

Петр Дарахвелидзе
Евгений Марков

Разработка

Web-служб
средствами

Delphi



+Дискета

- Архитектура распределенных приложений баз данных
- Использование XML и протокола SOAP
- Распределенные приложения и Web-службы
- Криптография и безопасность данных



МАСТЕР ПРОГРАММ

**Петр Дарахвелидзе
Евгений Марков**

**Разработка
Web-служб
средствами Delphi**

Санкт-Петербург
«БХВ-Петербург»

2003

УДК 681.3.06
ББК 32.973.26-018
Д20

Дарахвелидзе П. Г., Марков Е. П.

Д20 Разработка Web-служб средствами Delphi. — СПб.: БХВ-Петербург, 2003. — 672 с.: ил.

ISBN 5-94157-142-9

В книге рассматриваются принципы функционирования Web-служб и инструменты для их создания и отладки, а также способы обеспечения связи и устойчивой работы распределенных приложений с помощью протокола SOAP. Подробно обсуждаются базовые технологии взаимодействия объектов, созданных в различных средах, способы разработки приложений баз данных, современные методы создания Internet-приложений, в том числе с применением языка XML. Большое количество примеров, приведенных в книге, и прилагаемая дискета помогут легче и быстрее освоить материал книги.

Для программистов

УДК 681.3.06
ББК 32.973.26-018

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зав. редакцией	<i>Анна Кузьмина</i>
Редактор	<i>Татьяна Коротяева</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Оформление серии	<i>Via Design</i>
Дизайн обложки	<i>Игоря Цырульниковца</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.10.02.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 54,18.

Тираж 4000 экз. Заказ №

"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953 Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

Содержание

ВВЕДЕНИЕ.....	1
COM и COM+.....	2
DataSnap.....	2
XML.....	3
SOAP.....	3
WebSnap и WebBroker.....	4
Криптография и безопасность данных.....	4
Часть I. ПРИЛОЖЕНИЯ COM и COM+.....	5
Глава 1. Механизмы COM в Delphi.....	7
1.1. Базовые понятия.....	8
1.1.1. Объект.....	11
1.1.2. Интерфейс.....	12
1.1.3. Интерфейс <i>IUnknown</i>	14
1.1.4. Сервер.....	14
1.1.5. Библиотека COM.....	15
1.1.6. Фабрика класса.....	17
1.1.7. Библиотека типов.....	17
1.2. Объекты COM в Delphi.....	18
1.2.1. Класс <i>TComObject</i>	19
1.2.2. Класс <i>TTypedComObject</i>	21
1.2.3. Интерфейс <i>IUnknown</i> в Delphi.....	21
1.2.4. Тип глобального идентификатора.....	22
1.2.5. Класс <i>TInterfacedObject</i>	22
1.3. Фабрика класса в Delphi.....	23
1.3.1. Класс <i>TComObjectFactory</i>	23
1.3.2. Класс <i>TTypedComObjectFactory</i>	26
1.3.3. Класс <i>TComClassManager</i>	26

1.4. Сервер COM в Delphi.....	26
1.4.1. Класс <i>TComServer</i>	27
1.5. Библиотека типов в Delphi	28
1.6. Простой объект COM в составе внутреннего сервера.....	31
1.6.1. Создание объекта.....	31
1.7. Создание методов интерфейса	38
1.7.1. Регистрация внутреннего сервера.....	45
1.7.2. Использование интерфейсов внутреннего сервера COM.....	45
Резюме.....	48
Глава 2. Технология Автоматизация	49
2.1. Базовые понятия технологии Автоматизация	50
2.1.1. Интерфейсы Автоматизации	50
Интерфейс <i>IDispatch</i> и диспінтерфейсы	50
Дуальные интерфейсы	52
2.1.2. Библиотека типов.....	52
2.1.3. Маршалинг интерфейсов Автоматизации	52
2.1.4. Объект Автоматизации.....	53
2.1.5. Сервер Автоматизации.....	53
2.1.6. Контроллер Автоматизации	53
2.2. Реализация Автоматизации в Delphi	53
2.2.1. Интерфейсы автоматизации.....	54
Интерфейс <i>IDispatch</i>	54
2.2.2. Интерфейсы диспетчеризации.....	55
2.2.3. Дуальные интерфейсы	56
2.2.4. Объект Автоматизации.....	56
Класс <i>TAutoObject</i>	57
Обработка событий объекта Автоматизации	59
Создание методов — аналогов событий.....	62
Создание класса-оболочки	63
Инициализация объекта Автоматизации	63
2.2.5. Фабрика класса.....	65
Класс <i>TAutoIntfObject</i>	66
2.2.6. Сервер Автоматизации.....	66
2.2.7. Контроллер Автоматизации	67
2.3. Пример приложения Автоматизации	69
2.3.1. Сервер Автоматизации.....	69
2.3.2. Контроллер Автоматизации	72
Резюме.....	74
Глава 3. Компоненты ActiveX	75
3.1. Как работают элементы управления ActiveX.....	76
3.1.1. Контейнеры и регистрация элементов управления ActiveX.....	79

3.1.2. Предоставление методов.....	80
3.1.3. События.....	80
3.1.4. Свойства.....	81
3.1.5. Страницы свойств.....	81
3.1.6. Лицензирование.....	81
3.2. Реализация компонентов ActiveX в Delphi.....	82
3.2.1. Класс компонента ActiveX.....	83
3.2.2. Фабрика класса компонента ActiveX.....	84
3.2.3. Среда разработки Delphi как контейнер ActiveX.....	85
3.2.4. Регистрация компонентов ActiveX.....	85
3.3. Использование готовых компонентов ActiveX.....	85
3.3.1. Инсталляция готовых элементов управления ActiveX.....	86
3.3.2. Деинсталляция готовых элементов управления ActiveX.....	87
3.3.3. Пример инсталляции элемента управления <i>TWebBrowser</i>	87
3.4. Разработка собственных компонентов ActiveX.....	91
3.4.1. Превращение компонентов Delphi в компоненты ActiveX.....	92
3.4.2. Преобразование форм в формы ActiveX.....	93
Резюме.....	99

Глава 4. Технология COM+ (Microsoft Transaction Server) 100

4.1. Как работает MTS.....	101
4.1.1. Объект MTS.....	102
4.1.2. Транзакции.....	103
4.1.3. Контекст объекта MTS.....	104
4.1.4. Безопасность данных в MTS.....	105
Декларативная защита данных.....	105
Программная защита данных.....	106
4.1.5. Ресурсы.....	106
Активизация Just-in-time.....	106
Пулинг ресурсов.....	107
Освобождение ресурсов.....	107
Пулинг объектов.....	108
4.2. Создание приложений MTS в Delphi.....	109
4.2.1. Объекты транзакций MTS.....	109
Класс <i>TM TSAutoObject</i>	109
Создание объекта транзакции.....	111
Типы потоковой модели.....	113
Поведение объектов MTS в транзакциях.....	113
4.2.2. Удаленный модуль данных MTS.....	114
4.2.3. Распределители ресурсов.....	115
4.3. Тестирование и установка MTS компонентов.....	118
4.4. Оптимизация работы с MTS.....	120
4.4.1. Блокировка транзакций.....	120

4.4.2. Действия MTS.....	121
4.5. Пример простого объекта транзакции	121
4.5.1. Создание сервера.....	121
4.5.2. Создание клиента.....	123
4.5.3. Пример создания клиента и сервера в случае распределенной транзакции.....	124
Создание серверов.....	124
Разработка объекта распределенной транзакции	125
Создание клиента	127
Резюме.....	127

Часть II. ТЕХНОЛОГИИ ДОСТУПА К ДАННЫМ 129

Глава 5. Архитектура приложений баз данных 131

5.1. Общая структура приложения баз данных	133
5.1.1. Как работает приложение баз данных	133
5.1.2. Модуль данных	136
5.1.3. Подключение данных	137
5.1.4. Настройка компонента <i>TDataSource</i>	138
5.1.5. Отображение данных	140
5.2. Набор данных.....	141
5.2.1. Абстрактный набор данных.....	141
Навигация по набору данных.....	142
Поля набора данных.....	143
Редактирование набора данных	144
Обработка событий набора данных.....	145
5.2.2. Стандартные компоненты	146
Компонент таблицы	146
Компонент запроса	149
Компонент хранимой процедуры	151
5.2.3. Состояния набора данных.....	152
5.3. Индексы	155
5.3.1. Механизм подключения индексов	156
5.3.2. Список описаний индексов	156
5.3.3. Описание индекса	157
5.4. Параметры запросов и хранимых процедур	159
5.4.1. Класс <i>TParams</i>	162
5.4.2. Класс <i>TParam</i>	163
5.5. Механизмы управления данными.....	165
5.5.1. Связанные таблицы.....	165
Отношение "один-ко-многим"	165
Отношение "многие-ко-многим"	166
5.6. Поиск данных.....	166
5.7. Фильтры	167

5.8. Быстрый переход к помеченным записям.....	169
5.9. Поля.....	170
5.10. Объекты полей	171
5.10.1. Статические и динамические поля.....	172
5.10.2. Класс <i>TField</i>	174
5.10.3. Виды полей.....	178
Поля синхронного просмотра.....	179
Вычисляемые поля	181
Внутренние вычисляемые поля	182
Агрегатные поля.....	183
Объектные поля	183
5.10.4. Типы данных	184
Резюме.....	186
Глава 6. Технология dbExpress	187
6.1. Доступ к данным dbExpress	188
6.2. Драйверы доступа к данным.....	190
6.3. Соединение с сервером баз данных.....	190
6.4. Управление наборами данных.....	195
6.5. Транзакции	197
6.6. Использование компонентов наборов данных.....	198
6.6.1. Класс <i>TCustomSQLDataSet</i>	199
6.6.2. Компонент <i>TSQLDataSet</i>	201
6.6.3. Компонент <i>TSQLTable</i>	202
6.6.4. Компонент <i>TSQLQuery</i>	203
6.6.5. Компонент <i>TSQLStoredProc</i>	204
6.7. Компонент <i>TSQLClientDataSet</i>	205
6.7.1. Соединение с сервером БД.....	206
6.7.2. Сохранение изменений.....	206
6.7.3. Работа с записями	208
6.7.4. Обработка исключительных ситуаций	208
6.8. Способы редактирования данных.....	209
6.9. Интерфейсы dbExpress	213
6.9.1. Интерфейс <i>ISQLDriver</i>	213
6.9.2. Интерфейс <i>ISQLConnection</i>	214
6.9.3. Интерфейс <i>ISQLCommand</i>	215
6.9.4. Интерфейс <i>ISQLCursor</i>	216
6.10. Отладка приложений с технологией dbExpress.....	217
6.11. Распространение приложений с технологией dbExpress.....	219
Резюме.....	220
Глава 7. Использование ADO средствами Delphi	221
7.1. Основы ADO	221
7.1.1. Перечислители	224

7.1.2. Объекты соединения с источниками данных	225
7.1.3. Сессия	225
7.1.4. Транзакции	226
7.1.5. Наборы рядов	226
7.1.6. Команды	227
7.2. Провайдеры ADO	228
7.3. Реализация ADO в Delphi	229
7.3.1. Компоненты ADO	229
7.3.2. Механизм соединения с хранилищем данных ADO	230
7.4. Компонент <i>TADOConnection</i>	230
7.4.1. Настройка соединения	231
7.4.2. Управление соединением	236
7.4.3. Доступ к связанным наборам данных и командам ADO	239
7.4.4. Объект ошибок ADO	241
7.4.5. Транзакции	241
7.5. Наборы данных ADO	243
7.5.1. Класс <i>TCustomADODataSet</i>	244
Набор данных	244
Курсор набора данных	245
Локальный буфер	247
Состояние записи	248
Фильтрация	249
Поиск	250
Сортировка	251
Команда ADO	251
Групповые операции	252
7.5.2. Параметры	253
Класс <i>TParameters</i>	254
Класс <i>TParameter</i>	255
7.5.3. Компонент <i>TADODataSet</i>	256
7.5.4. Компонент <i>TADOTable</i>	257
7.5.5. Компонент <i>TADOQuery</i>	258
7.5.6. Компонент <i>TADOStoredProc</i>	258
7.6. Команды ADO	258
7.7. Объект ошибок ADO	260
7.8. Пример приложения ADO	261
7.8.1. Соединение с источником данных	265
7.8.2. Групповые операции	266
7.8.3. Фильтрация	266
7.8.4. Сортировка	266
Резюме	266

Часть III. РАСПРЕДЕЛЕННЫЕ ПРИЛОЖЕНИЯ БАЗ ДАННЫХ	269
Глава 8. Технология DataSnap. Механизмы удаленного доступа	271
8.1. Структура многозвенного приложения в Delphi.....	272
8.2. Трехзвенное приложение в Delphi.....	275
8.2.1. Сервер приложения.....	276
8.2.2. Клиентское приложение.....	277
8.3. Механизм удаленного доступа к данным DataSnap.....	278
8.3.1. Компонент <i>TDCOMConnection</i>	279
8.3.2. Компонент <i>TSocketConnection</i>	280
8.3.3. Компонент <i>TWebConnection</i>	283
8.3.4. Компонент <i>TCORBAConnection</i>	284
8.4. Вспомогательные компоненты — брокеры соединений	285
8.4.1. Компонент <i>TSimpleObjectBroker</i>	286
8.4.2. Компонент <i>TLocalConnection</i>	287
8.4.3. Компонент <i>TSharedConnection</i>	288
8.4.4. Компонент <i>TConnectionBroker</i>	288
Резюме.....	289
Глава 9. Сервер приложения.....	290
9.1. Структура сервера приложения.....	291
9.2. Интерфейс <i>IAppServer</i>	293
9.3. Удаленные модули данных	295
9.3.1. Удаленный модуль данных для сервера Автоматизации	296
9.3.2. Дочерние удаленные модули данных.....	301
9.4. Провайдеры данных	302
9.5. Интерфейс <i>IProviderSupport</i>	306
9.6. Регистрация сервера приложения.....	307
9.7. Пример простого сервера приложения	307
9.7.1. Главная форма сервера приложения.....	308
9.7.2. Главный удаленный модуль данных	309
9.7.3. Дочерний удаленный модуль данных	310
9.7.4. Регистрация сервера приложения	311
Резюме.....	312
Глава 10. Клиент многозвенного распределенного приложения.....	313
10.1. Структура клиентского приложения	314
10.2. Клиентские наборы данных	316
10.3. Компонент <i>TClientDataSet</i>	317
10.3.1. Получение данных от компонента-провайдера	318
10.3.2. Кэширование и редактирование данных.....	320
10.3.3. Управление запросом на сервере.....	322
10.3.4. Использование индексов.....	323
10.3.5. Сохранение набора данных в файле	325

10.3.6. Работа с данными типа BLOB	325
10.3.7. Представление данных в формате XML	326
10.4. Агрегаты	326
10.4.1. Объекты-агрегаты	327
10.4.2. Агрегатные поля.....	329
10.4.3. Группировка полей и использование индексов.....	330
10.5. Вложенные наборы данных.....	331
10.6. Дополнительные свойства полей клиентского набора данных	332
10.7. Обработка ошибок	332
10.8. Пример "тонкого" клиента.....	336
10.8.1. Соединение клиента с сервером приложения	338
10.8.2. Наборы данных клиентского приложения	339
Резюме.....	340
Часть IV. Основы разработки INTERNET-ПРИЛОЖЕНИЙ	341
Глава 11. Сокеты.....	343
11.1. Введение в архитектуру сетей.....	344
11.1.1. Модель OSI.....	344
11.1.2. Физический уровень	346
11.1.3. Протоколы канального уровня	347
Управление доступом к среде	348
Управление логическим каналом (Logical Link Control)	349
11.1.4. Функции сетевого уровня.....	350
11.1.5. Транспортный уровень	353
11.1.6. Концепции сессионного уровня, уровней представления и приложений.....	354
Сессионный (session) уровень.....	355
Уровень представления.....	355
Уровень приложения.....	355
11.2. Модель клиент/сервер и сокеты	356
11.2.1. Microsoft Windows и сокеты	357
11.3. Работа на уровне сокетов в Delphi.....	358
11.3.1. Компоненты Delphi, инкапсулирующие сокеты	358
11.3.2. Установление соединения	359
11.3.3. Синхронизация сокетов.....	360
11.3.4. Класс <i>TServerWinSocket</i>	361
11.3.5. Компонент <i>TServerSocket</i>	364
11.3.6. Класс <i>TClientWinSocket</i>	364
11.3.7. Компонент <i>TClientSocket</i>	365
Резюме.....	366

Глава 12. Криптографическая защита информации в Internet.....	367
12.1. Основные термины и понятия криптографии	368
12.1.1. Секретность.....	368
12.1.2. Аутентификация	370
12.1.3. Целостность.....	371
12.2. Цифровые подписи, сертификаты и их применение	372
12.2.1. Использование <i>CryptoAPI</i>	378
Структура <i>CryptoAPI</i>	379
Пример использования <i>CryptoAPI</i> — менеджер сертификатов.....	381
12.2.2. Служба сертификатов Microsoft	386
12.3. Протоколы Internet для защищенных соединений.....	387
12.3.1. Настройка SSL на стороне сервера IIS 5.....	390
12.3.2. Настройка протокола SSL на клиентской стороне	393
Резюме.....	395
Глава 13. Потоки и процессы	397
13.1. Обзор потоков	398
13.1.1. Потоки и процессы	399
13.1.2. Фоновые процедуры, или как обойтись без потоков.....	400
13.1.3. Приоритеты потоков	400
13.2. Класс <i>TThread</i>	404
13.3. Пример создания многопоточного приложения в Delphi.....	408
13.4. Проблемы синхронизации потоков	412
13.4.1. Тупики	413
13.4.2. Гонки.....	413
13.5. Средства синхронизации потоков.....	414
13.5.1. Событие	415
13.5.2. Взаимные исключения.....	417
13.5.3. Семафор.....	417
13.5.4. Критическая секция	418
13.5.5. Процесс. Порождение дочернего процесса.....	419
13.5.6. Поток	420
13.5.7. Консольный ввод.....	420
13.5.8. Оповещение об изменении в файловой системе.....	421
13.6. Локальные данные потока	422
13.7. Как избежать одновременного запуска двух копий одного приложения	423
Резюме.....	424

Часть V. Данные XML в распределенных приложениях	425
Глава 14. Использование XML	427
14.1. Что такое XML и для чего он предназначен	427
14.2. Основы синтаксиса XML	430
14.2.1. Пролог	431
14.2.2. Определение	433
14.2.3. Тело документа. Корневой элемент	434
14.3. Объектная модель документа	435
14.3.1. Интерфейсы семейства <i>IDOMNode</i>	436
Свойства <i>nodeType</i> , <i>nodeName</i> и <i>nodeValue</i>	437
Свойства и методы, управляющие другими вершинами	439
Пространства имен	441
Интерфейс <i>IDOMDocument</i>	441
14.3.2. Пример создания приложения, использующего модель DOM	442
14.4. Реализация модели DOM в Delphi	445
Модуль <i>Xmldom</i>	446
Модуль <i>Msxml</i>	446
Модуль <i>Msxmldom</i>	447
Модуль <i>XMLIntf</i>	447
Модуль <i>XMLDoc</i>	447
14.4.1. Интерфейс <i>IXMLNode</i> и его отличия от стандарта DOM	447
14.4.2. Взаимосвязь между всеми интерфейсами	450
14.4.3. Загрузка XML	451
Асинхронная загрузка	453
Функции, создающие экземпляр документа	453
14.4.4. Обработка ошибок анализатора	454
Пример использования интерфейсов <i>IXMLNode</i> и <i>IXMLDocument</i>	456
14.5. Анализатор MSXML, или Microsoft XML Core Services	469
Резюме	476

Глава 15. Использование данных в формате XML

15.1. Преобразование данных в формате XML	477
15.1.1. Схема преобразования данных XML	478
15.1.2. Формат пакета данных Delphi	479
15.1.3. Инструментарий преобразования данных XML	480
15.2. Утилита XML Mapper	481
15.2.1. Выбор исходного файла	482
15.2.2. Создание пакета данных и документа XML и сохранение преобразованных данных	483

15.2.3. Связывание элементов XML и полей пакета данных.....	484
15.2.4. Создание трансформационного файла и преобразование данных.....	485
15.3. Преобразование данных XML в распределенных приложениях.....	486
15.4. Использование данных XML в распределенных приложениях	489
15.4.1. Данные XML в клиентском наборе данных.....	491
15.4.2. Данные XML в документе XML	491
15.4.3. Данные XML на странице HTML	493
15.5. Пример приложения, использующего данные XML.....	495
Резюме.....	498
Часть IV. РАСПРЕДЕЛЕННЫЕ ПРИЛОЖЕНИЯ И WEB-СЛУЖБЫ	499
Глава 16. Серверные Web-приложения. Технология Web Broker	501
16.1. Публикация данных в Internet. Web-серверы	502
16.2. Введение в интерфейсы CGI и ISAPI	505
16.3. Типы серверных Web-приложений.....	507
16.4. Структура серверного Web-приложения в Delphi.....	508
16.4.1. Глобальный объект приложения	511
16.4.2. Web-модуль.....	513
16.4.3. Действия	516
16.4.4. Запросы и ответы	517
16.4.5. Компоненты-продюсеры	518
16.5. Страницы HTML в серверных Web-приложениях	520
16.5.1. Отображение данных.....	520
16.5.2. Ввод и редактирование данных	523
16.6. Cookies.....	527
16.7. Использование баз данных	528
16.7.1. Публикация записей таблиц баз данных	529
16.7.2. Генерация отчетов	530
16.7.3. Редактирование данных.....	535
16.8. Пример простого серверного Web-приложения.....	536
16.8.1. Главное окно приложения.....	537
16.8.2. Просмотр таблицы <i>Country</i>	538
16.8.3. Тестирование приложения	541
16.8.4. Перенос приложения на платформу ISAPI	542
Отладка ISAPI-приложений	542
Резюме.....	546
Глава 17. Протокол SOAP и Web-службы. Клиентская часть	547
17.1. Почему SOAP?.....	547
17.1.1. Delphi и <i>TSomeConnection</i> — потенциальные проблемы	548

17.2. SOAP: краткое содержательное описание.....	549
17.2.1. Web-службы.....	551
17.2.2. Описание протокола WSDL.....	552
17.3. Архитектура Web-служб в Delphi.....	557
17.4. Клиент Web-службы.....	558
17.4.1. Генерация интерфейса Web-службы.....	559
17.4.2. O, RIO, RIO.....	563
17.4.3. Как связаться с нужной службой.....	565
17.4.4. Решение коммуникационных проблем.....	566
17.4.5. Что такое <i>Invoke Registry</i> . Регистрация интерфейсов и типов данных.....	569
Резюме.....	572
Глава 18. Серверная часть Web-службы. Совместимость.....	573
18.1. Создание тестового примера — службы <i>SimpleEcho</i>	573
18.2. Назначение и настройки компонентов серверной части.....	576
18.2.1. Компонент <i>TWSDLHTMLPublish</i>	577
18.2.2. Компоненты <i>THTTTPSoapDispatcher</i> и <i>THTTTPSoapPascalInvoker</i>	580
18.2.3. Клиент службы <i>SimpleEcho</i>	581
18.2.4. Обработка исключительных ситуаций.....	582
18.2.5. Несколько слов об использовании <i>TSOAPConnection</i>	584
18.3. Средства разработки для SOAP: подход Microsoft.....	585
18.3.1. Краткий обзор архитектуры Web-служб на SOAP Toolkit.....	586
18.3.2. Сценарий клиентской части.....	588
18.3.3. Создание COM-объекта <i>SimpleEchoCOM</i> и публикация Web-службы на его основе.....	590
18.4. Использование утилиты SOAP Trace.....	593
Резюме.....	595
Глава 19. Технология WebSnap.....	596
19.1. Структура приложения WebSnap.....	596
19.1.1. Обработка запросов.....	598
19.1.2. Web-модули.....	599
Модуль данных.....	600
Модуль страницы.....	601
Модуль приложения.....	602
19.1.3. Компоненты уровня приложения.....	603
Управление компонентами приложения.....	603
Адаптер и глобальная переменная приложения.....	605
Информация о пользователях.....	606
Сессии.....	608
Управление файлами приложения.....	609

19.1.4. Компоненты-диспетчеры.....	609
19.1.5. Компоненты-продюсеры	611
19.1.6. Компоненты-адаптеры.....	611
19.2. Создание приложения WebSnap.....	615
19.3. Конструирование интерфейса и управление данными	618
19.3.1. Навигация по страницам приложения.....	620
19.3.2. Использование полей и действий компонентов-адаптеров	621
19.3.3. Взаимодействие с базами данных.....	624
Специализированные элементы управления	624
Просмотр данных	626
Редактирование данных.....	627
19.3.4. Обработка ошибок.....	628
19.4. Аутентификация пользователей	629
19.4.1. Списки пользователей и сессии	629
19.4.2. Создание страницы аутентификации.....	630
19.4.3. Настройка приложения.....	631
19.5. Использование XML и XSL.....	632
19.5.1. Использование файла XML и шаблона XSL.....	633
19.5.2. Использование набора данных и шаблона XSL	635
19.5.3. Дополнительный компонент <i>TAdapterXMLBuilder</i>	636
Резюме.....	638

ПРИЛОЖЕНИЕ. ОПИСАНИЕ ДИСКЕТЫ.....	639
--	------------

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ	643
-----------------------------------	------------

Введение

При первом знакомстве с Delphi разработчики наверняка обратили внимание на то, что большинство новых возможностей и компонентов ориентированы на создание распределенных Web приложений. И, безусловно, это не дань моде, в которой фирму Borland трудно заподозрить, а насущная необходимость.

Интересно проследить эволюцию взглядов сообщества разработчиков на возможности Internet. Первоначальное представление о глобальной сети, как об очередной "технологической игрушке", имеющей, впрочем, потрясающие перспективы не только общения, но и ведения бизнеса, неизбежно породило Internet-бум. Безусловно, памятен всем ажиотаж, когда использовать Internet часто пытались не те люди и не для тех целей. Естественно, это вызвало некоторое разочарование и откат. Тем не менее, взрыв интереса к Internet как универсальному и глобальному средству бизнес-коммуникации и последовавший спад активности в этой сфере позволили выявить реальные перспективы применения Internet не только как важнейшего технологического средства, но и как системообразующего фактора проектирования современных распределенных приложений.

Всю историю развития от бизнес-приложений для мейнфреймов и до современных распределенных систем можно представить как последовательность шагов, каждый из которых делал приложения более глобальными. Безусловно, с точки зрения беспристрастного исторического исследования, такой взгляд страдает некоторой однобокостью, но в рамках данной книги представляется вполне оправданным, так как позволяет сосредоточиться на исследуемом вопросе. В этом контексте широкое использование Internet в качестве сетевой среды для распределенных бизнес-приложений является закономерным и обоснованным решением.

Популярность Internet как системообразующей платформы для распределенных приложений основана также и на крайне своевременном появлении языка XML, который позволяет унифицировать процесс представления

структуры данных. Задача описания данных XML решается при помощи другого языка — XSL (eXtended Stylesheet Language) и его модификации XSLT (XSL Transformations). И это очень важно, так как разработчики получили универсальное средство представления этих данных в море разнообразных клиентских приложений, Web-браузеров, протоколов и их расширений.

В этой книге авторы постарались подробно осветить все основные аспекты создания Web-приложений при помощи богатого инструментария среды разработки Delphi. Однако не это, а точнее не только это является главным преимуществом Delphi, как среды для создания Web-приложений. Очень важно, что Delphi позволяет осуществить полный цикл разработки сложных приложений с распределенной инфраструктурой, использующих различные технологические решения.

Действительно, разнообразие технологий и компонентов позволяет создавать приложения для любых заказчиков и выполнять большинство их пожеланий. И чтобы не быть голословными, давайте совершим небольшой экскурс в современные Internet-технологии, о которых пойдет речь в этой книге.

COM и COM+

А начнем мы с небольшого обзора технологии COM (Component Object Model) и ее дочерних технологий. В этот момент искушенный читатель может поморщиться — "Опять COM!". Однако авторы сочли необходимым обратиться к этой теме, хотя на первый взгляд вопросы применения COM и иже с ней довольно далеки от Web-приложений. Де факто, многие самые современные Internet-технологии используют возможности COM как последнее звено в цепочке передачи данных — если можно так выразиться, как аналог "последней мили" сетевой инфраструктуры.

Не меньшее значение имеет и технология COM + (и ее ближайший предок — старый добрый Microsoft Transaction Server), помогающая решить проблемы масштабируемости приложений для платформы Windows 2000.

DataSnap

Рассказывая о Web-приложениях, не следует забывать о том, что Internet — безусловно замечательное и перспективное, но лишь "транспортное средство" для получения данных клиентами распределенных бизнес-приложений. Поэтому особенности архитектуры таких приложений и вопросы их взаимодействия с серверами баз данных по-прежнему актуальны.

Технология DataSnap, унаследовавшая и развившая возможности небезызвестной для профессионалов Delphi технологии MIDAS, предназначена для решения именно таких задач. Здесь речь пойдет о многозвенных распределенных приложениях и способах доступа к разнообразным источникам данных.

Необходимо отдельно сказать несколько слов о разработанной и реализованной фирмой Microsoft объектной модели доступа к данным ADO (ActiveX Data Objects), которая де факто стала стандартом в этой области. Она позволяет решить проблему распространения и инсталляции программного обеспечения промежуточного слоя доступа к данным для платформы Windows.

XML

Термин "XML" слышали все, сам язык знают многие, но далеко не все представляют горизонты практического применения этого средства. Аббревиатура XML расшифровывается как eXtensible Markup Language — расширяемый язык разметки. Разрабатывается он консорциумом World Wide Web Consortium (сокращенно W3C, www.w3.org/XML). В консорциум входят представители крупнейших фирм-разработчиков, учебных и научных заведений.

XML потому и называется "расширяемым", что дает возможность любому специалисту определить и построить свою систему тегов, описывающую его собственную предметную область.

Кроме того, XML лежит в основе многих других технологий, используемых в Delphi для создания Web-приложений.

SOAP

Со времен основания и до настоящего времени разработчики Web-приложений сталкиваются с проблемой совместимости распределенных приложений, части которых функционируют на различных программных и аппаратных платформах. В этой области предлагались различные решения. К примеру, архитектура CORBA (Common Object Request Broker Architecture) позволяет создавать полноценные мультиплатформенные решения, но требует установки специализированного клиентского программного обеспечения. Если вы хотите распространять решения на основе VisiBroker — поставляемого с Delphi брокера объектных запросов — вам следует отдельно позаботиться о приобретении прав на него. (Кстати, поддержка CORBA обеспечена в Delphi, а набор соответствующих компонентов, инструментов и утилит входит в технологию DataSnap.)

Протокол Simple Object Access Protocol (SOAP) поддерживается всеми основными платформами, представленными в Internet, не требует сложной клиентской части и обеспечивает безопасную передачу данных по Сети. Появление SOAP обусловлено хорошо известной проблемой — необходимостью совместимости приложений, работающих на разных платформах. За два года использования SOAP одобрен большинством IT-сообщества.

Набор компонентов Delphi, реализующих SOAP, позволяет конструировать полноценные клиентские и серверные части Web-приложений. Кроме этого, мы рассмотрим Microsoft SOAP Toolkit — инструмент для публикации объектов Автоматизации в качестве Web-служб и обсудим вопросы создания Web-служб.

WebSnap и WebBroker

И наконец, разработчики Delphi сумели объединить в едином технологическом решении все основные этапы создания Web-приложений. Технологии WebSnap и WebBroker упрощают и ускоряют выполнение "рутинных" операций создания пользовательского интерфейса приложений, доступа к данным, аутентификации пользователей. Кроме этого, разработчики наверняка оценят легкость встраивания в исходный код скриптов VBScript или JScript.

Широкое использование шаблонов HTML и XML является несомненным преимуществом WebSnap, так как существенно упрощает модернизацию наиболее изменчивой и подвижной части готовых приложений — пользовательского интерфейса.

Криптография и безопасность данных

Сеть Internet по сути является открытой, децентрализованной и минимально управляемой. На ранних этапах развития программирования для Internet проблема защиты данных волновала немногих, а разговоры о создании промышленных стандартов в этой сфере носили теоретический характер. В настоящее же время парадигма защиты любых важных данных, пересылаемых по Сети, не подлежит сомнению.

Поэтому в этой книге мы не могли обойти вниманием столь важную тему. Здесь рассматривается круг прикладных задач криптографии, затем вводятся понятия цифровой подписи и сертификата. Далее, на примере менеджера сертификатов, мы изучим особенности интерфейса CryptoAPI и закончим вопросами настройки Web-средств и компонентов, используемых в распределенных приложениях для криптографической защиты.



Часть I

Приложения COM и COM+

Глава 1. Механизмы COM в Delphi

Глава 2. Технология Автоматизация

Глава 3. Компоненты ActiveX

Глава 4. Технология COM+
(Microsoft Transaction Server)

Глава 1

Механизмы COM в Delphi



В основе многих технологий, рассматриваемых в этой книге, лежит *технология COM (Component Object Model, Многокомпонентная Модель Объектов)* и технологии, созданные на ее базе. Поэтому мы начнем именно с рассмотрения главных возможностей COM.

Во все времена одной из актуальнейших задач, стоящих перед специалистами, было обеспечение взаимодействия между отдельными программами. Для ее решения использовался целый арсенал различных способов и приемов.

На заре существования Windows были внедрены *разделяемые файлы, буфер обмена и технология динамического обмена данными (Dynamic Data Exchange, DDE)*.

Для обеспечения обмена данными и предоставления служб был разработан первый вариант *технологии связывания и внедрения объектов (Object Linking and Embedding, OLE 1)*. Она предназначалась для создания составных документов — того, к чему мы все уже давно привыкли. Эта технология была во многом несовершенной, и на смену ей пришла технология OLE 2. Она позволяет решить более общую проблему — предоставление разными программами собственных функций (служб) друг другу и правильное использование ими этих функций.

Для решения этой проблемы помимо OLE был разработан целый ряд технологий. В их основе лежит базовая технология Component Object Model (COM). Она описывает способ взаимодействия программ разных типов. Одна часть программного обеспечения предоставляет собственные службы, а другая получает к ним доступ. При этом не имеет значения, где расположены эти части — в одном процессе, в разных процессах на одном компьютере или на разных компьютерах.

Для приложений, созданных на основе спецификации COM, также не важно, какой язык программирования использовался при их разработке — если стандарт COM соблюден, взаимодействие осуществляется без помех.

Дополнительные возможности разработчикам распределенных приложений на основе COM дает модификация базовой технологии — *распределенная модель COM (Distributed COM, DCOM)*.

В настоящее время COM используется в самых различных областях разработки программного обеспечения. На основе COM созданы технологии Автоматизации (Automation), ActiveX, ActiveForm, Microsoft Transaction Server. Без применения объектов COM не смогут функционировать распределенные приложения. Независимо от того, работают ли они в локальной сети или используют Internet.

Delphi предоставляет разработчику набор инструментов для создания полноценных приложений COM.

Далее в этой главе обсуждаются основные части спецификации COM и методика создания объектов и интерфейсов COM в Delphi. Значительное внимание уделяется Редактору библиотеки типов — основному инструменту, который облегчает работу с объектами COM в проекте.

В этой главе рассматриваются следующие темы:

- объекты и интерфейсы;
- интерфейс Iunknown;
- библиотеки типов;
- фабрики классов;
- виды серверов.

1.1. Базовые понятия

В технологии COM приложение предоставляет для использования свои службы, применяя для этого *объекты COM*. Одно приложение содержит как минимум один объект. Каждый объект имеет один или несколько *интерфейсов*. Каждый интерфейс объединяет *методы* объекта, которые обеспечивают доступ к *свойствам* (данным) и выполнение операций. Обычно в интерфейсе объединяются все методы, выполняющие операции одного типа или работающие с однородными свойствами.

Клиент получает доступ к службам объекта только через интерфейс и его методы. Этот механизм является ключевым. Клиенту достаточно знать несколько базовых интерфейсов, чтобы получить исчерпывающую информацию о составе свойств и методов объекта. Поэтому любой клиент может работать с любым объектом, независимо от их среды разработки. Согласно спецификации COM, уже созданный интерфейс не может быть изменен ни при каких обстоятельствах. Это гарантирует постоянную работоспособность приложений на основе COM, невзирая на любые модернизации.

Объект всегда работает в составе *сервера COM*. Сервер может быть динамической библиотекой или исполняемым файлом. Объект может иметь собственные свойства и методы или использовать данные и службы сервера.

Для доступа к методам объекта клиент должен получить указатель на соответствующий интерфейс. Для каждого интерфейса существует собственный указатель. После этого клиент может использовать службы объекта, просто вызывая его методы. Доступ к свойствам объектов осуществляется только через методы объектов.

Предположим, что объект COM встроен в электронную таблицу и обеспечивает доступ к математическим операциям. Будет логично разделить математические функции на группы по типам и создать для каждой группы собственный интерфейс. Например, можно выделить линейные, тригонометрические, агрегатные функции и т. д. На рис. 1.1 объект расположен внутри сервера — электронной таблицы. Интерфейсы обозначены маленькими кружками, связанными с объектом. Пусть интерфейс линейных функций называется `ILinear`, а интерфейс агрегатных функций — `IAggregate`.

Примечание

Согласно принятым в Delphi правилам именования, названия интерфейсов начинаются с заглавной буквы `I`. Напомним также, что имена классов начинаются с заглавной буквы `T`.

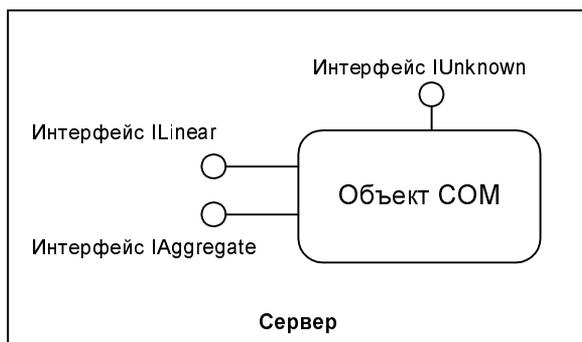


Рис. 1.1. Сервер, объект и его интерфейсы

Примечание

Согласно правилам представления объектов COM, базовый интерфейс `IUnknown` (см. рис. 1.1), имеющийся у любого объекта, обозначается как кружок, примыкающий к верхней стороне прямоугольника объекта. Остальные интерфейсы располагаются справа или слева.

На рис. 1.2 представлена схема взаимодействия клиента с объектом COM.

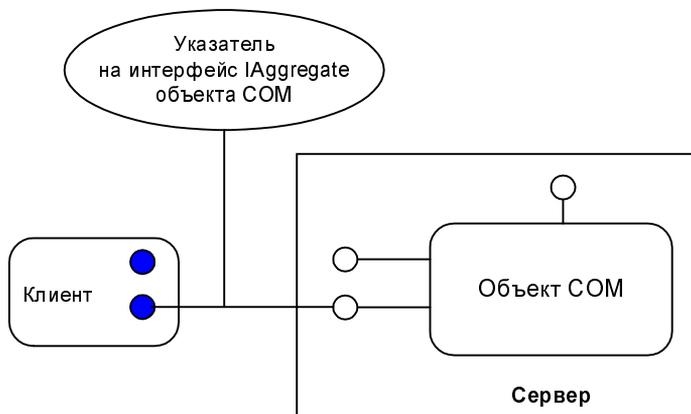


Рис. 1.2. Схема взаимодействия клиента и объекта COM

Предположим, что среди методов интерфейса `IAggregate` имеется метод вычисления среднего. Чтобы получить доступ к агрегатной функции расчета среднего, клиент должен получить указатель на интерфейс `IAggregate`, а затем обратиться к этой функции.

Взаимодействие между клиентом и объектом обеспечивается базовыми механизмами COM. При этом от клиента скрыто, где именно расположен объект: в адресном пространстве того же процесса, в другом процессе или на другом компьютере. Поэтому с точки зрения разработчика клиентского ПО использование функций электронной таблицы выглядит как обычное обращение к методу объекта. Механизм обеспечения взаимодействия между удаленными элементами COM называется *маршалингом* (*marshalling*).

Возникает закономерный вопрос — как проходит создание и инициализация объекта COM при первом обращении клиента? Сомнительно, чтобы операционная система самостоятельно создавала экземпляры всех зарегистрированных в ней классов в надежде, что один из них понадобится. А ведь для работы объекта требуются еще и серверы. Представьте, что каждый раз при загрузке Windows настойчиво запускает Word, Excel, Internet Explorer и т. д.

Любой объект COM является обычным экземпляром некоторого класса, описывающего его свойства и методы. Информация обо всех зарегистрированных и доступных в данной операционной системе классах COM собрана в специальной *библиотеке COM*, которая используется для запуска экземпляра класса — объекта.

Сначала клиент обращается к библиотеке COM, передавая ей имя требуемого класса и необходимого в первую очередь интерфейса. Библиотека находит нужный класс и сначала запускает сервер, который затем создает объект —

экземпляр класса. После этого библиотека возвращает клиенту указатели на объект и интерфейс. В последующей работе клиент может обращаться непосредственно к объекту и его интерфейсам.

После создания наступает очередь инициализации — объект должен загрузить необходимые данные, считать настройки из системного реестра и т. д. За эту часть работы отвечают специальные объекты COM, которые называются *моникерами* (*monikers*). Они работают скрытно от клиента. Обычно моникер создается вместе с классом.

Вполне реальным представляется одновременное обращение нескольких клиентов к одному объекту. При соответствующих настройках для каждого клиента создается отдельный экземпляр класса. За выполнение этой операции отвечает специальный объект COM, который называется *фабрикой класса*.

Наконец, остался нерассмотренным последний вопрос — как клиент может получить информацию об объекте. Например, разработчик клиентского ПО знает, что электронная таблица создана в соответствии со спецификацией COM, но не имеет понятия об объектах COM, которые предоставляют клиентам ее службы. Для разрешения подобных ситуаций разработчик объекта COM может распространять вместе с объектом информацию о типе. Она включает сведения об интерфейсах, их свойствах и методах, параметрах методов.

Эта информация содержится в *библиотеке типов*, которая создается при помощи специального языка описания интерфейса (*Interface Definition Language, IDL*).

1.1.1 Объект

Центральным элементом COM является объект. Приложения, поддерживающие COM, включают в себя один или несколько объектов COM. Каждый объект представляет собой экземпляр соответствующего класса и содержит один или несколько интерфейсов. Что же такое объект COM?

Не вдаваясь пока в подробности реализации объектов COM в Delphi, можно сказать, что объект COM несколько отличается от обычного объекта.

Любой объект является экземпляром некоторого класса, т. е. представляет собой переменную объектного типа. Поэтому объект обладает набором свойств и методов, которые работают с этими свойствами. Объекты создаются в соответствии с основополагающими принципами ООП, такими как: инкапсуляция, наследование и полиморфизм. Объекты COM всем этим требованиям удовлетворяют (существуют особенности наследования).

Применительно к объектам вообще понятие интерфейса объекта, как он был определен выше, не используется. В первом приближении можно считать, что все методы объекта составляют его единственный интерфейс, а указателем интерфейса является указатель на объект.

Объект COM может иметь любое число интерфейсов (если это число больше нуля), причем каждый интерфейс обладает собственным указателем. Это первое отличие объектов COM от обычных.

Примечание

Некоторые языки программирования, например, Java, позволяют объекту иметь несколько интерфейсов.

У объектов COM есть особенность еще в одном объектном механизме — наследовании. Вообще различают два способа наследования. Наследование реализации подразумевает передачу родителем потомку всего программного кода. Наследование интерфейса означает передачу только объявления методов, их программный код потомок должен предоставить самостоятельно.

Объекты COM поддерживают только наследование интерфейса, избегая тем самым возможного нарушения инкапсуляции родителя. Тем не менее, просто так выбросить наследование реализации нельзя. Вместо нее объекты COM используют механизм *включения*, т. е. при необходимости потомок вызывает нужный метод родителя. Также применяется механизм *агрегирования*, когда один или несколько интерфейсов одного объекта на время включаются в другой объект путем передачи указателей.

Таким образом, объект COM с точки зрения ООП несомненно является объектом. Однако как ключевой элемент технологии COM он обладает рядом особенностей реализации базовых механизмов.

1.1.2. Интерфейс

Если объект COM является ключевым элементом реализации COM, то интерфейсы — это центральное звено идеологии COM. Как двум принципиально разным объектам обеспечить взаимодействие друг с другом? Ответ прост: им необходимо заранее договориться о том, как они будут общаться (Авторы намеренно не используют слово "язык", так как оно может вызвать нежелательные ассоциации с языком программирования, а как раз этот фактор не имеет никакого значения при взаимодействии элементов COM.).

Интерфейс как раз является тем средством, которое позволяет клиенту правильно обратиться к объекту COM, а объекту ответить так, чтобы клиент его понял.

Рассмотрим небольшой пример. На улице случайно встретились два человека: местный житель (объект COM) и заблудившийся иностранец (клиент). Предусмотрительный иностранец захватил с собой словарь (библиотека типов или интерфейс `IUnknown`). Иностранцу нужны знания местного жителя о городе. Он достал ручку и бумагу и, заглянув в словарь, составил фразу и старательно перерисовал незнакомые слова на бумагу. Местный житель оказался не промах. Он прочитал фразу, отобрал у иностранца словарь, составил по нему собственную фразу и тоже написал ее на бумаге.

И все закончилось хорошо: довольный клиент (иностранец) получил от объекта COM (местного жителя) результат работы службы (информацию о дороге), а местный житель ушел со словарем.

Как вы уже догадались, в этом примере интерфейсом является бумага и ручка: иностранец не знает чужого языка, зато знает, *как* правильно спросить, чтобы ему ответили.

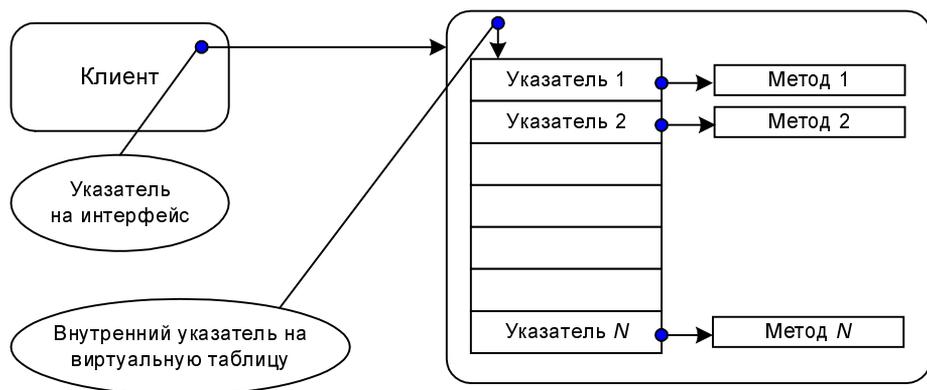
Для идентификации каждый интерфейс имеет два атрибута. Во-первых, это его имя, составленное из символов в соответствии с правилами используемого языка программирования. Каждое имя должно начинаться с символа "I". Это имя используется в программном коде. Во-вторых, это *глобальный уникальный идентификатор* (Globally Unique Identifier, GUID), который представляет собой гарантированно уникальное сочетание символов, практически не повторяемое ни на одном компьютере в мире. Для интерфейсов такой идентификатор носит название IID (Interface Identifier).

В общем случае клиент может не знать, какие интерфейсы есть у объекта. Для получения их перечня используется базовый интерфейс IUnknown (см. разд. 1.1.3 данной главы), который есть у любого объекта COM.

Затем клиенту необходимо знать, какие методы имеет выбранный им интерфейс. Для этого разработчик должен распространять описание методов интерфейсов вместе с объектом. Эту задачу помогает решать язык IDL (он также используется в библиотеках типов). Его синтаксис очень похож на C++.

Теперь осталось сделать самое важное — правильно вызвать метод. Для этого в COM описана реализация интерфейса на основе стандартного двоичного формата, обеспечивающая независимость от языка программирования.

Доступный клиенту указатель интерфейса ссылается на внутренний указатель объекта и, через него, на специальную виртуальную таблицу (рис. 1.3).



Объект COM

Рис. 1.3. Внутренняя структура интерфейса COM

Эта таблица содержит указатели на все методы интерфейса (Не правда ли, очень похоже на таблицу виртуальных методов объекта в ООП?).

Примечание

Первые три строки таблицы интерфейса всегда заняты под методы интерфейса `IUnknown`, так как любой интерфейс COM является его наследником.

В результате, вызов клиентом метода проходит по цепочке указателей и получает указатель на конкретный метод, а затем выполняется соответствующий программный код.

1.1.3. Интерфейс *IUnknown*

Каждый объект COM обязательно имеет интерфейс `IUnknown`. У этого интерфейса всего три метода, но они играют ключевую роль в функционировании объекта.

Метод `QueryInterface` возвращает указатель на интерфейс объекта, идентификатор ID которого передается в параметре метода. Если такого интерфейса объект не имеет, метод возвращает `NULL`.

Обычно при первом обращении к объекту клиент получает указатель на интерфейс. Поскольку любой интерфейс является потомком `IUnknown`, то у него есть и метод `QueryInterface`. Следовательно, в общем случае не важно, какой именно интерфейс может использовать клиент. При помощи метода `QueryInterface` он может получить доступ к любому интерфейсу объекта.

Интерфейс `IUnknown` обеспечивает работу еще одного важного механизма объекта COM — механизма учета ссылок. Объект должен существовать до тех пор, пока его использует хотя бы один клиент. При этом клиент не может самостоятельно уничтожить объект, ведь с ним могут работать и другие клиенты.

При передаче наружу очередного указателя на интерфейс объект увеличивает специальный счетчик ссылок на единицу. Если один клиент передает другому указатель на интерфейс этого объекта, то клиент, получающий указатель, обязан еще раз увеличить счетчик ссылок. Для этого используется метод `AddRef` интерфейса `IUnknown`.

При завершении работы с интерфейсом клиент обязан вызвать метод `Release` интерфейса `IUnknown`. Этот метод уменьшает счетчик ссылок на единицу. После обнуления счетчика объект уничтожает себя.

1.1.4. Сервер

Сервер COM представляет собой приложение или динамическую библиотеку, которые могут содержать один или несколько объектов одного или разных классов. Различают три типа серверов.

Внутренний сервер (in-process server) реализуется динамическими библиотеками, которые подключаются к приложению-клиенту и работают в одном с ним адресном пространстве.

Локальный сервер (local server) создается отдельным процессом, который работает на одном компьютере с клиентом.

Удаленный сервер (remote server) создается процессом, который работает не на том компьютере, на котором выполняется клиент.

Рассмотрим локальный сервер. Получаемый клиентом указатель интерфейса в этом случае ссылается на специальный *proxy-объект* COM (назовем его *заместителем*), который функционирует внутри клиентского процесса. Заместитель предоставляет клиенту те же интерфейсы, что и вызываемый объект COM на локальном сервере. Получив вызов от клиента, заместитель упаковывает его параметры и при помощи служб операционной системы передает вызов в процесс сервера. На локальном сервере вызов передается еще одному специализированному объекту — *заглушке (stub)*, который распаковывает вызов и передает его требуемому объекту COM. Результат вызова возвращается клиенту в обратном порядке.

Рассмотрим удаленный сервер. Он функционирует так же, как и локальный сервер. Однако передача вызовов между двумя компьютерами осуществляется средствами DCOM — с помощью механизма *вызова удаленных процедур (Remote Procedure Call, RPC)*.

Для обеспечения работы локальных и удаленных серверов используется механизм маршрутирования и демаршрутирования. *Маршрутирование* реализует единый в рамках технологии COM формат упаковки параметров запроса, *демаршрутирование* обеспечивает распаковку. В описанных выше реализациях серверов за выполнение этих операций отвечают заместитель и заглушка. Объекты этих типов создаются совместно с основным объектом COM с помощью IDL.

1.1.5. Библиотека COM

Для выполнения базовых функций и использования интерфейсов в операционной системе существует специальная *библиотека COM* (конкретная реализация может быть различной). Доступ к возможностям библиотеки осуществляется стандартным способом — с помощью вызова функций. Согласно спецификации, имена всех библиотечных функций начинаются с приставки "Co".

При установке приложения, поддерживающего COM, в системный реестр записывается информация обо всех реализуемых этим приложением объектах COM:

- ❑ *идентификатор класса (Class Identifier, CLSID)*, который однозначно определяет класс объекта;
- ❑ тип сервера объекта — внутренний, локальный или удаленный;
- ❑ полное имя динамической библиотеки или исполняемого файла — для локальных и внутренних серверов;
- ❑ полный сетевой адрес — для удаленных серверов.