

Евгений Марков
Владимир Никифоров



Delphi 2005

для .NET

- Архитектура Microsoft .NET
- Языки программирования Delphi и C#
- Windows Forms
- VCL.NET
- ADO.NET, DBP.NET
- ESO

**Наиболее
полное
руководство**

В ПОДЛИННИКЕ®

**Евгений Марков
Владимир Никифоров**

Delphi 2005 для .NET

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.068+800.92Delphi2005
ББК 32.973.26-018.1
М27

Марков Е. П., Никифоров В. В.

М27 Delphi 2005 для .NET. — СПб.: БХВ-Петербург, 2005. — 896 с.: ил.
ISBN 5-94157-701-X

Рассмотрены практические аспекты программирования в Borland Delphi 2005 для .NET. Описаны вопросы реализации .NET в Delphi, а также синтаксис и объектные модели двух языков программирования — Delphi и C#. Показаны особенности разработки приложений для двух основных типов приложений — Windows Forms и VCL.NET. Большое внимание уделено созданию приложения для работы с базами данных. Рассмотрены технологии ADO.NET и BDP.NET, а также создание приложений VCL.NET, поддерживающих известные технологии доступа к данным — Borland DataBase Engine .NET, dbExpress .NET, InterBase Express .NET, dbGo и др. Описаны приемы создания приложений на основе технологии ECO, использующей перспективную архитектуру разработки приложений MDA. Излагаемый материал сопровождается примерами.

Для подготовленных программистов

УДК 681.3.068+800.92Delphi2005
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Леонид Кочин</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн обложки	<i>Игоря Цырульникова</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.07.05.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 72,24.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ОАО "Техническая книга"
190005, Санкт-Петербург, Измайловский пр., 29

ISBN 5-94157-701-X

© Марков Е. П., Никифоров В. В., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Оглавление

ЧАСТЬ I. ОСНОВЫ.....	19
Глава 1. Что и как можно разрабатывать в Delphi 2005.....	21
Многоязычная среда разработки.....	21
Язык программирования Delphi	22
Язык программирования C#	24
Язык программирования Visual Basic.....	24
Программные платформы.....	25
Компоненты .NET и VCL	26
Какие приложения можно создавать в Delphi.....	27
Перенос существующих приложений на платформу .NET.....	29
Резюме	30
Глава 2. Введение в архитектуру Microsoft .NET	31
Три вопроса о .NET	33
Что это такое?	34
Зачем это нужно?.....	35
Как это работает?.....	37
Сборки, метаданные и промежуточный код	39
Язык MSIL.....	40
Сборка (Assembly)	41
Метаданные.....	43
Особенности разработки приложений .NET	44
Пространства имен.....	45
Общая система типов (Common Type System)	46
Встроенные и определенные пользователем типы данных.....	46
Перечисления	46
Классы.....	46
Интерфейсы	47
Делегаты	47
Указатели.....	47
Массивы	47
Правила межязыкового взаимодействия Common Language Specification	47

.NET Framework.....	48
Common Language Runtime	48
Компиляторы.....	51
Безопасность приложения	51
Библиотека базовых классов .NET.....	52
.NET Framework SDK.....	52
Высокоуровневые службы	53
ASP.NET	53
ADO.NET.....	54
Windows Forms.....	54
Резюме.....	54
Глава 3. Язык программирования Delphi	57
Объектно-ориентированное программирование	57
Классы и объекты	57
Поля, свойства и методы	60
События	61
Инкапсуляция	62
Наследование.....	62
Полиморфизм	63
Методы	65
<i>Abstract</i>	65
<i>Sealed</i>	65
<i>Static</i>	66
<i>Virtual</i> и <i>Dynamic</i>	66
<i>Override</i>	67
Перегрузка методов.....	69
Области видимости свойств и методов.....	70
Пространство имен.....	71
Синтаксис языка Delphi	72
Типы данных.....	72
Функции преобразования типов	74
Операторы	76
Структурные типы	76
Циклы	78
Обработка исключительных ситуаций.....	79
Резюме.....	80
Глава 4. Язык программирования C#.....	81
Типы данных.....	83
Пространство имен.....	86
Классы	88
<i>Static</i>	89
<i>Virtual</i> и <i>Override</i>	89

<i>Abstract</i>	91
<i>Sealed</i>	92
Конструкторы	93
Синтаксис языка C#	95
Операторы	95
Константы	96
Строки	96
Формат вывода и форматирование строк	96
Массивы	97
Циклы	99
Условные предложения	101
Обработка исключительных ситуаций	103
Ввод/вывод	105
Резюме	108
Глава 5. Реализация .NET в Delphi	109
Общая система типов (Common Type System)	110
Типы данных	110
Классы	111
Интерфейсы	111
Делегаты	112
Правила межязыкового взаимодействия (Common Language Specification)	112
Пространства имен	112
Компиляция в промежуточный язык, сборки, метаданные	116
Управление памятью и сборка мусора	119
Реализация высокоуровневых служб .NET в Delphi	120
Windows Forms	120
ADO.NET	121
ASP.NET	121
Резюме	121
Глава 6. Инструментарий разработчика	123
Интегрированная среда разработки приложений	123
Окно приветствия <i>Welcome Page</i>	124
Палитра инструментов	125
Редактор кода	126
Режим <i>Sync Edit</i>	127
Рефакторинг	128
Контекстная помощь	129
Ошибки	130
Список точек останова	130
Резюме	131

ЧАСТЬ II. ПРИЛОЖЕНИЯ WINDOWS FORMS	133
Глава 7. Приложение и проект	135
Главный модуль проекта.....	135
Файл формы	137
Классы <i>Control, UserControl, Form</i>	140
Классы элементов управления (<i>Controls</i>).....	140
Классы компонентов (<i>Components</i>).....	141
Классы диалоговых окон (<i>Common Dialog Boxes</i>).....	141
Описание экземпляра класса	141
Резюме	146
Глава 8. Элементы управления	147
Компонент <i>Label</i>	147
Компонент <i>LinkLabel</i>	149
Компонент <i>TextBox</i>	150
Компонент <i>Button</i>	152
Компонент <i>Panel</i>	154
Компонент <i>CheckBox</i>	155
Компонент <i>RadioButton</i>	158
Компонент <i>ListBox</i>	160
Компонент <i>ComboBox</i>	163
Компонент <i>CheckedListBox</i>	165
Компонент <i>PictureBox</i>	168
Компонент <i>ImageList</i>	170
Компоненты <i>HScrollBar</i> и <i>VScrollBar</i>	171
Компонент <i>NumericUpDown</i>	172
Компонент <i>DomainUpDown</i>	173
Компонент <i>DateTimePicker</i>	175
Компонент <i>MonthCalendar</i>	177
Компонент <i>Timer</i>	178
Резюме	180
Глава 9. Стандартные программные механизмы.....	181
Интерфейс переноса Drag and Drop	181
Усовершенствованное масштабирование	188
Управление мышью	189
Резюме	190
Глава 10. Меню и панель инструментов.....	191
Компонент <i>MainMenu</i>	191
Компонент <i>ContextMenu</i>	194
Компонент <i>ToolBar</i>	197
Резюме	201

Глава 11. Диалоги.....	203
Стандартные компоненты диалога	203
Компонент <i>OpenFileDialog</i>	204
Компонент <i>SaveFileDialog</i>	207
Компоненты <i>PrintDialog</i> , <i>PrintDocument</i> , <i>PageSetupDialog</i> и <i>PrintPreviewDialog</i>	209
Компонент <i>FontDialog</i>	213
Компонент <i>ColorDialog</i>	214
Резюме	215
Глава 12. Состояние приложения	217
Компонент <i>StatusBar</i>	217
Компонент <i>ProgressBar</i>	220
Компонент <i>TrackBar</i>	222
Компонент <i>ToolTip</i>	224
Компонент <i>NotifyIcon</i>	224
Компонент <i>HelpProvider</i>	226
Компонент <i>ErrorProvider</i>	228
Резюме	230
Глава 13. Ввод данных	231
Ввод и обработка текста	231
Класс <i>Font</i>	231
Компонент <i>TextBox</i>	232
Компоненты <i>ComboBox</i> и <i>ListBox</i>	234
Компонент <i>DomainUpDown</i>	235
Компонент <i>RichTextBox</i>	235
Ввод данных в числовых форматах	238
Ввод даты и времени.....	239
Компонент <i>MonthCalendar</i>	240
Компонент <i>DateTimePicker</i>	242
Ввод двоичных данных.....	245
Резюме	247
Глава 14. Работа с файлами.....	249
Файл как объект файловой системы	249
Класс <i>File</i>	250
Класс <i>FileInfo</i>	255
Пути и каталоги	257
Класс <i>Directory</i>	258
Класс <i>DirectoryInfo</i>	260
Поиск файла.....	262
Потоки	264
Класс <i>StreamReader</i>	264

Класс <i>StreamWriter</i>	266
Класс <i>FileStream</i>	268
Асинхронный режим доступа к данным	270
Класс <i>MemoryStream</i>	272
Операции ввода/вывода	273
Создание файла и запись данных	273
Открытие файла и чтение данных	275
Резюме	276
Глава 15. Перечислители, списки, коллекции	277
Что такое коллекция	278
Как устроена коллекция	279
Интерфейс <i>ICollection</i>	280
Интерфейс <i>IList</i>	280
Интерфейс <i>IEnumerable</i>	282
Интерфейс <i>IEnumerator</i>	282
Класс <i>CollectionBase</i>	283
Коллекция строк	283
Управление коллекциями	284
Резюме	285
Глава 16. Иерархическое представление данных	287
Компонент <i>TreeView</i>	288
Класс <i>TreeNode</i>	293
Компонент <i>ListView</i>	296
Класс <i>ListViewItem</i>	302
Класс <i>ListViewSubItem</i>	304
Резюме	304
Глава 17. Использование XML	305
Что такое XML	305
Основы синтаксиса XML	308
Пролог	310
Определение	311
Тело документа. Корневой элемент	312
Объектная модель документа	313
Интерфейсы семейства <i>IDOMNode</i>	314
Свойства <i>nodeType</i> , <i>nodeName</i> и <i>nodeValue</i>	316
Свойства и методы, управляющие другими вершинами	317
Пространства имен	319
Интерфейс <i>IDOMDocument</i>	320
Пример создания приложения, использующего модель DOM	320
Реализация модели DOM в приложениях .NET	322
Класс <i>XmlNode</i>	323
Класс <i>XmlElement</i>	328

Класс <i>XMLAttribute</i>	329
Класс <i>XMLDocument</i>	329
Резюме	334
ЧАСТЬ III. ПРИЛОЖЕНИЯ VCL.NET	335
Глава 18. Приложение и проект	337
Проект как основа разработки приложения	337
Класс <i>TApplication</i>	345
Атрибуты приложения	351
Обработка сообщений.....	352
Реакция на действия пользователей.....	355
Система помощи.....	356
Резюме	357
Глава 19. Меню и действия	359
Редактор меню	360
Как работает меню	361
Главное меню приложения	364
Всплывающее меню	367
Действия. Компонент <i>TActionList</i>	368
События, связанные с действиями.....	369
Свойства, распространяемые на клиентов действия	371
Прочие свойства	372
Стандартные действия	373
Категория <i>Edit</i>	376
Категория <i>Search</i>	376
Категория <i>Help</i>	376
Категория <i>File</i>	377
Категория <i>Dialog</i>	377
Категория <i>Window</i>	377
Категория <i>Tab</i>	377
Категория <i>List</i>	377
Категория <i>Internet</i>	379
Категория <i>Format</i>	380
Категория <i>Dataset</i>	380
Категория <i>Tools</i>	380
Компонент <i>TActionManager</i>	381
Изменение и настройка внешнего вида панелей.....	383
Ручное редактирование коллекций панелей и действий.....	384
Резюме	387
Глава 20. Списки и коллекции	389
Список строк.....	390
Класс <i>TStrings</i>	390
Класс <i>TStringList</i>	391

Список указателей	399
Класс <i>TList</i>	399
Пример использования списка указателей	402
Коллекции	406
Класс <i>TCollection</i>	407
Класс <i>TCollectionItem</i>	408
Резюме	408
Глава 21. Файлы и потоки.....	411
Потоки	411
Базовые классы <i>TStream</i> и <i>THandleStream</i>	412
Класс <i>TFileStream</i>	414
Класс <i>TMemoryStream</i>	416
Класс <i>TStringStream</i>	416
Резюме	417
Глава 22. Использование графики.....	419
Графические инструменты Delphi.....	419
Класс <i>TFont</i>	419
Класс <i>TPen</i>	421
Класс <i>TBrush</i>	422
Класс <i>TCanvas</i>	422
Класс <i>TGraphic</i>	427
Класс <i>TPicture</i>	429
Класс <i>TMetafile</i>	431
Класс <i>TIcon</i>	432
Класс <i>TBitmap</i>	433
Компонент <i>TImage</i>	435
Использование диалогов для загрузки и сохранения графических файлов	436
Класс <i>TClipboard</i>	437
Класс <i>TScreen</i>	439
Резюме	441
ЧАСТЬ IV. ПРИЛОЖЕНИЯ БАЗ ДАННЫХ .NET.....	443
Глава 23. Архитектура приложений баз данных .NET	445
Как работает приложение баз данных	446
Соединение с источником данных.....	450
Адаптер данных.....	450
Набор данных	451
Отображение данных	452
Методика доступа к данным в приложении БД.....	452
Резюме	453

Глава 24. Приложения ADO.NET	455
Основы ADO	457
Провайдеры ADO	459
Соединение с источником данных	460
Пулинг соединений	464
Управление транзакциями	465
Обработка ошибок	470
Использование адаптера данных	472
Отбор данных и генерация набора данных	472
Выборка из одной таблицы	476
Вставка, изменение, удаление данных командами SQL	477
Схема связывания данных	480
Параметры	483
Набор данных	489
Таблицы данных	491
Колонка таблицы	495
Автоинкрементные колонки	498
Вычисляемые колонки	499
Агрегатные колонки	501
Фильтрация и поиск данных	502
Первичный ключ таблицы	503
Запись таблицы	504
Управление данными	510
Сортировка, поиск данных	513
Ограничения	515
Отношения	519
Просмотры	521
Команды SQL	525
Пользовательский интерфейс	530
Компоненты <i>Label</i> и <i>LinkLabel</i>	532
Компонент <i>Button</i>	532
Компонент <i>TextBox</i>	532
Компонент <i>CheckBox</i>	532
Компонент <i>RadioButton</i>	533
Компонент <i>ComboBox</i>	533
Компонент <i>ListBox</i>	533
Компонент <i>CheckedListBox</i>	534
Компоненты <i>TreeView</i> и <i>ListView</i>	534
Компонент <i>DateTimePicker</i>	534
Компоненты <i>TrackBar</i> , <i>ProgressBar</i> , <i>VScrollBar</i> и <i>HScrollBar</i>	534
Компонент <i>NumericUpDown</i>	534
Компонент <i>DomainUpDown</i>	535
Компоненты <i>GroupBox</i> и <i>StatusBar</i>	535
Компонент <i>RichTextBox</i>	535

Компонент <i>DataGrid</i>	535
Подключение данных	535
Отображение данных	537
Навигация по записям таблицы	538
Работа с ячейками таблицы	538
Сортировка данных	539
Отображение отношений между таблицами	539
Резюме	540
Глава 25. Приложения BDP	543
Доступ к данным	544
Механизм отображения "живых" данных	547
Компонент <i>BdpCommandBuilder</i>	548
Перенос данных между разными источниками данных	549
Работа с гетерогенными источниками данных	551
Обмен гетерогенными данными	554
Многотабличный набор данных	555
Работа с удаленными источниками данных	556
Удаленный сервер приложения	558
Клиентское приложение	560
Пример разработки распределенного приложения	562
Резюме	563
ЧАСТЬ V. ПРИЛОЖЕНИЯ БАЗ ДАННЫХ VCL.NET	565
Глава 26. Архитектура приложений баз данных VCL.NET	567
Набор данных	568
Абстрактный набор данных	568
Стандартные компоненты	574
Компонент таблицы	575
Компонент запроса	577
Компонент хранимой процедуры	580
Индексы в наборе данных	581
Механизм подключения индексов	582
Список описаний индексов	582
Описание индекса	583
Использование описаний индексов	584
Параметры запросов и хранимых процедур	586
Класс <i>TParams</i>	589
Класс <i>TParam</i>	590
Состояния набора данных	592
Поля	595
Объекты полей	595
Статические и динамические поля	598
Класс <i>TField</i>	600

Виды полей	604
Поля синхронного просмотра	605
Вычисляемые поля	607
Внутренние вычисляемые поля	608
Агрегатные поля.....	608
Объектные поля	609
Ограничения.....	610
Как работает приложение баз данных	612
Модуль данных	615
Подключение набора данных	616
Настройка компонента <i>TDataSource</i>	617
Отображение данных	619
Резюме	620
Глава 27. Процессор баз данных BDE.NET	621
Архитектура и функции BDE	622
Псевдонимы баз данных и настройка BDE	626
Соединение с источником данных.....	635
Компоненты доступа к данным.....	640
Класс <i>TBDEDataSet</i>	640
Класс <i>TDBDataSet</i>	645
Компонент <i>TTable</i>	646
Компонент <i>TQuery</i>	652
Компонент <i>TStoredProc</i>	654
Резюме	656
Глава 28. Технология dbExpress .NET.....	657
Доступ к данным dbExpress.....	658
Драйверы доступа к данным.....	659
Соединение с сервером баз данных	660
Управление наборами данных.....	665
Транзакции.....	668
Использование компонентов наборов данных.....	669
Класс <i>TCustomSQLDataSet</i>	670
Компонент <i>TSQLDataSet</i>	672
Компонент <i>TSQLTable</i>	673
Компонент <i>TSQLQuery</i>	674
Компонент <i>TSQLStoredProc</i>	675
Компонент <i>TSimpleDataSet</i>	676
Способы редактирования данных	679
Интерфейсы dbExpress	685
Интерфейс <i>ISQLDriver</i>	685
Интерфейс <i>ISQLConnection</i>	685
Интерфейс <i>ISQLCommand</i>	687
Интерфейс <i>ISQLCursor</i>	688

Отладка приложений с технологией dbExpress	689
Распространение приложений с технологией dbExpress.....	691
Резюме	692
Глава 29. Технология InterBase Express для .NET	693
Механизм доступа к данным InterBase Express	694
Компонент <i>TIBDatabase</i>	694
Компонент <i>TIBTransaction</i>	699
Компоненты доступа к данным.....	703
Область дескрипторов <i>XSQLDA</i>	705
Структура <i>XSQLVAR</i>	706
Компонент <i>TIBTable</i>	707
Компонент <i>TIBQuery</i>	708
Компонент <i>TIBStoredProc</i>	710
Компонент <i>TIBDataSet</i>	710
Компонент <i>TIBSQL</i>	712
Обработка событий	715
Информация о состоянии базы данных.....	717
Компонент <i>TIBDatabaseInfo</i>	717
Компонент <i>TIBSQLMonitor</i>	719
Резюме	720
Глава 30. Технология dbGo	721
Компоненты dbGo	721
Механизм соединения с хранилищем данных ADO.....	722
Компонент <i>TADOConnection</i>	723
Настройка соединения	723
Управление соединением.....	728
Доступ к связанным наборам данных и командам ADO.....	731
Объект ошибок ADO	734
Транзакции.....	734
Наборы данных ADO	735
Класс <i>TCustomADODataset</i>	736
Набор данных	736
Курсор набора данных.....	737
Локальный буфер	739
Состояние записи	740
Фильтрация.....	742
Поиск.....	743
Сортировка	743
Команда ADO.....	744
Групповые операции.....	745
Параметры	746
Класс <i>TParameters</i>	747
Класс <i>TParameter</i>	748

Компонент <i>TADODataSet</i>	749
Компонент <i>TADOTable</i>	750
Компонент <i>TADOQuery</i>	751
Компонент <i>TADOStoredProc</i>	751
Команды ADO	752
Объект ошибок ADO.....	754
Резюме.....	754

ЧАСТЬ VI. РАСПРЕДЕЛЕННЫЕ ПРИЛОЖЕНИЯ БАЗ ДАННЫХ

VCLNET	757
---------------------	------------

Глава 31. Архитектура распределенных приложений 759

Парадигма распределенных вычислений	760
Архитектура распределенных приложений	763
Уровень представления данных	767
Уровень обработки данных	768
Уровень управления данными.....	768
Уровень хранения данных	769
Расширения базовых уровней	770
Уровень бизнес-интерфейса	771
Уровень доступа к данным	771
Резюме	771

Глава 32. Технология DataSnap..... 773

Структура многозвенного приложения в Delphi.....	774
Трехзвенное приложение в Delphi 2005	776
Сервер приложения	777
Клиентское приложение.....	778
Механизм удаленного доступа к данным DataSnap.....	779
Компонент <i>TDCOMConnection</i>	779
Вспомогательные компоненты-брокеры соединений	781
Компонент <i>TSimpleObjectBroker</i>	781
Компонент <i>TLocalConnection</i>	783
Компонент <i>TSharedConnection</i>	783
Компонент <i>TConnectionBroker</i>	784
Резюме	785

Глава 33. Клиент многозвенного распределенного приложения..... 787

Структура клиентского приложения.....	788
Компонент <i>TClientDataSet</i>	789
Получение данных от компонента-провайдера.....	790
Кэширование и редактирование данных	792
Управление запросом на сервере	794
Использование индексов.....	795

Сохранение набора данных в файлах.....	797
Работа с данными типа BLOB	798
Представление данных в формате XML	799
Агрегаты.....	799
Объекты-агрегаты.....	800
Агрегатные поля	802
Группировка и использование индексов	804
Вложенные наборы данных.....	804
Дополнительные свойства полей клиентского набора данных	805
Обработка ошибок.....	806
Пример "тонкого" клиента	809
Соединение клиента с сервером приложения	814
Наборы данных клиентского приложения.....	815
Сервер приложения	816
Резюме	818
Глава 34. Преобразование пакетов данных в формате XML.....	819
Преобразование данных в формате XML.....	819
Схема преобразования данных XML	820
Формат пакета данных Delphi	821
Утилита XML Mapper	822
Выбор исходного файла	823
Создание пакета данных и документа XML и сохранение преобразованных данных	824
Связывание элементов XML и полей пакета данных.....	825
Создание трансформационного файла и преобразование данных	826
Резюме	827
ЧАСТЬ VII. ПРИЛОЖЕНИЯ ESO	829
Глава 35. Архитектура MDA	831
Основные понятия	832
Архитектура разработки приложений на основе моделей.....	833
Типы моделей	834
Уровни модели.....	834
Этапы разработки	835
Преобразование модели PIM в PSM	837
Многоплатформенные модели	838
Технологический фундамент	839
Что нужно знать об UML	839
OCL	841
Стандарты метамоделирования.....	841
XML и XMI	841
Резюме	842

Глава 36. Технология ESO	843
Что такое ESO	844
Проект ESO.....	845
Инструментарий ESO	850
Менеджер модели <i>Model View</i>	850
Редактор UML.....	851
Дизайнер объектного пространства	855
Общая методика разработки приложений ESO	857
Платформенно-независимая модель	858
Пакет.....	859
Класс	859
Атрибут	861
Оператор	862
Отношения	862
Ассоциация	863
Обобщение\воплощение	865
Платформенно-зависимая модель	865
Объектное пространство.....	866
Класс объектного пространства	867
Общие управляющие компоненты-дескрипторы	868
Компонент <i>ReferenceHandle</i>	871
Компонент <i>VariableHandle</i>	872
Компонент <i>ExpressionHandle</i>	873
Компонент <i>OclVariables</i>	874
Компонент <i>OclPSHandle</i>	874
Пользовательский интерфейс	875
Автоматическая генерация форм	876
Управление данными	877
Управление списками.....	878
Drag and Drop	879
Связывание визуальных компонентов с данными	879
Доступ к данным	880
Использование языка OCL	882
Пример приложения ESO	883
Вычислительно-независимая модель	883
Платформенно-независимая модель	884
Платформенно-зависимая модель	887
Резюме	889
Предметный указатель	891

ГЛАВА 3



Язык программирования Delphi

Начиная знакомство с Borland Developer Studio и следующей версией языка Delphi — очередным этапом развития столь мощного объектно-ориентированного языка, — прежде всего необходимо обратить внимание на саму идеологию объектно-ориентированного программирования (ООП). Этот экскурс вооружит читателя терминами, столь часто употребляемыми нами в последующих главах.

Объектно-ориентированное программирование

Вот уже как минимум десять лет программисты Delphi живут в терминах ООП и все это время теория ООП изменяется, подстраиваясь под нужды разработчиков программного обеспечения, меняя, однако, своей сути, в основе которой три краеугольных камня:

- инкапсуляция;
- наследование;
- полиморфизм.

Данные понятия неотделимы от двух других, принятых в Delphi и C#, — класса и объекта. Рассмотрим эти сущности, опираясь на синтаксис языка Delphi (отличия, характерные для C#, будут описаны в *главе 4*).

Классы и объекты

Классом в Delphi называется структура языка, которая может иметь в своем составе переменные, функции и процедуры. Переменные, в зависимости от предназначения, называются *полями* или *свойствами*. Процедуры и функции класса называются *методами*. Соответствующий классу тип будем называть *объектным типом* или *объектом*:

```

type
  TMyObject = class (TObject)
    MyField: Integer;
    function MyMethod: Integer;
end;

```

В этом примере описан класс `TMyObject`, содержащий поле `MyField` и метод `MyMethod`.

Существует также возможность внутри описания класса объявить не только свойство и/или метод, но и класс (так называемый вложенный тип данных):

```

type
  className = class [abstract | sealed] (ancestorType)
    memberList
    type
      nestedTypeDeclaration
    memberList
end;

```

Следующий пример (листинг 3.1) демонстрирует описание и доступ к полям и методам вложенного класса.

Листинг 3.1. Организация доступа к полям и методам вложенного класса

```

type
  TOuterClass = class
    strict private
    myField: Integer;
    public
    type
      TInnerClass = class
        public
        myInnerField: Integer;
        procedure innerProc;
      end;
    procedure outerProc;
  end;

```

Реализацию метода `innerProc` теперь можно осуществить с помощью следующей конструкции:

```

procedure TOuterClass.TInnerClass.innerProc;
begin
  ...
end;

```

Поля объекта аналогичны полям записи (record). Это данные, уникальные для каждого созданного в программе экземпляра класса. Описанный здесь класс `TMyObject` имеет одно поле — `MyField`.

Методы — это процедуры и функции, описанные внутри класса и предназначенные для операций над его полями. В состав класса входит указатель на специальную таблицу, где содержится вся информация, нужная для вызова методов. От обычных процедур и функций методы отличаются тем, что им при вызове передается указатель на тот объект, который их вызвал. Поэтому обрабатываться будут поля именно этого объекта. Внутри метода указатель на вызвавший его объект доступен под зарезервированным именем `Self`.

Понятие свойства будет подробно рассмотрено далее. Пока можно определить его как поле, доступное для чтения и записи не напрямую, а через соответствующие методы.

Классы могут быть описаны либо в секции интерфейса модуля, либо на верхнем уровне вложенности секции реализации. Не допускается описание классов "где попало", т. е. внутри процедур и других блоков кода.

Опережающее объявление классов разрешено, что иллюстрирует листинг 3.2.

Листинг 3.2. Пример опережающего объявления классов

```
type
  TFirstObject = class;
  TSecondObject = class(TObject)
    F1st : TFirstObject;
    ...
  end;
  TFirstObject = class(TObject)
    ...
  end;
```

Чтобы использовать класс в программе, нужно, как минимум, объявить переменную этого типа. Переменная объектного типа называется *экземпляром класса*, или *объектом*:

```
var
  AMyObject: TMyObject;
```

В приведенном фрагменте кода переменная `AMyObject` на самом деле является указателем, содержащим адрес объекта, а сам объект создается с помощью специального метода — *конструктора* объекта.

```
AMyObject := TMyObject.Create;
```

Созданный экземпляр класса уничтожается другим методом — *деструктором*:

```
AMyObject.Destroy;
```

Но в целях проверки указателя на область памяти, где размещен экземпляр класса на отличие указателя от Nil, рекомендуется вызвать метод `AMyObject.Free`, который автоматически вызовет метод уничтожения объекта — `Destroy`.

Поля, свойства и методы

Поля являются переменными, объявленными внутри класса. Они предназначены для хранения данных во время работы экземпляра класса (объекта). Ограничений на тип полей в классе не предусмотрено. В описании класса поля должны предшествовать методам и свойствам. Обычно поля служат для обеспечения выполнения операций внутри класса.

Разработка серьезного приложения в Delphi подразумевает использование нескольких классов, разделенных в соответствии с логикой разрабатываемого приложения. Несомненно, встает вопрос о возможности взаимодействия классов друг с другом или с программными элементами приложения. Для таких целей служат свойства, описываемые с помощью зарезервированного слова `property`.

Свойства представляют собой атрибуты, наиболее целостно описывающие объект. Например, обычная кнопка, размещенная на форме приложения, обладает такими свойствами, как цвет (`Color`), размеры (`Width`, `Height`), положение (`Left`, `Top`), надпись на кнопке (`Caption`) и т. п.

Так как свойство обеспечивает обмен данными с внешней средой, то для доступа к его значению необходимы специальные методы класса. Поэтому обычно свойство определяется тремя элементами: полем и двумя методами, которые осуществляют его чтение/запись:

```
type
  TAnObject = class(TObject)
    function GetColor: TSomeType;
    procedure SetColor(ANewValue: TSomeType);
    property AColor: TSomeType read GetColor write SetColor;
  end;
```

В данном примере доступ к значению свойства `AColor` осуществляется через вызовы методов `GetColor` и `SetColor`. Однако в обращении к этим методам в явном виде нет необходимости, достаточно написать

```
AnObject.AColor := AValue;
AVariable := AnObject.AColor;
```

и компилятор самостоятельно оттранслирует обращение к свойству в вызовы методов. Таким образом, внешне свойство выглядит в точности как обычное поле, но за всяким обращением к нему могут стоять нужные вам действия.

События

Все, что происходит с вашим приложением в процессе работы под управлением операционной системы Windows, — это последовательность наступления тех или иных событий. В Delphi предусмотрен механизм отслеживания и управления подобными событиями.

Событие — это свойство процедурного типа, предназначенное для создания пользовательской реакции на то или иное входное воздействие:

```
property OnMyEvent: TMyEvent read FOnMyEvent write FOnMyEvent;
```

Здесь `FOnMyEvent` — поле процедурного типа, содержащее адрес некоторого метода. Присвоить такому свойству значение — значит указать объекту адрес метода, который будет вызываться в момент наступления события. Такие методы называют обработчиками событий. Например, запись

```
Application.OnActivate := MyActivatingMethod;
```

означает, что при активизации объекта `Application` (так называется объект, соответствующий работающему приложению) будет вызван метод-обработчик `MyActivatingMethod`.

Внутри библиотеки времени выполнения (RTL) Delphi вызовы обработчиков событий находятся в методах, обрабатывающих сообщения Windows. Выполнив необходимые действия, этот метод проверяет, известен ли адрес обработчика, и, если это так, вызывает его:

```
if Assigned(FOnMyEvent) then FOnMyEvent(Self);
```

События имеют разные число и тип параметров в зависимости от происхождения и предназначения. Общим для всех является параметр `Sender`, он указывает на объект-источник события. Самый простой тип — `TNotifyEvent` — не имеет других параметров:

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

Тип метода, предназначенный для извещения о нажатии клавиши, предусматривает передачу программисту кода этой клавиши, о передвижении мыши — ее текущих координат и т. п. Все события в Delphi принято предварять префиксом `On`: `OnKeyPress`, `OnMouseMove` и т. п. Например, на форме `Form1` расположен компонент отображения статичного текста `TLabel1`. Тогда для обработки щелчка мышью (событие `OnClick`) будет создана заготовка метода `TForm1.Label1Click`:

```
procedure TForm1.Label1Click(Sender: TObject);  
begin  
...  
end;
```

Поскольку события являются свойствами объекта, их значения можно изменять в любой момент во время выполнения программы. Эта замечательная возможность называется делегированием. Можно в любой момент взять способы реакции на события у одного объекта и присвоить (делегируют) их другому:

```
Object1.OnMouseMove := Object2.OnMouseMove;
```

Принцип делегирования позволяет избежать трудоемкого процесса порождения новых дочерних классов для каждого специфического случая, заменяя его простой подстановкой процедур. При необходимости можно выбирать один из нескольких возможных вариантов обработчиков событий.

Инкапсуляция

Как правило, объект — это сложная конструкция, свойства и поведение составных частей которой находятся во взаимодействии. К примеру, если мы моделируем взлетающий самолет, то после набора им определенной скорости и отрыва от земли принципы управления им полностью изменяются. Поэтому в объекте "самолет" явно недостаточно просто увеличить значение поля "скорость" на несколько километров в час, такое изменение должно автоматически повлечь за собой целый ряд других.

При создании программных объектов подобные ситуации можно моделировать, связывая со свойствами необходимые методы. Понятие инкапсуляции соответствует этому механизму.

Классическое правило объектно-ориентированного программирования утверждает, что для обеспечения надежности нежелателен прямой доступ к полям объекта, чтение и обновление их содержимого должно выполняться вызовом соответствующих методов. Это правило и называется инкапсуляцией.

Наследование

Второй краеугольный камень ООП — наследование. Этот простой принцип означает, что если вы хотите создать новый класс, лишь немного отличающийся от старого, то совершенно нет необходимости в переписывании заново уже существующих полей и методов. Вы объявляете, что новый класс

```
TNewObject = class(TOldObject);
```

является *потомком* или *дочерним классом* старого класса `TObject`, называемого *предком* или *родительским классом*, и добавляете к нему новые поля, методы и свойства, — иными словами, все то, что нужно при переходе от общего к частному.

В Delphi все классы являются потомками базового класса `TObject`. Поэтому если вы создаете дочерний класс прямо от `TObject`, то в определении его можно не упоминать. Следующие два выражения одинаково верны:

```
TMyObject = class(TObject);  
TMyObject = class;
```

Первый вариант, хотя он и более длинный, предпочтительнее — для устранения возможных неоднозначностей.

Унаследованные от класса-предка поля и методы доступны в дочернем классе; если имеет место совпадение имен методов, то говорят, что они перекрываются.

Полиморфизм

Рассмотрим внимательно следующий пример (листинг 3.3). Пусть у нас имеется некое обобщенное поле для хранения данных — класс `TField` и три его потомка — для хранения строк, целых и вещественных чисел.

Листинг 3.3. Пример реализации принципа полиморфизма

```
type  
  TField = class  
    function GetData: string; virtual; abstract;  
  end;  
  TStringField = class(TField)  
    FData : string;  
    function GetData: string; override;  
  end;  
  TIntegerField = class(TField)  
    FData : Integer;  
    function GetData: string; override;  
  end;  
  TExtendedField = class(TField)  
    FData : Extended;  
    function GetData: string; override;  
  end;  
  ...
```