

ПРОГРАММИРОВАНИЕ В **DELPHI** 2005



ОБЗОР НОВШЕСТВ
DELPHI 2005 IDE

ОСОБЕННОСТИ
ПРОГРАММИРОВАНИЯ
НА ПЛАТФОРМЕ
WINDOWS 2000/XP/2003

СЕКРЕТЫ СОЗДАНИЯ
ПРИЛОЖЕНИЙ ADO.NET

МНОГОУРОВНЕВЫЕ
ПРИЛОЖЕНИЯ,
КОМПОНЕНТНОЕ
ПРОГРАММИРОВАНИЕ

ПРИМЕРЫ НАПИСАНИЯ
ГРАФИЧЕСКИХ
И МУЛЬТИМЕДИЙНЫХ
ПРИЛОЖЕНИЙ

PRO
ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

УДК 681.3.068+800.92Delphi2005
ББК 32.973.26-018.2
Б83

Боровский А. Н.

Б83 Программирование в Delphi 2005. — СПб.: БХВ-Петербург,
2005. — 448 с.: ил.

ISBN 5-94157-409-6

Книга посвящена разработке в Delphi 2005 различных типов приложений для Windows 2000/XP/2003. Описаны приемы программирования Win32 с учетом специфики Windows 2000/XP/2003, архитектура .NET и особенности создания приложений Windows Forms и VCL.Forms. Рассмотрены разработка приложений bdExpress, WebSnap и WebBroker, а также интернет-приложений с использованием компонентов Internet Direct 10. Уделено внимание многоуровневому компонентному программированию и бизнес-ориентированному моделированию с помощью компонентов ECO.

Описаны технологии ADO.NET, Borland Data Provider, ASP.NET и разработка приложений баз данных с помощью ADO.NET и ASP.NET. Рассмотрено создание мультимедиа-приложений с использованием расширенных возможностей графики GDI+, а также .NET и DirectX 9 SDK.

Для программистов

УДК 681.3.068+800.92Delphi2005
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольга Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Иины Тачиной</i>
Оформление обложки	<i>Игоря Цырульникова</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.03.05.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 36,12.

Тираж 4000 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 ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-409-6

© Боровский А. Н., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Оглавление

Предисловие	9
Глава 1. Новое в языке программирования Delphi	13
Новшества в Delphi Language.....	14
Новая модель идентификаторов.....	14
Пространства имен.....	15
Новые типы данных.....	16
Работа со строками.....	19
Новые конструкции языка.....	21
Новые конструкции Delphi Language в Delphi 2005.....	21
Цикл <i>for in do</i>	21
Встраиваемые процедуры и функции.....	22
Новые символы в идентификаторах.....	23
Многомерные динамические массивы.....	24
Новые элементы, введенные в Delphi 8.....	26
Новые определители видимости элементов классов.....	26
Декларация новых типов внутри классов.....	28
Декларация констант внутри классов.....	29
Новые типы классов.....	30
Перегрузка операторов в классах.....	31
Перегрузка перегруженных операторов.....	36
Помощники классов.....	37
Атрибуты классов.....	39
Вызов функций Windows API из среды .NET.....	41
Вызов функций из разделяемых библиотек.....	43
Директивы компилятора для .NET и ключевое слово <i>unsafe</i>	44
Перенос программ Win32 на платформу .NET.....	45
Проблема указателей.....	45
Глава 2. Интегрированная среда разработки Delphi 2005	49
Что нового по сравнению с Delphi 7?.....	49
Стартовая страница.....	49

Главное окно.....	50
Палитра инструментов.....	51
Инспектор объектов.....	52
Окно менеджера проекта.....	52
Окно редактора исходных текстов.....	52
Менеджер установленных компонентов.....	53
Утилита Borland Reflection.....	54
Интеграция Delphi IDE и средств контроля версий.....	56
Мастер Satellite Assembly Wizard.....	58
Что нового по сравнению с Delphi 8?.....	59
Особенности работы компилятора и отладчика.....	61
Контроль изменений исходных текстов.....	62
Структура справочной системы Delphi 2005.....	64
Глава 3. Программирование на платформе Win32.....	67
Работа со строками.....	68
Обработка сообщений.....	70
Взаимодействие между процессами.....	78
Сообщение <i>WM_COPYDATA</i>	79
Именованные каналы.....	81
Файлы, отображаемые в память.....	85
Потоки и блокирующие функции.....	93
Дочерние процессы и неименованные каналы.....	97
Службы Windows 2000+.....	102
Инструмент исследователя.....	107
Глава 4. Разработка приложений баз с помощью компонентов VCL и VCL.NET.....	109
Утилита Data Explorer.....	110
Приложения dbExpress.....	111
Улучшение процедуры авторизации.....	115
Компонент <i>TSQLDataSet</i>	118
Компонент <i>TClientDataSet</i>	119
Интерактивные приложения баз данных.....	120
Низкоуровневое редактирование записей.....	122
Автоматическая генерация индексов.....	124
Преобразование записей.....	125
Работа с базами данных InterBase.....	127
Работа с BDE.....	129
Глава 5. Интернет-программирование.....	131
Замечания по поводу Internet Direct.....	131
Исключения в Indy.....	131
FTP-клиент.....	132
Отладчик Web App Debugger.....	135
Технология WebBroker.....	137
Основа объектной модели приложений WebBroker.....	137
Компоненты-генераторы контента.....	140

Обработчики событий <i>OnBeforeDispatch</i> и <i>OnAfterDispatch</i>	141
Простейшее приложение WebBroker	141
Технология WebSnap	148
Концепция Adapter Actions	151
Программа просмотра изображений	153
Web-службы	156
Глава 6. Введение в язык C#	163
Типы данных	166
Указатели и небезопасный код	167
Параметры-переменные	168
Динамические массивы	168
Конструкторы классов	169
Перекрытие методов	169
Оператор <i>foreach</i>	170
Служба VabelCode	171
Глава 7. Программирование на платформе .NET	173
Что такое .NET?	173
Общая среда выполнения	174
Общий промежуточный язык	175
Общая система типов	175
"Песочница" .NET	176
Общая библиотека классов .NET	176
Служба обращения к базовой платформе	177
Расширяемые метаданные	177
Атрибуты	177
Исполняемые файлы .NET	177
Сборки .NET	178
Создание сборки DLL	181
Динамическая загрузка сборок-библиотек	184
Добавление подписи в exe-файл	186
Управление памятью	187
Сборка мусора	188
Управление памятью и программирование в Delphi для .NET	189
Конструкторы объектов	189
Метод <i>Finalize</i>	189
Метод <i>Dispose</i>	189
Что нельзя делать в .NET	194
Ввод/вывод	194
Потоки ввода/вывода	194
Изолированное хранение данных	197
Мониторинг изменений файловой системы	201
Утилита ILDASM	203
Потоки .NET	205
Синхронизация потоков	211
Использование эnumераторов	215

Несколько полезных рецептов.....	217
Определение расположения специальных папок Windows.....	217
Просмотр переменных окружения	218
Глава 8. Приложения VCL Forms	221
Формы VCL Forms	221
Классы .NET в приложении VCL Forms	223
Объекты автоматизации.....	227
Глава 9. Приложения Windows Forms	231
Метод <i>OnPaint</i> и событие <i>Paint</i>	237
Фоновый рисунок для формы приложения	239
События .NET и делегаты.....	242
Обработка сообщений Windows.....	246
Расположение компонентов в форме.....	247
Сохранение ресурсов в приложении	247
Ресурсы и интернационализация.....	249
Компонент <i>ToolTip</i>	251
Элементы управления Windows Forms.....	251
Дополнительные возможности GDI+.....	253
Окно непрямоугольной формы.....	253
Использование компонентов ActiveX в приложениях Windows Forms	257
Классы <i>WebRequest</i> и <i>WebResponse</i>	260
Единицы измерения.....	264
Печать в приложениях Windows Forms.....	265
Выбор принтера и вывод данных	265
Компонент <i>PrintPreviewControl</i>	268
Диалоговые окна печати.....	269
Механизм Drag and Drop.....	270
Глава 10. Разработка приложений баз данных с помощью ADO.NET.....	275
Знакомство с Borland Data Provider.....	275
Компонент <i>BdpConnection</i>	276
Компонент <i>BdpDataAdapter</i>	277
Компонент <i>BdpCommand</i>	280
Знакомство с компонентами ADO.NET	283
Интерфейсы ADO.NET.....	283
Интерфейс <i>IDbConnection</i>	283
Интерфейс <i>IDbCommand</i>	283
Интерфейс <i>IDataReader</i>	284
Интерфейс <i>IDataAdapter</i>	284
Программа просмотра данных	285
Модификация данных	289
Визуальное программирование приложений ADO.NET.....	295
Компонент <i>DataView</i>	296
Глава 11. Моделирование приложений с помощью ESO	299
Создаем ESO-приложение	299

Глава 12. Разработка приложений ASP.NET.....	307
Введение в ASP.NET.....	307
Преимущества ASP.NET.....	308
Домены приложений.....	308
Разработка простейшего приложения ASP.NET в Delphi 2005	308
Анатомия приложения ASP.NET, созданного в Delphi 2005.....	312
Страницы со встроенным кодом.....	320
Классы <i>HttpRequest</i> и <i>HttpResponse</i>	322
Свойства класса <i>HttpRequest</i>	323
Методы и свойства класса <i>HttpResponse</i>	323
Сохранение состояния в перерывах между транзакциями	324
Проблема сохранения состояния.....	324
Пример сохранения состояния: программа-калькулятор.....	325
Сохранение данных в масштабах приложения	329
Сохранение данных с помощью сессий	332
Использование технологии AutoPostBack.....	336
Взаимодействие с элементами управления HTML.....	339
Как это работает?	340
Загрузка файлов на сервер.....	341
Создание Web-сервиса электронной почты	343
Компоненты-валидаторы.....	345
Компонент <i>RegularExpressionValidator</i>	345
Регулярные выражения в ASP.NET	345
Компонент <i>CustomValidator</i>	348
Связывание данных.....	350
Глава 13. Приложения ASP.NET и базы данных	357
Механизм связывания данных и базы данных	357
Компоненты <i>DataList</i> и <i>DataGrid</i>	359
Шаблоны	359
Использование в шаблонах элементов управления ASP.NET.....	363
Компонент <i>DataGrid</i>	370
Компоненты <i>DB Web</i>	373
Глава 14. Web-службы ASP.NET	375
Создание сервера и клиента Web-служб в Delphi 2005	375
Разработка клиента для сторонней Web-службы.....	379
Разработка собственного сервера и клиента Web-служб	383
Сохранение состояния на сервере Web-служб.....	387
Глава 15. Разработка многоуровневых приложений и компонентов.....	389
Трехуровневая модель приложения.....	389
Компонентное программирование.....	390
Многоуровневое приложение ASP.NET	406
Глава 16. Графика и мультимедиа в Delphi 2005.....	413
Работа с изображениями	413
Просмотр изображений	413

Вращение изображений	415
Отсечение изображений	417
Другие трансформации изображений	422
Наклон изображений	422
Создание полупрозрачных изображений	424
Преобразование цвета	426
Класс <i>ColorMatrix</i>	426
Вывод текста с использованием узора	429
Преобразование форматов графических файлов	430
Воспроизведение анимации	431
Воспроизведение видеоклипов	433
Воспроизведение wav-файлов с помощью DirectX	437
Заключение	439
Приложение. Описание компакт-диска	441
Литература и интернет-источники	442
Предметный указатель	443



ГЛАВА 4

Разработка приложений баз с помощью компонентов VCL и VCL.NET

Материал этой главы будет полезен как тем программистам, которые хотят разрабатывать приложения баз данных на платформе Win32, так и тем, кто хочет использовать для разработки таких приложений компоненты VCL.NET. Компоненты VCL.NET были введены в Delphi 8 (в которой отсутствовала среда разработки на платформе Win32) именно для того, чтобы упростить перенос приложений баз данных, написанных с применением компонентов VCL в среду .NET. Для краткости в этой главе мы будем говорить "VCL", подразумевая под этим компоненты VCL и VCL.NET. Глава 8 будет посвящена особенностям компонентов VCL.NET, но повторять в ней то, что сказано здесь, мы не будем.

Набор компонентов VCL содержит наборы компонентов dbExpress, InterBase и BDE. В таком порядке мы и рассмотрим эти инструменты. Набор компонентов InterBase, как следует из названия, предназначен исключительно для взаимодействия с СУБД InterBase. Две другие технологии поддерживают несколько наиболее популярных СУБД, включая и InterBase. Но прежде нам понадобится создать тестовую базу данных (БД), на которой мы будем опробовать наши примеры. Для этого мы воспользуемся программой Microsoft SQL Server 2000 или ее бесплатным аналогом MSDE (далее мы будем говорить просто — SQL Server).

Примечание

Поскольку эта и другие главы, посвященные работе с базами данных, ориентированы на SQL-базы данных, для понимания приводимых далее примеров вам понадобятся минимальные знания языка SQL. Описание SQL выходит за рамки этой книги. Желающим изучить SQL и MS SQL Server можно порекомендовать книгу: Мамаев Е. SQL Server 2000 в подлиннике. — СПб.: БХВ-Петербург, 2001.

На SQL Server необходимо создать базу данных DelphiDemo, в которой должна существовать таблица PriceList, содержащая информацию о перечне

услуг, предлагаемых некоторой дизайнерской фирмой и расценках на них. На сервере должна существовать учетная запись с именем DelphiUser и паролем letmein, причем пользователь DelphiUser должен иметь полные права доступа к БД DelphiDemo и быть владельцем таблицы PriceList. На компакт-диске вы можете найти файл DelphiDemo.sql, который представляет собой скрипт, автоматизирующий создание описанной базы данных и таблицы в MS SQL Server 2000.

Примечание

Вы, конечно, можете создать свою базу данных, однако в этом случае вам придется вносить соответствующие изменения и во все примеры работы с базами данных, приведенные далее.

Утилита Data Explorer

Утилита Data Explorer, входящая в состав дистрибутива Delphi, может быть полезна при отладке программ, работающих с базами данных. Часто бывает так, что программа не может установить связь с базой данных, особенно если сервер баз данных расположен на другом компьютере. При этом не всегда можно ответить на вопрос, почему не устанавливается связь — из-за ошибок в программе или по каким-то другим причинам. Утилита Data Explorer помогает разрешить этот вопрос, а также выполнить некоторые другие функции, полезные при работе с базами данных. Окно утилиты (рис. 4.1) разделено на две части.

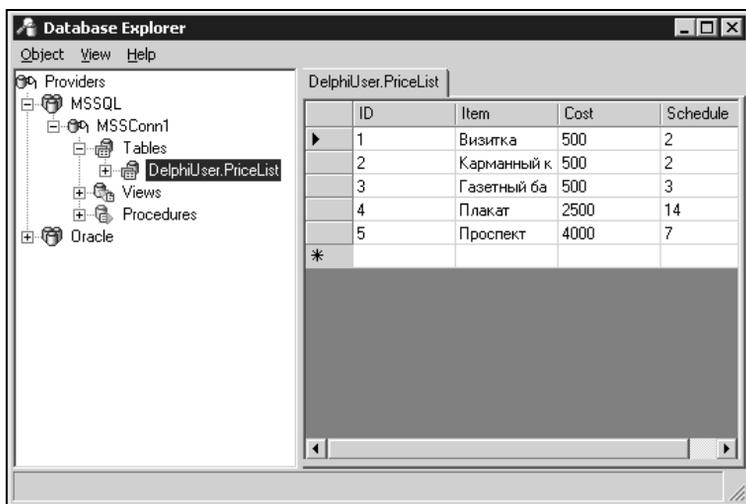


Рис. 4.1. Утилита Data Explorer

В левой части окна находится иерархический список всех соединений, соответствующий перечню соединений dbExpress. С помощью контекстных меню элементов этого списка можно настраивать параметры выбранного соединения на подключение к конкретному серверу баз данных. Далее можно протестировать само подключение. Если подключение выполнено успешно, иерархический список раскрывается, и в нем отображаются объекты подключенной базы данных. С помощью этого списка можно просматривать, например, содержимое доступных таблиц баз данных (оно отображается в правой части окна).

С помощью утилиты Data Explorer можно не только просматривать, но и редактировать таблицы, добавляя новые записи или изменяя значения полей. Для этого нужно заполнить очередную запись, а затем выбрать команду **Update** контекстного меню (таким образом вы можете заполнить таблицу PriceList для первоначальной работы).

Приложения dbExpress

Теперь мы можем приступить к написанию простейшего приложения для работы с базой данных. Рассмотрим общую структуру приложения баз данных Delphi, использующего dbExpress (рис. 4.2).

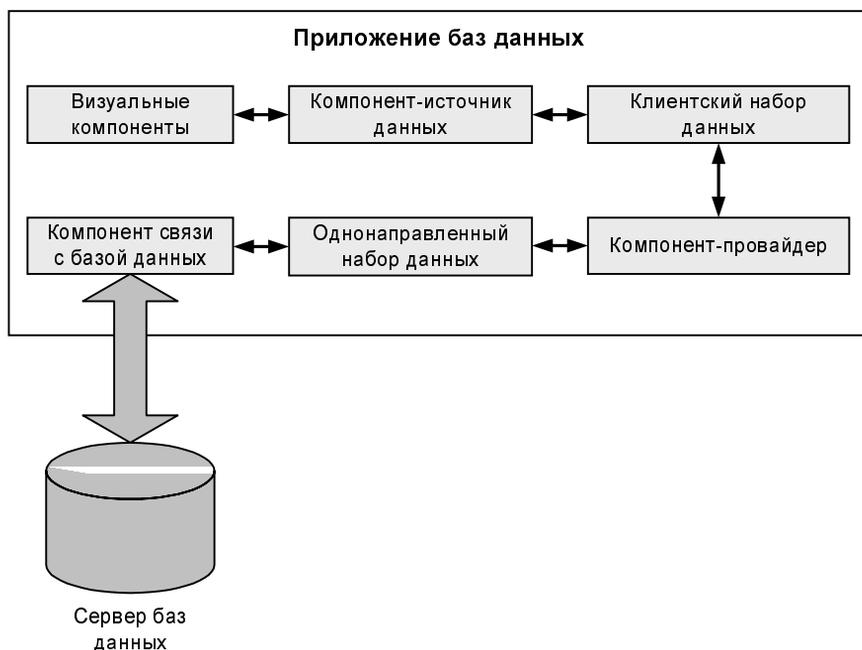


Рис. 4.2. Структура приложения dbExpress

Основной технологии dbExpress являются драйверы доступа к базам данных, которые, как правило, реализованы в виде разделяемых библиотек. Благодаря концепции независимых драйверов, технология dbExpress является расширяемой. Ряд сторонних разработчиков предлагает драйверы dbExpress для СУБД, которые не поддерживаются набором драйверов от Borland. Некоторые разработчики предлагают драйверы-аналоги драйверов Borland, обладающие большей функциональностью. Установив новый драйвер dbExpress можно расширить возможности программирования приложений баз данных.

Приложение для работы с dbExpress данных состоит из трех частей: компонентов интерфейса пользователя, компонентов доступа к данным и компонента, устанавливающего связь с базой данных. Независимость этих трех групп компонентов друг от друга обеспечивает гибкость приложений dbExpress. Например, для того чтобы перейти от использования одного сервера баз данных к другому (или даже сменить технологию доступа к базам данных), достаточно заменить (или перенастроить) компонент связи с базой данных. Никаких других изменений в приложение вносить не придется.

Роль компонента связи с базой данных в приложениях dbExpress выполняет компонент `TSQLConnection`, расположенный на странице **dbExpress** палитры инструментов.

К компонентам доступа к данным относятся `TSQLDataSet` (однонаправленный набор данных), расположенные на странице **Data Access** компоненты `TClientDataSet` (клиентский набор данных), `TDataSetProvider` (компонент-провайдер) и `TDataSource` (компонент-источник данных).

Компоненты пользовательского интерфейса, задачей которых является отображение информации, хранимой в базе данных, и управление этой информацией, размещены на странице **Data Controls** палитры инструментов.

Компонент связи с базой данных хранит информацию об используемом сервере баз данных, а также сведения, необходимые для подключения к серверу. Однонаправленный набор данных хранит базовый SQL-запрос, направляемый серверу баз данных. В ответ на этот запрос сервер передает данные, которые компонент, реализующий *однонаправленный набор данных*, преобразует в пакет и передает компоненту-провайдеру. Компонент-провайдер является связующим звеном между однонаправленным и клиентским наборами данных.

Основное различие между однонаправленным и клиентским набором данных заключается в том, что клиентский набор данных хранит в памяти массив записей, полученных в результате запроса к серверу баз данных. Компоненты, реализующие клиентские наборы данных, предоставляют произвольный доступ к записям, хранимым в массиве. При этом возможно не только считывание записей из массива, но и их изменение и добавление, поэтому набор и называется *двунаправленным*. Однонаправленные наборы

данных могут работать в каждый момент времени лишь с одной записью (текущей), при этом произвольный доступ к записям, поддерживаемый клиентскими наборами данных, невозможен.

Если пользователь вносит изменения в базу данных с помощью графического элемента управления, измененные записи сначала передаются клиентскому набору данных, который передает их компоненту-провайдеру, а тот, в свою очередь передает их с помощью соответствующих методов однонаправленного набора данных компоненту, осуществляющему связь с базой данных.

Напишем приложение для просмотра таблицы PriceList. Разработка нашего приложения начинается с настройки компонента TSQLConnection:

1. Разместите этот компонент в форме приложения VCL Forms (будет добавлен объект SQLConnection1) и дважды щелкните мышью по пиктограмме компонента в окне формы.
2. В открывшемся окне (рис. 4.3) в списке **Connection Name** выберите пункт **MSSQLConnection**. Таблица **Connection Settings** позволяет настроить свойства соединения. Самые важные поля в этой таблице — **HostName** (имя компьютера, на котором размещен сервер баз данных), **DataBase** (имя базы данных) **User_Name** и **Password** (соответственно, имя учетной записи и пароль). Поле **OS Authentication** позволяет выбрать тип авторизации на сервере баз данных — средствами операционной системы или средствами сервера.

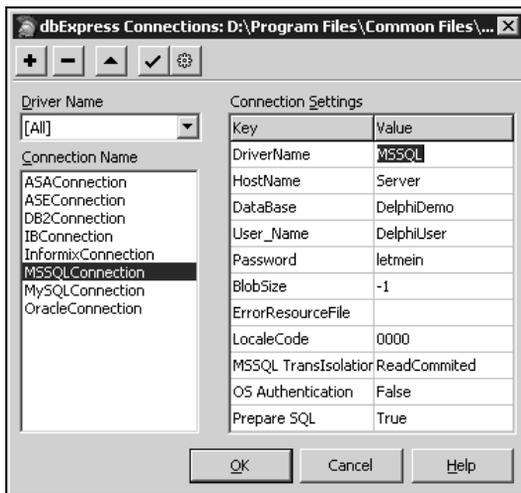


Рис. 4.3. Окно dbExpress Connections

Настроив соединение и закрыв окно **dbExpress Connections**, вы можете проверить, удастся ли программе установить соединение с сервером. Для этого

в инспекторе объектов присвойте свойству `Connected` объекта `SQLConnection1` значение `True`. Если при этом не было выдано сообщения об ошибке, значит, связь с сервером баз данных установлена.

Теперь можно приступить к настройке однонаправленного набора данных. Размещаем в форме приложения компонент `TSQLDataSet` (будет добавлен объект `SQLDataSet1`). Прежде всего, этот объект необходимо связать с объектом `SQLConnection1` (это делается в инспекторе объектов с помощью свойства `SQLConnection` объекта `SQLDataSet1`). Выше отмечалось, что компонент `TSQLDataSet` хранит SQL-запрос к базе данных, который находится в свойстве `CommandText` объекта `SQLConnection1`. Назначьте этому свойству запрос:

```
select * from [DelphiUser].[PriceList]
```

Наша следующая задача — создать и настроить компонент-провайдер. Размещаем в форме приложения компонент `TDataSetProvider` (будет добавлен объект `DataSetProvider1`). Свойству `DataSet` этого объекта следует назначить ссылку на `SQLDataSet1`, а свойству `Enabled` — присвоить значение `True`. Теперь добавляем в проект компонент `TClientDataSet` (объект `ClientDataSet1`). Посредством свойства `ProviderName` связываем этот объект с объектом `DataSetProvider1`, а свойству `Active` присваиваем значение `True`.

Для того чтобы наше приложение баз данных могло использовать визуальные компоненты пользовательского интерфейса, связанные с базой данных (*data-aware components*), мы должны добавить в цепочку компонентов компонент `TDataSource` (объект `DataSource1`). Свойству `DataSet` этого объекта следует присвоить ссылку на объект `ClientDataSet1`. Компонент `TDataSource` выполняет роль "крючка", к которому прикрепляются все компоненты пользовательского интерфейса, связанные с базой данных. У каждого такого компонента есть свойство `DataSource`, которое должно ссылаться на объект `TDataSource`.

Нам осталось только добавить компонент для визуального отображения содержимого таблицы. В качестве такового мы используем компонент `TDBGrid`, расположенный на странице **Data Controls** палитры инструментов. Размещая этот компонент в форме, необходимо связать его с объектом `DataSource1`, как было описано выше.

Работающее приложение показано на рис. 4.4.

Если вы знакомы с языком SQL, то можете оценить, как много работы компоненты `bdExpress` выполнили за вас. Для правильного отображения данных объекту `DBGrid` требуется информация о типах полей таблицы, и эти сведения были получены компонентом автоматически. Автоматизация многих рутинных задач, возникающих при разработке приложений баз данных, является причиной того, что Delphi считается одним из наиболее удобных средств разработки таких приложений.



ID	Вид работ	Стоимость	Срок выполнения
1	визитка	500	от 2 дней
2	карманный календарь	500	от 2 дней
3	газетная реклама	500	от 3 дней
4	листочка 1/3 формата А4	500	7 дней
5	листочка формата А3	1000	7 дней
6	буклет формата А3	2500	7 дней
7	плакат формата А2	1500	7 дней
8	проспект	4500	7 дней
9	билборд 3Х6 м	3000	7 дней
10	флаг	100	от 2 дней

Рис. 4.4. Приложение dbExpress

Улучшение процедуры авторизации

Работая с созданным нами приложением, вы наверняка заметили, что всякий раз при запуске приложения у вас запрашиваются имя пользователя и пароль для подключения к серверу баз данных. Но ведь эти данные уже хранятся в настройках соединения. Нельзя ли сделать так, чтобы программа использовала эту информацию из настроек? Можно. Для этого необходимо присвоить свойству `LoginPrompt` объекта `SQLConnection1` значение `False`.

Вообще говоря, хранение имени пользователя и пароля в настройках соединения является небезопасным и неэффективным. Определив настройки подключения в одном проекте, вы можете использовать их и в других проектах. Это происходит потому, что Delphi сохраняет данные о настройках соединений dbExpress в файле, который является общедоступным (это файл `dbxconnections.ini`, хранящийся в одном из каталогов, созданных Delphi в процессе установки). Ниже приводится фрагмент данного файла, соответствующий соединению с SQL Server.

```
[MSSQLConnection]
DriverName=MSSQL
HostName=Server
DataBase=DelphiDemo
User_Name=DelphiUser
Password=letmein
BlobSize=-1
ErrorResourceFile=
LocaleCode=0000
MSSQL TransIsolation=ReadCommitted
OS Authentication=False
```

Как видим, и имя пользователя, и пароль хранятся в этом файле открытым текстом. Кроме того, записанные в настройки имя пользователя и пароль

попадут в двоичный текст скомпилированной программы. Это, во-первых, небезопасно, а во-вторых, негибко, ведь имя пользователя и тем более пароль могут быть изменены администратором сервера баз данных.

Автоматическая авторизация на сервере баз данных может быть удобна в процессе разработки и отладки приложения. При подготовке окончательного релиза поля `User_Name` и `Password` в настройках соединения следует оставить пустыми, а свойству `LoginPrompt` объекта `SQLConnection1` присвоить значение `True`. В этом случае программа всегда будет запрашивать у пользователя имя и пароль, посылая эти данные непосредственно на сервер.

Вы можете русифицировать стандартное диалоговое окно авторизации. Для этого необходимо отредактировать форму этого диалогового окна. Для VCL-приложений файлы соответствующего модуля имеют имена `DBLogDlg.pas` и `DBLogDlg.dfm` и находятся в каталоге `<DelphiRoot>\source\Win32\db`, а для приложений VCL.NET файлы модуля называются `Borland.Vcl.DBLogDlg.pas` и `Borland.Vcl.DBLogDlg.nfm` и расположены в каталоге `<DelphiRoot>\source\dotNet\db`. Скопируйте эти файлы в каталог `<DelphiRoot>\lib`. Откройте соответствующие модули в Delphi IDE и отредактируйте надписи. Удалите из этого каталога файлы `DBLogDlg.dcu` и `Borland.Vcl.DBLogDlg.dcuil`. Теперь, при следующем запуске программы, у вас появится окно русифицированного диалога авторизации (рис. 4.5).

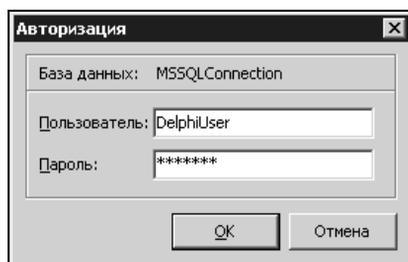


Рис. 4.5. Русифицированный диалог авторизации

Мы можем и дальше усовершенствовать процедуру авторизации. По умолчанию, если пользователь вводит неправильный пароль, приложение генерирует исключение и его работа завершается. То же самое происходит, если пользователь щелкает кнопку **Отмена**. Такое поведение программы нельзя назвать удобным. Следующая процедура-обработчик события `OnShow` главной формы (листинг 4.1) выводит диалоговое окно авторизации до тех пор, пока пользователь либо не введет правильные реквизиты, либо не нажмет кнопку **Отмена**. В последнем случае приложение завершается, не создавая никаких исключений.

Листинг 4.1. Улучшенная процедура авторизации

```
procedure TForm1.FormShow(Sender: TObject);
var
    User, Password : String;
    LogDlg : TLoginDialog;
begin
    if SQLConnection1.Connected then SQLConnection1.Close;
    LogDlg := TLoginDialog.Create(nil);
    while LogDlg.ShowModal <> mrCancel do
    begin
        SQLConnection1.Params.Values['User_Name'] := LogDlg.UserName.Text;
        SQLConnection1.Params.Values['Password'] := LogDlg.Password.Text;
        try
            SQLConnection1.Connected := True;
            LogDlg.Free;
            Exit;
        except
            end;
    end;
    LogDlg.Free;
    Application.Terminate;
end;
```

В этой процедуре мы используем класс `TLoginDialog`, определенный в модуле `DBLogDlg`, который следует включить в раздел `uses`. Мы заполняем свойство `Params` объекта `SQLConnection1` значениями имени и пароля, введенными пользователем. Свойство `Params` имеет тип `TStringList` и хранит настройки соединения с базой данных. Настройки хранятся в виде списка строк формата

имя_параметра=<значение>

где имена параметров и значения соответствуют таблице **Connection Settings** редактора соединений.

Далее мы пытаемся соединиться с базой данных, присваивая свойству `SQLConnection1.Connected` значение `True`. Делается это в блоке перехвата исключений, т. к. если программе не удастся установить соединение с базой данных (например, если пользователь ввел неправильный пароль), будет вызвано исключение. В случае возникновения исключения диалоговое окно авторизации выводится снова.