Тема 3. Введение в Windows Forms.

Цель занятия:

Изучить основы Windows Forms, работу с элементами управления, создание простых приложений.

Учебные вопросы:

1. Введение.
2. Структура проекта Windows Forms.
3. Основные элементы управления Windows Forms.
4. Архитектура приложения Windows Forms.
5. Работа с компонентами ввода и вывода.
6. Настройка свойств элементов.
7. Обработка событий.
8. Заключение.

1. Введение.

Windows Forms — это библиотека для создания графического интерфейса пользователя (GUI) в Windows-приложениях.

Она была представлена в 2002 году как часть первой версии .NET Framework.

Она стала заменой старой технологии разработки графических интерфейсов для Windows, такой как Win32 API и MFC (Microsoft Foundation Classes).

WinForms была разработана для упрощения создания desktop-приложений с использованием языка C# и других .NET-языков.

Основные принципы:

- Использование форм (Forms) как контейнеров для элементов управления.
- Работа с элементами управления (кнопки, текстовые поля, списки и т.д.).
- Обработка событий (например, нажатие кнопки, изменение текста).
- Простая интеграция с .NET-библиотеками (ADO.NET для работы с базами данных, System.Drawing для графики и т.д.).

Преимущества Windows Forms:

- Простота использования. WinForms имеет интуитивно понятный API, что делает ее идеальной для начинающих разработчиков..
- Быстрая разработка. Готовые элементы управления (кнопки, текстовые поля, списки) позволяют быстро создавать функциональные приложения.
- Широкая поддержка. WinForms поддерживается всеми версиями .NET Framework и .NET Core/.NET 5+. Большое количество документации и примеров.
- Интеграция с .NET. Легко интегрируется с другими .NET-библиотеками (например, ADO.NET для работы с базами данных).
- Стабильность. WinForms — это зрелая технология, которая используется уже более 20 лет.

Недостатки Windows Forms:

- Устаревший дизайн. WinForms не поддерживает современные возможности графики, такие как анимация, прозрачность и сложные визуальные эффекты.
- Ограниченная кроссплатформенность. Хотя WinForms была портирована на .NET Core/.NET 5+, она изначально разрабатывалась для Windows и имеет ограниченную поддержку других платформ.
- Отсутствие гибкости. WinForms не поддерживает современные подходы к разработке интерфейсов, такие как MVVM (Model-View-ViewModel).
- Ограниченная поддержка высоких DPI. WinForms может некорректно отображаться на экранах с высоким разрешением.
- Меньше возможностей для кастомизации. По сравнению с WPF, WinForms предлагает меньше возможностей для создания нестандартных интерфейсов.

Сравнение с другими технологиями GUI

1. Windows Forms vs WPF (Windows Presentation Foundation)

Критерий	Windows Forms	WPF
Графика	Простая, основана на GDI+.	Современная, основана на DirectX.
Гибкость	Ограниченная поддержка кастомизации.	Высокая гибкость (XAML, стили, шаблоны).
Производительность	Высокая для простых интерфейсов.	Высокая, но требует больше ресурсов.
Кроссплатформенность	Ограниченная (в основном Windows).	Ограниченная (в основном Windows).
Сложность изучения	Простая.	Более сложная из-за XAML и MVVM.
Поддержка DPI	Ограниченная.	Полная поддержка высоких DPI.

Сравнение с другими технологиями GUI

2. Windows Forms vs UWP (Universal Windows Platform)

Критерий	Windows Forms	UWP
Графика	Простая, основана на GDI+.	Современная, основана на DirectX.
Кроссплатформенность	Ограниченная (в основном Windows).	Кроссплатформенная (Windows 10+).
Поддержка устройств	Только десктоп.	Десктоп, мобильные устройства, Xbox.
Современные функции	Отсутствуют.	Поддержка сенсорных устройств, Live Tiles.
Сложность изучения	Простая.	Средняя (XAML, асинхронное программирование).

Когда использовать Windows Forms?

- Внутренние корпоративные приложения.
- Приложения для работы с оборудованием.
- Приложения для работы с устаревшими системами.
- Простые пользовательские интерфейсы.
- Быстрая разработка прототипов.
- Приложения для образовательных целей.
- Приложения для малого бизнеса
- Приложения для работы с локальными данными
- Приложения для специализированных задач
- Приложения для поддержки старых версий Windows

Когда выбрать другие технологии?

- WPF. Если нужен современный интерфейс с поддержкой анимаций, сложной графики и кастомизации. Для приложений, где важна поддержка высоких DPI.
- UWP/WinUI. Для современных приложений с поддержкой сенсорных устройств и Fluent Design. Если нужно поддерживать Windows 10/11 и другие устройства (Xbox, HoloLens).
- Avalonia/MAUI. Для кроссплатформенных приложений (Windows, macOS, Linux, Android, iOS).

2. Структура проекта Windows Forms.

Структура проекта Windows Forms (WinForms) обычно включает в себя несколько папок и файлов.

Далее рассмотрим основные компоненты проекта.

Основные файлы проекта

Program.cs

Это точка входа приложения.

Содержит метод Main, который запускает приложение.

Пример:

```
static class Program
{
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1()); // Запуск главной формы
    }
}
```

Form1.cs (по умолчанию)

Основной файл формы.

Содержит пользовательский код, который вы пишете для обработки событий и добавления логики.

Пример:

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

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

Form1.Designer.cs

Автоматически генерируемый файл.

Содержит код, который создает и настраивает элементы управления на форме.

Пример:

```
partial class Form1
{
    private System.ComponentModel.IContainer components = null;
    private System.Windows.Forms.Button button1;

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

    private void InitializeComponent()
    {
        this.button1 = new System.Windows.Forms.Button();
        this.SuspendLayout();
        this.button1.Location = new System.Drawing.Point(100, 100);
        this.button1.Name = "button1";
        this.button1.Size = new System.Drawing.Size(75, 23);
        this.button1.TabIndex = 0;
        this.button1.Text = "Нажми меня";
        this.button1.UseVisualStyleBackColor = true;
        this.button1.Click += new System.EventHandler(this.button1_Click);
```

Когда вы добавляете элемент на форму в дизайнере Windows Forms и настраиваете его свойства, Visual Studio автоматически генерирует соответствующий код в файле .Designer.cs

Form1.resx

Файл ресурсов формы.

Содержит локализованные строки, изображения, иконки и другие ресурсы, связанные с формой.

3. Основные элементы управления Windows Forms.

Windows Forms предоставляет множество элементов управления (контролов), которые позволяют создавать интерактивные пользовательские интерфейсы для desktop-приложений.

Кнопка (Button)

Назначение: выполнение действий при нажатии (например, сохранение данных, закрытие формы).

Основные свойства:

- Text: текст на кнопке (например, "Сохранить").
- Enabled: активна ли кнопка (true/false).
- Visible: видима ли кнопка (true/false).
- Событие: Click (обработка нажатия).

Метка (Label)

Назначение: отображение текста (например, подписи к полям).

Основные свойства:

- Text: текст метки (например, "Имя:").
- Font: шрифт текста.
- ForeColor: цвет текста.
- BackColor: цвет фона

Текстовое поле (TextBox)

Назначение: ввод и редактирование текста.

Основные свойства:

- Text: текст в поле.
- Multiline: многострочный режим (true/false).
- ReadOnly: запрет на редактирование (true/false).

Событие: TextChanged (изменение текста).

Многострочный редактор (RichTextBox)

Назначение: ввод и редактирование текста с поддержкой форматирования.

Основные свойства:

- Text: текст в редакторе.
- Font: шрифт текста.
- SelectionColor: цвет выделенного текста.

Особенности: поддерживает вставку изображений, изменение шрифтов и цветов.

MessageBox — это стандартный диалог в Windows Forms, который используется для вывода сообщений пользователю. Он позволяет отображать информационные сообщения, предупреждения, ошибки, а также запрашивать подтверждение действий.

Основные возможности MessageBox:

Вывод сообщений:

- Простые сообщения с текстом и кнопкой "OK".
- Сообщения с заголовком, иконкой и кнопками.

Кнопки: OK, OKCancel, YesNo, YesNoCancel, RetryCancel, AbortRetryIgnore.

Иконки: Information, Warning, Error, Question.

Результат: Возвращает значение типа DialogResult, которое указывает, какую кнопку нажал пользователь (например, Yes, No, OK, Cancel).

```
// Простое сообщение
MessageBox.Show("Привет, мир!");

// Сообщение с заголовком и кнопками
DialogResult result = MessageBox.Show(
    "Вы уверены, что хотите выйти?", // Текст сообщения
    "Подтверждение",                 // Заголовок
    MessageBoxButtons.YesNo,         // Кнопки
    MessageBoxIcon.Question          // Иконка
);

// Обработка результата
if (result == DialogResult.Yes)
{
    MessageBox.Show("Вы выбрали Да!");
}
else
{
    MessageBox.Show("Вы выбрали Нет.");
}
```

Свойства элементов управления в Windows Forms играют ключевую роль в настройке их внешнего вида, поведения и взаимодействия с пользователем. Рассмотрим некоторые основные свойства.

1. Name

Назначение: уникальное имя элемента управления, используемое для обращения к нему в коде.

Тип: **string**.

Примечание: имя должно быть уникальным в пределах формы.

2. Text

Назначение: текст, отображаемый на элементе управления (например, текст кнопки, метки или заголовок формы).

Тип: **string**.

Примечание: для некоторых элементов (например, TextBox) это свойство также используется для хранения введенного текста.

```
button1.Text = "Нажми меня";
label1.Text = "Введите имя:";
```

3. Size

Назначение: размер элемента управления (ширина и высота).

Тип: Size.

Примечание: размер можно задать как в конструкторе, так и через свойства Width и Height.

```
button1.Size = new Size(100, 30); // Ширина = 100, высота = 30
```

4. Location

Назначение: расположение элемента управления на форме (координаты X и Y относительно верхнего левого угла формы).

Тип: **Point**.

Примечание: координаты отсчитываются от верхнего левого угла формы.

```
button1.Location = new Point(50, 50); // X = 50, Y = 50
```

5. Enabled

Назначение: определяет, активен ли элемент управления (можно ли с ним взаимодействовать).

Примечание: неактивные элементы обычно отображаются "серыми" и не реагируют на действия пользователя.

```
button1.Enabled = false; // Кнопка неактивна
```

6. Visible

Назначение: определяет, видим ли элемент управления на форме.

Тип: **bool**.

Примечание: скрытые элементы не отображаются на форме, но остаются в памяти.

```
button1.Visible = false; // Кнопка скрыта
```

7. BackColor

Тип: **Color**.

Назначение: цвет фона элемента управления.

```
button1.BackColor = Color.LightBlue;
```

8. ForeColor

Тип: **Color**.

Назначение: цвет текста элемента управления.

```
button1.ForeColor = Color.Red;
```

9. Font

Тип: Font.

Назначение: шрифт текста элемента управления.

```
button1.Font = new Font("Arial", 12, FontStyle.Bold);
```

10. Anchor

Тип: AnchorStyles (перечисление).

Назначение: определяет, как элемент управления привязывается к краям формы при изменении ее размеров.

```
button1.Anchor = AnchorStyles.Top | AnchorStyles.Left;
```

11. Dock

Тип: DockStyle (перечисление).

Назначение: определяет, как элемент управления "пристыковывается" к краям формы.

```
button1.Dock = DockStyle.Top;
```

12. Cursor

Тип: Cursor.

Назначение: курсор мыши, который отображается при наведении на элемент управления.

```
button1.Cursor = Cursors.Hand;
```

Методы

Методы — это функции, которые можно вызывать для выполнения определенных действий с элементом управления.

Show()

Назначение: делает элемент управления видимым.

Пример:

```
button1.Show();
```

Hide()

Назначение: скрывает элемент управления.

Пример:

```
button1.Hide();
```

Focus()

Назначение: устанавливает фокус на элемент управления.

Пример:

```
textBox1.Focus();
```

События

События — это действия, которые происходят с элементом управления (например, нажатие кнопки или изменение текста). Разработчик может подписаться на эти события и выполнять определенный код в ответ на них.

Click

Назначение: происходит при нажатии на элемент управления (например, кнопку).

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

TextChanged

Назначение: происходит при изменении текста в элементе управления (например, TextBox).

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    label1.Text = textBox1.Text;
}
```

Load

Назначение: происходит при загрузке формы.

```
private void Form1_Load(object sender, EventArgs e)
{
    MessageBox.Show("Форма загружена!");
}
```

MouseEnter

Назначение: происходит, когда указатель мыши попадает на элемент управления.

```
private void button1_MouseEnter(object sender, EventArgs e)
{
    button1.BackColor = Color.Yellow;
}
```

MouseLeave

Назначение: происходит, когда указатель мыши покидает элемент управления.

```
private void button1_MouseLeave(object sender, EventArgs e)
{
    button1.BackColor = Color.LightGray;
}
```

Как подписаться на события?

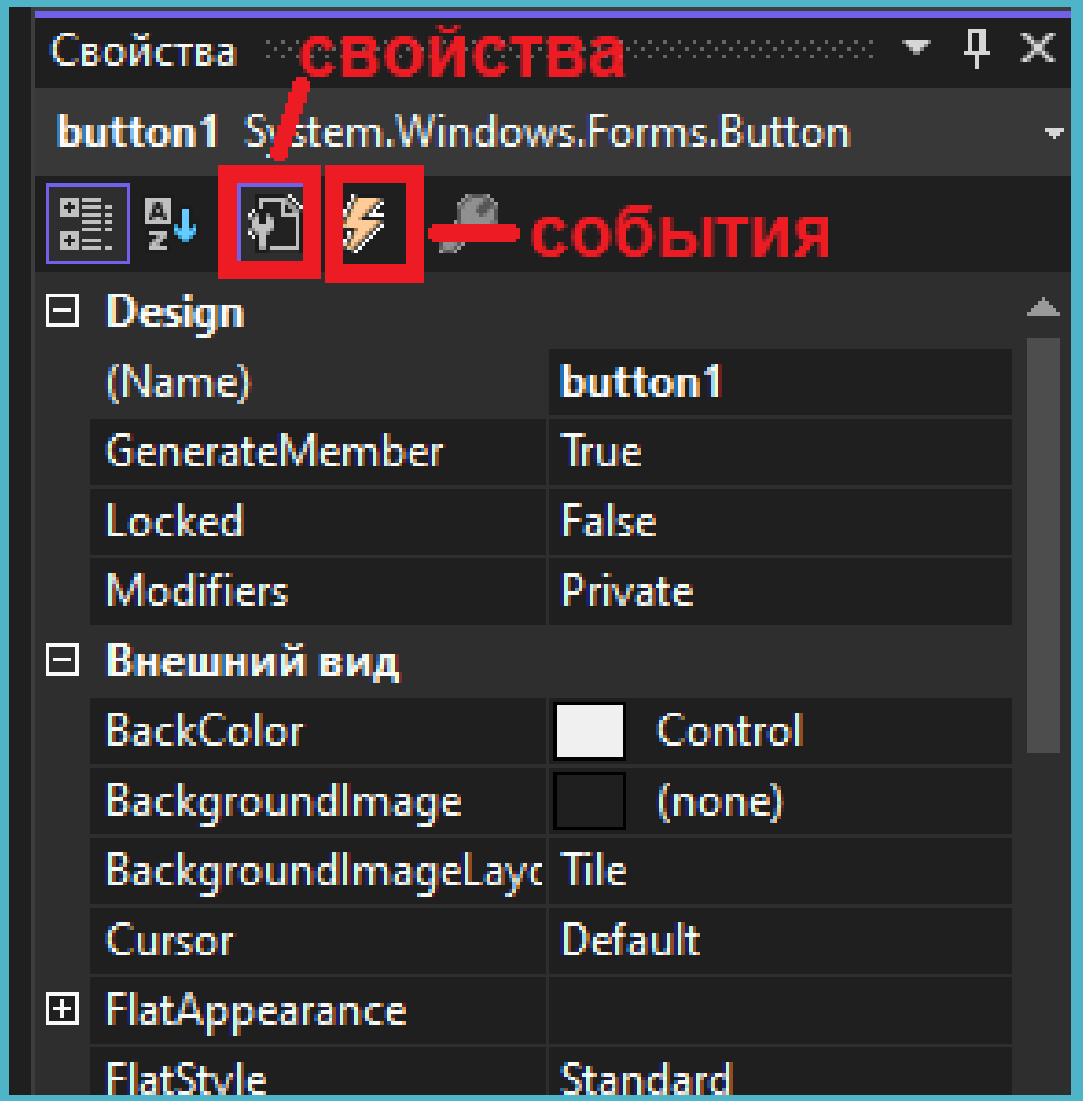
События можно подключать через конструктор Visual Studio или вручную в коде.

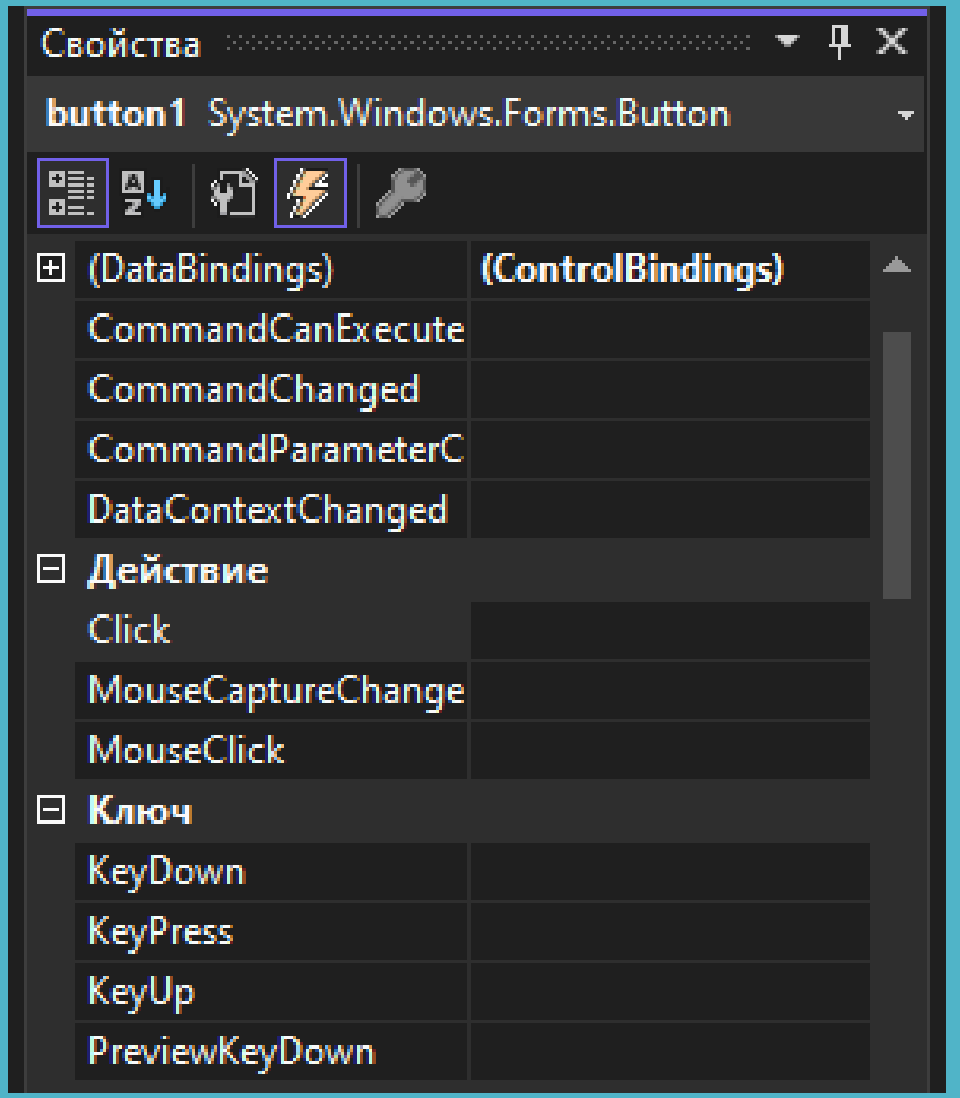
1. Через конструктор Visual Studio:

- Выберите элемент управления на форме.
- В окне Свойства перейдите на вкладку События.
- Дважды щелкните на нужное событие (например, Click), чтобы создать обработчик.

2. Вручную в коде:

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


Свойства
свойства
button1 System.Windows.Forms.Button
события
Design
(Name)
button1
GenerateMember
True
Locked
False
Modifiers
Private
Внешний вид
BackColor
Control
BackgroundImage
(none)
BackgroundImageLayc
Tile
Cursor
Default
FlatAppearance
FlatStyle
Standard


Свойства
button1 System.Windows.Forms.Button
(DataBindings)
(ControlBindings)
CommandCanExecute
CommandChanged
CommandParameterC
DataContextChanged
Действие
Click
MouseCaptureChange
MouseClick
Ключ
KeyDown
KeyPress
KeyUp
PreviewKeyDown

4. Архитектура приложения Windows Forms.

Основная архитектура приложения построена вокруг формы (Form) как главного контейнера и элементов управления (Controls), являющихся дочерними объектами.

Форма (Form)

Назначение: форма является основным контейнером для всех элементов управления. Это окно, которое отображается пользователю.

Основные свойства:

- Text: заголовок формы.
- Size: размер формы.
- BackColor: цвет фона формы.
- StartPosition: начальное положение формы на экране.

События:

- Load: происходит при загрузке формы.
- FormClosing: происходит перед закрытием формы.
- Paint: происходит при отрисовке формы.

Элементы управления (Controls)

Назначение: элементы управления (например, кнопки, текстовые поля, метки) используются для взаимодействия с пользователем.

Основные свойства:

- Text: текст, отображаемый на элементе.
- Size: размер элемента.
- Location: расположение элемента на форме.
- Enabled: активен ли элемент.
- Visible: видим ли элемент.

События:

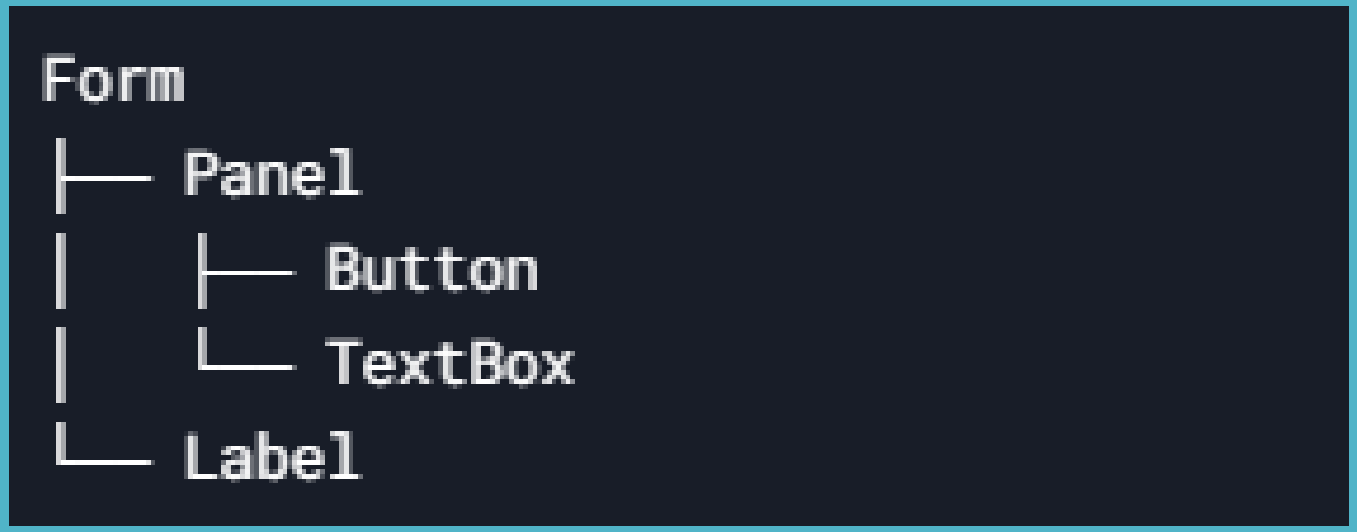
- Click: происходит при нажатии на элемент.
- TextChanged: происходит при изменении текста.
- MouseEnter: происходит при наведении мыши на элемент.

Иерархия элементов управления

Форма является корневым элементом, который содержит другие элементы управления.

Элементы управления могут содержать другие элементы управления (например, Panel, GroupBox).

Пример иерархии:



5. Работа с компонентами ввода и вывода.

TextBox — это один из наиболее часто используемых элементов управления в Windows Forms, позволяющий пользователю вводить и отображать одну строку текста.

Основные свойства TextBox и их назначение

Text: Содержит текущий текст, отображаемый в элементе управления.

MaxLength: Ограничивает максимальное количество символов, которые пользователь может ввести.

PasswordChar: Заменяет каждый введенный символ указанным символом (обычно используется для создания полей ввода паролей).

ReadOnly: Делает элемент управления только для чтения, пользователь не сможет вводить в него текст.

Multiline: Разрешает ввод нескольких строк текста.

Настройка свойств в дизайнере форм

Добавление TextBox на форму: Перетащите элемент TextBox с панели инструментов на форму.

Изменение свойств в окне свойств: Выделите TextBox и в окне свойств (Properties) найдите нужное свойство и измените его значение.

Настройка свойств в коде

```
public Form1()
{
    InitializeComponent();

    // Создание нового TextBox
    TextBox textBox1 = new TextBox();

    // Настройка свойств
    textBox1.Text = "Введите текст";
    textBox1.MaxLength = 20;
    textBox1.PasswordChar = '*';
    textBox1.ReadOnly = true;
    textBox1.Multiline = true;
    textBox1.ScrollBars = ScrollBars.Vertical;
    textBox1.Font = new Font("Arial", 12);
    textBox1.ForeColor = Color.Blue;
    textBox1.BackColor = Color.Yellow;
    textBox1.BorderStyle = BorderStyle.FixedSingle;
    textBox1.TextAlign = HorizontalAlignment.Center;

    // Добавление TextBox на форму
    this.Controls.Add(textBox1);
}
```

RichTextBox — это элемент управления в Windows Forms, позволяющий отображать и редактировать форматированный текст. В отличие от обычного TextBox, который ограничивается простой текстовой строкой, RichTextBox предоставляет широкий набор возможностей для форматирования, таких как изменение шрифта, цвета, выравнивания и многое другое.

Основные свойства

Text: Содержит весь текст в элементе управления в виде простой строки.

Rtf: Содержит текст в формате RTF (Rich Text Format), который позволяет сохранять форматирование.

Font: Устанавливает шрифт для отображения текста.

ForeColor: Устанавливает цвет переднего плана (текста).

BackColor: Устанавливает цвет фона.

ReadOnly: Делает элемент управления только для чтения.

Multiline: Разрешает ввод нескольких строк текста.

WordWrap: Определяет, будет ли текст автоматически переноситься на новую строку.

SelectionFont: Устанавливает шрифт для выделенного текста.

SelectionColor: Устанавливает цвет для выделенного текста.

SelectionBackColor: Устанавливает цвет фона для выделенного текста.

Основные события

TextChanged: Возникает при изменении текста в элементе управления.

SelectionChanged: Возникает при изменении выделения текста.

Форматирование текста

RichTextBox позволяет форматировать текст на уровне символов и абзацев. Для форматирования текста обычно используется выделение (Selection).

Изменение шрифта: Свойство SelectionFont позволяет изменить шрифт, размер и стиль выделенного текста.

Изменение цвета: Свойства SelectionColor и SelectionBackColor позволяют изменить цвет текста и фона соответственно.

Выравнивание: С помощью свойств SelectionAlignment можно изменить выравнивание текста (по левому краю, по центру, по правому краю).

Отступы: Свойства SelectionIndent и SelectionRightIndent позволяют установить отступы для абзацев.

Другие стили: Можно применять различные стили, такие как жирное, курсив, подчеркивание и т.д.

Дополнительные возможности

Загрузка и сохранение файлов: RichTextBox поддерживает загрузку и сохранение файлов в форматах RTF и TXT.

Поиск и замена текста: Методы Find и Replace позволяют выполнять поиск и замену текста.

Вставка изображений: Можно вставлять изображения в RichTextBox.

Вывод информации в Windows Forms

В Windows Forms вывод информации можно организовать с помощью элементов управления, таких как **Label**, или диалогового окна **MessageBox**.

Основные свойства **Label:**

Text: текст, отображаемый на метке.

Font: шрифт текста.

ForeColor: цвет текста.

BackColor: цвет фона метки.

AutoSize: автоматическое изменение размера метки в зависимости от содержимого.

Пример использования Label

```
public partial class MainForm : Form
{
    public MainForm()
    {
        InitializeComponent();

        // Создание метки
        Label label1 = new Label();
        label1.Text = "Привет, мир!";
        label1.Location = new Point(50, 50);
        label1.Font = new Font("Arial", 12);
        label1.ForeColor = Color.Blue;
        label1.AutoSize = true;

        // Добавление метки на форму
        this.Controls.Add(label1);
    }
}
```

Основные методы **MessageBox:**

MessageBox.Show(): отображает диалоговое окно с сообщением.

Параметры:

text: текст сообщения.

caption: заголовок окна.

buttons: кнопки (например, OK, YesNo).

icon: иконка (например, Information, Warning, Error).

Примеры использования MessageBox

Простое сообщение

```
MessageBox.Show("Привет, мир!");
```

Сообщение с заголовком

```
MessageBox.Show("Привет, мир!", "Сообщение");
```

Сообщение с кнопками и иконкой

```
DialogResult result = MessageBox.Show("Вы уверены?", "Подтверждение", MessageBoxButtons.YesNo, M
essageBoxIcon.Question);
if (result == DialogResult.Yes)
{
    MessageBox.Show("Вы выбрали Да!");
}
else
{
    MessageBox.Show("Вы выбрали Нет!");
}
```

6. Настройка свойств элементов.

Настройка свойств элементов управления в Windows Forms может выполняться двумя способами:

1. **Через окно Properties** в Visual Studio.
2. **Программно** в коде.

Кроме того, важно понимать, как привязывать события к элементам управления.

Привязка событий к элементам.

Двойной щелчок по элементу для создания обработчика события.

Выберите элемент на форме (например, кнопку).

Дважды щелкните по элементу. Visual Studio автоматически создаст обработчик события (например, button1_Click) и откроет файл кода.

Пример:

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

Также можно привязать событие к элементу управления через панель **Свойства (Properties)**, используя вкладку **События (Events)**.

Выберите элемент управления на форме (например, кнопку, текстовое поле и т.д.).

Откройте панель Свойства:

Нажмите F4.

Или выберите в меню: View → Properties Window.

В панели Свойства найдите и нажмите на иконку с изображением молнии Events Icon. Это переключит панель на вкладку События.

Вы увидите список всех событий, доступных для выбранного элемента управления.

Привязка события

Найдите нужное событие в списке (например, Click, TextChanged, MouseEnter и т.д.).

Дважды щелкните на пустом поле рядом с именем события. Visual Studio автоматически:

Создаст обработчик события в коде.

Перейдет в файл кода, где можно написать логику обработки события.

Ручное создание обработчиков в коде

Вы можете вручную создать обработчик события и привязать его к элементу.

Пример:

```
    // Создание кнопки
    Button button1 = new Button();
    button1.Text = "Нажми меня";
    button1.Location = new Point(50, 50);

    // Привязка события Click
    button1.Click += Button1_Click;

    // Добавление кнопки на форму
    this.Controls.Add(button1);
}

private void Button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Кнопка нажата!");
}
```

7. Обработка событий.

Обработка событий — это ключевая часть создания интерактивных приложений.

События позволяют реагировать на действия пользователя, такие как нажатие кнопки, изменение текста или выбор элемента из списка.

Основные события

Click для кнопок

Назначение: происходит при нажатии на кнопку.

Пример:

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

TextChanged для текстовых полей

Назначение: происходит при изменении текста в текстовом поле.

Пример:

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    label1.Text = "Введено: " + textBox1.Text;
}
```

SelectedIndexChanged для списков

Назначение: происходит при изменении выбранного элемента в списке (например, ComboBox или ListBox).

Пример:

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    MessageBox.Show("Выбран элемент: " + comboBox1.SelectedItem);
}
```

8. Заключение.

Мы рассмотрели ключевые аспекты разработки приложений с использованием Windows Forms.

Основные понятия

Форма (Form)

Форма — это основное окно приложения, которое служит контейнером для всех элементов управления.

Свойства:

- Text: заголовок формы.
- Size: размер формы.
- BackColor: цвет фона.

События:

- Load: происходит при загрузке формы.
- FormClosing: происходит перед закрытием формы.

Элементы управления (Controls)

Элементы управления — это компоненты, которые позволяют взаимодействовать с пользователем (например, кнопки, текстовые поля, метки).

Основные элементы:

Button:кнопка.

Label: метка для отображения текста.

TextBox: текстовое поле для ввода текста.

RichTextBox: многострочный текстовый редактор

MessageBox: диалоговое окно

Свойства:

Text: текст элемента.

Size: размер элемента.

Location: расположение элемента на форме.

Enabled: активен ли элемент.

Visible: видим ли элемент.

Свойства

Свойства позволяют настраивать внешний вид и поведение элементов управления.

Примеры:

BackColor: цвет фона.

ForeColor: цвет текста.

Font: шрифт текста.

MaxLength: максимальное количество символов в текстовом поле.

PasswordChar: символ для отображения в поле пароля.

Методы

Методы — это функции, которые выполняют определенные действия с элементами управления.

Примеры:

Show(): делает элемент видимым.

Hide(): скрывает элемент.

Focus(): устанавливает фокус на элемент.

BringToFront(): перемещает элемент на передний план.

События

События — это действия, которые происходят с элементами управления (например, нажатие кнопки, изменение текста).

Основные события:

Click: происходит при нажатии на элемент.

TextChanged: происходит при изменении текста.

Обработка событий:

Обработчики событий создаются автоматически через панель События или вручную в коде.

Список литературы:

1. Введение в Windows Forms
2. Создание приложения Windows Forms в Visual Studio
3. Знакомство с Windows Forms

Материалы лекций:

https://github.com/ShViktor72/Education

Обратная связь:

colledge20education23@gmail.com

Домашнее задание:

Задание 1: Создание простого приложения с использованием Windows Forms

Создайте новое приложение Windows Forms в Visual Studio.

Добавьте на форму следующие элементы управления:

- Кнопку (Button) с текстом "Нажми меня".
- Метку (Label) с текстом "Привет, мир!".
- Текстовое поле (TextBox) для ввода текста.

Настройте свойства элементов:

- Установите цвет фона кнопки на светло-голубой.
- Установите шрифт метки на Arial, размер 12, жирный.

Сделайте текстовое поле многострочным (Multiline).

Напишите обработчик события для кнопки, который будет выводить в MessageBox текст, введенный в текстовое поле.

Задание 2: Работа с событиями

Создайте форму с двумя кнопками: "Показать" и "Скрыть".

Добавьте на форму метку с текстом "Это видимая метка".

Напишите обработчики событий для кнопок:

- При нажатии на кнопку "Показать" метка должна становиться видимой.
- При нажатии на кнопку "Скрыть" метка должна скрываться.

Добавьте обработчик события MouseEnter для метки, который будет изменять цвет фона метки на желтый при наведении курсора мыши.

Задание 3: Работа с TextBox и RichTextBox

Создайте форму с двумя текстовыми полями: обычное TextBox и RichTextBox.

Добавьте кнопку "Копировать".

Напишите обработчик события для кнопки, который будет копировать текст из TextBox в RichTextBox.

Добавьте возможность форматирования текста в RichTextBox:

- Создайте кнопки для изменения шрифта (жирный, курсив, подчеркивание).
- Добавьте возможность изменения цвета текста и фона выделенного текста.

Задание 4: Использование MessageBox

Создайте форму с кнопкой "Подтверждение".

Напишите обработчик события для кнопки, который будет выводить MessageBox с вопросом "Вы уверены?" и кнопками "Да" и "Нет".

В зависимости от выбора пользователя (Да/Нет), выведите соответствующее сообщение в MessageBox (например, "Вы выбрали Да!" или "Вы выбрали Нет.").