

Podręcznik programisty WPF 4.5!

Adam Nathan

WPF 4.5

KSIĘGA EKSPERTA

SAMS

Helion 

Tytuł oryginału: WPF 4.5 Unleashed

Tłumaczenie: Paweł Gonera (wstęp, rozdz. 1 – 14), Ireneusz Jakóbik (rozdz. 15 – 22, dodatek)

ISBN: 978-83-283-0141-2

Authorized translation from the English language edition, entitled: WPF 4.5 UNLEASHED, ISBN 0672336979; by Adam Nathan; published by Pearson Education, Inc, publishing as SAMS Publishing. Copyright © 2014 by Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education Inc.

Polish language edition published by HELION S.A. Copyright © 2015.

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 Wydawnictwo HELION 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 Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/wp4ke.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/wp4ke>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

O autorze	11
------------------------	-----------

Wstęp	13
--------------------	-----------

Kto powinien przeczytać tę książkę?	14
---	----

Wymagania dotyczące oprogramowania	15
--	----

Przykładowe kody	16
------------------------	----

W jaki sposób jest zorganizowana ta książka?	16
--	----

Użyte konwencje	18
-----------------------	----

Część I Podstawy

Rozdział 1. Dlaczego WPF?	21
--	-----------

Spojrzenie w przeszłość	22
-------------------------------	----

Początki WPF	23
--------------------	----

Ewolucja WPF	26
--------------------	----

Podsumowanie	30
--------------------	----

Rozdział 2. XAML bez tajemnic	31
--	-----------

Definicja XAML	33
----------------------	----

Elementy i atrybuty	34
---------------------------	----

Przestrzeń nazw	36
-----------------------	----

Elementy właściwości	39
----------------------------	----

Konwertery typów	40
------------------------	----

Rozszerzenia znaczników	42
-------------------------------	----

Elementy potomne obiektów	45
---------------------------------	----

Łączenie XAML z kodem proceduralnym	50
---	----

XAML2009	58
----------------	----

Słowa kluczowe XAML	63
---------------------------	----

Podsumowanie	66
--------------------	----

Rozdział 3. Podstawy WPF	67
Przegląd hierarchii klas	67
Drzewo logiczne i wizualne	69
Właściwości zależne	75
Podsumowanie	88
 Część II Tworzenie aplikacji WPF	
Rozdział 4. Sterowanie rozmiarem i pozycją elementów oraz transformacje	89
Kontrolowanie rozmiaru	90
Kontrolowanie pozycji	95
Stosowanie transformacji	98
Podsumowanie	108
 Rozdział 5. Układy z panelami	109
Element Canvas	110
Element StackPanel	112
Proste panele	131
Obsługa nadmiaru treści	132
Łączymy wszystko ze sobą — tworzenie zwijanego, dokowanego panelu o zmiennej wielkości, takiego jak w Visual Studio	140
Podsumowanie	151
 Rozdział 6. Zdarzenia wejściowe — klawiatura, mysz, piórko i dotknięcia	153
Zdarzenia kierowane	153
Zdarzenia klawiatury	162
Zdarzenia myszy	164
Zdarzenia piórka	167
Zdarzenia dotknięcia	170
Polecenia	181
Podsumowanie	186
 Rozdział 7. Struktura aplikacji i jej instalowanie	187
Standardowe aplikacje pulpitu	187
Klasa Application	191
Aplikacje pulpitu bazujące na nawigacji	203
Aplikacje w stylu gadżetów	214
Aplikacje XAML dla przeglądarki	216
Luźne strony XAML	223
Podsumowanie	224

Rozdział 8. Użycie funkcji pulpitu Windows	225
Listy szybkiego dostępu	225
Dostosowanie elementów paska zadań	237
Tryb Aero Glass	242
Podsumowanie	247

Część III Kontrolki

Rozdział 9. Kontrolki zawartości	249
Przyciski	251
Proste kontenery	255
Kontenery z nagłówkami	260
Podsumowanie	262

Rozdział 10. Kontrolki elementów	263
Wspólne funkcje	264
Selektory	269
Menu	305
Inne kontrolki elementów	309
Podsumowanie	315

Rozdział 11. Kontrolki obrazu, tekstu i inne	317
Kontrolka Image	317
Kontrolki tekstu i piórka	319
Dokumenty	326
Kontrolki zakresu	341
Kontrolka Calendar	343
Podsumowanie	346

Część IV Zaawansowane funkcje

Rozdział 12. Zasoby	347
Zasoby binarne	347
Zasoby logiczne	355
Podsumowanie	364

Rozdział 13. Wiązanie danych	367
Wprowadzenie do wiązania obiektów	367
Sterowanie wyświetlaniem	380
Modyfikowanie widoku kolekcji	391
Dostawcy danych	402

Zagadnienia zaawansowane	409
Łączymy wszystko ze sobą — klient Twittera w czystym XAML	418
Podsumowanie	420

Rozdział 14. Style, szablony, skórki i tematy 421

Style	422
Szablony	436
Skórki	462
Tematy	468
Podsumowanie	473

Część V Multimedia

Rozdział 15. Grafika dwuwymiarowa 475

Klasa Drawing	476
Klasa Visual	493
Klasa Shape	505
Klasa Brush	513
Efekty	530
Poprawianie wydajności renderowania	532
Podsumowanie	536

Rozdział 16. Grafika trójwymiarowa 537

Wprowadzenie do grafiki trójwymiarowej	538
Klasa Camera i układy współrzędnych	542
Klasa Transform3D	554
Klasa Model3D	564
Klasa Visual3D	588
Klasa Viewport3D	594
Przekształcenia współrzędnych dwu- i trójwymiarowych	597
Podsumowanie	604

Rozdział 17. Animacja 605

Animacje w kodzie proceduralnym	606
Animacje w XAML	618
Animacje typu keyframe	627
Funkcje ułatwiające	634
Animacje i Visual State Manager	639
Podsumowanie	646

Rozdział 18. Dźwięk, wideo i mowa	649
Dźwięk	649
Wideo	654
Mowa	659
Podsumowanie	666

Część VI Tematy zaawansowane

Rozdział 19. Współpraca z technologiami innymi niż WPF	667
Osadzanie kontrolki Win32 w aplikacjach WPF	670
Osadzanie kontrolki WPF w aplikacjach Win32	683
Osadzanie kontrolki Windows Forms w aplikacjach WPF	690
Osadzanie kontrolki WPF w aplikacjach Windows Forms	695
Łączenie zawartości DirectX z zawartością WPF	699
Osadzanie kontrolki ActiveX w aplikacjach WPF	704
Podsumowanie	708

Rozdział 20. Kontrolki użytkownika i kontrolki niestandardowe	711
Tworzenie kontrolki użytkownika	713
Tworzenie kontrolki niestandardowej	722
Podsumowanie	738

Rozdział 21. Układy z niestandardowymi panelami	739
Komunikacja między obiektami nadrzędnymi a podrzędnymi	740
Tworzenie panelu SimpleCanvas	744
Tworzenie panelu SimpleStackPanel	747
Tworzenie panelu OverlapPanel	750
Tworzenie panelu FanCanvas	755
Podsumowanie	758

Rozdział 22. Powiadomienia tostowe	759
Warunki wstępne	759
Wysyłanie powiadomienia tostowego	762
Szablony tostów	763
Zdarzenia powiadomień	766
Powiadomienia planowane	767
Podsumowanie	768

Dodatek A	Zabawa z odczytywaniem i zapisywaniem XAML	769
	Wprowadzenie	769
	Pętla węzłowa	772
	Odczytywanie XAML	773
	Zapisywanie do aktywnych obiektów	778
	Zapisywanie do XML	779
	Klasa XamlServices	780
	Skorowidz	783

ROZDZIAŁ 4.

Sterowanie rozmiarem i pozycją elementów oraz transformacje

W tym rozdziale:

- Kontrolowanie rozmiaru
- Kontrolowanie pozycji
- Stosowanie transformacji

Podczas budowania aplikacji WPF jednym z pierwszych zadań do wykonania jest ustawienie zestawu kontrolki na powierzchni aplikacji. Sterowanie wymiarami i pozycją kontrolki (jak również innych elementów) jest tworzeniem **układu**. WPF zawiera rozbudowaną infrastrukturę wspierającą bogaty w możliwości system układu.

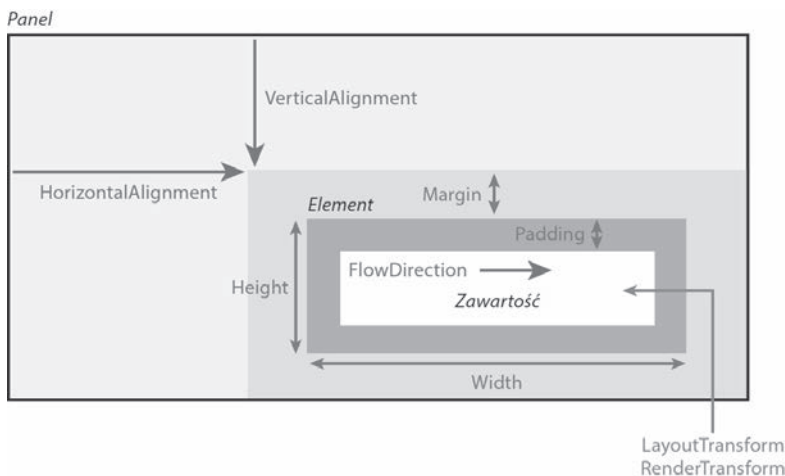
Układy w WPF bazują na interakcjach pomiędzy elementami nadrzędnymi a podrzędnymi. Elementy nadrzędne i ich elementy potomne współpracują ze sobą w celu określenia końcowej wielkości i położenia. Choć to element nadrzędny ostatecznie wskazuje elementowi podrzédnemu miejsce, w którym powinien być narysowany, oraz miejsce, jakie może zająć, to mimo wszystko jest to współpraca, a nie dyktatura. Obiekty nadrzędne *odpytują* podrzędne o ilość miejsca, jakiego potrzebują, i na tej podstawie podejmują końcową decyzję.

Elementy nadrzędne wspierające rozmieszczanie wielu elementów podrzędnych są nazywane **panelami** i dziedziczą po klasie `System.Windows.Controls.Panel`. Wszystkie elementy biorące udział w procesie tworzenia układu (zarówno elementy nadrzędne, jak i podrzędne) dziedziczą po `System.Windows.UIElement`.

Ponieważ układy w WPF są tak obszernym i ważnym tematem, w książce tej poświęcone im są trzy rozdziały:

- Rozdział 4., „Sterowanie rozmiarem i pozycją elementów oraz transformacje”
- Rozdział 5., „Układy z panelami”
- Rozdział 21., „Układy z niestandardowymi panelami”

W tym rozdziale skupię się na elementach podrzędnych, pokazując standardowe sposoby kontrolowania układu pomiędzy elementami podrzędnymi. Aspekt ten jest kontrolowany za pomocą kilku właściwości (których większość jest przedstawiona na rysunku 4.1) dla dowolnego elementu wewnątrz dowolnego panelu. Właściwości związane z wymiarami to `Margin`, `Padding`, `Height` i `Width`, a związane z położeniem — `VerticalAlignment`, `HorizontalAlignment` i `FlowDirection`. Ponadto elementy mogą mieć dodane do siebie transformacje (`LayoutTransform`, `RenderTransform`), które mogą wpływać zarówno na rozmiary, jak i położenie.



Rysunek 4.1. Główne właściwości układu elementów podrzędnych przedstawiane w tym rozdziale

W następnym rozdziale będziemy kontynuować temat, analizując różne panele dostępne w WPF, z których każdy ustawia elementy podrzędne na inny sposób. Tworzenie niestandardowych paneli jest zaawansowanym tematem, przedstawionym w końcowej części książki.

Kontrolowanie rozmiaru

Za każdym razem, gdy następuje wyznaczenie układu (na przykład zmieniony został rozmiar okna), elementy podrzędne przesyłają nadrzędnemu panelowi ich oczekiwane wymiary. Elementy WPF zwykle *dopasowują się do zawartości*, czyli próbują być wystarczająco duże, aby zmieścić całą zawartość, ale nie większe (nawet w `Window` działa w ten sposób, o ile jawnie ustawimy właściwość `SizeToContent`, jak zostało to zrobione w poprzednim rozdziale). Na rozmiary te można wpływać w konkretnych obiektach potomnych za pomocą kilku prostych właściwości.

Właściwości `Height` i `Width`

Wszystkie obiekty `FrameworkElement` mają proste właściwości `Height` oraz `Width` (typu `double`), jak również właściwości `MinHeight`, `MaxHeight`, `MinWidth` i `MaxWidth`, które pozwalają określić zakres akceptowalnych wartości. Wszystkie te właściwości można łatwo ustawić w elementach za pomocą kodu proceduralnego lub poprzez XAML.

Elementy w naturalny sposób pozostają możliwie małe, więc jeżeli użyjemy `MinHeight` lub `MinWidth`, będą miały podaną wysokość i szerokość, o ile ich zawartość nie wymusi większych rozmiarów. Dodatkowo wymiary te mogą być ograniczone za pomocą `MaxHeight` oraz `MaxWidth` (o ile wartości te są większe niż ich odpowiedniki zaczynające się od `Min`). Jeżeli jawnie określimy wielkość za pomocą `Height` i `Width` i jednocześnie użyjemy odpowiadających im właściwości `Min` lub `Max`, to wartości `Height` i `Width` będą miały pierwszeństwo, o ile mieszczą się w zakresie od `Min` do `Max`. Domyślną wartością `MinHeight` i `MinWidth` jest 0, a domyślną wartością `MaxHeight` i `MaxWidth` jest `Double.PositiveInfinity` (która może być ustawiona w XAML za pomocą "Infinity").

Ostrzeżenie



Unikaj jawnego ustawiania wymiarów!

Nadawanie kontrolkom jawnych wymiarów, szczególnie dziedziczących po `ContentControls`, takich jak `Button` i `Label`, niesie ze sobą ryzyko obcinania tekstu, gdy użytkownik zmieni ustawienia czcionki systemowej lub jeżeli tekst zostanie przetłumaczony na inny język. Dlatego warto unikać ustawiania na sztywno wielkości, o ile nie jest to absolutnie niezbędne. Na szczęście jawne ustawianie wielkości jest rzadko potrzebne — dzięki zastosowaniu paneli opisanych w następnym rozdziale.

SCHODZIMY GŁĘBIEJ

Specjalna długość "Auto"

Właściwości `Height` i `Width` klasy `FrameworkElement` mają domyślną wartość `Double.NaN` (gdzie `NaN` pochodzi od określenia *nie liczba*), co oznacza, że element będzie tak duży, jak potrzebuje jego zawartość. Ustawienie to może być również określone jawnie w XAML za pomocą "NaN" (wielkość liter ma znaczenie) lub preferowanego "Auto" (gdzie wielkość liter nie ma znaczenia) dzięki konwerterowi typów `LengthConverter` dołączonemu do tych właściwości. Aby sprawdzić, czy właściwości te są automatycznie wymiarowane, można użyć statycznej metody `Double.IsNaN`.

Aby nie było to tak proste, `FrameworkElement` zawiera kilka innych właściwości związanych z wielkością:

- `DesiredSize` (dziedziczący po `UIElement`),
- `RenderSize` (dziedziczący po `UIElement`),
- `ActualHeight` oraz `ActualWidth`.

W przeciwieństwie do pozostałych sześciu właściwości, które są *wejściem* do procesu określania układu, są to właściwości tylko do odczytu, reprezentujące *wyjście* z procesu tworzenia układu. Wartość `DesiredSize` dla elementu jest obliczana, gdy tworzony jest układ, na bazie wartości innych właściwości (takich jak wspomniane wcześniej właściwości `Width`, `Height`, `MinXXX` i `MaxXXX`) oraz ilości miejsca udostępnianego przez element nadrzędny. Jest ona używana wewnętrznie przez panele.

RenderSize reprezentuje końcowy rozmiar elementu po zakończeniu tworzenia układu, a ActualHeight i ActualWidth są dokładnie takie same jak RenderSize.Height i RenderSize.Width. To prawda: zależnie od tego, czy w elemencie określone będą jawnie wielkość, zakres akceptowalnych rozmiarów, czy nic nie zostanie zdefiniowane, zachowanie elementu nadrzędnego może zmienić wynikowy rozmiar na ekranie. Te trzy właściwości są więc użyteczne, gdy chcemy w programie korzystać z wielkości elementu. Wartości pozostałych właściwości związanych z rozmiarami nie są interesujące z punktu widzenia logiki. Na przykład, gdy nie ustawimy ich jawnie, wartością Height i Width jest Double.NaN, niezależnie od wielkości elementu.

Wszystkie te właściwości są przedstawione w rozdziale 21.

Ostrzeżenie



Należy zachować uwagę, pisząc kod korzystający z ActualHeight i ActualWidth (lub RenderSize)!

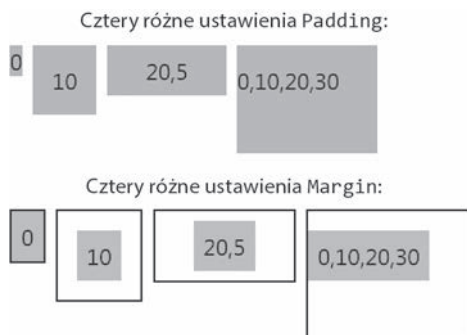
Za każdym razem, gdy zachodzi proces ładowania, aktualizowana jest wartość właściwości RenderSize każdego elementu (i jednocześnie ActualHeight oraz ActualWidth). Jednak tworzenie układu jest wykonywane asynchronicznie, więc nie można zawsze polegać na wartościach tych właściwości. Bezpiecznie można z nich korzystać wyłącznie w handlerze zdarzenia LayoutUpdated zdefiniowanego w UIElement.

Alternatywnie w elemencie UIElement zdefiniowana jest metoda UpdateLayout, która pozwala wymusić synchroniczne wykonanie oczekujących aktualizacji, ale należy unikać jej stosowania. Oprócz tego, że częste wywołania UpdateLayout mogą ograniczyć wydajność z powodu niepotrzebnego przetwarzania układu, to nie ma gwarancji, że używane elementy prawidłowo obsługują potencjalne ponowne wejście do metod związanych z układem.

Właściwości Margin i Padding

Właściwości Margin i Padding są dwoma podobnymi właściwościami, które są również związane z wielkością elementu. Wszystkie obiekty dziedziczące po FrameworkElement posiadają właściwość Margin, a wszystkie dziedziczące po Controls (jak również Border) mają właściwość Padding. Jedyna różnica jest taka, że Margin steruje ilością miejsca wokół *zewnątrznych* krawędzi elementu, natomiast Padding steruje ilością miejsca wokół jego *wewnętrznych* krawędzi.

Zarówno Margin, jak i Padding są typu System.Windows.Thickness, interesującej klasy, która może reprezentować jedną, dwie lub cztery wartości double. Znaczenie tych wartości jest zademonstrowane na listingu 4.1, na którym stosowane są różne ustawienia Padding i Margin do kontrolki Label. Drugi zestaw elementów Label ma przypisane wartości Border, ponieważ w przeciwnym razie ustawienia marginesów nie będą zauważalne. Na rysunku 4.2 pokazany jest wynik dla każdego elementu Label, gdyby był umieszczony na elemencie Canvas (panel opisany następnym rozdziałem). Choć nie jest to pokazane na rysunku, Margin pozwala na wartości ujemne, natomiast Padding — nie.



Rysunek 4.2. Efekt użycia Padding i Margin

Listing 4.1. Użycie Padding i Margin z jedną wartością, dwoma lub czterema wartościami

```
<!-- PADDING: -->

<!-- 1 wartość: to samo wypełnienie dla wszystkich czterech stron: -->
<Label Padding="0" Background="Orange">0</Label>
<Label Padding="10" Background="Orange">10</Label>

<!-- 2 wartości: lewa i prawa strona uzyskują pierwszą wartość,
góra i dół uzyskują drugą wartość: -->
<Label Padding="20,5" Background="Orange">20,5</Label>

<!-- 4 wartości: lewo,góra,pravo,dół: -->
<Label Padding="0,10,20,30" Background="Orange">0,10,20,30</Label>

<!-- MARGIN: -->

<Border BorderBrush="Black" BorderThickness="1">
  <!-- Brak marginesu: -->
  <Label Background="Aqua">0</Label>
</Border>

<Border BorderBrush="Black" BorderThickness="1">
  <!-- 1 wartość: ten sam margines dla wszystkich czterech stron: -->
  <Label Margin="10" Background="Aqua">10</Label>
</Border>

<Border BorderBrush="Black" BorderThickness="1">
  <!-- 2 wartości: lewa i prawa strona uzyskują pierwszą wartość,
góra i dół uzyskują drugą wartość: -->
  <Label Margin="20,5" Background="Aqua">20,5</Label>
</Border>

<Border BorderBrush="Black" BorderThickness="1">
  <!-- 4 wartości: lewo,góra,pravo,dół: -->
  <Label Margin="0,10,20,30" Background="Aqua">0,10,20,30</Label>
</Border>
```

Element `Label` ma domyślną wartość `Padding` równą 5, ale może być ona zmieniona na dowolną inną. Dlatego właśnie na listingu 4.1 jawnie ustawiamy wartość `Padding` w `Label` na 0. Bez tego jawnego ustawienia wyglądałby on jak piąty element `Label` (demonstrujący niejawną wartość `Margin` równą 0), a wizualne porównanie z pozostałymi wartościami `Padding` byłoby mylące.

SCHODZIMY GŁĘBIEJ

Składnia właściwości `Thickness`

Dostępna dla właściwości `Margin` i `Padding` składnia rozdzielana przecinkami jest udostępniana przez (cóż innego mogłoby to być) konwerter typów. `System.Windows.ThicknessConverter` tworzy obiekt `Thickness` bazujący na wprowadzanym ciągu znaków. Klasa `Thickness` posiada dwa konstruktory, jeden oczekuje pojedynczej wartości `double`, a drugi czterech. Dlatego może on być używany w C# w następujący sposób:

```
myLabel.Margin = new Thickness(10);           // Identycznie jak Margin="10" w XAML
myLabel.Margin = new Thickness(20,5,20,5);    // Identycznie jak Margin="20,5" w XAML
myLabel.Margin = new Thickness(0,10,20,30);    // Identycznie jak Margin="0,10,20,30" w XAML
```

Zwróć uwagę, że wygodna składnia z dwoma wartościami jest skrótem dostępnym tylko przez konwerter typów!

FAQ



Z jakich jednostek korzysta WPF?

Konwerter typów `LengthConverter` skojarzony z różnymi właściwościami długości obsługuje jednostki podawane jako `cm`, `pt`, `in` lub `px` (domyślnie).

Domyślnie wszystkie wymiary bezwzględne, takie jak liczby używane we właściwościach związanych z wymiarami, są podawane w *pikselach niezależnych od urządzenia*. Te „piksele logiczne” mają reprezentować 1/96 cala, niezależnie od ustawienia DPI ekranu. Pamiętaj, że piksele niezależne od urządzenia są zawsze podawane jako wartości `double`, więc mogą być ułamkowe.

Dokładne wymiary 1/96 cala nie są istotne, choć wymiar ten został wybrany na podstawie wyświetlacza 96 DPI, w którym piksele niezależne od urządzenia są identyczne z fizycznymi pikselami. Oczywiście zależy to również od wielkości wyświetlacza. Jeżeli aplikacja narysuje linię 1-calową na ekranie laptopa, to linia ta na pewno będzie dłuższa, gdy podłączę to tego laptopa projektor!

To, co *jest* ważne, to fakt, że wymiary te są niezależne od DPI. Jednak sama ta funkcja nie uniemożliwia zmniejszania elementów przy zwiększeniu rozdzielczości ekranu. Aby uzyskać niezależność od rozdzielczości, konieczna jest funkcja automatycznego skalowania, przedstawiona w następnym rozdziale.

Właściwość Visibility

Może wydawać się dziwne wspomnianie właściwości `Visibility` (zdefiniowanej w `UIElement`) w kontekście układu, ale faktycznie ma ona na niego wpływ. Właściwość `Visibility` nie jest typu `Boolean`, ale trójstanowego typu wyliczeniowego `System.Windows.Visibility`. Jego wartości mają następujące znaczenie:

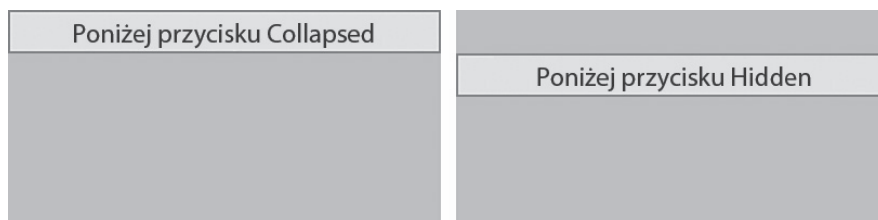
- **Visible** — element jest rysowany i wchodzi w skład układu,
- **Collapsed** — element jest niewidoczny i nie wchodzi w skład układu,
- **Hidden** — element jest niewidoczny, *ale wchodzi w skład układu*.

Element o widoczności `Collapsed` ma rozmiar zero, natomiast `Hidden` zachowuje swoją początkową wielkość (na przykład nie zmieniają się jego wartości `ActualHeight` i `ActualWidth`). Różnica pomiędzy `Collapsed` a `Hidden` jest pokazana na rysunku 4.3, na którym porównujemy następujący `StackPanel` z elementem `Button` o widoczności `Collapsed`:

```
<StackPanel Height="100" Background="Aqua">
  <Button Visibility="Collapsed">Przycisk Collapsed</Button>
  <Button>Poniżej przycisku Collapsed</Button>
</StackPanel>
```

z następującym elementem `StackPanel` zawierającym `Button` o widoczności `Hidden`:

```
<StackPanel Height="100" Background="Aqua">
  <Button Visibility="Hidden">Przycisk Hidden</Button>
  <Button>Poniżej przycisku Hidden</Button>
</StackPanel>
```



Rysunek 4.3. W przeciwieństwie do przycisku `Collapsed` przycisk `Hidden` nadal zajmuje miejsce

Kontrolowanie pozycji

W tym podrozdziale nie będziemy mówić o pozycjonowaniu elementów za pomocą współrzędnych (X,Y). Panele nadrzędne definiują własne, unikatowe mechanizmy pozwalające potomkom na samodzielne pozycjonowanie (za pomocą właściwości dołączanych lub po prostu kolejności dodawania do elementu nadrzędnego). Jednak kilka mechanizmów jest wspólnych dla wszystkich potomków `FrameworkElement` i właśnie one będą tematem tego podrozdziału. Mechanizmy te są związane z wyrównywaniem i koncepcją nazywaną **kierunkiem przepływu**.

Wyrównywanie

Właściwości `HorizontalAlignment` oraz `VerticalAlignment` pozwalają elementom sterować wykorzystaniem wolnego miejsca w panelu nadrzędnym. Każda właściwość posiada odpowiadający jej typ wyliczeniowy o tej samej nazwie, umieszczony w przestrzeni nazw `System.Windows`, dający nam następujące wartości:

- **`HorizontalAlignment`** — `Left`, `Center`, `Right` oraz `Stretch`,
- **`VerticalAlignment`** — `Top`, `Center`, `Bottom` oraz `Stretch`.

Domyślną wartością dla obu właściwości jest `Stretch`, choć wiele kontrolki zmienia to ustawienie w swoich stylach tematów. Efekt działania `HorizontalAlignment` można zobaczyć, umieszczając kilka elementów `Button` w `StackPanel` i nadając im różne wartości typu wyliczeniowego:

```
<StackPanel>
  <Button HorizontalAlignment="Left" Background="Red">Left</Button>
  <Button HorizontalAlignment="Center" Background="Orange">Center</Button>
  <Button HorizontalAlignment="Right" Background="Yellow">Right</Button>
  <Button HorizontalAlignment="Stretch" Background="Lime">Stretch</Button>
</StackPanel>
```

Wynik jest pokazany na rysunku 4.4.



Rysunek 4.4. Efekt działania `HorizontalAlignment` dla elementów `Button` wewnątrz `StackPanel`

Te dwie właściwości są użyteczne wyłącznie w przypadku, gdy panel nadrzędny daje potomkom więcej miejsca, niż potrzebują. Na przykład dodanie wartości `VerticalAlignment` do elementów wewnątrz `StackPanel` użytego na rysunku 4.4 nie spowoduje różnicy, ponieważ każdy element ma dokładnie taką wysokość, jaka jest potrzebna (nie mniej, nie więcej).

SCHODZIMY GŁĘBIEJ

Interakcja pomiędzy wyrównaniem `Stretch` a jawnym ustawieniem rozmiaru elementu

Gdy element korzysta z wyrównania `Stretch` (poziomo lub pionowo), to jawne ustawienie właściwości `Width` lub `Height` ma wyższy priorytet. Również `MaxWidth` i `MaxHeight` mają wyższy priorytet, ale wyłącznie wtedy, gdy te wartości są mniejsze niż uzyskane na podstawie rozciągnięcia. Podobnie `MinWidth` i `MinHeight` mają wyższy priorytet, ale wyłącznie wtedy, gdy te wartości są większe niż uzyskane na podstawie rozciągnięcia. Gdy wartość `Stretch` jest użyta w kontekście ograniczenia wielkości elementu, działa jak wyrównanie `Center` (lub `Left`, jeżeli element jest zbyt duży, aby mógł być wycentrowany w elemencie nadrzędnym).

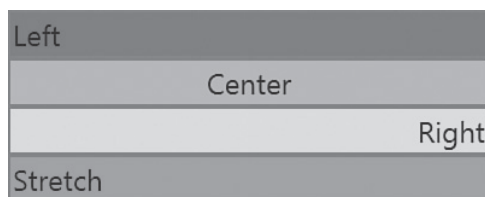
Wyrównanie zawartości

W klasie `Control` oprócz właściwości `HorizontalAlignment` i `VerticalAlignment` zdefiniowane są również właściwości `HorizontalContentAlignment` i `VerticalContentAlignment`. Właściwości te określają, w jaki sposób można sterować wypełnianiem przestrzeni przez zawartość wewnątrz kontrolki (dlatego relacja pomiędzy wyrównaniem i wyrównaniem zawartości jest podobna jak relacja między `Margin` a `Padding`).

Właściwości wyrównania zawartości korzystają z tych samych typów wyliczeniowych co odpowiednie właściwości wyrównania, więc zapewniają te same opcje. Jednak domyślną wartością dla `HorizontalContentAlignment` jest `Left`, a domyślną wartością dla `VerticalContentAlignment` jest `Top`. Ale nie dzieje się tak w przypadku poprzednio pokazanych kontrolki `Button`, ponieważ ich styl tematu nadpisuje te ustawienia (przypomnij sobie kolejność priorytetów dla dostawców wartości właściwości z poprzedniego rozdziału — wartości domyślne mają najniższy priorytet i są przebijane przez style).

Na rysunku 4.5 pokazany jest efekt użycia `HorizontalContentAlignment`. Jest to poprzedni fragment XAML — ma tylko zmienioną nazwę właściwości:

```
<StackPanel>
  <Button HorizontalContentAlignment="Left" Background="Red">Left</Button>
  <Button HorizontalContentAlignment="Center" Background="Orange">Center</Button>
  <Button HorizontalContentAlignment="Right" Background="Yellow">Right</Button>
  <Button HorizontalContentAlignment="Stretch" Background="Lime">Stretch</Button>
</StackPanel>
```



Rysunek 4.5. Efekt działania `HorizontalContentAlignment` dla elementów `Button` wewnątrz `StackPanel`

Na rysunku 4.5 element `Button` z `HorizontalContentAlignment="Stretch"` może wyglądać inaczej, niż się spodziewamy. Wewnętrzny `TextBlock` jest rzeczywiście rozciągnięty, ale `TextBlock` nie jest prawdziwym elementem `Control` (lecz tylko `FrameworkElement`), więc nie ma tej samej notacji dla rozciągania wewnętrznego tekstu.

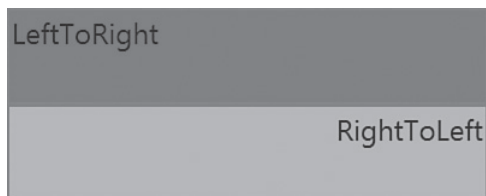
Właściwość `FlowDirection`

Właściwość `FlowDirection` jest właściwością `FrameworkElement` (i kilku innych klas), która może odwrócić sposób, w jaki przepływa wewnętrzna zawartość elementu. Odnosi się to do niektórych paneli i ich pozycjonowania potomków, jak również do sposobu, w jaki zawartość jest wyrównywana wewnątrz kontrolki potomnych. Właściwość ta jest typu `System.Windows.FlowDirection`, który zawiera dwie wartości: `LeftToRight` (domyślny dla `FrameworkElement`) oraz `RightToLeft`.

W założeniach właściwość `FlowDirection` powinna być ustawiona na `RightToLeft`, gdy bieżące ustawienia regionalne wskazują na język, który jest czytany od prawej do lewej. Powoduje to zamianę znaczenia „w lewo” i „w prawo” w przypadku ustawień takich jak wyrównywanie zawartości. Można to zademonstrować za pomocą poniższego kodu XAML, w którym elementy `Button` wymuszają wyrównanie zawartości na `Top` i `Left`, a następnie stosowane są dwie wartości `FlowDirection`:

```
<StackPanel>
  <Button FlowDirection="LeftToRight"
    HorizontalContentAlignment="Left" VerticalContentAlignment="Top"
    Height="40" Background="Red">LeftToRight</Button>
  <Button FlowDirection="RightToLeft"
    HorizontalContentAlignment="Left" VerticalContentAlignment="Top"
    Height="40" Background="Orange">RightToLeft</Button>
</StackPanel>
```

Wynik jest pokazany na rysunku 4.6.



Rysunek 4.6. Efekt działania `FlowDirection` dla elementów `Button` z wyrównaniem zawartości `Top` i `Left`

Zwróć uwagę, że `FlowDirection` nie wpływa na kolejność liter w elementach `Button`. Litery angielskie zawsze są ułożone od lewej do prawej, a arabskie zawsze od prawej do lewej. Jednak `FlowDirection` odwraca ten porządek dla pozostałych elementów interfejsu użytkownika, które zwykle muszą pasować do kierunku liter.

Właściwość `FlowDirection` musi być jawnie ustawiona tak, aby pasowała do bieżących ustawień regionalnych (może to być wykonane w głównym elemencie). Powinien to być element procesu lokalizacji.

Stosowanie transformacji

WPF zawiera kilka wbudowanych transformacji 2D (dziedziczących po `System.Windows.Media.Transform`), które pozwalają na zmianę rozmiaru i położenia elementów niezależnie od przedstawionych wcześniej właściwości. Niektóre pozwalają zmieniać elementy w bardziej egzotyczne sposoby, na przykład obracając je lub wykrzywiając.

Wszystkie klasy `FrameworkElement` mają dwie właściwości typu `Transform`, które mogą być używane do stosowania tych transformacji:

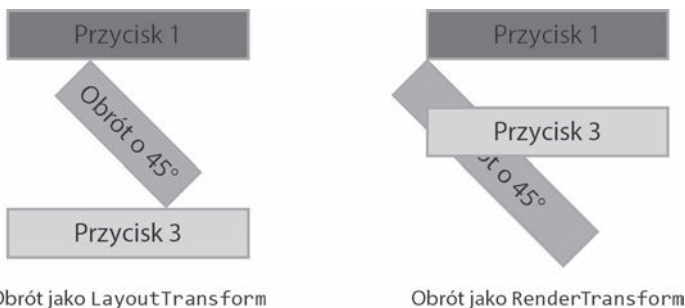
- `LayoutTransform`, która jest stosowana *przed* ułożeniem elementu,
- `RenderTransform` (dziedziczone po `UIElement`), która jest stosowana *po* zakończeniu procesu określania układu (bezpośrednio przed narysowaniem elementu).

FAQ

**Jak korzystać z transformacji 3D?**

W Silverlight oraz XAML dla aplikacji Windows Store elementy mają właściwość `Projection`, która pozwala na łatwe wykonywanie transformacji perspektywy. Jednak elementy WPF nie mają tej właściwości. Zamiast tego konieczne jest użycie funkcji 3D opisanych w rozdziale 16., „Grafika trójwymiarowa”.

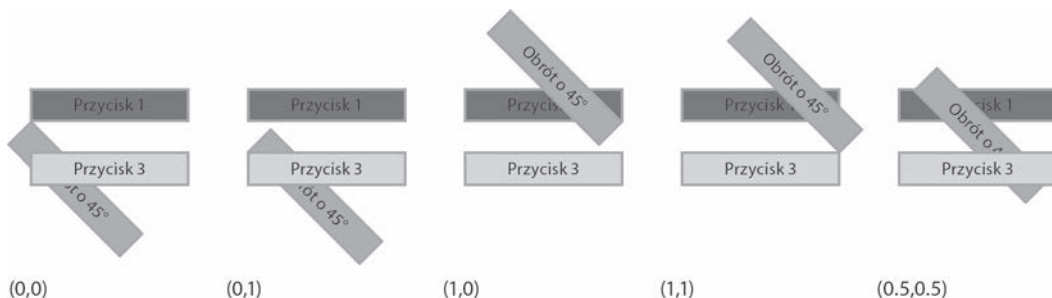
Na rysunku 4.7 zademonstrowana jest różnica pomiędzy stosowaniem transformacji o nazwie `Rotate` ➔ `Transform` jako `LayoutTransform` a użyciem transformacji `RenderTransform`. W obu przypadkach transformacja jest stosowana w drugim z trzech kolejnych elementów `Button` ze `StackPanel`. Gdy jest wykorzystywana jako `LayoutTransform`, trzeci element `Button` jest odpychany, a szerokość drugiego elementu `Button` nie jest ograniczana przez szerokość `StackPanel`. Gdy jest ona stosowana jako `RenderTransform`, trzeci element `Button` jest umieszczany tak, jakby drugi element `Button` nie był obracany, a szerokość drugiego elementu `Button` jest szerokością `StackPanel`.



Rysunek 4.7. Różnica pomiędzy `LayoutTransform` a `RenderTransform` widoczna dla środkowego elementu `Button` w `StackPanel`

Klasa `UIElement` posiada również wygodną właściwość `RenderTransformOrigin`, reprezentującą punkt startowy transformacji (punkt, który pozostaje nieruchomy). W przypadku transformacji `RotateTransform` użytej na rysunku 4.7 punktem początkowym dla przycisku jest lewy górny narożnik, wokół którego obracany jest element. Z kolei transformacje `LayoutTransform` nie mają punktu początkowego, ponieważ położenie transformowanego elementu jest określane przez zasady tworzenia układu panelu nadrzędnego.

Właściwość `RenderTransformOrigin` może mieć przypisany obiekt `System.Windows.Point`, którego domyślną wartością jest `(0,0)`. Reprezentuje to lewy górny narożnik, jak jest to pokazane na rysunku 4.7. Punkt początkowy `(0,1)` reprezentuje lewy dolny narożnik, `(1,0)` górny prawy, a `(1,1)` to prawy dolny narożnik. Aby ustawić punkt początkowy poza granicami elementu, należy użyć wartości większej od 1, a aby ustawić go wewnątrz elementu, należy użyć wartości ułamkowych. Dlatego `(0.5,0.5)` reprezentuje środek obiektu. Na rysunku 4.8 przedstawiono pięć często używanych punktów początkowych używanych dla `RenderTransform` z rysunku 4.7.



Rysunek 4.8. Pięć często używanych wartości `RenderTransformOrigin` wykorzystanych dla obracanego elementu `Button` z rysunku 4.7

Dzięki `System.Windows.PointConverter` wartość `RenderTransformOrigin` może być zdefiniowana w XAML za pomocą dwóch liczb oddzielonych przecinkami (bez nawiasów). Na przykład obiekt `Button` obracany wokół swojego środka, tak jak po prawej stronie rysunku 4.8, może być utworzony w następujący sposób:

```
<Button RenderTransformOrigin="0.5,0.5" Background="Orange">
<Button.RenderTransform>
  <RotateTransform Angle="45"/>
</Button.RenderTransform>
  Obrót o 45°
</Button>
```

Można się zastanawiać, dlaczego ktoś chciałby mieć obracający się przycisk w aplikacji! Faktycznie, transformacja taka wygląda głupio w przypadku standardowych kontrolki z domyślnym stylem. Może to mieć większy sens dla aplikacji intensywnie korzystającej z tematów, ale nawet przy domyślnym stylu kontrolki transformacje mogą stanowić przyjemny element wykończeniowy, szczególnie gdy będą używane z animacjami.

W tym podrozdziale przedstawię pięć wbudowanych transformacji 2D, znajdujących się w przestrzeni nazw `System.Windows.Media`:

- `RotateTransform`
- `ScaleTransform`
- `SkewTransform`
- `TranslateTransform`
- `MatrixTransform`

Transformacja `RotateTransform`

Transformacja `RotateTransform`, zademonstrowana w poprzednim punkcie, pozwala na obracanie elementu zgodnie z wartościami trzech właściwości typu `double`:

- **Angle** — kąt obrotu, wyrażany w stopniach (domyślna wartość = 0),
- **CenterX** — środek obrotu poziomo (domyślna wartość = 0),
- **CenterY** — środek obrotu pionowo (domyślna wartość = 0).

Domyślnie punkt (CenterX,CenterY) o współrzędnych (0,0) reprezentuje lewy górny narożnik. Właściwości CenterX i CenterY są przydatne wyłącznie wtedy, gdy transformacja RotateTransform jest używana jako RenderTransform, ponieważ gdy jest używana jako LayoutTransform, pozycja jest i tak określana przez panel nadrzędny.

FAQ



Jaka jest różnica pomiędzy użyciem właściwości CenterX i CenterY w transformacji takiej jak RotateTransform a użyciem właściwości RenderTransformOrigin z UIElement?

Gdy transformacja jest stosowana do UIElement, właściwości CenterX i CenterY wydają się początkowo nadmiarowe, ponieważ istnieje właściwość RenderTransformOrigin. Oba mechanizmy sterują punktem początkowym transformacji i oba działają tylko w przypadku, gdy transformacja jest używana jako RenderTransform.

Jednak CenterX i CenterY pozwalają na bezwzględne pozycjonowanie punktu początkowego w przeciwieństwie do względnego pozycjonowania w RenderTransformOrigin. Ich wartości są podawane z użyciem pikseli niezależnych od urządzenia, więc prawy górny narożnik elementu o długości 20 może być wskazany przez podanie 20 w CenterX i 0 w CenterY, a nie punktu (1,0). Dodatkowo, gdy łączone jest wiele transformacji RenderTransform (co zostanie opisane w dalszej części rozdziału), właściwości CenterX i CenterY w poszczególnych transformacjach pozwalają na precyzyjne sterowanie procesem. Na koniec pojedyncze wartości double w CenterX i CenterY są łatwiejsze do użycia przy wiązaniu danych niż wartość Point w RenderTransformOrigin.

Z tych powodów właściwość RenderTransformOrigin jest zwykle mniej użyteczna niż CenterX i CenterY. W przypadku często wykorzystywanego obrotu elementów wokół ich środka względna wartość (0.5,0.5) właściwości RenderTransformOrigin jest łatwa do wpisania w XAML, natomiast wykonanie tego samego za pomocą CenterX i CenterY wymaga napisania kodu proceduralnego obliczającego bezwzględne przesunięcie.

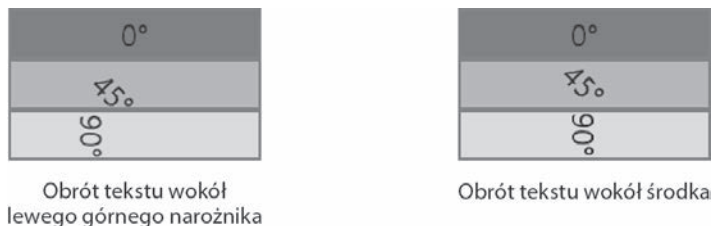
Zwróć uwagę, że możliwe jest używanie RenderTransformOrigin dla elementu jednocześnie z CenterX oraz CenterY dla transformacji. W tym przypadku dwie wartości X i Y są ze sobą łączone, aby wyliczyć ostateczny punkt początkowy.

Na rysunkach 4.7 i 4.8 pokazane są obrócone elementy Button, natomiast na rysunku 4.9 widać, co się stanie, gdy transformacja RotateTransform będzie użyta jako RenderTransform *na wewnętrznej zawartości* elementu Button przy dwóch różnych wartościach RenderTransformOrigin. Aby to osiągnąć, należy zamienić prosty napis wewnątrz elementu Button na jawnie zdefiniowany element TextBlock:

```
<Button Background="Orange">
  <TextBlock RenderTransformOrigin="0.5,0.5">
    <TextBlock.RenderTransform>
      <RotateTransform Angle="45"/>
    </TextBlock.RenderTransform>
  </TextBlock>
</Button>
```

Rysunek 4.9.

Użycie `RotateTransform` na zawartości elementów `Button` w `StackPanel`



Element `TextBlock` wewnątrz `Button` po lewej stronie rysunku 4.9 może nie wydawać się obrócony wokół lewego górnego narożnika, ponieważ elementy `TextBlock` są nieco większe niż tekst. Gdy nadamy w elemencie `TextBlock` wartość właściwości `Background`, obrót zacznie mieć sens. Jest to pokazane na rysunku 4.10.

Rysunek 4.10.

Wewnętrzne elementy `TextBlock` ze zmienionym tłem, obrócone wokół lewego górnego narożnika



Transformacja `RotateTransform` posiada konstruktory z parametrami, które pozwalają na podanie kąta lub kąta oraz środka obrotu, co upraszcza tworzenie transformacji w kodzie proceduralnym.

Transformacja `ScaleTransform`

Transformacja `ScaleTransform` pozwala na powiększanie lub zmniejszanie elementu w pionie, poziomie lub w obu kierunkach. Transformacja ta posiada cztery proste właściwości typu `double`:

- **ScaleX** — mnożnik dla szerokości elementu (wartość domyślna = 1),
- **ScaleY** — mnożnik dla wysokości elementu (wartość domyślna = 1),
- **CenterX** — środek dla skalowania w poziomie (domyślna wartość = 0),
- **CenterY** — środek dla skalowania w pionie (domyślna wartość = 0).

Wartość 0.5 w `ScaleX` powoduje zmniejszenie elementu o połowę, natomiast wartość 2 w `ScaleX` powiększa jego szerokość. Właściwości `CenterX` oraz `CenterY` działają w ten sam sposób co w przypadku `RotateTransform`.

Na listingu 4.2 stosujemy `ScaleTransform` do trzech elementów `Button` w `StackPanel`, demonstrując możliwość niezależnego skalowania ich wysokości lub szerokości. Wynik jest pokazany na rysunku 4.11.

Rysunek 4.11.

Skalowanie elementów `Button` z listingu 4.2



Listing 4.2. Użycie `ScaleTransform` do elementów `Button` w `StackPanel`

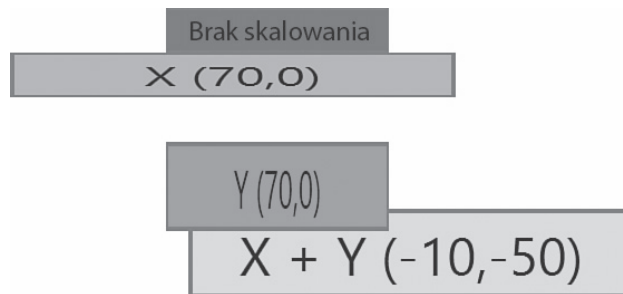
```

<StackPanel Width="100">
  <Button Background="Red">Brak skalowania</Button>
  <Button Background="Orange">
    <Button.RenderTransform>
      <ScaleTransform ScaleX="2"/>
    </Button.RenderTransform>
  </Button>
  <Button Background="Yellow">
    <Button.RenderTransform>
      <ScaleTransform ScaleX="2" ScaleY="2"/>
    </Button.RenderTransform>
  </Button>
  <Button Background="Lime">
    <Button.RenderTransform>
      <ScaleTransform ScaleY="2"/>
    </Button.RenderTransform>
  </Button>
</StackPanel>

```

Na rysunku 4.12 przedstawione są te same elementy `Button` z listingu 4.2 (i rysunku 4.11), ale z jawnie ustawionymi wartościami `CenterX` oraz `CenterY`. Punkt reprezentowany przez każdą parę jest wyświetlany w każdym z elementów `Button`. Zwróć uwagę, że zielony element `Button` (trzeci od góry) nie jest przesunięty w prawo tak jak pomarańczowy (drugi od góry), pomimo że użyta jest ta sama wartość `CenterX`, równa 70. Dzieje się tak, ponieważ `CenterX` ma znaczenie tylko wtedy, gdy `ScaleX` ma wartość inną niż 1, a `CenterY` ma znaczenie tylko wtedy, gdy `ScaleY` jest wartością inną niż 1.

Rysunek 4.12.
Przyciski z listingu 4.2,
ale z jawnie podanymi
środkami skalowania



Tak jak w przypadku innych transformacji klasa `ScaleTransform` posiada kilka konstruktorów z parametrami ułatwiającymi tworzenie obiektów w kodzie proceduralnym.

SCHODZIMY GŁĘBIEJ

Interakcja pomiędzy `ScaleTransform` a wyrównaniem `Stretch`

Gdy użyjemy `ScaleTransform` jako `LayoutTransform` na elemencie, który jest już rozciągnięty w wymiarach skalowania, da to efekty tylko wtedy, gdy poziom skalowania będzie większy niż naturalna wielkość rozciągniętego już elementu.

FAQ

**Jak transformacje takie jak `ScaleTransform` wpływają na właściwości `ActualHeight` i `ActualWidth` w `FrameworkElement` lub właściwość `RenderSize` z `UIElement`?**

Użycie transformacji na `FrameworkElement` nigdy nie zmienia wartości tych właściwości. Dzieje się tak, gdy jest ona użyta jako `RenderTransform` lub `LayoutTransform`. Dlatego z powodu transformacji właściwości te mogą przekłamywać faktyczną wielkość elementów na ekranie. Na przykład wszystkie elementy `Button` na rysunkach 4.11 oraz 4.12 mają te same wartości `ActualHeight`, `ActualWidth` oraz `RenderSize`.

Te „kłamstwa” mogą Cię zaskoczyć, ale są one w dobrej wierze. Dyskusyjne jest, czy takie wartości powinny być prezentowane dla niektórych transformacji. Co ważniejsze, zadaniem transformacji jest zmiana wyglądu elementu bez „wiedzy” samego elementu. Dając elementom „wrażenie”, że są rysowane normalnie, pozwala to na dołączenie dowolnej kontrolki i jej transformację bez specjalnego traktowania.

FAQ

**W jaki sposób `ScaleTransform` wpływa na `Margin` oraz `Padding`?**

Właściwość `Padding` jest skalowana wraz z resztą zawartości (ponieważ `Padding` jest wewnętrznym składnikiem elementu), ale właściwość `Margin` nie jest skalowana. Podobnie jak `ActualHeight` i `ActualWidth`, numeryczna właściwość `Padding` nie zmienia się pomimo wizualnego skalowania.

Transformacja `SkewTransform`

Dzięki transformacji `SkewTransform` możemy pochylić element w sposób definiowany za pomocą czterech właściwości typu `double`:

- **`AngleX`** — stopień pochylenia w poziomie (wartość domyślna = 0),
- **`AngleY`** — stopień pochylenia w pionie (wartość domyślna = 0),
- **`CenterX`** — środek dla pochylenia w poziomie (domyślna wartość = 0),
- **`CenterY`** — środek dla pochylenia w pionie (domyślna wartość = 0).

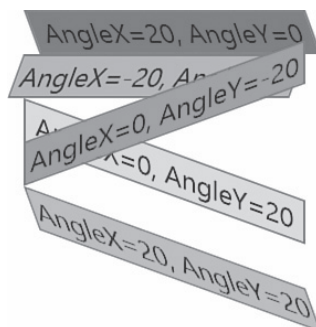
Właściwości te działają bardzo podobnie do poprzednich transformacji. Na rysunku 4.13 przedstawiony jest wynik użycia transformacji `SkewTransform` zastosowanej jako `RenderTransform` na kilku elementach `Button` przy użyciu domyślnego środka pochylenia w lewym górnym rogu.

Transformacja `TranslateTransform`

Dzięki translacji `TranslateTransform` możemy przesunąć element zgodnie z dwoma właściwościami typu `double`:

- **`X`** — wartość przesunięcia w poziomie (wartość domyślna = 0),
- **`Y`** — wartość przesunięcia w pionie (wartość domyślna = 0).

Rysunek 4.13.
Zastosowanie
SkewTransform
do elementów Button
w StackPanel



Translacja `TranslateTransform` nie działa, gdy zostanie użyta jako `LayoutTransform`, ale zastosowanie jej jako `RenderTransform` pozwala na „szturchnięcie” elementu w wybraną stronę. Najbardziej prawdopodobne jest, że wykonane to będzie w sposób dynamiczny, w zależności od akcji użytkownika (lub animacji). Jeśli chodzi o wszystkie panele opisane w następnym rozdziale, raczej nie będzie potrzebna transformacja `TranslateTransform` do utworzenia statycznego interfejsu użytkownika.

Transformacja `MatrixTransform`

Translacja `MatrixTransform` jest mechanizmem niskiego poziomu, który może być używany do tworzenia własnych transformacji 2D. Posiada ona jedną właściwość `Matrix` (typu `System.Windows.Media.Matrix`) reprezentującą macierz transformacji afinicznej o rozmiarach 3×3 . Jeżeli nie jesteś miłośnikiem algebry liniowej, powinieneś wiedzieć, że wszystkie poprzednie transformacje (lub dowolna ich kombinacja) mogą być wyrażone za pomocą `MatrixTransform`.

Macierz 3×3 zawiera następujące wartości:

$$\begin{bmatrix} M11 & M12 & 0 \\ M21 & M22 & 0 \\ \text{OffsetX} & \text{OffsetY} & 1 \end{bmatrix}$$

Wartości ostatniej kolumny są stałe, ale pozostałe sześć wartości może być ustawiane we właściwości typu `Matrix` (o pokazanych nazwach) lub poprzez konstruktor z sześcioma parametrami odpowiadającymi wartościom z macierzy w kolejności wierszy.

SCHODZIMY GŁĘBIEJ

Konwerter typów dla `MatrixTransform`

Transformacja `MatrixTransform` jest jedyną transformacją, która ma zdefiniowany konwerter typów, co pozwala nam użyć go jako prostego ciągu znaków w XAML (ten konwerter typów nosi nazwę `TransformConverter` i jest skojarzony z abstrakcyjną klasą `Transform`, ale obsługuje wyłącznie `MatrixTransform`). Na przykład można przesunąć element `Button` o 10 jednostek w prawo i 20 jednostek w dół za pomocą następującej składni:

```
<Button RenderTransform="1,0,0,1,10,20" />
```

ciąg dalszy

Lista ta reprezentuje wartości M11, M12, M21, M22, OffsetX oraz OffsetY. Wartości 1, 0, 0, 1, 0, 0 tworzą macierz jednostkową (oznaczającą brak transformacji), więc aby transformacja `MatrixTransform` działała tak jak `TranslateTransform`, wystarczy użyć macierzy jednostkowej, a następnie dodać wartości OffsetX i OffsetY jako wartości X i Y w `TranslateTransform`. Skalowanie może być wykonane przez użycie pierwszej i czwartej wartości (1 w macierzy jednostkowej) jako `ScaleX` i `ScaleY`. Obroty i pochylenie są bardziej skomplikowane, ponieważ wymagają użycia sinusów, cosinusów i kątów wyrażonych w radianach.

Jeżeli potrafisz korzystać z notacji macierzowej, to ten spójny (choć mniej czytelny) sposób reprezentowania transformacji pozwoli Ci zaoszczędzić dużo czasu, gdy wpisujesz XAML ręcznie.

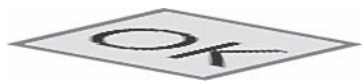
Łączenie transformacji

Łączenie wielu transformacji, na przykład obracanie elementu w czasie jednoczesnego skalowania, może być wykonane na kilka różnych sposobów. Można jednocześnie użyć `LayoutTransform` oraz `RenderTransform`. Można również użyć reprezentacji `MatrixTransform` do uzyskania połączonego efektu. Najlepiej jednak skorzystać z możliwości klasy `TransformGroup`.

`TransformGroup` jest kolejną klasą dziedziczącą po `Transform` (więc może być użyta wszędzie tam, gdzie stosowane są transformacje), a jej zadaniem jest łączenie podrzędnych obiektów `Transform`. W kodzie proceduralnym można dodać transformacje do kolekcji `Children`, natomiast zapis w XAML wygląda następująco:

```
<Button>
<Button.RenderTransform>
  <TransformGroup>
    <RotateTransform Angle="45"/>
    <ScaleTransform ScaleX="5" ScaleY="1"/>
    <SkewTransform AngleX="30"/>
  </TransformGroup>
</Button.RenderTransform>
OK
</Button>
```

Na rysunku 4.14 pokazany jest wynik tych trzech transformacji zastosowanych dla elementu `Button`.



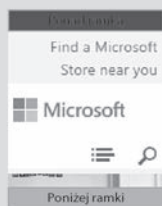
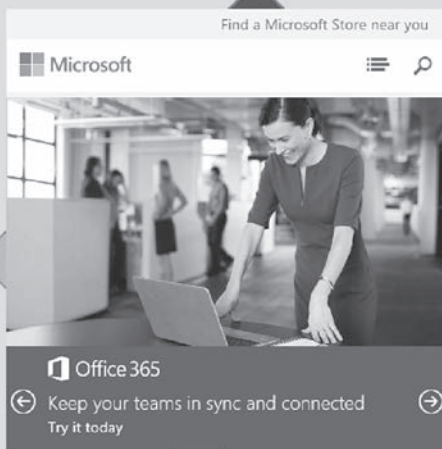
Rysunek 4.14. Element `Button`, który został obrócony, przeskalowany i pochyłony

Dla zapewnienia maksymalnej wydajności WPF wylicza połączoną transformację na podstawie podrzędnych elementów z `TransformGroup`, a następnie stosuje tę wynikową (tak jak w przypadku samodzielnego użycia `MatrixTransform`). Zwróć uwagę, że do `TransformGroup` można dodać wiele instancji tej samej transformacji. Na przykład użycie dwóch osobnych instancji `RotateTransform` o kącie 45° daje obrót o 90°.

Ostrzeżenie**Nie wszystkie obiekty FrameworkElement obsługują transformacje!**

Elementy zawierające treść, która nie pochodzi z WPF, nie obsługują transformacji, mimo że posiadają właściwości `LayoutTransform` i `RenderTransform`. Na przykład element `HwndHost` używany do osadzania zawartości GDI i opisany w rozdziale 19., „Współpraca z technologiami innymi niż WPF”, nie obsługuje ich. Element `Frame`, kontrolka pozwalająca na osadzenie HTML (opisana w rozdziale 9., „Kontrolki zawartości”), obsługuje je, o ile nie zawiera kodu HTML. W przeciwnym razie `ScaleTransform` może być używana do zmiany wielkości, ale wewnętrzna zawartość nie podlega skalowaniu.

Na rysunku 4.15 przedstawiony jest `StackPanel` zawierający kilka elementów `Button` oraz element `Frame` zawierający stronę WWW (ograniczony do wielkości 100×100). Gdy cały `StackPanel` jest obracany i skalowany, element `Frame` próbuje się skalować, ale się nie obraca. W efekcie tego ukrywa większość obróconych przycisków.

Normalny `StackPanel``StackPanel` z `RotateTransform` oraz `ScaleTransform`

Rysunek 4.15. Element `Frame` z zawartością HTML odpowiada na `ScaleTransform`, ale ignoruje pozostałe transformacje

Podsumowanie

Na tym kończymy przegląd właściwości układu, z których elementy potomne mogą korzystać w celu zmiany wyglądu na ekranie. W tym rozdziale zapoznaliśmy się również z pierwszymi elementami interfejsu użytkownika, które nie występują w Win32 ani Windows Forms: obróconymi i pochylonymi kontrolkami!

Jednak najważniejszym elementem układów są nadrzędne panele. W tym rozdziale dla uproszczenia korzystaliśmy wyłącznie ze `StackPanel`, ale w następnym wprowadzimy obok niego również pozostałe panele.

Skorowidz

.NET Framework 4.5, 15

A

Aero Glass, 242, 244

akapit, 332

akceleracja sprzętowa, 25

akcja budowania

Content, 348

Resource, 348

aktualizacja wielkości kontrolki, 687

analiza różnic, 359

animacja, 605

Visual State Manager, 639

bazująca na liczniku, 607

bazująca na ścieżkach, 633

dyskretna keyframe, 631

funkcje ułatwiające, 634

keyframe, 627, 630

okna, 613

wiążanie danych, 629

w kodzie proceduralnym, 606

w kształcie zygzaka, 628

w XAML, 618

z wykorzystaniem klatek, 607

animowanie

położenia elementu, 624

ręczne, 606

wielkości elementu, 624

API Win32, 676

aplikacja Galeria zdjęć, 244

aplikacje

bazujące na WPF, 26

ClickOnce, 217

desktopowe, 760, 768

DirectX, 25

dla przeglądarki, 216

pulpitu, 187, 203

pulpitu Windows, 225

w stylu gadżetów, 214

wielowątkowe, 196

Win32, 683

Windows Forms, 695

WPF, 690

XAML, 216–219

XBAP, 216

aranżowanie, 742

artefakty, 546

asynchroniczne dołączanie danych, 407

atrybut

specjalny, 63

STAThreadAttribute, 685

ThemeInfoAttribute, 471

właściwości, 35

zdarzenia, 35

automatyczne

rozpakowywanie, 409

wymiarowanie, 123

B

BAML, Binary Application Markup Language, 55, 58
 bąbelkowanie, 158
 bezpieczeństwo, 221
 bezpośrednie odwołanie do zasobów, 363
 biblioteka
 DirectX, 22
 OpenGL, 22
 Pixar Shader Effects Library, 532
 System.Speech.dll, 660, 663
 WPF Shell Integration, 248
 Binding, 399
 Block, 328
 BlockUIContainer, 329
 bloki zakotwiczone, 333
 błąd, 414
 HRESULT, 61
 poprawności, 431
 błędy wiązania danych, 389

C

Calendar, 343
 CAML, Compiled Application Markup Language, 56
 Canvas, 110
 dołączane właściwości, 111
 przyciski, 111
 właściwości elementów podrzędnych, 111
 centrowanie skalowania, 439
 cieniowanie, 563, 584
 ClickOnce, 202, 217
 clipart, 491, 511
 ComboBox, 269, 451
 ComboBoxItem, 274
 ContentControl, 250
 ContextMenu, 308
 ControlTemplate, 443
 czas
 drzemki, 767
 rozruchu na zimno, 197
 trwania animacji, 611, 617
 części kontrolki, 451
 część kontrolna, 732
 czyszczenie wartości lokalnej, 83

czytnik

RSS, 418, 419
 XAML, 773

D

dane XML, 404
 DataContext, 379
 DataGrid, 279
 dodatkowe dane wiersza, 283
 dodawanie danych, 284
 edycja danych, 284
 interakcja ze schowkiem, 283
 kolumny generowane automatycznie, 281
 typy kolumn, 281
 usuwanie danych, 284
 wirtualizacja, 283
 zamrażanie kolumn, 284
 zaznaczanie wierszy, 282
 DatePicker, 345
 debugger, 686
 definiowanie
 gramatyki, 666
 rotacji, 559
 właściwości, 62
 zachowania kontrolki, 715
 zasobów, 362
 zasobów binarnych, 348
 deklaracja PriorityBinding, 418
 dekompilowanie, 57
 dekorator, 137
 diagnozowanie błędów, 659
 DirectX, 22, 25, 699
 DockPanel, 117
 Dock, 117
 przyciski, 117, 118
 właściwości elementów podrzędnych, 119
 dodawanie
 animacji, 84
 elementów WPF, 55
 identyfikatorów lokalizacji, 354
 inercji, 176
 komentarzy, 337
 nakładki, 239
 skrótów, 761
 właściwości zależnych, 718

dokument FlowDocument, 327
 dokumenty, 326
 przepływu, 326, 327, 336
 XPS, 327
 dołączanie zdarzeń, 159
 domyślna szerokość wierszy, 124
 dostawca
 danych, 402
 domyślnego klucza, 428
 właściwości dołączanych, 87
 dostęp do zasobów
 binarnych, 349, 351
 dynamicznych, 361
 logicznych, 359
 statycznych, 361
 w kodzie proceduralnym, 353
 w miejscu pochodzenia, 352
 wbudowanych, 352
 dostępna przestrzeń, 124
 dostosowanie
 elementów, 237
 przepływu danych, 409
 wyglądu pola wyboru, 270
 wyświetlania danych, 390
 zawartości miniatury, 240
 dotyk, 170
 dotyk wielokrotny, 170
 dowiązania poleceń, 185
 drzewo
 logiczne, 69, 70
 wizualne, 69, 71, 72, 74
 wizualnego szablonu, 444
 duch, 492, 517
 dyktowanie treści, 664
 dymek, 238
 dyrektywa
 clr-namespace, 49
 Uid, 354
 using System, 38
 dyskretna animacja keyframe, 632
 działanie
 JumpTask, 229
 listy szybkiego dostępu, 229
 OverlapPanel, 750
 Setter, 425
 Shape, 508
 XamlReader, 775

dziedziczenie wartości właściwości, 79, 81, 443
 dziennik, 208, 209
 dziennik NavigationWindow, 220
 dźwięk, 649

E

edytor Blend for Visual Studio, 15
 efekt, 530
 odbicia lustrzanego, 527
 odbicia na żywo, 530
 przezroczystości, 520
 ekran powitalny, 197
 elastyczność handlerów zdarzeń, 62
 element
 Binding, 399
 Block, 328
 BlockUIContainer, 329
 Button, 442, 445
 Calendar, 343
 Canvas, 110
 CheckBox, 253
 ComboBoxItem, 274
 ContentControl, 441
 ContextMenu, 308
 DataGrid, 279
 DatePicker, 345
 DockPanel, 117
 Expander, 261
 FanCanvas, 758
 Floater, 336
 FlowDocument, 328
 Frame, 259
 GeometryModel3D, 572
 Grid, 119
 GridSplitter, 126
 GroupBox, 260
 Image, 317
 InkCanvas, 319, 324
 Inline, 332
 JumpPath, 234
 JumpTask, 229
 Label, 256
 List, 329
 ListBox, 267, 275
 ListView, 277

element

- MediaTimeline, 657
- Menu, 306
- MyHwndHost, 678
- Paragraph, 328, 331, 332
- PasswordBox, 319, 324
- ProgressBar, 342, 466
- RadioButton, 254
- Ribbon, 285
- RibbonGroup, 294
- RibbonTab, 293
- RichTextBox, 319, 323
- Run, 321
- ScrollViewer, 135
- Section, 328
- Separator, 306
- Setter, 425, 429
- SharedSizeGroup, 144
- SimpleCanvas, 744
- Slider, 342
- StackPanel, 112
- StatusBar, 313
- Style, 423, 427, 460, 623
- TabControl, 278
- Table, 329, 331, 332
- TaskDialog, 245
- TextBlock, 320
- TextBox, 319, 322
- ToolBar, 312
- ToolTip, 256, 257, 303
- TreeView, 309
- UICulture, 354
- Viewbox, 137
- Visual, 533
- Window, 158
- WrapPanel, 114

elementy

- Binding, 418
- DirectX, 699
- Drawing, 541
- heterogeniczne, 424
- keyframe, 627
- kolekcji, 46
- Model3D, 541
- nadrzędne, 89

- nadrzędne szablonu, 437
- obiektywne, 35
- odrzucone, 236
- podrzędne, 264, 741
- potomne Block, 328, 331
- potomne Inline, 336
- potomne obiektów, 45
- usunięte, 236
- Windows Runtime, 669
- właściwości, 39

etap

- aranżowania, 742
- pomiarów, 740

- Expander, 261

F

- filtrowanie, 397
- Flash, 668
- FlowDocument, 327, 331, 332
- FlowDocumentReader, 340
- fokus, 680
- fokus klawiatury, 163
- format pikseli, 319
- formatowanie
 - elementów Run, 321
 - napisów, 380, 382
- formularz Windows Forms, 696
- Frame, 259
- funkcja
 - CreateWindow, 676, 685
 - DialogBox, 684
 - DialogFunction, 685
 - GetKeyState, 681
 - Initialize, 702
 - QuadraticEase, 641
 - TranslateAccelerator, 681
- funkcje
 - potęgowe, 634
 - pulpitu Windows, 225
 - ułatwiające, 634
 - BackEase, 635
 - BounceEase, 635
 - CircleEase, 635
 - ElasticEase, 636

ExponentialEase, 636
 SineEase, 636
 własne, 637
 WPF, 14, 23
 XAML, 34
 XAML2009, 58

G

gadżety, 214
 galerie, 304
 GDI, 22, 25
 generowanie kodu źródłowego, 56
 geometrie
 podstawowe, 479
 zagregowane, 483
 gesty, 185
 gradient, 528
 grafika
 clipart, 511
 dwuwymiarowa, 475
 trójwymiarowa, 537
 wektorowa, 532
 gramatyka, 664, 666
 Grid, 119, 129
 imitowanie Canvas, 129
 imitowanie DockPanel, 130
 imitowanie StackPanel, 130
 właściwości elementów podrzędnych, 130
 GridSplitter, 125–128
 GridView, 278
 GroupBox, 260
 grupa pobierania, 221
 grupowanie, 394, 395
 elementów, 395
 ulepszone, 397

H

handler AboutDialog_MouseRightButtonDown, 158
 handlers zdarzeń, 62, 78, 155, 169, 171
 hasło, 324
 hierarchia klas, 67
 hiperłącza, 207
 HWND, 708

I

identyfikator
 AppUserModelId, 768
 AppUserModelID, 761
 lokalizacji, 354
 iloczyn logiczny, 435
 Image, 317
 imitowanie
 Canvas, 129
 DockPanel, 130
 StackPanel, 130
 implementacja
 animacji, 607
 czytnika RSS, 418
 ducha, 495
 elementu FanCanvas, 755
 interfejsów API, 671
 SimpleCanvas, 744, 747
 SimpleStackPanel, 747
 właściwości zależnych, 76
 zdarzenia kierowanego, 154
 importer kontroltek ActiveX, 705
 inercja, 176
 inercja obrotu, 180
 InkCanvas, 324
 Inline, 332
 instalator Windows, 202
 instalowanie aplikacji, 187, 224
 pulpitu, 202
 XBAP, 221
 integracja Windows Forms z WPF, 697
 IntelliSense, 66
 interakcja
 z zasobami, 364
 ze schowkiem, 283
 interfejs
 aplikacji, 195
 ICollectionView, 392, 394
 ICollectionViewLiveShaping, 402
 ICommand, 181
 IDictionary, 60
 MDI, 195
 UI Automation, 737
 użytkownika, 337–340
 wstążki, 285

interfejsy API Windows Runtime, 760
 interoperacyjność, 708
 interoperacyjność COM, 673
 interpolacja, 563
 liniowa, 610, 627
 między kłatkami, 631
 spline, 630
 izolowany magazyn, 201

J

jednostka
 cm, 94
 in, 94
 pt, 94
 px, 94
 język
 BAML, 55
 C#, 148
 C++/CLI, 673, 683
 CAML, 56
 MSIL, 55
 SSML, 660
 XAML, 38
 JumpPath, 234
 JumpTask, 230, 232

K

kalendarz, 343
 kamera
 MatrixCamera, 552
 OrthographicCamera, 551
 PerspectiveCamera, 551
 przesuwanie, 547
 właściwość LookDirection, 545, 548, 550
 właściwość Position, 543
 właściwość UpDirection, 549
 kanał alfa, 528, 576
 Kaxaml, 33
 kąt rotacji, 560
 keyframe, 629
 klasa
 Application, 191, 194
 ApplicationCommands, 182
 BezierSegment, 495

Binding, 404, 412–415
 BitmapCache, 533, 535
 BitmapCacheBrush, 535
 BitmapEffect, 531
 Brush, 513
 Button, 252
 Camera, 542
 CollectionViewSource, 400
 CombinedGeometry, 486
 ComponentCommands, 182
 CompositeCollection, 416
 ContainerUIElement3D, 591
 ContentControl, 250
 ContentElement, 69
 Control, 69, 97, 736
 CountToBackgroundConverter, 387, 388
 D3DImage, 699, 700, 704
 DashStyle, 491
 DateTimeToDateConverter, 395
 DependencyObject, 68
 DesignerProperties, 738
 DiscreteXXXKeyFrame, 632
 DispatcherTimer, 606
 DispatcherObject, 68
 DoubleAnimationUsingKeyFrames, 627
 Drawing, 476, 492, 496
 DrawingBrush, 522
 DrawingContext, 494, 496
 DrawingImage, 478
 DrawingVisual, 493, 503
 EasingFunctionBase, 638
 EditingCommands, 182
 Effect, 530
 ElementHost, 695, 696
 Ellipse, 507
 EllipseGeometry, 478
 FlowDocument, 327
 FrameworkContentElement, 69
 FrameworkElement, 69, 411
 FrameworkPropertyMetadata, 720
 Freezable, 68
 Geometry, 479, 541
 Geometry3D, 541, 579
 GeometryConverter, 487
 GeometryDrawing, 481

GeometryGroup, 484, 486
 GeometryModel3D, 572
 GrammarBuilder, 666
 HwndHost, 674
 HwndSource, 683, 688
 ImageDrawing, 478
 ItemsControl, 263, 264, 273, 404
 JpgValidationRule, 412, 431
 JumpItem, 232
 JumpTask, 227, 232
 Line, 508
 LinearXXXKeyFrame, 630, 633
 ListCollectionView, 402
 ManipulationDelta, 174
 Material, 572
 MediaCommands, 182
 MediaElement, 652, 654
 MediaPlayer, 651
 MediaTimeline, 652
 Model3D, 562
 Model3DGroup, 586
 ModelUIElement3D, 590
 ModelVisual3D, 588
 MouseEventArgs, 165
 MultiBinding, 416
 NavigationCommands, 182
 Object, 68
 ObjectDataProvider, 407, 409
 Panel, 89
 Path, 510
 PathFigure, 482, 489, 495
 PathGeometry, 489
 PathSegment, 489
 PauseStoryboard, 656
 Pen, 490, 517
 Photo, 376
 Photos, 408
 PlayingCard, 725, 731
 Polygon, 510
 Polyline, 509
 Popup, 374
 PriorityBinding, 417
 ProgressPie, 472
 PromptBuilder, 660
 QueryContinueDragEventArgs, 167
 RangeBase, 341
 RectangleGeometry, 506
 RenderTargetBitmap, 533
 RepeatButton, 253
 ResumeStoryboard, 656
 RibbonCommands, 182
 RibbonContextualTabGroup, 302
 RibbonWindow, 288
 RoutedUICommand, 183
 Selector, 269
 ShaderEffect, 531
 Shape, 505, 506
 SimpleQuadraticEase, 638
 SortDescription, 400
 SoundPlayer, 650
 SoundPlayerAction, 650
 SpeechSynthesizer, 662
 StreamGeometry, 483
 StylusDevice, 168, 169
 System.Windows.Style, 422
 SystemColors, 364, 469
 SystemCommands, 182
 SystemFonts, 364, 469
 SystemParameters, 364, 469
 TemplateBindingExtension, 440
 Timeline, 627
 ToggleButton, 253
 ToolTip, 257
 ToolTipService, 258
 Touch, 173
 TouchEventArgs, 171
 TouchPoint, 171
 Transform, 541
 Transform3D, 541, 553
 Trigger, 430
 UIElement, 68, 99, 156, 541
 UIElement3D, 69, 541, 589
 UserControl, 714
 Vector3DAnimation, 548
 Viewport2DVisual3D, 592
 Viewport3D, 594
 Visual, 68, 493, 496, 541
 Visual3D, 68, 541, 588
 VisualBrush, 525, 536
 VisualStateGroup, 642

klasa

- VisualStateManager, 735
- WeakEventManager, 161
- Webcam, 670
- Window, 52, 188
- WindowInteropHelper, 689, 694, 702
- WindowsFormsHost, 693
- XamlMember, 774
- XamlProperty, 773
- XamlReader, 51, 769, 774, 779
- XamlServices, 780, 782
- XamlXmlReader, 771, 775–777
- XmlDataProvider, 403, 404, 407, 420
- XmlReader, 771

klasy

- bazowe, 68
- dyskretne keyframe, 632
- generyczne, 59
- implementujące animacje, 608
- interoperacyjności, 708
- odczytujące XAML, 769
- ułatwiające keyframe, 633
- WPF, 68

klawiatura, 162

klawisz Tab, 680

klawisze skrótu, 679, 682

klucz, 428

- słownikowy, 59
- szablonu, 406
- zasobu, 364

kod

- C#, 148, 151
- niezarządzany, 673
- proceduralny, 45, 50, 57, 362, 368
- skórki, 466
- zarządzany, 673

kolejność

- nawijania wierzchołków, 581
- przetwarzania, 36
- zdarzeń, 237

kolekcja

- Positions, 580
- Resources, 449
- SortDescriptions, 392
- TextureCoordinates, 585
- Transitions, 644

TriggerAction, 429

Triggers, 433

kolor

- elementu, 242
- odbity, 575, 579
- powiadomienia toastowego, 763

kolumny generowane automatycznie, 281

kombinacja martwego klawisza, 162

komentarze, 337–341

kompilowanie XAML, 53

kompozycja w pamięci podręcznej, 533

komunikacja między obiektami, 740

komunikat o błędzie, 412, 414, 678

konflikt wyzwalaczy, 435

konsolidacja

- handlerów zdarzeń, 161
- pędzli koloru, 357

konstruktory niedomyślne, 60

kontekst danych, 379

kontener, 591

- manipulacji, 176
- nawigacji, 204, 205
- prosty, 255
- z nagłówkami, 260

kontrawariancja delegatów, 161

kontrola poprawności, 84, 412–414, 431

kontrola poprawności grupy dowiązań, 415

kontrolka, *Patrz także* element

- ActiveX, 706
- ComboBox, 269
- elementów z nagłówkiem, 286
- FileInputBlock, 733
- FileInputBox, 714, 733
- MonthCalendar, 697
- PlayingCard, 723–732, 758
- PropertyGrid, 691
- ScrollBar, 135
- ToggleButton, 727
- Viewbox, 138
- WebBrowser, 218
- Webcam, 673
- Win32, 680
- Win32 Webcam, 670
- WPF Button, 249
- WPF Expander, 696
- WPF WebBrowser, 259

kontrolki

- ActiveX, 667, 704
- elementów, 263, 309, 315
- kalendarza, 343, 345
- niestandardowe, 711
 - definiowanie zachowania, 722
 - interfejs użytkownika, 728
 - zasoby, 724
- obrazu, 317
- piórka, 319
- szablony, 436
- tekstu, 319
- użytkownika, 711
 - definiowanie zachowania, 715
 - dodawanie właściwości zależnych, 718
 - dodawanie zdarzeń kierowanych, 721
 - zabezpieczanie, 717
- wbudowane, 250
- Win32, 667
- Windows Forms, 667
- WPF
 - części nazwane, 452–454
 - stany, 454–457
 - wstążki, 290, 291, *Patrz także* Ribbon
 - zakresu, 341
 - zawartości, 249, 250

kontrolowanie

- działania animacji, 611
- odtworzenia mediów, 656
- pozycji, 95
- rozmiaru, 90

konwersja PromptBuilder, 662

konwertery

- typów, 40, 48, 60, 105, 125
- użycie, 41
- wyszukiwanie, 42
- wartości, 386, 391, 445, 450

korzystanie z zasobów, 428

kostki przycisków, 597

krzywe Béziera, 480

kształtowanie na bieżąco, 392, 402

L

liczba funkcji, 219

licznik DispatcherTimer, 606

LINQ, Language Integrated Query, 402

List, 329

lista szybkiego dostępu, 227, 228

ListBox, 263–267, 275–277

ListView, 277

listy, 46

listy szybkiego dostępu, 225

lokalizator zasobów, 352

lokalizowanie zasobów binarnych, 354

luźne strony XAML, 223

Ł

ładowanie synchroniczne, 51

łańcuchowa postać geometrii, 487

łączenie

- DirectX z WPF, 699

- materiałów, 579

- obiektów Transform3D, 561

- transformacji, 106

M

mapowanie

- połysku, 579

- przestrzeni trójwymiarowej, 600

- trójwymiarowych punktów, 604

martwa strefa, 545

maski przezroczystości, 528

materiał

- AmbientMaterial, 576

- DiffuseMaterial, 573, 576

- EmissiveMaterial, 573, 577

- MaterialGroup, 573

- SpecularMaterial, 572, 578

mechanizm

- dynamicznych zasobów, 465

- PInvoke, 673

- rozszerzalności, 87

- VSM, 451

Menu, 306

menu aplikacji, 298

MenuItem

- właściwości, 307

- zdarzenia, 307

MessageBox, 245

metadane, 659

metoda

ArrangeOverride, 743, 754
 AttachToWindow, 670
 BeginAnimation, 612, 614
 BuildWindowCore, 676
 Cancel, 178
 CanExecute, 181
 ChangeVisualState, 736
 Clear, 392
 Color.FromValues, 514
 Complete, 178
 Convert, 396
 ConvertBack, 388
 CreateToastNotifier, 763
 DependencyProperty.Register, 720
 DestroyWindowCore, 676
 DrawGeometry, 496
 DrawVideo, 656
 EaseInCore, 639
 EnsureHandle, 702
 Execute, 181
 FindName, 52
 GetErrors, 415
 GetIntermediateTouchPoints, 171
 GetIsInDesignMode, 738
 GetPosition, 169
 GetTemplateChild, 733
 GetTouchPoint, 171
 GetValue, 720
 GetVisualChild, 497
 GlassHelper.ExtendGlassFrame, 243
 GoToState, 735, 737
 Guid.NewGuid, 61
 HitTestCore, 504
 InitializeComponent, 56
 IsKeyDown, 163
 Load, 780
 Main, 193
 Marshal.GetExceptionForHR, 61
 Measure, 741
 MeasureOverride, 741, 743, 754, 757
 Navigate, 206
 OnApplyTemplate, 733
 OnMnemonic, 682
 OnNoMoreTabStops, 681

OnRender, 656
 osłonowa zdarzenia, 155
 Parse, 66, 780
 PinToStart, 761
 Recognize, 663
 ReportBoundaryFeedback, 178
 Save, 50, 781
 SetBinding, 369, 370
 SetStackBuffer, 703
 SetValue, 720
 ShowDialog, 200
 Skip, 779
 SpeakSsml, 660
 SpeakSsmlAsync, 660
 StopLoading, 209
 ToXml, 662
 Transform, 781
 TransformToAncestor, 597, 600
 TransformToDescendant, 600
 TranslateAccelerator, 682
 viewSource_Filter, 401
 Visual.AddVisualChild, 498
 WndProc, 693

metody

fabryki, 61
 klasy DrawingContext, 494
 klasy XamlServices, 780
 nawigacji, 398
 statyczne, 369

mieszanie szablonów, 459

miniatura, 240

miniatura paska zadań, 241

model 3D, 539

modyfikowanie

nawigowania, 313
 widoku kolekcji, 391
 wyglądu kontrolki, 422
 zawartości XAML, 782

mowa, 659

rozpoznawanie, 663
 synteza, 660

mysz, 164

N

nadmiar treści, 132
 nakładka na element, 239
 narzędzie, *Patrz* program
 NavigationUIVisibility, 204
 NavigationWindow, 204
 nawias klamrowy, 43
 nawigacja, 203, 206, 210, 213, 398, 679
 domyślna, 399
 zintegrowana, 220
 nazywanie elementów XAML, 52
 niezgodne typów danych, 387
 normalizowanie danych wejściowych, 779
 normalna, 583
 null, 396

O

obcinanie elementów podrzędnych, 133
 obiekt

 Application, 196
 Binding, 368
 BindingExpression, 411
 Button, 52
 D3DImage, 702
 DateTime, 396
 DrawingContext, 656
 DrawingVisual, 493
 EmissiveMaterial, 577
 ExceptionValidationRule, 413
 Figure, 334
 Frame, 208
 GeometryDrawing, 481
 GeometryGroup, 485
 Guid, 61
 HWND, 685, 686
 HwndSource, 686, 688
 Image, 176
 JumpItemsRejectedEventArgs, 237
 JumpList, 227
 MatrixTransform3D, 561
 MediaElement, 654, 655, 658
 mesh, 586
 Model3DGroup, 586
 NavigationFailedEventArgs, 259

NavigationService, 205
 OneTime, 409
 OneWay, 409
 OneWayToSource, 409
 Page, 205
 PathFigure, 482
 PathGeometry, 488
 PropertyChanged, 411
 ScheduledToastNotification, 767
 SpecularMaterial, 578
 Storyboard, 626
 StreamGeometry, 483
 Trigger, 78, 79
 Visual, 496, 498
 VisualTransition, 642
 Webcam, 671
 Window, 51
 WindowsFormsHost, 693
 XamlObjectWriter, 60
 obiekty
 docelowe, 622
 keyframe, 629, 630
 Light, 563
 nadrzędne, 89
 RoutedUICommand, 183
 Storyboard, 619
 Transform3D, 561
 VisualState, 639
 obracanie, 174, 176
 obrót, 176
 obsługa
 błędów, 259, 413, 415
 dotyku, 170
 dźwięku, 649
 fokusu, 163
 klawiszy skrótu, 679, 682
 komentarzy, 340
 nadmiaru treści, 132
 nawigacji, 679
 poleceń, 181
 tematów, 471
 typów generycznych, 59
 wideo, 654
 wierszy tekstu, 323
 wyjątków, 414
 zdarzeń, 171

- odczytywanie
 - listingu, 775
 - tekstu, 660
 - XAML, 769, 773
- odtwarzacz wideo, 657
- odtworzenie mediów, 656
- odwrotna rotacja, 558
- ograniczenia użycia stylów, 427
- ograniczenie docelowego typu, 440
- okna dialogowe, 198
 - własne, 199
 - wspólne, 198
- okno
 - MessageBox, 246
 - modalne Win32, 683
 - modalne Windows Forms, 694, 695
 - modalne WPF, 690, 698
 - TaskDialog, 246
 - WPF, 674
- określanie
 - gramatyki, 664
 - obiektu docelowego, 622
 - właściwości docelowej, 620
- OpenGL, 22
- osadzanie
 - kontrolerek, 667
 - ActiveX, 704, 707
 - Win32, 670, 677, 680
 - Windows Forms, 690
 - WPF, 683, 695
 - kontrolki PropertyGrid, 691, 693
- oświetlenie, 563

P

- pakiet
 - SDK, 15
 - WPF Toolkit, 26
- pamięć podręczna, 217
- panel, 89
 - FanCanvas, 755, 757
 - OverlapPanel, 750
 - SimpleCanvas, 744
 - SimpleStackPanel, 747
- panele
 - elementów, 266
 - nadrzędne, 740
 - niestandardowe, 739
 - niezadokowane, 142
 - o zmiennej wielkości, 140
 - proste, 131
 - zadokowane, 140, 143
 - zwijane, 140
- Paragraph, 328
- parametr
 - overlap, 757
 - Size, 741
- parser
 - WPF XAML, 48
 - XAML, 50
 - XAML2009, 48
- pasek
 - postępu, 238, 342
 - przewijania, 136
 - przycisków, 147
 - szybkiego dostępu, 299, 300
 - zadań, 237, 238
- PasswordBox, 324, 451
- Path, 404
- pełne zaufanie, 220
- pędzel
 - DrawingBrush, 521
 - ImageBrush, 524
 - LinearGradientBrush, 515
 - RadialGradientBrush, 518
 - SolidColorBrush, 513, 574
 - VisualBrush, 525, 527
- pędzle
 - gradientowe, 518
 - kolorowe, 356, 513
 - pokrywające, 521
- pętla
 - komunikatów, 192
 - węzłowa, 772
 - węzłowa niestandardowa, 778
- piksele logiczne, 94
- piórko, 167
- pismo odręczne, 324
- plik
 - .manifest, 217
 - AxMSTSLib.dll, 705
 - imageres.dll, 230

- MSTSCLib.dll, 705
- shell32.dll, 230
- System.Windows.Forms.dll, 705
- WindowsFormsIntegration.dll, 705
- pliki
 - .dll, 56, 231
 - .exe, 231
 - .g.cs, 56
 - .g.vb, 56
 - .resources, 355
 - .xaml, 32, 54, 223, 350
 - .xbap, 217
 - BAML, 55
 - kodu ukrytego, 54
- pobieranie
 - na żądanie, 222
 - plików, 221
- podklasy klasy
 - Drawing, 476
 - Effect, 531
 - Model3D, 562
 - Transform3D, 553
- podpowiedzi ekranowe, 302
- podział XAML, 360
- pole
 - AccelerationRatio, 617
 - AutoReverse, 615
 - BeginTime, 614, 615
 - DecelerationRatio, 617
 - Duration, 615
 - EasingFunction, 617
 - FillBehavior, 618
 - From, 613
 - IsAdditive, 617
 - IsCumulative, 617
 - RepeatBehavior, 615
 - SpeedRatio, 615
 - To, 612
 - wyboru, 270
- polecenia, 181
 - łańcuchowe geometrii, 489
 - wbudowane, 181
- polecenie Help, 185
- pomiar, 740
- pompa komunikatów, 192
- potrójne przecinki, 353
- powiadamianie o zmianach, 78
- powiadomienia
 - planowane, 767
 - tostowe, 759, 762, 765
- powłoka Windows, 237
- pozycja elementów, 95
- prawoskrętność, 543
- prefiks
 - mc, 777
 - x, 60
- profile przestrzeni barw, 514
- program
 - Blend, 462
 - FxCop, 782
 - Kaxaml, 418
 - LocBaml, 355
- ProgressBar, 342, 447, 451, 466
- projekt dla wielu języków, 354
- przebieg nawigacji, 213
- przechwytywanie
 - klatek, 656
 - pisma odręcznego, 324
 - właściwości, 446
 - wskaźnika myszy, 167
- przeciągnij i upuść, 166
- przeglądarka XAMLPAID2009, 37
- przejścia, 642, 680
- przejścia VisualTransition, 645
- przekształcenia współrzędnych, 597
- przekształcenie
 - RotateTransform3D, 558
 - RotationTransform3D, 558
 - ScaleTransform3D, 556
 - TranslateTransform3D, 555
- przepływ danych, 409
- przestrzenie nazw, 36
 - .NET, 37
 - niejawne, 37
 - WPF XML, 38
 - XAML, 63
 - XML, 38
- przestrzeń
 - barw
 - scRGB, 513
 - sRGB, 513

przestrzeń
 nazw
 System.Windows.Shell, 248
 świata, 542
 przesunięcie, 174, 176, 557, 558
 przesuwanie kamery, 547
 przetwarzanie
 sekwencji {}, 382
 właściwości i zdarzeń, 36
 XAML, 50
 przewijanie, 268
 przewijanie zawartości, 135
 przezroczyste materiały, 576
 przezroczystość, 520
 przybornik XAML Toolkit, 782
 przycisk
 anulowania, 252
 domyślny, 252
 WPF, 35
 przyciski, 251, 445
 miniatury, 241
 w stylu kostek, 595, 596
 przypisanie do właściwości, 379
 przywracanie stanu aplikacji, 201
 pulpit Windows, 225

R

reakcja na błąd, 431
 reguła prawej dłoni, 543, 581
 reguły kontroli poprawności, 412, 414
 rendering, 499
 renderowanie, 497
 Ribbon, 285
 dostosowywanie zmian wielkości, 293–295
 galerie, 304
 kontrolki wstążki, 289
 menu aplikacji, 298
 pasek szybkiego dostępu, 299
 podpowiedzi ekranowe, 302
 skrótów klawiszowe, 296
 zakładki kontekstowe, 301
 zasady, 286
 zmiana rozmiaru, 291
 RibbonGallery, 305
 RibbonGalleryItem, 304

RibbonGroup, 287, 292, 294
 RibbonSplitButton, 304
 RibbonTab, 293
 RibbonToolTip, 303
 RichTextBox, 323
 rodzaje kamer, 551
 rotacja, 559
 RotateTransform3D, 558
 rozmiar
 elementów, 90
 kolumny, 123
 RibbonGroup, 294
 RibbonTab, 293
 wiersza, 123
 rozpoznawanie mowy, 663
 rozproszenie światła, 573
 rozszerzalne części XAML, 49
 rozszerzanie akapitu, 332
 rozszerzenia znaczników, 42, 45, 65
 RSS, Really Simple Syndication, 406
 Run
 elementy jawne, 322
 elementy niejawne, 322
 rysowanie
 domu, 538
 tekstu, 319, 320
 w GDI, 475
 w WPF, 475
 rzutowanie
 perspektywiczne, 552
 prostokątne, 552

S

SAPI SDK, 660
 ScaleTransform3D, 556
 schematy kolorów, 473
 ScrollViewer, 135, 268
 SDK, Software Development Kit, 15
 Section, 328
 sekwencja {}, 382
 SelectiveScrollingGrid, 132
 Selector, 310
 selektory, 269
 selektory szablonów, 386
 siatka, 582, 586

- Silverlight, 33, 221, 259, 668
- skalowanie, 137, 138, 174, 176, 557
 - mapy bitowej, 318
 - poza osiami, 558
- składnia gwiazdki, 124
- skórka, 421, 465, 462
 - electric, 466
 - Light and Fluffy, 466
- Slider, 342
- słowa kluczowe XAML, 63–65
- słownik Application.Resources, 464
- słowniki, 47, 729
 - standardowe, 470
 - tematów, 471
 - XAML, 33
- słowo kluczowe
 - Code, 57
 - Name, 52
 - partial, 54
 - x:Arguments, 60
 - x:FactoryMethod, 61
- sortowanie, 392
- sortowanie grup, 396
- specyfikacja XAML, 34
- sprawdzanie pisowni, 323
- SRGS, 664
- StackPanel, 95–112
 - elementy Button, 113
 - właściwości elementów podrzędnych, 113
- stany
 - aplikacji, 201
 - kontrolki, 454–457, 734
- StatusBar, 313
- sterowanie
 - położeniem elementu, 335
 - pozycją, 89
 - przewijaniem, 268
 - rozmiarem, 89
 - wyglądem szablonu, 460
 - wyświetlaniem, 380
- stosowanie, *Patrz* użycie
- strategie kierowania
 - Bubbling, 155
 - Direct, 155
 - Tunneling, 155
- strona
 - startowa, 120–122
 - wysyłanie danych, 211
 - zwracanie danych, 213
 - XAML, 223
- strony trójkąta, 581
- struktura
 - aplikacji, 187
 - Color, 513, 514
 - GridLength, 125
 - HwndSourceParameters, 685
- strumieniowanie, 658
- strumień węzłów XAML, 775–777
- style, 421, 422, 459–461
 - nazwane, 427
 - niejawne, 427
 - tematu, 469, 473
 - typowane, 427
 - WPF Themes, 474
- suma logiczna, 434
- sygnatura TaskDialog, 245
- synteza mowy, 660
- systemowe typy danych, 60
- szablon, 421, 440, 445
 - GroupStyle, 394
 - kołowy, 447
- szablony
 - danych, 383, 385, 391
 - dla tematu, 468, 469
 - kontrollek, 436, 451, 459, 461
 - powiadomień tostowych, 764
 - tostów, 763

Ś

- ścieżki właściwości, 265, 399
- środek
 - rotacji, 560
 - skali, 557
- światło, 563
 - AmbientLight, 567, 571
 - DirectionalLight, 563
 - PointLight, 564
 - SpotLight, 565

T

TabControl, 278
 Table, 329
 tablica elementów, 504
 TabPanel, 131
 TaskDialog, 245, 247
 technologie interfejsów użytkownika, 668
 temat, 421, 468

- Aero, 470
- AeroLite, 470
- Klasyczny Windows, 470
- Zune, 470

 tematy Windows, 473
 TemplateBinding, 443, 445
 testowanie trafień, 493, 499–504, 512
 testy trafień

- trójwymiarowe, 593
- wizualne, 493, 499

 TextBlock, 320
 TextBox, 322, 451
 TextElement, 328
 TextSearch, 276
 Toolbar, 311
 ToolbarOverflowPanel, 131
 ToolbarPanel, 131
 ToolbarTray, 131
 ToolTip, 258
 transformacja, 98

- 3D, 99
- MatrixTransform, 105
- RotateTransform, 100
- ScaleTransform, 102, 134, 137, 439
- SkewTransform, 104
- TranslateTransform, 104

 TranslateTransform3D, 555
 TreeView, 309–311
 TreeViewItem, 311
 Trigger, 430
 tryb

- Aero Glass, 242
- EaseIn, 639
- EaseInOut, 639
- Mixed, 686
- natychmiastowy, 475
- opóźniony, 475

tworzenie

aplikacji, 196
 aplikacji XBAP, 216
 dokumentów przepływu, 327
 grafiki dwuwymiarowej, 476
 interfejsu użytkownika, 337, 713, 728
 kontrolki niestandardowej, 722
 kontrolki użytkownika, 713
 niejawnych stylów, 427
 obiektów

- metody fabryk, 61
- niedomyślne konstruktory, 60

 okien dialogowych, 198
 panelu, 140

- FanCanvas, 755
- OverlapPanel, 750
- SimpleCanvas, 744
- SimpleStackPanel, 747

 podzespołu satelickiego, 355
 przezroczystego materiału, 576
 reguł kontroli poprawności, 412, 413
 strony startowej, 120, 121
 szablonu, 461
 układu, 89

typ

Key, 163
 MatrixTransform3D, 561
 wyliczeniowy

- BaseValueSource, 82
- BindingMode, 409
- ClickMode, 251
- DragAction, 167
- DragDropEffects, 166
- Gender, 280
- JumpItemRejectionReason, 237
- MouseButton, 165
- PixelFormat, 319
- PresentationTraceLevel, 390
- ReasonSessionEnding, 194
- RoutingStrategy, 155
- ScrollBarVisibility, 135
- ShutdownMode, 194
- Stretch, 137
- StretchDirection, 138
- TaskbarItemProgressState, 238

- TouchAction, 171
- UpdateSourceTrigger, 410
- Visibility, 95, 389

typy

- danych
 - .NET, 608
 - wbudowane, 60
 - WPF, 608
- generyczne, 59
- wyzwalaczy, 430

U

- układ, 89
- układ współrzędnych
 - dwuwymiarowy, 542
 - prawoskrętny, 543
 - trójwymiarowy, 542
- układy z panelami, 109
- ukrycie przycisku, 467
- ulepszanie wyświetlania danych, 376
- UniformGrid, 131
- upraszczanie interfejsu użytkownika, 467
- URL, 352
- uruchamianie poleceń gestami, 185
- usuwanie
 - dowiązania, 369
 - zawijania, 139
- uszkodzenie interfejsu użytkownika, 468
- użycie
 - Aero Glass, 243
 - animacji, 609
 - Application.Run, 192
 - ClickOnce, 222
 - ControlTemplate, 443
 - DirectX, 704
 - dziennika, 208, 209
 - geometrii, 479
 - hiperłączy, 207
 - JumpPath, 235
 - kontrolki Webcam, 673
 - konwerterów typów, 41
 - konwerterów wartości, 386
 - listy szybkiego dostępu, 227
 - łańcuchów XML, 765
 - obiektu Storyboard, 626

- paska postępu, 238
- rozłączonych źródeł, 416
- skórek, 462
- stylów, 427
- szablonu, 445
- szablonu danych, 383
- transformacji, 98
- ThemeInfoAttribute, 471
- wiązania, 368, 370
- właściwości .NET, 372
- właściwości Content, 440
- właściwości CustomCategory, 232
- właściwości istniejących, 446
- XmlDataProvider, 406
- zasobów, 362
- zasad kontroli, 412
- zdarzeń manipulacji, 174

V

- Vector3D, 548
- Viewbox, 137, 138
 - StretchDirection, 138
- Visual, 533
- Visual State Manager, 639
- Visual Studio 2012, 15
- VSM, Visual State Manager, 451

W

- wartościowanie, 83
- wartość NaN, 91
- wersja systemu operacyjnego, 248
- wiązanie
 - danych, 367, 378, 380, 386, 402, 420, 629
 - do właściwości .NET, 371
 - hierarchiczne, 405
 - kompletnego obiektu, 373
 - niezgodnych typów danych, 387
 - obiektów, 367
 - surowe, 375
 - TwoWay, 409, 410
 - właściwości .NET, 373
 - z kolekcją, 375
 - z metodą, 408
- wideo, 654

- widok kolekcji, 391
- widoki dodatkowe, 399
- widoki domu, 540
- wielkości procentowe, 124
- wielokrotne użycie
 - animacji, 611
 - szablonu, 443
- wielokrotny wybór, 276
- wiersz poleceń, 193
- Win32, 667
- Windows Forms, 667
- wirtualizacja, 283, 311
- wirtualizacja paneli, 114
- wizualne testy trafień, 493, 499
- własne
 - funkcje ułatwiające, 637
 - kategorie, 232, 233
 - komponenty, 219
 - sortowanie, 393
- własny pędzel koloru, 356
- właściwości, 62
 - .NET, 371
 - BitmapCache, 534
 - docelowe, 368, 620
 - dołączane, 84, 87, 273
 - efektów, 531
 - elementu nadrzędnego szablonu, 440
 - ListBox, 276
 - ManipulationDelta, 174
 - MenuItem, 307
 - NearPlaneDistance, 545
 - MeshGeometry3D, 580
 - osłonowe, 77
 - PathFigure, 489
 - PathGeometry, 489
 - PathSegment, 489
 - Pen, 490
 - ScrollView, 135
 - StylusDevice, 168
 - typu DataTemplate, 384
 - UpdateSourceTrigger, 410
 - Viewbox, 138
 - zależne, 75, 411
 - źródłowe, 368

- właściwość
 - Action, 167
 - ActiveEditingMode, 326
 - AddedItems, 273
 - AlternationCount, 264
 - AlternationIndex, 264
 - Arguments, 232
 - Background, 243
 - BeginTime, 627
 - BitmapEffect, 531
 - BlackoutDates, 345
 - CacheMode, 535, 593
 - Camera.Transform, 550
 - CancelButtonStyle, 466
 - CanContentScroll, 268
 - CenterX, 101
 - ClipToBounds, 133, 134, 594
 - ColorInterpolationMode, 516
 - Content, 186, 206, 250, 260, 440
 - ControlSizeDefinition, 295
 - CurrentItem, 398
 - CustomCategory, 232
 - DashStyle, 491
 - DateTime, 394, 399
 - DefaultDrawingAttributes, 325
 - Delay, 343, 411
 - DisplayDate, 344
 - DisplayMemberPath, 264, 265, 404
 - Dock, 117
 - Duration, 612
 - Effect, 530
 - EscapePressed, 167
 - Expansion, 174
 - Extended, 275
 - Fill, 439
 - FillRule, 483
 - Flat, 490
 - FlowDirection, 97, 113, 116
 - GradientOrigin, 520
 - GroupDescriptions, 394
 - GroupStyle.Default, 395
 - Handled, 156, 158
 - HasItems, 264
 - HasLineInfo, 774
 - Header, 313

HeaderTemplate, 394
Height, 90
HorizontalAnchor, 334
ICollectionView, 400
Icon, 307
InkCanvas, 425
Inlines, 321
InputGestureText, 307
Interval, 343
IsAsynchronous, 407
IsCheckable, 307
IsChecked, 723
IsDefault, 252
IsDefaulted, 252
IsDown, 162
IsEditable, 270, 272
IsGrouping, 264
IsIndeterminate, 342
IsMainMenu, 306
IsMinimized, 287
IsReadOnly, 270
IsRepeat, 162
IsSelected, 269
IsSelectionActive, 269
IsSharedSizeScope, 129
IsTextSearchCaseSensitive, 273
IsTodayHighlighted, 344
IsToggled, 162
IsUp, 162
ItemHeight, 115
Items, 377
ItemsPanel, 264, 266
ItemsSource, 377
ItemTemplate, 383
ItemWidth, 115
JournalOwnership, 208
JumpItems, 236
KeyboardDevice, 162
KeySpline, 631
KeyStates, 162, 167
KeyTime, 629
LayoutTransform, 98
LookDirection, 545, 548, 550
Margin, 92
Multiple, 275
Name, 52
NavigationUIVisibility, 208
NodeType, 774
Normals, 583, 584
OpacityMask, 529
Orientation, 113, 342
OriginalSource, 156
Overlay, 239
Padding, 92
Path, 235, 370
PenLineCap, 490
Position, 543
QuickAccessToolBar, 299
RederAtScale, 535
RelativeSource, 371
RenderTransform, 98
RenderTransformOrigin, 101
RepeatBehavior, 626
Rotation, 174
RoutedEvent, 156
Scale, 174
SelectedDates, 344
SelectedIndex, 269
SelectedItem, 269
SelectedValue, 269
Shape.Stroke, 506
ShowGridLines, 123
ShowOnDisabled, 258
Single, 275
SortDescriptions, 392
Source, 156, 206, 317
SpeedRatio, 627
SpreadMethod, 517, 519
Square, 490
Stretch, 137, 522
StringFormat, 381
TabletDevice, 168
TargetProperty, 620
TemplatedParent, 437
Text, 183
TextFormattingMode, 319
TextHintingMode, 320
TextRenderingMode, 320
TextWrapping, 323
Thickness, 94

właściwość

- TickPlacement, 343
- TimeSpan, 612
- ToolTip, 302
- TouchDevice, 171
- Transitions, 642
- Translation, 174
- TriangleIndices, 582
- UpDirection, 549, 550
- ValidatesOnDataErrors, 414
- ValidatesOnExceptions, 414
- VerticalAnchor, 335
- View, 277
- Viewbox, 523, 524
- Visibility, 95, 302
- Width, 90
- XPath, 403
- zawartości, 45

WPF, Windows Presentation Foundation, 13, 21, 67

WPF 3.5, 27

WPF 3.5 SP1, 27

WPF 4, 28

WPF 4.5, 29

WPF Visualizer, 74

wprowadzanie hasła, 324

WrapPanel, 114, 139

- FlowDirection, 116
- ItemHeight, 115
- ItemWidth, 115
- Orientation, 114
- przyciski, 115
- właściwości elementów podrzędnych, 116

wskaźnik myszy, 167

współdzielenie, 362

- elementu Style, 425
- rozmiarów, 127
- stylów, 424
- źródła, 379

współrzędne tekstur, 585

wstążka, *Patrz* Ribbon

wyczyszczenie obiektu Binding, 369

wydajność, 66, 506

wydajność renderowania, 532

wygląd kontrolki Windows Forms, 694

wyjątek InvalidOperationException, 359

wyłączanie

- działania klamry, 44
- konwersji typów, 60

wymiarowanie interaktywne, 125

wymiary

- bezwzględne, 123
- proporcjonalne, 123

wymuszanie, 84

wyrażenie logiczne, 434

wyrównywanie, 96

- Stretch, 96, 103
- zawartości, 97

wyspa danych XML, 403

wysyłanie

- danych, 211
- powiadomienia tostowego, 762

wyszukiwanie

- konwerterów typów, 42
- zasobów, 358

wyświetlanie, 380

- danych, 376, 390
- dokumentów przepływu, 336
- obiektu, 496

wywołanie InitializeComponent, 190

wyznaczenie wartości bazowej, 82

wyzwalacz

- danych, 79, 430, 433
- dla Button.IsMouseOver, 438
- dla IsMouseOver, 439
- EventTrigger, 619, 623
- właściwości, 78, 430, 625
- zdarzenia, 79, 430

wyzwalacze, 429, 446

- iloczyn logiczny, 435
- suma logiczna, 434

wzorzec

- dekorator, 137
- słabego zdarzenia, 161

X

XAML, 14, 31, 33, 693

XAML Cruncher, 33

XAML Silverlight, 782

XAML2009, 58

XAML2009, 33, 37

XamlPadX, 33
 XBAP, XAML Browser Applications, 216
 XmlDataProvider, 403
 XPath, 404
 XPS, XML Paper Specification, 327, 352

Z

zabezpieczanie kontrolek użytkownika, 717
 zachowanie stanów wizualnych, 446, 451
 zakładki kontekstowe, 301
 zakres
 Bold, 332
 Italic, 332
 Underline, 332
 zakres elementu Style, 427
 zamiana
 kolejności przekształceń, 557
 słów na tekst, 663
 zamrażanie kolumn, 284
 zapisywanie
 do aktywnych obiektów, 778
 do XML, 779
 stanu aplikacji, 201
 XAML, 769
 zarządzanie
 obiektami VisualState, 643
 zaznaczonym elementem, 378
 zasady
 kontroli, 412
 binarne, 347, 349
 dynamiczne, 359, 463
 logiczne, 355
 niepodlegające współdzieleniu, 362
 osadzone, 658
 statyczne, 359
 systemowe, 364
 XAML, 355
 zasób
 dynamiczny, 465
 kompilowany, 466
 zastosowania klasy Application, 194
 zawartość DirectX, 700
 zawijanie, 139
 zaznaczanie wierszy, 282
 zbiór słowników tematów, 471

zdarzenia
 .NET, 155
 bąbelkujące, 156
 dołączane, 159, 160
 dotknięcia, 170
 kierowane, 153–155, 161, 721
 kierowane w akcji, 156
 klawiatury, 162
 manipulacji, 174
 MenuItem, 307
 myszy, 164, 165
 nawigacji, 210
 piórka, 167, 169
 powiadomień, 766
 tunelowane, 156
 wejściowe, 186
 zdarzenie
 CanExecuteChanged, 181
 Click, 154, 251
 ErrorsChanged, 415
 FocusableChanged, 163
 FrameReported, 173
 IsMouseDownDirectlyOverChanged, 164
 KeyDown, 155, 156, 162
 Loaded, 691
 ManipulationBoundaryFeedback, 178
 ManipulationCompleted, 174, 176
 ManipulationDelta, 174, 180
 ManipulationInertia, 176
 ManipulationStarting, 174, 180
 MediaFailed, 659
 MouseDownDoubleClick, 165
 MouseDownLeftButtonDown, 158
 MouseMove, 156
 MouseDownRightButtonDown, 156, 158
 NavigationFailed, 259
 PreviewMouseDownDoubleClick, 165
 Rendering, 607
 SelectionChanged, 160, 273
 SpeechRecognized, 663
 ToastNotification, 767
 TouchDown, 174
 ValueChanged, 341
 zestaw funkcji, 218
 z-fighting, 546

zgodność znaczników, 777

zmiana

formatu pikseli, 318

koloru kontenera, 432

zmienna wielkość kontrolki, 688

zmniejszanie

elementu, 134

rozdzielczości bitmapy, 535

zwracanie danych, 213

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ 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 WYDAWNICZA

 **Helion SA**

Doskonały przewodnik dla programistów WPF!

Windows Presentation Foundation (WPF) to nowoczesna technologia, pozwalająca tworzyć zaawansowane aplikacje dla systemu Windows. Dzięki jej możliwościom sprawnie i bezproblemowo poradzisz sobie z każdym zadaniem — niezależnie od tego, czy chcesz stworzyć tradycyjną aplikację albo niesamowity interfejs 3D, czy połączyć animacje i multimedia. Co więcej, WPF sprawdza się świetnie zarówno na tradycyjnych komputerach, jak i na urządzeniach mobilnych oraz dużych ekranach telewizorów. Brzmi zachęcająco? Przekonaj się sam!

Jeśli sięgniesz po tę książkę, zdobędziesz kompletne źródło informacji na temat Windows Presentation Foundation. Na samym początku zapoznasz się z historią WPF, a następnie przejdziesz do odkrywania tajników XAML oraz podstaw tworzenia oprogramowania. Kolejne rozdziały zawierają bezcenną wiedzę na temat układów okien, zdarzeń, struktury aplikacji, wykorzystania pulpitu systemu Windows oraz używania kontrolki. Ponadto dowiesz się stąd, jak wiązać dane, korzystać ze stylów, skórek, z szablonów i tematów oraz pracować z materiałami multimedialnymi. A potem zdobędziesz zaawansowaną wiedzę na temat grafiki 3D, animacji oraz współpracy z technologiami innymi niż WPF. Książka ta jest wspaniałym kompendium wiedzy na temat WPF w wersji 4.5.

Dzięki tej książce:

- poznasz platformę WPF w wersji 4.5
- wykorzystasz jej potencjał przy tworzeniu aplikacji dla systemu Windows
- przekonasz się, jak pracować z materiałami multimedialnymi
- zobaczysz pola, na których możliwa jest współpraca z innymi technologiami
- opanujesz potencjał WPF

Adam Nathan — guru WPF, główny architekt oprogramowania w Startup Business Group firmy Microsoft. Wcześniej główny programista i architekt w serwisie Popfly (pierwszy projekt firmy Microsoft oparty na Silverlight). Jego książki zdobyły ogromną popularność i są powszechnie uznawane za lekturę obowiązkową.

Helion

35064 numer katalogowy
księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Sprawdź najnowsze promocje:
• <http://helion.pl/promocje>
Książki najchętniej czytane:
• <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
• <http://helion.pl/nowosci>

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

ISBN 978-83-283-0141-2



cena: 129,00 zł

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ



KOD KORZYŚCI

SAMS