско име и парола. Направете необходимите промени на техните свойства.
13. Добавете 2 бутона: Login и Cancel. Задайте на свойството им DialogResult съответно стойности: OK и Cancel. Отговор OK ще означава, че има такъв потребител и той е въвел правилна парола. Проверките ще добавим в другите слоеве следващите упражнения.
14. За да извикате прозореца за login трябва да създадете обект от класа на вашата форма (LoginForm). За новия обект извикайте метода ShowDialog(). Съобразете върнатия от него резултат (например ако е върнато Cancel не трябва да се променя нищо).
15. Добавете още една форма във вашата View папка, но този път специалният AboutBox.cs. Задайте на бутона ѝ OK да затваря формата. Разгледайте кода на формата, забележете че текстовете, които се извеждат се вземат от файла AssemblyInfo.cs. Променете ги там.

За домашно оправете дизайна на формите и довършете всичко по заданието, което сте пропуснали.

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