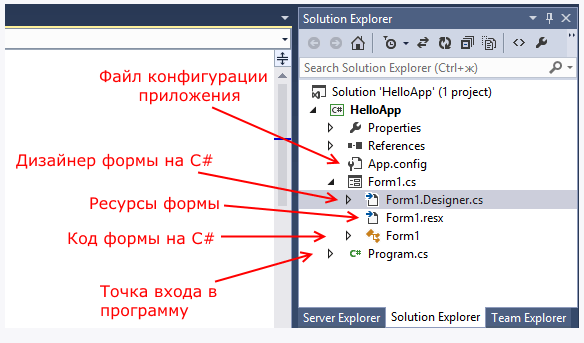
# Работа с формами

## Основы форм

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

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  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | namespace HelloApp  {      partial class Form1      {          /// <summary>          /// Required designer variable.          /// </summary>          private System.ComponentModel.IContainer components = null;            /// <summary>          /// Clean up any resources being used.          /// </summary>          /// <param name="disposing">true if managed resources should be disposed; 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, 13F);              this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;              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.

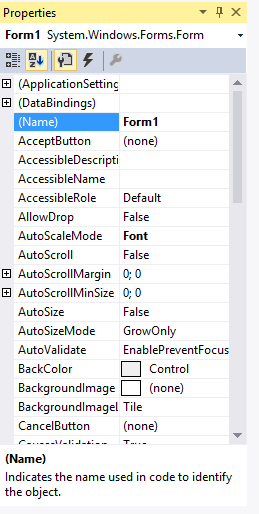
Еще один файл - *Form1.resx* - хранит ресурсы формы. Как правило, ресурсы используются для создания однообразных форм сразу для нескольких языковых культур.

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | 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.

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:

* **Name**: устанавливает имя формы - точнее имя класса, который наследуется от класса Form
* **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
* **BackgroundImage**: указывает на фоновое изображение формы
* **BackgroundImageLayout**: определяет, как изображение, заданное в свойстве BackgroundImage, будет располагаться на форме.
* **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
* **Cursor**: определяет тип курсора, который используется на форме
* **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.
* **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
* **ForeColor**: цвет шрифта на форме
* **FormBorderStyle**: указывает, как будет отображаться граница формы и строка заголовка. Устанавливая данное свойство в None можно создавать внешний вид приложения произвольной формы
* **HelpButton**: указывает, отображается ли кнопка справки формы
* **Icon**: задает иконку формы
* **Location**: определяет положение по отношению к верхнему левому углу экрана, если для свойства StartPosition установлено значение Manual
* **MaximizeBox**: указывает, будет ли доступна кнопка максимизации окна в заголовке формы
* **MinimizeBox**: указывает, будет ли доступна кнопка минимизации окна
* **MaximumSize**: задает максимальный размер формы
* **MinimumSize**: задает минимальный размер формы
* **Opacity**: задает прозрачность формы
* **Size**: определяет начальный размер формы
* **StartPosition**: указывает на начальную позицию, с которой форма появляется на экране
* **Text**: определяет заголовок формы
* **TopMost**: если данное свойство имеет значение true, то форма всегда будет находиться поверх других окон
* **Visible**: видима ли форма, если мы хотим скрыть форму от пользователя, то можем задать данному свойству значение false
* **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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | 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:

|  |  |
| --- | --- |
| 1 | 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:

* **Manual**: Положение формы определяется свойством Location
* **CenterScreen**: Положение формы в центре экрана
* **WindowsDefaultLocation**: Позиция формы на экране задается системой Windows, а размер определяется свойством Size
* **WindowsDefaultBounds**: Начальная позиция и размер формы на экране задается системой Windows
* **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:

|  |  |
| --- | --- |
| 1 | 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:

|  |  |
| --- | --- |
| 1  2 | 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:

|  |  |
| --- | --- |
| 1 | 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:

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

Например, расположим форму по центру экрана:

|  |  |
| --- | --- |
| 1 | this.StartPosition = FormStartPosition.CenterScreen; |