Урок 7

Введение в Windows Forms

Осваиваем графический интерфейс. «Сложное» ООП. Немного о наследовании, событиях и делегатах. Первое графическое приложение.

От консольного приложения к Windows Forms

Windows Forms — технология создания приложений под Windows. Существуют также другие технологии программирования под Windows, например, Windows Presentation Foundation или то же консольное приложение.

Технология создания приложений Windows Forms является одной из самых простых для понимания функционирования программ в Windows.

MessageBox.Show — вместо Console.WriteLine

Начнём знакомство с программированием под Windows со знакомства с методом Show класса MessageBox. В Windows Forms-приложениях он поможет выводить информацию о ваших переменных и организовывать диалоги.

```
 // Требуется добавить в проект ссылку на сборку System.Windows.Forms
using System.Windows.Forms;
class ShowForm
{
    public static void Main()
    {
        // MessageBox простейший способ вывода информации на экран
        // У метода Show большое количество перезагрузок, с помощью
        // которых можно управлять видом окна сообщения
        MessageBox.Show("Сообщение из Windows Forms!");
                MessageBox.Show("Сообщение из Windows Forms с заголовком",
"Заголовок");
           if (MessageBox.Show("Вы уверены, что хотите продолжить обучение?",
"Заголовок",
            MessageBoxButtons.OKCancel) == DialogResult.OK)
            System.Console.WriteLine("Вы нажали OK");
        else System.Console.WriteLine("Вы нажали Cancel");
        System.Console.WriteLine("Можно выводить сообщения в консоль");
           System.Console.WriteLine("Не стоит сейчас слишком задерживаться на
изучение MessageBox.Show. У вас еще будет такая возможность");
        System.Console.ReadKey();
    }
}
```

События и их обработчики

Событийная модель управления является в современном мире операционных систем преобладающей. Чтобы понять, как работает Windows-приложение, нужно познакомиться с событиями.

Рассмотрим пример обработки события от таймера.

```
using System;
using System.Timers;
class TimerEvent
```

```
{
    public static void Main()
    {
        Timer timer = new Timer();
// Здесь мы используем готовый делегат из пространства имен System.Timers
//  Можно оставить просто название метода. Здесь представлен такой сложный механизм,
//  чтобы дать понять, что для назначения события используется делегат, который,
// в свою очередь, предназначен, чтобы содержать в себе ссылку на метод
        timer.Elapsed += new ElapsedEventHandler(TimerEventHandler);
        timer.Interval = 1000;
        timer.Enabled = true;
        Console.ReadKey();
    }
       // Все события в Windows Forms строятся по этому шаблону
       // Сначала идет информация о том, кто создал событие, а потом передается информация о событии
      // С помощью объекта sender мы можем узнать, кто сгенерировал событие
      // А с помощью объекта e - информацию о событии
    private static void TimerEventHandler(object sender, ElapsedEventArgs e)
    {
        // Выведем информацию о том, что сгенерировало событие
        Console.WriteLine(sender.ToString());
        // И информацию, переданную событию
        Console.WriteLine(e.SignalTime.ToString());
    }
}
```

Класс Form

Следующим шагом будет знакомство с базовым классом, на основе которого строится большинство приложений в Windows Forms.

В этом примере мы создаём простейшее приложение, в котором создаём объект класса Form. Этот объект представляет окно с заголовком и областью для элементов. Конечно, первоначально никаких элементов на объекте нет. А окно закрывается сразу же после закрытия консольного приложения.

```
// Требуется добавить в проект ссылку на сборку System.Windows.Forms
using System.Threading;// Для создания паузы перед закрытием приложения
using System.Windows.Forms;
class ShowForm
{
    public static void Main()
    {
                        // Создаем объект класса Form
        Form form = new Form();
                        // Обращаемся к некоторым свойствам графического окна
        form.Text = "Это простое графическое окно";
        form.Width = 100;
                        // Показываем этот объект
        form.Show();
                         // Если не сделать паузу, то консольное окно и форма закроются
```

```
        Thread.Sleep(3000);
    }
}
```

Диспетчер очереди сообщений Windows

Чтобы наша форма продолжала работать под управлением Windows, требуется создать очередь сообщений и передать созданное окно в эту очередь. Для этого используется метод Run класса Application. Его использование продемонстрировано в следующем примере:

```
// Требуется добавить в проект ссылку на сборку System.Windows.Forms
using System.Windows.Forms;
// Пример создания диспетчера обработки событий Windows
// Приложения в современных ОС обрабатывают сообщения от операционных систем
// как бы находясь постоянно в закольцованном состоянии, пока приложение не
сообщает
//, что закончило работу
class TwoForms
{
    public static void Main()
    {
        Form form1 = new Form();
        Form form2 = new Form();
          form1.Text = "Эта форма запущена с использованием метода Run класса
Application";
             form2.Text = "Это форма для демонстрации возможности создавать
несколько форм";
        form2.Show();
        Application.Run(form1);
        MessageBox.Show("Application.Run() вернул " +
        "управление в метод Main. До свидания", "Приложение \"Две формы\"");
    }
}
```

В этом примере показано, что нам ничего не мешает создать несколько форм. Пока форма, которая находится в очереди сообщений, не закроется, приложение будет считаться запущенным.

Самое частое событие в жизни Windows Forms

Событие Click, может быть, и не самое частое в «жизни» приложения Windows Forms, но точно будет одним из самых частых в вашей жизни, так как именно его придётся обрабатывать чаще всего.

```
// Требуется добавить в проект ссылку на сборку System.Windows.Forms
using System;
using System.Windows.Forms;
class ClickEvent
{
    public static void Main()
    {
        Form form = new Form();
        form.Text = "Событие Click";
        // У формы есть событие Click
             //  в  System.Windows.Form  описан  делегат  EventHandler(Обработчик
```

```
события),
         // который описывает сигнатуру методов, которые можно подключать на
событие
        // Можно записать просто Form_Click
        form.Click += new EventHandler(Form_Click);
        Application.Run(form);
    }
    private static void Form_Click(object sender, EventArgs e)
    {
        // Посмотрим, что же вызвало событие
        Console.WriteLine(sender.ToString());
        MessageBox.Show("Щелкнули по форме!", "Щелк");
    }
}
```

Событие, которое нельзя пропустить

Теперь стоит понять, что приложение в Windows выполняется, обрабатывая различные события от пользователя или операционной системы. Отдельно стоит остановиться на перерисовке окна формы. Когда операционная система отправляет форме сообщение, что оно должно быть перерисовано, то вызывается событие Paint, а событие Paint в свою очередь вызывает обработчик события. Вот так сложно.

```
  //  Требуется  добавить  в  проект  ссылки  на  сборки  System.Drawing  и
System.Windows.Forms;
using System;
using System.Drawing;
using System.Windows.Forms;
class PaintEvent
{
    public static void Main()
    {
        Form form = new Form();
        form.Text = "Событие Paint";
        // У формы есть событие Paint,
        // в System.Windows.Form описан делегат PaintEventHandler,
          // который описывает сигнатуру методов, которые можно подключать на
событие
               //Создаем  делегат  и  указываем,  что  он  указывает  на  метод
MyPaintHandler

        form.Paint += new PaintEventHandler(MyPaintHandler);
        Application.Run(form);
    }
    static void MyPaintHandler(object objSender, PaintEventArgs pea)
    {
           // Получаем ссылку на класс Graphics, в котором содержатся поля и
методы для рисования на форме
        Graphics grfx = pea.Graphics;
          // Очищаем форму, закрашивая ее цветом
        grfx.Clear(Color.Chocolate);
          // Будем в заголовке окна менять время, чтобы лучше понять, когда же
срабатывает это событие
        (objSender as Form).Text = DateTime.Now.ToLongTimeString();
          // А также посмотрим, что же вызывает это событие
```

```
        Console.WriteLine(objSender.ToString());
    }
}
```

Свойство Graphics содержит экземпляр класса Graphics, определённого в пространстве имен System.Drawing.Graphics (важнейший класс библиотеки WindowsForms, по важности не уступающий Form). Этот класс служит для рисования графики и вывода текста на форме.

Наследники класса Form

В предыдущей программе показан способ создания и применения объекта Form, но обычно это делается по-другому. Более гибкий и привлекательный подход — создание класса-наследника Form.

```
class MyForm: Form
{
…
}
```

Можно создать объект этого класса, как аргумент Application.Run в Main. Или определить в программе ещё один класс, предназначенный только для статистического метода Main:

```
class MyProgram
{
  public static void Main()
  {
    Application.Run(new MyForm());
  }
}
```

Поскольку MyForm — класс-потомок Form, у него есть доступ ко всем открытым и защищённым методам, свойствам и событиям класса Form. Следовательно, в своём конструкторе он может задавать свойства формы.

```
class MyForm: Form
{
  public MyForm()
  {
    Text = "My Inherited Form";
    Width *= 2;
  }
}
```

В C# у конструктора такое же имя, как и у класса, но нет возвращаемого значения. Это первый метод в этом уроке, который не определён как статический. Конструктор применяется к объекту типа MyForm. Перед свойствами не нужно указывать имя объекта. На самом деле объект MyForm не

существует, пока не создан в Main. Для ссылки на текущий объект в конструкторе, методе или свойстве можно использовать ключевое слово this из C#.

```
this.Text = "My Inherited Form";
```

Также можно установить обработчики событий.

```
class MyForm: Form
{
  public MyForm()
  {
    Text = "My Inherited Form";
    Width *= 2;
    Click += MyClicker;
    Paint += MyPainter;
  }
  void MyClicker(object objSrc, EventArgs args)
  {
    MessageBox.Show("The button has been clicked!", "Click");
  }
  void MyPainter(object objSrc, PaintEventArgs args)
  {
    Graphics grfx = args.Graphics;
    grfx.DrawString("Hello, Windows Forms", Font,
    SystemBrushes.ControlText, 0, 0);
  }
}
```

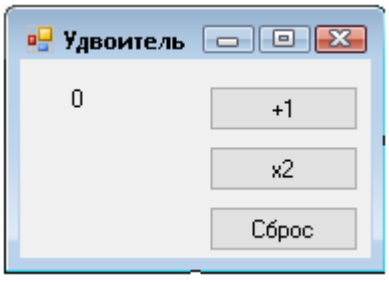
Создаём первое приложение

Вы получили некоторое представление об устройстве приложения Windows. При создании приложения Visual Studio берёт огромную часть работы на себя, и на первых порах можно программировать, имея поверхностное представление о внутренней организации приложения. Что мы с вами и сделаем.

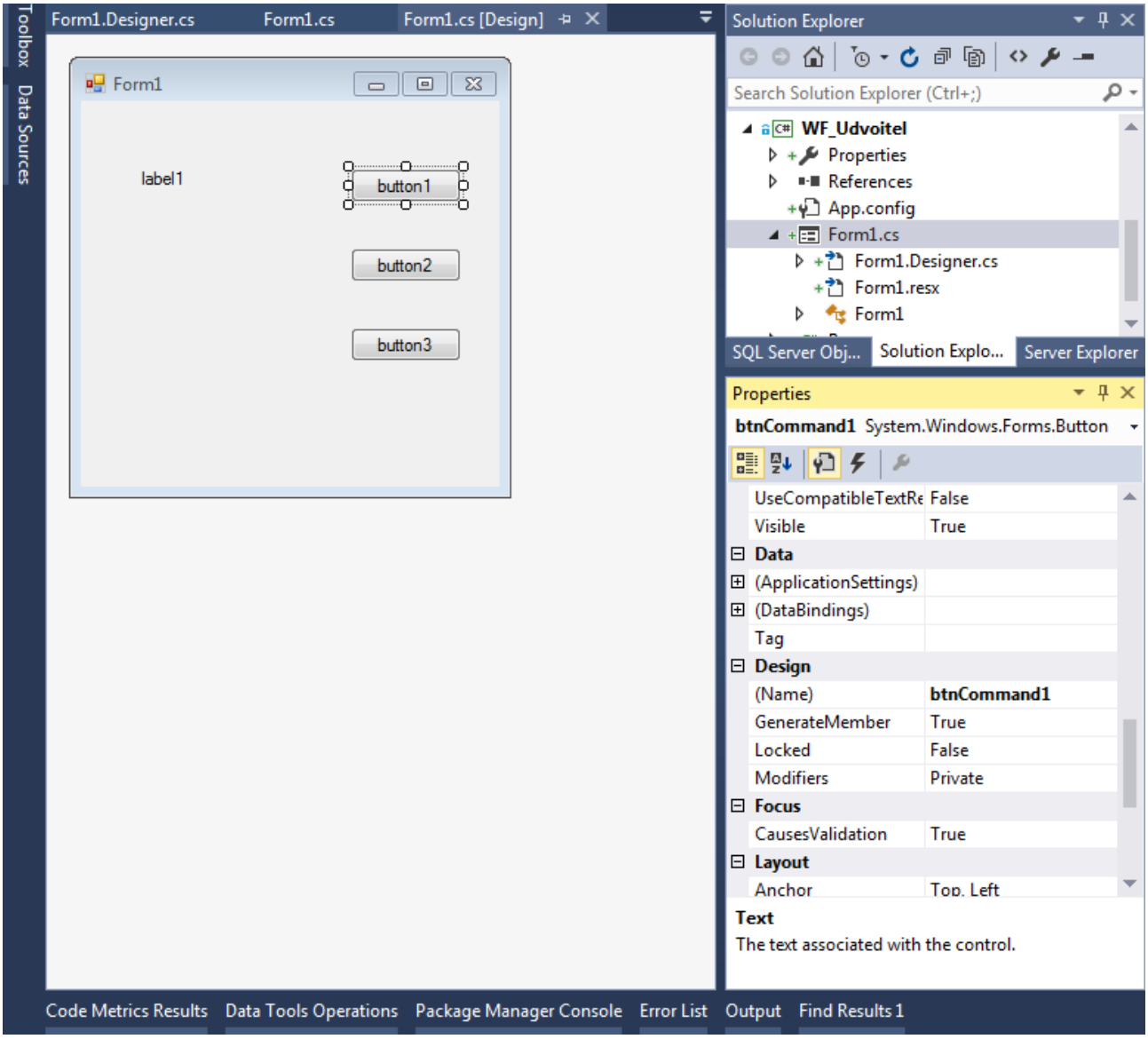
Создадим простую программу в Windows Forms, реализующего исполнителя «Удвоитель».

Запустите Visual Studio. Создайте новый проект — приложение Windows Forms. Название проекта — WF_Udvoitel.

Перетащите из панели элементов три кнопки (Button) и метку (Label), как показано на рисунке.



Переименуйте кнопки и метки, задав им имена (Name) btnCommand1, btnCommand2, btnReset, lblNumber соответственно.



Щёлкните два раза по каждой кнопке, чтобы создать обработчики событий. Напишите код для каждого обработчика. Задайте свойство Text для каждого элемента.

У вас должен получиться примерно такой текст:

```
using System;
using System.Windows.Forms;
namespace WF_Udvoitel
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
      btnCommand1.Text = "+1";
      btnCommand2.Text = "x2";
      btnReset.Text = "Сброс";
```

```
    lblNumber.Text = "0";
       this.Text = "Удвоитель";
         }
         private void btnCommand1_Click(object sender, EventArgs e)
         {
             lblNumber.Text = (int.Parse(lblNumber.Text) + 1).ToString();
         }
         private void btnCommand2_Click(object sender, EventArgs e)
         {
             lblNumber.Text = (int.Parse(lblNumber.Text) * 2).ToString();
         }
         private void btnReset_Click(object sender, EventArgs e)
         {
             lblNumber.Text = "1";
         }
     }
}
```

Домашнее задание

1. а) Добавить в программу «Удвоитель» подсчёт количества отданных команд удвоителю.
   б) Добавить меню и команду «Играть». При нажатии появляется сообщение, какое число должен получить игрок. Игрок должен получить это число за минимальное количество ходов.
   в) *Добавить кнопку «Отменить», которая отменяет последние ходы. Используйте обобщенный класс Stack.
   Вся логика игры должна быть реализована в классе с удвоителем.

2. Используя Windows Forms, разработать игру «Угадай число». Компьютер загадывает число от 1 до 100, а человек пытается его угадать за минимальное число попыток. Компьютер говорит, больше или меньше загаданное число введенного.
   а) Для ввода данных от человека используется элемент TextBox;
   б) **Реализовать отдельную форму с TextBox для ввода числа.

Старайтесь разбивать программы на подпрограммы. Переписывайте в начало программы условие и свою фамилию. Все программы сделать в одном решении.

В свойствах проектах в качестве запускаемого проекта укажите “Текущий выбор”

Дополнительные материалы

1. Пошаговое руководство. Создание простого приложения с помощью Visual C# или Visual Basic.
2. Пошаговые руководства по Windows Forms.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Павловская Т.А. Программирование на языке высокого уровня. — СПб: Питер, 2009.
2. Петцольд Ч. Программирование на C#. Т.1. — М.: Русская редакция, 2001.
3. Шилдт Г. C# 4.0. Полное руководство. — М.: ООО «И.Д. Вильямс», 2011.

4. MSDN.