

Andrzej Stefańczyk

„Tworzenie aplikacji okienkowych”

Niniejszy **darmowy** ebook zawiera fragment pełnej wersji pod tytułem:

["Sekrety języka C#"](#)

Aby przeczytać informacje o pełnej wersji, [kliknij tutaj](#).

Darmowa publikacja dostarczona przez

[Strefe Ebookow](#)

Niniejsza publikacja może być kopiowana, oraz dowolnie rozprowadzana tylko i wyłącznie w formie dostarczonej przez Wydawcę. Zabronione są jakiegokolwiek zmiany w zawartości publikacji bez pisemnej zgody wydawcy. Zabrania się jej odsprzedaży, zgodnie z [regulaminem Wydawnictwa Złote Myśli](#).

© Copyright for Polish edition by ZloteMysli.pl

Data: 05.07.2005

Tytuł: „Tworzenie aplikacji okienkowych”

Fragment utworu: [„Sekrety języka C#”](#)

Autor: Andrzej Stefańczyk

Korekta techniczna i skład: Anna Grabka

Niniejsza publikacja może być kopiowana, oraz dowolnie rozprowadzana tylko i wyłącznie w formie dostarczonej przez Wydawcę. Zabronione są jakiegokolwiek zmiany w zawartości publikacji bez pisemnej zgody wydawcy. Zabrania się jej odsprzedaży, zgodnie z [regulaminem Wydawnictwa Złote Myśli](#).

Internetowe Wydawnictwo Złote Myśli

Złote Myśli s.c.

ul. Plebiscytowa 1

44-100 Gliwice

WWW: www.ZloteMysli.pl

EMAIL: kontakt@zlotemysli.pl

Wszelkie prawa zastrzeżone.

All rights reserved.

SPIS TREŚCI

<u>CZEŚĆ II. TWORZENIE APLIKACJI OKIENKOWYCH</u>	4
<u>ROZDZIAŁ 1. PODSTAWY WINDOWS FORMS</u>	4
<u>WPROWADZENIE</u>	4
<u>GENEROWANIE APLIKACJI WINDOWS FORMS</u>	4
<u>ROZDZIAŁ 2. PRACA Z FORMĄ</u>	6
<u>TWORZENIE FORMY</u>	6
<u>WŁAŚCIWOŚCI FORMY</u>	7
<u>OBSŁUGA ZDARZEŃ</u>	8
<u>METODY FORMY</u>	10
<u>PRZYKŁADOWA APLIKACJA</u>	10
<u>ROZDZIAŁ 3. KORZYSTANIE Z PROSTYCH KONTROLEK</u>	12
<u>DODAWANIE KONTROLEK DO FORMY</u>	12
<u>ORGANIZOWANIE KONTROLEK NA FORMIE</u>	12
<u>WSPÓLNE CECHY KONTROLEK</u>	13
<u>Właściwości</u>	13
<u>Zdarzenia</u>	14
<u>Metody</u>	15
<u>JAK SKORZYSTAĆ Z WIEDZY ZAWARTEJ W PEŁNEJ WERSJI EBOOKA?</u>	16
<u>SPIS TREŚCI PEŁNEJ WERSJI</u>	

Tworzenie aplikacji okienkowych

Rozdział 1. Podstawy Windows Forms

Wprowadzenie

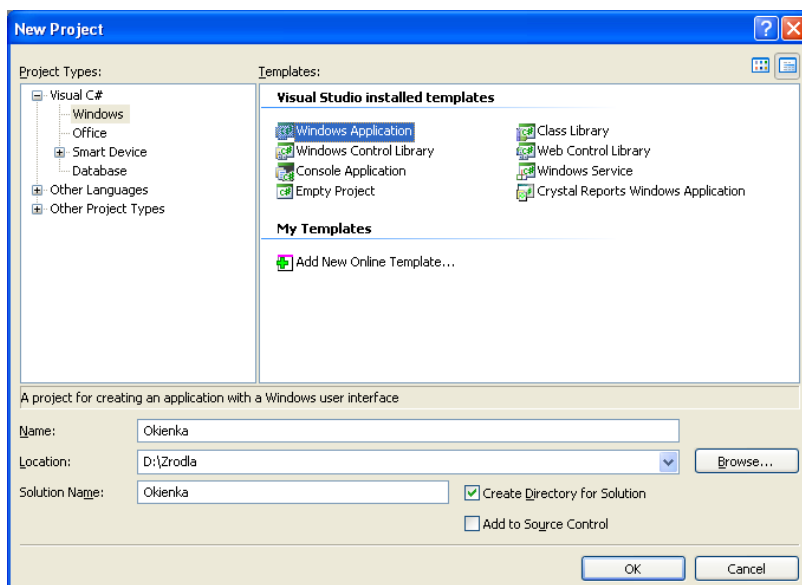
Windows Forms jest zbiorem elementów służących do tworzenia aplikacji okienkowych. Jednakże tworzenie aplikacji okienkowych to nie tylko tworzenie kodu, ale również projektowanie interfejsu użytkownika. Interfejs użytkownika jest jednym z podstawowych elementów aplikacji okienkowej i decyduje w znacznym stopniu o jej atrakcyjności. To właśnie interfejs jest warstwą, z którą styka się użytkownik (wchodzi w interakcje).

Środowisko Microsoft Visual Studio.NET 2005 wspiera proces tworzenia aplikacji okienkowych w pełnym zakresie. Zintegrowane narzędzia do projektowania interfejsu pozwalają na wygodne i szybkie budowanie aplikacji okienkowych.

Generowanie aplikacji Windows Forms

Aby wygenerować szablon projektu aplikacji *Windows Forms* należy:

1. Utworzyć nowy projekt wybierając opcję *Project...* z menu *File->New*.
2. W obszarze *Project Types* wybrać typ projektu *Windows*.
3. W obszarze *Templates* wybrać wzorzec *Windows Application*.
4. W obszarze *Name* wpisać nazwę projektu.
5. W obszarze *Location* wskazać ścieżkę katalogu głównego, w którym będzie umieszczony katalog projektu.
6. Upewniamy się, że w obszarze *Solution* wybrano z listy *Create new Solution*.
7. Naciskamy przycisk *Ok*.



Rysunek 27. Generowanie szablonu aplikacji okienkowej

Po wygenerowaniu szablonu aplikacji okienkowej zostanie utworzony projekt zawierający kod pozwalający na wyświetlenie okna głównego aplikacji.

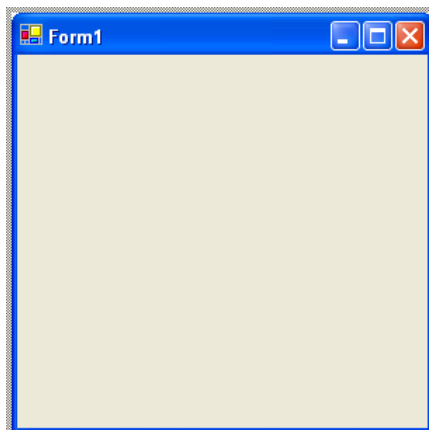
Rozdział 2. Praca z formą

Tworzenie formy

Forma jest podstawowym elementem interfejsu użytkownika aplikacji okienkowej Windows. Jest to projekt okna, które zostanie wyświetlone w celu prezentacji danych użytkownikowi oraz wczytywaniu danych od użytkownika.

Po wygenerowaniu szablonu aplikacji okienkowej, tworzona jest zawsze forma podstawowa. Na formie tej w trakcie projektowania interfejsu można umieszczać szereg różnych kontroltek (np.: menu, paski narzędzi, przyciski, etc.) oraz przypisywać im określone zachowania (np. reakcję na naciśnięcie przycisku czy wybranie opcji z menu).

Bazową klasą dla każdej formy jest klasa *Form*, znajdująca się w przestrzeni *System.Windows.Forms* (klasa *Form* dziedziczy z klasy *Control*).

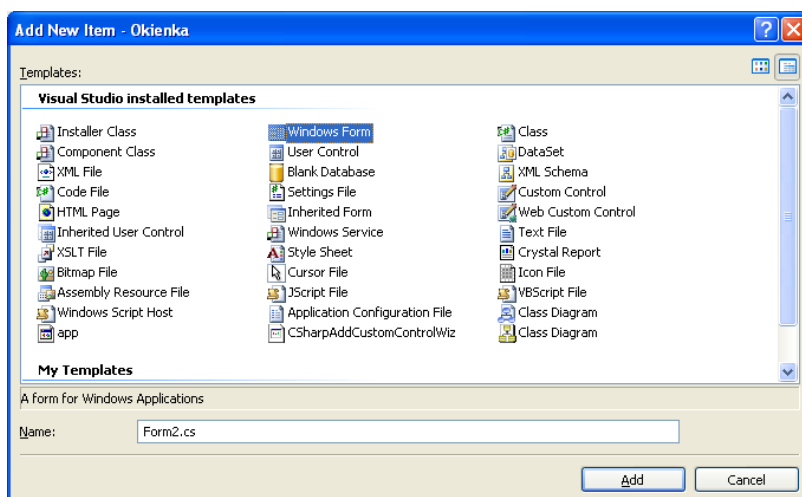


Rysunek 28. Wygenerowana forma

Oczywiście większość aplikacji posiada więcej niż jedno okno, więc jeżeli nasza aplikacja ma posługiwać się większą liczbą okien, musimy do projektu dodać nową formę.

W celu dodania nowej formy do projektu, należy wykonać następujące czynności:

1. W oknie *Solution Explorer* kliknąć prawym klawiszem myszy na nazwie projektu.
2. Z menu *Add* wybrać opcję *Windows Form...*
3. W oknie *Add New Item* w polu *Name* wpisać nazwę pliku dla nowej formy.
4. Nacisnąć przycisk *Add*.





Rysunek 29. Wygląd okna Add New Item

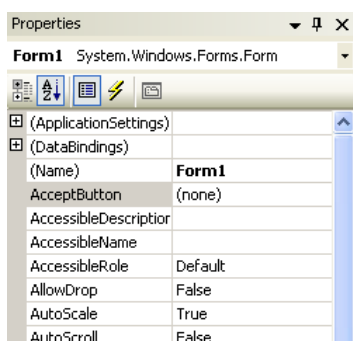
Każda forma posiada:

- **Właściwości** – pozwalające na zmianę wyglądu formy;
- **Metody** – pozwalające na zdefiniowanie zachowania formy;
- **Zdarzenia** – pozwalające na interakcję z użytkownikiem.

Właściwości formy

Forma posiada szereg właściwości pozwalających na zmianę jej wyglądu. Właściwości form ustawia się za pośrednictwem okna **Properties Window**. Znając nazwę właściwości, którą chcemy zmienić, należy kliknąć na jej nazwę i po prawej stronie wpisać lub wybrać z listy określoną wartość właściwości.

Właściwości okna mogą być wyświetlane w grupach funkcyjnych (ikona ) lub w porządku alfabetycznym (ikona )





Rysunek 30. Fragment okna Properties z aktywną listą właściwości

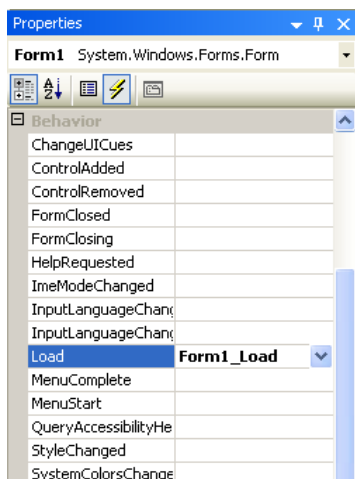
Poniższe zestawienie zawiera opis najczęściej używanych właściwości formy oraz przyjmowanych wartości domyślnych:

Właściwość	Opis	Wartość domyślna
(Name)	Nazwa formy (chodzi o nazwę klasy dla formy, której będziemy używać tworząc obiekty nie zaś napisu wyświetlanego na pasku tytułowym okna).	Form1, Form2, ...
AcceptButton	Określa, który przycisk ma pełnić rolę domyślnego przycisku akceptacji (reagować na naciśnięcie klawisza ENTER).	(none)

AllowDrop	Określa, czy forma akceptuje komunikaty „przeciągnij i upuść”.	<i>False</i>
AutoSize	Określa, czy forma ma być automatycznie dopasowana do kontekstu.	<i>False</i>
BackColor	Określa kolor tła formy.	<i>Control</i>
CancelButton	Określa, który przycisk ma pełnić rolę domyślnego przycisku anulowania (reagować na naciśnięcie klawisza ESC).	<i>(none)</i>
ControlBox	Określa, czy forma wyświetla przyciski kontroli okna w pasku tytułowym. Menu może zawierać przyciski minimalizacji, maksymalizacji, pomocy oraz zamknięcia.	<i>True</i>
Cursor	Określa rodzaj kursora myszy wyświetlany w czasie, gdy znajdzie się on w obrębie formy.	<i>Default</i>
Enabled	Określa czy forma jest dostępna.	<i>True</i>
Font	Określa rodzaj czcionki używanej przez formę.	<i>Microsoft Sans Serif; 8,25pt</i>
ForeColor	Określa kolor tekstu i grafiki formy.	<i>ControlText</i>
FormBorderStyle	Określa wygląd okna (okno rozszerzalne, dialogowe, bez ramki, narzędziowe, etc.).	<i>Sizable</i>
HelpButton	Określa, czy okno posiada przycisk pomocy.	<i>False</i>
Icon	Określa ikonę dla formy.	<i>domyślna</i>
IsMdiContainer	Określa, czy forma pełni rolę kontenera MDI.	<i>False</i>
Location	Określa pozycję górnego lewego rogu formy.	<i>0,0</i>
Locked	Określa, czy kontrolki mogą być przesuwane i czy można zmieniać ich rozmiar.	<i>True</i>
MaximizeBox	Określa, czy forma posiada przycisk maksymalizacji na pasku tytułowym.	<i>True</i>
MaximumSize	Określa maksymalny rozmiar formy.	<i>0, 0 (czyli dowolny)</i>
Menu	Określa, które menu jest głównym dla okna.	<i>(none)</i>
MinimizeBox	Określa, czy forma posiada przycisk minimalizacji na pasku tytułowym.	<i>True</i>
MinimumSize	Określa minimalny rozmiar formy.	<i>0, 0</i>
Opacity	Określa poziom widoczności/przezroczystości formy (100% widoczne, 0% przezroczyste).	<i>100%</i>
ShowInTaskBar	Określa czy okno ma być widoczne w oknie zadań	<i>True</i>
Size	Określa rozmiar początkowy formy.	<i>300; 300 (zaprojektowany)</i>
StartPosition	Określa pozycję pierwszego pojawienia się formy.	<i>WindowsDefaultLocation</i>
Text	Określa tytuł formy pojawiający się na pasku tytułowym.	<i>Form1, Form2, ...</i>
TopMost	Określa, czy forma jest oknem najbardziej widocznym.	<i>False</i>
WindowState	Określa sposób pojawienia się formy (normalna, zminimalizowana lub zmaksymalizowana).	<i>Normal</i>

Obsługa zdarzeń

Forma posiada listę zdarzeń, które mogą zostać powiązane z metodą reagującą na wystąpienie określonego zdarzenia (np.: pojawienie się okna). Dodawanie metod reagujących na zdarzenie następuje za pośrednictwem okna **Properties Window**. Okno to pozwala zarówno na zmianę właściwości, jak i przypisywanie zdarzeniom metod. Tryb edycji właściwości jest aktywny w momencie naciśnięcia ikony **Properties** , natomiast tryb edycji zdarzeń w momencie naciśnięcia ikony **Events** .



Rysunek 31. Okno **Properties** w trybie edycji zdarzeń (na rysunku widać przypisaną zdarzeniu **Load** metodę **Form1_Load**)

Poniższe zestawienie zawiera opis najczęściej używanych zdarzeń oraz opis przypadków ich występowania:

Zdarzenie	Opis
Activated	Występuje, gdy forma jest aktywowana przez użytkownika lub instrukcję.
Click	Występuje, gdy nastąpi kliknięcie na formę.
Deactivate	Występuje, gdy forma jest deaktywowana (traci focus).
DoubleClick	Występuje, gdy nastąpi podwójne kliknięcie na formę.
Enter	Występuje, gdy nastąpi wejście do formy.
FormClosed	Występuje, gdy forma zostanie zamknięta (po zamknięciu).
FormClosing	Występuje, gdy forma jest zamykana (przed zamknięciem).
KeyDown	Występuje, gdy nastąpi naciśnięcie klawisza (przekazywany jest kod klawisza).
KeyPress	Występuje, gdy nastąpi naciśnięcie klawisza (przekazywany jest znak klawisza).
KeyUp	Występuje, gdy nastąpi zwolnienie klawisza (przekazywany jest kod klawisza).
Leave	Występuje, gdy nastąpi opuszczenie formy.
Load	Występuje przed pierwszym pojawieniem się formy (zdarzenie to występuje przed wywołaniem metody Show i jest używane w przypadku, gdy istnieje potrzeba wykonania pewnych instrukcji jeszcze zanim pokaże się forma, zazwyczaj jest to dobry moment do przypisania domyślnych wartości formie i jej kontrolkom).
MouseDown	Występuje, gdy zostanie naciśnięty przycisk myszy.
MouseMove	Występuje, gdy kursor myszy jest poruszany w obszarze formy.
MouseUp	Występuje, gdy zostanie zwolniony przycisk myszy.
Move	Występuje, gdy forma zmienia położenie.
Paint	Występuje, gdy forma jest przerysowywana.

Resize	Występuje, gdy forma zmienia swój rozmiar.
---------------	--

Przykład ustawienia tytułu okna:

```
private void Form1_Load(object sender, EventArgs e)
{
    this.Text = "Aplikacja testowa";
}
```

Metody formy

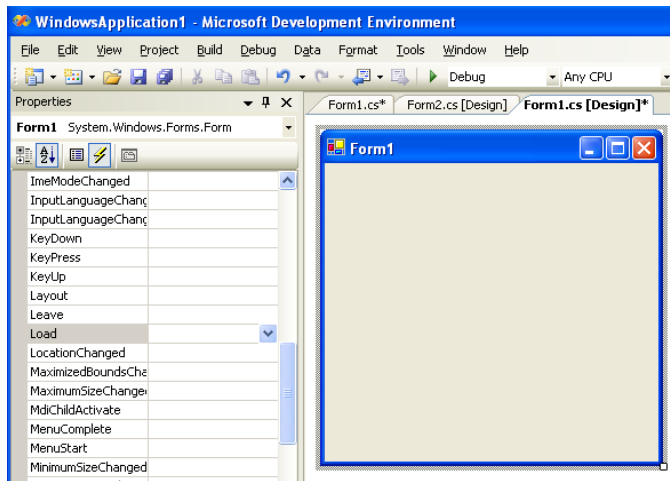
Forma posiada szereg metod, które pozwalają na definiowanie zachowania formy. Poniższa tabela zawiera zestawienie najczęściej używanych metod:

Metoda	Opis
<i>Activate</i>	Aktywuje formę.
<i>Close</i>	Zamyka formę.
<i>Focus</i>	Sprawia, że forma uzyskuje focus.
<i>Hide</i>	Ukrywa formę.
<i>Refresh</i>	Wymusza odświeżenie (odrysowanie) całej formy i jej kontroltek.
<i>Show</i>	Pokazuje formę.
<i>ShowDialog</i>	Pokazuje formę jako modalne okno dialogowe.
<i>Update</i>	Wymusza odrysowanie widocznej części formy.

Przykładowa aplikacja

Napiszmy sobie teraz prostą aplikację złożoną z dwóch form (głównej i dodatkowej), które po uruchomieniu aplikacji pojawią się na ekranie obok siebie:

1. Generujemy nowy projekt aplikacji okienkowej (patrz „Generowanie aplikacji Windows Forms”).
2. Dodajemy nową formę (patrz „Tworzenie formy”).
3. Klikamy na projekcie formy podstawową (*Form1*) i w oknie *Properties* wybieramy zdarzenie *Load*.



Rysunek 32. Okno projektu formy **Form1** oraz okno **Properties** z zaznaczonym zdarzeniem **Load**

4. Tworzymy metodę do zdarzenia i łączymy ją ze zdarzeniem podwójnie klikając na puste pole po prawej stronie nazwy zdarzenia **Load**. Zostanie utworzona metoda **Form1_Load**, która będzie wywoływana po wystąpieniu zdarzenia **Load**:

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

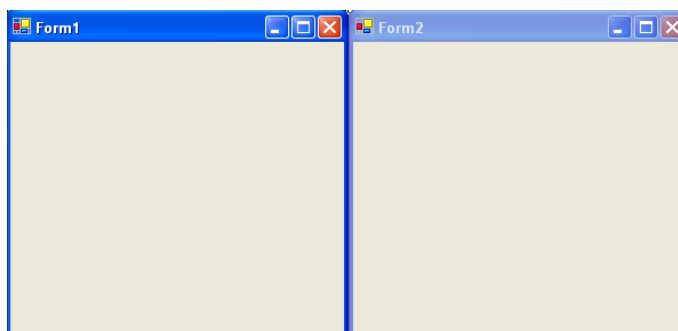
5. Wpisujemy kod, który utworzy nowe okno i ustawi jego początkowe położenie z prawej strony okna głównego. Pełna metoda wygląda tak:

```
private void Form1_Load(object sender, EventArgs e)
{
    Form2 frm = new Form2(); // tworzymy instancję nowej formy

    frm.Show(); // wyświetlamy formę
    frm.Location = new Point(this.Location.X + this.Size.Width,
                             this.Location.Y); // zmieniamy jej położenie
}
```

Jak widać, w pierwszej kolejności musieliśmy utworzyć instancję dla nowej formy i użyć metody **Show**, aby ukazała się ona na ekranie. Następnie wykorzystując właściwość **Location**, zmieniliśmy jej położenie początkowe na zgodne w pionie z formą podstawową i przesunięte o szerokość formy podstawowej w poziomie (do tego celu posłużyliśmy się dodatkowo właściwością **Size**).

6. Po skompilowaniu i uruchomieniu otrzymujemy wynik działania programu.



Rysunek 33. Wynik działania programu

Rozdział 3.

Korzystanie z prostych kontroltek

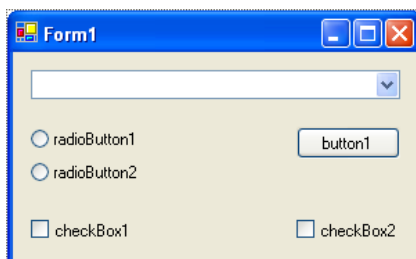
Dodawanie kontroltek do formy

Kontrolki są obiektami, które są umieszczane na formie. Przykładami kontroltek są: przyciski, pola tekstowe, etykiety, listy rozwijane, drzewa, itp.

Dodanie kontrolki do formy odbywa się poprzez wybór z okna **Toolbox** właściwej kontrolki i przeciągnięcie jej na formę. Rozmiar i położenie kontrolki można dopasować zarówno w czasie przeciągania przed lub też po upuszczeniu jej na formę.

Jeżeli chcemy dopasować rozmiar i położenie w czasie upuszczania, wystarczy chwycić kontrolkę i umieścić kursor myszy w miejscu, gdzie ma się znajdować lewa górna krawędź kontrolki. Następnie trzymając lewy klawisz myszy wciśnięty poruszać się w prawy dolny róg. Kiedy rozmiar jest właściwy zwalniamy lewy klawisz.

Jeżeli upuszczamy kontrolkę bez dopasowywania jej w czasie przeciągania, wystarczy kliknąć lewym klawiszem myszy w miejscu, w którym ma się znajdować lewa górna krawędź kontrolki, a wszelkie operacje przesunięcia i dopasowywania rozmiaru wykonać później.

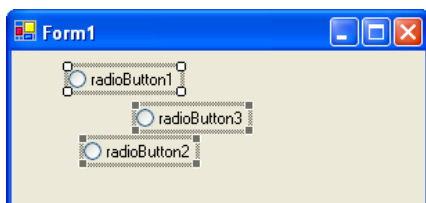


Rysunek 34. Projekt formy z przykładowymi kontrolkami (zaraz po umieszczeniu ich na formie)

Organizowanie kontroltek na formie

Po umieszczeniu kontroltek na formie można je dowolnie formatować. W celu ich równomiernego rozmieszczenia oraz dopasowania wielkości, można posłużyć się jedną z wielu opcji formatowania, dostępnych w menu **Format** środowiska Visual Studio C#.NET 2005.

W pierwszej kolejności należy zaznaczyć kontrolkę lub grupę kontroltek (grupę wybiera się trzymając wciśnięty klawisz **Shift** i wskazując kolejne kontrolki). Następnie wybieramy opcje formatowania. Grupę kontroltek można również przesuwać na formatce w taki sam sposób jak pojedyncze kontrolki.

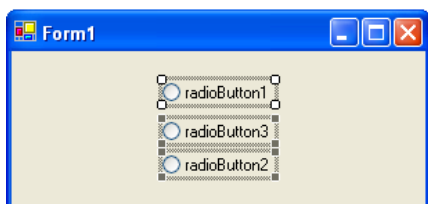


Rysunek 35. Przykładowa grupa kontroltek przygotowanych do sformatowania

Menu **Format** zawiera dużą ilość opcji pogrupowanych w podmenu, które służą do formatowania grupy

kontrolerek:

- **Align** – pozwala na wyrównanie kontrolerek do jednej linii,
- **Make Same Size** – pozwala na wyrównanie rozmiarów kontrolerek,
- **Horizontal Spacing** – pozwala na zmianę poziomego rozmiaru odstępu między kontrolkami,
- **Vertical Spacing** – pozwala na zmianę pionowego rozmiaru odstępu między kontrolkami,
- **Center in Form** – pozwala na umieszczenie kontrolerek w centrum formy,
- **Order** – pozwala na zmianę porządku kontrolerek,
- **Lock Controls** – pozwala na zablokowanie kontrolerek do modyfikacji.



Rysunek 36. Wyrównanie do lewej i umieszczenie w centrum

Wspólne cechy kontrolerek

Ze względu na to, że wszystkie kontrolki dziedziczą z klasy **Control**, posiadają zbiór wspólnych cech, które mają wspólne znaczenie.

Właściwości

Poniższe zestawienie zawiera najczęściej używane właściwości kontrolerek (dostępnych z okna **Properties**):

Właściwość	Opis
(Name)	Nazwa obiektu utworzonej kontrolki, którego będziemy używać.
Anchor	Określa, które brzegi są zakotwiczone do formy i dla których ma zostać zachowana odległość od krawędzi formy (określone brzegi będą rozszerzane wraz z formą w taki sposób, aby odległość od krawędzi formy była zachowana).
AutoSize	Określa, czy kontrolka będzie automatycznie dopasowywana do kontekstu (np.: im dłuższy tekst tym większy rozmiar kontrolki).
BackColor	Określa kolor tła danej kontrolki.
BackgroundImage	Określa jaki obrazek pełniący rolę tła ma być wyświetlany w kontrolce.
ContextMenuStrip	Określa rodzaj menu kontekstowego skojarzonego z kontrolką.
Cursor	Określa rodzaj kursora myszy, jaki pojawi się, gdy znajdzie się on w obszarze kontrolki.
Dock	Określa rodzaj zakotwiczenia kontrolki (np.: wypełnienie całego wnętrza).
Enabled	Określa, czy kontrolka jest dostępna w trybie do edycji.
FlatStyle	Określa wygląd kontrolki oraz jej zachowanie. Właściwość ta może przyjąć wartości: <ul style="list-style-type: none">• Standard – (domyślny) kontrolka z krawędziami 3d• System – wygląd kontrolki uzależniony jest od systemu operacyjnego• Flat – kontrolka płaska

	<ul style="list-style-type: none"> • Popup – kontrolka jest płaska do momentu, gdy kursor nie znajdzie się w jej obszarze, wtedy się podnosi
Font	Określa czcionkę używaną przez kontrolkę.
ForeColor	Określa kolor tekstu i grafiki używany przez kontrolkę.
Location	Określa położenie lewej górnej krawędzi kontrolki.
Locked	Określa, czy kontrolka może być modyfikowana na etapie projektowania.
MaximumSize	Określa maksymalny rozmiar kontrolki.
MinimumSize	Określa minimalny rozmiar kontrolki.
Modifiers	Określa modyfikator dostępu dla obiektu kontrolki.
RightToLeft	Określa, czy kontrolka ma stosować tryb z prawej do lewej dla języków stosujących taki zapis.
Size	Określa rozmiar kontrolki w pikselach.
Text	Określa tekst jaki zawiera kontrolka (np.: etykieta przycisku).
Visible	Określa, czy kontrolka jest widoczna po wyświetleniu formy.

Zdarzenia

Poniższe zestawienie zawiera najczęściej używane zdarzenia dla kontrolki (dostępne z okna *Properties*):

Zdarzenie	Opis
AutoSizeChanged	Występuje, gdy zmianie ulegnie właściwość AutoSize .
BackColorChanged	Występuje, gdy zmianie ulegnie właściwość BackColor .
BackgroundImageChanged	Występuje, gdy zmianie ulegnie właściwość BackgroundImage .
Click	Występuje, gdy nastąpi kliknięcie na kontrolkę.
CursorChanged	Występuje, gdy zmianie ulegnie właściwość Cursor .
DockChanged	Występuje, gdy zmianie ulegnie właściwość Dock .
EnabledChanged	Występuje, gdy nastąpi zmiana trybu dostępności kontrolki.
Enter	Występuje, gdy kontrolka staje się aktywną kontrolką formy.
FontChanged	Występuje, gdy zmianie ulegnie właściwość Font .
ForeColorChanged	Występuje, gdy zmianie ulegnie właściwość ForeColor .
KeyDown	Występuje, gdy naciśnięty zostanie klawisz (przekazywany jest kod klawisza).
KeyPress	Występuje, gdy naciśnięty zostanie klawisz (przekazywany jest znak klawisza).
KeyUp	Występuje, gdy zwolniony zostanie klawisz (przekazywany jest kod klawisza).
Leave	Występuje, gdy kontrolka przestaje być aktywną kontrolką formy.
MouseClick	Występuje, gdy nastąpi kliknięcie myszą.
MouseDoubleClick	Występuje, gdy nastąpi podwójne kliknięcie myszą.
MouseDown	Występuje, gdy nastąpi naciśnięcie klawisza myszy.
MouseEnter	Występuje, gdy kursor znajdzie się w obszarze kontrolki.
MouseHover	Występuje, gdy kursor myszy zatrzyma się w obszarze kontrolki po wystąpieniu zdarzenia MouseEnter .
MouseLeave	Występuje, gdy kursor opuści obszar kontrolki.

<i>MouseMove</i>	Występuje, gdy nastąpi zmiana pozycji kursora myszy w obszarze kontrolki.
<i>MouseUp</i>	Występuje, gdy nastąpi zwolnienie klawisza myszy.
<i>Resize</i>	Występuje, gdy zmieniany jest rozmiar kontrolki.
<i>SizeChanged</i>	Występuje, gdy zmianie ulegnie właściwość <i>Size</i> .
<i>TextChanged</i>	Występuje, gdy zmianie ulegnie właściwość <i>Text</i> .
<i>Validated</i>	Występuje, gdy kontrolka została sprawdzona (zdarzenie może być wykorzystane do przetwarzania sprawdzonych wcześniej wartości charakterystycznej dla kontrolki np.: formatu danych).
<i>Validating</i>	Występuje, gdy kontrolka jest sprawdzana. Aby anulować sprawdzanie należy ustawić właściwość <i>Cancel</i> na wartość <i>True</i> (zdarzenie może być wykorzystane do sprawdzenia poprawności wartości charakterystycznej dla kontrolki np.: formatu danych).
<i>VisibleChanged</i>	Występuje, gdy nastąpi zmiana widoczności kontrolki.

Podstawowym zdarzeniem dla kontrolki jest zdarzenie *Click*. Aby stworzyć metodę obsługującą to zdarzenie wystarczy dwukrotnie kliknąć na kontrolkę w projekcie formy, w której się znajduje. Zostanie wtedy utworzona pusta metoda, którą możemy wypełnić kodem wykonującym jakąś akcję. Metoda wywoływana będzie zawsze, gdy nastąpi kliknięcie na kontrolkę. Przykładowo dla przycisku o nazwie `button1` zostanie wygenerowane zdarzenie:

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

Zdarzenie może występować z określonymi parametrami, które są przekazywane do metody obsługującej w postaci argumentów wejściowych.

Do argumentów tych należą:

- obiekt, który wysłał zdarzenie (`sender`);
- obiekt (klasa *EventArgs* lub klasa potomna tej klasy) zawierający parametry wywołania zdarzenia (np.: dla zdarzenia *KeyDown* jest to klasa *KeyEventArgs*, która zawiera parametry dotyczące kodu naciśniętego klawisza).

Metody

Poniższe zestawienie zawiera najczęściej używane metody dla kontrolki:

Metoda	Opis
<i>Focus</i>	Sprawia, że kontrolka uzyskuje focus.
<i>Hide</i>	Ukrywa kontrolkę.
<i>Invalidate</i>	Wymusza odrysowanie kontrolki.
<i>Refresh</i>	Wymusza odrysowanie kontrolki i wszystkich innych kontrolki, dla których dana kontrolka jest macierzysta.
<i>Show</i>	Pokazuje kontrolkę.
<i>Update</i>	Wymusza odrysowanie obszaru kontrolki.

Jak skorzystać z wiedzy zawartej w pełnej wersji ebooka?

Interesuje Cię tematyka programistyczna? Chciałbyś poznać sekrety języka C# i nauczyć się nim sprawnie posługiwać? Teraz także Ty, możesz poznać wiedzę programisty jednej z największych firm informatycznych w Polsce (Prokom S.A.), udostępnioną w formie praktycznego kursu języka C#.

To pierwsza tego typu publikacja elektroniczna w Polsce.

Zobacz szczegóły, na stronie: <http://c-sharp.zlotemysli.pl/>

