



Jacek Matulewski

# Visual Studio 2017

Tworzenie aplikacji Windows w języku C#

Helion 

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.

Opieka redakcyjna: Ewelina Burska  
Projekt okładki: Studio Gravite/Olsztyn  
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION  
ul. Kościuszki 1c, 44-100 GLIWICE  
tel. 32 231 22 19, 32 230 98 63  
e-mail: [helion@helion.pl](mailto: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/vs17za>  
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-3825-8

Copyright © Helion 2018

Printed in Poland.

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

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

# Spis treści

|   |           |
|---|-----------|
| <b>Wstęp .....</b>  | <b>15</b> |
| <b>Część I. Projektowanie aplikacji WPF sterowanych zdarzeniami .....</b> | <b>17</b> |
| <b>Rozdział 1. Szybki start. Pierwsza aplikacja WPF .....</b>             | <b>19</b> |
| Wzorzec widoku autonomicznego .....                                       | 19        |
| Tworzenie projektu .....  | 20        |
| Projektowanie interfejsu .....  | 21        |
| Kilka uwag na temat kodu XAML opisującego interfejs okna .....            | 25        |
| Zdarzenia .....   | 26        |
| Własności .....   | 32        |
| Zapisywanie i odtwarzanie stanu aplikacji .....                           | 33        |
| Zadania .....   | 37        |
| <b>Rozdział 2. Notatnik. Przegląd komponentów WPF .....</b>               | <b>39</b> |
| Projektowanie interfejsu aplikacji i menu główne .....                    | 39        |
| Tworzenie projektu aplikacji i jej interfejsu .....                       | 40        |
| Zmiana nazwy okna .....   | 42        |
| Zmiana ikony okna i aplikacji .....                                       | 42        |
| Zgodność projektu z wersjami platformy .NET .....                         | 43        |
| Pasek stanu .....   | 43        |
| Menu główne aplikacji .....   | 44        |
| Okna dialogowe i pliki tekstowe .....                                     | 45        |
| Wybór pliku za pomocą okna dialogowego .....                              | 46        |
| Zapisywanie tekstu do pliku .....   | 49        |
| Potwierdzenie zamknięcia aplikacji .....                                  | 50        |
| Czyszczenie zawartości pola tekstowego .....                              | 54        |
| Menu Edycja .....   | 55        |

|   |    |
|---|----|
| Menu Widok .....                                      | 56 |
| Pozycje menu z możliwością zaznaczania .....          | 56 |
| Kolor tła. Współpraca z obiektami Windows Forms ..... | 58 |
| Czcionki .....  | 61 |
| Drukowanie .....                                      | 65 |
| Obsługa klawiszy skrótów .....                        | 67 |
| Ikony menu .....                                      | 68 |
| Pasek narzędzi .....                                  | 71 |
| Wstążka .....   | 73 |
| Menu aplikacji .....                                  | 73 |
| Pasek szybkiego dostępu .....                         | 76 |
| Zakładki .....  | 77 |
| Lokalizacja .....                                     | 80 |
| Zadania .....   | 84 |

### **Rozdział 3. Zdarzenia trasowane (routed events)**

#### **i polecenia trasowane (routed commands) ..... 87**

|   |    |
|---|----|
| Pojedyncza kontrolka .....                            | 87 |
| Zagnieżdżanie przycisków .....                        | 89 |
| Kontrola przepływu zdarzeń trasowanych .....          | 90 |
| Przerwanie serii .....                                | 92 |
| Bulgotanie (bubbling) i tunelowanie (tunneling) ..... | 93 |
| Dynamiczne tworzenie przycisków zagnieżdżonych .....  | 94 |
| Polecenia trasowane .....                             | 95 |

### **Rozdział 4. Przeciągnij i upuść (drag & drop) ..... 103**

|   |     |
|---|-----|
| Interfejs przykładowej aplikacji .....                  | 104 |
| Rozszerzanie kontrolki ListBox .....                    | 105 |
| Inicjacja procesu przeciągania i przenoszone dane ..... | 107 |
| Akceptacja upuszczenia elementu .....                   | 109 |
| Reakcja na upuszczenie elementu .....                   | 110 |
| Przenoszenie elementów między aplikacjami .....         | 111 |
| Opóźnione inicjowanie procesu przenoszenia .....        | 114 |
| Przenoszenie wielu elementów .....                      | 115 |
| Zadania .....   | 118 |

---

|  |            |
|--|------------|
| <b>Rozdział 5. Choinka. Zabawa w WPF .....</b>               | <b>119</b> |
| Okno o dowolnym kształcie .....                              | 119        |
| Przesuwanie okna .....                                       | 121        |
| Zamykanie okna .....   | 122        |
| Splash screen .....  | 124        |
| Ikony w obszarze powiadomiania .....                         | 125        |
| Odtwarzanie pliku dźwiękowego .....                          | 128        |
| Zadania .....  | 129        |
| <br>   |            |
| <b>Rozdział 6. Gra Reversi. Model i widok .....</b>          | <b>131</b> |
| Model — silnik gry .....                                     | 132        |
| Stan planszy .....   | 133        |
| Konstruktor klasy .....                                      | 134        |
| Implementacja zasad gry .....                                | 134        |
| Obliczanie liczb pól zajętych przez graczy .....             | 137        |
| Testy jednostkowe .....                                      | 138        |
| Widok .....  | 142        |
| Graficzna prezentacja planszy .....                          | 142        |
| Interakcja z użytkownikiem .....                             | 146        |
| Historia ruchów .....  | 147        |
| Wykrywanie szczególnych sytuacji w grze .....                | 148        |
| Komputer gra w Reversi .....                                 | 153        |
| Rozbudowa silnika .....                                      | 153        |
| Jak znaleźć najlepszy ruch? .....                            | 154        |
| Gra z komputerem .....                                       | 160        |
| Menu .....   | 161        |
| Zadania .....  | 164        |
| Zadania zaawansowane .....                                   | 164        |
| <br>   |            |
| <b>Rozdział 7. Separacja modułów .....</b>                   | <b>167</b> |
| Kontrolka prezentująca planszę .....                         | 168        |
| Interfejs to kontrakt .....                                  | 178        |
| Biblioteka .....   | 180        |
| Zadania .....  | 182        |
| <br>   |            |
| <b>Rozdział 8. Przechowywanie danych w plikach XML .....</b> | <b>183</b> |
| Podstawy języka XML .....                                    | 183        |
| Deklaracja .....   | 183        |
| Elementy .....   | 184        |

|  |            |
|--|------------|
| Atrybuty .....   | 184        |
| Komentarze .....   | 184        |
| LINQ to XML .....  | 185        |
| Tworzenie pliku XML za pomocą klas XDocument i XElement .....    | 185        |
| Pobieranie wartości z elementów o znanej pozycji w drzewie ..... | 188        |
| Odwzorowanie struktury pliku XML w kontrolce TreeView .....      | 190        |
| Zapisywanie danych do obiektów. Kursy walut NBP .....            | 193        |
| Zapisywanie danych z kolekcji do pliku XML .....                 | 196        |
| Zadania .....  | 197        |
| <b>Rozdział 9. Multimedia .....</b>                              | <b>199</b> |
| Odtwarzanie wideo .....  | 199        |
| Synteza mowy .....   | 203        |
| Zadanie .....  | 205        |
| <b>Część II. XAML .....</b>                                      | <b>207</b> |
| <b>Rozdział 10. Budowanie złożonych kontroltek .....</b>         | <b>209</b> |
| Konfiguracja przycisku w podoknie Properties .....               | 209        |
| Pędzle .....   | 213        |
| Formatowanie tekstu na przycisku .....                           | 216        |
| Kontrola ułożenia elementów w pojemniku .....                    | 218        |
| <b>Rozdział 11. Style .....</b>                                  | <b>221</b> |
| Siatka i wiele kontroltek .....                                  | 221        |
| Zasoby okna .....  | 224        |
| Style .....  | 225        |
| Wyzwalacze .....   | 227        |
| Zasoby aplikacji .....   | 228        |
| <b>Rozdział 12. Transformacje i animacje .....</b>               | <b>233</b> |
| Transformacje kompozycji i renderowania .....                    | 233        |
| Uruchamianie transformacji w wyzwalaczu stylu .....              | 239        |
| Animacje .....   | 240        |
| Animacja w stylu .....   | 242        |
| Funkcje w animacji .....   | 243        |
| Animacja koloru .....  | 245        |

---

|  |            |
|--|------------|
| <b>Rozdział 13. Szablony kontroltek .....</b>                  | <b>247</b> |
| <b>Rozdział 14. Projektowanie własnych kontroltek .....</b>    | <b>251</b> |
| User Control .....   | 251        |
| Custom Control .....   | 253        |
| <b>Rozdział 15. Przegląd pojemników WPF .....</b>              | <b>257</b> |
| Pojemniki (Layout Containers) .....                            | 257        |
| Kontrolki ułożenia (Layout Controls) .....                     | 263        |
| Projektowanie własnego pojemnika .....                         | 267        |
| Listy (Items Controls) .....                                   | 269        |
| Szablony .....   | 270        |
| Zestaw przydatnych list .....                                  | 273        |
| Zadania .....  | 279        |
| <b>Część III. MVVM .....</b>                                   | <b>281</b> |
| <b>Rozdział 16. Wzorzec MVVM .....</b>                         | <b>283</b> |
| Model .....  | 284        |
| Widok .....  | 284        |
| Model widoku .....   | 285        |
| <b>Rozdział 17. Implementacja modelu i modelu widoku .....</b> | <b>287</b> |
| Model .....  | 287        |
| Warstwa dostępu do danych .....                                | 288        |
| Model widoku .....   | 289        |
| Alternatywne rozwiązania .....                                 | 292        |
| Ratujemy widok .....   | 294        |
| Zadania .....  | 295        |
| <b>Rozdział 18. Wiązanie danych (data binding) .....</b>       | <b>297</b> |
| Instancja modelu widoku i kontekst danych .....                | 297        |
| Alternatywne rozwiązanie .....                                 | 298        |
| Wiązanie pozycji suwaków i koloru prostokąta .....             | 299        |
| Zmiany w code-behind .....                                     | 300        |
| Implementacja interfejsu INotifyPropertyChanged .....          | 301        |
| Powiadomienia w alternatywnych modelach widoku .....           | 305        |
| Interfejs INotifyDataErrorInfo .....                           | 310        |
| Klasa ObservedObject .....                                     | 311        |

|  |            |
|--|------------|
| <b>Rozdział 19. Konwersja danych w wiązaniu .....</b>  | <b>313</b> |
| Prosta konwersja typów .....   | 313        |
| Konwersja klas Color i SolidColorBrush .....   | 315        |
| Multibinding .....   | 317        |
| Wiązanie między kontrolkami .....  | 318        |
| Konwersje „wbudowane” .....  | 321        |
| Zadania .....  | 321        |
| <br>   |            |
| <b>Rozdział 20. Polecenia (commands) .....</b>   | <b>323</b> |
| Interfejs ICommand .....   | 323        |
| Przycisk uruchamiający polecenie .....   | 324        |
| Sprawdzanie, czy wykonanie polecenia jest możliwe .....                                      | 327        |
| Resetowanie stanu suwaków po naciśnięciu klawisza .....                                      | 328        |
| Klasa RelayCommand .....   | 329        |
| Zdarzenia a polecenia .....  | 331        |
| Zamykanie okna .....   | 334        |
| Zadanie .....  | 335        |
| <br>   |            |
| <b>Rozdział 21. Zachowania, własności zależności i własności doczepione .....</b>            | <b>337</b> |
| Zachowania (behaviors) .....   | 337        |
| Własność zależności (dependency property) .....  | 339        |
| Własność doczepiona (attached property) i zachowanie doczepione<br>(attached behavior) ..... | 343        |
| Zadania .....  | 345        |
| <br>   |            |
| <b>Rozdział 22. Testy jednostkowe .....</b>  | <b>347</b> |
| Testy jednostkowe w Visual Studio 2015 i 2017 .....  | 348        |
| Uruchamianie testów .....  | 350        |
| Testy wielokrotne .....  | 352        |
| Dostęp do prywatnych pól testowanej klasy .....  | 353        |
| Atrapy obiektów (mock objects) .....   | 355        |
| Testowanie konwerterów .....   | 359        |
| Testowanie wyjątków .....  | 360        |
| <br>   |            |
| <b>Rozdział 23. Powtórzenie .....</b>  | <b>363</b> |
| Model .....  | 363        |
| Widok .....  | 364        |
| Model widoku .....   | 366        |
| Wiązanie .....   | 367        |



|  |            |
|--|------------|
| Konwerter .....  | 368        |
| Wzorzec MVVM .....   | 370        |
| Zadania .....  | 371        |
| <b>Część III. MVVM w przykładach .....</b>   | <b>373</b> |
| <b>Rozdział 24. Okna dialogowe w MVVM .....</b>                                      | <b>375</b> |
| Klasa bazowa okna dialogowego .....  | 375        |
| Polecenia wykonywane przed wyświetleniem<br>i po wyświetleniu okna dialogowego ..... | 378        |
| Okno dialogowe MessageBox .....  | 382        |
| Warunkowe wyświetlenie okna dialogowego .....  | 385        |
| Okna dialogowe wyboru pliku .....  | 387        |
| Łańcuch okien dialogowych .....  | 391        |
| Okna dialogowe z dowolną zawartością .....   | 393        |
| Zadania .....  | 402        |
| <b>Rozdział 25. Kontrolki w kontekście MVVM .....</b>                                | <b>403</b> |
| Kontrolka MVVM .....   | 403        |
| Kontrolka prawie MVVM .....  | 409        |
| Kontrolka FileBrowse .....   | 412        |
| Kontrolka FontDialogBox .....  | 416        |
| Zadania .....  | 423        |
| <b>Rozdział 26. Notatnik w MVVM .....</b>  | <b>425</b> |
| Widok .....  | 426        |
| Model .....  | 428        |
| Model widoku .....   | 430        |
| Zmiana rozmiaru czcionki rolką myszy .....   | 432        |
| Polecenia .....  | 434        |
| Wybór czcionki .....   | 437        |
| Drukowanie .....   | 441        |
| Zawijanie wierszy i inne ustawienia .....  | 442        |
| Edycja .....   | 444        |
| Historia zmian .....   | 450        |
| Klawisze skrótów .....   | 453        |
| Zadania .....  | 455        |

|   |            |
|---|------------|
| <b>Rozdział 27. Grafika w WPF .....</b>             | <b>457</b> |
| Kształty. Zegar .....                               | 457        |
| Model widoku .....                                  | 458        |
| Widok — zegar cyfrowy .....                         | 460        |
| Wykrycie trybu projektowania .....                  | 461        |
| Widok — zegar analogowy .....                       | 461        |
| Zmiana kształtu okna .....                          | 465        |
| Efekty .....  | 469        |
| Cień .....  | 469        |
| Potok renderowania .....                            | 473        |
| Własny efekt .....                                  | 474        |
| Kompilacja z pakietem Microsoft.HLSL.CSharpVB ..... | 485        |
| Grafika per pixel .....                             | 488        |
| Zadania .....                                       | 492        |
| <br>  |            |
| <b>Rozdział 28. Kolekcje w MVVM .....</b>           | <b>495</b> |
| Model .....   | 495        |
| Operacje CRUD .....                                 | 499        |
| Przechowywanie danych w pliku XML .....             | 499        |
| Model widoku zadania .....                          | 501        |
| Kolekcja w modelu widoku .....                      | 505        |
| Zadania .....                                       | 507        |
| <br>  |            |
| <b>Rozdział 29. Szablony danych .....</b>           | <b>509</b> |
| Prezentacja kolekcji w widoku .....                 | 509        |
| Style elementów kontrolki ListBox .....             | 511        |
| Konwertery .....                                    | 513        |
| Zapisywanie danych przy zamknięciu okna .....       | 517        |
| Modyfikacje kolekcji .....                          | 519        |
| Polecenia CRUD .....                                | 526        |
| Sortowanie .....                                    | 527        |
| Użycie okien dialogowych .....                      | 529        |
| Zadania .....                                       | 534        |
| <br>  |            |
| <b>Rozdział 30. Usługa REST .....</b>               | <b>535</b> |
| Protokół HTTP .....                                 | 535        |
| Dlaczego usługi REST? .....                         | 536        |
| Tworzenie usługi REST .....                         | 537        |
| Rejestrowanie zdarzeń .....                         | 537        |
| Model .....   | 539        |

---

|  |     |
|--|-----|
| Kontroler. Metody GET .....                                    | 542 |
| Wybór formatu zwracanych danych .....                          | 546 |
| Kontroler. Szablon adresu api/{controller}/{action}/{id} ..... | 548 |
| Korzystanie z metod GET usługi REST .....                      | 550 |
| Metody POST .....  | 552 |
| Metoda DELETE .....  | 554 |
| Zadania .....  | 558 |

## **Dodatek A. LINQ ..... 559**

|  |     |
|--|-----|
| Operatory LINQ .....                                   | 560 |
| Pobieranie danych (filtrowanie i sortowanie) .....     | 562 |
| Analiza pobranych danych .....                         | 563 |
| Wybór elementu .....                                   | 563 |
| Weryfikowanie danych .....                             | 563 |
| Prezentacja w grupach .....                            | 564 |
| Łączenie zbiorów danych .....                          | 564 |
| Łączenie danych z różnych źródeł (operator join) ..... | 565 |
| Możliwość modyfikacji danych źródła .....              | 566 |

## **Dodatek B. Pakiety NuGet ..... 569**

|  |     |
|--|-----|
| Instalacja nuget.exe .....                         | 570 |
| Tworzenie pakietu z projektu biblioteki klas ..... | 571 |
| Publikacja pakietu .....                           | 572 |
| Test pakietu .....                                 | 574 |
| Zależności między pakietami .....                  | 574 |
| Tworzenie pakietu z zestawem bibliotek .....       | 575 |
| Pakiet dla wielu platform .....                    | 576 |

## **Dodatek C. Kowariancja i kontrawariancja typów parametrycznych ..... 579**

## **Skorowidz ..... 585**

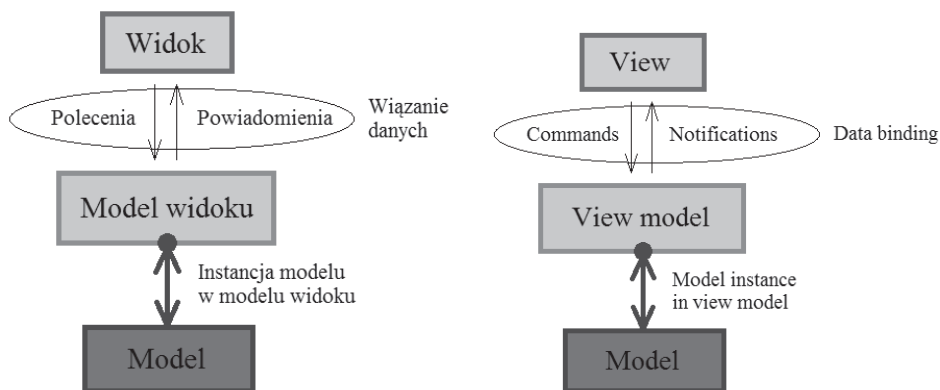


## Rozdział 16.

# Wzorzec MVVM

Część trzecia książki jest poświęcona przedstawieniu wzorca MVVM. To bardzo ważny wzorzec stosowany nie tylko w aplikacjach opartych na WPF, ale również w tzw. aplikacjach uniwersalnych (ang. *Universal Windows Platform* — UWP), a wcześniej także w projektach Silverlight. Jako przykładu użyjemy prostej aplikacji *KoloryWPF* opisanej w rozdziale 1. W tej części będziemy ją stopniowo zmieniać w projekt zgodny ze wzorcem MVVM. Dzięki temu poznamy nie tylko sam wzorzec, ale również wspierające go technologie zawarte w WPF. W kolejnej części wykorzystamy zdobytą w ten sposób wiedzę w kilku przykładowych projektach.

Aplikacja we wzorcu MVVM składa się z trzech warstw: modelu, modelu widoku i widoku (rysunek 16.1). W najprostszym przypadku, takim jak w aplikacji *KolorWPF*, poszczególne warstwy mogą się składać tylko z jednej klasy, ale zwykle jest ich oczywiście więcej.



**RYСУNEK 16.1.** Warstwy aplikacji we wzorcu MVVM (z lewej polska, a z prawej angielska terminologia)

## Model

Funkcja warstwy modelu jest najbardziej intuicyjna — odpowiada funkcjom modeli w innych wzorcach projektowych, chociażby w klasycznej dwuwarstwowej architekturze model-widok lub we wzorcach MVC i MVP. Model przechowuje dane stanowiące stan systemu, do którego należy. To oznacza, że musi zawierać definicje typów, które będą mogły te dane przechowywać. Powinien także zawierać logikę, która będzie tych danych dotyczyła. Całość nie powinna zależeć od technologii specyficznych dla jakiejś konkretnej platformy lub technologii. Powinna być w pełni przenaszalna. Najlepiej, gdyby jedyną używaną w nich przestrzenią nazw była przestrzeń `System`<sup>1</sup>.

Klasy modelu nie mogą, i to jest bardzo ważne, znać żadnych szczegółów dotyczących wyższych warstw projektu — nie powinny w żaden sposób od nich zależeć. Model powinien być całkowicie autonomiczny. To m.in. stwarza bardzo dogodne warunki do jego testowania.

Kluczowy w projektowaniu warstwy modelu, tak jak generalnie w programowaniu obiektowym, jest podział odpowiedzialności — należy jasno ustalić, za co odpowiedzialne są poszczególne klasy lub zbiory klas. Część odpowiedzialności może, a nawet powinna, być wydzielona do osobnych modułów w warstwie modelu. Na przykład za trwały zapis danych można uczynić odpowiedzialny osobny moduł dostępu do danych (ang. *data access layer* — DAL), który może być statyczną klasą narzędziową przyjmującą instancje klas domenowych i zapisującą ich stan. Podobnie logika modelu może być wydzielona do osobnego modułu tzw. logiki biznesowej (ang. *business logic layer* — BLL), która operuje na instancjach klas modelu.

## Widok

Widok jest odpowiedzialny za kontakt z użytkownikiem. W WPF, a także w aplikacjach UWP, widokiem jest kod XAML opisujący graficzny interfejs użytkownika (ang. *graphical user interface* — GUI). Z widokiem związana jest klasa okna, w której w pierwszej części książki bez oporu umieszczaliśmy metody zdarzeniowe. Tworzy ona tzw. kod zaplecza widoku, czyli *code-behind*. Zgodnie z zaleceniami wzorca MVVM kod ten powinien być

---

<sup>1</sup> Dobrym pomysłem jest tworzenie tej warstwy w metodologii projektowania domenowego (ang. *domain-driven design* — DDD). W wielkim uproszczeniu oznacza to, że projektujemy zbiór klas składających się na model razem z ekspertem w dziedzinie, której program ma dotyczyć. Często jest to klient lub osoba przez niego wskazana. Wówczas należy uważnie słuchać słownictwa, jakiego ów ekspert używa, bo często stosowane przez niego rzeczowniki są dobrymi kandydatami na nazwy podstawowych klas modelu. Z kolei czasowniki towarzyszące tym rzeczownikom będą prawdopodobnie nazwami kluczowych metod. Przy czym w DDD nie chodzi tylko o wybieranie nazw klas i metod, ale przede wszystkim o ich zawartość i wyznaczenie relacji między klasami. Ma ona odzwierciedlać relacje pojawiające się w języku używanym przez eksperta. To oczywiście trywializacja, ale dobrze oddaje ideę DDD.

ograniczony do minimum, a najlepiej, żeby go w ogóle nie było. W tym sensie wzorzec MVVM całkowicie odwraca wzorzec widoku autonomicznego. Głównym powodem unikania kodu C# w warstwie widoku, a przynajmniej w klasie okna, jest to, że kod ten, jako silnie związany z kontrolkami, jest nieprzenaszalny, a przy tym też trudny do przetestowania. Ponadto zanurzenie logiki prezentacyjnej w widoku znacząco utrudnia współpracę między projektantami interfejsu tworzącymi widok a programistami odpowiedzialnymi za niższe warstwy aplikacji. Zmniejsza też elastyczność projektu, utrudniając tym samym jego zmiany.

## Model widoku

Model widoku jest abstrakcją widoku. Jeżeli możemy sobie wyobrazić kilka wariantów graficznego interfejsu użytkownika naszej aplikacji, dla różnych użytkowników, środowisk i platform, to model widoku w tych wszystkich przypadkach powinien pozostawać taki sam. Analogicznie możemy sobie wyobrazić różne stoły, różnej wielkości i o różnych kształtach, z trzema lub czterema nogami, nie zmienia to jednak definicji stołu jako miejsca, przy którym można usiąść i coś na nim położyć. Podobnie wiele może być projektów widoku. Ale model widoku musi być jak definicja stołu, jego zapisana idea — powinien być jak najprostszy, lecz kompletny. Powinien wobec tego zawierać tylko elementy konieczne do określenia, do czego mają być użyte widoki. Warto podjąć wysiłek, żeby doprowadzić kod modelu widoku do jak najwyższego poziomu abstrakcji. Z tych górnolotnych rozważań wynika, że najlepszym sprawdzianem poprawności modelu widoku są zmiany wprowadzane w widoku. Tych w trakcie rozwijania projektu zwykle nie brakuje. Jeżeli model widoku jest dobrze zaprojektowany, takie zmiany widoku powinny się odbyć bez jego modyfikacji. Pamiętaj jednak, że — jak wiele dobrych praktyk w informatyce — jest to raczej cel, do którego dążymy, niż twarde wymaganie, stawiane osobie projektującej model widoku.

Funkcją modelu widoku jest udostępnienie widokowi instancji klas z warstwy modelu (na rysunku 2.1 odpowiada to ruchowi do góry) oraz zmienianie stanu tych instancji w wyniku działań użytkownika wykrytych w warstwie widoku (ruch w dół). W tym drugim przypadku model widoku odpowiedzialny jest m.in. za weryfikację przekazywanych danych. Model widoku odgrywa więc rolę pośrednika między warstwami modelu i widoku, a jednocześnie adaptera dla przekazywanych danych. Owo pośredniczenie najczęściej odbywa się w taki sposób, że obiekty modelu są prywatnymi polami modelu widoku. Model widoku udostępnia je lub ich części w swoich własnościach, jest wobec tego świadomy warstwy modelu, nie powinien być natomiast świadomy warstwy widoku — to widok powinien być świadomy modelu widoku. Połączenie między modelem widoku a widokiem jest zwykle bardzo „luźne”. Oparte jest nie na odwołaniach w kodzie C#, lecz na wiązaniach danych umieszczonych w kodzie XAML. To luźne wiązanie (ang. *binding*) wspomaga niezależną pracę nad widokiem i modelem widoku i znakomicie ułatwia wprowadzanie zmian w poszczególnych warstwach, z całkowitym ich przebudowaniem włącznie.

Ta druga zaleta jest szczególnie warta docenienia, choć jest ona w większym lub mniejszym stopniu zaletą wszystkich wzorców z wyraźnie rozdzielonymi warstwami (modułami).

W modelu widoku zapisana jest cała logika prezentacyjna określająca procedury kontaktu z użytkownikiem z uwzględnieniem weryfikacji danych. Mimo tego pozostaje łatwa do testowania, nie ma w niej bowiem odwołań do kontrolki ani założonej bezpośrednio interakcji z użytkownikiem.



Doskonale zdaję sobie sprawę, że dla osób, które nie miały jeszcze kontaktu ze wzorcem MVVM albo chociażby z MVP lub MVC, większość powyższych zdań o modelu widoku jest trudna do zrozumienia. Zadaniem kolejnych rozdziałów z tej części książki będzie wyjaśnienie tego na konkretnym przykładzie. Po przeczytaniu dalszych rozdziałów warto wrócić do niniejszego i przeczytać go jeszcze raz, w całości lub przynajmniej w części dotyczącej modelu widoku. To powinno pomóc poukładać sobie w głowie wiedzę o MVVM.

W przypadku aplikacji *KoloryWPF* modelem może być prosta klasa opisująca kolor, zawierająca tylko trzy lub cztery składowe typu `byte`. Będzie to klasa w stylu POCO (z ang. *plain old CLR object*). Jej jedynym zadaniem będzie przechowywanie danych, bez żadnej rozbudowanej logiki, dla której w tym projekcie nie ma po prostu zapotrzebowania. Odpowiedzialność za zapis stanu modelu pozostawimy osobnej klasie statycznej również należącej do warstwy modelu. Prostota naszej aplikacji spowoduje, że model widoku będzie z początku równie prosty i w istocie bardzo podobny do samego modelu. Z czasem dodamy do niego elementy charakterystyczne dla klas modelu widoku, m.in. polecenia i mechanizm powiadomień. A ponieważ podstawowym celem aplikacji jest możliwość kontrolowania trzech składowych koloru, model widoku musi udostępniać własności reprezentujące te składowe. Oprócz tego wyposażymy go w metodę, którą potem przekształcimy w tzw. polecenie, umożliwiające zapis stanu aplikacji (czyli *de facto* stanu modelu).

To nie jest oczywiście jedyna architektura, jaką można sobie wyobrazić dla tej aplikacji. Dobrym modelem mogłaby być przecież klasa `Properties.Settings` utworzona przez Visual Studio w momencie określania ustawień aplikacji. Przy takim założeniu naszym jedynym zadaniem pozostaje napisanie modelu widoku, który tę klasę udostępniłby widokowi. Można również rozważyć klasę `System.Windows.Media.Color` jako klasę modelu, ale nie uważam, żeby korzystanie z klas przeznaczonych do budowania interfejsu było dobrym pomysłem na tworzenie modelu. Dlatego pozostaniemy przy rozwiązaniu „kanonicznym”, lecz pamiętając, że wzorec MVVM pozwala na pewne wariacje (por. projekt *ZegarWPF* z rozdziału 28.).

Ostrzegalem już, że aplikacja, którą od tego momentu będziemy przebudowywać, jest bardzo prosta, w kontekście uczenia się wzorca MVVM to jednak moim zdaniem zaleta. Brak szczegółów związanych z bardziej skomplikowanym projektem pozwoli czytelnikowi łatwiej dostrzec istotę wzorca.



# Skorowidz

## A

- akcja Execute, 377
- analiza pobranych danych, 563
- animacja DoubleAnimation, 241, 245
- animacje, 123, 240
  - funkcje, 243
  - kolorów, 245
  - w stylu, 242
- aplikacje sterowane zdarzeniami, 17
- argument sender, 79
- arkusze stylów CSS, 221
- atrapy obiektów, mock objects, 355
- atrybut, 184
  - Angle, 234
  - Background, 226
  - CallerMemberNameAttribute, 303
  - Click, 46
  - Closing, 53
  - Command, 325
  - ContentProperty, 393
  - CornerRadius, 266
  - Duration, 241
  - Fill, 32
  - FontSize, 216
  - Foreground, 226
  - Header, 45
  - Height, 23
  - HorizontalAlignment, 23
  - Icon, 70
  - InputGestureText, 44
  - IsCancel, 396
  - IsCheckable, 56
  - IsDefault, 396
  - KeyDown, 32
  - LoadedBehavior, 199

- Margin, 23
- Name, 26
- Orientation, 258
- PreviewMouseLeftButtonDown, 107
- RelativeSource, 325
- RepeatBehavior, 241
- Storyboard.TargetProperty, 241
- StringFormat, 321, 460
- Style, 226
- TargetType, 226
- TextChanged, 51
- Title, 42
- ValueChanged, 27
- VerticalAlignment, 23
- Width, 23
- x:Class, 25
- xmlns, 25

AV, autonomous view, 19

## B

- biblioteka
  - Microsoft.Expression.Interaction.dll, 332, 337, 467, 518
  - System.Drawing.dll, 73, 125
  - System.Speech.dll, 203
  - System.Windows.Controls.Ribbon.dll, 73
  - System.Windows.Form, 59, 73s.dll, 125
  - System.Windows.Interactivity.dll, 332, 337, 432, 467, 518
  - Windows Forms, 58
- biblioteki
  - .NET, 59
  - DLL, 180
  - przenośne PCL, 295

Blend for Visual Studio 2017, 216

BLL, buisness logic layer, 284  
 bulgotanie, bubbling, 93, 101

## C

Choinka, 119  
 cień, 469  
   obróć, 472  
 code-behind, 27, 46, 284, 300, 301  
   kontrolki, 409, 413  
 CRUD, create, read, update, delete, 499  
 CSS, cascading style sheets, 221  
 czcionka, 61, 276, 277  
   konfiguracja, 63  
   wygląd, 439  
   zmiana rozmiaru, 432  
   zmienianie, 65

## D

DAL, data access layer, 284, 288  
 data binding, 297  
 debugowanie, 28  
 definiowanie  
   stylu, 225  
   typów inwariantnych, 579  
   wartości doczepionej, 397  
   własności, 33  
   zachowania, 338  
   zachowania doczepionego, 343  
 deklaracja, 183  
 deserializator, 551  
 dodawanie biblioteki do referencji, 181  
 dostęp  
   do danych, 284  
   do pól prywatnych, 353  
 drag & drop, 103  
 drukowanie, 65, 441  
 DRY, Don't Repeat Yourself, 54, 221  
 drzewo  
   kontrolki XAML, 28, 29  
   TreeView, 269  
   węzłów, 190  
   XML, 191  
 DTO, Data Transfer Object, 289

dynamiczne tworzenie  
   planszy, 144  
   przycisków, 94  
 dyrektywa using, 46  
 dźwięk, 128, 204

## E

edycja  
   kolekcji, 519  
   tekstu, 444  
   ustawień aplikacji, 34  
 edytor kodu XAML, 21  
 efekt cienia, 469  
 efekty własne, 474  
 ekran powitalny, 124  
 element, 184  
   applicationSettings, 34, 37  
   Bold, 217  
   Button, 213  
   CustomContentDialogBox, 417  
   DoubleAnimation, 241  
   Ellipse, 461  
   EventTrigger, 332  
   główny, 189  
   GradientStop, 214  
   Grid, 23, 221, 251  
   GridSplitter, 104  
   InvokeCommandAction, 332  
   LinearGradientBrush, 224  
   ListBox, 104  
   MediaElement, 199  
   MenuItem, 44  
   MessageDialogBox, 385, 391, 531  
   Rectangle, 21  
   Ribbon, 73  
   ScrollViewer, 433  
   Separator, 44  
   Setter, 226, 239  
   SimpleMessageDialogBox, 377, 529  
   Slider, 21  
   StatusBarItem, 44  
   System.Runtime.PropertyInfo, 399  
   TextBlock, 106, 216  
   TextBox, 40, 47  
   ToolBar, 71  
   UserControl, 251

- userSettings, 34
- Window, 25, 97
- Window.Resources, 523

etykieta

- przycisku, 216
- TextBlock, 271

## F

filtrowanie, 562

focus, 28, 328, 344

format

- JSON, 546
- XML, 546

formatowanie ścieżki pliku, 392

formaty zwracanych danych, 546

formularz, 521

funkcje

- w animacji, 243
- wygladzania, 244

funktionalności pola edycyjnego, 433

## G

garbage collector, 31

gra Reversi, 131

- granie z komputerem, 160

- historia ruchów, 147

- implementacja, 134

- interakcja z użytkownikiem, 146

- menu, 161

- model, 132

- obliczanie liczb pól, 137

- prezentacja planszy, 142

- rozbudowa silnika, 153

- stan planszy, 133

- testy jednostkowe, 139

- tworzenie kontrolki, 168

- widok, 142

- wykrywanie sytuacji, 148

- wyszukiwanie ruchu, 154

gradient, 214

- liniowy, 216

- radialny, 214

graficzny interfejs użytkownika, GUI, 39, 284

grafika, 457

grupowanie

- danych, 564

- transformacji, 237

GUI kontrolki, 412

## H

HLSL, High-level Shader Language, 474

HTTP, Hypertext Transfer Protocol, 535

## I

ikona

- Brush Resources, 210

- menu, 68

- okna, 42

- w zasobniku, 125

implementacja

- interfejsu ICommand, 323

- interfejsu INotifyPropertyChanged, 301, 305

- modelu, 287

- modelu widoku, 287

inicjator obiektu, 35

inicjowanie

- koloru, 31

- procesu przenoszenia, 107

instalacja nuget.exe, 570

instancja modelu widoku, 297

IntelliSense, 29

interfejs

- aplikacji, 21, 39, 178

- ICommand, 95, 323

- IComparable, 527

- IDataErrorInfo, 310

- IEnumerable<>, 560, 563

- IMultiValueConverter, 317, 318

- INotifyCollectionChanged, 505, 507, 509

- INotifyDataErrorInfo, 310

- INotifyPropertyChanged, 301, 305, 307, 377, 458, 502

interfejsy silników gier planszowych, 179

interpolacja liniowa, 158

**J**

## język

- HLSL, 474
- XML, 183

**K**

## katalog projektu, 21

## kierunek cienia, 472

## klasa

- BlurEffect, 469
- BooleanToVisibilityConverter, 442
- Brush, 299
- Brushes, 273, 274
- ByteToDoubleConverter, 314
- Color, 287, 315
- ColorConverter, 272
- CommandDialogBox, 387
- Control, 63, 253
- CustomContentDialogBox, 397
- DataObject, 108
- DropShadowEffect, 469
- File, 46
- Fonts, 276
- Graphics, 457
- List<>, 527
- MediaPlayer, 199
- MessageBox, 386
- MessageDialogBox, 385
- ObjectDataProvider, 273
- ObservedObject, 311
- OpenFileDialog, 48
- Panel, 257
- Path, 46
- PrintDialog, 65
- RelayCommand, 329, 437, 517
- Resources, 82
- ReversiSilnik, 133
- ShaderEffect, 476
- Shape, 457
- SolidColorBrush, 315
- StackPanel, 218
- TextBox, 450
- UIElement, 233
- WebApiConfig, 546
- Window, 186

XDocument, 185, 187, 189

XmlTextReader, 188

## klasy

- konwertera, 369
- modelu, 284, 364, 429
- modelu widoku, 366, 430, 459
- okien dialogowych wyboru pliku, 388
- potomne, 153
- testów jednostkowych, 349
- własnego efektu, 476

klawisze skrótów, 67, 453

kod zapleczka widoku, *Patrz code-behind*

## kolekcja

- obiektów RuntimePropertyInfo, 274
- Triggers, 227

## kolekcje

- modyfikacje, 519
- prezentacja w widoku, 509
- w modelu widoku, 505
- w MVVM, 495

kolor tła, 58

kolory, 288

komentarz, 184

komórka, 222

## kompilacja

- shadera, 475
- warunkowa, 356

## konfiguracja

- czcionki, 63
- okna zapisu pliku, 49
- pędzla, 211
- przycisku, 209

konstruktor, 187

klasy, 134

## kontekst

- danych, 297
- wiązania widoku, 431

kontrawariancja typów parametrycznych, 579

kontrola przepływu zdarzeń trasowanych, 90

kontroler, 542, 543, 548

kontrolka, 87, *Patrz także element*

Border, 266

Button, 203, 264

CheckBox, 91, 92

ComboBox, 203, 270

DatePicker, 524

FileBrowse, 412, 413, 415

- FontDialogBox, 416, 439
  - kod widoku, 417
  - kontekst wiązania, 420
  - konwerter czcionki, 416
  - testowanie, 422
- GridSplitter, 104
- ItemsControl, 269
- Label, 200
- ListBox, 91, 100, 105, 147, 269, 511
- MediaElement, 199
- MenuItem, 56
- MVVM, 403
- Popup, 265
- prawie MVVM, 409
- ScrollViewer, 263
- Slider, 22, 203
- TextBlock, 44, 216, 264
- TextBox, 46, 55
- TreeView, 190, 192, 193
- ViewBox, 264
- kontrolki
  - zmiana ułożenia, 234
  - niestandardowe, Custom Control, 249, 253
  - style, 225
  - szablony, 247
  - ułożenia, Layout Controls, 263
  - użytkownika, User Control, 249, 251
  - własne, 168, 251
  - WPF, 21, 39
  - wyzwalacze, 227
  - złożone, 209
- konwersja
  - typów, 313
  - wiele-do-jednego, 317
- konwerter, 313, 368, 408, 513
  - AlternationConverter, 321
  - BooleanToVisibilityConverter, 321
  - BorderGapMaskConverter, 321
  - DataGridLengthConverter, 321
  - JournalEntryUnifiedViewConverter, 321
  - MenuScrollingVisibilityConverter, 321
  - ProgressBarBrushConverter, 321
  - ProgressBarHighlightConverter, 321
  - ZoomPercentageConverter, 321
- konwertery
  - czcionek, 401, 416
  - godzin, 463

- koloru, 316
  - obliczające kąt, 484
- kończenie gry, 152
- kowariancja typów parametrycznych, 579
- kształt kursora, 109
- kształty, 457
- kursy walut NBP, 193

## L

- LINQ, Language Integrated Query, 185, 559
  - grupowanie danych, 564
  - łączenie zbiorów danych, 564
  - metody rozszerzające, 560
  - modyfikacja danych źródła, 566
  - operator join, 565
  - operatory, 560
  - pobieranie danych, 562
  - struktura zapytania, 562
  - weryfikowanie danych, 563
- LINQ to DataSet, 567
- LINQ to Entity, 567
- LINQ to Objects, 560, 567
- LINQ to XML, 185, 500
- lista, 147
  - czcionek, 276
  - dekoracji tekstu, 278
  - grubości czcionki, 277
  - Items Controls, 269
  - kolorów, 275
  - ListBox, 90, 269
  - ListView, 269
  - rozwijana ComboBox, 203
  - style elementów, 511
  - zadań, 497
- log, 538
- logika biznesowa, 284
- lokalizacja, 80

## Ł

- łańcuch okien dialogowych, 391, 532
- łączenie
  - danych z różnych źródeł, 565
  - zbiorów danych, 564

**M**

macierz transformacji, 238

mechanizm

- Live Unit Testing, 350
- przeciągnij i upuść, 103
- Reflection, 273

menu, 161

- aplikacji, 73
- Edycja, 55
- główne, 44
- Widok, 56
- z ikonami, 71

metoda

- AddElementToNode, 190
- Arrange, 267
- CanExecute, 327
- Close, 334
- Convert, 318
- ConvertBack, 315
- createFlowDocument, 65
- DELETE, 554
- DependencyProperty.RegisterAttached, 343
- Descendants, 190
- DoDragDrop, 108
- Element, 189
- Equals, 555
- Execute, 325
- Executed, 100
- GET, 542, 545, 549, 550
- GetItemAt, 106
- GetModel, 507
- Load, 189
- Measure, 267
- MediaElement.Play, 199
- MessageBox.Show, 385
- Nodes, 189
- PopulateTreeViewWithXmlFile, 190
- POST, 552
- PreviewExecuted, 100
- PrintDialog.PrintDocument, 65
- sliderR\_ValueChanged, 29
- Sort, 527, 528
- TextBox.Redo, 450
- TextBox.Undo, 450
- Type.GetProperties, 274

Window.Close, 122

Window\_Closed, 328

zdarzeniowa kliknięcia przycisku, 525

metody

- konwertujące priorytet zadania, 496
- parsujące, 194
- rozszerzające LINQ, 560
- zdarzeniowe, 29, 55, 57
  - przerwanie sekwencji wywoływań, 92

model, 132, 284, 287, 363, 428, 495, 539

model widoku, 285, 289, 292, 293, 366, 430, 458, 501

alternatywny, 305

kolekcje, 505

pojedynczego zadania, 502

moduł, 167

dostępu do danych, 284

logiki biznesowej, 284

modyfikacja

danych źródła, 566

kolekcji, 519

modyfikator ref, 59

mowa, 203

multibinding, 317, 524

multimedia, 199

MVVM, model-view-viewmodel, 16, 281, 370

kolekcje, 495

kontrolki, 403

model, 284, 363

model widoku, 285, 366

notatnik, 425

okna dialogowe, 375

struktura aplikacji, 370

warstwy aplikacji, 283

wiązanie danych, 297, 367

widok, 284, 364

mysza

zmiana rozmiaru czcionki, 432

**N**

naciśnięcie klawisza Escape, 32

nazwa okna, 42

Notatnik, 39

drukowanie, 65

ikona okna, 42

ikony menu, 68

- klawisze skrótów, 67
- lokalizacja, 80
- menu Edycja, 55
- menu główne, 44
- menu Widok, 56
- okna dialogowe, 45
- pasek narzędzi, 71
- pasek stanu, 43
- projektowanie interfejsu, 39
- wstążka, 73

Notatnik w MVVM, 425

- drukowanie, 441
- edycja, 444
- funkcjonalności pola edycyjnego, 433
- historia zmian tekstu, 450
- klawisze skrótów, 453
- kod widoku, 426
- model, 428
- model widoku, 430
- polecenia, 434
- ustawienia, 442
- widok, 426
- wybór czcionki, 437
- zawijanie wierszy, 442
- zmiana rozmiaru czcionki, 432

NuGet, 332, 474, 569

- publikacja pakietu, 572
- test pakietu, 574
- tworzenie pakietu, 571

NuGet CLI, 570

## O

obiekt

- DataObject, 108, 115
- FlowDocument, 65
- RotateTransform, 234
- typu WritableBitmap, 488
- XDocument, 188
- XElement, 188

obiekty

- typu
  - System.Reflection.RuntimePropertyInfo, 274
  - Windows Forms, 58

obrót, 234, 250, 472

odczyt atrybutu elementu, 189

odpowiedź, response, 535

odśmiecacz, 31

odtwarzacz MediaElement, 199

odtwarzanie

- dźwięku, 128, 204
- wideo, 199

odwołanie do stylu, 230

okno

- New Project, 20
- o dowolnym kształcie, 119
- przesuwanie, 121
- Window, 25
- zamykanie, 122

okno dialogowe, 375, 529

- elementy, 436
- FontDialogBox, 437
- klasa bazowa, 375
- klasy, 388
- łańcuch, 532
- łańcuch wywołań, 391
- MessageBox, 382
- polecenia, 378
- wartości doczepione, 397
- warunkowe wyświetlanie, 385
- wyboru pliku, 387
- wybór czcionki, 64
- wybór pliku, 46, 202
- z dowolną zawartością, 393
- z formularzem, 533
- zapis pliku, 49

operacja

- DELETE, 536
- GET, 536
- POST, 536
- PUT, 536

operacje

- CRUD, 499, 526, 536
- protokołu HTTP, 544

operator join, 565

operatory LINQ, 560

otwieranie pliku, 390

## P

pakiet

- dla wielu platform, 576
- Microsoft.HLSL.CSharpVB, 485

- pakiet
  - NuGet, 569
  - z zestawem bibliotek, 575
  - zależności, 574
- parametr shadera, 477
- parser XAML, 216
- parsowanie ręczne pliku XML, 551
- pasek
  - narzędzi, 71
  - stanu, 43
  - szybkiego dostępu, 76
- pędzel, 211, 213
  - Brush, 299
  - LinearGradientBrush, 214, 224
  - RadialGradientBrush, 214
- pętla do..while, 136
- pixel, 488
- plansza, 133, 168
  - kolory, 172
  - rozmiar, 171
  - tworzenie, 170
  - zdarzenia, 174, 175
  - zmiana kolorów, 173
- plik
  - App.config, 34
  - MainWindow.xaml.cs, 47
  - Resources.pl.resx, 83
  - Settings.Designer.cs, 35
  - Settings.settings, 35
  - user.config, 37
- pliki
  - .csproj, 571
  - .ico, 42
  - .nuspec, 571–576
  - dźwiękowe, 128
  - tekstowe, 45
  - XML, 183
- plótno Canvas, 261
- pobieranie
  - danych, 562
  - nazw kolorów, 274
- pochylenie, 237
- POCO, plain old CLR object, 286
- podgląd okna, 21
- podokno
  - Live Property Explorer, 28
  - Live Visual Tree, 28
  - Properties, 21, 209
    - konfiguracja przycisku, 209
  - Solution Explorer, 34
  - Test Explorer, 350
  - Toolbox, 21
- podpowiadanie kodu, 29
- podwójna animacja DoubleAnimation, 241
- pojemnik, 257, 267
  - Canvas, 219, 261
  - ComboBox, 203
  - DockPanel, 40, 143, 219, 257, 259
  - Grid, 23, 25, 40, 104, 219, 257
  - ListBox, 104
  - Layout Containers, 257
  - RelativePanel, 262
  - StackPanel, 218, 257, 275
  - UniformGrid, 260
  - WrapPanel, 258
- pojemniki
  - układanie kontrolki, 218
  - własne, 267
- pole
  - edycyjne, 449
    - TextBox, 203
  - opcji CheckBox, 90
  - tekstowe
    - czyszczenie zawartości, 54
- polecenia, 331, 434
  - CRUD, 526
  - paska narzędzi, 437
  - przed i po wyświetleniu komunikatu, 378
  - sprawdzanie wykonania, 327
  - trasowane, routed commands, 95
  - uruchamianie, 324
- polecenie, command, 323
  - CommandBefore, 529
  - CommandCancel, 385
  - CommandNo, 385
  - CommandOK, 385
  - CommandYes, 385
  - Run IntelliTests, 349
  - Show, 377, 391, 529
- położenie shaderów, 481
- potok renderowania, 473
- potwierdzenie zamknięcia aplikacji, 50
- powiadomienia, 305
- pozycja piksela, 491



prezentacja kolekcji, 509  
priorytet zadania, 496  
projektowanie  
  biblioteki, 180  
  interfejsu, 21  
  interfejsu aplikacji, 39  
  kontrolki MVVM, 403  
  kod modelu, 404  
  kod modelu widoku, 405  
  kod XAML, 403  
  konwerter, 407  
  wiązanie, 407, 408  
  pojemnika, 267  
prostokąt Rectangle, 21  
protokół HTTP, 535  
próbnik tekstury, 479  
przechowywanie  
  danych, 183, 499  
  składowych koloru, 287  
  stanu kolekcji, 500  
przeciągnij i upuść, 103  
  akceptacja upuszczenia, 109  
  inicjacja procesu, 107  
  opóźniona, 114  
  przenoszenie elementów, 111, 115  
  upuszczenie elementu, 110  
przestrzeń nazw, 26  
  JacekMatulewski.WpfUtils, 191  
  Microsoft.VisualStudio.TestTools.UnitTesting, 353  
  Microsoft.Win32, 48  
  Notatnik.NET.Properties, 81  
  System.Drawing, 59, 61  
  System.Globalization, 392  
  System.IO, 46  
  System.Speech, 203, 205  
  System.Speech.Synthesis, 203  
  System.Windows.Controls, 257, 397  
  System.Windows.Controls.Ribbon, 79  
  System.Windows.Data, 392  
  System.Windows.Input, 324  
  System.Windows.Media, 60  
  System.Windows.Media.Animation, 124  
  System.Windows.Media.Effects, 474  
  System.Windows.RoutedEventArgs, 88  
  System.Windows.Shapes, 46  
  System.Xml.Linq, 187

przesuwanie okna, 121  
przesyłanie referencji, 326  
przycisk  
  Button, 203  
  formatowanie etykiety, 216  
  uruchamiający polecenie, 324  
przyciski  
  transformacje, 233  
  zagnieżdżone, 89, 94  
publikacja pakietu, 572

## R

Reflection, 273  
reguła DRY, 221  
rejestrwanie  
  uruchomienie metody, 91  
  zdarzeń, 537  
renderowanie, 236, 473  
resetowanie stanu suwaków, 328  
REST, 535  
rozmazanie, blur, 469  
rozszerzanie  
  kontrolki ListBox, 105  
  klasy Window, 186  
rzutowanie, 313

## S

scenorys, 123  
sekcja  
  get, 33  
  set, 33  
separacja modułów, 167  
shader, 477, 482  
siatka, 221  
  Grid, 23  
  podział, 222  
  rozmieszczenie kontrolki, 223  
silnik gry, 132  
skalowanie, 237  
słownik zasobów, resource dictionary, 371  
słowo kluczowe this, 58, 106  
sortowanie, 276, 527, 562  
Speech API, 203  
splash screen, 124

- stan
    - aplikacji, 34
    - suwaków, 328
  - struktura
    - aplikacji MVVM, 283, 370
    - drzewa, 190
    - pliku XML, 190
    - zapytania LINQ, 562
  - styl, 221, 225, 516
    - animacja, 242
    - elementów listy, 511
    - przycisku, 512
    - w zasobach aplikacji, 229
    - w zasobach globalnych, 230
  - suwak
    - resetowanie, 328
    - Slider, 21, 203
  - synteza mowy, 203
  - szablon, 270, 272
    - adresu, 548
    - danych, 509
    - kontrolerek, 247
    - kontrolera, 543
  - szum, 489
- T**
- tarcza zegara, 464
  - testowanie
    - kontrolki FontDialogBox, 422
    - modelu widoku, 460
    - konwerterów, 359
    - pakietu, 574
    - modelu widoku, 357
    - wyjątków, 360
  - testy
    - jednostkowe, 138, 347
    - na żywo, 350
    - uruchamianie, 350
      - w Visual Studio, 348
    - wielokrotne, 352
  - timer, 160, 202, 461, 483
  - tło, 58
    - animacja koloru, 245
    - przezroczyste, 120
    - przycisku, 407
  - transformacja MatrixTransform, 238
  - transformacje
    - grupowanie, 237
    - obrót, 234
    - pochylenie, 237
    - skalowanie, 237
    - translacja, 237
    - uruchamianie w wyzwalaczu, 239
    - kompozycji, 233, 236
    - renderowania, 235
  - translacja, 237
  - tryb
    - debugowania, 28
    - selekcji, 28
  - tunelowanie, tunneling, 93, 101
  - tworzenie
    - instancji modelu widoku, 298
    - kontrolerek, 168, 251
    - kontrolerek złożonych, 209
    - pakietu, 571, 575
    - pliku XML, 185
    - pojemnika, 267
    - projektu, 20
    - timera, 202
    - usługi REST, 537
    - własności, 33
    - zadania, 523
  - typ danych
    - byte, 313
    - double, 313
    - object, 313
  - typy
    - inwariantne, 579
    - kontrawariantne, 579, 581
    - kowariantne, 579, 582
    - parametryczne, 356
    - wyliczeniowe, 169
- U**
- ukrywanie przycisku, 513
  - upuszczenie przenoszonego elementu, 110
  - uruchamianie
    - polecenia, 324
    - testów, 350
  - usługa REST, 181, 535, 536
    - format zwracanych danych, 546
    - kontroler, 542, 548

- metoda DELETE, 554
- metoda GET, 550
- metoda POST, 552
- model, 539
- rejestrwanie zdarzeń, 537
- tworzenie, 537
- ustawienia
  - pliku efektu, 486
  - projektu, 43
- usuwanie zadania, 556, 557
- UWP, Universal Windows Platform, 15, 283, 318
- użycie zasobów, 81

## V

- Visual Studio
  - testy jednostkowe, 348

## W

- warstwa dostępu do danych, DAL, 288
- warstwy aplikacji, 283
  - model, 284
  - model widoku, 285
  - widok, 284
- wartość null, 301, 334
- warunkowe wyświetlanie okna dialogowego, 386, 387
- wersje platformy .NET, 43
- weryfikowanie danych, 563
- wiązanie
  - danych, data binding, 297, 299, 367
    - konwersja danych, 313
    - polecenia, 323
    - powiadomienia, 305
  - dwustronne, 299
  - koloru, 319
  - między kontrolkami, 318
  - widoku, 431
- wideo, 199
- widok, 142, 284, 294, 364, 426, 460
  - prezentacja kolekcji, 509
- wielowiązanie danych, 317, 401
- Windows Forms, 58
- własności zależności, 267, 378

- własność
  - ActualWidth, 271
  - AllowDrop, 109
  - Angle, 240
  - AssociatedObject, 338
  - Background, 119, 213
  - Center, 209
  - Color, 31, 299
  - ColumnDefinitions, 221, 261
  - Command, 99
  - CommandAfter, 378
  - CommandBefore, 378
  - CommandParameter, 389
  - CommandProperty, 378
  - Content, 209, 393ContentSize, 267
  - DataContext, 298, 301
  - DialogBypassButton, 386
  - Direction, 469
  - DockPanel.Dock, 41, 259
  - doczepiona, attached property, 343
  - Effect, 469
  - FilePath, 389, 412
  - Fill, 30, 213, 299, 301
  - Filter, 48
  - FontFamily, 439
  - FontSize, 439
  - FontStyle, 439
  - FontWeight, 439
  - Foreground, 210, 213, 439
  - Handled, 92
  - HorizontalAlignment, 209
  - HorizontalContentAlignment, 209
  - IsChecked, 56, 265, 442
  - IsDialogBypassed, 386
  - IsEnabled, 97, 343
  - IsMouseOver, 227
  - IsOpen, 265
  - LayoutTransform, 233, 239, 457
  - ListBox.ItemsSource, 274
  - ListBox.SelectedItem, 108
  - MinimalChildSize, 267
  - Minimum, 30
  - ObjectInstance, 274
  - Opacity, 123
  - Priorytet, 496
  - RenderTransform, 233, 234, 457

własność

- Root, 189
- RowDefinitions, 221
- ScaleX, 242
- SelectedText, 55
- Source, 91
- StrokeThickness, 457
- TextDecorations, 63
- TextWrapping, 56
- UriSource, 476
- VerticalAlignment, 209
- VerticalContentAlignment, 209
- Visibility, 56, 417
- zależności, dependency property, 339

WPF, Windows Presentation Foundation, 15

wrapper, 48

współrzędne teksturowania, 482

wstążka, 73

- menu aplikacji, 73
- pasek szybkiego dostępu, 76
- zakładki, 77

wybór

- czcionki, 64, 437, 439
- elementu, 563
- formatu danych, 546
- koloru, 61
- pliku, 387

wyciek pamięci, 31

wygląd czcionki, 439

wygładzanie, 244

wyjątek NullReferenceException, 301

wykrycie trybu projektowania, 461

wypełnienie Fill, 299

wyświetlenie warunkowe okna dialogowego, 385

wywoływania cykliczne, 490

wyzwalacz, 227

- stylu, 239
- transformacji, 239

wzorzec

- MVVM, 283, 370
- widoku autonomicznego, 19

## X

XAML, 207

- użycie stylów, 516

XML, Extensible Markup Language, 183

- atributy, 184
- deklaracja, 183
- elementy, 184
- komentarze, 184
- pobieranie wartości z elementów, 188
- przechowywanie danych, 183, 499
- struktura pliku, 190
- tworzenie plików, 185
- zapisywanie danych, 196

## Z

zachowania, behaviors, 337

- własności zależności, 341

zachowanie doczepione, attached behavior, 343

zaczepianie kontrolki, 25

zagnieżdżanie przycisków, 89, 94

zakładka Events, 29

zakładki wstążki, 77

zamykanie

- aplikacji, 50
- okna, 53, 122, 334
- zapisywanie danych, 517

zapisywanie

- danych, 517
- danych do obiektów, 193
- do pliku XML, 196
- tekstu do pliku, 49
- ustawień, 36

zapytanie, request, 535

- LINQ, 499

zasada DRY, 54

zasoby aplikacji, 228

zawijanie wierszy, 442

zdarzenia, 26, 331

- rejestrowanie, 537

zdarzenia trasowane, routed events, 87

- kontrola przepływu, 90

zdarzenie

- Click, 54
- Closed, 186, 332
- Closing, 123
- CollectionChanged, 507
- DragEnter, 103, 109
- DragOver, 103

Executed, 97  
KeyDown, 334  
MediaFailed, 199  
MouseDown, 103, 121  
MouseMove, 121  
MouseUp, 121  
PhonemeReached, 204  
PreviewKeyDown, 67, 122  
PreviewMouseLeftButtonDown, 107, 114  
PreviewMouseWheel, 433  
PropertyChanged, 307, 308  
SpeakCompleted, 204  
SpeakProgress, 204  
TextChanged, 51  
Window.Closed, 323  
Zegar, 457  
  analogowy, 461  
  cyfrowy, 460  
  efekty, 469  
  model widoku, 458  
  widok, 460, 461

zmiana  
  czcionki, 65, 400  
  etykiety przycisku, 88  
  ikony okna, 42  
  koloru, 28, 31  
  koloru tła, 90  
  kształtu okna, 465  
  nazwy okna, 42  
  położenia shaderów, 481  
  pozycji suwaka, 27  
  rozmiaru czcionki, 432

## Ż

źródło danych, 273



# PROGRAM PARTNERSKI

— GRUPY HELION —

- 
1. ZAREJESTRUJ SIĘ
  2. PREZENTUJ KSIĄŻKI
  3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion**

## Programuj w C# i odkryj możliwości Visual Studio 2017!



Microsoft Visual Studio to znakomite środowisko programistyczne, w którym bardzo wydajnie, a przy tym z przyjemnością, można projektować i testować aplikacje desktopowe, mobilne i webowe w kilku językach programowania, z wykorzystaniem bogatego zbioru bibliotek oraz interesujących dodatków dostępnych w usłudze NuGet, wydawnie skracających czas pracy. Jeśli chcesz poznać możliwości tego pakietu dla języka C# i nauczyć się tworzyć kompletne, dobrze przemyślane i doskonale działające aplikacje, nie zwlekaj, tylko czym prędzej sięgnij po tę książkę.

Pomoże Ci ona zorientować się, jakie techniki oraz jakie wzorce projektowe będą najlepsze dla Twoich projektów aplikacji desktopowych Windows Presentation Foundation (WPF). Krok po kroku przejdziesz przez proces ich tworzenia z wykorzystaniem narzędzi na różnym poziomie zaawansowania. Poznasz kontrolki WPF i wzorzec architektoniczny MVVM (model – widok – model widoku). Opanujesz język XAML służący do opisu graficznego interfejsu użytkownika w WPF, w szczególności style, transformacje i animacje, a także szablony kontrolki oraz szablony danych. W tej książce jest również miejsce na zagadnienia związane z wykorzystaniem shaderów, tj. prostych programów napisanych w języku HLSL, które na bieżąco sprawdzają, jak kontrolki są rysowane przez karty graficzne, oraz na omówienie przykładu, w którym aplikacja pobiera dane z prywatnej chmury w postaci usługi REST działającej na serwerze. A gdy poznasz już wszystkie przykłady i zrobisz zaproponowane na ich końcach zadania, ze zdumieniem stwierdzisz, że umiesz zbudować porządną aplikację desktopową WPF!

### Twórz fantastyczne aplikacje z Visual Studio i C# w WPF!

- Pierwsza aplikacja i przegląd kontrolki WPF
- Języki C# i XAML, a nawet odrobinę HLSL
- Budowanie złożonych kontrolki, szablony
- Style, transformacje i animacje
- Pojemniki i listy oraz prezentacja danych w WPF
- Wzorzec MVVM: model – widok – model widoku
- Wiązania, konwersja danych w wiązaniach, polecenia
- Zachowania, własności zależności i własności doczepione
- Testy jednostkowe
- Rozwiązanie problemu okien dialogowych w MVVM

**Helion**



helion.pl



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

Sprawdź nasze szkolenia!

SZKOLENIA



AKADEMIA IT & BUSINESS

WWW.SZKOLENIA.HELION.PL

**KOD KORZYŚCI**  
Sięgnij po więcej! ▶



ISBN 978-83-283-3825-8



9 788328 338258

**INFORMATYKA W NAJLEPSZYM WYDANIU**

Cena: 89,00 zł