



Tworzenie nowoczesnych aplikacji graficznych w WPF

Dobry interfejs graficzny aplikacji Windows? Tylko z WPF!

- Poznaj metody tworzenia nowoczesnych i spójnych GUI
- Naucz się korzystać z możliwości WPF i języka XAML
- Dowiedz się, jak łączyć atrakcyjne interfejsy z logiką programów

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek 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.

Redaktor prowadzący: Michał Mrowiec

Projekt okładki: Jan Paluch

Fotografia na okładce została wykorzystana za zgodą Shutterstock.com

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)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie?twnoap>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Kody źródłowe wybranych przykładów dostępne są pod adresem:

<ftp://ftp.helion.pl/przyklady/twnoap.zip>

ISBN: 978-83-246-3500-9

Copyright © Helion 2012

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

| | |
|---|-----------|
| Podziękowania | 11 |
| Od autora | 13 |
| Wstęp | 15 |
| Wymagane oprogramowanie | 15 |
| Kody źródłowe | 16 |
| Użyte oprogramowanie | 16 |
| Organizacja podręcznika | 16 |
| Rozdział 1. Wprowadzenie do WPF | 19 |
| 1.1. Rozwój interfejsu użytkownika | 19 |
| 1.2. Cechy charakteryzujące WPF | 20 |
| 1.3. Wersje WPF | 22 |
| 1.3.1. WPF 3.5 | 23 |
| 1.3.2. WPF 4 | 23 |
| Rozdział 2. XAML | 25 |
| 2.1. Wiadomości podstawowe | 26 |
| 2.2. XAML a XML | 28 |
| 2.2.1. Znaki specjalne XAML | 29 |
| 2.3. Tagi i atrybuty a klasy, właściwości i zdarzenia | 30 |
| 2.4. Przestrzenie nazw | 31 |
| 2.4.1. Używanie typów z innych przestrzeni nazw | 34 |
| 2.5. Sposoby określania wartości właściwości | 35 |
| 2.6. Konwertery typów | 37 |
| 2.7. Rozszerzenia znaczników | 37 |
| 2.8. Elementy zagnieżdżone | 41 |
| 2.8.1. Element typu content | 42 |
| 2.8.2. Kolekcje | 43 |
| 2.8.3. Wartość, która może być przekonwertowana na obiekt | 45 |
| 2.9. XAML i kod proceduralny | 45 |
| 2.9.1. Kod proceduralny i nieskompilowany XAML | 46 |
| 2.9.2. Kod proceduralny i skompilowany XAML | 46 |
| 2.10. XAML 2009 | 49 |
| 2.11. Słowa kluczowe XAML | 49 |
| Pytania testowe | 52 |
| Odpowiedzi na pytania | 55 |

| | |
|---|-----------|
| Rozdział 3. Podstawy WPF | 57 |
| 3.1. Architektura systemu | 57 |
| 3.2. Hierarchia klas | 59 |
| 3.3. Drzewa logiczne i drzewa prezentacji | 61 |
| 3.3.1. Drzewa logiczne | 61 |
| 3.3.2. Drzewa prezentacji | 64 |
| 3.4. Właściwości zależne | 68 |
| 3.4.1. Implementacja właściwości zależnych | 69 |
| 3.4.2. Powiadamianie o zmianie, wyzwalacze właściwości | 72 |
| 3.4.3. Dziedziczenie wartości właściwości | 74 |
| 3.4.4. Ostateczna wartość właściwości | 77 |
| 3.4.5. Właściwości dołączone | 78 |
| Pytania testowe | 80 |
| Odpowiedzi do pytań | 81 |
| Rozdział 4. Zarządzanie układem graficznym wewnątrz paneli | 83 |
| 4.1. Czym jest układ elementów | 84 |
| 4.2. Sterowanie rozmiarem | 85 |
| 4.2.1. Height i Width | 85 |
| 4.2.2. Margin i Padding | 86 |
| 4.2.3. Visibility | 88 |
| 4.3. Sterowanie pozycją | 89 |
| 4.3.1. HorizontalAlignment i VerticalAlignment | 89 |
| 4.3.2. HorizontalContentAlignment i VerticalContentAlignment | 90 |
| 4.3.3. FlowDirection | 90 |
| 4.3.4. Transformacje | 91 |
| Pytania testowe | 96 |
| Odpowiedzi do pytań | 98 |
| Rozdział 5. Panele | 99 |
| 5.1. Przegląd i klasyfikacja paneli | 99 |
| 5.2. Panele interfejsu użytkownika | 101 |
| 5.2.1. Canvas | 102 |
| 5.2.2. StackPanel | 104 |
| 5.2.3. WrapPanel | 105 |
| 5.2.4. DockPanel | 107 |
| 5.2.5. Grid | 109 |
| 5.3. Tworzenie własnych paneli | 118 |
| 5.3.1. Implementacja pomiarów | 119 |
| 5.3.2. Implementacja aranżacji | 120 |
| 5.3.3. Tworzenie bezużytecznego i użytecznego panelu | 121 |
| 5.3.4. Tworzenie panelu kołowego | 125 |
| 5.4. Pozostałe panele | 132 |
| 5.4.1. TabPanel | 132 |
| 5.4.2. ToolBarOverflowPanel | 132 |
| 5.4.3. UniformGrid | 133 |
| 5.4.4. VirtualizingStackPanel | 133 |
| 5.4.5. SelectiveScrollingGrid | 134 |
| 5.5. Możliwości obsługi wypełnienia zawartości | 134 |
| 5.5.1. Przycinanie | 134 |
| 5.5.2. Przewijanie | 135 |
| 5.5.3. Skalowanie | 136 |
| Pytania testowe | 137 |
| Odpowiedzi do pytań | 141 |

| | |
|--|------------|
| Rozdział 6. Kontrolki | 143 |
| 6.1. Podział kontrolek | 143 |
| 6.2. Kontrolki typu Content | 145 |
| 6.2.1. Przyciski | 146 |
| 6.2.2. Kontrolki informacyjne | 153 |
| 6.2.3. Kontenery | 157 |
| 6.3. Kontrolki typu Items | 159 |
| 6.3.1. Selektory | 159 |
| 6.3.2. Kontrolki Menu | 178 |
| 6.3.3. Pozostałe kontrolki typu Items | 180 |
| 6.4. Kontrolki Ink, tekstowe i inne | 183 |
| 6.4.1. Kontrolki zakresu | 183 |
| 6.4.2. Kontrolki Ink i tekstowe | 185 |
| 6.4.3. Dokumenty | 194 |
| 6.4.4. Kontrolki z datami | 201 |
| 6.4.5. Kontrolka Image | 205 |
| Pytania testowe | 206 |
| Odpowiedzi do pytań | 211 |
| Rozdział 7. Struktura i rodzaje aplikacji WPF | 213 |
| 7.1. Standardowa aplikacja okienkowa | 213 |
| 7.1.1. Klasa Window | 214 |
| 7.1.2. Klasa Application | 217 |
| 7.1.3. SplashScreen | 221 |
| 7.1.4. Okna dialogowe | 221 |
| 7.1.5. Zapamiętywanie i odczytywanie stanu aplikacji | 226 |
| 7.1.6. Okna o niestandardowych kształtach | 228 |
| 7.2. Aplikacje nawigacyjne | 230 |
| 7.2.1. Realizacja nawigacji | 231 |
| 7.2.2. Przekazywanie danych pomiędzy stronami | 236 |
| 7.3. Aplikacje WPF w przeglądarce internetowej | 241 |
| 7.4. Metody instalacji aplikacji WPF | 243 |
| Pytania testowe | 245 |
| Odpowiedzi do pytań | 247 |
| Rozdział 8. WPF i Windows 7 | 249 |
| 8.1. Dostosowywanie paska zadań | 249 |
| 8.2. Listy szybkiego dostępu | 254 |
| 8.2.1. JumpTask | 255 |
| 8.2.2. JumpPath | 258 |
| 8.3. Aero Glass | 260 |
| Pytania testowe | 264 |
| Odpowiedzi do pytań | 265 |
| Rozdział 9. Zdarzenia i polecenia | 267 |
| 9.1. RoutedEvent | 267 |
| 9.1.1. Implementacja zdarzeń RoutedEvent | 271 |
| 9.1.2. Zdarzenia RoutedEvent w praktyce | 275 |
| 9.2. Zdarzenia związane z cyklem życia obiektów | 277 |
| 9.3. Zdarzenia wejściowe | 278 |
| 9.3.1. Zdarzenia myszy | 278 |
| 9.3.2. Zdarzenia klawiatury | 281 |
| 9.3.3. Zdarzenia dotyku i manipulacji | 282 |

| | |
|---|------------|
| 9.4. Polecenia | 285 |
| 9.4.1. Polecenia wbudowane | 286 |
| 9.4.2. Zastosowanie poleceń w praktyce | 289 |
| Pytania testowe | 292 |
| Odpowiedzi do pytań | 293 |
| Rozdział 10. Zasoby | 295 |
| 10.1. Zasoby binarne | 295 |
| 10.1.1. Zasoby binarne w kodzie XAML | 297 |
| 10.1.2. Zasoby binarne w kodzie proceduralnym | 299 |
| 10.2. Zasoby logiczne | 301 |
| 10.2.1. Zasięg zdefiniowanego zasobu | 304 |
| 10.2.2. Zasoby statyczne i dynamiczne | 306 |
| 10.2.3. Czy wszystko może być zasobem? | 308 |
| 10.2.4. Zasoby logiczne i kod proceduralny | 310 |
| 10.2.5. Krótko o zasobach systemowych | 311 |
| Pytania testowe | 312 |
| Odpowiedzi do pytań | 314 |
| Rozdział 11. Wiązanie danych | 315 |
| 11.1. Podstawy wiązania danych | 316 |
| 11.1.1. Definiowanie pierwszego wiązania | 316 |
| 11.1.2. Definiowanie wiązania danych w kodzie XAML | 320 |
| 11.1.3. Definiowanie wiązania ze zwykłymi właściwościami .NET | 322 |
| 11.1.4. Definiowanie wiązania z całą kolekcją | 326 |
| 11.2. Kontrolowanie wyświetlania wiązanych danych | 330 |
| 11.2.1. Formatowanie stringu | 331 |
| 11.2.2. Szablony danych | 332 |
| 11.2.3. Konwertery | 336 |
| 11.3. Zarządzanie kolekcją za pomocą widoku | 341 |
| 11.3.1. Sortowanie | 341 |
| 11.3.2. Grupowanie | 343 |
| 11.3.3. Filtrowanie | 345 |
| 11.4. Walidacja danych | 347 |
| 11.4.1. Wbudowane mechanizmy walidacji | 348 |
| 11.4.2. Definiowanie własnej walidacji | 353 |
| 11.5. Sterowanie przepływem danych | 354 |
| 11.6. Wsparcie dla wielu źródeł danych | 356 |
| 11.6.1. CompositeCollection | 356 |
| 11.6.2. MultiBinding | 356 |
| 11.6.3. PriorityBinding | 357 |
| Pytania testowe | 357 |
| Odpowiedzi do pytań | 360 |
| Rozdział 12. Style, szablony, skórki i motywy | 361 |
| 12.1. Style | 362 |
| 12.1.1. Zarządzanie współdzieleniem stylu | 365 |
| 12.2. Wyzwalacze | 369 |
| 12.2.1. Property triggers | 370 |
| 12.2.2. Data triggers | 371 |
| 12.2.3. Warunki logiczne w wyzwalaczach | 375 |
| 12.3. Szablon kontrolki — ControlTemplates | 377 |
| 12.3.1. Zastosowanie wyzwalaczy w szablonach | 382 |
| 12.3.2. Zależność od wartości właściwości rodziców | 385 |
| 12.4. Umieszczanie szablonów kontrolki w stylach | 390 |

| | |
|---|------------|
| 12.5. Bardziej złożone szablony | 391 |
| 12.5.1. Części | 394 |
| 12.5.2. Visual State Manager | 398 |
| 12.6. Skórki | 404 |
| 12.7. Motywy | 409 |
| Pytania testowe | 413 |
| Odpowiedzi do pytań | 417 |
| Rozdział 13. Grafika 2D | 419 |
| 13.1. Pędzle — klasa Brush | 419 |
| 13.1.1. Pędzle malujące kolorem | 421 |
| 13.1.2. Pędzle pokrywające | 427 |
| 13.1.3. BitmapCacheBrush | 441 |
| 13.2. Kształty | 445 |
| 13.2.1. „Ciężkie” kształty — klasa Shape | 447 |
| 13.2.2. „Lekkie” kształty — klasa Geometry | 454 |
| 13.3. Klasa Drawing | 464 |
| 13.3.1. Klasa Pen | 468 |
| 13.3.2. Podsumowanie kształtów prostych, czyli tworzymy clipart | 469 |
| Pytania testowe | 472 |
| Odpowiedzi do pytań | 477 |
| Rozdział 14. Grafika 3D | 479 |
| 14.1. Pierwsza scena 3D w interfejsie użytkownika | 480 |
| 14.2. Umiejscowienie grafiki 3D w interfejsie użytkownika | 483 |
| 14.3. Układ współrzędnych w przestrzeni 3D | 484 |
| 14.4. Punkt widzenia — Camera | 485 |
| 14.4.1. Rodzaje kamer | 489 |
| 14.5. Transformacje 3D | 492 |
| 14.5.1. Translacja | 495 |
| 14.5.2. Skalowanie | 496 |
| 14.5.3. Obrót | 498 |
| 14.5.4. Grupowanie transformacji | 500 |
| 14.6. Podstawowy element sceny 3D — Model3D | 501 |
| 14.6.1. Oświetlenie — klasa Light | 506 |
| 14.6.2. Materiał — klasa Material | 511 |
| 14.6.3. Model3DGroup | 516 |
| 14.7. Visual3D | 517 |
| 14.7.1. ModelVisual3D | 518 |
| 14.7.2. Viewport2DVisual3D | 519 |
| 14.7.3. UIElement3D | 521 |
| 14.8. Viewport3D | 524 |
| Pytania testowe | 527 |
| Odpowiedzi do pytań | 531 |
| Rozdział 15. Animacje | 533 |
| 15.1. Pierwsza animacja | 534 |
| 15.2. Klasy implementujące animacje | 537 |
| 15.3. Właściwości sterujące przebiegiem animacji | 540 |
| 15.3.1. Duration | 540 |
| 15.3.2. From, To, By | 541 |
| 15.3.3. BeginTime | 543 |
| 15.3.4. AutoReverse | 543 |
| 15.3.5. RepeatBehavior | 544 |
| 15.3.6. SpeedRatio i przyspieszenie animacji | 545 |

| | |
|--|------------|
| 15.4. Umieszczenie animacji w kodzie XAML | 546 |
| 15.4.1. Wyzwalacze EventTrigger | 547 |
| 15.4.2. Klasy BeginStoryboard i Storyboard | 548 |
| 15.5. Animacje keyframe | 559 |
| 15.5.1. Interpolacja liniowa | 561 |
| 15.5.2. Interpolacja dyskretna | 563 |
| 15.5.3. Interpolacja Spline | 565 |
| 15.6. Animacje PathGeometry | 567 |
| 15.7. Funkcje ułatwiające | 568 |
| Pytania testowe | 570 |
| Odpowiedzi do pytań | 574 |
| Skorowidz | 575 |

Rozdział 8.

WPF i Windows 7

W tym rozdziale:

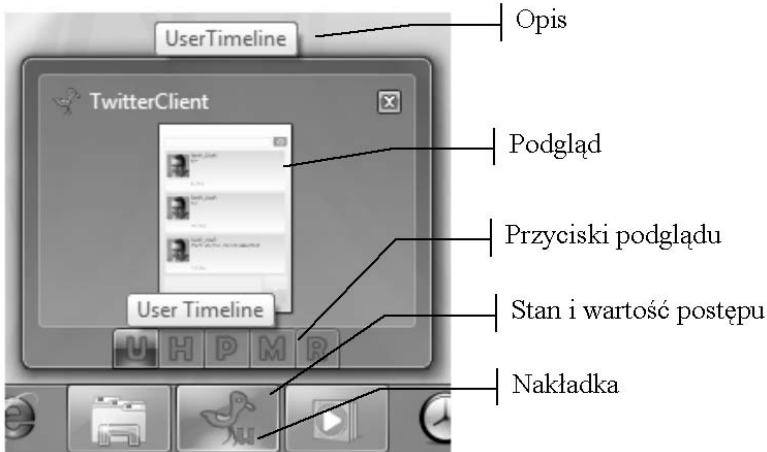
- ◆ dostosowywanie paska zadań,
- ◆ listy szybkiego dostępu,
- ◆ Aero Glass.

Windows 7 posiada nowe funkcje w porównaniu ze swoimi poprzednikami — należą do nich na przykład listy szybkiego dostępu. WPF w związku z tym posiada API, za pomocą którego możemy w naszych aplikacjach wykorzystywać te nowe funkcje — o tym właśnie będzie ten rozdział. Zobaczysz, jak można wpływać na informacje pokazywane w pasku zadań, jak w aplikacji WPF można manipulować tym, co jest wyświetlane w podglądzie aplikacji, oraz nauczysz się umieszczać zawartość w liście szybkiego dostępu. Na końcu rozdziału poruszę kwestie związane z interfejsem Aero Glass — zmienimy naszą aplikację `TwitterClient` tak, by całe okno wraz z listą wiadomości było przezroczyste.

8.1. Dostosowywanie paska zadań

Pasek zadań w systemie Windows 7 ma trochę inny wygląd niż jego poprzednicy, posiada również pewne nowe funkcje. Zapewne każdy zauważył, że w momencie najechania kursorem na pasek zadań poszczególne ikony oznaczające aplikacje są kolorowane dominującą barwą. Dodatkowo w przypadku niektórych programów można zaobserwować, że jeżeli wystąpi błąd ich działania, ich ikona w pasku zadań staje się czerwona. Jest jeszcze jedna funkcjonalność paska zadań, z którą na pewno każdy miał okazję się spotkać — jeżeli pobieramy jakieś dane z internetu, ikona przeglądarki w pasku zadań wskazuje poziom postępu procesu pobierania — jest to również jedna z nowych możliwości manipulacji zachowaniem paska zadań. Wszystkie te funkcjonalności z wyjątkiem zmiany koloru podświetlania ikony w pasku zadań (w tym przypadku jedyną opcją zmiany koloru podświetlania aplikacji jest zmiana kolorów ikony) da się zaimplementować w aplikacji WPF.

Zacznijmy od tego, co nowego możemy zaobserwować w związku z paskiem zadań (pomijamy na razie kwestie list szybkiego dostępu omówionych w kolejnych podrozdziałach). Rysunek 8.1 pokazuje podgląd aplikacji oraz wygląd paska zadań wraz z jego poszczególnymi elementami (jako przykład w tym rozdziale posłuży nam aplikacja *TwitterClient*).



Rysunek 8.1. Pasek zadań wraz z podglądem programu *TwitterClient* oraz opisem poszczególnych jego elementów

Klasą, za pomocą której można manipulować paskiem zadań, jest *TaskbarItemInfo*. Poniższy fragment kodu pokazuje, jak zdefiniowano opis, który widoczny jest na rysunku 8.1 (*UserTimeline*).

```
<Window.TaskbarItemInfo>
  <TaskbarItemInfo Description="{Binding ElementName=Window, Path=TimeLine}"/>
</Window.TaskbarItemInfo>
```

Opis jest ustawiany przez wpisanie do właściwości *Description* obiektu *TaskbarItemInfo* odpowiedniego napisu. W powyższym kodzie napis jest zdefiniowany za pomocą wiązania danych z właściwością *TimeLine* przechowującą typ wyliczeniowy zawierający wartości *User Timeline*, *Home Timeline*, *Private Timeline*, *Mentions* i *Retweete By Me* — czyli grupy wiadomości. Ponieważ wartości tego typu są wpisywane do zmiennej przy każdej zmianie wartości właściwości *TimeLine*, utworzone wiązanie danych zapewni samoczynne odświeżanie opisów.



Wskazówka

„Grupy wiadomości” to wiadomości serwisu Twitter, które są zgrupowane tematycznie, przykładowo *UserTimeLine* zawiera tylko wiadomości napisane przez nas, *HomeTimeLine* to wiadomości nasze i osób, które śledzimy. W języku angielskim nazywane są one *time lines*. Niestety tej nazwy nie da się poprawnie przetłumaczyć na język polski — jedyne, co przyszło mi do głowy, to właśnie „grupy wiadomości”. Ponieważ tłumaczenie wydaje mi się niezbyt trafione, a niestety lepszego nie jestem w stanie wymyślić (Google też ;)), proponuję stosowanie angielskiego określenia *time line*.

Właściwości zależne

Aby powyższe wiązania zadziałały, właściwość `TimeLine` nie może być zwykłą właściwością, ponieważ technika wiązania danych wymaga, aby źródło (w tym przypadku właściwość `TimeLine`) powiadamiało cel wiązania (właściwość `Description`) o zmianie swojej zawartości — w przeciwnym razie właściwość `Description` przechowywałaby ciągle tę samą, wpisaną za pierwszym razem wartość właściwości `TimeLine`. Właściwość `TimeLine`, aby mogła powiadamiać o swojej zmianie, musiała zostać zdefiniowana jako właściwość zależna, co przedstawia poniższy fragment kodu.

```
DependencyProperty.Register("TimeLine", typeof(TimeLines), typeof(MainWindow));
public TimeLines TimeLine
{
    get { return (TimeLines)GetValue(TimeLineProperty); }
    set { SetValue(TimeLineProperty, value); }
}
```

Zagadnienie właściwości zależnych omówione jest dokładnie w rozdziale „Podstawy WPF”.

Zagadnienie wiązania danych jest omówione w rozdziale „Wiązanie danych”, jeśli więc zapis w powyższym kodzie nie jest dla Ciebie zrozumiały, nie przejmuj się — wystarczy wiedzieć, że dzięki niemu samoczynnie będą się zmieniały podpowiedzi po przełączeniu `time line` na inny — rysunek 8.2 pokazuje opis po zmianie `time line` na `HomeTimeLine`.

Rysunek 8.2.

Zmiana opisu
po zmianie `time line`



Zamiast używać wiązania danych, moglibyśmy zdefiniować wartość w następujący sposób:

```
Description="Aplikacja TwitterClient"
```

co spowodowałoby wyświetlenie napisu „Aplikacja TwitterClient” zamiast napisu widocznego na rysunku 8.2.

Następną w kolejności informacją prezentowaną w pasku zadań, a pokazaną na rysunku 8.1, jest podgląd. W tym przypadku nie mamy dużego wpływu na to, jak może być w nim prezentowana nasza aplikacja — jedyną możliwość manipulacji to ustawienie odpowiednich marginesów tak, by tylko część okna była widoczna. Takiego ustawienia dokonuje się za pomocą właściwości `ThumbnailClipMargin` typu `Thickness`, czyli za

pomocą czterech liczb typu `Double` określamy odstęp z każdej ze stron prostokąta (kolejno od lewej, od góry, od prawej i od dołu) — przykład definicji poniżej.

```
<TaskbarItemInfo ThumbnailClipMargin="5,5,5,480" .../>
```

Działanie marginesów i jednocześnie powyższego fragmentu kodu pokazuje rysunek 8.3 — widać, że tylko górna część aplikacji jest pokazywana — dolna część stanowiąca 480 jednostek została „obcięta”.

Rysunek 8.3.

W wyniku zastosowania właściwości `ThumbnailClip`

↪ `Margin` w podglądzie prezentowana jest tylko część aplikacji



Kolejną z udostępnionych możliwości dostosowywania wyglądu paska zadań jest umieszczanie w nim przycisków. Realizuje się to przy wykorzystaniu właściwości `ThumbButtonInfos`, przechowującej kolekcję obiektów `ThumbButtonInfo`. Ważne, że nie są to przyciski dziedziczące po klasie `UIElement`. Posiadają jednak podstawowe funkcjonalności przycisków. Niestety (a może i „na szczęście”) zawartością takiego przycisku może być jedynie obrazek — nie można więc po prostu wpisać do niego zawartości tekstowej, która zostanie zaprezentowana. Sposób definiowania powyższych przycisków przedstawia poniższy fragment kodu. Jego działanie możemy obserwować na wszystkich powyższych rysunkach.

```
<TaskbarItemInfo ...>
  <TaskbarItemInfo.ThumbButtonInfos>
    <ThumbButtonInfo Description="User Timeline"
      ↪ImageSource="Images\UGreen.png" Click="user_Click"/>
    <ThumbButtonInfo Description="Home Timeline"
      ↪ImageSource="Images\HGreen.png" Click="home_Click"/>
    <ThumbButtonInfo Description="Public Timeline"
      ↪ImageSource="Images\PGreen.png" Click="public_Click"/>
    <ThumbButtonInfo Description="Mentions" ImageSource="Images\MGreen.png"
      ↪Click="mentions_Click"/>
    <ThumbButtonInfo Description="Retweeted By Me"
      ↪ImageSource="Images\RGreen.png" Click="rbm_Click"/>
  </TaskbarItemInfo.ThumbButtonInfos>
</TaskbarItemInfo>
```

Jak widzimy, opisy, które pojawiały się na rysunkach nad przyciskami, zdefiniowane były za pomocą właściwości `Description`. Dla każdego z przycisków można również definiować uchwyt dla zdarzenia kliknięcia.

Pod przyciskami widoczny jest już tylko sam pasek zadań. Tam mamy do dyspozycji dwie możliwości zmiany jego wyglądu — zdefiniowanie nakładki dla ikony aplikacji w pasku oraz wskazanie stanu i postępu danego procesu.

Element `TaskbarItemInfo` oferuje dodatkowe możliwości informowania użytkownika o stanie aplikacji i wykonania operacji, odpowiednio za pomocą właściwości `ProgressValue` i `ProgressState`. `ProgressValue` jest właściwością typu `Double` przyjmującą wartości od 0 do 1 — oznaczające procent wykonania czynności. Na wszystkich dotychczasowych rysunkach właściwość ta była ustawiona na wartość 1, co oznaczało 100% wykonania danego procesu. Druga właściwość — `ProgressState` — może przyjmować następujące wartości¹:

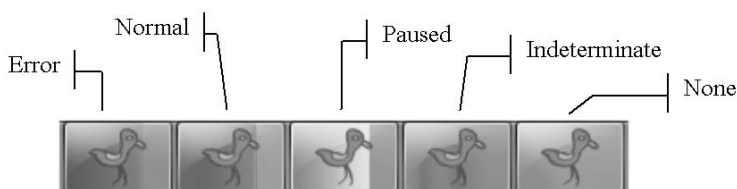
- ♦ `Error` — oznacza błąd, charakteryzuje się czerwonym wypełnieniem dookoła ikony aplikacji;
- ♦ `Normal` — stan normalny — zielone wypełnienie dookoła ikony aplikacji;
- ♦ `Paused` — pauza — pomarańczowe wypełnienie dookoła ikony aplikacji;
- ♦ `Indeterminate` — nieokreślony — stosowany w przypadku monitorowania procesu, o którego zaawansowaniu nie mamy żadnych informacji — prezentowany jest w postaci zielonego paska, który jest animowany w kierunku od lewej do prawej;
- ♦ `None` — domyślna wartość, stan postępu wykonania czynności nie jest prezentowany w pasku zadań.

Za jej pomocą możemy informować użytkownika o tym, w jakim stanie znajduje się w danym momencie aplikacja.

Na wszystkich powyższych rysunkach mogliśmy zobaczyć następujące ustawienia powyższych właściwości:

```
<TaskbarItemInfo ProgressState="Normal" ProgressValue="1" .../>
```

Rysunek 8.4 prezentuje natomiast wygląd aplikacji `TwitterClient` w pasku zadań w zależności od różnych wartości właściwości `ProgressState` oraz `ProgressValue` równej 0.75.



Rysunek 8.4. Wygląd paska zadań aplikacji `TwitterClient` dla właściwości `ProgressValue = '0.75'` oraz różnych wartości właściwości `ProgressState`

Na koniec zostały nam nakładki. Nakładka również była widoczna na każdym z poprzednich obrazków — jest to mały obrazek prezentowany w pasku zadań w prawym dolnym rogu obok ikony aplikacji. Działa tylko wtedy, gdy pasek zadań pokazuje duże ikony. Nakładki stosowane są do przekazywania dodatkowych informacji użytkownikowi. Na powyższych przykładach widoczne były naprzemiennie literki *u* i *h* — odpowiednio skróty od *UserTimeLine* i *HomeTimeLine*. Służyły więc w tym przypadku do przekazywania informacji użytkownikowi o tym, jaki time line aktualnie ogląda

¹ Ibidem, s. 246.

w pasku zadań. Definicja nakładki jest bardzo prosta — do właściwości `Overlay` klasy `TaskbarItemInfo` należy wstawić nazwę obrazka, który ma być prezentowany w nakładce, co przedstawia poniższy fragment kodu.

```
<TaskbarItemInfo Overlay="Images\URed.png" ... />
```

Dynamiczna zmiana nakładki

Rysunki z tego podrozdziału nie pokazywały ciągle tej samej nakładki — zmieniała się ona wraz ze zmianą `time line` — nakładki były zamieniane w trakcie działania aplikacji. Jednym ze sposobów zmiany obrazka w nakładce jest zdefiniowanie wiązania danych pomiędzy właściwością `Overlay` a zmieniającą się właściwością `TimeLine` (podobnie jak to miało miejsce w przypadku zmiany opisów aplikacji). `TimeLine` przechowuje jednak jedynie oznaczenia `time line`, podczas gdy `Overlay` chce dostać dane obrazka, który ma być umieszczony w nakładce. Konieczne więc byłoby użycie konwertera (zagadnienie konwerterów jest dokładnie omówione w rozdziale „Wiązanie danych”), który zamieniłby wartość `TimeLine` na określoną reprezentację obrazka.

Inną możliwością jest zdefiniowanie zasobu, na przykład w oknie `Window`, jak w poniższym kodzie (zagadnienie zasobów zostanie omówione w rozdziale „Zasoby”).

```
<Window.Resources>
  <DrawingImage x:Key="HRedOverlay">
    <DrawingImage.Drawing>
      <ImageDrawing ImageSource="Images\HRed.png" Rect="0.0,16,16"/>
    </DrawingImage.Drawing>
  </DrawingImage>
  <DrawingImage x:Key="URedOverlay">
    <DrawingImage.Drawing>
      <ImageDrawing ImageSource="Images\URed.png" Rect="0.0,16,16"/>
    </DrawingImage.Drawing>
  </DrawingImage>
  ...
</Window.Resources>
```

Następnie w momencie zmiany `time line`, czyli na przykład po naciśnięciu przycisku z paska zadań, należałoby odnaleźć konkretny zasób i wstawić jego wartość do właściwości `Overlay`:

```
this.taskBarItemInfo.Overlay = (DrawingImage)FindResource("HRedOverlay");
```

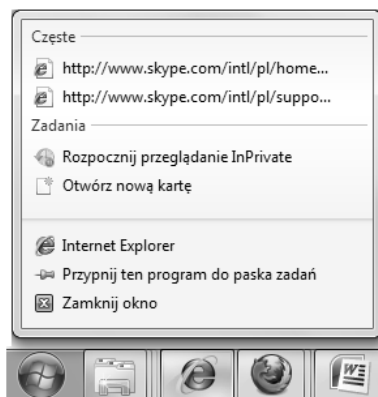
8.2. Listy szybkiego dostępu

Lista szybkiego dostępu jest tym, co pokazuje się po naciśnięciu prawym przyciskiem myszy na kontrolce aplikacji w pasku zadań lub przeciągnięciu kursora myszy do góry z jednocześnie przyciśniętym lewym przyciskiem myszy. Przykład listy szybkiego dostępu programu Internet Explorer pokazuje rysunek 8.5.

Domyślna lista szybkiego dostępu jest dołączana do każdej aplikacji w systemie Windows 7. Aplikacja może mieć listę szybkiego dostępu zarówno wtedy, kiedy działa, jak i wtedy, gdy jest wyłączona. Przykład domyślnej listy szybkiego dostępu dla aplikacji `TwitterClient` przedstawia rysunek 8.6.

Rysunek 8.5.

Lista szybkiego dostępu programu Internet Explorer w Windows 7



Rysunek 8.6. *Lista szybkiego dostępu dla programu TwitterClient, gdy aplikacja jest uruchomiona (po lewej) oraz gdy aplikacja jest zamknięta (po prawej)*

Oczywiście aby lista po prawej stronie rysunku 8.6 mogła być pokazana, konieczne było przypięcie aplikacji do paska zadań.

Listę szybkiego dostępu można zmieniać za pomocą właściwości dołączonej `JumpList`. ↪`JumpList` ustawionej na wartość `JumpList`. Lista jest dołączana do powłoki jeden raz, zaraz po uruchomieniu aplikacji, więc umieszczamy jej definicję w klasie `App`. Alternatywną możliwością ustawienia listy szybkiego dostępu jest wywołanie metody `JumpList.SetJumpList` w kodzie proceduralnym.

Klasa `JumpList` posiada właściwość `JumpItems`, do której można wpisać zawartość listy. Mogą nią być dwie klasy dziedziczące po `JumpItem` — `JumpTask` i `JumpPath`². W skrócie można powiedzieć, że `JumpTask` to połączenie kierujące do programu, natomiast `JumpPath` to połączenie prowadzące do pliku. W kolejnych podrozdziałach omówię każdą z tych klas.

8.2.1. JumpTask

Element `JumpTask` reprezentuje zadania do wykonania. Na rysunku 8.5 zadaniami są *Rozpocznij przeglądanie InPrivate* i *Otwórz nową kartę*. W praktyce służą one do uruchomienia aplikacji z parametrami linii poleceń. Przykład definicji elementu `JumpList` w aplikacji `TwitterClient` przedstawia poniższy fragment kodu:

² A. Nathan, *WPF 4 Unleashed*, op. cit., s. 234, <http://msdn.microsoft.com/en-us/library/system.windows.shell.jumpitem.jumpitems.aspx>, 2012-01-22.

```

<Application x:Class="TwitterClient.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="MainWindow.xaml">
    <JumpList.JumpList>
        <JumpList>
            <JumpTask Title="User time line" Arguments="UserTimeLine"
                Description="Uruchamia TwitterClient w trybie User time line"/>
            <JumpTask Title="Home time line" Arguments="HomeTimeLine"
                Description="Uruchamia TwitterClient w trybie Home time line"/>
        </JumpList>
    </JumpList.JumpList>
    ...
</Application>

```

W powyższym kodzie utworzono dwa elementy `JumpTask`. Pierwszy z nich uruchamia kolejną instancję aplikacji `TwitterClient` z argumentem `UserTimeLine`. Drugi uruchamia tę aplikację z parametrem `HomeTimeLine`. Co to za argumenty? Są to po prostu argumenty linii poleceń, które możemy odczytać na przykład tak:

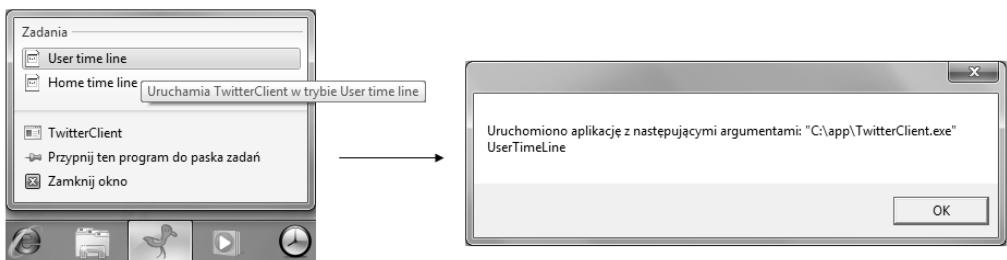
```

MessageBox.Show("Uruchomiono aplikację z następującymi argumentami: "+
    ↪Environment.CommandLine);

```

i od wyniku uzależnić działanie aplikacji (więcej informacji o metodach odczytywania parametrów przekazanych w linii poleceń znajduje się w rozdziale „Struktura i rodzaje aplikacji WPF”).

Rysunek 8.7 pokazuje zarówno wygląd listy zdefiniowanej w powyższym fragmencie kodu, jak i to, co przekazywane jest do aplikacji po wybraniu pierwszego zadania. Do odczytania przekazanych parametrów została wykorzystana klasa `Environment` i jej właściwość `CommandLine` (jej użycie widać w pokazanym powyżej fragmencie kodu). `CommandLine` przechowuje dwie wartości — pierwsza wartość to ścieżka do aplikacji, natomiast druga to argument przekazany do aplikacji. Argumenty te zostały wyświetlone w konstruktorze klasy `MainWindow` za pomocą okienka `MessageBox`.



Rysunek 8.7. *JumpList* ze zdefiniowanym elementem *JumpTask* i argumenty przekazane do aplikacji wyświetlone za pomocą *MessageBox*

Co do samego wyglądu listy, to widać, że podpowiedź wyświetlana do każdego z zadań definiowana jest za pomocą właściwości `Description` klasy `JumpTask`. Element `Title` określa natomiast treść, która pokaże się jako odnośnik do danego zadania.



Wskazówka

Przeanalizujemy krótko, jak zachowuje się `TwitterClient` po wybraniu zadania z listy. W przypadku gdy aplikacja jest zamknięta, przekazywane są odpowiednie argumenty, po czym `TwitterClient` jest uruchamiany. Spójrzmy teraz na działanie listy szybkiego dostępu dla programu Internet Explorer. Po wybraniu zadania *Otwórz nową kartę* gdy aplikacja jest wyłączona, uruchamiany jest Internet Explorer z nową kartą — czyli analogicznie do naszego przykładu. A teraz spójrzmy, co się stanie, gdy dla już uruchomionej aplikacji wybierzemy to samo zadanie — nowa karta otwierana jest w tej samej instancji aplikacji. Co natomiast dzieje się w naszym przykładzie? Za każdym razem otwierana jest nowa aplikacja. Ten problem można rozwiązać przy użyciu klasy `Microsoft.VisualBasic.ApplicationServices.WindowsFormsApplicationBase` — umożliwia ona zaimplementowanie komunikacji z parametrami przekazywanymi przez linię poleceń w czasie działania jednej instancji aplikacji.

Istnieją dodatkowe możliwości definiowania listy szybkiego dostępu. Przykład definicji zadania, które uruchamia aplikację `mspaint.exe`, oraz ikony dla tego zadania przedstawia poniższy fragment kodu definiujący kolejny element `JumpTask`:

```
<JumpTask/>
<JumpTask Title="Paint"
  Description="Otwiera program Paint"
  ApplicationPath="%WINDIR%\system32\mspaint.exe"
  IconResourcePath="%WINDIR%\system32\mspaint.exe"/>
<JumpTask/>
</JumpList>
```

Na rysunku 8.8 przedstawiono efekt działania tej definicji — widać, że w liście pojawiła się dodatkowa pozycja wraz z ikoną (odpowiednio właściwości `ApplicationPath` oraz `IconResourcePath`).

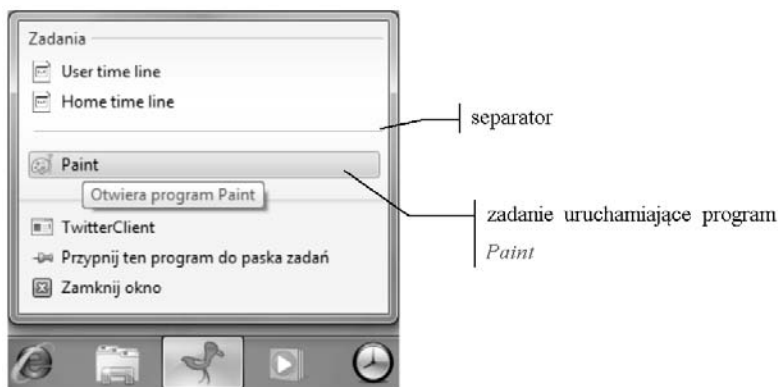


Wskazówka

Biblioteki `%WINDIR%\System32\shell32.dll` i `%WINDIR%\System32\imageres.dll` zawierają wiele ikon przydatnych do zastosowania w listach szybkiego dostępu.

Rysunek 8.8.

Program `mspaint.exe` uruchamiany za pomocą jednego z zadań



Po wybraniu tego zadania zostanie uruchomiony program `mspaint.exe`. Widać, że zarówno ikony, jak i program lokalizowane są za pomocą zmiennych systemowych. Dodatkowo przed i po zadaniu `Paint` umieszczone zostały definicje elementów `JumpTask`

bez żadnych atrybutów — służą one do zdefiniowania separatorów również widocznych na rysunku 8.8.

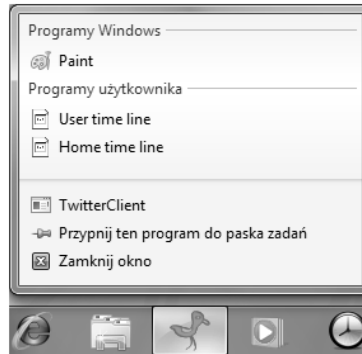
Windows 7 umożliwia również zgrupowanie elementów znajdujących się na liście szybkiego dostępu. Służy do tego właściwość `CustomCategory` typu string klasy `JumpItem`. Przykład definicji przedstawia poniższy fragment kodu:

```
<JumpTask CustomCategory="Programy użytkownika" Title="User time line"
  ↳Arguments="UserTimeLine"
  Description="Uruchamia TwitterClient w trybie User time line"/>
<JumpTask CustomCategory="Programy użytkownika" Title="Home time line"
  ↳Arguments="HomeTimeLine"
  Description="Uruchamia TwitterClient w trybie Home time line"/>
<JumpTask/>
<JumpTask CustomCategory="Programy Windows" Title="Paint"
  Description="Otwiera program Paint"
  ApplicationPath="%WINDIR%\system32\mspaint.exe"
  IconResourcePath="%WINDIR%\system32\mspaint.exe"/>
<JumpTask/>
```

Grupy elementów są więc identyfikowane za pomocą nazwy grupy, do której chcą przynależeć. Co by się stało, gdyby dla jednego z elementów właściwość `CustomCategory` została niezdefiniowana? W takim przypadku zostałaby utworzona dla tego elementu domyślna grupa o nazwie „Zadania”, którą mogliśmy zobaczyć na przykład na rysunku 8.7.

Wygląd listy po zdefiniowaniu grup tematycznych przedstawia rysunek 8.9.

Rysunek 8.9.
*Poszczególne zadania
przydzielone do grup*



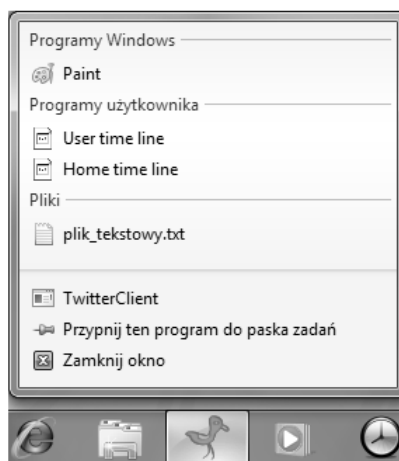
8.2.2. JumpPath

Za pośrednictwem `JumpPath` określa się elementy wskazujące na ścieżki do plików, które mogą być uruchomione przez daną aplikację. Przykład definicji i działania tego elementu przedstawiony jest w poniższym fragmencie kodu.

```
<JumpList.JumpList>
  <JumpList>
    <JumpPath Path="C:\plik_tekstowy.txt" CustomCategory="Pliki"/>
  ...
  </JumpList>
</JumpList.JumpList>
```

Jak widać, definicja `JumpPath` wygląda analogicznie do `JumpTask`. Ścieżka do pliku tekstowego została zdefiniowana za pomocą właściwości `Path` oraz umieszczona została w grupie *Pliki* — właściwość `CustomCategory` jest dziedziczona po klasie `JumpItem`. Na rysunku 8.10 przedstawiono wygląd elementu `JumpPath` w liście szybkiego dostępu.

Rysunek 8.10.
Element `JumpPath`
w liście szybkiego dostępu



Aby zdefiniowany plik `plik_tekstowy.txt` mógł się pokazać na liście, konieczne jest, by znajdował się we wskazywanej przez nas lokalizacji. Ponieważ nasza aplikacja `TwitterClient` nie posiada funkcjonalności otwierania plików tekstowych, po wybraniu pliku `plik_tekstowy.txt` z listy zostanie uruchomiona druga instancja aplikacji `TwitterClient`.

W pokazanym fragmencie kodu definiującego element `JumpPath` widać ścieżkę bezwzględną do pliku. Jest to jedyna możliwość wskazania lokalizacji plików w przypadku elementu `JumpPath`³. Dlatego też aplikacje, które wykorzystują ten element, najczęściej definiują go w kodzie proceduralnym, co pozwala wykorzystać bieżące ustawienia, np. zmienne środowiskowe.

Na koniec tego podrozdziału spójrzmy jeszcze raz na listę `JumpList` programu Internet Explorer. Zawiera ona grupę o nazwie *Częste*. Pokazywana jest ona często zamiennie z inną grupą o nazwie *Najnowsze* — czyli po prostu listą często i ostatnio otwieranych plików i zadań. Na liście tej znajdują się pliki i zadania uruchamiane często lub ostatnio za pomocą okna dialogowego do otwierania plików lub konkretnej aplikacji, jeżeli dany rodzaj pliku został z nią skojarzony.

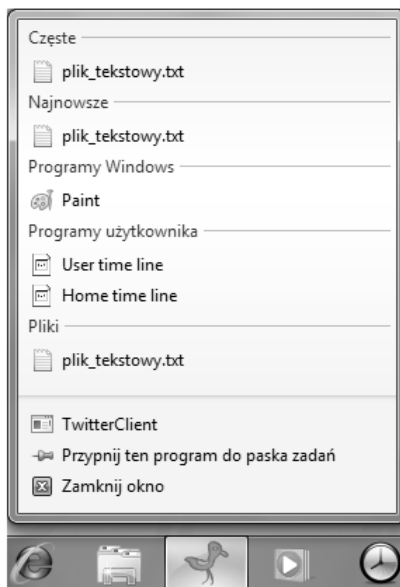
Aby taka grupa pokazała się na naszej liście, wystarczy ustawić dwie właściwości klasy `JumpList` odpowiedzialne za pokazywanie grupy najczęściej otwieranych plików i ostatnio otwieranych plików — odpowiednio `ShowFrequentCategory` i `ShowRecentCategory` — co przedstawia kod zamieszczony poniżej.

```
<JumpList ShowFrequentCategory="True" ShowRecentCategory="True">
...
</JumpList>
```

³ Ibidem, s. 241.

Efekt widoczny jest na liście JumpList programu TwitterClient — rysunek 8.11.

Rysunek 8.11.
Element JumpList
ze zdefiniowanymi
właścwościami
ShowFrequentCategory
i ShowRecentCategory



Na rysunku widać, że zarówno w kategorii *Częste*, jak i *Najnowsze* prezentowane są te same pliki — dlatego też zazwyczaj lista ostatnio otwieranych plików jest stosowana zamiennie z listą najczęściej otwieranych. Plik *plik_tekstowy.txt* znalazł się na liście najczęstszych i ostatnich, ponieważ pliki *.txt* zostały skojarzone w moim systemie z aplikacją TwitterClient, po czym otwarte przy jej użyciu. Generalnie tym, które pliki znajdują się na liście, zajmuje się powłoka Windows, ale możliwe jest też wymuszenie dodania elementów do omawianych grup poprzez wywołanie funkcji `JumpList.AddToRecentCategory`, której argumentami są albo zadania — `JumpTask`, albo pliki — `JumpPath`.

8.3. Aero Glass

Aero Glass jest terminem określającym okno, którego ramki są przezroczyste. Część będąca obramowaniem może być rozciągnięta na część użytkową aplikacji, co pozwala na uzyskanie efektu przezroczystej powłoki. Przykłady aplikacji wykorzystujących Aero Glass to Internet Explorer i Windows Media Player. Istnieją dwa ograniczenia co do możliwości definiowania przezroczystego obszaru⁴:

- ◆ Obszar ten musi być prostokątem — nie może to być inny kształt.
- ◆ Obszar przezroczysty jest rozszerzeniem ramki okna, więc nie może być zdefiniowany wewnątrz okna — jedyne opcje to albo przezroczysta ramka dookoła okna, albo całe przezroczyste okno.

⁴ M. MacDonald, *Pro WPF Windows Presentation Foundation in .NET 3.0*, op. cit., s. 242.

Efekt rozciągnięcia obramowania okna Aero Glass można uzyskać, wykorzystując API biblioteki *dwmapi.dll*, konkretnie funkcję `DwmExtendFrameIntoClientArea`. Za jej pomocą można rozszerzyć obramowanie okna na cały obszar roboczy lub jego część. Aby było możliwe użycie funkcji pochodzącej z kodu niezarządzanego, konieczne jest skorzystanie z mechanizmu `PInvoke` (ang. *Platform Invocation Service*). Umożliwia on przy wykorzystaniu atrybutu `[DllImport]` importowanie funkcji zapisanych w kodzie niezarządzanym w bibliotekach DLL. Poniżej znajduje się przykład definicji klasy `AeroGlass`, w której zdefiniowana jest metoda `ExtendGlass` używająca metody `DwmExtendFrameIntoClientArea`. Na początku definiujemy strukturę, która będzie przechowywać odpowiednie marginesy. Następnie za pomocą `Platform Invocation Services` deklarujemy użycie metod z biblioteki. Jedną z nich to wspomniana `DwmExtendFrameIntoClientArea`, druga będzie nam służyła do sprawdzenia, czy kompozycja pulpitu jest włączona.

Kompozycja pulpitu

Kompozycja pulpitu (ang. *Desktop composition*) to funkcjonalność zaprezentowana pierwszy raz w Windows Vista. Zmienia ona sposób, w jaki aplikacje pokazują piksele na ekranie, mianowicie jeżeli kompozycja jest włączona, nie rysują one okien bezpośrednio na ekranie, lecz ich instrukcje odnośnie do rysowania obiektów są przekierowywane do urządzenia graficznego. Dopiero to urządzenie renderuje ostateczną zawartość. Kompozycją pulpitu zarządza DMW, czyli `Desktop Windows Manager`. Funkcjonuje on jako usługa systemowa i może być wyłączony za pomocą narzędzi administracyjnych. Stosując kompozycję pulpitu, DMW udostępnia szereg efektów na ekranie, do których należą między innymi wsparcie dla wysokich rozdzielczości czy omawiane okna typu „Glass”⁵.

Rozszerzenie ramki okna nastąpi jedynie w przypadku, gdy kompozycja pulpitu jest włączona oraz wersja systemu nie jest niższa niż 6 (wersje systemów operacyjnych Windows Vista i późniejsze).

W kolejnych liniach widzimy: pobranie uchwytu dla okna, zdefiniowanie jego tła na przezroczyste, pobranie rozdzielczości ekranu (aby marginesy zostały poprawnie obliczone) oraz wywołanie metody `DwmExtendFrameIntoClientArea` z odpowiednio przeliczonymi wartościami struktury `MARGINS`.

```
using System;
using System.Runtime.InteropServices;
using System.Windows;
using System.Windows.Interop;
using System.Windows.Media;

[StructLayout(LayoutKind.Sequential)]
public struct MARGINS
{
    public MARGINS(Thickness thickness, float dpiX, float dpiY)
    {
        Left = (int)(thickness.Left * dpiX);
        Right = (int)(thickness.Right * dpiX);
        Top = (int)(thickness.Top * dpiY);
        Bottom = (int)(thickness.Bottom * dpiY);
    }
}
```

⁵ <http://msdn.microsoft.com/en-us/library/windows/desktop/aa969540%28v=vs.85%29.aspx>, 2012-01-22.

```

    }
    public int Left;
    public int Right;
    public int Top;
    public int Bottom;
}

public class AeroGlass
{
    //deklaracja metod z biblioteki dwmapi.dll
    [DllImport("dwmapi.dll", PreserveSig = false)]
    static extern void DwmExtendFrameIntoClientArea(
        IntPtr hWnd, ref MARGINS pMarInset);
    [DllImport("dwmapi.dll", PreserveSig = false)]
    static extern bool DwmIsCompositionEnabled();

    public static void ExtendGlass(Window window, Thickness margin)
    {
        //jeżeli kompozycja jest nieustawiona lub system jest w starszej wersji, nic nie robimy
        if (DwmIsCompositionEnabled() && Environment.OSVersion.Version.Major > 5)
        {
            //uchwyt dla okna
            WindowInteropHelper helper = new WindowInteropHelper(window);
            HwndSource mainWindowSrc = (HwndSource)HwndSource.FromHwnd(helper.Handle);

            //ustawienie tła okna na przezroczyste
            window.Background = Brushes.Transparent;
            mainWindowSrc.CompositionTarget.BackgroundColor = Colors.Transparent;
            //pobieramy rozdzielczość
            System.Drawing.Graphics desktop =
                System.Drawing.Graphics.FromHwnd(mainWindowSrc.Handle);
            float dpiX = desktop.DpiX / 96;
            float dpiY = desktop.DpiY / 96;
            //ustawiamy marginesy
            MARGINS margins = new MARGINS(margin, dpiX, dpiY);
            //zwiększamy przezroczystą ramkę o stosowny rozmiar
            DwmExtendFrameIntoClientArea(mainWindowSrc.Handle, ref margins);
        }
        else
        {
            window.Background = SystemColors.WindowBrush;
        }
    }
}

```

Teraz trzeba jeszcze w naszej klasie `MainWindow` wywołać metodę `ExtendGlass` klasy `AeroGlass`. Poniżej zamieszczono przykład użycia w nadpisanej metodzie `OnSourceInitialized`. Przekazany argument `new Thickness(-1)` oznacza, że ramka okna zostanie rozszerzona na całą jego powierzchnię.

```

protected override void OnSourceInitialized(EventArgs e)
{
    base.OnSourceInitialized(e);
    AeroGlass.ExtendGlass(this, new Thickness(-1));
    IntPtr hWnd = new WindowInteropHelper(this).Handle;
    HwndSource.FromHwnd(hWnd).AddHook(new HwndSourceHook(WndProc));
}

```

Do uruchomienia pozostało jeszcze jedno — ponieważ użytkownik może włączyć lub wyłączyć kompozycję pulpitu, należy zadbać o włączenie lub wyłączenie w zależności od sytuacji naszego okna. Można do tego wykorzystać metodę `WndProc`, która przetwarza wszystkie wiadomości wysyłane do okna.

```
private IntPtr WndProc(IntPtr hwnd, int msg, IntPtr wParam, IntPtr lParam, ref bool
    handled)
{
    if (msg == WM_DWMCOMPOSITIONCHANGED)
    {
        AeroGlass.ExtendGlass(this, new Thickness(-1));
        handled = true;
    }
    return IntPtr.Zero;
}
```

`WM_DWMCOMPOSITIONCHANGED` to stała zdefiniowana w klasie, przechowująca kod wiadomości o zmianie okna (`0x031E`), co pokazano poniżej.

```
private const int WM_DWMCOMPOSITIONCHANGED = 0x031E;
```

Spójrzmy więc, jak wygląda „szklany” `TwitterClient` — rysunek 8.12.

Rysunek 8.12.
*TwitterClient z poszerzoną
ramką Aero Glass*



Pytania testowe

1. Klasą, która umożliwi zmianę koloru tła okienka w pasku zadań po najechaniu kursorem myszy na ikonę aplikacji, jest:
 - a) `TaskbarItemInfo`,
 - b) `ThumbButtonInfo`,
 - c) `JumpList`.
2. Zawartość którego z elementów wyświetlanych w oknie podpowiedzi i w pasku zadań może być zdefiniowana jak obrazek?
 - a) nakładka,
 - b) opis,
 - c) zawartość przycisku `ThumbButtonInfo`.
3. Która z poniższych wartości właściwości `ProgressState` wskazuje stan, w którym proces jest wykonywany, lecz nie da się określić stopnia jego zaawansowania?
 - a) `Normal`,
 - b) `None`,
 - c) `Indeterminate`.
4. Wybierz zdanie, które jest prawdziwe dla poniższej definicji elementu `TaskbarItemInfo`:

```
<TaskbarItemInfo ProgressState="Error" ProgressValue="50"/>
```

 - a) Okno programu w pasku zadań zostanie zakolorowane na pomarańczowo, postęp wykonania procesu to 50%.
 - b) Okno programu w pasku zadań zostanie zakolorowane na czerwono, postęp wykonania procesu to 100%.
 - c) Zdefiniowanie wartości 50 spowoduje błąd kompilacji, ponieważ wartość wychodzi poza dopuszczalne granice.
5. Wybierz prawdziwe zdanie.
 - a) `JumpTask` służy do definiowania akcji uruchomienia programu z parametrami przekazywanymi przez linię poleceń.
 - b) `JumpPath` służy do definiowania akcji uruchomienia programu z parametrami przekazywanymi przez linię poleceń.
 - c) Lista `JumpList` jest podłączana raz, po uruchomieniu programu.
6. Wybierz prawdziwe zdanie opisujące poniżej zdefiniowaną listę `JumpList`.

```
<JumpList.JumpList>  
  <JumpList>  
    <JumpTask/>  
  <JumpTask/>
```



```
<JumpTask />
<JumpTask CustomCategory="Programy użytkownika" Title="Testowy program"
    Description="Uruchamia bieżącą aplikację"/>
</JumpList>
</JumpList.JumpList>
```

- a) Do standardowej listy aplikacji zostaną dodane trzy separatory i nazwana grupa zatytułowana „Programy użytkownika”, przechowująca element o nazwie „Testowy program”.
 - b) Do standardowej listy aplikacji zostanie dodana grupa zatytułowana „Programy użytkownika”, przechowująca element o nazwie „Testowy program”, separatory zostaną zignorowane, ponieważ nowa grupa zawiera już swój separator.
 - c) Do standardowej listy aplikacji zostanie dodany jeden separator, następnie zostaną utworzone dwie puste pozycje, po czym zostanie dodana nazwana grupa zatytułowana „Programy użytkownika”, przechowująca element o nazwie „Testowy program”.
7. Wybierz poprawne zdania charakteryzujące klasę `JumpPath`.
- a) Jest elementem wskazującym na ścieżki do plików, które mogą być uruchomione przez daną aplikację.
 - b) Do właściwości `Path` może zostać wpisana jedynie ścieżka bezwzględna do pliku.
 - c) `JumpPath` dziedziczy po `JumpItem`.
8. Aby w liście `JumpList` pokazywane były ostatnio otwierane pliki, należy:
- a) Ustawić właściwość `ShowRecentCategory` klasy `JumpList` na wartość `True`.
 - b) Dodać pliki, które mają być pokazywane, za pomocą metody `AddToRecentCategory` klasy `JumpList`.
 - c) Dodać pliki, które mają być pokazywane, za pomocą metody `AddToRecentCategory` tylko wówczas, gdy ich pokazywanie ma być wymuszone.
9. Aby poszerzyć okno Aero Glass, należy użyć funkcji:
- a) `WndProc`,
 - b) `DwmIsCompositionEnabled`,
 - c) `DwmExtendFrameIntoClientArea`.

Odpowiedzi do pytań

1. a
2. a, c
3. c
4. b

5. a, c

6. b

7. a, b, c

8. a, c

9. c

Skorowidz

A

adres relatywny, 300
Aero Glass, 260
akcja
 EnterActions, 373
 ExitActions, 373
aktualna wartość właściwości, 549
aktualne rozmiary panelu, 436
algorytm
 wyszukiwania zasobu, 304
 skalowania, 206
animacja, 533
 AccelerationRatio, 545
 AutoReverse, 543
 BeginTime, 543
 czas trwania, 545
 DecelerationRatio, 546
 DoubleAnimationUsingKey
 ↳Frames, 565
 DoubleAnimationUsingPath,
 458
 Duration, 540
 From, To, By, 541
 keyframe, 559, 562
 koloru pędzla, 555
 obrotu prostokąta, 555
 odległości, 555
 PathGeometry, 567
 przezroczystości, 383
 RepeatBehavior, 544
 SpeedRatio, 545
 sterowanie przebiegiem,
 550, 552
 StringAnimationUsingKey
 ↳Frames, 564
 w szablonie, 402
 właściwości Width, 542

zdefiniowana w stylu, 557
zdefiniowana w
 wyzwalaczu, 525
zdefiniowane, 538
zmieniająca kolor, 534
zmieniająca szerokość
 przycisku, 540
aplikacja
 mspaint.exe, 257
 TeaPotPlayer, 550
 TwitterClient, 223, 249, 255
aplikacje
 nawigacyjne, 230
 XBAP, 241
aranżacja elementów, 120
architektura WPF, 58
atrybut x:Class, 216, 219
atrybuty
 tagów, 30
 znacznika, 28
autogenerowanie kolumn, 173
automatyczne zaznaczanie
 elementów, 330
autoryzacja, 299

B

biblioteka
 ControlLibrary, 413
 DirectX, 58
 milcore.dll, 58
 PresentationCore.dll, 58
 PresentationFramework.dll,
 58
 Twitterizer, 326
 User32, 58
 WindowsBase.dll, 58
 WindowsCodecs.dll, 58

Bubbling, 269
budowanie aplikacji WPF, 45

C

Canvas, 102
cechy WPF, 20
cel
 animacji, 553
 definiowania walidacji, 347
 polecenia, 288
 wiązania, 318, 321
Code – behind file, 48
czajnik, 514
części, 394
części w klasach, 397, 398
część PART_Track, 395

D

data wysłania wiadomości, 335
definiowanie
 animacji, 458
 animacji keyframe, 560
 figur path, 462
 grupowania, 343
 grupy kształtów, 461
 konwertera, 339
 obiektu GeometryDrawing,
 467
 pędzla gradientowego, 307
 prostokąta, 445
 przezroczystości, 437
 przodu samochodu, 502
 przycisku, 363
 samochodu, 480, 517
 sortowania, 342
 stylów, 406

- definiowanie
 - stylu z szablonem, 390
 - szablonu, 333
 - szablonu kontrolki, 379, 388
 - szklanej powłoki, 380
 - trójkąta, 503
 - twarzy ludzika, 469
 - uchwyty, 317
 - wiązania, 316, 320, 326
 - własnej walidacji, 353
 - właściwości zależnej, 325
 - wyzwalacza EventTrigger, 547
 - zasobu, 304, 309
 - Direct, 269
 - długość wektora, 487
 - DockPanel, 107
 - dodawanie przestrzeni nazw, 335
 - dokumenty, 194, 200
 - dostęp bezpośredni do zasobów, 310
 - drukowanie, 223
 - drzewa
 - logiczne, 61
 - prezentacji, 64
 - dynamiczna aktualizacja
 - wartości właściwości, 40
 - dzbanek, 514, 521
 - działanie wyzwalaczy, 384
 - dziedziczenie wartości
 - właściwości, 74, 364, 387
 - dziennik, 233, 234
- E**
- efekt
 - cienia, 383
 - domyślnego zastosowania szablonu, 384
 - działania skórek, 408
 - odbicia, 440
 - przezroczystości, 439
 - rozbłysku, 380
 - rybiego okna, 558
 - element
 - DataGrid, 171
 - dziecko, 42
 - FlowDocument, 196
 - JumpList, 260
 - JumpPath, 259
 - JumpTask, 255
 - rodzic, 42
 - TaskbarItemInfo, 253
 - TextBlock, 66
 - Title, 256
 - typu content, 42
 - elementy zagnieżdżone, 41
 - elipsa, 450
 - etykieta, 153
- F**
- fabryka obiektów, 308
 - figura PathGeometry, 568
 - filtrowanie, 345
 - formatowanie stringu, 331, 332
 - funkcja BounceEase, 569
 - funkcje ułatwiające, easing functions, 568–571
- G**
- GDI, Graphics Device Interface, 19
 - gesty, 194
 - grafika 2D, 419
 - grafika 3D, 479
 - Grid, 109
 - grubość krawędzi, 449
 - grupowanie, 343
 - elementów, 175
 - transformacji, 500
 - grupy stanów, 400, 401
- H**
- hierarchia klas, 59
 - hiperłącze, 232
- I**
- implementacja
 - metody OnStartup, 220
 - metody Main, 219
 - minijęzyka, 463
 - właściwości zależnych, 69
 - inkrementacja kąta, 128
 - instalacja aplikacji WPF, 243
 - interfejs
 - ICollectionView, 341, 343, 345
 - ICommand, 286
 - INotifyCollectionChange, 326
 - IValueConverter, 337
 - użytkownika, 101
 - interpolacja
 - dyskretna, 563
 - liniowa, 559, 561
 - Spline, 565
- J**
- język
 - XAML, 26, 28
 - XML, 28
- K**
- kamera, 487, 489
 - OrthographicCamera, 490
 - PerspectiveCamera, 491, 492
 - katalog Themes, 411
 - klasa
 - AeroGlass, 261
 - AffineTransform, 493
 - AmbientLight, 482, 506
 - Application, 217
 - ApplicationCommands, 287, 291
 - ArcSegment, 457
 - BeginStoryboard, 536, 548
 - BezierSegment, 457
 - Binding, 318, 353
 - BitmapImage, 300
 - BitmapCache, 441
 - BitmapImage, 206
 - Brush, 419
 - Button, 143, 147
 - Camera, 485
 - Canvas, 101
 - CircularPanel, 128
 - ColorAnimation, 535
 - CombinedGeometry, 459, 460
 - ComponentCommands, 287
 - CompositeCollection, 356
 - ContentPresenter, 65–68
 - Country, 164
 - CustomDialogBox, 225
 - DataTemplate, 373
 - DataTrigger, 371
 - DiffuseMaterial, 512
 - DirectionalLight, 508
 - DockPanel, 101
 - Drawing, 464
 - DrawingBrush, 430, 466
 - DrawingImage, 466
 - DrawingVisual, 466
 - EditingCommands, 287

- Ellipse, 450
- EmissiveMaterial, 514
- Employee, 167
- Employees, 374
- EventManager, 271
- FlowDocument, 195
- Frame, 233
- FrameworkElement, 85, 100, 119, 120, 191
- FrameworkPropertMetadata, 71
- Geometry, 445, 454
- GeometryDrawing, 465–467
- GeometryGroup, 459
- GeometryModel3D, 511
- GradientStop, 426
- Grid, 101
- GridSplitter, 116
- GroupStyle, 176
- HeaderedContentControl, 157
- Hyperlink, 232
- Image, 205
- ImageSource, 300
- Inline, 198
- InlineUIContainer, 199
- IsolatedStorageFile, 226
- IsolatedStorageFileStream, 226
- JumpItem, 258
- JumpList, 255
- KeyEventArgs, 281
- KeySpline, 565
- Label, 332
- Light, 506
- Line, 451
- LineBreak, 199
- LineSegment, 456
- LogicalTreeHelper, 63
- MainWindow, 262
- ManipulationDeltaEventArgs, 284
- MarkupExtension, 39
- Material, 511
- MaterialGroup, 511
- MatrixTransform, 493
- MediaCommands, 287
- MeshGeometry3D, 502
- Model3D, 501
- Model3DGroup, 516
- ModelVisual3D, 518
- MouseEventArgs, 278
- MyButton, 272
- MyCollection, 328
- NavigationCommands, 287
- NavigationService, 231
- NavigationWindow, 233
- Okno, 47
- Page, 230
- PageFunction, 238, 239
- Panel, 84
- Paragraph, 195
- Path, 452, 462
- PathFigureCollection, 463
- PathGeometry, 459
- Pen, 468
- PinDialogBox, 352
- PointLight, 509
- PolyBezierSegment, 457
- Polygon, 452
- Polyline, 451
- PolyLineSegment, 456
- PolyQuadraticBezierSegment, 457
- Popup, 156
- PrintDialog, 223
- QuadraticBezierSegment, 457
- Rectangle, 449
- RenderTargetBitmap, 441
- ResourceDictionary, 303
- RotateTransform3D, 498
- RoutedPropertyChangesEvent
 - ↳ Args, 317
- RoutedUICommand, 287
- SaveFileDialog, 223
- ScaleTransform3D, 496
- Selector, 159
- Shape, 445, 447
- SpecularMaterial, 515
- SpotLight, 510
- StackPanel, 101
- Storyboard, 536, 553
- StreamGeometry, 463
- Style, 364
- System.DispatcherObject, 59
- System.Object, 59
- System.Windows.Controls.
 - Panel, 99
- System.Windows.Conent
 - ↳ Element, 61
- System.Windows.Controls.
 - ContentControl, 145
- System.Windows.Controls.
 - Control, 60
- System.Windows.Controls.
 - ScrollViewer, 135
- System.Windows.Controls.
 - Viewbox, 136
- System.Windows.Data.
 - Binding, 316
- System.Windows.Dependency
 - ↳ Object, 59
- System.Windows.Expression, 78
- System.Windows.Framework
 - ↳ ContentElement, 61
- System.Windows.Framework
 - ↳ Element, 60
- System.Windows.Freezable, 60
- System.Windows.IUElement, 60
- System.Windows.Markup.
 - MarkupExtention, 39
- System.Windows.Media.
 - Media3D.Visual3D, 61
- System.Windows.Media.
 - Visual, 60
- System.Windows.Style, 362
- System.Windows.UIElement
 - ↳ 3D, 61
- SystemColors, 409
- TabPanel, 101
- TemplateBinding, 385
- TemplateBindingExtension, 385
- TextElements, 195
- TileBrush, 427
- Timeline, 552
- ToolBarOverflowPanel, 101
- ToolTipService, 155
- Transform3DGroup, 500
- TranslateTransform3D, 495
- Trigger, 370
- Tweet, 322
- UIElement, 66, 79, 85
- UIElement3D, 521
- UniformGrid, 101
- Uri, 300
- View, 341
- Viewport2DVisual3D, 519
- Viewport3D, 483, 524
- VirtualizingPanel, 101
- VirtualizingStackPanel, 100, 101
- Visual3D, 517
- VisualStateManager, 401
- VisualTreeHelper, 65
- Window, 214
- WindowsFormsApplication
 - ↳ Base, 257
- WrapPanel, 101
- XamlReader, 46
- XamlWriter, 46

- klasy częściowe, partial class, 47
 - klasy dziedziczące
 - po Geometry, 446
 - po Light, 506
 - po Model3D, 501
 - po Shape, 446
 - po UIElement3D, 521
 - po Visual3D, 517
 - klasy implementujące animacje, 537, 539
 - klasy systemowe, 312
 - klasy typu
 - AnchoredBlock, 199
 - Inline, 198
 - Span, 198
 - klasyfikacja paneli, 99
 - kod proceduralny, Code – behind file, 46
 - kolejność zdarzeń, 284
 - kolekcja, 43
 - akcji, 373
 - SortDescriptions, 342
 - TriggerActionCollection, 547
 - komentarze w XAML, 29
 - kompilator
 - csc.exe, 47
 - xamlc.exe, 47
 - kompilowanie aplikacji WPF, 214
 - kompozycja pulpitu, Desktop composition, 261
 - kontener, 157
 - Frame, 230
 - NavigationWindow, 230
 - kontenery z nagłówkiem, 157
 - kontrolka, 143
 - Button, 147, 286
 - Calendar, 201
 - CheckBox, 150, 286
 - ComboBox, 161
 - ContextMenu, 179
 - DataGrid, 170–175, 374
 - DatePicker, 204
 - Expander, 158, 176
 - FlowDocument, 197
 - Frame, 157
 - GroupBox, 157
 - Image, 205
 - InkCanvas, 191, 193
 - Label, 153
 - ListBox, 166
 - ListBox w 3D, 520
 - ListView, 167, 169
 - Menu, 179
 - MenuItem, 286
 - OpenFileDialog, 227
 - PasswordBox, 190
 - Popup, 156
 - ProgressBar, 184
 - RadioButton, 151
 - RepeatButton, 147
 - RichTextBox, 189
 - ScrollBar, 393
 - Slider, 185, 316
 - StatusBar, 183
 - TabControl, 177, 178
 - TextBlock, 187
 - TextBox, 188
 - Thumb, 395
 - ToggleButton, 148
 - ToolBar, 181
 - ToolTip, 154, 156
 - TreeView, 180, 405
 - kontrolki
 - informacyjne, 153
 - Ink, 183
 - Menu, 178
 - tekstowe, 185
 - typu Content, 144, 145
 - typu Items, 159
 - typu selektor, 160
 - z datami, 201
 - zakresu, 183
 - konwersja stringu, 331
 - konwertery, 336
 - konwertery typów, 37
 - korzeń dokumentu XAML, 28
 - kreator
 - ClickOnes, 244
 - Windows Installer, 244
 - kształty, 445
 - połączone, 459
 - proste, 454
 - złożone, 456
- L**
- licznik
 - kliknięć, 280
 - kolekcji, 325
 - znaków, 320, 332
 - linia, 451
 - lista, 43
 - przeźrzeni WPF, 33
 - szybkiego dostępu, 254–259
- Ł**
- łączenie
 - materiałów, 515
 - właściwości z zasobem, 311
 - zdarzeń, 275
- M**
- mapowanie
 - nazw XAML, 32
 - przeźrzeni nazw, 34
 - maska przezroczystości, 438, 439
 - materiał, 502, 511
 - DiffuseMaterial, 512, 515
 - SpecularMaterial, 516
 - mechanizm PInvoke, 261
 - menedżer stanów, 399, 401
 - metadane, 70
 - metoda
 - AddHandler, 268, 271
 - ArrangeOverride, 85, 120, 122
 - CanExecute, 286
 - CommonClickHandler, 270
 - Convert, 337
 - ConvertBack, 337, 338
 - DependencyPropertyHelper.
 - ↪ GetValueSource, 77
 - DwmExtendFrameIntoClient
 - ↪ Area, 261
 - Execute, 286
 - FindResource, 310
 - GetDefaultView, 342
 - GetIntermediateTouchPoints, 283
 - GetPointFromEllipse, 128
 - GetPosition, 279
 - GetTouchPoint, 283
 - GetValue, 70
 - InitializeComponent, 47, 216
 - Load, 46
 - LoadAsync, 46
 - LoadTweets, 327
 - Main, 219
 - MeasureOverride, 84, 119, 123
 - myButton_Click, 71
 - Navigate, 231, 236, 237
 - OnClosed, 227
 - OnInitialized, 227
 - OnSourceInitializes, 262
 - ProvideValue, 39

Refresh, 231
 RegisterRoutedEvent, 271
 RemoveHandler, 268, 271
 SetBinding, 318
 SetResourceReference, 311
 SetValue, 70
 ShowDialog, 224
 Siteoforigin, 299
 TeaPot_MouseEnter, 522
 TeaPot_MouseLeave, 522
 ToString, 66, 330
 TryFindResource, 310
 WndProc, 263

metody

obsługi zdarzeń, 239
 odwoływania się do zasobów, 310

Microsoft Expression Blend 4, 16

minijęzyk, 464

Model3D, 501

motyw domyślny, 412

motywy, 409

N

nakładka, 253

nawigacja, 231, 234

nazwa zasobu, 305

notacja klamrowa, 39

O

obiekt

ContentPresenter, 388
 Drawing, 206
 GeometryDrawing, 467
 ItemsPresenter, 176
 klasy Application, 218
 klasy Brush, 420
 klasy Window, 216, 218
 Model3DGroup, 482
 reagujący na zdarzenia, 523
 SortDescription, 342
 TwitterViewModel, 324
 Viewbox, 381
 Window, 214

obiekty

DiscretePointKeyFrame, 564
 dziedziczące po Geometry, 454
 KeyFrames, 564
 SplineDoubleKeyFrame, 565

TabItem, 178
 typu DragEventArgs, 281
 obliczanie ostatecznej wartości, 77
 obrót, 493, 498
 kadru, 488
 kamery, 488
 obsługa
 efektów wizualnych, 398
 filtra, 345
 Gesture, 194
 przepełnienia zawartości, 135
 strony PageFunction, 241
 wartości null, 237
 obszar przezroczysty, 260
 odwołania do zasobów binarnych, 298
 ograniczenia definicji klasy, 35
 okna
 dialogowe, 221
 niestandardowe, 228
 okno
 Aero Glass, 261
 aplikacji WPF, 215
 MessageBox, 222, 270
 OpenFileDialog, 223
 opakowanie animacji, 548
 oświetlenie, 506

P

panel, 83, 99
 Canvas, 102, 447
 CircularPanel, 131
 DockPanel, 107, 483
 Grid, 109, 113, 306
 SelectiveScrollingGrid, 134
 StackPanel, 104, 174, 435, 444
 TabPanel, 132
 ToolBarOverflowPanel, 132
 UniformGrid, 133
 VirtualizingStackPanel, 133
 WrapPanel, 105
 parametr availableSize, 120
 parametry
 nazwane, 40
 pozycyjne, 40
 parser XAML, 32
 pasek
 postępu, 184
 przewijania, 395
 zadań, 249, 250, 253

perspektywa, 491
 pędzel, 419
 BitmapCacheBrush, 441, 443
 DrawingBrush, 421, 428, 431
 gradientowy, 301
 ImageBrush, 421, 432, 513
 LinearGradientBrush, 421, 423, 438
 RadialGradientBrush, 421, 425, 438
 SolidColorBrush, 405, 420–422, 514
 VisualBrush, 421, 433, 435
 pędzle
 malujące kolorem, 421
 pokrywające, 427
 Pinvoke, Platform Invocation Service, 261
 plik
 Aero.NormalColor.xaml, 412
 App.xaml, 219
 ButtonTemplate.xaml, 405
 Classic.xaml, 412
 Common.xaml, 406
 ContentImage.png, 296
 Generic.xaml, 412
 ResourceTest.exe, 296
 ResourceTest.g.resources, 296
 Skin1.xaml, 406
 SystemColorSkin.xaml, 409
 pliki
 .baml, 47
 .g.cs, 47
 .xaml, 15
 przechowujące izolowane, 226
 podpisywanie pliku .xbap, 242
 podział animacji, 539
 polecenia, 285, 287
 minijęzyka, 464
 wbudowane, 286
 polecenie
 Help, 289, 291
 NotACommand, 291
 powiadamianie o zmianie, change notification, 72
 pozycja
 kamery, 486
 kursora, 279
 prezentacja
 danych, 329
 dokumentów, 200
 panelu CirclePanel, 113

program

- Kaxaml, 26
- mage.exe, 243
- mageUI.exe, 243
- Microsoft Expression Blend, 26, 398
- Microsoft Visual Studio, 26
- XAML Cruncher, 26
- XAMLPAID 2009, 26
- XamlPadX, 26
- ZAM3D, 514

propagacja zdarzenia

- MouseLeftButtonDown, 271

prostokąt, 449

protokół OAuth, 347

przebieg animacji, 540

przechowalnia, store, 226

przechowywanie danych, 228

przeciąganie elementów, 281

przekazywanie danych

PageFunction, 238

właściwości aplikacji, 238

za pomocą parametrów, 236

przekazywanie informacji

między stronami, 238

przekazywanie parametrów do konstruktora, 30

przekierowanie do strony, 231

przekształcanie dat, 340

zapełnienie zawartości, 134

przepływ danych, 354

przestrzeń abstrakcyjna, 226

przestrzeń nazw, 31, 33

System.Windows.Media.

Animation, 538

WPF, 33

przesunięcie, 495

przetwarzanie komunikatów,

218

przewijanie, 135, 235

przezroczystość pędzli, 426, 438

prycinanie elementów, 134

przycisk, 146

Button, 147

MyButton, 413

RadioButton, 151

RepeatButton, 147

ToggleButton, 148

punkt widzenia, 485

punkty kontrolne, 565

R

rejestrowanie

właściwości, 71

zdarzeń, 271

renderowanie danych, 336

reprezentacja pojedynczej

wiadomości, 322

rodzaje kamer, 489

rodzaje właściwości, 67

rozmiar

automatyczny kolumn, 114

bezwzględny kolumn, 114

proporcjonalny kolumn, 115

rozmięszczenie elementów,

103–109

rozpoznawanie gestów, 194

rozproszenie, 509

rozszerzenia

wbudowane, 41

XAML, 39

znaczników, 38, 302

znaczników WPF, 41

rozszerzenie

DynamicResource, 306

StaticExtension, 40

StaticResource, 303, 306

x:Binding, 321

x:Reference, 321

x:Static, 38

rysowanie

krawędzi, 449

linii, 468

S

samochód, 480

scena 3D, 483

schemat

application, 299

definiowania szablonu

danych, 334

przebiegu dziedziczenia, 76

SelectiveScrollingGrid, 134

selektory, 159

setter, 365, 375, 393

skalowanie, 136, 206, 496

składniki szablonu kontrolki,

394

skórka domyślna, 408

skórki, 404

słowa kluczowe XAML, 50–52

słownik ResourceDictionary,

405

słowniki, 44

słowo kluczowe

Component, 298

Key, 302

Name, 46

Siteoforigin, 298

x:Class, 48

sortowanie, 175, 341

splash screen, 221

StackPanel, 104

stan

aplikacji, 226

klawisza klawiatury, 281

MouseOver, 402

stany

przycisków, 235

przycisku ToggleButton, 150

w grupie, 400, 401

sterowanie przebiegiem

animacji, 549, 552

strona, 230

strona główna aplikacji, 230

struktura

Matrix3D, 493

uchwyty, 269

styl, 362, 365

buttonStyle, 365

extendedButtonStyle, 365

style

ograniczanie użycia, 367

uogólnienie użycia, 365

współdzielone, 365

ze zdefiniowanym

uogólnieniem, 367

właściwość, 404

system graficzny, 19

szablon

ControlTemplate, 378

Thumb, 395

szablony

danych, 332

dla przycisku, 381

kontrolki, 377, 388

przycisków RepeatButton,

394

złożone, 391

szklana powłoka, 380

szklany TwitterClient, 263

Ś

ścieżka

animacji, 563, 566

bezwzględna do pliku, 259

- środowisko
 - .NET 4.0, 15
 - Visual Studio, 15
- światło
 - AmbientLight, 507
 - DirectionalLight, 508
 - PointLight, 509, 510
 - SpotLight, 511
- T**
- TabPanel, 132
- tag xmlns, 32
- tagi
 - XAML, 30
 - XML, 31
- właściwość, 394
- tło elementu ListBox, 303
- ToolBarOverflowPanel, 132
- transformacja
 - 3D, 492
 - AxisAngleRotation3D, 550
 - elementów, 91
 - LayoutTransform, 91, 437
 - MatrixTransform, 95
 - obrót, 493
 - RenderTransform, 91, 437
 - RotateTransform, 92
 - RotateTransform3D, 499
 - ScaleTransform, 94
 - ScaleTransform3D, 497
 - skalowanie, 493
 - SkewTransform, 94
 - translacja, 493
 - TranslateTransform, 94, 458
 - TranslateTransform3D, 493
- translacja, 493, 495
- trójkąt, 504
- tryb
 - bąbelkowy, 269
 - bezpośredni, 269
 - tunelowy, 268
 - wiązania danych, 355
- Tunneling, 268
- twarz ludzika, 447, 469
- TwitterClient
 - z poszerzoną ramką, 263
 - z wiązaniem właściwości, 327
- tworzenie
 - dokumentów, 195
 - kolumn, 116
 - panelu, 121
 - panelu kołowego, 125
 - własnych paneli, 118
- typy
 - generyczne, 539
 - zasobów binarnych, 296
- U**
- uchwyty, 239
 - do zdarzeń, 270, 273
 - przycisków, 235
- układ
 - elementów, 84
 - graficzny, layout, 83, 100
 - współrzędnych 3D, 484
- umieszczenie kamery, 485, 487
- umieszczanie
 - animacji w stylu, 556
 - zawartości w 3D, 519
- UniformGrid, 133
- uogólnienie dla zmienianych
 - właściwości, 366
- ustalanie wartości właściwości
 - zależnej, 78
- ustawienia pędzla
 - BitmapCacheBrush, 443
- użycie
 - dziennika, 234
 - konwertera, 338
 - rozszerzenia, 38
 - URI w kodzie
 - proceduralnym, 300
- V**
- VirtualizingStackPanel, 133
- Visual State Manager, 398
- Visual Studio Ultimate 2010, 16
- W**
- walidacja
 - DataErrorValidationRule,
 - 350–352
 - danych, 347
 - ExceptionValidationRule,
 - 348
 - własna, 354
 - wartości właściwości, 35
 - wartość ostateczna właściwości,
 - 77
 - warunek AND, 376
 - warunek OR, 376
 - wbudowane mechanizmy
 - walidacji, 348
 - wektor normalny, 505
- wersje WPF, 22
- wiązanie danych, 40, 165, 315
 - CompositeCollection, 356
 - MultiBinding, 356
 - PriorityBinding, 357
 - w kodzie C#, 320
 - w kodzie XAML, 320
- wiązanie polecenia, 288
- wiązanie uchwytów dla zdarzeń
 - z poleceniem, 290
 - wiązanie z kolekcją, 326, 329
 - wiązanie z właściwościami
 - .NET, 322
 - wiązanie z właściwością
 - zależną, 323, 324
 - wiązanie za pomocą
 - TemplateBinding, 386
- widok, View, 341
- wielkość elementów WPF, 85
- Windows Installer, 243
- Windows SDK, 243
- własne okno dialogowe, 224, 226
- właściwość
 - .NET, 322
 - docelowe animacji, 554
 - dołączone, attached
 - properties, 78, 80
 - elementów, 35
 - klasy ContentPresenter, 66
 - klasy KeyEventArgs, 281
 - klasy ManipulationDelta,
 - 284
 - klasy MenuItem, 179
 - klasy MeshGeometry3D, 503
 - zależne, dependency
 - properties, 68, 73, 251, 324
 - zwykłe .NET, 324
- właściwość
 - AccelerationRatio, 559
 - Action, 281
 - ActualHeight, 86
 - ActualWidth, 86
 - AllowDrop, 281
 - AllowsTransparency, 229
 - AutoGenerateColumns, 172
 - AutoReverse, 543
 - Background, 100, 123, 555
 - BackMaterial, 502
 - BeginTime, 543
 - BitmapScallingMode, 205
 - By, 543
 - Canvas, 569
 - ChangedButton, 280
 - CircleBeginning, 112, 125

- właściwość
 - CircleTotal, 112, 125
 - ClipToBounds, 135
 - ColumnDefinitions, 110
 - ColumnSpan, 113
 - Command, 288
 - CommandBinding, 290
 - Content, 326, 387
 - Count, 72, 326
 - CustomCategory, 258, 259
 - DecelerationRatio, 559
 - DesiredSize, 86
 - Display, 186
 - DisplayMemberParh, 329
 - Dock, 108
 - Duration, 540
 - ElementName, 321
 - EnableClearType, 443
 - EnterActions, 559
 - EscapePressed, 281
 - ExitActions, 559
 - Fill, 386, 449
 - Filter, 345
 - FirstDependencyProperty, 70
 - FlowDirection, 90, 104
 - From, 541
 - FrozenColumnCount, 175
 - Grid.Column, 110
 - Grid.Row, 110
 - GridResizeDirection, 116
 - GroupName, 153
 - Header, 157
 - Height, 85
 - HirizontalContentAlignment, 90
 - HorizontalAlignment, 89, 123
 - InternalChildren, 100
 - IsEditable, 161, 162
 - IsHitTestVisible, 278
 - IsReadOnly, 161
 - ItemHeight, 106
 - ItemSource, 168
 - ItemsPanel, 166
 - ItemsSource, 159
 - ItemWidth, 106
 - JournalEntry.KeepAlive, 234
 - JournalOwnership, 233
 - JumpItems, 255
 - KeyStates, 281
 - LastChildFill, 108
 - LookDirection, 486, 487
 - Margin, 86
 - MaxHeight, 85
 - MaxLength, 320
 - MaxWidth, 85
 - Member, 39
 - MergedDictionaries, 303, 406
 - MinHeight, 85
 - MinWidth, 85
 - Mode, 355
 - MouseButtonState, 278
 - NavigateUri, 232
 - Opacity, 383
 - Orientation, 104, 106
 - Padding, 87
 - Panel.ZIndex, 100
 - PathGeometry, 567
 - Position, 486
 - ProgressState, 253
 - ProgressValue, 253
 - Properties, 220
 - RemoveFromJournal, 234
 - RenderAtScale, 442
 - RenderSize, 86
 - RepeatBehavior, 544
 - Resource, 302
 - RowDetailsTemplate, 173
 - RowSpan, 113
 - SelectedDate, 203
 - SelectedDates, 203
 - SelectionMode, 166
 - SharedSizeGroup, 117
 - ShowGridLines, 111, 114
 - ShowNavigationUI, 242
 - SnapsToDevicePixels, 442
 - SortDescriptions, 342
 - Source, 321
 - StrechDirection, 136
 - Stretch, 136
 - StretchDirection, 137
 - StringFormat, 331
 - Stroke, 449
 - StrokeThickness, 449
 - TextSearch, 164
 - TextSearch.Text, 163, 165
 - TextSearch.TextPath, 163, 165
 - ThumbnailClipMargin, 252
 - TimeLine, 251
 - To, 541
 - ToolTipService.Placement, 155
 - TriangleIndices, 505
 - Triggers, 382
 - Tweets.Count, 326
 - UpdateSourceTrigger, 353, 355
 - UpDirection, 488, 489
 - UserId, 220
 - ValidationRules, 353
 - VerticalAlignment, 89
 - VerticalContentAlignment, 90
 - ViewboxUnits, 432
 - ViewportUnits, 432
 - Visibility, 88, 278
 - Width, 85
 - WPF, Windows Presentation Foundation, 19, 57, 419
 - WPF 3.5, 23
 - WPF 4, 23
 - WrapPanel, 105
 - wrapper, 268, 272
 - wyjątek
 - InvalidOperationException, 120
 - wyłączanie domyślnych stylów, 411
 - wysokość przycisku, 40
 - wyszukiwanie słowników motywów, 412
 - wyświetlanie
 - czcionki, 186
 - informacji o błędzie, 348
 - wiązanych danych, 330
 - tekstu, 187, 188
 - wyzwalacze, triggers, 369
 - Data triggers, 370, 371, 374
 - Event triggers, 370, 536, 547
 - Property triggers, 72, 370
 - stosowane w szablonach, 382
 - warunki logiczne, 375
 - wzór
 - czas trwania animacji, 545
 - na kolor, 513
 - na rozproszenie, 509
 - obwód elipsy, 126
 - punkt na elipsie, 127
- X**
- XAML, Extensible Application Markup Language, 25
 - nieskompilowany, 46
 - skompilowany, 46
 - XAML 2009, 49
 - XBAP, XAML Browser Application, 241

Z

- zagnieżdżanie elementów WPF, 42
- założenia animacji, 535
- Zam3D, 16
- zamrażanie kolumn, 175
- zaokrąglanie rogów prostokąta, 319
- zapis danych do ustawień aplikacji, 228
- zasada prawej ręki, 487
- zasięg zdefiniowanego zasobu, 304
- zasoby
 - binarne, 295
 - dynamiczne, 306, 522
 - logiczne, 301–303, 310
 - słownikowe, 412
 - statyczne, 306
 - systemowe, 311
 - w kodzie proceduralnym, 299
 - w kodzie XAML, 297
- zasób `textBlockBackground`, 306
- zbiór linii, 451
- zdarzenia
 - bąbelkowe, 275
 - dołączone, 267
 - dotyku, 283, 285
 - generowane przez mysz, 523
 - klawiatury, 281
 - manipulacji, 283, 285
 - myszy, 278, 280
 - `RoutedEvent`, 267, 271, 275
 - routowalne, 267, 268
 - tunelowe, 275
 - wejściowe, 267, 278
- zdarzenie
 - `Activated`, 216
 - `AutoGeneratingColumn`, 172
 - `CanExecuteChanged`, 286
 - `Click`, 240
 - `DropDownClosed`, 161
 - `DropDownOpened`, 161
 - `Gesture`, 193
 - `Initialized`, 277
 - `KeyDown`, 275
 - kliknięcie hiperłącza, 240
 - `Loaded`, 277
 - `ManipulationDelta`, 284
 - `MouseLeftButtonDown`, 270
 - `MouseMove`, 279
 - `MouseWheel`, 279
 - `MyMouseEnter`, 274
 - `MyMouseLeave`, 274
 - `PreviewKeyDown`, 275
 - `PreviewMouseMove`, 275
 - `PreviewMouseWheel`, 279
 - `PreviewTextInput`, 276
 - `PropertyChanged`, 325
 - `RoutedEvent`, 268
 - `Unloaded`, 277
 - `WłaściwośćChanged`, 325
- zmiana
 - kolorów, 537
 - motywu systemu, 410
 - nakładki, 254
 - rozmiarów okna, 103
 - skórki, 404
 - wyglądu przycisków, 364
- znaki specjalne XAML, 29
- związane elementy, 319

Ż

- źródło
 - polecenia, 287
 - wiązania, 318, 321

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>

Windows Presentation Foundation to nowoczesny system graficzny, umożliwiający tworzenie i wyświetlanie atrakcyjnych wizualnie aplikacji okienkowych dla środowiska Windows. Technologia ta powstała w odpowiedzi na stale rosnącą potrzebę dostarczania wysokiej jakości grafiki, której prezentacja opiera się na wykorzystaniu akceleracji sprzętowej i nie wiąże się z dużym obciążeniem zasobów systemowych. Oddzielenie warstwy interfejsu użytkownika od logiki aplikacji umożliwia definiowanie interfejsów przy użyciu deklarytywnego języka XAML. Upraszcza tym samym proces tworzenia przejrzystych i elastycznych GUI. Każdy szanujący się autor programów działających pod kontrolą systemów Windows powinien dobrze opanować WPF, zwłaszcza jeśli ma na celu produkowanie naprawdę interesujących aplikacji okienkowych przy możliwie niewielkim nakładzie sił i środków. Pomoże mu w tym odpowiednie źródło wiedzy takie jak książka *Tworzenie nowoczesnych aplikacji graficznych w WPF*. W ciekawy i przystępny sposób zostały w niej zaprezentowane najważniejsze informacje na temat korzystania z Windows Presentation Foundation, tworzenia nowoczesnych interfejsów użytkownika, zastosowania języka XAML, a także wyświetlania grafiki i animacji oraz obsługi zdarzeń związanych z używaniem różnego rodzaju urządzeń wskazujących.

- Zasady tworzenia graficznych interfejsów użytkownika
- Charakterystyka platformy WPF i języka XAML
- Struktura i typy aplikacji WPF oraz metody ich tworzenia
- Używanie paneli i zarządzanie układem ich składników
- Korzystanie z różnego rodzaju kontrolki
- Definiowanie interfejsów aplikacji systemu Windows 7
- Obsługa zdarzeń pochodzących z urządzeń wejściowych
- Używanie zasobów i wiązanie danych
- Stosowania stylów, szablonów, skórek i motywów
- Prezentacja grafiki 2D, 3D i animacji

A wszystko to z ilustrowane wieloma praktycznymi i ciekawymi przykładami!

Sięgnij po jedną z nielicznych książek poświęconych technologii WPF i twórz nowoczesne aplikacje graficzne!

Nr katalogowy: 7932



Księgarnia internetowa
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje:

🔗 <http://helion.pl/promocje>

Książki najchętniej czytane:

🔗 <http://helion.pl/miastca/lezy>

Zamów informacje o nowościach:

🔗 <http://helion.pl/novosci>

Helion SA

ul. Kościuski 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

helion.pl
księgarnia
internetowa

Cena 89,00 zł

ISBN 978-83-246-3500-9



9 788324 635009

Informatyka w najlepszym wydaniu