

» Idź do

- Spis treści
- Przykładowy rozdział

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2008

Visual Basic 2008. Warsztat programisty

Autor: Rod Stephens

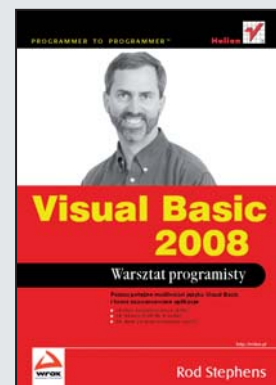
Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-1842-2

Tytuł oryginału: [Visual Basic 2008](#)

[Programmer's Reference](#)

Format: 172x245, stron: 1224



Poznaj potężne możliwości języka Visual Basic i twórz zaawansowane aplikacje

- Jak pisać zaawansowane programy?
- Jak dobrać kontrolki do zadań?
- Jak dzielić program na mniejsze części?

Visual Basic jest niezwykle wydajnym językiem, pozwalającym na tworzenie zaawansowanych aplikacji, m.in. dzięki wykorzystywaniu jednego z najpotężniejszych środowisk programistycznych, jakie kiedykolwiek powstały – Visual Studio. Środowisko to udostępnia narzędzia takie, jak edytory graficzne czy funkcja automatycznego uzupełniania, które sprawiają, że budowa aplikacji staje się intuicyjna i znacznie sprawniejsza. Najnowsza wersja tego języka – Visual Basic 2008 – oferuje także opcje formularzy XAML i nowe kontrolki, a ponadto z aplikacją zintegrowano SQL Server 2005 Compact Edition, dzięki czemu pojawiło się wbudowane wsparcie dla języka LINQ w komunikacji z bazą danych.

Książka „Visual Basic 2008. Warsztat programisty” jest doskonałym podręcznikiem zarówno dla początkujących, jak i zaawansowanych programistów. Zawiera ona opis technologii i środowiska programistycznego, a także wiele przykładów i porad z dokładnie przedstawionym oraz przetestowanym kodem, zgodnym z wersją 2008 tego języka. Przykłady te pozwalają dogłębnie zrozumieć wszelkie jego zaawansowania. Z książki dowiesz się, jak korzystać z formularzy i kontrolki oraz zwiększać funkcjonalność istniejących klas i obsługiwać błędy. Nauczysz się rysować obrazy przy użyciu interfejsu urządzenia graficznego, a także stosować wiele różnych obiektów, aby budować zaawansowane i nowoczesne aplikacje.

- Edytor kodu Visual Basic
- Struktura programu i modułu
- Typy danych, zmienne i stałe
- Operatory
- Procedury i funkcje
- Instrukcje sterujące
- Obsługa błędów
- Składnia zapytań i funkcje LINQ
- Tworzenie niestandardowych kontrolki
- Tworzenie nazw i klasy kolekcyjne
- Grafika i tekst
- Przetwarzanie obrazów
- Drukowanie i raportowanie
- Obiekty systemu plików
- Windows Communication Foundation

Bądź profesjonalistą – szlifuj swój warsztat programisty!

Spis treści

O autorze	19
Podziękowania	20
Wstęp	21
Część I IDE	37
Rozdział 1. Wprowadzenie do IDE	39
Wygląd IDE	40
Konfiguracje IDE	41
Projekty i rozwiązania	42
Uruchamianie IDE	43
Tworzenie projektu	44
Zapisywanie projektu	47
Podsumowanie	49
Rozdział 2. Menu, paski narzędzi i okna	51
Menu	51
Menu File	52
Menu Edit	55
Menu View	56
Menu Project	57
Menu Build	62
Menu Debug	63
Menu Data	63
Menu Format	64
Menu Tools	65
Menu Test	69
Menu Window	70
Menu Community	71
Menu Help	71

Paski narzędzi	71
Okna podrzędne	72
Okno Toolbox	73
Okno Properties	75
Podsumowanie	76
Rozdział 3. Konfiguracja IDE	77
Dodawanie poleceń	77
Usuwanie poleceń	79
Modyfikowanie poleceń	79
Tworzenie skrótów klawiszowych	81
Podsumowanie	82
Rozdział 4. Projektant formularzy Windows	83
Ustawianie opcji projektanta	83
Dodawanie kontroltek	85
Zaznaczanie kontroltek	85
Kopiowanie kontroltek	87
Przenoszenie i zmiana rozmiaru kontroltek	88
Aranżacja kontroltek	88
Ustawianie własności kontroltek	88
Grupowe ustawianie własności	89
Ustawianie własności kilku kontroltek	89
Używanie inteligentnych znaczników	90
Polecenia w oknie Properties	91
Dodawanie kodu do kontroltek	91
Podsumowanie	93
Rozdział 5. Projektant WPF	95
Ostrzeżenie o wczesnej wersji	95
Wprowadzenie do okien projektanta	97
Dodawanie kontroltek	98
Zaznaczanie kontroltek	99
Kopiowanie kontroltek	100
Przenoszenie i zmiana rozmiaru kontroltek	100
Ustawianie własności	102
Grupowe ustawianie własności kontroltek	102
Dodawanie kodu do kontroltek	103
Podsumowanie	103
Rozdział 6. Edytor kodu Visual Basica	105
Ikony na marginesie	106
Funkcja Outlining	107
Chmurki	109
Funkcja IntelliSense	109
Kolorowanie kodu i wyróżnianie	111
Fragmenty kodu, które nadają się do użycia w innych aplikacjach	113
Używanie fragmentów kodu	114
Tworzenie fragmentów kodu do użytku w innych programach	115
Edytor kodu w czasie działania programu	117
Podsumowanie	118

Rozdział 7. Usuwanie błędów	119
Menu Debug	119
Menu Debug — podmenu Windows	122
Okno Breakpoints	126
Okna Command i Immediate	127
Podsumowanie	129

Część II Wstęp do języka Visual Basic

Rozdział 8. Wybieranie kontrolki Windows Forms

Przeglądanie kontrolki	133
Wybieranie kontrolki	138
Kontrolki kontenery	139
Wybór opcji	141
Wprowadzanie danych	142
Wyświetlanie danych	142
Informowanie użytkownika	143
Inicjowanie akcji	144
Wyświetlanie grafiki	146
Wyświetlanie okien dialogowych	146
Wspieranie innych kontrolki	147
Kontrolki niestandardowe	147
Podsumowanie	148

Rozdział 9. Używanie kontrolki Windows Forms

Kontrolki i komponenty	151
Tworzenie kontrolki	153
Tworzenie kontrolki w czasie projektowania	153
Wstawianie kontrolki do kontenerów	154
Tworzenie kontrolki w czasie działania programu	155
Własności	157
Własności w czasie projektowania	157
Własności w czasie działania programu	162
Przydatne własności kontrolki	164
Własności położenia i rozmiaru	168
Metody	169
Zdarzenia	169
Tworzenie procedur obsługi zdarzeń w czasie projektowania	170
Słowo kluczowe WithEvents	171
Tworzenie procedur obsługi zdarzeń w czasie działania programu	172
Zdarzenia tablic kontrolki	173
Zdarzenia walidacji	173
Podsumowanie	178

Rozdział 10. Formularze Windows

Przezroczystość	180
Okna O programie, ekrany powitalne i formularze logowania	183
Kursory	185
Ikony	187
Ikony aplikacji	188
Ikony powiadomienia	188

Własności adoptowane przez kontrolki potomne	189
Metody przywracające ustawienia domyślne własności	190
Przestanianie procedury WndProc	190
SDI i MDI	192
Aplikacje MDI	193
Zdarzenia MDI	196
MDI a SDI	198
Listy ostatnio używanych plików	199
Okna dialogowe	201
Kreatory	203
Podsumowanie	204
Rozdział 11. Wybieranie kontrolki WPF	205
Przegląd kontrolki	206
Kontrolki kontenery	207
Wybieranie opcji	209
Wprowadzanie danych	210
Wyświetlanie danych	210
Informacje dla użytkownika	210
Inicjowanie akcji	211
Prezentowanie grafiki i mediów	212
Elementy nawigacji	213
Zarządzanie dokumentami	214
Cyfrowy atrament	214
Podsumowanie	215
Rozdział 12. Używanie kontrolki WPF	217
Koncepty WPF	217
Oddzielenie interfejsu użytkownika od kodu źródłowego	218
Hierarchie kontrolki WPF	219
WPF w IDE	220
Edytowanie kodu XAML	221
Edytowanie kodu Visual Basica	224
Język XAML	227
Obiekty	228
Zasoby	230
Style	232
Szablony	233
Transformacje	235
Animacje	236
Rysowanie figur geometrycznych	239
Procedury w WPF	244
Dokumenty	249
Dokumenty płynne	249
Dokumenty o ustalonym formatowaniu	251
Dokumenty XPS	252
Podsumowanie	252
Rozdział 13. Okna WPF	255
Aplikacje okienkowe	255
Aplikacje na stronach	258
Aplikacje przeglądarkowe	258
Aplikacje ramkowe	260

Aplikacje PageFunction	261
Kreatory	264
Podsumowanie	268
Rozdział 14. Struktura programu i modułu	269
Ukryte pliki	269
Struktura plików z kodem	274
Regiony kodu	275
Kompilacja warunkowa	276
Przestrzenie nazw	284
Typograficzne elementy kodu	286
Komentarze	286
Komentarze XML	287
Znak kontynuacji wiersza	290
Łączenie wierszy	291
Etykiety wierszy	291
Podsumowanie	292
Rozdział 15. Typy danych, zmienne i stałe	293
Typy danych	294
Znaki oznaczające typy	296
Konwersja typów danych	299
Konwersja zawężająca	299
Metody analizy typów danych	301
Konwersja rozszerzająca	302
Deklarowanie zmiennych	302
Lista atrybutów	303
Dostępność	303
Słowo kluczowe Shared	305
Słowo kluczowe Shadows	305
Słowo kluczowe ReadOnly	308
Słowo kluczowe Dim	308
Słowo kluczowe WithEvents	309
Nazywanie zmiennych	311
Lista wartości brzegowych	311
Słowo kluczowe New	313
Określanie typu zmiennej	313
Wyrażenie inicjujące	314
Deklarowanie kilku zmiennych za jednym razem	318
Opcje Option Explicit i Option Strict	319
Zasięg zmiennych	322
Zasięg blokowy	322
Zasięg proceduralny	323
Zasięg modułowy	324
Zasięg przestrzeni nazw	324
Ograniczanie zasięgu	325
Deklaracje parametrów	326
Procedury własności	328
Typy wyliczeniowe	330
Anonimowe typy danych	333
Typy dopuszczające wartość pustą	334

Stałe	335
Dostępność	335
Typ stałej	335
Wyrażenie_inicjujące	336
Delegaty	336
Konwencje nazewnicze	338
Podsumowanie	340
Rozdział 16. Operatory	341
Operatory arytmetyczne	341
Operatory konkatencji	342
Operatory porównywania	343
Operatory logiczne	345
Operatory bitowe	346
Priorytety operatorów	347
Operatory przypisania	348
Klasa StringBuilder	349
Typ Date i klasa TimeSpan	351
Przeciążanie operatorów	354
Operatory z typami dopuszczającymi wartość pustą	357
Podsumowanie	358
Rozdział 17. Podprocedury i funkcje	359
Podprocedury	359
Lista atrybutów	360
Tryb dziedziczenia	364
Dostępność	365
Nazwa podprocedury	366
Parametry	367
Klauzula Implements interfejs.podprocedura	374
Instrukcje	375
Funkcje	376
Procedury własności	377
Metody rozszerzające	378
Funkcje lambda	379
Rozluźnione delegaty	381
Metody częściowe	384
Podsumowanie	386
Rozdział 18. Instrukcje sterujące	387
Instrukcje decyzyjne	387
Jednowierszowa instrukcja If Then	387
Wielowierszowa instrukcja If Then	389
Instrukcja Select Case	390
Wartości wyliczeniowe	392
Instrukcja IIF	393
Instrukcja If	395
Instrukcja Choose	395
Instrukcje pętlowe	397
Pętla For Next	397
Różne typy zmiennej pętlowej pętli For Next	400
Pętla For Each	401

Enumeratory	403
Iteratory	405
Instrukcje Do Loop	405
Pętla While End	406
Instrukcje Exit i Continue	407
Instrukcja GoTo	407
Podsumowanie	410
Rozdział 19. Obsługa błędów	413
Błędy a nieplanowane sytuacje	413
Przechwytywanie błędów	414
Przechwytywanie niespodziewanych zdarzeń	416
Globalna obsługa błędów	418
Strukturalna obsługa błędów	419
Obiekty wyjątków	422
Obiekty klasy StackTrace	423
Zgłaszanie wyjątków	423
Niestandardowe klasy wyjątków	426
Klasyczna obsługa błędów w Visual Basicu	427
Instrukcja On Error GoTo	428
Instrukcja On Error Resume Next	429
Instrukcja On Error GoTo 0	430
Instrukcja On Error GoTo -1	430
Tryb obsługi błędów	432
Strukturalna a klasyczna obsługa błędów	432
Obiekt Err	434
Debugowanie	435
Podsumowanie	436
Rozdział 20. Kontrolki i obiekty baz danych	437
Automatyczne łączenie z danymi	437
Łączenie ze źródłami danych	438
Wstawianie kontrolek danych na formularz	441
Obiekty tworzone automatycznie	445
Inne obiekty danych	447
Przeglądanie danych	448
Obiekty połączenia	449
Obiekty transakcji	452
Adaptery danych	454
Obiekty poleceń	459
Obiekt klasy DataSet	460
Klasa DataTable	464
Klasa DataRow	467
Klasa DataColumn	468
Klasa DataRelation	470
Ograniczenia	472
Klasa DataView	475
Klasa DataRowView	478
Proste wiązanie danych	479
Klasa CurrencyManager	480
Złożone wiązanie danych	483
Podsumowanie	487

Rozdział 21. LINQ	489
Wprowadzenie do LINQ	491
Podstawy składni zapytań LINQ	493
Klauzula From	493
Klauzula Where	494
Klauzula Order By	495
Klauzula Select	496
Wykorzystanie wyników zapytań LINQ	498
Zaawansowana składnia zapytań LINQ	499
Słowo kluczowe Join	499
Klauzula Group By	501
Funkcje agregujące	504
Operacje na zbiorach	504
Ograniczanie ilości zwracanych wyników	505
Funkcje LINQ	506
Metody rozszerzające LINQ	508
Zapytania oparte na metodach	508
Zapytania oparte na metodach i funkcje lambda	510
Rozszerzanie LINQ	512
LINQ to Objects	514
LINQ to XML	515
Literały XML	515
LINQ w XML	516
LINQ z XML	518
LINQ to ADO.NET	521
LINQ to SQL i LINQ to Entities	521
LINQ to DataSet	522
Podsumowanie	525
Rozdział 22. Tworzenie niestandardowych kontroltek	527
Kontrolki niestandardowe — informacje ogólne	528
Tworzenie projektu budowy kontrolki	529
Ustawianie ikony dla okna Toolbox	529
Testowanie w oknie UserControl Test Container	530
Tworzenie projektu testowego	531
Testowanie kontrolki	532
Implementowanie własności, metod oraz zdarzeń	532
Przypisywanie atrybutów	534
Zachowanie w czasie projektowania i działania programu	536
Kontrolki pochodne	536
Przesłanie funkcjonalności klasy nadrzędnej	539
Ukrywanie funkcjonalności klasy nadrzędnej	540
Kontrolki złożone	541
Budowanie kontroltek od początku	543
Komponenty	544
Niewidoczne kontrolki	545
Wybieranie klasy kontrolki	546
Kontrolki i komponenty w projektach wykonywalnych	547
Klasa UserControl w projektach wykonywalnych	547
Dziedziczące kontrolki UserControl w projektach wykonywalnych	548
Klasa Control w projektach wykonywalnych	548

Kontrolki dziedziczone w projektach wykonywalnych	549
Komponenty w projektach wykonywalnych	549
Zabezpieczenia niestandardowych komponentów	549
Asemblacje z silną nazwą	550
Organ podpisujący	552
Podsumowanie	552
Rozdział 23. Operacje typu „przeciągnij i upuść” oraz schowek	555
Zdarzenia typu „przeciągnij i upuść”	556
Prosty przykład	557
Sprawdzanie, jakie typy danych są dostępne	559
Przeciąganie w obrębie aplikacji	560
Przymywanie upuszczonych plików	561
Przeciąganie obiektów	562
Zmienianie nazw formatów	565
Przeciąganie kilku typów danych	566
Używanie schowka	568
Podsumowanie	571
Rozdział 24. Funkcja Kontrola konta użytkownika	573
Funkcja Kontrola konta użytkownika — informacje ogólne	573
Projektowanie programów pod kątem funkcji Kontrola konta użytkownika	575
Zwiększanie uprawnień programów	578
Użytkownik	578
Program wywołujący	578
Program wywoływany	579
Podsumowanie	580
Część III Programowanie obiektowe	583
Rozdział 25. Podstawy programowania obiektowego	585
Klasy	585
Hermetyzacja	588
Dziedziczenie	589
Hierarchie dziedziczenia	590
Uszczegółowienie i uogólnienie	591
Związki „ma” i „jest”	593
Dodawanie i modyfikowanie własności klas	594
Dziedziczenie interfejsów	596
Polimorfizm	597
Przeciążanie	598
Metody rozszerzające	599
Podsumowanie	600
Rozdział 26. Klasy i struktury	603
Klasy	603
Lista atrybutów	604
Słowo kluczowe Partial	604
Dostępność	605
Słowo kluczowe Shadows	606
Dziedziczenie	607

Klauzula Of lista_typów	609
Klauzula Inherits klasa_nadrzędna	609
Klauzula Implements interfejs	610
Struktury	613
Struktury nie mogą dziedziczyć	614
Struktury są typami wartościowymi	614
Zajmowanie pamięci	615
Serta i stos	615
Przypisywanie obiektów	617
Przekazywanie parametrów	618
Opakowywanie i odpakowywanie	619
Tworzenie egzemplarzy klas — szczegóły	620
Tworzenie egzemplarzy struktur — szczegóły	622
Zbieranie nieużytków	624
Metoda Finalize	625
Metoda Dispose	628
Stałe, własności i metody	629
Zdarzenia	631
Deklarowanie zdarzeń	631
Zgłaszanie zdarzeń	633
Przechwytywanie zdarzeń	634
Deklarowanie niestandardowych zdarzeń	636
Zmienne współdzielone	640
Metody współdzielone	641
Podsumowanie	644
Rozdział 27. Przestrzenie nazw	645
Instrukcja Imports	646
Importowanie automatyczne	648
Aliasy przestrzeni nazw	649
Elementy przestrzeni nazw	650
Główna przestrzeń nazw	650
Tworzenie przestrzeni nazw	651
Klasy, struktury i moduły	652
Rozwiązywanie przestrzeni nazw	653
Podsumowanie	656
Rozdział 28. Klasy kolekcyjne	657
Czym jest kolekcja	657
Tablice	658
Wymiary tablicy	659
Dolne granice	660
Zmienianie rozmiaru	661
Prędkość	661
Inne cechy klasy Array	662
Kolekcje	665
Klasa ArrayList	665
Klasa StringCollection	667
Kolekcje ze ścisłą kontrolą typów	667
Kolekcje tylko do odczytu ze ścisłą kontrolą typów	669
Klasa NameValueCollection	670

Słowniki	671
Klasa ListDictionary	672
Klasa Hashtable	673
Klasa HybridDictionary	674
Słowniki ze ścisłą kontrolą typów	675
Inne klasy ze ścisłą kontrolą typów	676
Klasa StringDictionary	677
Klasa SortedList	677
Klasa CollectionsUtil	677
Stosy i kolejki	678
Stos	678
Kolejka	679
Klasy ogólne	681
Podsumowanie	683
Rozdział 29. Typy ogólne	685
Zalety klas ogólnych	685
Definiowanie klas ogólnych	686
Konstruktory klas ogólnych	688
Więcej niż jeden typ	688
Typy ograniczone	690
Używanie typów ogólnych	692
Aliasy Imports	693
Klasy pochodne	693
Predefiniowane klasy ogólne	694
Metody ogólne	694
Typy ogólne a metody rozszerzające	695
Podsumowanie	696
Część IV Grafika	699
Rozdział 30. Podstawy rysowania	701
Przegląd technik rysowania	701
Przestrzenie nazw obiektów rysujących	704
Przestrzeń nazw System.Drawing	704
Przestrzeń nazw System.Drawing.Drawing2D	705
Przestrzeń nazw System.Drawing.Imaging	708
Przestrzeń nazw System.Drawing.Text	709
Przestrzeń nazw System.Drawing.Printing	710
Klasa Graphics	712
Metody rysujące	712
Metody wypełniające	717
Inne własności i metody obiektów klasy Graphics	717
Wyładzanie krawędzi	720
Podstawy transformacji	722
Transformacje — techniki zaawansowane	726
Zapisywanie i przywracanie stanu obiektów klasy Graphics	729
Zdarzenia rysowania	731
Podsumowanie	733

Rozdział 31. Pędzle, pióra i ścieżki	735
Klasa Pen	735
Własność Alignment	738
Własność CompoundArray	739
Niestandardowe końcówki linii	740
Transformacje pióra	741
Klasa Brush	743
Klasa SolidBrush	744
Klasa TextureBrush	744
Klasa HatchBrush	746
Klasa LinearGradientBrush	747
Klasa PathGradientBrush	751
Obiekty klasy GraphicsPath	756
Kwestia zbierania nieużytków	759
Podsumowanie	762
Rozdział 32. Tekst	763
Rysowanie tekstu	764
Formatowanie tekstu	764
Własność FormatFlags	766
Tabulatory	769
Przycinanie	770
Metoda MeasureString	771
Metryka fontu	774
Podsumowanie	777
Rozdział 33. Przetwarzanie obrazów	779
Klasa Image	779
Klasa Bitmap	781
Ładowanie obiektów klasy Bitmap	782
Zapisywanie obiektów klasy Bitmap	783
Implementowanie własności AutoRedraw	785
Operacje na pojedynczych pikselach	787
Obiekty klasy Metafile	791
Podsumowanie	794
Rozdział 34. Drukowanie	795
Jak nie drukować	795
Podstawy drukowania	796
Drukowanie tekstu	800
Ustawianie wydruku na środku	806
Dopasowywanie obrazów na stronie	809
Upraszczenie rysowania i drukowania	810
Podsumowanie	813
Część V Interakcja ze środowiskiem	815
Rozdział 35. Konfiguracja i zasoby	817
Przestrzeń nazw My	818
Obiekt Me a przestrzeń nazw My	818
Najważniejsze sekcje przestrzeni nazw My	819

Środowisko	819
Ustawianie zmiennych środowiskowych	820
Funkcja Environ	821
Obiekt System.Environment	821
Rejestr	823
Rodzime funkcje Visual Basica	824
Przestrzeń nazw My.Computer.Registry	826
Pliki konfiguracji	828
Pliki zasobów	832
Zasoby aplikacji	833
Używanie zasobów aplikacji	833
Zasoby wbudowane	835
Zasoby zewnętrzne	836
Zasoby lokalizacyjne	837
Klasa ComponentResourceManager	838
Klasa Application	841
Własności klasy Application	841
Metody klasy Application	843
Zdarzenia klasy Application	844
Podsumowanie	846
Rozdział 36. Strumienie	847
Klasa Stream	848
Klasa FileStream	849
Klasa MemoryStream	850
Klasa BufferedStream	851
Klasy BinaryReader i BinaryWriter	851
Klasy TextReader i TextWriter	853
Klasy StringReader i StringWriter	854
Klasy StreamReader i StreamWriter	855
Metody OpenText, CreateText oraz AppendText	856
Niestandardowe klasy strumieniowe	857
Podsumowanie	858
Rozdział 37. Obiekty systemu plików	859
Uprawnienia	859
Metody Visual Basica	860
Metody plikowe	860
Metody systemu plików	862
Pliki z dostępem sekwencyjnym	862
Pliki z dostępem swobodnym	863
Pliki z dostępem binarnym	865
Klasy platformy .NET Framework	866
Klasa Directory	866
Klasa File	867
Klasa DriveInfo	869
Klasa DirectoryInfo	870
Klasa FileInfo	871
Klasa SystemInfo	873
Klasa FileSystemWatcher	873
Klasa Path	875

Obiekt My.Computer.FileSystem	877
Własność My.Computer.FileSystem.SpecialDirectories	879
Podsumowanie	879
Rozdział 38. Windows Communication Foundation	881
Ostrzeżenie	882
Konceptje WCF	882
Przykład użycia biblioteki WCF	883
Budowa wstępnej wersji usługi	883
Tworzenie projektu QuoteService	886
Testowanie usługi QuoteService	888
Tworzenie klienta QuoteClient	889
Hostowanie usługi	890
Podsumowanie	891
Rozdział 39. Przydatne przestrzenie nazw	893
Przestrzenie nazw wysokiego poziomu	894
Przestrzeń nazw Microsoft	894
Przestrzeń nazw System	895
Zaawansowane przykłady	896
Wyrażenia regularne	897
XML	899
Kryptografia	901
Refleksja	905
Direct3D	909
Podsumowanie	916
Dodatki	917
Dodatek A Własności, metody i zdarzenia kontrolki	919
Dodatek B Deklaracje zmiennych i typy danych	929
Dodatek C Operatory	937
Dodatek D Deklarowanie podprocedur i funkcji	943
Dodatek E Instrukcje sterujące	947
Dodatek F Obsługa błędów	953
Dodatek G Komponenty i kontrolki Windows Forms	955
Dodatek H Kontrolki WPF	1035
Dodatek I Dodatki Power Pack	1041
Dodatek J Obiekty formularzy	1045
Dodatek K Klasy i struktury	1059

Dodatek L LINQ	1063
Dodatek M Typy ogólne	1073
Dodatek N Grafika	1077
Dodatek O Przydatne klasy wyjątków	1089
Dodatek P Specyfikatory formatu daty i godziny	1093
Dodatek Q Inne specyfikatory formatu	1097
Dodatek R Klasa Application	1103
Dodatek S Przestrzeń nazw My	1107
Dodatek T Strumienie	1121
Dodatek U Klasy systemu plików	1127
Dodatek V Indeks przykładów	1143
Skorowidz	1165

8

Wybieranie kontroltek Windows Forms

Kontrolki są niezwykle ważną częścią każdej interaktywnej aplikacji. Dostarczają użytkownikom informacje (Label, ToolTip, TreeView, PictureBox) oraz organizują je, dzięki czemu łatwiej jest je zrozumieć (GroupBox, Panel, TabControl). Umożliwiają wprowadzanie danych (TextBox, RichTextBox, ComboBox, MonthCalendar), wybieranie opcji (RadioButton, CheckBox, ListBox), kontrolowanie aplikacji (Button, MenuStrip, ContextMenuStrip) oraz uzyskiwanie dostępu do obiektów znajdujących się poza programem (OpenFileDialog, SaveFileDialog, PrintDocument, PrintPreviewDialog). Niektóre kontrolki oferują wsparcie dla innych kontroltek (ImageList, ToolTip, ContextMenuStrip, ErrorProvider).

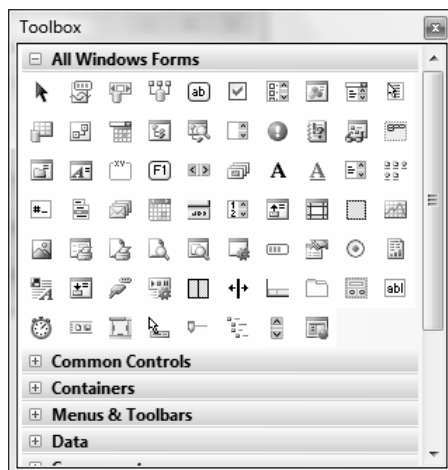
W tym rozdziale znajduje się tylko bardzo krótki opis standardowych kontroltek Windows Forms oraz kilka wskazówek pomagających w podjęciu decyzji, której z nich użyć w określonym celu. Znacznie więcej informacji na ten temat (najważniejsze własności, metody i zdarzenia) zawiera Dodatek G.

Przeglądanie kontroltek

Na rysunku 8.1 przedstawiono okno Visual Basica *Toolbox* ze standardowymi kontrolkami Windows Forms. Ponieważ pozwala ono zarówno na dodawanie, jak i usuwanie kontroltek, na Twoim komputerze może zawierać nieco inny ich zestaw.

Rysunek 8.1.

Visual Basic
udostępnia dużą
liczbę
standardowych
kontrolki Windows
Forms



W poniższej tabeli znajdują się krótkie opisy kontrolki widocznych na rysunku 8.1. Zachowano kolejność kontrolki na rysunku (zaczynając od pierwszego rzędu na górze i patrząc od lewej do prawej).

Kontrolka	Przeznaczenie
Rząd 1	
Pointer	Jest to narzędzie wskaźnika, a nie kontrolka. Kliknięcie go spowoduje odznaczenie wszystkich kontrolki zaznaczonych na formularzu. Później będzie można zaznaczyć nowe.
BackgroundWorker	Wykonuje zadanie asynchronicznie, a kiedy skończy, powiadamia program główny.
BindingNavigator	Tworzy interfejs użytkownika do nawigacji przez źródło danych. Udostępnia na przykład przyciski pozwalające użytkownikowi poruszanie się do przodu i wstecz przez dane, a także dodawanie i usuwanie rekordów itd.
BindingSource	Hermetyzuje źródło danych formularza i udostępnia metody nawigacji po nich.
Button	Prosty przycisk. Kiedy użytkownik go kliknie, program wykona jakąś akcję.
CheckBox	Pole do zaznaczania i czyszczenia przez użytkownika.
CheckedListBox	Lista elementów z polami wyboru, które użytkownik może zaznaczać i czyścić.
ColorDialog	Umożliwia użytkownikowi wybranie standardowego lub niestandardowego koloru.
ComboBox	Pole tekstowe z listą rozwijaną, w którym użytkownik może wprowadzić lub wybrać wartość tekstową.
ContextMenuStrip	Menu pojawiające się po kliknięciu kontrolki prawym przyciskiem myszy. Należy ustawić własność ContextMenuStrip dla dowolnej kontrolki; reszta odbędzie się automatycznie.

Kontrolka	Przeznaczenie
Rząd 2	
DataGridView	Kontrolka siatki, która pozwala względnie łatwo wyświetlać duże ilości skomplikowanych danych z powiązaniem hierarchicznymi lub typu sieciowego.
DataSet	Znajdujący się w pamięci schowek danych z własnościami podobnymi do relacyjnej bazy danych. Przechowuje obiekty reprezentujące tabele, które zawierają wiersze i kolumny. Może też przedstawiać wiele koncepcji baz danych, na przykład indeksy i powiązania kluczy obcych.
DateTimePicker	Pozwala użytkownikowi wybrać datę i godzinę w jednym z kilku stylów.
DirectoryEntry	Reprezentuje węzeł w hierarchii Active Directory.
DirectorySearcher	Przeszukuje hierarchię Active Directory.
DomainUpDown	Pozwala użytkownikowi przewijać listę opcji do wyboru za pomocą klikania strzałek skierowanych w górę i w dół.
ErrorProvider	Wyświetla obok kontrolki wskaźnik błędu.
EventLog	Daje dostęp do dzienników zdarzeń systemu Windows.
FileSystemWatcher	Powiadamia aplikację o zmianach w katalogu lub pliku.
FlowLayoutPanel	Wyświetla swoje kontrolki w rzędach lub kolumnach. Na przykład ustawia je jedną obok drugiej, aż skończy się miejsce w poziomie. Wtedy przechodzi do nowego rzędu.
Rząd 3	
FolderBrowserDialog	Pozwala na wybór folderu.
FontDialog	Pozwala ustawić własności pisma (nazwę czcionki, rozmiar, pogrubienie itd.).
GroupBox	Dla przejrzystości grupuje powiązane ze sobą kontrolki. Dodatkowo definiuje domyślną grupę RadioGroup dla przycisków RadioButton, które zawiera.
HelpProvider	Wyświetla pomoc dla kontrolki, które ją posiadają, jeśli użytkownik aktywuje kontrolkę i naciśnie klawisz <i>F1</i> .
HScrollBar	Poziomy pasek przewijania.
ImageList	Zawiera zbiór obrazów, których mogą używać inne kontrolki. Na przykład te wyświetlane przez kontrolkę TabControl na jej kartach są przechowywane w kontrolce ImageList. W swoim kodzie również można pobierać obrazy na własny użytek z listy ImageList.
Label	Wyświetla tekst, którego użytkownik nie może zmodyfikować, zaznaczyć kliknięciem ani przeciągnąć.
LinkLabel	Wyświetla etykietę, której niektóre części mogą być hiperłączami. Kiedy użytkownik kliknie hiperłącze, program wykona jakieś działania.
ListBox	Wyświetla listę elementów do wyboru dla użytkownika. W zależności od tego, jakie ustawienia własności tej kontrolki wprowadzono, będzie można zaznaczyć jeden lub kilka elementów.
ListView	Wyświetla listę elementów w jednym z czterech możliwych widoków — LargeIcon, SmallIcon, List oraz Details.

Kontrolka	Przeznaczenie
Rząd 4	
MaskedTextBox	Pole tekstowe, które wymaga, by wprowadzane w nim dane miały określony format (na przykład numer telefonu lub kod pocztowy).
MenuStrip	Reprezentuje główne menu, podmenu i elementy menu formularza.
MessageQueue	Umożliwia komunikację pomiędzy różnymi aplikacjami.
MonthCalendar	Wyświetla kalendarz, który pozwala użytkownikowi wybrać daty.
NotifyIcon	Wyświetla ikonę w zasobniku systemowym lub polu stanu.
NumericUpDown	Pozwala użytkownikowi zmienić liczbę za pomocą strzałek.
OpenFileDialog	Pozwala użytkownikowi wybrać plik do otwarcia.
PageSetupDialog	Pozwala użytkownikowi ustawić własności drukowanych stron. Na przykład umożliwia określenie podajnika papieru drukarki, rozmiaru strony, marginesów i orientacji (pionowa lub pozioma).
Panel	Kontener kontrolki. Za pomocą własności <code>Anchor</code> i <code>Dock</code> można sprawić, że kontrolka ta będzie zmieniać rozmiar — zawarte w niej kontrolki również ulegną modyfikacji. Kontener ten może automatycznie wyświetlać paski przewijania i definiować grupę <code>RadioButton</code> dla wszystkich przycisków <code>RadioButton</code> w niej się znajdujących.
PerformanceCounter	Daje dostęp do liczników wydajności systemu Windows.
Rząd 5	
PictureBox	Wyświetla obraz. Dodatkowo udostępnia przydatną powierzchnię do rysowania.
PrintDialog	Wyświetla standardowe okno dialogowe drukowania. Użytkownik może wybrać w nim drukarkę, strony do wydrukowania, a także ustawić opcje drukarki.
PrintDocument	Reprezentuje dane, które mają zostać wysłane do drukarki. Program może wykorzystywać ten obiekt do drukowania i wyświetlania widoków druku.
PrintPreviewControl	Wyświetla podgląd wydruku w obrębie jednego z formularzy aplikacji.
PrintPreviewDialog	Wyświetla podgląd wydruku w standardowym oknie dialogowym.
Process	Pozwala programowi na interakcję z procesami oraz na uruchamianie i zatrzymywanie ich.
ProgressBar	Wyświetla szereg kolorowych pasków, które wskazują postęp długiej operacji.
PropertyGrid	Wyświetla informacje o obiekcie w formacie podobnym do używanego w oknie <i>Properties</i> w czasie projektowania.
RadioButton	Reprezentuje jedną z wzajemnie wykluczających się opcji. Kiedy użytkownik kliknie taki przycisk, Visual Basic odznaczy wszystkie pozostałe kontrolki <code>RadioButton</code> w tej samej grupie. Grupy są definiowane przez kontrolki <code>GroupBox</code> i <code>Panel</code> oraz klasę <code>Form</code> .
ReportViewer	Wyświetla raport wygenerowany lokalnie lub zdalnie na serwerze raportowym. Zdalne przetwarzanie wymaga serwera raportującego Microsoft SQL Server 2005.

Kontrolka	Przeznaczenie
Rząd 6	
RichTextBox	Pole tekstowe, które obsługuje rozszerzenia tekstu wzbogaconego.
SaveFileDialog	Pozwala użytkownikowi nazwać plik, w którym program ma zapisać dane.
SerialPort	Reprezentuje port szeregowy oraz udostępnia metody do jego kontroli, odczytu z niego i zapisu do niego.
ServiceController	Reprezentuje usługę Windows i pozwala zarządzać usługami.
SplitContainer	Pozwala użytkownikowi przeciągać pionową lub poziomą linię, dzięki czemu można podzielić dostępną przestrzeń na dwie części.
Splitter	Udostępnia linię, którą użytkownik może przeciągać, aby podzielić dostępną przestrzeń pomiędzy dwie kontrolki. O aranżacji i rozmiarze tych ostatnich decydują własności Dock, kolejność na stosie i kontrolka Splitter. Kontrolka SplitterContainer automatycznie oddziela dwa kontenery, dzięki czemu zazwyczaj jest łatwiejsza w użyciu.
StatusStrip	Tworzy obszar (zazwyczaj w dolnej części formularza), w którym aplikacja może wyświetlać komunikaty o stanie, małe obrazki i inne wskaźniki stanu aplikacji.
TabControl	Wyświetla szereg kart związanych ze stroną, zawierających własne kontrolki. Użytkownik klika jedną z nich, aby wyświetlić odpowiednią stronę.
TableLayoutPanel	Wyświetla kontrolki na siatce.
TextBox	Wyświetla tekst, który użytkownik może zmienić.
Rząd 7	
Timer	Okresowo wyzwała zdarzenie. Program może w odpowiedzi na to zdarzenie podjąć jakieś działania.
ToolStrip	Wyświetla zbiór przycisków, menu rozwijanych i innych narzędzi, które pozwalają użytkownikowi kontrolować aplikację.
ToolStripContainer	Kontener, który pozwala kontrolce ToolStrip zadokować się przy niektórych lub wszystkich jego krawędziach. Kontrolkę ToolStripContainer można zadokować przy formularzu, aby umożliwić użytkownikowi zadokowanie kontrolki ToolStrip przy każdej krawędzi tego formularza.
ToolTip	Wyświetli chmurkę, jeśli użytkownik najedzie na odpowiednią kontrolkę.
TrackBar	Pozwala użytkownikowi przeciągać wskaźnik wzdłuż paska w celu wybrania wartości liczbowej.
TreeView	Wyświetla dane hierarchiczne w graficznej formie, która przypomina drzewo.
VScrollBar	Pionowy pasek przewijania.
WebBrowser	Przeglądarka internetowa w kontrolce. Tę ostatnią można umieścić na formularzu i za pomocą jej metod przejść do strony internetowej. Kontrolka ta wyświetla wyniki dokładnie tak samo, jak gdyby użytkownik używał samodzielnej przeglądarki. Jednym z pożytecznych jej zastosowań jest wyświetlanie pomocy internetowej.

Przykładowy program o nazwie *UseToolStripContainer*, który można pobrać z serwera FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/vb28wp.zip>), zawiera kontrolkę *ToolStripContainer* z dwiema kontrolkami *ToolStrip*. Można je przeciągać i dokować przy krawędziach kontrolki *ToolStripContainer*.

Szczegółowe opisy kontrolerek znajdują się w Dodatku G — „Komponenty i kontrolki Windows Forms”.

Wybieranie kontrolerek

Zapamiętanie wszystkich szczegółów dotyczących wymienionych kontrolerek jest bardzo trudne. Przy takiej liczbie narzędzi do wyboru czasami ciężko zdecydować, które z nich najlepiej się nada do wykonania określonego zadania.

Aby uprościć kod obsługujący błędy, powinno się wybierać kontrolki o jak największym zakresie funkcjonalności, które mogą wykonać dane zadanie. Bardziej restrykcyjne zmniejszają prawdopodobieństwo podania przez użytkownika nieprawidłowych danych.

Restrykcyjne kontrolki podnoszą też poziom bezpieczeństwa aplikacji. Wyświetlając listę opcji do wyboru, zamiast pozwalać użytkownikom wpisać, co im się podoba, program może chronić sam siebie przed problemami. Na przykład dwoma najczęściej spotykanymi rodzajami ataków na strony internetowe jest przepełnienie bufora, które polega na wpisaniu w polu tekstowym znacznie więcej tekstu niż potrzeba, a także SQL injection — wpisanie w polu tekstowym specjalnie spreparowanego kodu, mającego na celu zmianę działania bazy danych. Podanie zestawu opcji zamiast zezwolenia użytkownikowi na wpisanie, co chce, oddala ryzyko tych dwóch problemów.

Załóżmy na przykład, że użytkownik musi wybrać jeden z rozmiarów — Small, Medium lub Large. Można pozwolić mu wprowadzić wybraną opcję do pola tekstowego, ale w takim przypadku mógłby napisać na przykład *Nutria*. Program musiałby sprawdzić, czy użytkownik wpisał złe słowo, a jeśli tak — wyświetlić komunikat o błędzie. Możliwe też, że trzeba by zająć cenne miejsce na ekranie w celu pokazania listy opcji do wyboru.

Lepszym pomysłem byłoby użycie grupy kontrolerek *RadioButton* lub kontrolki *ComboBox* z własnością *DropDownStyle* ustawioną na *DropDownList*. Dzięki temu użytkownik będzie widział listę pozycji do wyboru i mógł wybrać tylko jedną z prawidłowych opcji. Jeśli program inicjuje kontrolki jakimiś wartościami, zamiast pozostawiać je niezdefiniowane, wie, że zawsze zaznaczona jest jakaś poprawna opcja.

W kolejnych podrozdziałach umieszczono opisy różnych kategorii kontrolerek. Znajdują się tam też wskazówki, kiedy ich używać.

Kontrolki kontenery

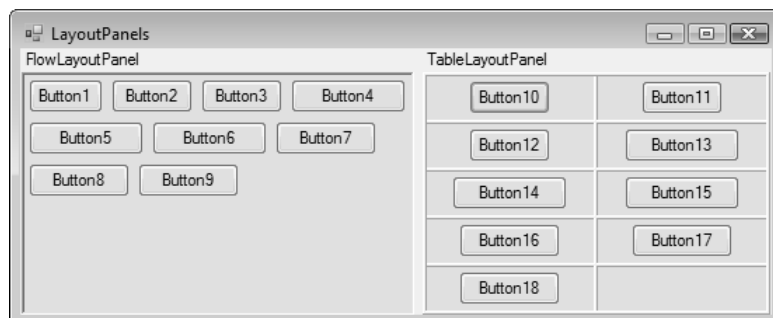
Te kontrolki zawierają, grupują i pomagają ustawiać inne kontrolki. Zaliczają się do nich `FlowLayoutPanel`, `TableLayoutPanel`, `GroupBox`, `Panel`, `TabControl` oraz `SplitContainer`.

Kontrolka `FlowLayoutPanel` ustawia znajdujące się w niej kontrolki w rzędach i kolumnach. Jeśli na przykład jej własność `FlowDirection` jest określona jako `LeftToRight`, ustawia ona swoje kontrolki w rzędach od lewej do prawej. Kiedy miejsce w jednym rzędzie się skończy, dana kontrolka przejdzie do kolejnego. Kontrolka `FlowLayoutPanel` jest szczególnie przydatna do tworzenia zestawów narzędzi. Warto jej używać również w sytuacjach, gdy trzeba wyświetlić jak najwięcej kontrolkek jednocześnie, a ich dokładne ustawienie nie jest zbyt ważne.

Kontrolka `TableLayoutPanel` wyświetla swoją zawartość na siatce. Wszystkie komórki w jednym wierszu mają tę samą wysokość, a wszystkie komórki w jednej kolumnie — jednakową szerokość. Natomiast kontrolka `FlowLayoutPanel` umieszcza kontrolki jedną obok drugiej, aż zapełni wiersz — wtedy przechodzi do nowego. Na rysunku 8.2 przedstawiono przykładowy program o nazwie *LayoutPanels* — można go pobrać z serwera FTP wydawnictwa Helion.

Rysunek 8.2.

Kontrolka `FlowLayoutPanel` umieszcza kontrolki blisko siebie, zaś `TableLayoutPanel` ustawia je na siatce



Kontrolka `GroupBox` znajduje zastosowanie w grupowaniu powiązanych ze sobą kontrolkek lub kontrolkek `RadioButton` w grupie (kontrolka `RadioButton` została opisana w podrozdziale „Wybór opcji”, który znajduje się w dalszej części tego rozdziału). Tworzy widoczne obramowanie i podpis, dzięki czemu użytkownik łatwiej domyśli się, o co chodzi w skomplikowanym formularzu.

Podstawową zasadą przy projektowaniu interfejsów użytkownika jest to, że można w jednym czasie ewaluować od pięciu do dziewięciu elementów. Lista pięciu czy sześciu opcji jest do zaakceptowania, ale spis, który zawiera kilkadziesiąt pozycji, może wprowadzać zamieszanie. Jeśli umieści się opcje w kategoriach oddzielonych od siebie wizualnie za pomocą kontrolki `GroupBox`, można sprawić, że interfejs będzie znacznie łatwiejszy do zrozumienia. Zamiast próbować zająć się kilkudziesięcioma opcjami do wyboru naraz, użytkownik może rozbić problem na mniejsze części i zająć się każdą grupą osobno.

Kontrolka `Panel` również może zawierać kontrolki `RadioButton` w grupie. Nie wyświetla jednak obramowania, przez co trzeba w jakiś inny sposób zaznaczyć, że dane przyciski należą do jednej grupy. Można na przykład użyć kilku kontrolki `Panel` w jednym rzędzie, z których każdy zawierałby kolumnę kontrolki `RadioButton`. Wtedy użytkownik mógłby wybrać po jednej opcji z każdej kolumny.

Jedną z najważniejszych właściwości kontrolki `Panel` jest jej zdolność do automatycznego wyświetlania pasków przewijania. Jeśli właściwość `AutoScroll` kontrolki `Panel` jest ustawiona na wartość `True`, a w kontrolce tej znajdzie się więcej treści, niż może ona pomieścić, automatycznie pojawiają się paski przewijania, aby użytkownik mógł zobaczyć całą jej zawartość. Jednak przewijanie w tę i z powrotem może być uciążliwe, dlatego nie jest to najlepszy wybór dla danych, które trzeba często przeglądać. Jeśli użytkownik musi ciągle przechodzić pomiędzy różnymi kontrolkami w przewijanym panelu, lepszym rozwiązaniem byłoby dla niego użycie kontrolki `TabControl`.

Kontrolka `TabControl` grupuje dane na kartach. Użytkownik może szybko przechodzić pomiędzy nimi. Kontrolka ta w razie potrzeby wyświetli paski przewijania, chociaż nie jest to zbyt zgrabne. `TabControl` dobrze sprawdza się w przypadku danych dających podzielić się w naturalny sposób na grupy, które można umieścić na poszczególnych kartach. Nie jest jednak dobrym rozwiązaniem, gdy użytkownik musi porównywać dane znajdujące się na jednej karcie z zawartymi w innej.

Kontrolka `SplitContainer` pozwala użytkownikowi podzielić obszar ekranu na dwa przystające do siebie regiony. Zawiera dwie kontrolki `Panel`, w których można umieszczać swoje własne kontrolki. Kiedy użytkownik przeciąga linię dzielącą oba panele, kontrolka odpowiednio zmienia rozmiary obu. Można ustawić właściwość `AutoScroll` tych paneli na `True`, aby w razie potrzeby automatycznie wyświetlały paski przewijania.

Kontrolka `SplitContainer` okazuje się przydatna, gdy formularz jest za mały, by pomieścić wszystkie dane, które program musi wyświetlić. Dzięki niej użytkownik może powiększyć jedną część ekranu kosztem drugiej. Jest to szczególnie przydatne, gdy w celu porównania wartości z tych dwóch regionów trzeba je jednocześnie widzieć.

Mimo że kontenery `SplitContainer` można zagnieżdżać w innych kontenerach tego samego typu, najłatwiej ich używać, kiedy oddzielają tylko dwa regiony. Duże grupy kontrolki `SplitContainer`, które oddzielają wiele obszarów, są zazwyczaj zatłoczone i mało czytelne.

Przykładowy program o nazwie *UseSplitter*, który można pobrać z serwera FTP wydawnictwa Helion, dzieli swój formularz na dwa regiony za pomocą kontrolki `SplitContainer`. Ta ostatnia zawiera dwie kontrolki: `Panel` i `Splitter`, co w sumie daje formularz podzielony na dwa regiony. Dzięki temu, że za pomocą kontrolki `Splitter` nie trzeba ustawiać właściwości `Dock` ani kolejności na stosie, jak w kontrolce `Splitter`, kontrolka `SplitContainer` jest łatwiejsza w użyciu.

Opisane tu kontenery pomagają odpowiednio ustawiać zawarte w nich kontrolki. Właściwości `Anchor` i `Dock` wszystkich kontrolki znajdujących się wewnątrz tych kontenerów działają względem nich. Wyobraźmy sobie na przykład, że umieszczono serię przycisków z właściwością `Anchor = Top, Left, Right` wewnątrz kontrolki `SplitContainer`, dzięki czemu mają one taką samą szerokość, jaką posiada zawierający je panel. Kiedy użytkownik przeciągnie linię dzielącą, przyciski automatycznie zmieniają rozmiary, aby pasować do otaczającego je panelu.

Wybór opcji

Kontrolki opcji do wyboru pozwalają użytkownikowi wybrać odpowiednie wartości. Właściwie stosowane, pozwalają zminimalizować ryzyko wprowadzenia niepoprawnych danych. Dzięki temu potrzebna jest mniejsza ilość kodu obsługującego błędy.

Do kontrolki tych zaliczają się: `CheckBox`, `CheckedListBox`, `ComboBox`, `ListBox`, `RadioButton`, `DateTimePicker`, `MonthCalendar`, `DomainUpDown`, `NumericUpDown`, `TrackBar`, `HScrollBar` oraz `VScrollBar`.

Kontrolka `CheckBox` pozwala na zaznaczenie wybranej opcji bez względu na inne. Aby pozwolić na wybranie tylko jednej pozycji z kilku, należy użyć kontrolki `RadioButton`. Jeśli formularz wymaga więcej niż pięciu do siedmiu kontrolki `CheckBox`, warto rozważyć możliwość użycia w zamian kontrolki `CheckedListBox`.

Kontrolka `CheckedListBox` pozwala użytkownikowi wybrać spośród kilku niezależnych od siebie opcji. Zazwyczaj ma postać listy kontrolki `CheckBox` z paskami przewijania, które pokazują się w razie potrzeby.

Kontrolka `ComboBox` pozwala użytkownikowi na zdecydowanie się na jedną opcję. Jest ona szczególnie przydatna, gdy jej własność `DropDownStyle` jest ustawiona na `DropDownList`, ponieważ dzięki niej można wybrać daną pozycję z listy rozwijanej w dół. Aby umożliwić użytkownikowi zdecydowanie się na jakąś wartość lub wprowadzenie takiej, której nie ma na liście, należy ustawić własność `DropDownStyle` na `Simple` lub `DropDown`. Kontrolka ta spełnia podobną rolę do prostej kontrolki `ListBox`, ale zajmuje mniej miejsca.

Kontrolka `ListBox` wyświetla listę opcji do wyboru. Można ją tak skonfigurować, aby użytkownik mógł zdecydować się na jedną lub więcej pozycji. Kontrolka `ListBox` zajmuje więcej miejsca od `ComboBox`, ale może być łatwiejsza w użyciu, jeśli jej lista jest bardzo długa. Oczywiście gdy użytkownik ma do wyboru zbyt wiele opcji, łatwo może przez przypadek odznaczyć wszystkie zaznaczone wcześniej pozycje i zaznaczyć kolejny element. Aby ułatwić życie osobie korzystającej z komputera, warto zastanowić się nad użyciem w zamian kontrolki `CheckedListBox`, która nie jest obciążona tym problemem.

Kontrolka `RadioButton` pozwala wybrać jedną opcję ze zbioru. Na przykład trzy kontrolki tego typu mogą reprezentować opcje `Small`, `Medium` i `Large`. Jeśli użytkownik zaznaczy jedną z nich, pozostałe zostaną automatycznie odznaczone. Ten typ kontrolki jest przydatny, gdy lista opcji nie jest duża. Przynosi też korzyść pokazania użytkownikowi wszystkich opcji do wyboru naraz. Jeśli lista opcji jest długa, należy rozważyć wybór kontrolki `ListBox` lub `ComboBox`.

Kontrolki `DateTimePicker` i `MonthCalendar` pozwalają użytkownikowi wybierać daty i godziny. Zapewniają poprawność wprowadzonych danych, dzięki czemu są z reguły lepsze od innych kontrolki, które służą do wyboru daty i godziny. Jeśli na przykład pozwoli się użytkownikowi wprowadzić datę w kontrolce `TextBox` w postaci liczbowej, trzeba będzie dodatkowo napisać kod, który sprawdzi, czy nie wpisał on czegoś w rodzaju „29 luty 2008”.

Kontrolki `DomainUpDown` i `NumericUpDown` pozwalają przewijać listę opcji. Jeśli spis ten jest krótki, lepszym rozwiązaniem będzie wybranie kontrolki `ListBox` lub `ComboBox`. Jednak `DomainUpDown` i `NumericUpDown` zajmują bardzo mało miejsca, a więc mogą być przydatne na mocno zatflo-

czonych formularzach. Gdy użytkownik kliknie jedną ze strzałek kontrolki, będzie mógł bardzo szybko przewinąć wartości. Zatem kontrolki te mogą być przydatne w przypadku bardzo długich list opcji.

Kontrolka `TrackBar` pozwala na wybranie wartości liczbowej za pomocą przeciąganego wskaźnika. Zajmuje ona znacznie więcej miejsca od kontrolki `NumericUpDown`, ale jest też bardziej intuicyjna. Ponadto wymaga sporej zręczności przy dużym przedziale dozwolonych wartości.

`HScrollBar` i `VScrollBar` pozwalają użytkownikowi na wybór opcji liczbowej za pomocą przeciągania „kciuka”, podobnie jak kontrolka `TrackBar`. Wszystkie trzy mają nawet podobne własności. Główna różnica przejawia się w ich wyglądzie. Kontrolki `ScrollBar` są bardziej elastyczne, jeśli chodzi o rozmiary (`TrackBar` ma ustaloną wysokość w stosunku do szerokości), więc niektórym użytkownikom mogą wydawać się bardziej eleganckie. Jednak osoby korzystające z komputera są przyzwyczajone do zwykłego przewijania obszarów formularza, a więc użycie ich jako pasków do wybierania liczb może wywoływać zdziwienie.

Wprowadzanie danych

Czasami kontrolki opisane w poprzednim podrozdziale okazują się niepraktyczne. Na przykład nie da się wprowadzić rozsądnie długiej historii pracy lub komentarza za pomocą `ComboBox` lub `RadioButton`.

Kontrolki `RichTextBox`, `TextBox` i `MaskedTextBox` pozwalają na wpisywanie tekstu z niewieloma ograniczeniami. Są one najbardziej przydatne, gdy użytkownik musi wprowadzić duże ilości tekstu, który nie wymaga walidacji.

Kontrolka `TextBox` jest mniej skomplikowana i łatwiejsza w użyciu niż `RichTextBox`. Dlatego jeśli nie są konieczne dodatkowe funkcje tej drugiej, można używać pierwszej z wymienionych kontroltek. Jeżeli dodatkowe funkcje (takie jak możliwość wyboru kroju pisma, wcięcia, wyrównanie akapitów, indeks górny i dolny, różne kolory, cofanie i ponawianie więcej niż jednego kroku itd.) są niezbędne, należy użyć `RichTextBox`.

`MaskedTextBox` jest kontrolką `TextBox`, która wymaga, aby wprowadzane dane były w określonym formacie. Może na przykład wymuszać podawanie numeru telefonu następująco: 234-567-8901. Jest przydatna tylko w przypadku krótkich pól ze ściśle ustalonym formatem danych. Redukuje natomiast ryzyko wprowadzenia przez użytkownika niepoprawnych informacji.

Wyświetlanie danych

Kontrolki opisane w tym podrozdziale wyświetlają dane użytkownikowi. Należą do nich kontrolki `Label`, `DataGridView`, `ListView`, `TreeView` oraz `PropertyGrid`.

Pierwsza z nich wyświetla tekst, który użytkownik może tylko przeczytać — nie da się go ani zaznaczyć, ani zmodyfikować. Ponieważ nie można zaznaczyć tekstu, nie da się go również skopiować do schowka. Jeśli tekst ten zawiera wartość, którą użytkownik mógłby chcieć skopiować i wkleić gdzieś indziej (na przykład numer seryjny, numer telefonu, adres e-mail, adres URL itd.), można użyć kontrolki `TextBox` z własnością `ReadOnly` ustawioną na `True`.

Kontrolka `DataGridView` służy do wyświetlania danych tabelarycznych. Potrafi także wyświetlać po kilka tabel połączonych powiązaniem typu element główny-szczegół, które użytkownik może łatwo i szybko przeszukiwać. Istnieje również możliwość takiego skonfigurowania tej kontrolki, by można było aktualizować znajdujące się w niej dane.

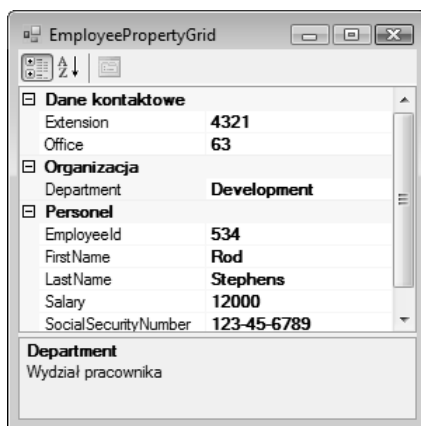
Kontrolka `ListView` wyświetla dane, które naturalnie wyglądają jak zbiór ikon lub lista wartości z kolumnami zawierającymi dodatkowe szczegóły. Można posortować te dane według kolumny z elementami lub szczegółami, ale wymaga to trochę pracy.

Kontrolka `TreeView` wyświetla dane hierarchiczne w formie drzewa — podobnego do tego, które zawiera katalogi w Explorersze Windows. Można pozwolić użytkownikowi na edytowanie etykiet jego węzłów.

Kontrolka `PropertyGrid` wyświetla informacje o obiekcie w formacie podobnym do używanego przez okno `Properties` w trybie projektowania. Pozwala ona użytkownikowi zorganizować własności alfabetycznie lub według kategorii. Umożliwia też edytowanie ich wartości. Na rysunku 8.3 przedstawiono przykładowy program o nazwie `EmployeePropertyGrid` (można go pobrać z serwera FTP wydawnictwa Helion), który wyświetla informacje o obiekcie `Employee` w kontrolce `PropertyGrid`.

Rysunek 8.3.

Kontrolka
`PropertyGrid`
wyświetla
własności obiektu



Informowanie użytkownika

W tym podrozdziale opiszę kontrolki dostarczające informacje użytkownikowi. Zaliczają się do nich `ToolTip`, `HelpProvider`, `ErrorProvider`, `NotifyIcon`, `StatusStrip` oraz `ProgressBar`. Ich głównym celem jest poinformowanie użytkownika, co się dzieje, bez zabrania mu kontynuowania wykonywania innych czynności. Na przykład kontrolka `ErrorProvider` oznacza pole jako niepoprawne, ale nie blokuje wprowadzania danych w innych miejscach.

`ToolTip` wyświetla krótką informację na temat przeznaczenia kontrolki, na którą użytkownik najechał kursorem. `HelpProvider` dostarcza bardziej szczegółowe informacje o przeznaczeniu danej kontrolki, kiedy użytkownik aktywuje ją i naciśnie klawisz `F1`. Wysokiej jakości aplikacje dostarczają zarówno chmurki (ang. *tooltip*), jak i pomoc w przycisku `F1` dla każdej kontrolki, której funkcja może nie być oczywista. Opcje te nie są zbyt natrętne

— i pojawiają się tylko wtedy, gdy użytkownik ich potrzebuje. Dlatego lepiej jest przesadzić ze zbyt dużą ilością pomocy, niż dostarczyć jej za mało.

`ErrorProvider` oznacza kontrolki zawierające niepoprawne dane. Lepiej jest używać kontrolek wyboru opcji, które nie pozwalają na wprowadzanie błędnych informacji, ta jednak jest przydatna, kiedy okazuje się to niemożliwe.

Kontrolka `NotifyIcon` wyświetla niewielką ikonę w obszarze powiadomień paska zadań, która informuje użytkownika o stanie aplikacji. Jest szczególnie przydatna w przypadku programów działających w tle, którym nie trzeba ciągle poświęcać uwagi. Jeśli aplikacja wymaga natychmiastowej reakcji ze strony użytkownika, powinna wyświetlać okno dialogowe lub ramkę z komunikatem, zamiast zdawać się na kontrolkę `NotifyIcon`.

Obszar powiadomień paska zadań, zwany również zasobnikiem systemu Windows, to niewielkie miejsce na pasku zadań. Po prawej stronie wyświetla godzinę i datę oraz ikony, które oznaczają status różnych działających aplikacji.

Kontrolka `StatusStrip` wyświetla obszar (zazwyczaj w dolnej części formularza), w którym program może podać użytkownikowi pewne informacje na temat swojego stanu. Może on mieć postać niewielkiej ikony lub krótkiego komunikatu tekstowego. Przekazuje znacznie więcej informacji niż kontrolka `NotifyIcon`, ale jest widoczny tylko wtedy, gdy widać formularz.

Kontrolka `ProgressBar` pokazuje, jaka część długiego zadania została ukończona. Zazwyczaj zadanie to jest wykonywane synchronicznie, przez co użytkownik nie ma nic więcej do roboty, jak tylko wpatrywać się w pasek postępu. Kontrolka `ProgressBar` pozwala zorientować się, że operacja nie utknęła w miejscu.

Inicjowanie akcji

Kontrolki każdego rodzaju reagują na zdarzenia, a więc każda z nich może inicjować akcję. Niemniej jednak użytkownicy oczekują wykonywania różnych czynności tylko od niektórych kontrolek. Na przykład naciśnięcie przycisku ma włączać jakąś akcję, ale kliknięcie etykiety lub pola wyboru nie powinno uruchamiać długich procesów.

Aby uniknąć zamieszania, należy do uruchamiania akcji korzystać tylko z tych kontrolek, które są w tym celu używane najczęściej. Należą do nich: `Button`, `MenuStrip`, `ContextMenuStrip`, `ToolStrip`, `LinkLabel`, `TrackBar`, `HScrollBar`, `VScrollBar` oraz `Timer`. Wymienione kontrolki — z wyjątkiem `Timer` — pozwalają użytkownikowi zainicjować akcję.

Wszystkie te kontrolki komunikują się z programem za pomocą procedur obsługi zdarzeń. Na przykład procedura obsługi zdarzenia `Click` kontrolki `Button` w normalnej sytuacji zmusza aplikację do wykonania jakichś czynności, kiedy użytkownik kliknie przycisk.

Inne kontrolki również dysponują zdarzeniami, które mogą inicjować akcje. Na przykład `CheckBox` udostępnia zdarzenia `CheckChanged` i `Click`, których można użyć do wykonania jakichś działań. Jeśli przechwyci się odpowiednie zdarzenia, będzie można użyć prawie każdej kontrolki do zainicjowania określonej akcji. Ponieważ głównym przeznaczeniem tych narzędzi nie jest wykonywanie kodu, nie zostały one wymienione w tym podrozdziale.

Kontrolka `Button` pozwala użytkownikowi zmusić program do wykonania wybranej funkcji. Przycisk jest zazwyczaj cały czas widoczny na formularzu, a więc najczęściej przydaje się, gdy trzeba często wykonywać jakąś akcję lub jest ona częścią tego, co stanowi główne zadanie aplikacji. Dla rzadziej wykonywanych akcji należy używać kontroltek `MenuStrip` i `ContextMenuStrip`.

Elementy w kontrolce `MenuStrip` również pozwalają użytkownikowi zmusić program do wykonania wybranych akcji. W jej przypadku konieczne jest wykonanie większej liczby czynności, niż gdy używa się przycisku, ponieważ trzeba otworzyć menu, zaleźć interesującą opcję i kliknąć ją. Należy jednak pamiętać o tym, że menu zajmują mniej miejsca na formularzu niż przyciski. Do często używanych opcji znajdujących się w menu można także przypisać skróty klawiszowe (typu `F5` lub `Ctrl+S`) — ich uruchamianie jest nawet łatwiejsze niż używanie przycisków.

`ContextMenuStrip` posiada takie same wady i zalety, jakie ma kontrolka `MenuStrip`. Jest ona jednak dostępna tylko z danych kontroltek na formularzu, a więc przydaje się dla poleceń mających zastosowanie jedynie w określonych warunkach. Na przykład *Zapisz* ma zastosowanie do wszystkich danych załadowanych przez program, a więc powinno znajdować się w kontrolce `MenuStrip`. Z polecenia usuwającego określony obiekt z rysunku da się skorzystać tylko w stosunku do tego obiektu. Dzięki umieszczeniu tej opcji w kontrolce `ContextMenuStrip`, która jest związana z obiektem, pozostaje ona ukryta, gdy użytkownik pracuje nad czymś innym. Dodatkowo sprawia, że związek pomiędzy akcją (usunięcie) a obiektem jest oczywisty — zarówno dla programu, jak i użytkownika.

Kontrolka `ToolStrip` łączy niektóre najlepsze cechy menu i przycisków. Wyświetla szereg przycisków, dzięki czemu można ich użyć bez wchodzenia do menu. Są one małe, a znajdują się w górnej części formularza, więc nie zajmują tyle miejsca co duże przyciski.

Zwykle przyciski lub przyciski kontrolki `ToolStrip`, które znajdują się na formularzu, reprezentują często używane polecenia z menu. Te ostatnie można uruchamiać za pomocą skrótów klawiszowych lub przycisków, jeśli nie pamięta się skrótów.

`LinkLabel` wyświetla tekst — podobnie jak kontrolka `Label`. Dodatkowo pokazuje niebieski tekst z podkreśleniem, specjalny kursor, kiedy użytkownik najeżdża na niego, a także uruchamia zdarzenie, gdy kliknie się ten tekst. Dzięki temu z kontrolki tej warto korzystać, gdy kliknięcie tekstu ma sprawiać, że wykonana zostanie jakaś akcja. Na przykład na stronie internetowej kliknięcie odnośnika spowoduje przejście do wskazywanej przez niego strony.

`TrackBar`, `HScrollBar` i `VScrollBar` pozwalają użytkownikowi wybrać wartość liczbową za pomocą przeciągania rączki. Jak napisałem wcześniej w podrozdziale „Wybór opcji”, dzięki tym kontrolkom można wybierać wartości liczbowe. Istnieje też możliwość używania ich do interaktywnego wykonywania pewnych akcji. Na przykład z pasków przewijania często korzysta się do przewijania części formularza. Bardziej ogólnie: służą do zmuszania programu do podejmowania działań na podstawie jakiejś nowej wartości. Na przykład za pomocą paska przewijania można pozwolić użytkownikowi wybrać komponenty czerwieni, zieleni i niebieskiego dla obrazu. W chwili zmiany przez niego wartości paska przewijania program aktualizuje kolory grafiki.

Kontrolka `Timer` wyzwala określoną akcję w równych odstępach czasu. Kiedy uruchamia ona swoje zdarzenie, program wykonuje akcję.

Wyświetlanie grafiki

Kontrolki opisane w tym podrozdziale służą do wyświetlania obrazów graficznych na ekranie lub prezentacji ich na wydruku. Są to `Form`, `PictureBox`, `PrintPreviewControl`, `PrintDocument` oraz `PrintPreviewDialog`.

Kontrolka `Form` (która także może wyświetlać obrazy graficzne) umożliwia rysowanie, ale z reguły lepiej jest rysować w `PictureBox` niż w samym formularzu. Dzięki temu łatwiej jest przenieść grafikę przy zmianie projektu formularza. Jeśli na przykład dojdiesz do wniosku, że obraz może być za duży, możesz z łatwością przenieść kontrolkę `PictureBox` do przewijanej kontrolki `Panel`. Znacznie trudniej byłoby przepisać kod, który miałby przesuwać rysunek z kontrolki `Form` do `PictureBox`.

Kontrolka `PrintPreviewControl` wyświetla podgląd wydruku obiektu `PrintDocument`. Program reaguje na zdarzenia wyzwalane przez obiekt `PrintDocument`. `PrintPreviewDocument` wyświetla wyniki w kontrolce znajdującej się na jednym z formularzy programu.

`PrintPreviewDialog` wyświetla obrazy z obiektu `PrintDocument` — podobnie jak kontrolka `PrintPreviewControl` — ale posiada własne okno dialogowe. Jeśli nie ma konieczności zaprojektowania w specjalny sposób podglądu wydruku, łatwiej jest użyć `PrintPreviewDialog`, niż budować własne okno dialogowe za pomocą kontrolki `PrintPreviewControl`. `PrintPreviewDialog` udostępnia wiele funkcji, które pozwalają użytkownikowi na powiększenie, przewijanie i zmienianie stron podglądanego dokumentu. Samodzielne zaimplementowanie ich wszystkich wymagałoby bardzo dużo pracy.

Wyświetlanie okien dialogowych

Visual Basic oferuje szeroki wybór okien dialogowych, które pozwalają użytkownikowi na dokonywanie standardowych wyborów. Użycie właściwego z nich jest zazwyczaj łatwe, ponieważ każde ma bardzo konkretne przeznaczenie. Poniższa tabela zawiera listę tych okien dialogowych i krótki opis ich funkcji.

Okno dialogowe	Przeznaczenie
<code>ColorDialog</code>	Wybór koloru.
<code>FolderBrowserDialog</code>	Wybór folderu (katalogu).
<code>FontDialog</code>	Wybór czcionki.
<code>OpenFileDialog</code>	Wybór pliku do otwarcia.
<code>PageSetupDialog</code>	Ustawienia strony do druku.
<code>PrintDialog</code>	Wydruk dokumentu.
<code>PrintPreviewDialog</code>	Wyświetlenie podglądu wydruku.
<code>SaveFileDialog</code>	Wybór pliku do zapisania.

Okna te demonstruje program *UseDialogs*, który można pobrać z serwera FTP wydawnictwa Helion.

Wspieranie innych kontrolki

Wiele kontrolki Visual Basica wymaga wsparcia ze strony innych tego typu narzędzi. Dwie kontrolki najczęściej używane przez inne to `ImageList` i `PrintDocument`. Zaliczają się do nich również `DataConnector` i `DataNavigator`.

`ImageList` przechowuje obrazy, które wyświetlają inne kontrolki. Kod programu może także pobierać z niej obrazy i używać ich w dowolny sposób.

`PrintDocument` wspomaga drukowanie i podglądanie wydruku. Generuje grafikę wysyłaną do drukarki albo kontrolki `PrintPreviewDialog` lub `PrintPreviewControl`.

`DataConnector` tworzy łącze pomiędzy źródłem danych a kontrolkami z nią związanymi. Za pomocą jej metod program może nawigować po danych, sortować je, filtrować oraz aktualizować. Ponadto kontrolka ta odpowiednio aktualizuje związane z nią kontrolki.

Kontrolka `DataNavigator` udostępnia metody do nawigowania po źródłach danych, takich jak `DataConnector`.

Kontrolki niestandardowe

W Visual Basicu jest dostępnych mnóstwo różnych kontrolki, które są gotowe do użycia. Nie są to jednak wszystkie te, z których można korzystać. Poprzez kliknięcie prawym przyciskiem myszy w oknie *Toolbox* i wybranie opcji *Choose Items* można uzyskać dostęp do ogromnej listy dostępnych w systemie komponentów .NET i COM.

Można także używać kontrolki utworzonych przez inne firmy, które oferują je w sprzedaży lub udostępniają bezpłatnie w sieci. Wiele z nich wykonuje specjalistyczne zadania, na przykład generuje kody kreskowe, tworzy kształty formularzy, zniekształca obrazy czy tworzy specjalne efekty graficzne.

Są też kontrolki rozszerzające funkcjonalność standardowych tego typu narzędzi. Istnieje kilka kontrolki rysujących dwu- i trójwymiarowe wykresy i grafy. Inne oferują bardziej rozbudowane usługi raportowania od narzędzi dostępnych w Visual Studio.

Gdy wpisze się frazę `windows forms controls` w dowolnej większej wyszukiwarce internetowej, będzie można znaleźć mnóstwo witryn internetowych, które oferują kontrolki do pobrania bezpłatnie lub za pewną kwotę. Oto kilka adresów, które warto odwiedzić:

- `MVPs.org` (www.mvps.org) — witryna prowadząca do zasobów udostępnianych przez ludzi związanych z programem Microsoftu Most Valuable Professional (MVP). Projekt `Common Controls Replacement Project` (ccrp.mvps.org) oferuje kontrolki duplikujące i rozszerzające standardowe kontrolki programu Visual Basic 6. Strona ta nie jest już rozwijana, ale niektóre ze starych kontrolki szóstej wersji Visual Basica mogą stać się inspiracją do budowy własnych nowych narzędzi. Witryna `MVPs.org` jest ponadto dobrym ogólnym źródłem danych.

- Windows Forms .NET (*windowsclient.net*) — oficjalna strona Microsoftu, założona dla społeczności zainteresowanej Windows Forms .NET.
- ASP.NET (*www.asp.net*) — oficjalna społeczność ASP.NET Microsoftu.
- Download.com (*www.download.com*).
- Shareware.com (*www.shareware.com*).
- Shareware Connection (*www.sharewareconnection.com*).

Wymienione wyżej witryny są dobre do rozpoczęcia poszukiwań, ale nie zamykają listy. Kontrolki do pobrania można znaleźć w setkach (o ile nie tysiącach) stron internetowych.

Gdy pobiera się niestandardowe kontrolki i produkty, zawsze należy zachowywać pewną powściągliwość. Dodanie każdego tego typu narzędzia do projektu powoduje uzależnienie tego ostatniego od niego. Gdy przenosi się efekt pracy do nowszej wersji Visual Basica, należy się upewnić, że tam również dana kontrolka będzie działała. Jeśli powstanie nowa wersja tego narzędzia, trzeba będzie sprawdzić, czy będzie działała w używanej edycji programu. Jeśli nie, można utknąć ze starą — już niewspieraną wersją kontrolki.

Problem staje się jeszcze poważniejszy, gdy kontrolki i narzędzia współpracują ze sobą. Jeśli zmieni się cokolwiek, konieczne będzie znalezienie zestawu wersji wszystkich narzędzi, które będą ze sobą zgodne.

*Osobiście staram się używać niestandardowych kontrolki jak najrzadziej, ponieważ przy pisaniu tej książki nie mogę wychodzić z założenia, że czytelnik posiada określone narzędzia innych firm. Korzystam z WinZipa (*www.WinZip.com*) i Internet Download Managera (*www.InternetDownloadManager.com*) poza projektami, ale nie wewnątrz nich.*

Kontrolki niestandardowych należy używać wtedy, gdy pozwalają zaoszczędzić dużo czasu. Jednak przed ich zastosowaniem warto przemyśleć, ile czasu zajęłoby poradzenie sobie bez nich, a ile zastąpienie ich innymi w razie potrzeby migracji do nowszej wersji Visual Basica.

Podsumowanie

Kontrolki stanowią główne połączenie pomiędzy użytkownikiem a aplikacją. Umożliwiają programowi wysyłanie danych do użytkownika, a użytkownikowi kontrolowanie programu. Kontrolki są wszędzie, w praktycznie każdej aplikacji działającej w systemie Windows. Tylko niewielka liczba programów uruchomionych w tle może się bez nich obyć.

W tym rozdziale krótko opisałem przeznaczenie standardowych kontrolki Visual Basica. Zamieściłem tu też wskazówki dotyczące odpowiednich z nich w przypadku wykonywania różnych zadań. Bardziej szczegółowy opis tych kontrolki znajduje się w Dodatku G.

Nawet gdy wie się wszystko o kontrolkach, nie ma się gwarancji, że stworzony interfejs użytkownika będzie odpowiedni. Projektowanie intuicyjnych i łatwych w użyciu interfejsów jest sztuką. Dobry projekt umożliwia użytkownikowi wykonywanie pracy w naturalny sposób, przy czym zminimalizowana jest ilość marnowanego czasu. Źle zaprojektowany interfejs może przeszkadzać w wykonywaniu czynności, a nawet zamienić najprostsze z nich w walkę z programem.

Informacje na temat budowy użytecznych aplikacji można znaleźć w książkach poświęconych projektowaniu interfejsów. Objasniono w nich typowe problemy i rozwiązania związane z tym ostatnimi. Wiele też można nauczyć się poprzez badanie istniejących już aplikacji, które są uznane za dobre. Przyjrzyj się układowi ich formularzy i okien dialogowych. Nie należy podkradać zastosowanych tam rozwiązań, ale spróbować zrozumieć, dlaczego kontrolki zostały w nich poukładane właśnie w taki sposób. Przyjrzyj się aplikacjom, które szczególnie lubisz, a do tego są wyjątkowo łatwe w użyciu. Porównaj je z programami, które uważasz za niezgrabne i mało przejrzyste.

Ten rozdział stanowi wprowadzenie do kontroltek Windows Forms. W rozdziale 9. — „Używanie kontroltek Windows Forms” — znajdziesz bardziej szczegółowe objaśnienie sposobów ich wykorzystywania. Zamieszczone tam zostaną opisy metod dodawania kontroltek do formularza w trakcie projektowania i działania programu, a także wskazówki, jak używać ich własności, metod i zdarzeń.