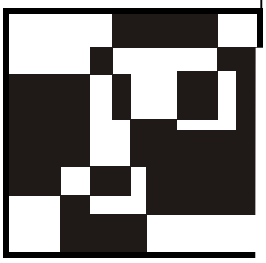

ГЛАВА 4

Работа с объектами и объектные модели

В предыдущей главе вы познакомились с синтаксическими конструкциями и встроенными функциями языка VBA, но с одними встроенными функциями много не сделаешь. Вся функциональность программ VBA, работающих в приложениях Microsoft Office, полностью построена на работе с классами и объектами.

Объектно-ориентированное программирование — это отдельная очень большая тема, которой посвящено множество книг. В этой главе будут рассмотрены только те моменты, которые важны с точки зрения практической работы при программировании в среде Microsoft Office.

4.1. Что такое классы и объекты

Классы формально определяются как блоки функциональности, которые можно использовать в программах. Для наших целей их можно считать "чертежами" для создания объектов. На основе этих "чертежей" создаются экземпляры классов — объекты. Для простоты можно представить себе, что в оперативной памяти компьютера по чертежу построили дом — объект, с которым можно что-то делать.

Наборы чертежей (в терминологии программирования — коллекции классов) обычно называются библиотеками типов. В Windows они "упаковываются" в файлы DLL или OCX (иногда и в файлы других типов, например, EXE или TLB). Такие библиотеки типов откомпилированы — чертежи из них можно использовать, но просмотреть их (т. е. просмотреть исходный код класса) нельзя, для такой ситуации существует специальный термин — технология "черного ящика".

Чаще всего в программе VBA создается объект определенного класса (по-английски это называется *instantiation*, создание экземпляра — *instance*), и

далее работа производится с этим объектом. В одной программе вполне можно использовать несколько разных объектов одного и того же класса.

В этой книге рассказывается только про использование в программах объектов, созданных на основе уже готовых чертежей (которые приготовили для вас разработчики Microsoft Office). Создание своих собственных классов — это отдельная большая тема, которая выходит за рамки данной книги.

4.2. Создание и удаление объектов

Создание объекта в VBA может производиться разными способами. Самый простой способ выглядит так:

```
Dim oApp As Object
Set oApp = CreateObject("Word.Application")
MsgBox oApp.UserName
```

Это так называемое *позднее связывание* (late binding). Вначале мы объявляем переменную `oApp` с возможностью ссылаться на любой объект, а затем присваиваем ей ссылку на создаваемый нами объект `Word.Application`. Такое позднее связывание хуже с точки зрения производительности и расхода оперативной памяти, кроме того, редактор Visual Basic отказывается нам подсказывать, какие свойства и методы есть у этого объекта. Поэтому позднее связывание имеет смысл использовать только тогда, когда вы собираетесь хранить в этой переменной, согласно логике вашей программы, объекты разных типов. В остальных ситуациях предпочтительнее использовать *раннее связывание* (early binding):

```
Dim oApp As Word.Application
Set oApp = CreateObject("Word.Application")
MsgBox oApp.UserName
```

В этом случае мы сразу присваиваем переменной `oApp` тип `Word.Application`, а потом присваиваем ей ссылку на создаваемый нами объект. Можно обойтись и без второй строки:

```
Dim oApp As New Word.Application
MsgBox oApp.UserName
```

Однако ключевое слово `New` не может использоваться при создании зависимых объектов, с ключевым словом `WithEvents` и при создании переменных встроенных типов (`String`, `Integer` и т. п.), поэтому иногда необходимо использовать только объявление с `Set`. Кроме того, в языке VBScript синтаксической конструкции `New` нет, поэтому если вы собираетесь использовать оба языка, лучше сразу привыкать к конструкции `CreateObject()`. Но поскольку

конструкция с `New` короче, в VBA традиционно чаще используется именно она.

Еще одна возможность создания объекта — *воспользоваться методом другого объекта*, который создаст нужный нам объект и возвратит ссылку на него напрямую или через коллекцию:

```
Dim oApp As New Word.Application
oApp.Documents.Add
Dim oDoc As Word.Document
Set oDoc = oApp.Documents(1)
oDoc.SaveAs "C:\docvba1.doc"
```

В этом примере мы вначале создаем объект `Word.Application`, затем при помощи метода `Add()` коллекции `Documents` создаем в этой коллекции новый документ, потом получаем на него ссылку для переменной `oDoc`, а потом вызываем метод `SaveAs()` созданного нами объекта документа.

Удаление объектов производится очень просто:

```
Set Объектная_переменная = Nothing
```

например:

```
Set oDoc = Nothing
```

В принципе, объект можно и не удалять — он будет удален автоматически после того, как последняя объектная переменная, которая на него ссылается, уйдет за область видимости (обычно когда закончит работу процедура, в которой он используется). Однако явное удаление объектов — это "правило хорошего тона", которое позволит вам при создании серьезных приложений избежать конфликтов имен и перерасходования ресурсов.

Еще один момент, связанный с удалением объектов. Не все объекты можно удалить при помощи синтаксической конструкции `Nothing`. Некоторые объекты требуют, чтобы их удаляли из памяти специальным способом. Например, для удаления из памяти объекта приложения Word, который мы создавали в нашем примере, нужно обязательно вызвать его метод `Quit()`, иначе он выдаст сообщение об ошибке.

4.3. Методы объекта

Как правило, объект нам нужен для того, чтобы воспользоваться его методами, свойствами или событиями.

Метод — это именованный набор действий, которые может выполнять данный объект. Он может выполнять какие-либо операции, принимать и возвращать значения.

Существует три способа вызова метода.

- Самый простой способ выглядит так:

  Объект.Метод

  например:

  ```
  oDoc.Activate
  ```

  При этом не возвращаются и не принимаются никакие параметры.

- Второй способ:

  Объект.Метод параметр1 [, *параметр2*, ... , *параметрN*]

  Параметры передаются путем перечисления через запятую. Например:

  ```
  oDoc.SaveAs "C:\doc12.doc"
  ```

  В этом случае мы игнорируем то, что возвращает метод и поэтому круглые скобки не нужны.

- Третий способ:

  Переменная = Объект.Метод(*параметр1* [, *параметр2*, ... , *параметрN*])

  например:

  ```
  Dim nCent
  nCent = oApp.CentimetersToPoints(10)
  MsgBox nCent
  ```

  В этом случае значение, которое возвращает метод, присваивается переменной. При этом использовать круглые скобки для передаваемых параметров обязательно. Даже если никакие параметры не передаются, круглые скобки все равно обязательны:

  Переменная = Объект.Метод()

4.4. Свойства объекта

Свойства объекта — это возможность получения доступа к информации, которая хранится в этом объекте. Через свойства можно получить эту информацию или изменить ее.

Извлечь информацию можно при помощи синтаксиса:

Переменная = Объект.Свойство

например:

```
sName = oApp.UserName
```

Изменить информацию в объекте при помощи свойства можно так:

```
Объект.Свойство = Значение
```

например:

```
oApp.ActivePrinter = "HP LaserJet 4"
```

Значение может быть:

- обычной константой (`10` или `"HP LaserJet 4"`);
- простым выражением `(10 + 5)`;
- свойством другого объекта:

```
Объект1.Свойство = Объект2.Свойство
```

- возвращаемым значением какого-либо метода:

```
Объект1.Свойство = Объект2.Метод()
```

Конечно, значения не всех свойств можно изменять. Некоторые свойства доступны только для чтения, другие — для чтения и записи, третьи (очень редко) — только для записи.

4.5. События объекта и объявление *WithEvents*

Событие — это действие, распознаваемое объектом, для которого можно запрограммировать отклик. Например, в качестве события можно использовать открытие или закрытие документа, щелчок мыши, нажатие клавиши. События запрятаны в глубь объектов и настоятельно рекомендуется их использовать уже рассмотренным нами способом — через выбор нужного объекта и его события в окне редактора кода Visual Basic. Однако в некоторых ситуациях события для объектов не появляются в окне редактора кода (например, это справедливо для очень важного объекта `Application`). В этом случае необходимо явно объявить этот объект с событиями — при помощи ключевого слова `WithEvents`, например, так:

```
Public WithEvents App As Word.Application
```

Делается это в области объявлений модуля. После этого в редакторе кода Visual Basic появляется новый объект `App` со всеми необходимыми событиями.

Подробно работу с событиями мы рассмотрим в следующей *гл. 5*, которая будет посвящена работе с формами и графическими элементами управления: кнопками, флажками, переключателями и т. п. Нам достаточно выбрать в списке объектов (левый верхний список в окне редактора кода) нужный гра-

фический элемент, затем в списке событий (справа от списка объектов) выбрать нужное событие, и в редакторе кода будет автоматически создана специальная событийная процедура. Код, который вы в нее впишете, будет автоматически выполнен при наступлении этого события (например, при щелчке мышью на кнопке формы).

4.6. Просмотр объектов

Элементарные знания о том, как создавать объекты и использовать их свойства, методы и события, у вас уже есть. Однако может возникнуть вопрос: как найти нужный объект и как определить, какие свойства, методы и события в нем имеются?

Основной инструмент для этой цели — **Object Browser**, утилита, которая интегрирована в редактор кода VBA. Чтобы ей воспользоваться, необходимо в окне редактора кода нажать клавишу <F2> и выбрать нужную библиотеку классов. Классы показываются в левом списке как прямоугольники с разноцветными "кирпичиками", методы — как летящий зеленый предмет, свойства — как надпись, на которую указывает рука, события — значок молнии. Свои значки предусмотрены для модулей и перечислений. Если нужно просмотреть библиотеку типов, которой еще нет в списке **Object Browser**, необходимо добавить ссылку на нее через меню **Tools | References** или пункт **References** в контекстном меню самого **Object Browser**. Однако необходимо учитывать:

- для полноценной работы с **Object Browser** необходимо разбираться в объектно-ориентированном программировании. Например, если мы просмотрим класс `CommandButton` из библиотеки `MSForms` (т. е. класс кнопки на форме), то увидим там далеко не все его свойства, методы и события. Причина в том, что многие свойства, методы и события этот класс наследует от класса `Control` — общего прародителя большинства элементов управления VBA;
- при помощи **Object Browser** вы сможете узнать только названия методов, свойств и событий и получить информацию о принимаемых параметрах и возвращаемых значениях. Получить информацию о том, что делает данный метод, что возвращает свойство, когда срабатывает событие, нельзя (иногда можно догадаться по названию). Эту информацию можно найти только в справке по данной библиотеке классов.

4.7. Объектные модели

Наборы объектов, которые предназначены для выполнения задач, относящихся к одной области, называются объектными моделями. Например, в объ-

ектной модели Excel предусмотрены объекты, представляющие само приложение Excel, рабочую книгу, отдельные листы на этой рабочей книге, наборы ячеек, диаграммы и т. п. В *гл. 10—15* будут подробно разобраны объектные модели приложений Microsoft Office: Word, Excel, Access, PowerPoint, Project, Outlook. Однако при программировании на языке VBA и при создании своих собственных приложений ограничиваться только объектными моделями приложений Office совсем не обязательно.

В операционную систему Windows встроено множество других объектных моделей, применение которых может очень сильно расширить возможности ваших приложений. Далее приведен список дополнительных объектных моделей, которые встроены в Windows или в другие продукты Microsoft (об этом будет сказано отдельно), которыми я пользуюсь очень активно. Справку по большинству этих объектных моделей можно найти в MSDN (Microsoft Developer Network, официальная документация для разработчиков). Получить из нее информацию можно со страницы **www.microsoft.com/msdn**. Кроме того, MSDN можно установить на свой компьютер с компакт-дисков или DVD.

Чтобы использовать возможности этих объектных моделей в своей программе, необходимо добавить ссылку на них в ваш проект. Делается это очень просто: в окне редактора Visual Basic выберите **Tools | References** и в списке установите флажок около нужной библиотеки.

Вот перечень наиболее интересных с точки зрения применения в своих приложениях объектных моделей:

- *Windows Script Host Object Model* (wshom.exe) — эта библиотека предназначена для автоматизации работы администраторов. Она предоставляет возможность программно работать с сетью, принтерами, реестром, ярлыками, журналом событий, позволяет запускать внешние приложения и передавать в них нажатия клавиш и консольные строки и т. п. Есть на всех компьютерах с Windows 2000, XP, 2003 (в качестве необязательного компонента имеется и в Windows 98 Second Edition и Windows NT 4.0);
- *Microsoft Scripting Runtime* (scrrun.dll) — еще одна библиотека для администраторов. Главное ее богатство — очень удобный (и при этом простой) набор классов для работы с файловой системой — дисками, каталогами, файлами, содержимым текстовых файлов и т. п. Поставляется в одном наборе с Windows Script Host Object Model;
- *Microsoft ADO* (набор файлов, начинающихся с "msado") — классы для работы с базами данных. Эта библиотека будет подробно рассмотрена в *гл. 9*. Также имеется на всех без исключения компьютерах под управлением Windows 2000, XP и Windows 2003 Server (обычно сразу несколько версий);

- *Microsoft SQLDMO Object Library* (файл sqldmo.dll) — набор классов для получения полного контроля над Microsoft SQL Server (возможность производить любые административные операции, выполнять запросы и т. п.). Имеется только на тех компьютерах, на которых установлен SQL Server версий 7.0, 2000 или 2005. В SQL Server 2005 она разбита на несколько частей — SMO (SQL Server Management Objects), RMO (Replication Management Objects) и AMO (Analysis Management Objects);
- *Microsoft CDO* (версия 1.21, for NTS версия 1.2, for Windows 2000 версия 1.0; файлы olemsg.dll, cdonts.dll, cdosys.dll) — наборы классов для работы с электронной почтой. Можно использовать для создания и отправки своих почтовых сообщений, просмотра новых сообщений в почтовом ящике и т. п. Есть на всех компьютерах под управлением Windows 2000, XP и Windows 2003 Server. На компьютерах с более старыми операционными системами обычно также имеется, поскольку эта библиотека устанавливается вместе с Microsoft Office;
- *Microsoft WMI Scripting v1.1* (wbemdisp.tlb) — расширение возможностей программ через программный интерфейс WMI (Windows Management Instrumentarium). Возможности совершенно невероятные: от управления скоростью вращения вентилятора (и вообще работы с любыми устройствами, про которые знает операционная система) до установки программного обеспечения, запуска процессов на удаленном компьютере, управления службами и т. п. В этой модели реализованы очень мощные возможности работы с событиями. Например, можно выполнять какой-либо код в ответ на запуск или завершение работы программы с определенным именем на удаленном компьютере, в ответ на создание или удаление файла в каталоге, появления записи в журнале событий и т. п. Эта объектная модель (вместе со службой WMI) встроена во все компьютеры под управлением Windows 2000, XP и Windows 2003 Server. Тем, кого интересуют приемы работы с WMI, я могу посоветовать обратиться к своей статье (точнее, к главе из учебного курса), которая доступна по адресу:

  www.askit.ru/progr_admin/progr_admin_m14.htm;
- *Active Directory Scripting Interface* (adsldp.dll, wldap32.dll, adsnt.dll, adsnds.dll, adsnw.dll) — взаимодействие с объектами в каталогах Active Directory, NT, NetWare, т. е. работа с учетными записями пользователей, группами, объектами компьютеров, принтеров и т. п. Также встроена в операционные системы Windows 2000, XP и Windows 2003 Server (как в серверные, так и в пользовательские версии);
- *объектная модель Windows Explorer*. Информации о ней в справке не так много, но иногда она очень удобна для выполнения различных операций с файлами на диске;

- *объектная модель Internet Explorer* — очень мощное и удобное средство для организации взаимодействия с пользователем. Позволяет показывать пользователю Web-страницы (последовательно меняя их, можно организовать "мультфильм"), флэш-ролики, демонстрировать видео- и аудиоклипы и т. п. Эта объектная модель очень удобна и для сбора информации от пользователей при помощи скриптов и форм HTML, хотя в Microsoft Office для этой цели чаще всего используются обычные графические формы (о которых рассказывается в *гл. 5*).

Для тех, кто никогда не работал с этими объектными моделями, их применение может показаться темным лесом, но на практике все не так страшно. Они изначально создавались таким образом, чтобы с ними удобно было работать пользователям, не являющимся профессиональными программистами (например, администраторам). По опыту слушателей учебного курса "Программирование для администраторов", где рассматриваются эти объектные модели, освоить их можно очень быстро.

Задание для самостоятельной работы 4: Применение внешней объектной модели Windows Script Host в приложениях VBA

Подготовка:

Откройте Word и создайте новый документ (обычным, не программным способом). Затем нажмите клавиши <Alt>+<F11>, чтобы открыть редактор Visual Basic, в окне **Project Explorer** щелкните правой кнопкой мыши по контейнеру вашего документа (он должен называться **Project (Документ1)**) и в контекстном меню выберите **Insert | Module**. Будет создан новый стандартный модуль. При помощи меню **Insert | Procedure** создайте в нем новую процедуру с именем `WSH()`.

ЗАДАНИЕ:

1. Добавьте при помощи меню **Tools | References** в проект этого документа ссылку на библиотеку Windows Script Host Object Model.
2. В процедуре `WSH()` создайте программные объекты `WScript.Network` и `WScript.Shell` и просмотрите свойства и методы этих объектов.
3. Добавьте в процедуру `WSH()` код, который бы:
   - принимал в текстовые переменные и печатал в документе Word значения свойств `ComputerName`, `UserName` и `UserDomain` объекта `WScript.Network`;

Примечание

Код для вывода текстовой переменной в Word может выглядеть так:

```
ThisDocument.Activate
Selection.TypeText Text:=(текстовая_переменная)
```

- вызывал метод `Run()` объекта `WScript.Shell` и передавал ему единственный текстовый параметр со значением `"calc"`;
- использовал свойство `Environment` объекта `WScript.Shell` для создания коллекции текстовых переменных с информацией о переменных окружения;
- печатал в документе Word все значения текстовых переменных из этой коллекции.

Примечание

Для переменных, которые вы будете использовать для создаваемой коллекции и ее элементов, следует использовать тип `Variant`.

Ответ к заданию 4

Итоговый код процедуры `WSH()` может быть таким:

```
Public Sub WSH()
    Dim oNetwork As WshNetwork
    Dim oShell As WshShell
    Dim sComputer As String
    Dim sDomain As String
    Dim sUser As String
    Dim oColl As Variant
    Dim sEnv As Variant

    'Создаем объекты
    Set oNetwork = CreateObject("WScript.Network")
    Set oShell = CreateObject("Wscript.Shell")

    'Получаем и печатаем значения свойств объекта Wscript.Network
    sComputer = oNetwork.ComputerName
    sDomain = oNetwork.UserDomain
    sUser = oNetwork.UserName
    ThisDocument.Activate
    Selection.TypeText Text:=(sComputer & vbCrLf & sDomain & vbCrLf & _
                              sUser & vbCrLf & vbCrLf)
```

```
    'Вызываем метод Run объекта Wscript.Shell
    oShell.Run "Calc"

    'Получаем коллекцию переменных окружения
    Set oColl = oShell.Environment

    'И выводим каждый элемент этой коллекции
    For Each sEnv In oColl
        Selection.TypeText Text:=(sEnv & vbCrLf)
    Next

    'Правило хорошего тона — удаляем созданные объекты из памяти
    Set oNetwork = Nothing
    Set oShell = Nothing

End Sub
```