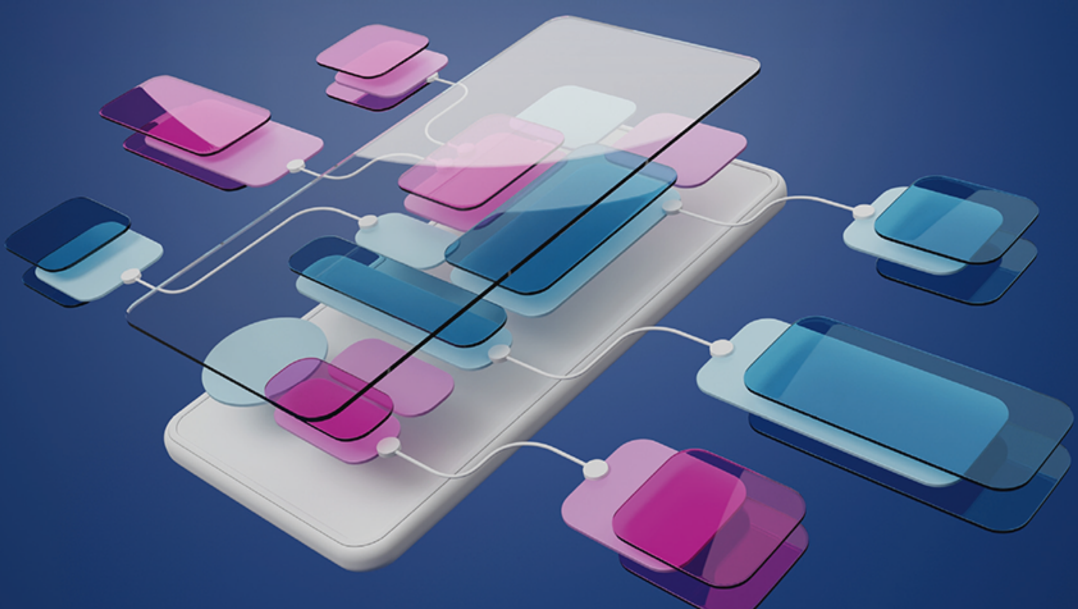
Łukasz Sosna

VISUAL STUDIO 2022

C# I .NET

Programowanie kontrolek



Helion

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki
Grafika na okładce została wykorzystana za zgodą Shutterstock.com.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/vs22ko
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-9140-6

Copyright © Helion S.A. 2023

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Wstęp **7**

ROZDZIAŁ 1. **Czym jest Visual Studio 2022** **9**

Instalacja oprogramowania oraz wybranie odpowiednich komponentów 9
Rejestracja nowego konta 12
Instalacja Microsoft .NET SDK 13
Instalacja .NET Runtime 14
Pierwsze uruchomienie Visual Studio Community 14
Tworzenie nowego projektu 16
Środowisko Visual Studio 17
Praca nad stworzonym formularzem i uruchamianie pierwszego programu 18
Zapisywanie projektu 21
Otwieranie projektu 21
Zamykanie projektu 21
Zamykanie edytora 22

ROZDZIAŁ 2. **Narzędzia Visual Studio** **23**

Przechodzenie do określonej linii 23
Przechodzenie do określonego pliku 23
Znajdowanie fragmentu kodu 24
Zamiana fragmentu kodu 25
Poszukiwanie tekstu w plikach 26
Zamiana tekstu w plikach 27
Cofanie zmian 28
Powtarzanie wycofanych zmian 29
Kopiowanie i wklejanie 29
Wycinanie i wklejanie 29
Zaznaczanie całego tekstu 30
Dodawanie zakładki 30
Przechodzenie do zakładki 31
Usuwanie zakładki 31
Usuwanie wszystkich zakładek 31
Przełączenie do kodu programu 31
Przełączenie do projektanta programu 32
Panel: Solution Explorer 32
Panel: Team Explorer 33
Panel: Server Explorer 34
Panel: Bookmark Window 34
Panel: Git Changes 34
Panel: Call Hierarchy 35

Kup książkę Poleć książkę

Panel: Class View 35
Panel: Object Browser 36
Panel: Error List 37
Panel: Output 38
Panel: Toolbox 38
Budowanie projektu 39
Zbudowanie projektu 40
Przebudowanie projektu 40
Czyszczenie budowy projektu 40
Uruchamianie projektu 41
Uruchamianie projektu ze śledzeniem błędów 41
Uruchamianie projektu bez śledzenia błędów 41

ROZDZIAŁ 3. **Programowanie aplikacji poprzez kontrolki** **43**

Przycisk (Button) 43
Tworzenie przycisku oraz przypisywanie do niego tekstu i lokalizacji 43
Określanie wysokości i szerokości 44
Wyrównanie tekstu względem granic przycisku 46
Rezultat naciśnięcia przycisku 47
Przypisywanie akcji 48
Pole tekstowe (TextBox) 50
Dodawanie pola tekstowego 50
Ustawianie szerokości i wysokości pola 51
Dodawanie do pola tekstowego możliwości pisania w wielolinii 52
Ustawianie akcji w celu wyświetlenia zawartości pola 54
Etykiety (Label) 56
Dodawanie pola opisu 56
Ustawianie szerokości pola opisu 58
Definiowanie akcji dla pola 59
Etykieta z odnośnikiem (LinkLabel) 61
Szerokość etykiety dopasowana do zawartości 63
Przejście do strony po kliknięciu etykiety z łączem 64
Pola wyboru (Checkbox) 67
Dodawanie pola wyboru do formularza 67
Ustawianie szerokości pola wyboru 68
Zaznaczanie niektórych opcji wyboru 70
Ustawianie akcji dla pól wyboru 71
Lista z polem wyboru (CheckedListBox) 75
Dodawanie listy do formularza 75
Wypełnianie listy danymi 76
Dodawanie akcji po wybraniu elementów 77
Lista (ComboBox) 80
Tworzymy listę rozwijaną 80
Uzupełnianie listy danymi oraz zmiana jej wyglądu 82
Dodawanie akcji po wybraniu elementu z listy 83

Kup książkę Poleć książkę

Wybieranie daty i czasu (DateTimerPicker) 86
Tworzenie pola wyboru daty 86
Ustawianie daty minimalnej oraz maksymalnej 87
Ustawianie formatu daty 88
Wyświetlanie daty w osobnym oknie przez wywołanie akcji 89
Lista wyboru (ListBox) 92
Dodawanie nowej listy do formularza 92
Wypełnianie listy danymi 93
Ustawienie możliwości wybrania tylko jednego elementu 94
Wybieranie domyślnej pozycji 95
Przycisk z akcją wyświetlającą wybór z listy w osobnym oknie 97
Lista elementów (ListView) 100
Dodawanie listy do formularza 100
Dodawanie opcji do listy w celu poprawy wyświetlania rekordów 102
Wyłączanie opcji zaznaczania kilku rekordów 104
Dodawanie linii w celu rozdzielenia rekordów 106
Dodawanie zaznaczania całego rekordu danych 109
Tworzenie akcji wyświetlającej nazwisko po zaznaczeniu w liście i kliknięciu przycisku 111
Pole numeryczne z kontrolkami do zmiany wartości (NumericUpDown) 115
Dodawanie do formularza pola numerycznego 116
Ustawienie zakresu liczb w polu numerycznym 117
Ustawienie wybranej wartości w polu numerycznym 118
Zmiana wartości w polu numerycznym powoduje wywołanie metody 119
Akcja przepisywania wartości wybranej w polu do pola tekstowego 122
Obraz (PictureBox) 126
Pola wyboru z możliwością wybrania tylko jednej pozycji (RadioButton) 130
Tworzenie pól wyboru zebranych w jednej kontrolce 130
Wywoływanie metody w trakcie zmiany wyboru pola 133
Wyświetlanie wybranych opcji w nowym oknie 137
Pomoc dla użytkownika (ToolTip) 143
Grupowanie elementów (GroupBox) 144
Podział okna (SplitContainer) 147
Panele (TabControl) 149
Tworzenie podziału głównego okna na panele 149
Dodawanie przycisku oraz akcji dla jednej opcji 151
Menu (MenuStrip) 154
Lista obrazów (ImageList) 157
Wybór koloru (ColorDialog) 159
Wybór czcionki (FontDialog) 162
Przeglądanie folderów (FolderBrowserDialog) 165
Wybór pliku do otwarcia (OpenFileDialog) 167
Wybór lokalizacji do zapisania pliku (SaveFileDialog) 171

Podsumowanie **176**

Kup książkę Poleć książkę

ROZDZIAŁ 3. **Programowanie aplikacji poprzez kontrolki**

Kontrolki to elementy aplikacji, którym w książkach dotyczących programowania nie poświęcano zbyt wiele miejsca. Najwyżej kilka stron oszczędnego omówienia. Dzięki kontrolkom można zbudować aplikację — to dzięki nim powstaje interfejs użytkownika. Któż budując nową aplikację, nie potrzebował pola do wprowadzania tekstu, przycisku, po którego kliknięciu była wykonywana jakaś akcja, czy też nowego okna wyświetlającego informacje na temat programu.

Utwórzmy nowy projekt o nazwie *ButtonForm*.

Przycisk (Button)

Przycisk jest elementem, któremu można przypisać akcję do wykonania po kliknięciu przycisku.

Tworzenie przycisku oraz przypisywanie do niego tekstu i lokalizacji

Utworzenie przycisku nie będzie wcale trudne. Wystarczy po prostu określić jego nazwę przypisaną do klasy przycisku, następnie podać jego lokalizację i wpisać tekst do jego zawartości. Na koniec musimy dodać przycisk do formularza (listing 3.1).

LISTING 3.1. Dodawanie przycisku do formularza

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

            this.Text = "ButtonForm";
            this.Width = 600;
            this.Height = 400;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;
```

Kup książkę Poleć książkę

```
            // Button
        Button firstButton = new Button();
        firstButton.Location = new Point(12, 12);

            firstButton.Text = "Click me!";

            Controls.Add(firstButton);
        }

    }
}
```

Aby dodać nowy przycisk:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) formularza.
3. Ukrywamy przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) naszej aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość aplikacji będzie można zmieniać także przez przeciąganie jej aktywnych brzegów, dlatego musimy ustawić `this.FormBorderStyle` na obramowanie pojedyncze.
6. Definiujemy obiekt przycisku, w którym wstawiamy naszą nazwę elementu, a następnie po znaku równości definiujemy utworzenie nowego elementu: `Button firstButton = new Button()`.
7. Następnie we właściwościach, które ma zdefiniowany przez nas element, wskazujemy jako lokalizację punkt określony przez podanie punktów `X` i `Y` w naszym formularzu (`firstButton.Location`).
8. Teraz przyszedł czas na wprowadzenie do przycisku napisu, który wprowadzamy we właściwości `firstButton.Text`.
9. Po zakończeniu tworzenia przycisku trzeba go jeszcze dodać do bieżącego formularza przez dodanie nowej kontrolki: `Controls.Add(firstButton)`.
10. Teraz wystarczy uruchomić program, aby zobaczyć efekt swojej pracy.

Określanie wysokości i szerokości

W poprzednim punkcie utworzyliśmy przycisk. Jednak jest on zbyt mały, aby zmieścić cały napis. Z pomocą przyjdą nam tutaj możliwości nadania mu rozmiarów przez określenie jego szerokości i wysokości (listing 3.2).

LISTING 3.2. Ustawianie szerokości i wysokości tworzonego przycisku

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

```
            this.Text = "ButtonForm";
                  this.Width = 600;
                  this.Height = 400;
                  this.MaximizeBox = false;
                  this.MinimizeBox = false;
                  this.StartPosition = FormStartPosition.CenterScreen;
                  this.FormBorderStyle = FormBorderStyle.FixedSingle;

                  // Button
            Button firstButton = new Button();
            firstButton.Location = new Point(12, 12);
                  firstButton.Size = new Size(554, 320);

                  firstButton.Text = "Click me!";

                  Controls.Add(firstButton);
              }
          }
      }
```

Aby ustawić wysokość i szerokość przycisku:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) formularza.
3. Ukrywamy przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) naszej aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość aplikacji będzie można zmieniać także przez przeciąganie jej aktywnych brzegów, dlatego musimy ustawić `this.FormBorderStyle` na obramowanie pojedyncze.
6. Definiujemy obiekt przycisku, w którym wstawiamy nazwę elementu, a następnie po znaku równości definiujemy utworzenie nowego elementu: `Button firstButton = new Button()`.
7. Następnie we właściwościach, które ma zdefiniowany przez nas element, wskazujemy jako lokalizację punkt określony przez podanie punktów `X` i `Y` w formularzu (`firstButton.Location`).
8. Określamy także szerokość oraz wysokość naszego przycisku (`firstButton.Size`).
9. Teraz przyszedł czas na wprowadzenie do przycisku napisu (`firstButton.Text`).
10. Po zakończeniu tworzenia przycisku trzeba go jeszcze dodać do bieżącego formularza przez dodanie nowej kontrolki: `Controls.Add(firstButton)`.
11. Teraz wystarczy uruchomić program, aby zobaczyć efekt swojej pracy.

Wyrównanie tekstu względem granic przycisku

Przycisk ma opcję wyrównania tekstu, który jest w nim zawarty, w zależności od ustawień. W ten sposób można wyrównać tekst nie tylko do środka, do lewej czy do prawej, ale także względem narożników (listing 3.3).

LISTING 3.3. Wyrównywanie względem rogów przycisku

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

            this.Text = "ButtonForm";
            this.Width = 600;
            this.Height = 400;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Button
            Button firstButton = new Button();
            firstButton.Location = new Point(12, 12);
            firstButton.Size = new Size(554, 320);

            firstButton.Text = "Click me!";

            firstButton.TextAlign = ContentAlignment.BottomLeft;

            Controls.Add(firstButton);
        }
    }
}
```

Aby wyrównać tekst względem rogu przycisku:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) formularza.
3. Ukrywamy przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) naszej aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość aplikacji będzie można zmieniać także przez przeciąganie jej aktywnych brzegów, dlatego musimy ustawić `this.FormBorderStyle` na obramowanie pojedyncze.
6. Definiujemy obiekt przycisku, w którym wstawiamy nazwę elementu, a następnie po znaku równości definiujemy utworzenie nowego elementu: `Button firstButton = new Button()`.

Kup książkę Poleć książkę

7. Następnie we właściwościach, które ma zdefiniowany przez nas element, wskazujemy jako lokalizację punkt określony przez podanie punktów `X` i `Y` w formularzu (`firstButton.Location`).
8. Określamy także szerokość oraz wysokość naszego przycisku (`firstButton.Size`).
9. Teraz przyszedł czas na wprowadzenie do przycisku napisu (`firstButton.Text`).
10. Wyrównanie tekstu ustawiamy przez wpisanie odpowiedniej wartości we właściwości `firstButton.TextAlign`.
11. Po zakończeniu tworzenia przycisku trzeba go jeszcze dodać do bieżącego formularza przez dodanie nowej kontrolki: `Controls.Add(firstButton)`.
12. Teraz wystarczy uruchomić program, aby zobaczyć efekt swojej pracy.

Rezultat naciśnięcia przycisku

Naciśnięcie przycisku zwraca rezultat, który możemy przechwycić jako informację zwrotną z danego komunikatu czy też okna utworzonego w celu zadania zapytania spoza zestawu domyślnych w Visual Studio (listing 3.4).

LISTING 3.4. Zwracanie przez przycisk wartości, na przykład po jego kliknięciu

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

            this.Text = "ButtonForm";
            this.Width = 600;
            this.Height = 400;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Button
            Button firstButton = new Button();
            firstButton.Location = new Point(12, 12);
            firstButton.Size = new Size(554, 320);

            firstButton.Text = "Click me!";

            firstButton.TextAlign = ContentAlignment.BottomLeft;

            firstButton.DialogResult = DialogResult.OK;

            Controls.Add(firstButton);
        }
    }
}
```

Kup książkę Poleć książkę

Aby przycisk zwracał wartość:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) formularza.
3. Ukrywamy przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) naszej aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość aplikacji będzie można zmieniać także przez przeciąganie jej aktywnych brzegów, dlatego musimy ustawić `this.FormBorderStyle` na obramowanie pojedyncze.
6. Definiujemy obiekt przycisku, w którym wstawiamy nazwę elementu, a następnie po znaku równości definiujemy utworzenie nowego elementu: `Button firstButton = new Button()`.
7. Następnie we właściwościach elementu wskazujemy jako lokalizację punkt określony przez podanie punktów `X` i `Y` na naszym formularzu (`firstButton.Location`).
8. Określamy także szerokość oraz wysokość naszego przycisku (`firstButton.Size`).
9. Wprowadzamy napis do wyświetlenia na przycisku (`firstButton.Text`).
10. Wyrównanie tekstu ustawiamy przez wpisanie odpowiedniej opcji we właściwości `firstButton.TextAlign`.
11. Musimy przypisać we właściwości przycisku wartość, którą ma zwrócić w wyniku kliknięcia. Niech tą wartością będzie `OK`. Przypisujemy ją do właściwości `firstButton.DialogResult`.
12. Po zakończeniu tworzenia przycisku trzeba go jeszcze dodać do bieżącego formularza przez dodanie nowej kontrolki: `Controls.Add(firstButton)`.
13. Teraz wystarczy uruchomić program, aby zobaczyć efekt swojej pracy.

Przypisywanie akcji

Jedną z najważniejszych opcji przycisku jest możliwość przypisania mu akcji (rysunek 3.1), czyli czynności wykonywanej w wyniku kliknięcia go myszą (listing 3.5).

RYSUNEK 3.1. Gotowy program z akcją przypisaną do przycisku

LISTING 3.5. Akcja wywoływana przez kliknięcie przycisku

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

            this.Text = "ButtonForm";
            this.Width = 600;
            this.Height = 400;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Button
            Button firstButton = new Button();
            firstButton.Location = new Point(12, 12);
            firstButton.Size = new Size(554, 320);

            firstButton.Text = "Click me!";

            firstButton.TextAlign = ContentAlignment.BottomLeft;

            firstButton.DialogResult = DialogResult.OK;

            firstButton.Click += new EventHandler(firstButton_Click);

            Controls.Add(firstButton);
        }

        private void firstButton_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Clicked!", "Alert!", MessageBoxButtons.OK);
        }
    }
}
```

Aby przypisać do przycisku akcję:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) formularza.
3. Ukrywamy przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) naszej aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość aplikacji będzie można zmieniać także przez przeciąganie jej aktywnych brzegów, dlatego musimy ustawić `this.FormBorderStyle` na obramowanie pojedyncze.
6. Definiujemy obiekt przycisku, w którym wstawiamy nazwę elementu, a następnie po znaku równości definiujemy utworzenie nowego elementu: `Button firstButton = new Button()`.

Kup książkę Poleć książkę

7. Następnie we właściwościach, które ma zdefiniowany przez nas element, wskazujemy jako lokalizację punkt określony przez podanie punktów X i Y w naszym formularzu (`firstButton.Location`).
8. Określamy także szerokość oraz wysokość naszego przycisku (`firstButton.Size`).
9. Wprowadzamy napis do wyświetlenia na przycisku (`firstButton.Text`).
10. Wyrównanie tekstu ustawiamy przez wpisanie odpowiedniej opcji we właściwości `firstButton.TextAlign`.
11. Musimy przypisać we właściwości przycisku wartość, którą ma zwrócić w wyniku kliknięcia. Niech tą wartością będzie `OK`. Przypisujemy ją do właściwości `firstButton.DialogResult`.
12. Do przycisku należy przypisać nowy element obsługi zdarzenia kliknięcia, a następnie zdefiniować metodę dla akcji przycisku: `firstButton.Click += new EventHandler(firstButton_Click)`.
13. Następnie poza `Form1()` definiujemy metodę do obsługi tego zdarzenia poprzez zapis `private void firstButton_Click(object sender, EventArgs e)`. Metoda ta ma dwa argumenty: pierwszym jest element, który wywołał dane żądanie, a drugim ewentualne błędy.
14. Kolejną rzeczą jest zdefiniowanie wewnątrz metody akcji, którą w tym przypadku będzie wyświetlenie okienka zawierającego informację o tym, że dany przycisk został kliknięty (`MessageBox.Show`).
15. Po zakończeniu tworzenia przycisku trzeba go jeszcze dodać do bieżącego formularza przez dodanie nowej kontrolki: `Controls.Add(firstButton)`.
16. Teraz wystarczy uruchomić program, aby zobaczyć efekt swojej pracy.

Pole tekstowe (TextBox)

Pole to jest przeznaczone do wprowadzania tekstu do tworzonego oprogramowania. Może ono być w formie jednoliniowego komunikatu lub w postaci wielolinowego tekstu. Do wykonania zadań z tego podrozdziału utwórzmy projekt o nazwie *InputForm*.

Dodawanie pola tekstowego

Tworzenie pola tekstowego polega na określeniu jego nazwy, położenia oraz rozmiarów, a następnie dodaniu go do formularza za pomocą odpowiedniej metody (listing 3.6).

LISTING 3.6. Dodawanie pola tekstowego do formularza

```
namespace InputForm
{
    public partial class Form1 : Form
    {
        public Form1()
        {
```

```
            InitializeComponent();

            this.Text = "InputForm";
            this.Width = 347;
            this.Height = 323;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Input
            TextBox firstInput = new TextBox();
            firstInput.Name = "inputBoxInForm";
            firstInput.Location = new Point(12, 12);

            Controls.Add(firstInput);
        }
    }
}
```

Aby uzyskać działający kod:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) naszego formularza.
3. Ustawiamy jako ukryty przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość okna programu będzie można zmieniać także przez przeciąganie jego aktywnych brzegów, dlatego należy ustawić `this.FormBorderStyle` na obramowanie nie pozwalające na takie działanie.
6. Tworzymy pole tekstowe przez przypisanie go do obiektu, w którym zawarta jest definicja pola: `TextBox firstInput = new TextBox()`.
7. Określamy nazwę pola (`firstInput.Name`).
8. Ustawiamy lokalizację przez podanie współrzędnych we właściwości `Location`.
9. Na koniec dodajemy pole tekstowe do formularza przez dodanie go jako kontrolki (`Controls.Add`).

Ustawianie szerokości i wysokości pola

Przy przekształcaniu pola w pole wieloliniowe bardzo ważnym elementem są jego szerokość oraz wysokość. Te parametry pozwolą nam na lepsze dopasowanie jego rozmiarów do planowanych czynności (listing 3.7).

LISTING 3.7. Pole tekstowe wraz z ustawionymi rozmiarami

```
namespace InputForm
{
    public partial class Form1 : Form
    {
        public Form1()
```

```
        {
            InitializeComponent();

            this.Text = "InputForm";
            this.Width = 347;
            this.Height = 323;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Input
            TextBox firstInput = new TextBox();
            firstInput.Name = "inputBoxInForm";
            firstInput.Location = new Point(12, 12);

            firstInput.Width = 300;
            firstInput.Height = 200;

            Controls.Add(firstInput);
        }
    }
}
```

Aby uzyskać działający kod:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) naszego formularza.
3. Ustawiamy jako ukryte przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość okna programu będzie można zmieniać także przez przeciąganie jego aktywnych brzegów, dlatego należy ustawić `this.FormBorderStyle` na obramowanie nie pozwalające na takie działanie.
6. Tworzymy pole tekstowe przez przypisanie go do obiektu, w którym zawarta jest definicja pola: `TextBox firstInput = new TextBox()`.
7. Określamy nazwę pola (`firstInput.Name`).
8. Ustawiamy lokalizację pola przez podanie współrzędnych we właściwości `Location`.
9. Teraz określimy szerokość oraz wysokość pola. Robimy to z użyciem właściwości `Width` i `Height`.
10. Na koniec dodajemy pole tekstowe do formularza przez dodanie go jako kontrolki (`Controls.Add`).

Dodawanie do pola tekstowego możliwości pisania w wielolinii

Kiedy już mamy pole tekstowe w odpowiednim rozmiarze, nie musimy w nim pisać tekstu w jednej linii i formatować go za pomocą rozmiaru czcionki, kroju, pogrubienia, podkreślenia itd. Dla tak przygotowanego pola można włączyć możliwość pisania wieloliniowego (listing 3.8).

LISTING 3.8. Dodawanie dla pola tekstowego możliwości pisania w wielolinii

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

            this.Text = "InputForm";
            this.Width = 347;
            this.Height = 323;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Input
            TextBox firstInput = new TextBox();
            firstInput.Name = "inputBoxInForm";
            firstInput.Location = new Point(12, 12);

            firstInput.Width = 300;
            firstInput.Height = 200;

            firstInput.Multiline = true;

            Controls.Add(firstInput);
        }
    }
}
```

Aby uzyskać działający kod:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) naszego formularza.
3. Ustawiamy jako ukryte przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość okna programu będzie można zmieniać także przez przeciąganie jego aktywnych brzegów, dlatego należy ustawić `this.FormBorderStyle` na obramowanie nie pozwalające na takie działanie.
6. Tworzymy pole tekstowe przez przypisanie go do obiektu, w którym zawarta jest definicja pola: `TextBox firstInput = new TextBox()`.
7. Określamy nazwę pola (`firstInput.Name`).
8. Ustawiamy lokalizację przez podanie współrzędnych we właściwości `Location`.

Kup książkę Poleć książkę

9. Teraz określamy szerokość oraz wysokość pola tekstowego. Robimy to z użyciem właściwości `Width` i `Height`.
10. Czas na ustawienie możliwości tworzenia tekstu z użyciem wielolinii. W tym celu we właściwości `Multiline` wprowadzamy wartość `true`, czyli potwierdzającą, że chcemy zapewnić możliwość wpisywania tekstu w wielu liniach.
11. Na koniec dodajemy pole tekstowe do formularza przez dodanie go jako kontrolki (`Controls.Add`).

Ustawianie akcji w celu wyświetlenia zawartości pola

Pole tekstowe oraz jego właściwości omówiliśmy w poprzednich podrozdziałach. Nadszedł więc czas, aby zrobić coś z wartością wpisywaną w naszym polu (rysunek 3.2). Najprostszym i najlepszym przykładem będzie wyświetlenie tekstu zawartego w polu tekstowym jako treści komunikatu (listing 3.9).

RYSUNEK 3.2. Formularz z polem tekstowym i przyciskiem powodującym wyświetlenie treści pola tekstowego w nowym oknie

LISTING 3.9. Definiowanie akcji, której celem jest wyświetlenie tekstu zapisanego w naszym polu tekstowym

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

            this.Text = "InputForm";
            this.Width = 347;
            this.Height = 323;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Input
            TextBox firstInput = new TextBox();
            firstInput.Name = "inputBoxInForm";
            firstInput.Location = new Point(12, 12);

            firstInput.Width = 300;
            firstInput.Height = 200;
```

Kup książkę Poleć książkę

```
            firstInput.Multiline = true;

            Controls.Add(firstInput);

            // Button
            Button firstButton = new Button();
            firstButton.Location = new Point(12, 222);

            firstButton.Text = "Click me!";

            firstButton.Height = 35;
            firstButton.Width = 300;

            firstButton.Click += new EventHandler(firstButton_Click);

            Controls.Add(firstButton);
        }

        private void firstButton_Click(object sender, EventArgs e)
        {
            string ValueOfField = ((TextBox)Controls["inputBoxInForm"]).Text;

            if (ValueOfField == "")
            {
                MessageBox.Show("Field empty!", "Alert!", MessageBoxButtons.OK);
            }
            else
            {
                MessageBox.Show(ValueOfField, "Alert!", MessageBoxButtons.OK);
            }
        }
    }
}
```

Aby uzyskać działający kod:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) naszego formularza.
3. Ustawiamy jako ukryte przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość okna programu będzie można zmieniać także przez przeciąganie jego aktywnych brzegów, dlatego należy ustawić `this.FormBorderStyle` na obramowanie nie pozwalające na takie działanie.
6. Tworzymy pole tekstowe przez przypisanie go do obiektu, w którym zawarta jest definicja pola: `TextBox firstInput = new TextBox()`.
7. Określamy nazwę pola (`firstInput.Name`).
8. Ustawiamy lokalizację pola przez podanie współrzędnych we właściwości `Location`.
9. Teraz ustawiamy szerokość oraz wysokość pola. Robimy to z użyciem właściwości `Width` i `Height`.

10. Czas na dodanie możliwości tworzenia tekstu z użyciem wielolinii. W tym celu we właściwości `Multiline` ustawiamy wartość `true`, czyli potwierdzającą, że chcemy zapewnić możliwość wpisywania tekstu w wielu liniach.
11. Na koniec dodajemy pole tekstowe do formularza przez dodanie go jako kontrolki (`Controls.Add`).
12. Musimy zdefiniować przycisk, za pomocą którego wyświetlimy komunikat z zawartością pola. W tym celu przypisujemy go do klasy odpowiedzialnej za jego tworzenie.
13. Następnie za pomocą właściwości `Location` określamy położenie przycisku.
14. Napis ustawiamy we właściwości `Text`.
15. Należy także zadbać, aby przycisk miał odpowiednie rozmiary, takie jak szerokość i wysokość (`Width` i `Height`).
16. Zdefiniujemy teraz akcję wywoływaną kliknięciem przycisku lewym przyciskiem myszy. Przypisujemy do właściwości wywołanie akcji. W pierwszym parametrze definiujemy metodę, która zostaje wywołana: `firstButton.Click += new EventHandler(firstButton_Click)`.
17. Teraz dodajemy gotowy przycisk do formularza (`Controls.Add`).
18. Definiujemy prywatną metodę dotyczącą kliknięcia przycisku, która będzie zawierała dwa argumenty. Pierwszym będzie obiekt, który spowodował wysłanie obecnego żądania, a drugim błędy, które mogą powstać podczas działania przycisku: `private void firstButton_Click(object sender, EventArgs e)`.
19. Pobieramy wartość pola za pomocą właściwości `Text`.
20. Sprawdzamy, czy wartość jest pusta. Jeśli jest, wówczas wyświetlamy komunikat o tym, aby uzupełnić pole.
21. W przypadku gdyby wartość pola została mu nadana, wyświetlamy wartość za pomocą metody.

Etykiety (Label)

Etykieta pozwala opisać pole, do którego należy wprowadzić wartość. My, jako twórcy programu, wiemy, jaką wartość należy wprowadzić w to pole, jednak osoby, które będą korzystać z naszego oprogramowania — nie. Utwórzmy nowy projekt o nazwie *LabelForm*.

Dodawanie pola opisu

Pole opisu to przygotowany przez programistów opis pola, w którym należy wprowadzić wartość albo z którego należy ją wybrać. Etykiety można zastosować jako opis funkcjonalności w programie czy też ostrzeżenie przed wybraniem nieprawidłowej wartości w polu (listing 3.10).

Kup książkę Poleć książkę

LISTING 3.10. Używanie pola etykiety

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

            this.Text = "LabelForm";
            this.Width = 560;
            this.Height = 160;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            // Label
            Label firstLabel = new Label();
            firstLabel.Location = new Point(20, 14);

            firstLabel.Text = "Enter the sentence:";

            Controls.Add(firstLabel);
        }
    }
}
```

Aby uzyskać działający kod:

1. Określamy nazwę naszej aplikacji (`this.Text`).
2. Następnie ustawiamy szerokość (`this.Width`) oraz wysokość (`this.Height`) formularza.
3. Ukrywamy przycisk do maksymalizacji (`this.MaximizeBox`) oraz minimalizacji (`this.MinimizeBox`) naszej aplikacji.
4. Aplikacja po uruchomieniu powinna być wyświetlona na środku ekranu (`this.StartPosition`).
5. Wielkość aplikacji będzie można zmieniać także przez przeciąganie jej aktywnych brzegów, dlatego musimy ustawić `this.FormBorderStyle` na obramowanie pojedyncze.
6. Następnie definiujemy naszą etykietę przez przypisanie jej do obiektu klasy odpowiedzialnej za tworzenie etykiety: `Label firstLabel = new Label()`.
7. Określamy położenie pola etykiety przez ustawienie w nim punktów położenia `X` i `Y` (`Location`).
8. Pora na wpisanie tekstu, który będzie się wyświetlał w naszym polu, a który ustawiamy we właściwości `Text`.
9. Na koniec dodajemy kontrolkę do naszego formularza z użyciem metody `Controls.Add(firstLabel)`, w której pierwszym argumentem jest nazwa kontrolki.

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

VISUAL STUDIO 2022

C# I .NET

Programowanie kontrolek

Nie trzeba siedzieć po uszy w programowaniu, by zauważyć pewną regułę: lwia część książek poświęconych temu zagadnieniu została napisana w podobny sposób. I nie chodzi o styl, środowisko czy język, lecz o strukturę. Prawidłowość tę zauważył Łukasz Sosna, który do zagadnienia programowania w języku C# i użytkowania środowiska .NET postanowił podejść w odmienny sposób. W najnowszej publikacji swojego autorstwa skupia się zatem na praktycznym wymiarze programowania, czyli używaniu dostępnych w oprogramowaniu Visual Studio 2022 komponentów zwanych kontrolkami.

Prezentuje sposoby oprogramowywania przycisków, menu, pól tekstowych i wyboru, etykiet, list, dymków podpowiedzi, paneli, okien z wyborem koloru, czcionki itd. Skąd taki wybór? Autor, doświadczony programista, doskonale zdaje sobie sprawę, że Visual Studio, C# i .NET zdobywają coraz większą popularność, gdyż idealnie nadają się do pisania programów na wszystkie systemy: komputery działające pod kontrolą Windowsa, Linuksa, macOS-a, a także na inteligentne telewizory, smartfony itd.

Jeżeli już znasz Visual Studio i jesteś początkującym programistą korzystającym z tego oprogramowania, to propozycja właśnie dla Ciebie. To też poradnik dla tych, którzy zetknęli się z różnymi językami programowania i chcą się nauczyć obsługi środowiska Visual Studio 2022 w języku C#.

W książce:

- **instalowanie Visual Studio 2022, .NET SDK, .NET Runtime**
- **debugowanie i uruchamianie aplikacji**
- **stosowanie kontrolek i budowanie przy ich użyciu nowych programów w C#, .NET, Visual Studio 2022**

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-9140-6

9 788328 391406

Cena: 49,90 zł