bhv®

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	*Екатерина Кондукова*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Татьяна Коротяева*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Оформление серии	*Via Design*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.10.02.
Формат $70\times100^{1}/_{16}$. 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.

ISBN 5-94157-142-9

Содержание

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

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.

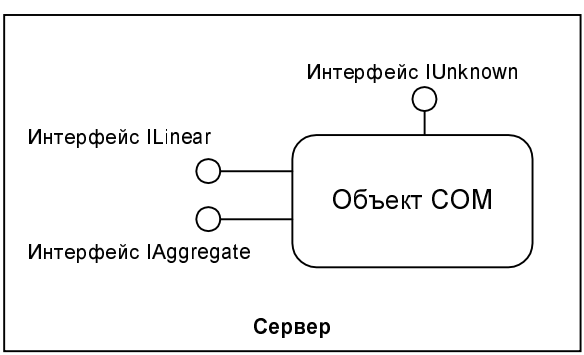

Рис. 1.1. Сервер, объект и его интерфейсы

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.

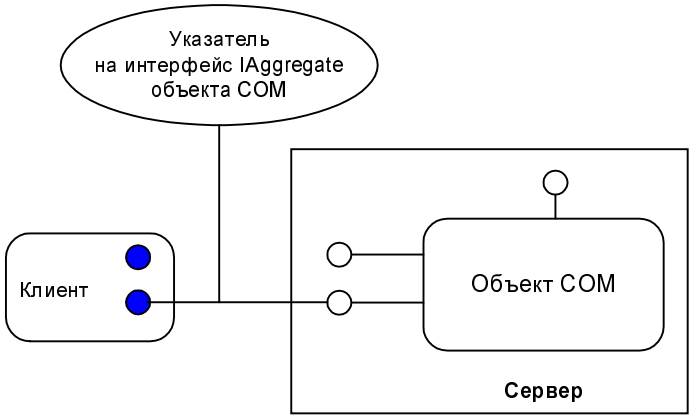

Рис. 1.2. Схема взаимодействия клиента и объекта COM

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. Обекты 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).

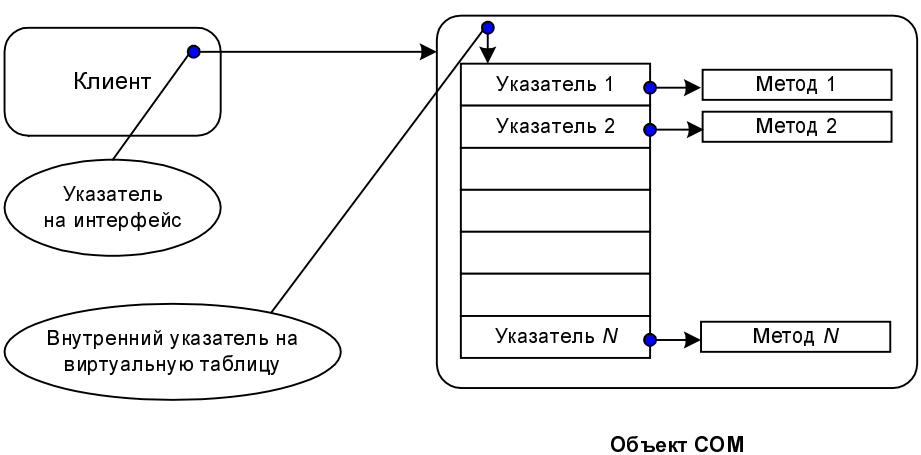

Рис. 1.3. Внутренняя структура интерфейса COM

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 исполняется 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 IID 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 маршалинга и демаршалинга. *Маршалинг* реализует единый в рамках технологии COM формат упаковки параметров запроса, *демаршалинг* 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.