

ВВЕДЕНИЕ В WINDOWS FORMS

План занятия

2

- Понятие Windows и Web Forms.
- Создание Windows-форм.
- Форматирование элементов управления.
- События в Windows-формах.

Типы приложений Windows

3

- Консольные приложения
- Приложения Windows Forms
- Web-приложения:
 - ASP.NET-приложения
 - Web-сервисы

Понятие Windows Forms

4

- **Windows Forms** - технология для платформы .NET Framework в форме набора библиотек, упрощающих выполнение типичных задач приложений (чтение и запись в файловую систему и т.п.).
- Возможности приложений Windows Forms в среде разработки Microsoft Visual Studio .NET:
 - вывод информации;
 - ввод данных пользователем;
 - обмен информацией с удаленными компьютерами через сетевое соединение

Win Forms: Элементы интерактивных интерфейсов

5

- **Форма** – визуальная поверхность, на которой отображается информация для пользователя.
- **Контрол (элемент управления)** – это элемент пользовательского интерфейса, отображающий данные или принимающий ввод данных.
- **Создание приложения** Win Forms – процесс добавления контролов на форму и описания реакций на действия пользователя, (щелчок мыши, нажатие клавиши и т.д.).
- Два главных пространства имен для создания приложений с графическим интерфейсом в .NET:
 - System.Windows.Forms
 - System.Web.UI

Win Forms: Элементы интерактивных интерфейсов

6

- Windows-приложения имеют стандартный *графический интерфейс* и используют *событийно-управляемое программирование*.
- Для создания Windows-приложения в C# используется шаблон **Приложение** *Windows Form*;
- **Форма** представляет собой окно и предназначена для размещения компонентов (элементов управления).
- Форма и ее компоненты находятся в пространстве имен *Windows.Forms*.

Основные типы Windows.Forms компонент

7

Application	Этот класс представляет саму суть приложения Windows Forms. При помощи методов этого класса можно обрабатывать сообщения Windows, запускать и прекращать работу приложения и т. п.
ButtonBase, Button, CheckBox, ComboBox, DataGrid, GroupBox, ListBox, Label, PictureBox	Эти классы (а также многие аналогичные им) представляют элементы графического интерфейса
Form	Этот тип представляет главную форму (диалоговое окно) приложения Windows Forms
ColorDialog, FileDialog, FontDialog, PrintPreviewDialog	Это готовые к употреблению диалоговые окна для выбора цветов, файлов, шрифтов и т. п.
Menu, MainMenu, MenuItem, ContextMenu	Эти типы предназначены для создания ниспадающих и контекстных меню
Clipboard, Help, Timer, Screen, ToolTip, Cursors	Разнообразные вспомогательные типы для организации интерактивных графических интерфейсов
StatusBar, Splitter, ToolBar, ScrollBar	Дополнительные элементы управления, размещаемые на форме

Порядок создания Windows-приложения

8

Для создания нового проекта нужно выполнить команду:

File – New Project... – Windows Application

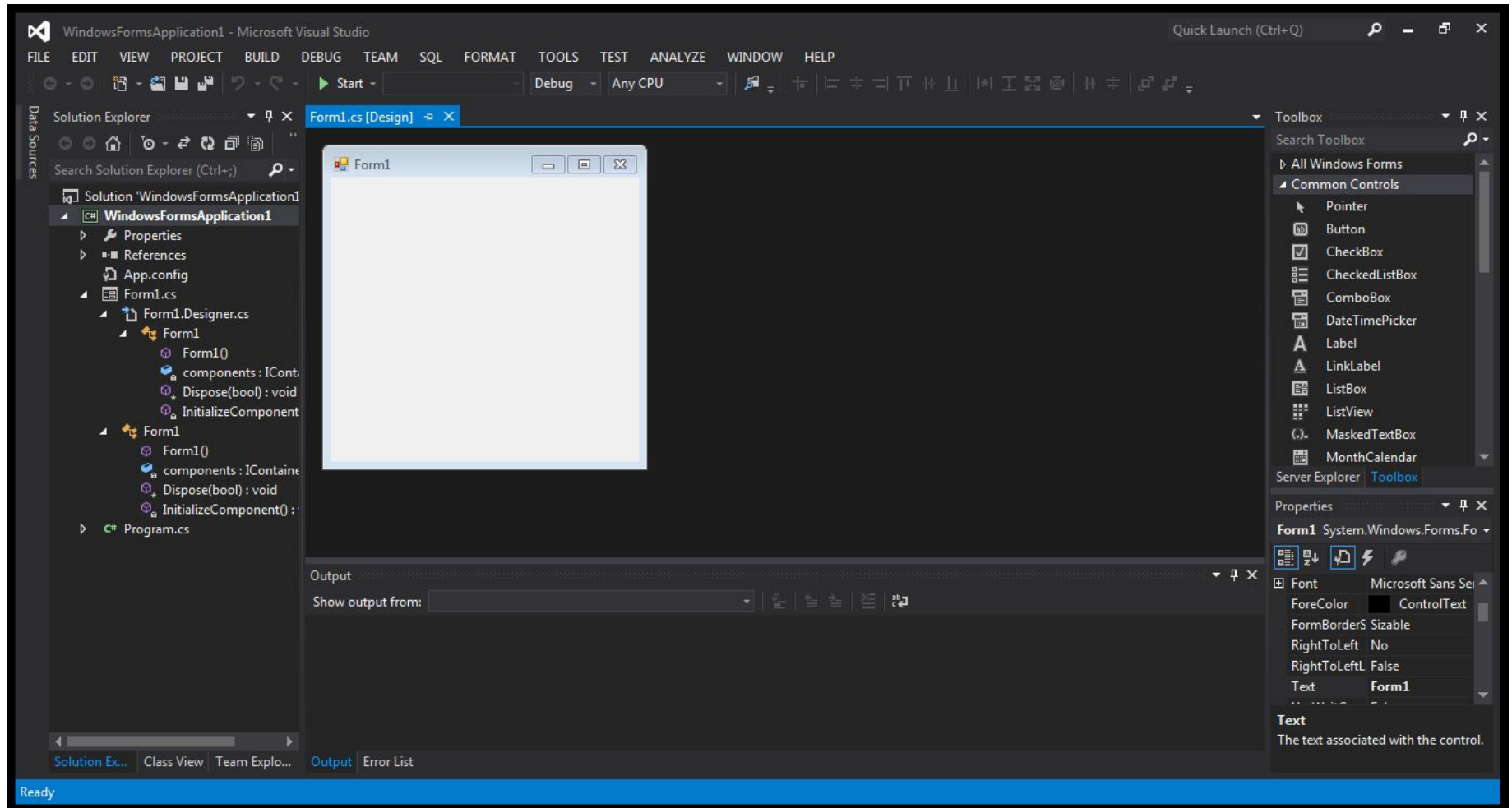
Средой будет сформирован шаблон приложения и заготовка формы Form1 на вкладке **Form1.cs [Design]** в центральной части окна.

Сбоку будет отображена палитра компонентов (панель инструментов) **ToolBox**

Если она отсутствует, открыть ее можно командой меню **View –ToolBox (Ctrl+Alt+X)**

Порядок создания Windows-приложения

9



Вид экрана

Порядок создания Windows-приложения

10

Создание приложения состоит из двух этапов:

- Визуальное проектирование – создание интерфейса приложения (конструирование форм).
- Разработка и реализация алгоритма решения задачи путем написания процедур обработки событий.

На **I** этапе нужно разработать вид всех окон приложения, определить их иерархию, а затем в среде создать нужное количество форм, разместить на них все необходимые компоненты (элементы управления) и установить их свойства с помощью окна свойств **Properties**.

Порядок создания Windows-приложения

11

- Визуальное проектирование – размещение на форме компонентов и задание их свойств с помощью окна свойств.

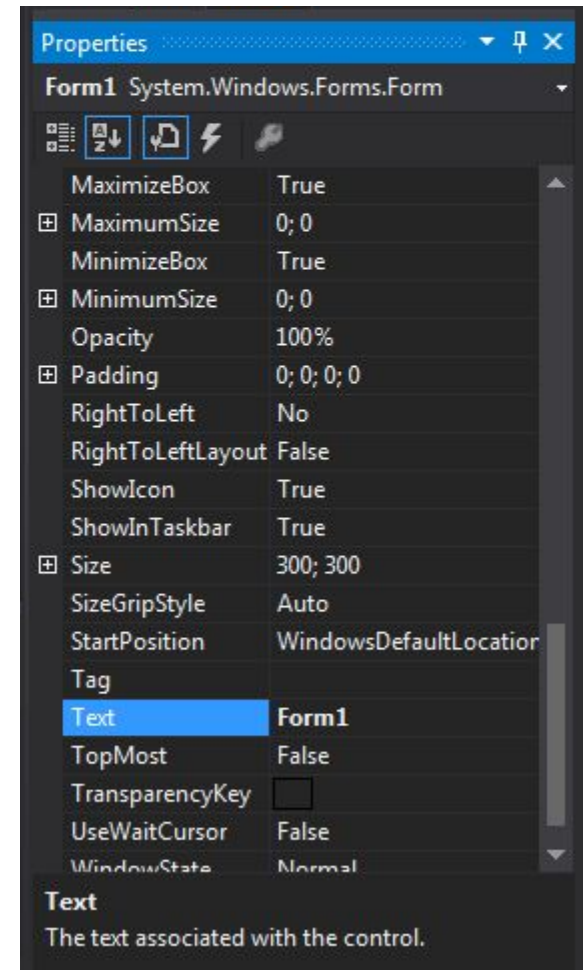
Компонент, чьи
свойства меняются

Свойства

События

Название
свойства

Значение
свойства



Порядок создания Windows-приложения

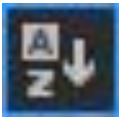
12

Свойства отражаются в алфавитном порядке или сгруппированы по категориям.

Выбор способа отображения осуществляется с помощью кнопок



Categorized – группировка по категориям.



Alphabetical – группировка по алфавиту.

Для размещения в окне формы элемента управления можно дважды щелкнуть по соответствующему значку в палитре компонентов или сделать один щелчок на компоненте и один щелчок на форме.

Form – Форма, Окно

13

- Модальная форма не позволяет пользователю переключиться на другие окна этого же приложения, пока не будет завершена работа с текущим окном (пример диалоговые окна, окна сообщений)
- Немодальная форма – позволяет переключаться на другие окна того же приложения.
- Каждое приложение содержит одну главную форму. При закрытии главной формы приложение закрывается.

Заготовка формы...

14

```
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 WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

Некоторые свойства класса Form

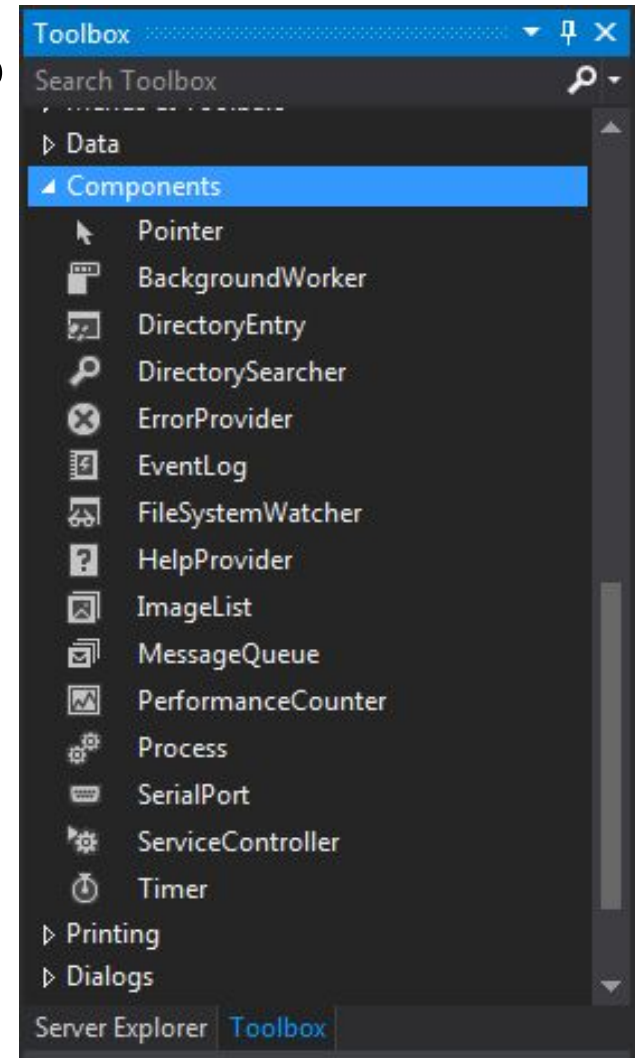
15

Свойство	Значение
AcceptButton	Позволяет задать кнопку, которая будет активироваться при нажатии на Enter
AutoSize	Будет ли форма автоматически изменять размер в зависимости от размера размещенных на ней компонентов
FormBorderStyle	Стиль рамки вокруг формы (изменяемый размер или нет)
CancelButton	Позволяет задать кнопку, которая будет активироваться при нажатии на Esc
ControlBox	Наличие системного меню в левом верхнем углу формы
MaximizedBox	Наличие кнопок в правом верхнем углу формы
ShowInTaskbar	Отображать ли форму на панели задач
StartPosition	Где форма будет располагаться при запуске приложения
WindowState	В каком состоянии форма будет запущена (свернутая, развернутая, обычная)

Основные компоненты и их свойства

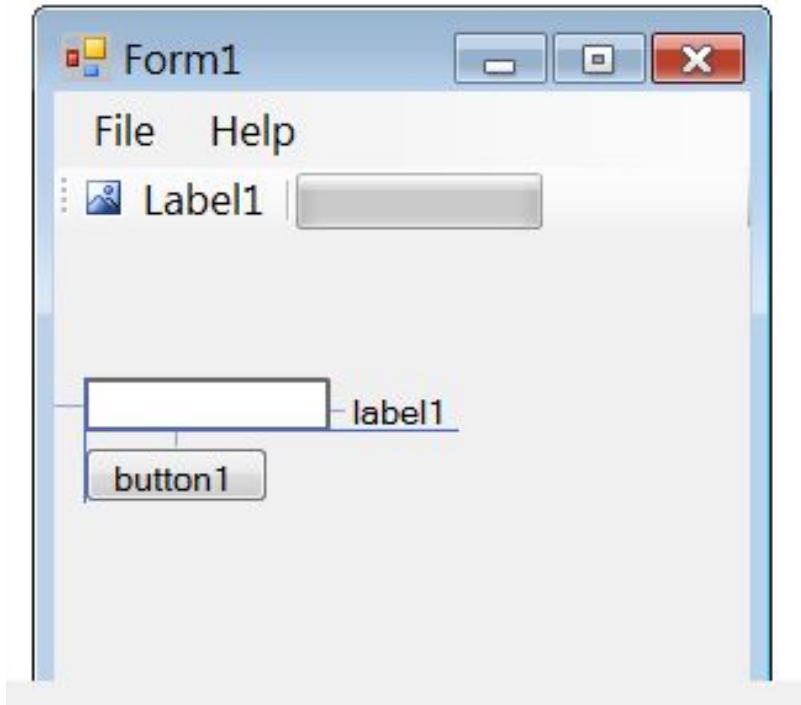
16

- Windows Forms содержит широкий спектр контролов различных видов, которые можно разместить на форму:
 - Текстовое поле (text box);
 - Кнопка (button);
 - Выпадающий список (drop-down box);
 - Переключатель (radio button);
 - Web-страница.
- Для размещения компонентов на форме необходимо открыть панель элементов:
Вид – Панель элементов
- Обычное назначение компонентов это получение данных от пользователя или информирование пользователя



Windows Forms Designer

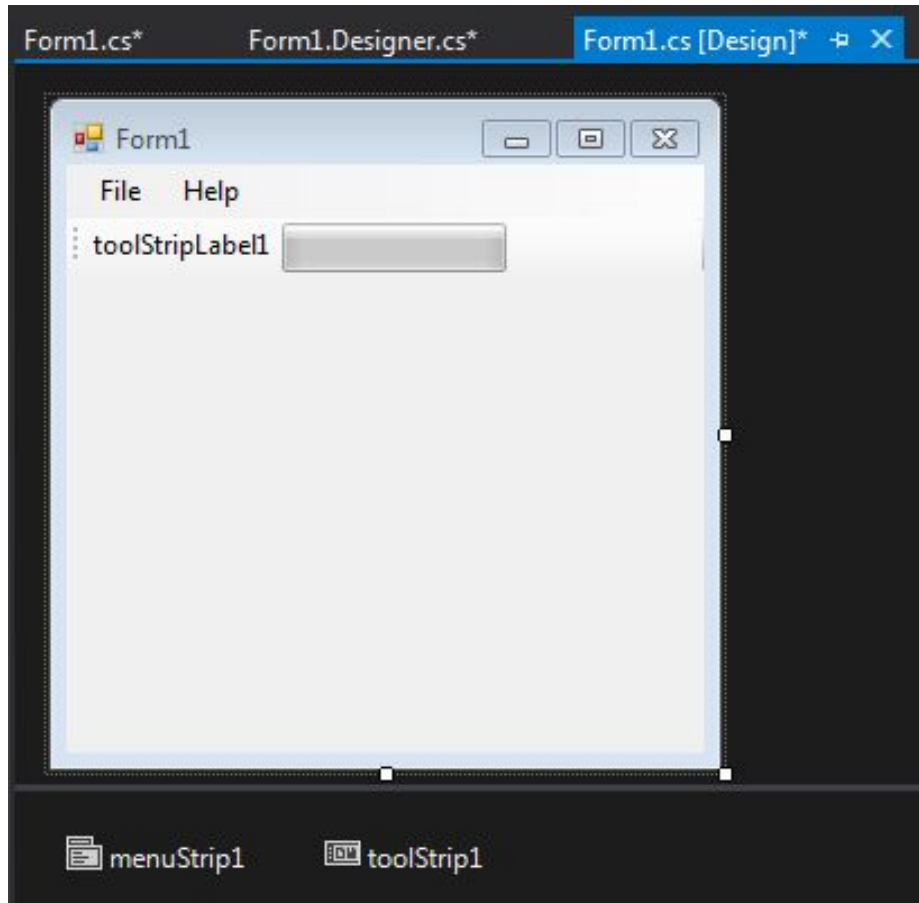
17



- С помощью инструментария Windows Forms Designer в среде Visual Studio можно:
- создавать Windows Forms-приложения посредством мыши, «перетаскивая» (drag-and-drop) нужные контролы на Windows-форму.
 - выравнивать контролы относительно друг друга и Windows-формы.

Создание эргономичных, интерактивных интерфейсов

18



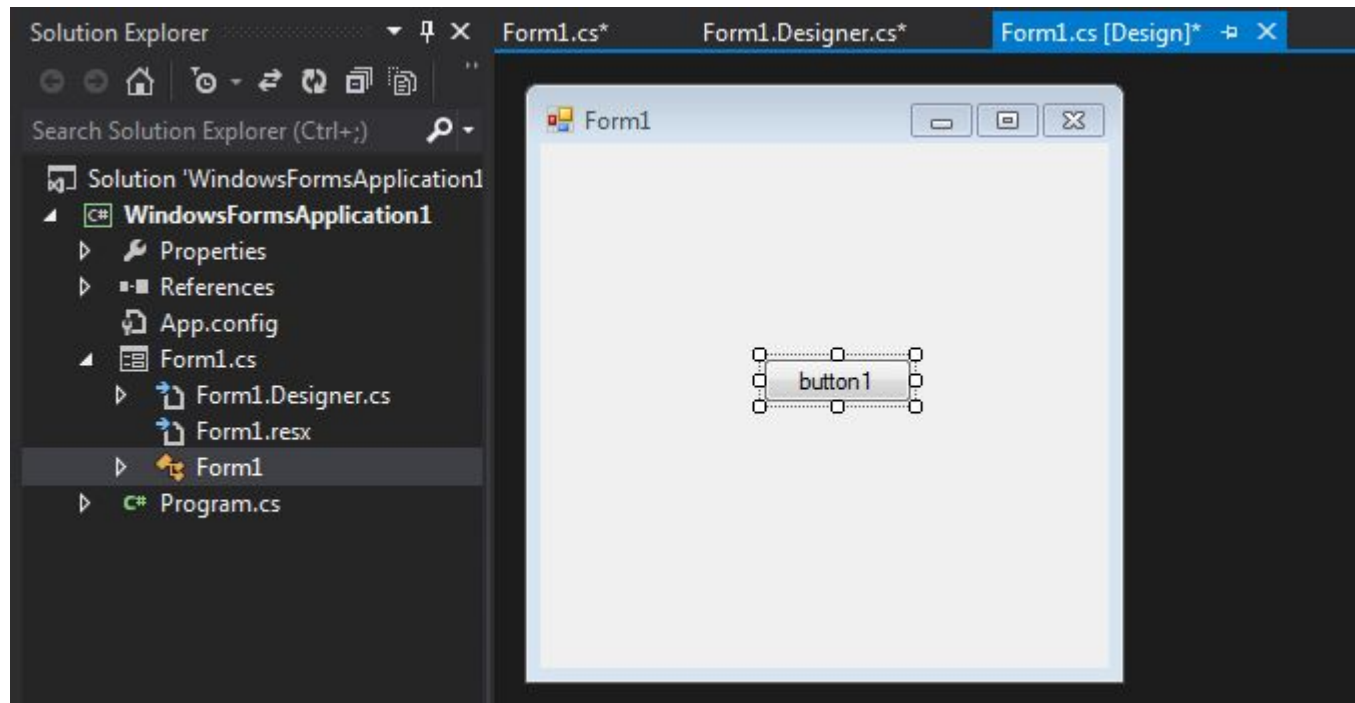
В Windows Forms встроены
контроли пользовательского
интерфейса, реализующие
важные особенности
приложений Microsoft Office

Так, контролы ToolStrip и
MenuStrip помогают
создавать панели
инструментов и меню,
которые содержат:

- текст
- Изображения
- Подменю
- другие контролы (textbox, и combobox и др.)

Создание эргономичных, интерактивных интерфейсов

19



Пример создания приложения `Form1` по технологии Win Forms в среде Visual Studio с элементом управления – кнопкой `button1`

Пользовательские элементы управления

20

- Формы ввода и распределенные приложения поддерживает возможность создания пользователями / разработчиками собственных элементов управления
- Для этого используется класс UserControl библиотеки Microsoft .NET Framework
- Кроме того, возможно использование свойств уже реализованных существующих элементов управления посредством наследования

Создание собственных элементов интерфейса

21

Для создания собственных элементов интерфейса пользователя, разработчиками применяются классы системного пространства имен `System.Drawing` из `Microsoft .NET Framework`, позволяющие осуществлять непосредственно на форме прорисовку:

- линий;
- окружностей;
- других видов графических объектов.

Порядок создания Windows-приложения

22

На **II** этапе создания Windows-приложения нужно разработать алгоритмы всех подпрограмм и установить, в результате наступления каких событий нужно будет выполнить созданные подпрограммы. Затем создать обработчики событий – методы, которые будут выполнены при наступлении того или иного события.

Шаблон текста приложения уже создан.

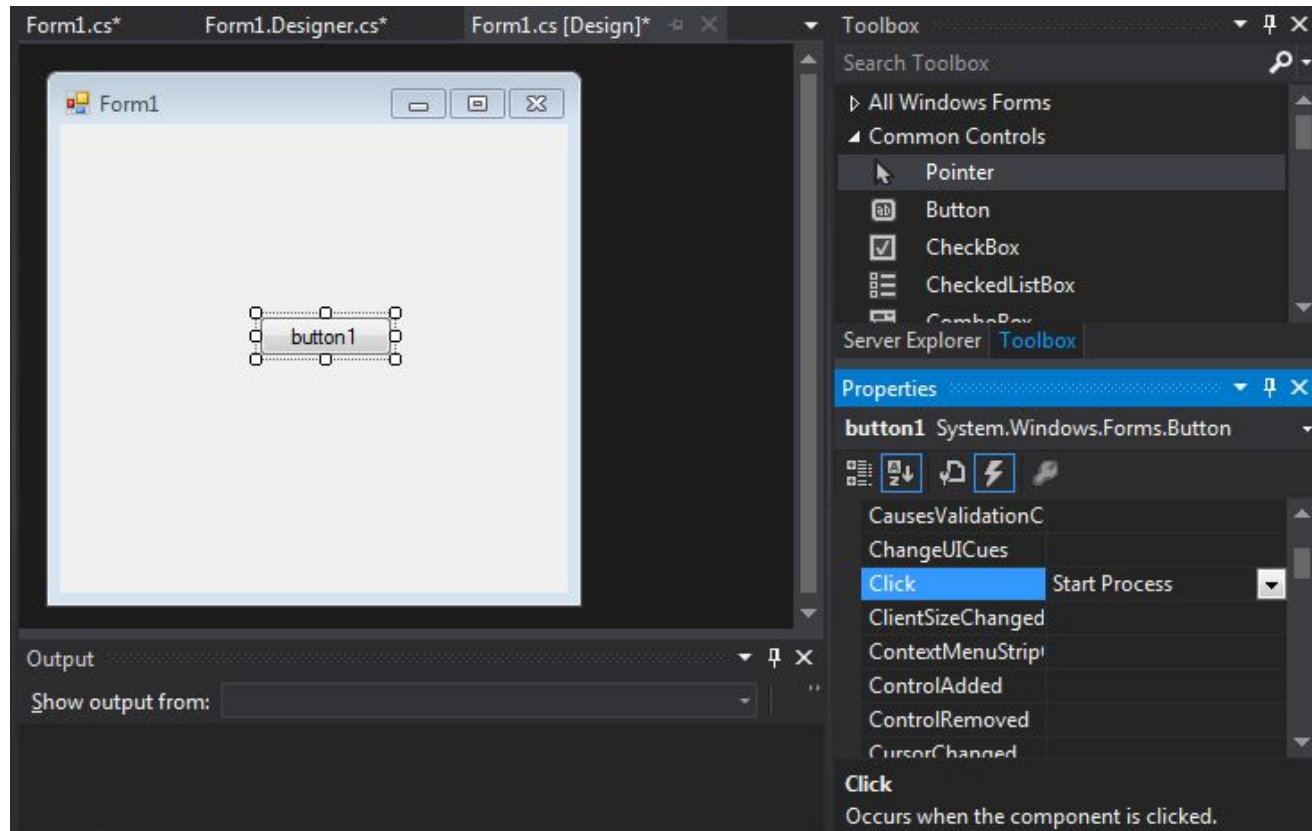
Увидеть текст файла, в котором будут размещаться обработчики событий, можно, щелкнув правой кнопкой мыши в окне **Solution Explorer** на файле **Form1.cs** и выбрав в контекстном меню команду **View Code**.

Обработка события

23

При взаимодействии пользователя с формой или контролем, генерируется **событие** (event).

Приложение реагирует на событие с помощью кода - обрабатывает его, как только оно происходит.

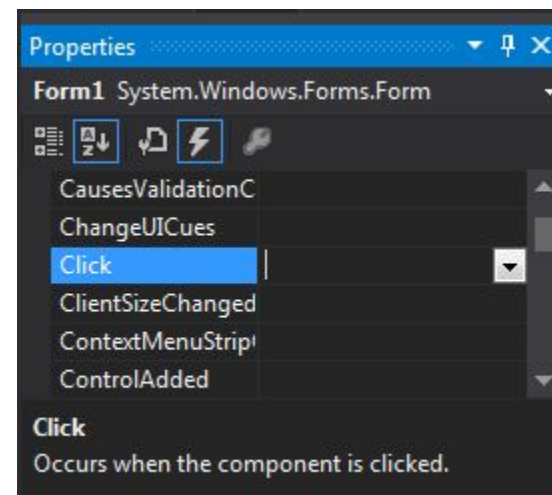


Порядок создания Windows-приложения

24

- Определение поведения – описывает алгоритм, какие действия должны выполняться при щелчке на кнопках, вводе текста, выборе пунктов меню и т.д. (т.е. по каким событиям необходимо выполнять действия).
- Заготовка для обработчика события формируется *двойным щелчком* на поле, расположенном справа от имени события.

Нажать тут :)



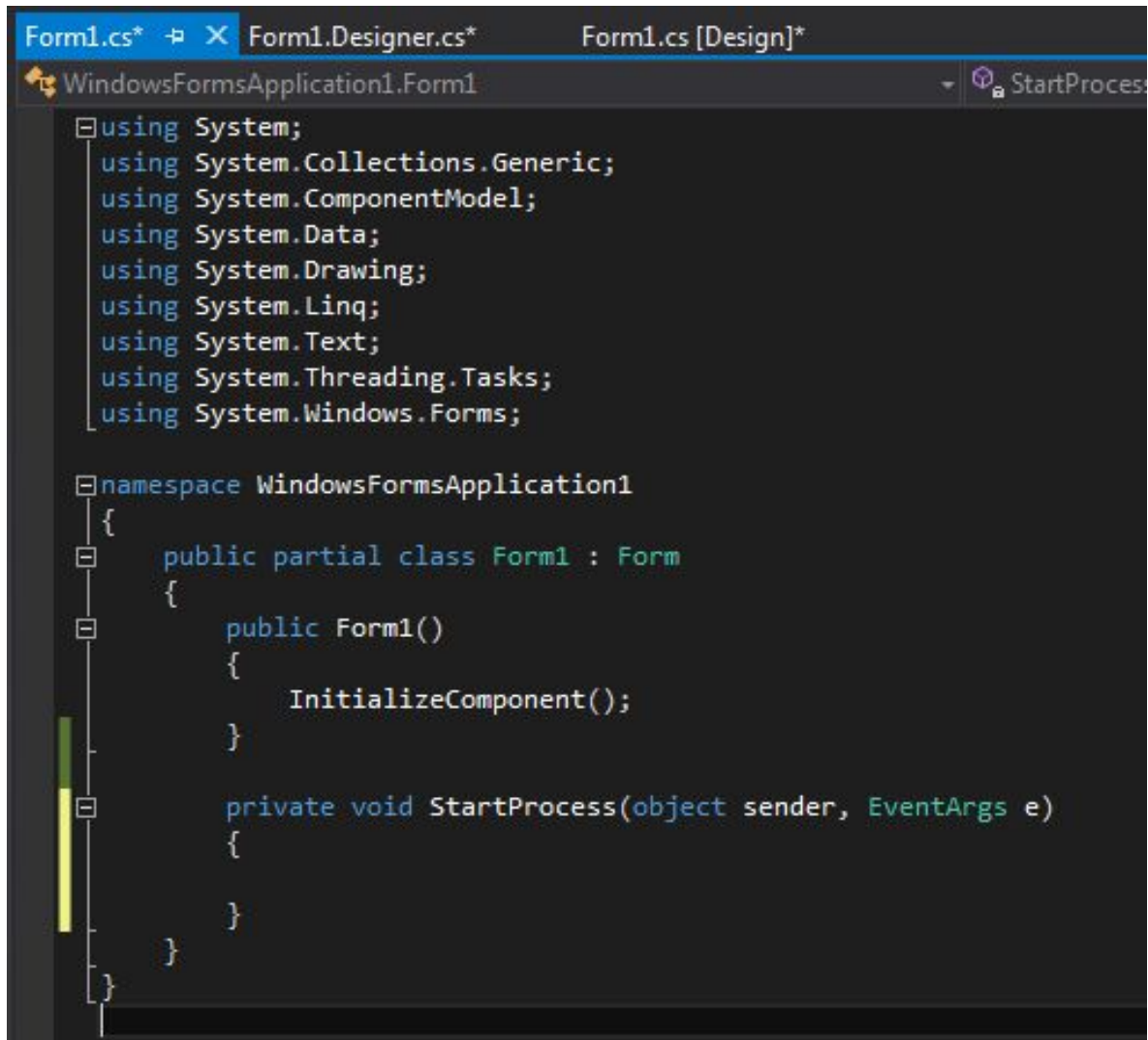
Создание обработчиков событий в Windows Forms

25

- Обработчик событий — это процедура в коде, определяющая действия, которые требуется выполнить при возникновении события, например когда пользователь нажимает кнопку или когда в очередь сообщений поступает очередное сообщение. При возникновении события выполняется обработчик (или обработчики) событий, который получает это событие. События могут быть назначены нескольким обработчикам, а методы, обрабатывающие отдельные события, могут динамически меняться. Можно также использовать конструктор Windows Forms Designer для создания обработчиков событий. .

Код обработки события на языке C#

26



```
Form1.cs*  Form1.Designer.cs*  Form1.cs [Design]*
WindowsFormsApplication1.Form1
StartProcess

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 WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void StartProcess(object sender, EventArgs e)
        {
        }
    }
}
```

Создание обработчиков событий в Windows Forms

27

При создании приложений наиболее часто используются следующие события:

- **Activated** – получение формой фокуса ввода.
- **Click, DoubleClick** – одинарный и двойной щелчки мышью
- **Closed** – закрытие формы.
- **Load** – загрузка формы.
- **KeyPress** – нажатие клавиши, имеющей ASCII-код.
- **MouseMove** – перемещение мыши и др.

Для автоматического создания шаблона обработчика события нужно в окне **Properties** перейти на вкладку **Events** (*с изображением молнии*).

Откроется список всех событий, которые может сгенерировать выбранный компонент.

Затем дважды щелкнуть на поле справа от имени нужного события.

При этом появляется вкладка окна редактора кода с заготовкой соответствующего обработчика.

Создание обработчиков событий в Windows Forms

28

Например, обработчик события нажатия кнопки **button1**:

```
private void button1_Click(object sender, EventArgs e)  
{  
}
```

Тот же эффект будет получен при выполнении двойного щелчка по кнопке **button1**

Имя обработчика события генерируется по следующему правилу:

ИмяЭлементаУправления_ИмяСобытия

Шаблон Windows-приложения

29

При создании приложения автоматически создаются несколько файлов. Рассмотрим основные.

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

Шаблон Windows-приложения

30

```
namespace WindowsApplication1  
{
```

Объявляется класс главной формы приложения, производный от класса Form пространства Windows.Forms

```
public partial class Form1 : Form  
{
```

Спецификатор **partial** означает ~~частичное~~ описание класса. Другая часть этого класса будет содержаться в другом файле.

```
public Form1( ) // конструктор  
{  
    InitializeComponent(); // статический закрытый метод  
    // класса Form1, код которого автоматически  
    // генерируется средой при добавлении новых компонентов  
} } }
```

Шаблон Windows-приложения

31

Именно в этом файле будут автоматически создаваться обработчики событий.

Файл **Form1.Designer.cs** (чтобы его увидеть, нужно дважды щелкнуть по его имени в окне Solution Explorer):

```
namespace WindowsApplication1
{
    partial class Form1 // продолжение описания класса
    {

        // описание поля интерфейсного типа – контейнера для
        // компонентов:
        private System.ComponentModel.IContainer components = null;
```

Шаблон Windows-приложения

32

```
protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}
```

// Этот метод будет при закрытии формы автоматически
// удалять все ресурсы

При размещении на форме компонента в методе
InitializeComponent() автоматически происходят изменения.

Шаблон Windows-приложения

33

Например, если на форме разместить кнопку, в классе появится закрытое поле:

```
private System.Windows.Forms.Button button1;
```

А в методе появятся следующие операторы:

```
this.button1 = new System.Windows.Forms.Button();  
this.button1.Location = new System.Drawing.Point(77, 42);  
    this.button1.Name = "button1";  
    this.button1.Size = new System.Drawing.Size(75, 23);  
    this.button1.TabIndex = 0;  
    this.button1.Text = "button1";  
    this.button1.UseVisualStyleBackColor = true;  
    this.Controls.Add(this.button1);
```

Шаблон Windows-приложения

34

При создании обработчика события в метод **InitializeComponent()** автоматически добавляется оператор, который регистрирует обработчик события.

Например, при создании метода-обработчика события нажатия кнопки **button1** в метод **InitializeComponent()** будет добавлен оператор:

```
this.button1.Click += new System.EventHandler(this.button1_Click);
```

Обработчики события можно также создавать вручную.

Шаблон Windows-приложения

35

Например, пусть автоматически создан обработчик события

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

Можно в классе Form1 создать еще один обработчик этого же события:

```
private void button1_Click_1(object sender, EventArgs e)
{
    MessageBox.Show("Ура!!!");
}
```

Шаблон Windows-приложения

36

Тогда в метод **InitializeComponent()** нужно добавить оператор:
`this.button1.Click += this.button1_Click_1;`

В этом случае при нажатии на кнопку **button1** во время выполнения программы будут по очереди выполнены оба метода.

Если планируется использовать в работе программы больше одного окна, нужно создать соответствующее количество форм командой меню:

Project – Add Windows Form... - Windows Form

При этом автоматически будут созданы файлы **Form2.cs**, **Form2.Designer.cs**, **Form3.cs**, **Form3.Designer.cs**, и т.д.

- Введение в платформу .NET Framework и ASP.NET
<http://www.intuit.ru/studies/courses/4455/712/lecture/10049>
- C#. Введение в программирование
<http://window.edu.ru/resource/674/41674/files/marchenko.pdf>
- Учебник по C# <http://www.dotsite.spb.ru/Tutorials/CSharp>
- Visual studio Express
<http://www.microsoft.com/visualstudio/rus/products/visual-studio-express-products>
- В. Фаронов. Программирование на языке C#. ПИТЕР, 2007
- Разработка приложений на C# в среде Visual Studio
В.М. Снетков <http://www.intuit.ru/department/se/csharpvs2005>
- Обучающие уроки по C#
<http://www.programmer-lib.ru/csharp.php>

СПАСИБО ЗА ВНИМАНИЕ!