helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

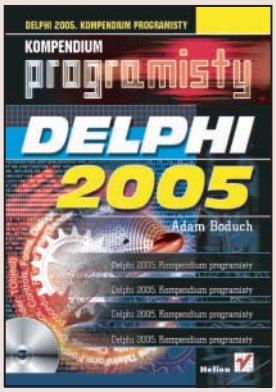
TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Delphi 2005. Kompendium programisty

Autor: Adam Boduch
ISBN: 83-7361-780-9
Format: B5, stron: 1048



Nowa wersja środowiska programistycznego Delphi to kolejne nowości – pełna integracja zarówno z platformą .NET, jak i Win32, nowe technologie tworzenia aplikacji, nowe komponenty i możliwość stosowania nie tylko języka Delphi, ale również C#. Za pomocą najnowszej wersji Delphi można tworzyć nie tylko „standardowe" aplikacje, ale również wykorzystywać możliwości oferowane przez .NET. Tworzenie aplikacji klient-serwer, usług sieciowych, intranetów i dynamicznych witryn WWW w oparciu o tę platformę stało się szybsze i łatwiejsze. Większe możliwości pociągnęły za sobą również sporo zmian w samym środowisku i sposobie korzystania z niego.

Książka „Delphi 2005. Kompendium programisty" przedstawia najnowszą wersję tego popularnego narzędzia. Jej pierwsza część opisuje zagadnienia podstawowe – interfejs użytkownika i zasady programowania, najważniejsze elementy platformy .NET oraz podstawy wizualnego tworzenia aplikacji. Kolejne części zawierają bardziej zaawansowane tematy – wykrywanie błędów w programach, tworzenie aplikacji dla platformy .NET, komunikacja z bazami danych oraz programowanie sieciowe. Czytając książkę, dowiesz się, jak tworzyć aplikacje i usługi sieciowe, jak korzystać z technologii XML i IntraWeb oraz jak tworzyć wydajne aplikacje klient-serwer.

- Interfejs użytkownika Delphi 2005
- Język programowania Delphi
- Podstawowe elementy .NET
- Wizualne tworzenie aplikacji i stosowanie komponentów
- Środowisko .NET Framework
- Zasady programowania obiektowego
- Wykrywanie błędów w aplikacjach
- Aplikacje bazodanowe
- Technologia ADO.NET i BDE.NET
- Wykorzystanie języka XML
- Tworzenie dynamicznych witryn WWW w języku ASP.NET
- Usługi sieciowe
- Podstawy języka C#

Jeśli planujesz wykorzystanie Delphi 2005 do budowania aplikacji dla platformy .NET, ta książka będzie dla Ciebie niezastąpionym źródłem informacji

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Tematyka książki 25

Część I 27

Rozdział 1. Wstęp do programowania 29

Historia informatyki 29
Czym jest programowanie? 30
Języki programowania 31
Asembler 33
Fortran 33
C 34
C++ 34
Perl 34
PHP 35
Turbo Pascal 35
Java 35
C# 36
Kilka słów o kompilatorach 36
Działanie kompilatorów 37
Który kompilator wybrać? 38
Dzień programisty 39
Delphi 39
Object Pascal 40
Delphi — czy warto? 40
Co będzie potrzebne w trakcie nauki? 41
Instalacja Delphi 41
Wersje Delphi 42
Cennik 42
Wymagania Delphi 43
Instalacja 43
Jak się uczyć? 44

Podstawowe okna Delphi ... 45
Welcome Page ... 46
Okno główne ... 47
Projektant formularzy ... 48
Paleta narzędzi ... 48
Inspektor obiektów ... 49
Menedżer projektu ... 51
Edytor kodu ... 52
Ukrywanie okna ... 54
Pierwszy projekt ... 54
Tworzenie projektu ... 54
Zapisywanie projektu ... 55
Uruchamianie programu ... 55
Zamykanie programu ... 56
Otwieranie projektu ... 56
Kompilacja projektu ... 57
Pliki projektu ... 57
Przydatne odnośniki ... 58
Test ... 59
FAQ ... 60
Podsumowanie ... 62

Rozdział 2. Wprowadzenie do .NET ... 63
Interfejs programistyczny ... 64
API systemu Windows ... 64
Wizja .NET ... 65
Składniki platformy .NET ... 66
Konkluzja ... 69
Delphi a .NET ... 69
Rodzaje aplikacji ... 70
Aplikacje konsolowe ... 70
Windows Forms ... 70
VCL ... 70
VCL.NET ... 71
Windows Forms vs VCL.NET ... 72
Formularze Web Forms ... 73
Składniki .NET Framework ... 74
Usługi sieciowe ... 74
Niezależność ... 75
Uniwersalność ... 75
Test ... 76
FAQ ... 77
Podsumowanie ... 78

Rozdział 3. Język Delphi ... 79
Aplikacje konsolowe ... 80
Zapisywanie projektu ... 80
Po kompilacji… ... 81
Najprostszy program ... 81
Podstawowa składnia ... 82
Czytanie kodu ... 82
Wielkość liter ... 82

Pamiętaj o średniku! ... 83
Bloki begin i end ... 83
Dyrektywa program ... 84
Komentarze ... 84
Umiejętne komentowanie ... 86
Wypisywanie tekstu ... 86
Położenie instrukcji ... 88
Instrukcja Writeln ... 88
Zmienne ... 89
Deklaracja zmiennych ... 89
Typy danych ... 90
Deklaracja kilku zmiennych ... 91
Przydział danych ... 92
Deklaracja zakresu danych ... 94
Restrykcje w nazewnictwie ... 94
Stałe ... 95
Domyślne typy stałych ... 95
Używanie zmiennych w programie ... 97
Łączenie danych ... 97
Tablice danych ... 99
Tablice jako stałe ... 100
Tablice wielowymiarowe ... 101
Tablice dynamiczne ... 102
Polecenia Low i High ... 103
Operatory ... 104
Operatory przypisania ... 104
Operatory porównania ... 105
Operatory logiczne ... 105
Operatory arytmetyczne ... 106
Operatory bitowe ... 108
Pozostałe operatory ... 109
Instrukcje warunkowe ... 109
Instrukcja if..then ... 109
Instrukcja case..of ... 112
Instrukcja else ... 115
Programowanie proceduralne ... 118
Procedury i funkcje ... 118
Procedury ... 118
Funkcje ... 121
Zmienne lokalne ... 122
Parametry procedur i funkcji ... 123
Parametry domyślne ... 125
Przeciążanie funkcji i procedur ... 126
Przekazywanie parametrów do procedur i funkcji ... 127
Procedury zagnieżdżone ... 129
Wplatanie funkcji i procedur ... 129
Własne typy danych ... 130
Tablice jako nowy typ ... 130
Aliasy typów ... 131
Rekordy ... 132
Przekazywanie rekordów jako parametrów procedury ... 132
Deklarowanie rekordu jako zmiennej ... 133
Instrukcja packed ... 134

Deklarowanie tablic rekordowych ... 134
Deklarowanie dynamicznych tablic rekordowych ... 135
Instrukcja wiążąca with ... 135
Programowanie strukturalne ... 136
Moduły ... 136
Tworzenie nowego modułu ... 137
Budowa modułu ... 137
Włączanie modułu ... 139
Funkcje wbudowane ... 139
Sekcje Initialization oraz Finalization ... 139
Dyrektywa forward ... 140
Konwersja typów ... 142
Rzutowanie ... 144
Parametry nieokreślone ... 144
Pętle ... 145
Pętla for..do ... 145
Pętla while..do ... 148
Pętla repeat..until ... 149
Pętla for-in ... 149
Polecenie Continue ... 151
Polecenie Break ... 152
Zbiory ... 153
Przypisywanie elementów zbioru ... 154
Odczytywanie elementów ze zbioru ... 154
Dodawanie i odejmowanie elementów zbioru ... 156
Typy wariantowe ... 156
VarType, VarTypeAsText ... 157
VarToStr ... 157
VarIs* ... 158
Pliki dołączane ... 158
Etykiety ... 159
Dyrektywy ostrzegawcze ... 160
Wstęp do algorytmiki ... 161
Schematy blokowe ... 161
Przykład — obliczanie silni ... 163
Efektywny kod ... 166
Instrukcje warunkowe ... 166
Typ Boolean w tablicach ... 167
Zbiory ... 169
Łączenie znaków w ciągach ... 169
Test ... 170
FAQ ... 171
Podsumowanie ... 173

Rozdział 4. IDE Delphi ... 175

Podstawowe paski narzędzi ... 175
Pasek Standard ... 176
Pasek View ... 176
Pasek Debug ... 177
Pasek Desktop ... 177
Pasek Custom ... 178
Pozostałe paski narzędzi ... 178

Repozytorium obiektów ... 178
Dodawanie projektu do repozytorium ... 179
Ustawienia repozytorium ... 180
Praca z paletą narzędzi ... 181
Umieszczanie komponentów na formularzu ... 182
Umieszczanie kilku komponentów naraz ... 182
Przycisk Categories ... 183
Szukanie obiektu ... 183
Przemieszczanie ikon ... 183
Menu palety narzędzi ... 183
Praca z komponentami ... 185
.NET ... 187
Komponent ... 187
Sterowanie komponentem ... 188
Praca z inspektorem obiektów ... 188
Edycja właściwości ... 189
Zdarzenia ... 191
Menu inspektora obiektów ... 191
Opcje inspektora obiektów ... 191
Projektant formularzy ... 192
Siatka pomocnicza ... 192
Usuwanie, kopiowanie i wklejanie ... 194
Zaznaczanie wielu komponentów ... 194
Menu projektanta formularzy ... 195
Pasek narzędziowy Align ... 196
Pasek narzędziowy Spacing ... 197
Pasek narzędziowy Position ... 198
Opcje projektanta formularzy ... 198
Projekty ... 199
Opcje projektu ... 200
Edytor WYSIWYG ... 205
Projektant strony WWW ... 206
Inspektor obiektów ... 207
Dodatkowe paski narzędzi ... 207
Edycja pozostałych plików ... 208
Opcje HTML w Delphi ... 209
Projektowanie interfejsu ... 210
Paski narzędziowe ... 210
Ikony dla paska narzędzi ... 211
Menu główne ... 213
Menu podręczne ... 214
Pozostałe elementy interfejsu ... 215
Kilka wersji językowych programu ... 218
Tworzenie angielskiej wersji językowej ... 219
Tłumaczenie projektu ... 220
Kompilacja projektu ... 221
Opcje Translation Manager ... 222
Opcje środowiska ... 223
Test ... 224
FAQ ... 224
Podsumowanie ... 226

Rozdział 5. Przegląd .NET Framework ... 227
Środowisko CLR ... 227
Kod pośredni IL ... 228
Kod zarządzany i niezarządzany ... 229
Moduł zarządzany ... 229
Podzespoły ... 230
Działanie CLR ... 230
System CTS ... 231
Specyfikacja CLS ... 232
Biblioteka klas ... 233
Moduły i przestrzenie nazw ... 233
Wieloznaczność ... 234
Główne przestrzenie nazw ... 235
Tworzenie przestrzeni nazw ... 237
Test ... 237
FAQ ... 238
Podsumowanie ... 238

Rozdział 6. Praca z kodem źródłowym ... 239
Na czym polega programowanie obiektowe? ... 239
Biblioteki wizualne ... 240
Podstawowy kod formularza ... 241
Formularz w VCL.NET ... 243
Sekcja uses ... 243
Klasa ... 244
Zmienna wskazująca na klasę ... 244
Plik *.nfm ... 244
Generowanie kodu ... 245
Nazwy komponentów ... 245
Programowanie zdarzeniowe ... 246
Generowanie zdarzeń ... 246
Lista zdarzeń w inspektorze obiektów ... 249
Jedno zdarzenie dla kilku obiektów ... 250
Edytor kodu ... 251
Ukrywanie kodu ... 252
Makra ... 252
Code Insight ... 252
Code Completion ... 253
Help Insight ... 254
Wyróżnianie błędów ... 254
Menu podręczne edytora kodu ... 255
Opcje edytora kodu ... 256
Skrawki kodu ... 262
To-Do List ... 263
Generowanie komentarza TODO ... 263
Menu podręczne okna ... 264
Szablony kodu ... 265
Refaktoryzacja ... 266
Deklaracja zmiennych ... 266
Deklaracja pól ... 267
Wyodrębnianie metod ... 267
Wyodrębnianie łańcucha zasobów ... 267

Zmiana nazwy ... 268
Szukanie modułu ... 269
Synchroniczna edycja ... 269
Wyszukiwanie odwołań ... 270
Menedżer projektu ... 271
Model View ... 272
Data Explorer ... 272
Menedżer historii ... 272
Test ... 274
FAQ ... 275
Podsumowanie ... 275

Rozdział 7. Programowanie obiektowe ... 277

Klasy ... 277
Składnia klasy ... 278
Do czego służą klasy? ... 278
Hermetyzacja ... 279
Dziedziczenie ... 280
Polimorfizm ... 280
Instancja klasy ... 280
Konstruktor ... 281
Destruktor ... 281
Pola ... 281
Metody ... 282
Tworzenie konstruktorów i destruktorów ... 283
Destruktory w .NET ... 286
Poziomy dostępu do klasy ... 290
Dziedziczenie ... 292
Przeciążanie metod ... 293
Typy metod ... 293
Przedefiniowanie metod ... 295
Typy zagnieżdżone ... 299
Parametr Self ... 300
Brak konstruktora ... 301
Brak instancji klasy ... 302
Class helpers ... 304
Klasy zaplombowane ... 305
Słowo kluczowe static ... 305
Właściwości ... 306
Parametr Sender procedury zdarzeniowej ... 309
Przechwytywanie informacji o naciśniętym klawiszu ... 310
Obsługa parametru Sender ... 312
Operatory is i as ... 313
Metody w rekordach ... 314
Interfejsy ... 315
Przeładowanie operatorów ... 316
Jakie operatory można przeładować? ... 317
Deklaracja operatorów ... 317
Binary i Unary ... 319
Wyjątki ... 319
Słowo kluczowe try..except ... 320
Słowo kluczowe try..finally ... 321

Słowo kluczowe raise ... 322
Klasa Exception ... 322
Selektywna obsługa wyjątków ... 323
Zdarzenie OnException ... 324
Identyfikatory ... 326
Boksowanie typów ... 327
Przykład wykorzystania klas ... 328
Zasady gry ... 328
Specyfikacja klasy ... 329
Zarys klasy ... 330
Sprawdzenie wygranej ... 333
Interfejs aplikacji ... 337
Tworzenie interfejsu graficznego ... 340
Gra Kółko i krzyżyk ... 341
Biblioteka VCL/VCL.NET ... 352
Klasa TApplication ... 353
Właściwości ... 356
Zdarzenia ... 361
Programowanie w .NET ... 365
Wspólny model programowania ... 366
Klasa System.Object ... 366
Test ... 368
FAQ ... 369
Podsumowanie ... 370

Rozdział 8. Podzespoły .NET ... 371

Czym jest COM? ... 371
Kontrolka w rozumieniu COM ... 372
Odrobinę historii ... 372
ActiveX ... 372
DCOM ... 373
Podstawowe podzespoły ... 373
Słowo o funkcji ShellExecute ... 375
Deasembler .NET ... 376
Obiektowość .NET ... 378
Transformacja modułu ... 380
Komponenty .NET ... 381
Przygotowanie komponentu w Delphi ... 382
Przygotowanie komponentu C# ... 383
Włączenie podzespołu w Delphi ... 385
Atrybuty podzespołu ... 385
Korzystanie z programu Reflection ... 388
Mechanizm reflection ... 389
Metadane ... 389
Funkcja GetType ... 389
Klasa System.Type ... 390
Ładowanie podzespołu ... 391
Przykład — własny program Reflection ... 392
Własne atrybuty ... 398
Aplikacje .NET Framework SDK ... 401
Global Assembly Cache Tool ... 402
WinCV ... 404

Narzędzie konfiguracji .NET Framework ... 404
PEVerify — narzędzie weryfikacji ... 405
Test ... 405
FAQ ... 406
Podsumowanie ... 406

Rozdział 9. Wykrywanie błędów w aplikacjach ... 407

Rodzaje błędów ... 408
Opcje kompilatora ... 409
Częste błędy programisty ... 411
Niezainicjalizowane zmienne obiektowe ... 411
Zwalnianie obiektów ... 412
Tablice ... 412
Wskaźniki ... 413
Rejestry ... 414
Stos ... 414
Sterta ... 415
Do czego służą wskaźniki? ... 415
Tworzenie wskaźnika ... 415
Przydział danych do wskaźników ... 417
Tworzenie wskaźników na struktury ... 418
Przydział i zwalnianie pamięci ... 419
Wartość pusta ... 420
Debuger Delphi ... 420
Interfejs Debug ... 420
Opcje projektu ... 421
Punkty przerwań ... 423
Polecenie Run to Cursor ... 428
Podgląd zmiennych ... 429
Inspektor śledzenia ... 431
Evaluate/Modify ... 432
Okno Call Stack ... 434
Okno Local Variables ... 434
Okno Thread Status ... 435
Okno Event Log ... 436
Okno modułów ... 437
Okno deasemblacji ... 438
Polecenie Go to Address ... 438
Okno Message View ... 439
Praca krokowa ... 440
Ikony na gutterze ... 440
Przekraczanie i wkraczanie ... 441
Opcje debugera ... 442
Strona Borland Debuggers ... 443
Zakładka Language Exceptions ... 444
Zakładka Native OS Exceptions ... 444
Zakładka Event Log ... 444
Menu związane z debugerem ... 445
Test ... 446
FAQ ... 447
Podsumowanie ... 448

Rozdział 10. Praca z plikami 449
Definicja pliku 450
Pliki tekstowe 450
Inicjalizacja 450
Tworzenie nowego pliku 451
Otwieranie istniejącego pliku 451
Odczyt plików tekstowych 452
Zapis nowych danych w pliku 453
Zapis danych na końcu pliku 454
Pliki amorficzne 455
Otwieranie i zamykanie plików 455
Tryb otwarcia pliku 456
Zapis i odczyt danych 456
Przykład działania — kopiowanie plików 457
Inne funkcje operujące na plikach 460
Funkcje operujące na katalogach 462
Pliki typowane 463
Deklaracja 463
Tworzenie pliku i dodawanie danych 464
Odczyt rekordu z pliku 465
Przykład działania — książka adresowa 465
Kopiowanie i przenoszenie plików 472
Kopiowanie 472
Przenoszenie pliku 472
Struktura TSHFileOpStruct 473
Strumienie 475
Podział strumieni 475
Prosty przykład na początek 476
Konstruktor klasy TFileStream 477
Pozostałe metody i właściwości klasy TStream 478
Właściwości 478
Metody 479
Praktyczny przykład 480
Wyszukiwanie 486
Rekord TSearchRec 487
Jak zrealizować wyszukiwanie? 487
Rekurencja 488
Praktyczny przykład 489
Informacja o dyskach 493
Pobieranie listy dysków 493
Pobieranie informacji o rozmiarze dysku 494
Pobieranie dodatkowych informacji 494
Obsługa plików w .NET 497
Klasy przestrzeni nazw System.IO 497
Praca z plikami 498
Praca z katalogami 500
Strumienie 502
Praca z plikami 502
Test 504
FAQ 505
Podsumowanie 507

Rozdział 11. Migracja do .NET ... 509
Czy warto przechodzić do .NET? ... 510
Ewolucja platform programistycznych ... 510
WinFX ... 511
Brakujące komponenty ... 512
Zmiany we właściwościach ... 512
Elementy języka ... 512
Wszystko jest klasą! ... 512
Przestrzenie nazw ... 513
Kompilacja warunkowa ... 513
Brakujące elementy ... 515
Ciągi znakowe w Delphi ... 523
Komunikaty ... 531
Destruktory ... 531
WinForms ... 532
Brak pliku *.dfm/*.nfm ... 533
VCL i WinForms ... 537
Platform Invoke ... 538
Wywołanie standardowe ... 539
Użycie atrybutu DLLImport ... 540
Parametry wyjściowe ... 541
Dane wskaźnikowe ... 543
Pobieranie danych z bufora ... 544
Kod zarządzany i niezarządzany ... 545
Używanie funkcji Win32 ... 545
Marshaling ... 546
Wady PInvoke ... 558
.NET a obiekty COM ... 558
Terminologia COM ... 559
Mechanizm COM Callable Wrappers ... 560
Przykładowy podzespół ... 561
Utworzenie biblioteki typu ... 564
Użycie biblioteki typu ... 564
Korzystanie z klasy COM ... 565
Kontrolki COM w aplikacjach .NET ... 569
Aplikacje sieciowe ... 570
Test ... 570
FAQ ... 571
Podsumowanie ... 572

Rozdział 12. Co nowego w Delphi 2005? ... 573
Trzy osobowości ... 573
IDE Delphi ... 574
Strona powitalna ... 574
Historia i kopie zapasowe ... 574
Okno struktury ... 575
Synchroniczna edycja ... 575
Help Insight ... 576
Wyróżnianie błędów ... 577
Paleta narzędzi ... 577
Wyszukiwanie ... 577
Pozostałe ... 578

Debuger ... 578
HTML ... 579
Bazy danych ... 579
dbGo for ADO ... 579
dbExpress ... 579
BDP.NET ... 579
Pozostałe ... 580
Refaktoryzacja ... 580
Zmiany w języku Delphi ... 580
Wplatanie funkcji i procedur ... 580
Pętla for-in ... 581
Kodowanie Unicode ... 583
Podsumowanie ... 583

Część II ... 585

Rozdział 13. Architektura bazodanowa Delphi ... 587

Czym jest baza danych? ... 587
Działanie baz danych ... 588
Rozwiązania alternatywne ... 589
Baza danych a własny mechanizm ... 590
Rodzaje baz danych ... 591
Bazy proste ... 591
Relacyjne bazy danych ... 591
Bazy danych typu klient-serwer ... 592
Wielowarstwowa architektura baz danych ... 593
Borland Database Engine ... 593
Sterowniki bazy danych ... 593
Zbiory danych ... 594
Komponenty bazodanowe ... 595
Praca z komponentami ... 599
Otwieranie i zamykanie zbioru danych ... 599
Nawigowanie wśród rekordów ... 600
Modyfikacja zawartości ... 602
Pola rekordu bazy danych ... 603
Edytor pól ... 607
Pola obliczeniowe ... 608
Pola przeglądowe ... 609
SQL Explorer ... 610
Tworzenie aliasu ... 610
Praca z tabelami ... 611
Tworzenie tabel ... 612
Tworzenie kolumn w kodzie programu ... 612
Filtrowanie danych ... 614
Wykorzystanie właściwości Filter ... 615
Zdarzenie OnFilterRecord ... 616
Wyszukiwanie rekordów ... 617
Metoda Locate ... 618
Metoda FindKey ... 619
Przykładowa aplikacja ... 619
Założenia ... 620
Tworzenie interfejsu ... 621

Kod źródłowy aplikacji ... 625
Ćwiczenia dodatkowe ... 636
Zakładki ... 636
Pozostałe komponenty bazodanowe ... 637
Komponent TDbGrid ... 637
Komponent TDbNavigator ... 637
Komponent TDbText ... 638
Komponent TDbEdit ... 639
Komponent TDbMemo ... 639
Komponent TDbRichEdit ... 639
Komponent TDbImage ... 639
Komponent TDbCheckBox ... 639
Test ... 640
FAQ ... 640
Podsumowanie ... 642

Rozdział 14. dbExpress ... 643

Architektura klient-serwer ... 643
Klient ... 644
Serwer ... 644
Klient-serwer oraz bazy lokalne ... 645
Język SQL ... 646
Baza MySQL ... 646
InterBase ... 647
Schemat tworzenia aplikacji klient-serwer ... 647
Analiza ... 647
Projekt ... 648
Budowa ... 648
Programowanie w SQL ... 648
Klient MySQL ... 649
Tworzenie bazy danych ... 651
Tworzenie tabel ... 651
Zmiana struktury tabeli ... 657
Indeksy ... 659
Dodawanie rekordów ... 663
Wyświetlanie informacji ... 664
Uaktualnianie zawartości ... 670
Usuwanie danych z tabeli ... 671
Eksport i import bazy ... 671
Aplikacja zarządzająca MySQL ... 672
Praca z programem MySQL Control Center ... 673
Praca z tabelami ... 674
InterBase ... 676
Uruchomienie ... 677
Program IBConsole ... 678
MySQL a InterBase ... 681
Tworzenie tabel InterBase ... 690
dbExpress ... 692
dbExpress a BDE ... 692
Połączenie z bazą danych ... 692
Komponent TSQLDataSet ... 698
Komponent TSQLMonitor ... 703

Pozostałe komponenty dbExpress ... 705
Dystrybucja aplikacji dbExpress ... 705
Okno Data Explorer ... 705
Wprowadzanie zmian w tabeli ... 707
Edytor połączeń ... 707
Test ... 708
FAQ ... 709
Podsumowanie ... 709

Rozdział 15. IBX ... 711

Dlaczego IBX? ... 711
Zalety IBX ... 712
Komponenty IBX ... 713
Komponent TIBDatabase ... 713
Komponent TIBDataSet ... 716
Komponenty reprezentujące dane ... 719
TIBSQLMonitor ... 720
Komponenty administracyjne ... 720
Łączenie ze zdalnym serwerem ... 720
Przykład wykorzystania IBX ... 721
Tworzenie bazy danych ... 722
Piszemy aplikację Helion DB! ... 729
Test ... 732
FAQ ... 732
Podsumowanie ... 733

Rozdział 16. Przegląd ADO.NET ... 735

Czym jest ADO? ... 735
Terminologia ... 736
Warstwy dostępu ... 737
Architektura baz danych ... 738
ADO.NET ... 739
Architektura ADO.NET ... 739
Źródło danych ... 739
Dostawca danych ... 740
Zbiory danych ... 740
ADO.NET w praktyce ... 741
Łączenie ze źródłem ODBC ... 741
Łączenie ze źródłem OleDB ... 742
Wysyłanie zapytań (MS Access) ... 743
Sterowniki zarządzane ... 746
Test ... 747
FAQ ... 748
Podsumowanie ... 749

Rozdział 17. BDP.NET ... 751

BDP.NET ... 751
Łączenie z bazą InterBase ... 752
Komponent BDPConnection ... 754
Komponent BdpCommand ... 754
Komponent BdpDataAdapter ... 755
Wyświetlanie zawartości tabeli ... 758

Praca z danymi ... 759
Zapytania SQL ... 760
Dystrybucja aplikacji BDP.NET ... 760
MySQL w ADO.NET ... 761
Korzystanie ze sterownika MySQL ... 762
Komponenty wizualne ... 768
Test ... 770
FAQ ... 770
Podsumowanie ... 771

Część III ... 773

Rozdział 18. XML ... 775

Niezależność XML ... 776
XHTML ... 776
Budowa dokumentu ... 776
Prolog ... 777
Znaczniki ... 778
Atrybuty ... 780
Podstawowa terminologia ... 780
Węzeł główny ... 781
Komentarze ... 781
Przestrzenie nazw ... 781
Składnia przestrzeni nazw ... 782
Przestrzenie nazw i atrybuty ... 782
DTD ... 783
Deklaracja elementu ... 784
Deklaracja atrybutu ... 784
DTD w osobnym pliku ... 786
Encje tekstowe ... 787
XSD ... 788
Nagłówek XSD ... 789
Elementy XSD ... 789
Typy danych ... 790
Typy proste ... 790
XML a bazy danych ... 793
XSL ... 793
DOM ... 794
SAX ... 795
Korzystanie z System.XML ... 795
Ładowanie pliku XML ... 795
Odczyt dowolnego elementu ... 796
Odczyt wartości atrybutów ... 798
Tworzenie pliku XML ... 801
Eksport danych do postaci XML ... 806
Modyfikacja plików ... 808
Test ... 811
FAQ ... 812
Podsumowanie ... 813

Rozdział 19. IntraWeb 815
Czym właściwie jest IntraWeb? 815
Dynamiczne strony WWW 816
CGI, ISAPI, NSAPI 816
PHP 818
Tworzenie projektu IntraWeb 819
Praca z komponentami 820
Praca z kodem 820
Uruchamianie programu 821
Serwer IntraWeb 821
Obsługa serwera 822
Generowanie zdarzeń 822
Zdarzenia zastępcze 823
Kilka formularzy w jednym projekcie 825
Funkcja ShowMessage w IntraWeb 826
Elementy HTML i JavaScript 827
Wysyłanie plików 828
IntraWeb jako rozszerzenie ISAPI 828
Konwertowanie aplikacji do ISAPI 829
Test 831
FAQ 831
Podsumowanie 831

Rozdział 20. ASP.NET 833
Dynamiczne strony WWW 833
ASP 834
ASP.NET 834
ASP i ASP.NET 835
Zmiany w kodzie 835
Kompilacja kodu 835
Migracja do ASP.NET 835
Zgodność ze standardem XHTML 838
Narzędzia 839
Edytor 839
Serwer 840
Instalacja ASP.NET 841
Co trzeba umieć? 842
ASP.NET w Delphi 842
Elementy interfejsu 843
Pierwszy projekt 844
Opcje ASP.NET 845
Web Forms 845
Przestrzeń nazw System.Web.UI 845
Praca z ASP.NET 846
Kontrolki działające po stronie serwera 847
Zdarzenia komponentów 848
Kontrolki Web Forms 851
Code Behind 856
Kontrolki użytkownika 858
Tworzenie kontrolek w Delphi 860
Komponenty .NET w ASP.NET 864
Konfiguracja stron ASP.NET 872

Sesje ... 875
Wysyłanie wiadomości e-mail ... 880
Monitorowanie stron ASP.NET ... 881
Pamięć podręczna ... 881
Bazy danych w ASP.NET ... 883
Łączenie się z bazą ... 883
Kontrolki bazodanowe ... 885
Technologie internetowe ... 887
Komponenty HTML Producer ... 888
WebBroker ... 888
Internet Express ... 888
WebSnap ... 888
IntraWeb ... 889
ASP.NET ... 889
Test ... 889
FAQ ... 890
Podsumowanie ... 891

Rozdział 21. Usługi sieciowe ... 893
Czym są usługi sieciowe? ... 893
Działanie usług sieciowych ... 895
HTTP ... 895
XML ... 895
Infrastruktura usług sieciowych ... 895
Użycie usług sieciowych ... 898
Wyszukiwarka google.com ... 898
Interfejs aplikacji ... 900
Ładowanie usługi sieciowej ... 900
Korzystanie z usługi Web ... 902
Usługi sieciowe w Delphi ... 908
Tworzenie usługi sieciowej ... 909
Podgląd usługi sieciowej ... 911
Usługa Web na stronie ASP.NET ... 915
Plik źródłowy *.asmx ... 917
Bezpieczeństwo usług sieciowych ... 917
Bazy danych ... 919
Projektowanie usługi ... 920
Sprawdzanie usługi sieciowej ... 927
Usługa sieciowa w aplikacji ASP.NET ... 927
Test ... 930
FAQ ... 930
Podsumowanie ... 931

Rozdział 22. Odpowiedzi ... 933
Rozdział 1. ... 933
Rozdział 2. ... 934
Rozdział 3. ... 934
Rozdział 4. ... 935
Rozdział 5. ... 935
Rozdział 6. ... 935
Rozdział 7. ... 936
Rozdział 8. ... 936

Rozdział 9. ... 937
Rozdział 10. ... 937
Rozdział 11. ... 937
Rozdział 13. ... 938
Rozdział 14 ... 938
Rozdział 15 ... 938
Rozdział 16 ... 939
Rozdział 17 ... 939
Rozdział 18 ... 939
Rozdział 19 ... 940
Rozdział 20 ... 940
Rozdział 21 ... 940

Dodatki ... 941

Dodatek A Podstawy języka C# ... 943

C# w Delphi ... 943
Biblioteka klas ... 944
Podstawowa składnia ... 944
C# jako język obiektowy ... 945
Metoda Main() ... 946
Zmienne ... 948
Deklaracja kilku zmiennych ... 949
Przydział danych ... 950
Stałe ... 950
Tablice ... 951
Tablice wielowymiarowe ... 952
Operatory ... 952
Operatory inkrementacji ... 954
Operatory przypisania ... 955
Operator rzutowania ... 955
Instrukcje warunkowe ... 956
Instrukcja if ... 956
Instrukcja switch ... 956
Operator trójoperandowy ... 957
Funkcje w C# ... 958
Parametry funkcji ... 958
Przekazywanie parametrów ... 959
Pętle ... 960
Pętla for ... 960
Pętla foreach ... 961
Pętla while ... 962
Pętla do..while ... 962
Klasy ... 963
Tworzenie instancji klasy ... 963
Metody ... 964
Pola ... 964
Konstruktor ... 965
Właściwości ... 965
Struktury ... 967
Podsumowanie ... 967

Dodatek B Akronimy 969

Dodatek C Spis przestrzeni nazw .NET 971

Dodatek D Słowniczek 977

Dodatek E Zasady pisania kodu 993
Stosowanie wcięć 994
Instrukcje begin i end 994
Styl „wielbłądzi'' w nazwach procedur 995
Stosuj wielkie litery 995
Parametry procedur 996
Instrukcja if 996
Instrukcja case 996
Obsługa wyjątków 997
Klasy 997
Komentarze 997
Pliki i nazwy formularzy 998
Notacja węgierska 998
Czy warto? 998

Skorowidz 1001

5.

Przegląd .NET Framework

W rozdziale 2. omówiłem podstawowe aspekty platformy .NET. Czytelnik powinien już znać podstawowe pojęcia związane z tą technologią, a także umieć pisać proste aplikacje w Delphi. Jak dotąd jednak przykłady prezentowane przeze mnie w rozdziale 3. były oparte na programowaniu dla Win32.

W tym rozdziale zajmiemy się tylko i wyłącznie programowaniem dla platformy .NET oraz dokładniej omówimy technologie związane z tą platformą.

Środowisko .NET Framework obejmuje swym zakresem wszystkie warstwy tworzenia oprogramowania: od systemu operacyjnego po bibliotekę klas (jak np. Windows Forms).

W tym rozdziale:

- przedstawię szczegóły działania środowiska CLR,
- omówię nowe pojęcia: kod zarządzany i niezarządzany,
- opiszę, czym jest CLS i CTS.

Środowisko CLR

Wspólne środowisko uruchomieniowe (CLR) stanowi podstawowy element platformy .NET Framework. W momencie uruchomienia aplikacji .NET, CLR odpowiada za jej sprawne wykonanie (załadowanie do pamięci), przydział pamięci, obsługę błędów itp. W standardowym modelu programowania — Win32, za tego typu czynności odpowiadał zwyczajnie system operacyjny (Windows). Po zainstalowaniu na komputerze środowiska .NET Framework, odpowiednie biblioteki systemu pozwalają na rozpoznanie, czy dany program jest aplikacją Win32, czy też

.NET. Jeżeli jest to aplikacja .NET, uruchomione zostaje środowisko CLR, pod kontrolą którego działa program. To, co dzieje się „w tle" nas nie interesuje, nie trzeba się tym przejmować.

W jaki jednak sposób system operacyjny rozpoznaje, która aplikacja jest aplikacją .NET? Dzieje się tak dlatego, że aplikacja wykonywalna .NET (plik *.exe*) jest inaczej zbudowana niż standardowe programy Win32. Tę kwestię postaram się wyjaśnić w paru kolejnych podrozdziałach.

Kod pośredni IL

Kompilatory działające pod kontrolą systemu Windows kompilują kody źródłowe do postaci 32-bitowego kodu maszynowego. W efekcie otrzymujemy aplikacje *.exe* czy biblioteki *.dll*. Taki sposób uniemożliwia przeniesienie aplikacji na urządzenia czy systemy, które działają pod kontrolą innych procesorów. Może także stwarzać problemy z innymi wersjami systemu operacyjnego (w tym wypadku Windows). Wszystko dlatego, że aplikacje wykonywalne komunikują się z API, które może różnić się w poszczególnych wersjach systemu.

Rozwiązaniem tego problemu jest kompilacja programu do kodu pośredniego, nazywanego *Common Language Infrastructure* (z ang. *architektura wspólnego języka*, CLI).



Wskazówka

CLI na platformie .NET jest często nazywany MSIL (*Microsoft Intermediate Language*) lub po prostu — IL.

Kod pośredni IL przypomina kod języka Asembler:

```
.method family hidebysig virtual instance void
        Dispose(bool Disposing) cil managed
{
  // Code size       30 (0x1e)
  .maxstack  2
  IL_0000:  ldarg.1
  IL_0001:  brfalse.s  IL_0016
  IL_0003:  ldarg.0
  IL_0004:  ldfld      class [System]System.ComponentModel.Container
                       WinForm.TWinForm1::Components
  IL_0009:  brfalse.s  IL_0016
  IL_000b:  ldarg.0
  IL_000c:  ldfld      class [System]System.ComponentModel.Container
                       WinForm.TWinForm1::Components
  IL_0011:  callvirt   instance void [System]System.ComponentModel.Container::Dispose()
  IL_0016:  ldarg.0
  IL_0017:  ldarg.1
  IL_0018:  call       instance void
                       [System.Windows.Forms]System.Windows.Forms.Form::Dispose(bool)
  IL_001d:  ret
} // end of method TWinForm1::Dispose
```

W takiej postaci kod IL jest kompilowany do kodu maszynowego, który może już zostać uruchomiony. Tak właśnie działa język Java, z którego pomysł zaczerpnął Microsoft projektując platformę .NET.

Wszystko to jest możliwe dzięki tzw. maszynom wirtualnym, czyli aplikacjom przystosowanym do konkretnej wersji systemu/procesora. Środowisko CLR odpowiada za kompilację kodu pośredniego na maszynowy, w trakcie uruchamiania aplikacji. Dzieje się to jednak na tyle szybko, że dla użytkownika jest to niezauważalne.



Wskazówka

Możliwe jest również jednorazowe skompilowanie danego programu od razu na kod maszynowy, dzięki temu przy każdym uruchamianiem programu są oszczędzane zasoby systemowe, potrzebne do uruchomienia kompilatora JIT (ang. *Just-In-Time*).

Kod zarządzany i niezarządzany

Platforma .NET definiuje dwa nowe pojęcia: kod zarządzany (*managed code*) oraz niezarządzany (ang. *unmanaged code*), które są istotne z punktu widzenia CLR. Kod niezarządzany jest zwykłym kodem, wykonywanym poza środowiskiem .NET, zatem określenie to oznacza stare aplikacje kompilowane dla środowiska Win32. Natomiast — jak nietrudno się domyśleć — kod zarządzany jest wykonywany pod kontrolą CLR.

Zmiany w Delphi 2005 oraz w .NET w porównaniu z Win32 są na tyle duże, że problemem staje się współdziałanie obu rodzajów aplikacji (aplikacji .NET oraz Win32), jak również korzystanie z zasobów starszych aplikacji Win32 — np. bibliotek DLL. Dlatego też w .NET w tym celu wykorzystuje się mechanizm zwany *marshalingiem* [1], który jest związany z określeniem sposobu, w jaki dane mają być przekazywane z kodu niezarządzanego do zarządzanego. Tym jednak nie należy się teraz przejmować — podejmę ten temat w rozdziale 11.

Moduł zarządzany

Każdy moduł zarządzany jest przenośnym plikiem systemu Windows. Moduły zarządzane są generowane przez kompilatory zgodne z platformą .NET, a w ich skład wchodzą następujące elementy:

- ✧ Nagłówek PE (ang. *PE Header*) — standardowy nagłówek pliku wykonywalnego systemu Windows,
- ✧ Nagłówek CLR (ang. *CLR Header*) — dodatkowe informacje, charakterystyczne dla danego środowiska CLR,

[1] Niestety nie znalazłem dobrego, polskiego odpowiednika tego słowa, które w pełni oddawałoby istotę rzeczy.

- Metadane (ang. *metadata*) — informacje o typach modułów używanych w programie oraz ich wzajemnych powiązaniach,
- Kod zarządzany (ang. *managed code*) — wspólny kod generowany przez kompilatory .NET (kod IL).



Wskazówka

Informacje o metadanych znajdują się w 8. rozdziale tej książki.

Podzespoły

To jest bardzo ważne pojęcie, którym nieraz będę się posługiwał w tej książce. W najprostszym ujęciu podzespołem (ang. *assembly*) nazywamy każdą aplikację działającą pod kontrolą .NET.

Każdy moduł zarządzany wymaga podzespołu. Jeden podzespół może zawierać jeden lub więcej modułów zarządzanych. Podzespół może zawierać również inne pliki, takie jak dokumenty, grafikę itp.

Na tym etapie zagadnienie to może wydać się Czytelnikowi niezwykle skomplikowane. Podzespoły mogą zawierać jeden lub więcej modułów zarządzanych, w których z kolei znajdują się kod zarządzany oraz metadane.

Działanie CLR

Wspomniałem wcześniej, że CLR zajmuje się uruchamianiem aplikacji napisanej w .NET oraz ogólnie — zarządzaniem procesem jej działania. Prześledźmy proces uruchamiania aplikacji w .NET. Oto kilka etapów, które przechodzi aplikacja od momentu, gdy użytkownik zarządzi jej uruchomienie:

- ładowanie klasy (ang. *Class Loader*),
- weryfikacja,
- kompilacja.

Class Loader

Dotychczas jedynym formatem, rozpoznawanym przez systemy Windows jako aplikacja wykonywalna, był stary format PE. Stary format PE zawierał skompilowany kod maszynowy aplikacji. Instalując bibliotekę .NET w systemie, dodawane jest uaktualnienie mówiące o nowym formacie PE, dzięki czemu system jest w stanie rozpoznać również nowy format plików wykonywalnych.

Skoro teraz aplikacja .NET jest rozpoznawana przez system, ten w momencie jej uruchamiania oddaje sterowanie do CLR. CLR odczytuje zawartość pliku oraz listę klas używanych w programie.

Lista klas jest odczytywana z przeróżnych miejsc — szczególnie jest to manifest oraz metadane, a także plik *.config*, który może być dołączany do programu. Po odczytaniu klas następuje obliczenie ilości pamięci potrzebnej, aby załadować klasy. Następnie klasy są ładowane do pamięci.

Weryfikacja

Po załadowaniu klas do pamięci zawartość programu, czyli metadane oraz kod IL, zostają poddane weryfikacji. Jest to ważny etap, gdyż w przypadku niepowodzenia kod IL nie zostanie przekazany kompilatorowi JIT.

Kompilator JIT

Kompilator JIT odgrywa znaczącą rolę w procesie uruchamiania aplikacji. Po weryfikacji kodu jest on przekazywany do kompilatora, który kompiluje go do kodu maszynowego, a następnie gotowy program zostaje załadowany do pamięci. Znajduje się on w tej pamięci do czasu zakończenia działania aplikacji.

System CTS

Wiadomo już, czym są typy języka programowania. Wspólny system typów (*Common Type System*) jest bogatym zbiorem typów opracowanych przez firmę Microsoft na potrzeby .NET. W rzeczywistości, podczas programowania w Delphi z użyciem VCL.NET lub *WinForms* i zastosowaniem typu `Integer`, korzystamy z typu `System.Int32`, który należy do specyfikacji CTS.

Otwórzmy teraz okno repozytorium i wybierzmy kategorię *Delphi for .NET Projects*, a następnie *Console Application*. Można zadeklarować dwie zmienne i spróbować skompilować program:

```
var
  I1 : Integer;
  I2 : System.Int32;
```

Kompilacja odbędzie się bezproblemowo. Typy `Integer` oraz `System.Int32` są sobie równoważne. Typy takie jak `Integer` czy `String` są zachowane ze względu na kompatybilność projektów ze starszymi wersjami Delphi. Programista nie musi uczyć się nazw nowych typów — wykorzystuje nazwy dobrze mu znane i kompatybilne z .NET. Typy języka Delphi oraz ich odpowiedniki w CTS znajdują się w tabeli 5.1.

System CTS jest składnikiem CLR, odpowiada za weryfikowanie i zarządzanie tymi typami. Wszystkie typy, również te przedstawione w tabeli 5.1 wywodzą się z głównej klasy — `System.Object`.

Tabela 5.1. *Typy CTS*

Typ Delphi	Typ .NET Framework
String	System.String
Boolean	System.Boolean
Char	System.Char
Double	System.Double
Integer	System.Int32
Smallint	System.Int16
Word	System.UInt16
Int64	System.Int64
Byte	System.Byte

Specyfikacja CLS

Do tej pory możliwości w komunikowaniu się pomiędzy aplikacjami były nieco ograniczone. Kod aplikacji w środowisku Win32 może być dzielony pomiędzy biblioteki DLL. Po umieszczeniu funkcji w bibliotece DLL, istnieje możliwość ich eksportu, co z kolei pozwala na ich wykorzystanie przez aplikacje EXE.

W środowisku .NET aplikacje mają pełną zdolność do komunikowania się. Jeden podzespół może wykorzystywać funkcje drugiego i na odwrót. Wszystko to dzięki kompilacji do kodu pośredniego IL, do którego są kompilowane wszystkie aplikacje, bez względu na to, czy są pisane w C#, czy w Delphi. Tak więc, jeżeli napisano program do szyfrowania danych, można (jeżeli tylko programista tego zechce) udostępnić jego funkcjonalność innym podzespołom. Praktyczne przykłady tego zagadnienia zaprezentuję w rozdziale 8.

Nie tylko język pośredni ma tu znaczenie, ale również specyfikacja CLS (ang. *Common Language Specification*, czyli *wspólna specyfikacja języków*). Jest to zestaw reguł określających nazewnictwo oraz inne kluczowe elementy języka programowania. Jeśli projektanci języka programowania, który docelowo ma działać dla .NET, chcą, aby był on kompatybilny z CLS oraz miał zdolność do komunikowania się z pozostałymi podzespołami, muszą dostosować swój produkt do określonych wymagań.



Wskazówka

Na stronach firmy Microsoft można także znaleźć dodatkowe informacje na temat specyfikacji CLS:

http://msdn.microsoft.com/net/ecma/.

Biblioteka klas

Czytelnik wciąż spotyka się ze słowami: *klasa*, *obiekt*, których zresztą często używam w tym rozdziale. Wspominałem wcześniej, że na platformie Win32 aplikacje korzystały z WinAPI, czyli z zestawu funkcji pomocnych przy programowaniu. Wkrótce po tym pojawiły się takie biblioteki jak VCL, które jeszcze bardziej ułatwiały programowanie (VCL korzysta z funkcji WinAPI).

Biblioteka klas .NET Framework (*.NET Framework Class Library*) stanowi zestaw setek klas, bibliotek, interfejsów, typów, które mają zastąpić WinAPI. W założeniu FCL ma połączyć funkcjonalność WinAPI oraz dodatkowych bibliotek, takich jak VCL czy MFC (ang. *Microsoft Fundation Classes*).

Dla przykładu — w rozdziale 3. korzystaliśmy z funkcji `Writeln`, `Readln`, które są funkcjami Delphi umożliwiającymi operacje na konsoli. Odpowiednikiem tych funkcji na platformie .NET jest klasa `Console` oraz metody `WriteLine` oraz `ReadLine`. Oto prosty przykład programu, który pobiera informacje o imieniu użytkownika. Program korzysta z funkcji konsolowych biblioteki FCL:

```
program Project3;

{$APPTYPE CONSOLE}

var
  S : System.String;

begin
  Console.WriteLine('Cześć! Podaj swoje imię!');
  S := Console.ReadLine;

  Console.WriteLine('Cześć ' + S + '! Ja mam na imię Adam!');
  Console.ReadLine;
end.
```