Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Таныгин Максим Олегович
Должность: и.о. декана факультета фундаментальной и прикладной информатики
Дата подписания: 21.09.2023 12:55:38
Уникальный программный ключ:
65ab2aa0d384efe8480e6a4c688eddbc475e411a

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
О.Г.Локтионова
«__» _____ 2017 г.

Создание приложения Windows Forms в среде Visual Studio Community 2017

Методические указания к лабораторным работам
по курсу «Языки объектно-ориентированного
программирования»

Курск 2017

УДК 681.3

Составитель Е.И.Аникина

Рецензент
Кандидат технических наук, доцент кафедры программной инженерии *Н.Н. Бочанова*

Создание приложения Windows Forms в среде Visual Studio Community 2017: методические указания к лабораторным работам по курсу «Языки объектно-ориентированного программирования»/Юго-Зап. гос. ун-т; сост. Е.И.Аникина. Курск, 2017. 13 с.

Содержит теоретические сведения и задания для выполнения лабораторных работ по изучению технологии создания и отладки оконных приложений Windows Forms в интегрированной среде разработки Visual Studio Community 2017.

Предназначено для студентов направлений подготовки, входящих в группы «Компьютерные и информационные науки» и «Информатика и вычислительная техника».

Текст печатается в авторской редакции.

Подписано в печать . Формат 60x84 1/16.
Усл. печ. л. . Уч.-изд. л. . Тираж 100 экз. Заказ . Бесплатно.

Юго-Западный государственный университет
305040, Курск, ул.50 лет Октября, 94.

Создание графического приложения

Для создания графических интерфейсов с помощью платформы .NET применяются разные технологии - Window Forms, WPF, приложения для магазина Windows Store (для ОС Windows 8/8.1/10). Однако наиболее простой и удобной платформой до сих пор остается Window Forms или формы. Данное руководство ставит своей целью дать понимание принципов создания графических интерфейсов с помощью технологии WinForms и работы основных элементов управления.

Для создания графического проекта нам потребуется среда разработки Visual Studio. Поскольку наиболее распространенная пока версия Visual Studio 2013, то для данного руководства я буду использовать бесплатную версию данной среды **Visual Studio Community 2013** которую можно найти на странице https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx.

После установки среды и всех ее компонентов, запустим Visual Studio и создадим проект графического приложения. Для этого в меню выберем пункт File (Файл) и в подменю выберем **New - > Project** (Создать - > Проект). После этого перед нами откроется диалоговое окно создания нового проекта:

В левой колонке выберем **Windows Desktop**, а в центральной части среди типов проектов - тип **Windows Forms Application** и дадим ему какое-нибудь имя в поле внизу. Например, назовем его *HelloApp*. После этого нажимаем OK.

После этого Visual Studio откроет наш проект с созданными по умолчанию файлами:

Большую часть пространства Visual Studio занимает графический дизайнер, который содержит форму будущего приложения. Пока она пуста и имеет только заголовок Form1. Справа находится окно файлов решения/проекта - Solution Explorer (Обозреватель решений). Там и находятся все связанные с нашим приложением файлы, в том числе файлы формы *Form1.cs*.

Внизу справа находится окно свойств - Properties. Так как у меня в данный момент выбрана форма как элемент управления, то в этом поле отображаются свойства, связанные с формой.

Теперь найдем в этом окне свойство формы Text и изменим его значение на любое другое:

Таким образом мы поменяли заголовок формы. Теперь перенесем на поле какой-нибудь элемент управления, например, кнопку. Для этого найдем в левой части Visual Studio вкладку **Toolbox (Панель инструментов)**. Нажмем на эту вкладку, и у нас откроется панель с элементами, откуда мы можем с помощью мыши перенести на форму любой элемент:

Найдем среди элементов кнопку и, захватив ее указателем мыши, перенесем на форму:

Это визуальная часть. Теперь приступим к самому программированию. Добавим простейший код на языке C#, который бы выводил сообщение по нажатию кнопки. Для этого мы должны перейти в файл кода, который связан с этой формой. Если у нас не открыт файл кода, мы можем нажать на форму правой кнопкой мыши и в появившемся меню выбрать View Code (Посмотреть файл кода):

Однако воспользуемся другим способом, чтобы не писать много лишнего кода. Наведем указатель мыши на кнопку и щелкнем по ней двойным щелчком. Мы автоматически попадаем в файл кода *Form1.cs*, который выглядит так:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HelloApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

              private void button1_Click(object sender,
EventArgs e)
        {

        }
    }
}
```

Добавим вывод сообщения по нажатию кнопки, изменив код следующим образом:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace HelloApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

              private void button1_Click(object sender,
EventArgs e)
        {
            MessageBox.Show("Привет");
        }
    }
}
```

Запуск приложения

Чтобы запустить приложение в режиме отладки, нажмем на клавишу F5 или на зеленую стрелочку на панели Visual Studio. После этого запустится наша форма с одинокой кнопкой. И если мы нажмем на кнопку на форме, то нам будет отображено сообщение с приветствием.

После запуска приложения студия компилирует его в файл с расширением exe. Найти данный файл можно, зайдя в папку проекта и далее в каталог bin/Debug или bin/Release

Рассмотрев вкратце создание проекта графического приложения, мы можем перейти к обзору основных компонентов и начнем мы с форм.

Работа с формами

Внешний вид приложения является нам преимущественно через формы. Формы являются основными строительными блоками. Они предоставляют контейнер для различных элементов управления. А механизм событий позволяет элементам формы отзываться на ввод пользователя, и, таким образом, взаимодействовать с пользователем.

При открытии проекта в Visual Studio в графическом редакторе мы можем увидеть визуальную часть формы - ту часть, которую мы видим после запуска приложения и куда мы переносим элементы с панели управления. Но на самом деле форма скрывает мощный функционал, состоящий из методов, свойств, событий и прочее. Рассмотрим основные свойства форм.

Если мы запустим приложение, то нам отобразится одна пустая форма. Однако даже такой простой проект с пустой формой имеет несколько компонентов:

Несмотря на то, что мы видим только форму, но стартовой точкой входа в графическое приложение является класс Program, расположенный в файле *Program.cs*:

```
1    using System;
2    using System.Collections.Generic;
3    using System.Linq;
4    using System.Threading.Tasks;
5    using System.Windows.Forms;
6
7    namespace HelloApp
8    {
9        static class Program
10       {
11           [STAThread]
12           static void Main()
13           {
14               Application.EnableVisualStyles();
15               Application.SetCompatibleTextRenderingDefault(false);
16               Application.Run(new Form1());
17           }
18       }
19   }
```

Сначала программой запускается данный класс, затем с помощью выражения `Application.Run(new Form1())` он запускает форму Form1. Если вдруг мы захотим изменить стартовую форму в приложении на какую-нибудь другую, то нам надо изменить в этом выражении Form1 на соответствующий класс формы.

Сама форма сложна по содержанию. Она делится на ряд компонентов. Так, в структуре проекта есть файл *Form1.Designer.cs*, который выглядит примерно так:

```
1    namespace HelloApp
2    {
3        partial class Form1
4        {
5            /// <summary>
6            /// Required designer variable.
7            /// </summary>
8            private System.ComponentModel.IContainer components = null;
9
10           /// <summary>
11           /// Clean up any resources being used.
```

```
        /// </summary>
        /// <param name="disposing">true if managed resources sh
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.SuspendLayout();
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F,
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode
            this.ClientSize = new System.Drawing.Size(284, 261);
            this.Name = "Form1";
            this.Text = "Привет мир!";
            this.ResumeLayout(false);

        }

        #endregion

    }
}
```

Здесь объявляется частичный класс формы Form1, которая имеет два метода: `Dispose()`, который выполняет роль деструктора объекта, и `InitializeComponent()`, который устанавливает начальные значения свойств формы.

При добавлении элементов управления, например, кнопок, их описание также добавляется в этот файл.

Но на практике мы редко будем сталкиваться с этим классом, так как они выполняет в основном дизайнерские функции - установка свойств объектов, установка переменных.

Еще один файл - *Form1.resx* - хранит ресурсы формы. Как правило, ресурсы используются для создания однообразных форм сразу для нескольких языковых культур.

И более важный файл - *Form1.cs*, который в структуре проекта называется просто Form1, содержит код или программную логику формы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HelloApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

По умолчанию здесь есть только конструктор формы, в котором просто вызывается метод `InitializeComponent()`, объявленный в файле дизайнера *Form1.Designer.cs*. Именно с этим файлом мы и будем больше работать.

Основные свойства форм

С помощью специального окна Properties (Свойства) справа Visual Studio предоставляет нам удобный интерфейс для управления свойствами элемента:

Большинство этих свойств оказывает влияние на визуальное отображение формы. Пробежимся по основным свойствам:

1. **Name**: устанавливает имя формы - точнее имя класса, который наследуется от класса `Form`
2. **BackColor**: указывает на фоновый цвет формы. Щелкнув на это свойство, мы сможем выбрать тот цвет, который нам подходит из списка предложенных цветов или цветовой палитры
3. **BackgroundImage**: указывает на фоновое изображение формы
4. **BackgroundImageLayout**: определяет, как изображение, заданное в свойстве BackgroundImage, будет располагаться на форме.
5. **ControlBox**: указывает, отображается ли меню формы. В данном случае под меню понимается меню самого верхнего уровня, где находятся иконка приложения, заголовок формы, а также кнопки минимизации формы и крестик. Если данное свойство имеет значение false, то мы не увидим ни иконку, ни крестика, с помощью которого обычно закрывается форма
6. **Cursor**: определяет тип курсора, который используется на форме
7. **Enabled**: если данное свойство имеет значение false, то она не сможет получать ввод от пользователя, то есть мы не сможем нажать на кнопки, ввести текст в текстовые поля и т.д.
8. **Font**: задает шрифт для всей формы и всех помещенных на нее элементов управления. Однако, задав у элементов формы свой шрифт, мы можем тем самым переопределить его
9. **ForeColor**: цвет шрифта на форме
10. **FormBorderStyle**: указывает, как будет отображаться граница формы и строка заголовка. Устанавливая данное свойство в None можно создавать внешний вид приложения произвольной формы
11. **HelpButton**: указывает, отображается ли кнопка справки формы
12. **Icon**: задает иконку формы
13. **Location**: определяет положение по отношению к верхнему левому углу экрана, если для свойства `StartPosition`установлено значение `Manual`
14. **MaximizeBox**: указывает, будет ли доступна кнопка максимизации окна в заголовке формы
15. **MinimizeBox**: указывает, будет ли доступна кнопка минимизации окна
16. **MaximumSize**: задает максимальный размер формы
17. **MinimumSize**: задает минимальный размер формы
18. **Opacity**: задает прозрачность формы
19. **Size**: определяет начальный размер формы
20. **StartPosition**: указывает на начальную позицию, с которой форма появляется на экране
21. **Text**: определяет заголовок формы
22. **TopMost**: если данное свойство имеет значение `true`, то форма всегда будет находиться поверх других окон
23. **Visible**: видима ли форма, если мы хотим скрыть форму от пользователя, то можем задать данному свойству значение`false`
24. **WindowState**: указывает, в каком состоянии форма будет находиться при запуске: в нормальном, максимизированном или минимизированном

Программная настройка свойств

С помощью значений свойств в окне Свойства мы можем изменить по своему усмотрению внешний вид формы, но все то же самое мы можем сделать динамически в коде. Перейдем к коду, для этого нажмем правой кнопкой мыши на форме и выберем в появившемся контекстном меню View Code (Просмотр кода). Перед нами открывается файл кода *Form1.cs*. Изменим его следующим образом:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HelloApp
{
    public partial class Form1 : Form
    {
        public Form1()
          {
              InitializeComponent();
              Text = "Hello World!";
              this.BackColor = Color.Aquamarine;
              this.Width = 250;
              this.Height = 250;
        }
    }
}
```

В данном случае мы настроили несколько свойств отображения формы: заголовок, фоновый цвет, ширину и высоту. При использовании конструктора формы надо учитывать, что весь остальной код должен идти после вызова метода `InitializeComponent()`, поэтому все установки свойств здесь расположены после этого метода.

Установка размеров формы

Для установки размеров формы можно использовать такие свойства как Width/Height или Size. Width/Height принимают числовые значения, как в

вышеприведенном примере. При установке размеров через свойство Size, нам надо присвоить свойству объект типа Size:

```
this.Size = new Size(200,150);
```

Объект Size в свою очередь принимает в конструкторе числовые значения для установки ширины и высоты.

Начальное расположение формы

Начальное расположение формы устанавливается с помощью свойства `StartPosition`, которое может принимать одно из следующих значений:

1. **Manual**: Положение формы определяется свойством Location
2. **CenterScreen**: Положение формы в центре экрана
3. **WindowsDefaultLocation**: Позиция формы на экране задается системой Windows, а размер определяется свойством Size
4. **WindowsDefaultBounds**: Начальная позиция и размер формы на экране задается системой Windows
5. **CenterParent**: Положение формы устанавливается в центре родительского окна

Все эти значения содержатся в перечислении `FormStartPosition`, поэтому, чтобы, например, установить форму в центре экрана, нам надо прописать так:

```
this.StartPosition = FormStartPosition.CenterScreen;
```

Фон и цвета формы

Чтобы установить цвет как фона формы, так и шрифта, нам надо использовать цветовое значение, хранящееся в структуре Color:

```
this.BackColor = Color.Aquamarine;
this.ForeColor = Color.Red;
```

Кроме того, мы можем в качестве фона задать изображение в свойстве `BackgroundImage`, выбрав его в окне свойств или в коде, указав путь к изображению:

```
this.BackgroundImage = Image.FromFile("C:\Users\Eugene\Pictures\3332.jpg");
```

Чтобы должным образом настроить нужное нам отображение фоновой картинки, надо использовать свойство `BackgroundImageLayout`, которое может принимать одно из следующих значений:

1. **None**: Изображение помещается в верхнем левом углу формы и сохраняет свои первоначальные значения
2. **Tile**: Изображение располагается на форме в виде мозаики
3. **Center**: Изображение располагается по центру формы
4. **Stretch**: Изображение растягивается до размеров формы без сохранения пропорций
5. **Zoom**: Изображение растягивается до размеров формы с сохранением пропорций

Например, расположим форму по центру экрана:

```
  this.StartPosition                                    =
FormStartPosition.CenterScreen;
```