Лабораторная работа № 7

Тема: Разработка реляционной базы данных в среде Visual Studio на языке C#
Цель работы: Разработка базы данных в среде Visual Studio на языке C#
Оборудование: IBM – совместимые компьютеры.
Место проведения: Компьютерный класс.
Техника безопасности: См. инструкцию.

Теоретическое описание

Свойство в Си-шарп – это член класса, который предоставляет удобный механизм доступа к полю класса (чтение поля и запись). Свойство представляет собой что-то среднее между полем и методом класса. При использовании свойства, мы обращаемся к нему, как к полю класса, но на самом деле компилятор преобразовывает это обращение к вызову соответствующего неявного метода. Такой метод называется аксессор (*accessor*). Существует два таких метода: *get* (для получения данных) и *set* (для записи). Объявление простого свойства имеет следующую структуру:

```
[модификатор доступа] [тип] [имя_свойства]
{
  get
  {
    // тело аксессора для чтения из поля
  }

  set
  {
    // тело аксессора для записи в поле
  }
}
```

Ход работы

1. Создание проекта

Запустить MS Visual Studio. Создать приложение по шаблону Windows Forms Application.

Создание нового проекта осуществляется командой

File->New Project...

В окне «New Project» задаются следующие настройки (рисунок 2):

- в поле «Installed Templates» нужно выбрать «Visual C#»;
- в перечне шаблонов Visual C# выбирается Windows Forms Application;
- назвать проект (Например Simple 2)

2. Создание базы

В контекстном меню выбрать «Открыть папку в проводнике» (Рисунок 1) . Перейти в папку Bin-Debug и создать базу данных Access. Пересохранить базу в формате Test1.mdb. В базе данных создать таблицу Person

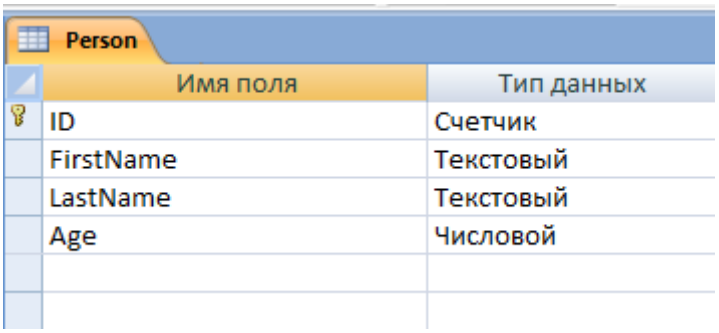

Рисунок 1

Сохраняем и закрываем базу данных.

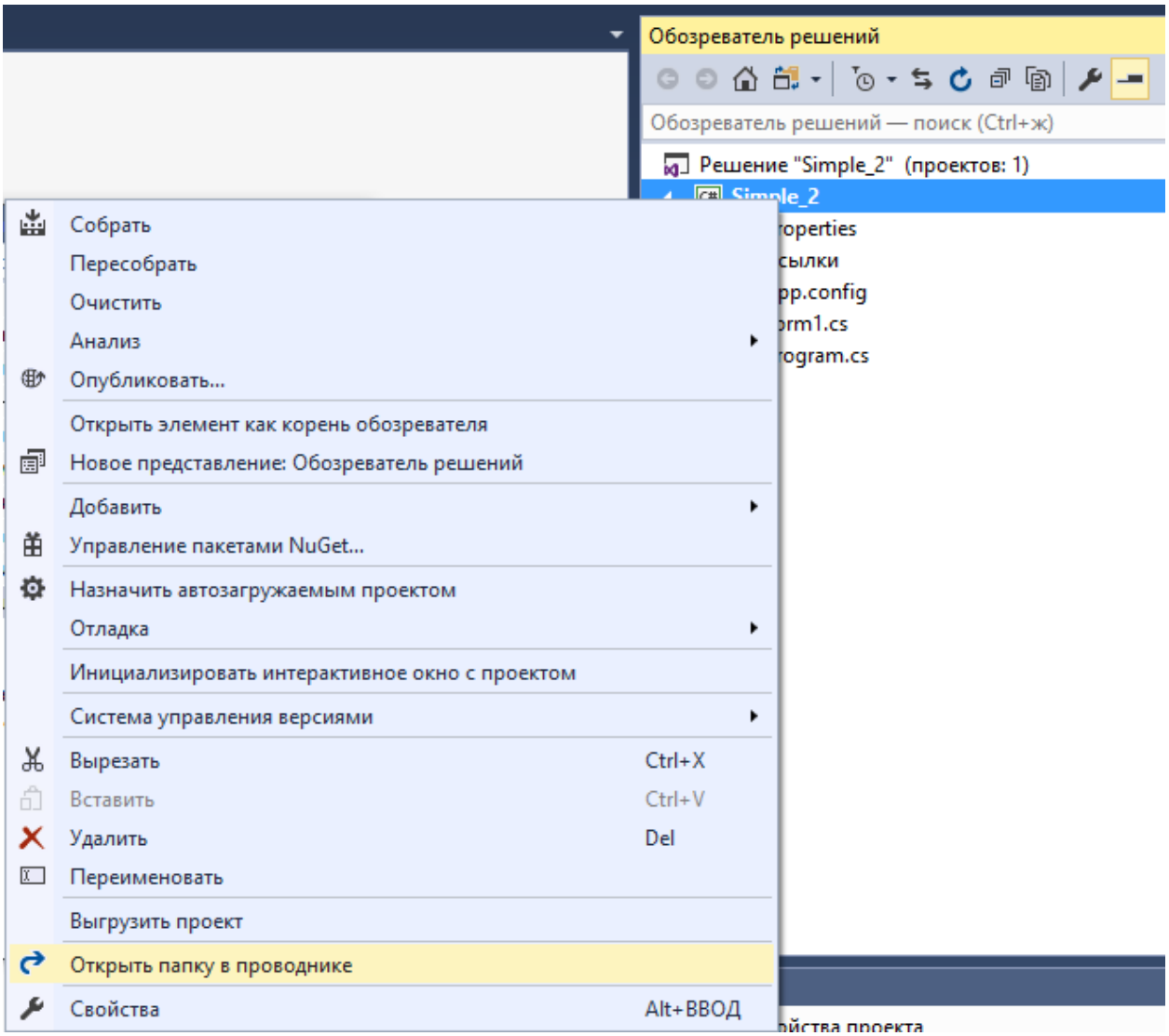

Рисунок 2

3. Создадим пакет для проекта

Выбрать в контекстном меню проекта Добавить – Создать папку (Рисунок 3)

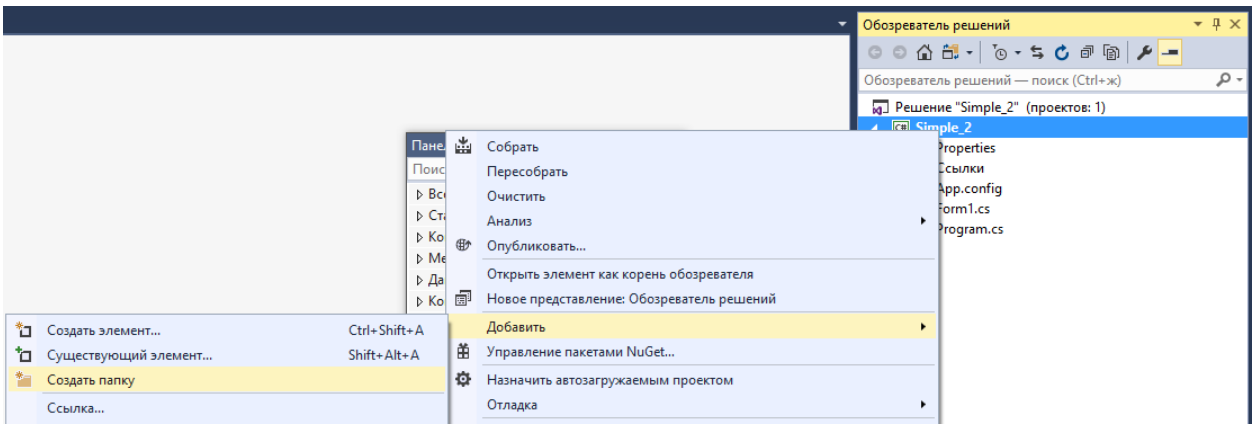

Рисунок 3

Задать название Controller (Рисунок 4).

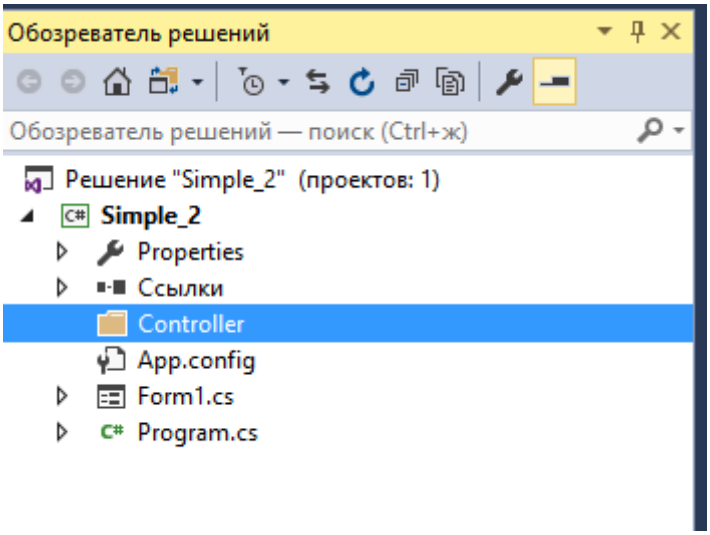

Рисунок 4

4. Создание класса.

Выбрать в контекстном меню пункт Класс (Рисунок 5)

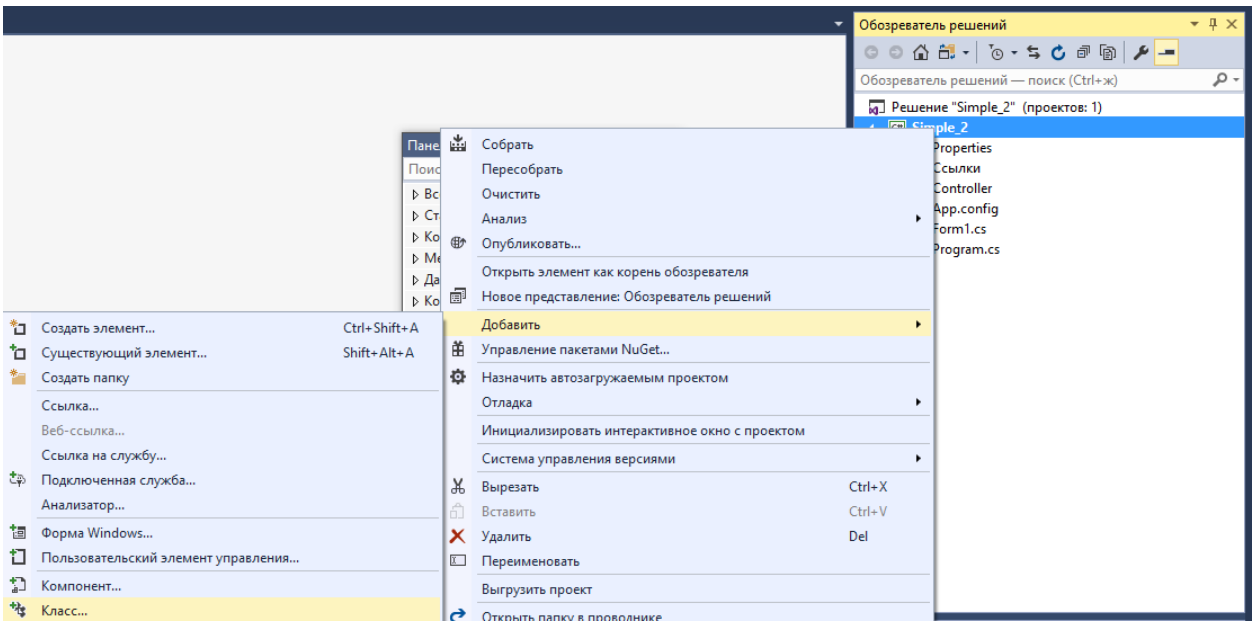

Рисунок 5

Задать классу имя ConnectionString.cs

В классе определим статическое поле **ConnStr**

```
namespace Simple_2
{
    class ConnectionString
    {
        public static string ConnStr

    }
}
```

Внутри поля определим аксессор get и вставляем ссылку из контекстного меню Ссылки-Добавить ссылку (Рисунок 6)

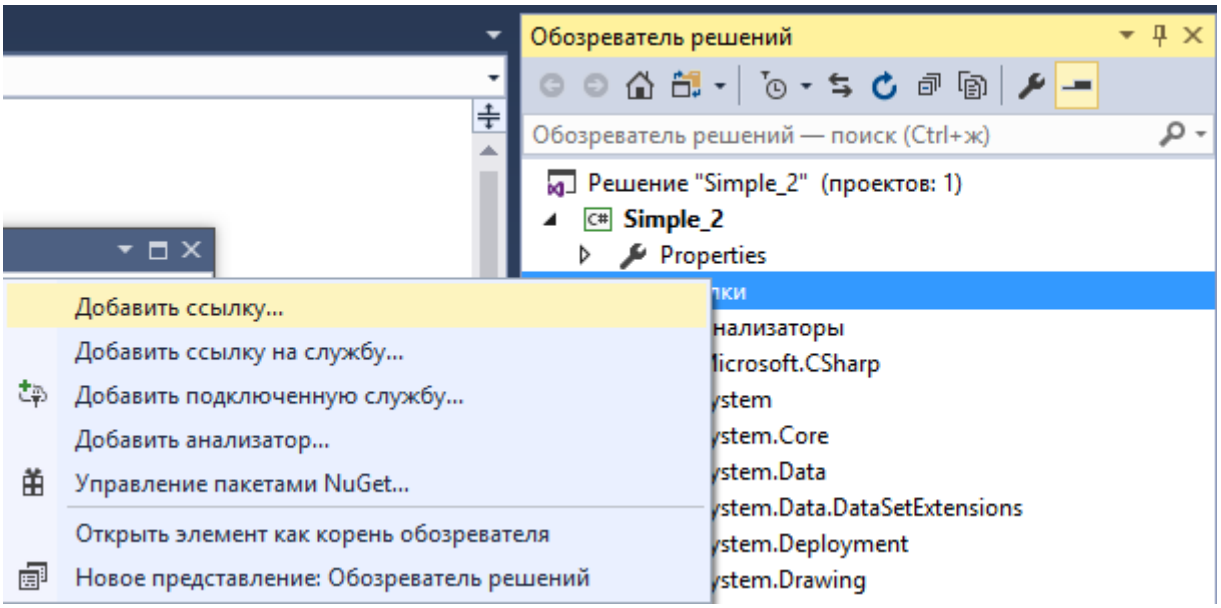

Рисунок 6

В открывшемся окне выбрать System.Configuration (Рисунок 7) для версии VS 2017 и выше

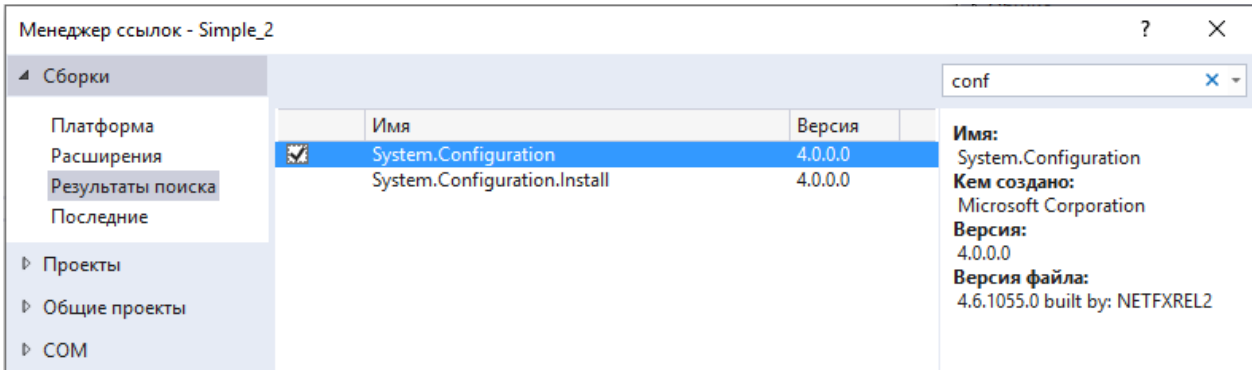

Рисунок 7

Для версии младше VS 2017 рисунок 8

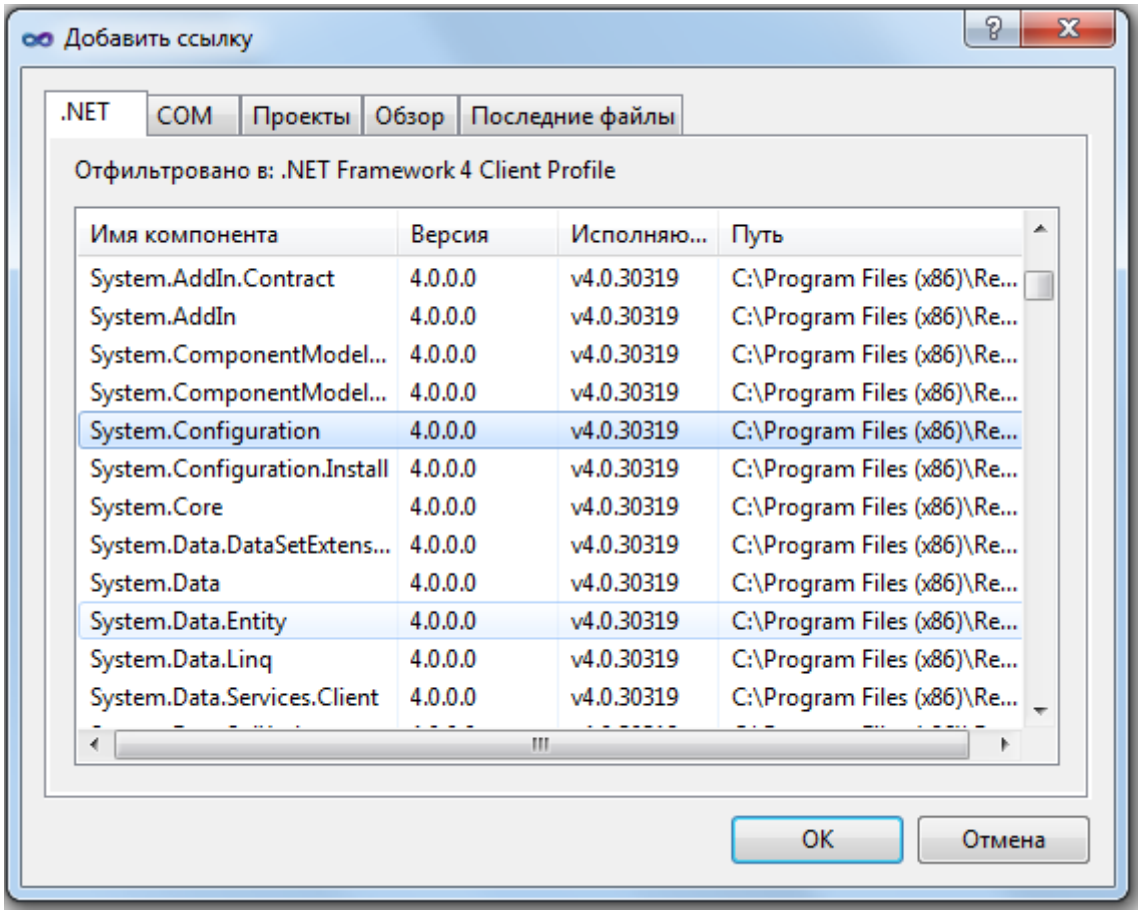

Рисунок 8

Добавить в модули `using System.Configuration;`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration;
```

В статическое поле добавляем

```
public static string ConnStr
        {
            get
            {
                return ConfigurationManager.ConnectionStrings[].ConnectionString;
            }
        }
```

Создадим строку подключения. Выберем в контекстом меню проекта **Свойства** затем **Параметры**.

Задаем имя ConnStr, тип – Строка подключения, Область видимости – Приложение.

Для установки параметров Значения выбрать **(…)** и в окне Свойства подключения выбрать Источник данных - Файл базы данных Microsoft Access (OLE DB).

В строке Имя файла базы данных указать место расположения базы данных Test1.mdb (Разработанную базу)

Проверить подключение.

Убрать маршрут к файлу и оставить только имя файла (Рисунок 9)

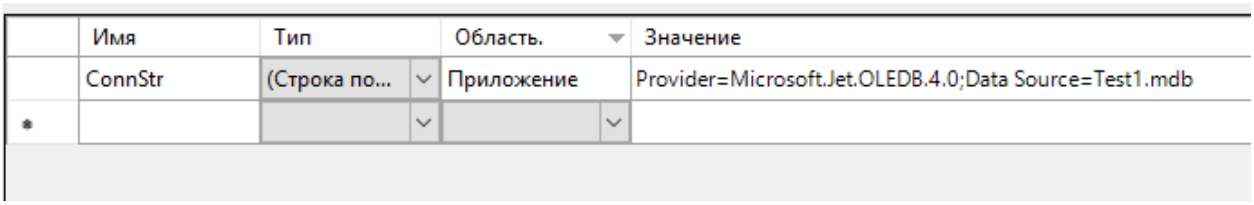

Рисунок 9

В файле конфигурации приложения скопировать путь к файлу `Simple_2.Properties.Settings.ConnStr` и вставим его в статическую переменную

Получиться такой результат

```
namespace Simple_2
{
    class ConnectionString
    {
        public static string ConnStr
        {
            get
            {
                return
ConfigurationManager.ConnectionStrings["Simple_2.Properties.Settings.ConnStr"].Connection
String;
            }
        }

    }
}
```

Создадим еще один класс Query (Рисунок 10).

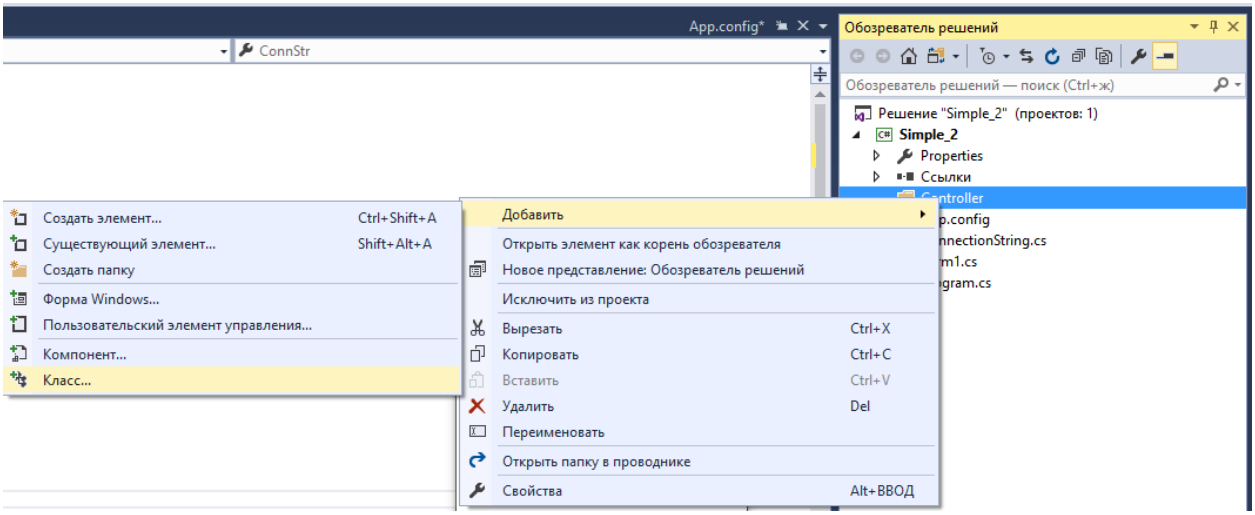

Рисунок 10

В этом классе определим объекты для выполнения запросов. Но сначала подключим базы данных

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb; //Подключение базы данных
using System.Data;

namespace Simple_2.Controller
{
    class Query
    {
        OleDbConnection connection;
        OleDbCommand command;
        OleDbDataAdapter dataAdapter;
        DataTable bufferTable;

        public Query (string Conn)
        {
            connection = new OleDbConnection(Conn);
            bufferTable = new DataTable();
        }
        public DataTable upDatePerson() //обновление данных
        {
            connection.Open();
            dataAdapter = new OleDbDataAdapter("SELECT * FROM Person", connection);
            //очистка данных
            bufferTable.Clear();
            // заполнение таблицы
            dataAdapter.Fill(bufferTable);
            //закрываем соединение
            connection.Close();
            //возвращение результата
            return bufferTable;
        }
        //метод добавления
        public void Add(string FirstName, string LastName, int Age)
        {
            connection.Open(); //открыть соединение
            command = new OleDbCommand("INSERT INTO Person (FirstName, LastName, Age)
VALUES(@FirstName, @LastName, Age)", connection);
            command.Parameters.AddWithValue("FirsName", FirstName);
```

```
            command.Parameters.AddWithValue("LastName", LastName);
            command.Parameters.AddWithValue("Age", Age);
            command.ExecuteNonQuery();
            connection.Close(); // закрыть соединение
        }
        //метод удаления записи
       public void Delete(int ID)
        {
            connection.Open();
            command = new OleDbCommand($"DELETE FROM Person WHERE ID = {ID}",
connection);
            command.ExecuteNonQuery();
            connection.Close();
               }
    }

}
```

Установите на форму компонент DataGridView для отображение данных в таблице, кнопки (Button) для добавления, обновления, удаления данных и для ввода данных используем компонент textBox. В результате форма приложения получит вид (Рисунок11).

Сверху над каждым компонентом textBox установить компоненты Label и задать описание вводимых данных (Фамилия, Имя, Возраст).

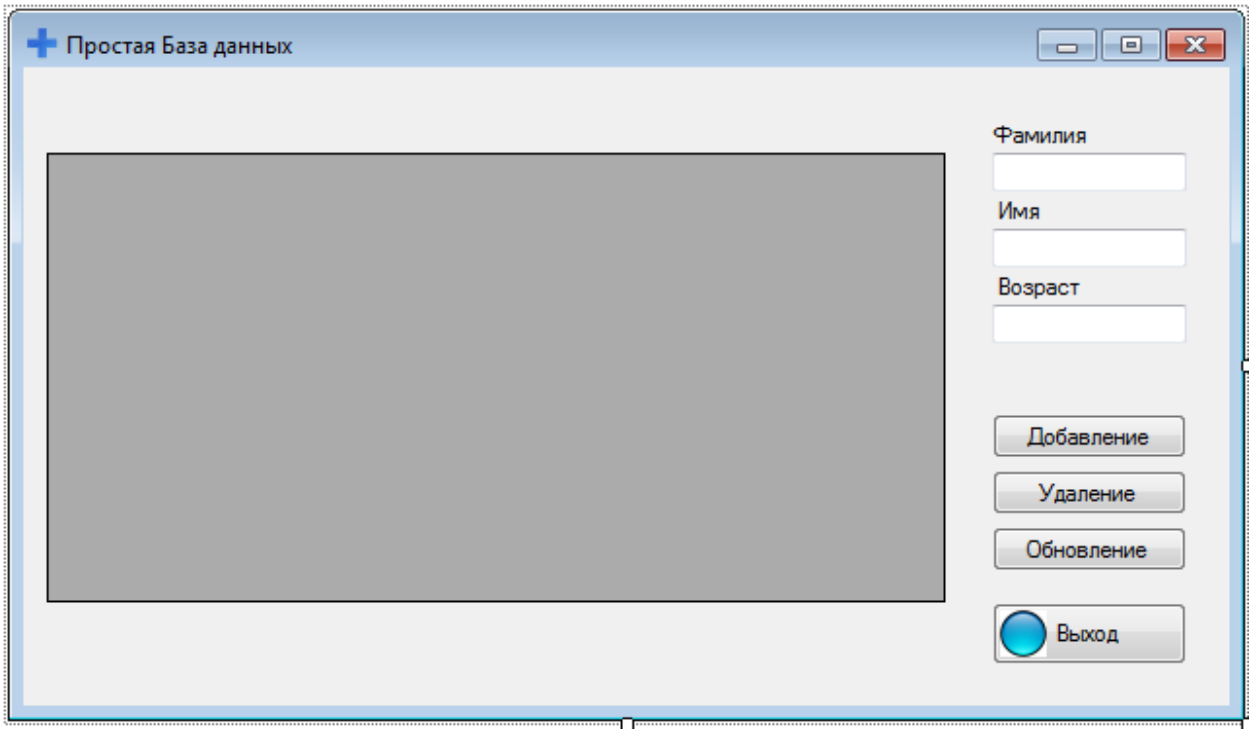

Рисунок 11

Добавим `using Simple_2.Controller;`

Добавляем **Query controller** в код, добавляем строку подключения, события обновления, добавления, удаления данных.

После этого код примет следующий вид

```
namespace Simple_2
{
    public partial class Form1 : Form
    {
        Query controller;
        public Form1()
        {
            InitializeComponent();
            controller = new Query(ConnectionString.ConnStr); //строка подключения
        }

        private void button1_Click(object sender, EventArgs e)
        {
            dataGridView1.DataSource = controller.upDatePerson(); // обновление данных
        }

        private void button2_Click(object sender, EventArgs e)
        {
            controller.Add(textBox1.Text, textBox2.Text, int.Parse(textBox3.Text)); //
добавление данных
        }

        private void button3_Click(object sender, EventArgs e)
        {

controller.Delete(int.Parse(dataGridView1.Rows[dataGridView1.CurrentRow.Index].
Cells["ID"].Value.ToString()));
        }
    }
}
```

Проверьте работоспособность программы.

После операций добавления и удаления, необходимо выполнять обновления для просмотра результата работа программы.

Задания

1. Добавить кнопку для закрытия приложения
2. Установить необходимое свойство для размещения формы по центру экрана после запуска приложения
3. Отображать данные на форме в таблице после запуска приложения
4. Изменить код программы, чтобы после ввода данных отображались введенные данные в таблице.
5. Установить поведение таблицы – только для чтения.
6. При вводе возраста ограничить ввод только цифровых значений