Джон Уокенбах. Excel 2010. Профессиональное программирование на VBA

Джон Уокенбах, один из крупнейших специалистов по Excel, подробно рассказывает о возможностях языка VBA (Visual Basic for Applications). Книга будет полезна, как начинающим, так и опытным программистам VBA. На момент публикации заметки вышла следующая версия книги для Excel 2013.

Джон Уокенбах. Excel 2010. Профессиональное программирование на VBA. – М.: Диалектика, 2010 – 944 с.

Купить книгу в Ozon или Лабиринте

Часть I. Введение в Excel

Часть V. Профессиональные методы программирования

Глава 16. Разработка утилит Excel с помощью VBA

Глава 17. Работа со сводными таблицами

Глава 18. Управление диаграммами

Глава 19. Концепция событий Excel

Глава 20. Взаимодействие с другими приложениями

Глава 21. Создание и использование надстроек

Часть VI. Разработка приложений

Глава 22. Работа с лентой

Глава 23. Работа с контекстными меню

Глава 24. Предоставление справки в приложениях

Глава 25. Разработка пользовательских приложений

Часть VII. Дополнительные темы

Глава 26. Вопросы совместимости

Глава 27. Управление файлами с помощью VBA

Глава 28. Управление компонентами Visual Basic

Глава 29. Модули классов

Глава 30. Работа с цветом

Глава 31. Часто задаваемые вопросы о программировании в Excel

* * *

Введение в Excel. В первой части книги Уокенбах знакомит с основами Excel. Я привожу лишь пару трюков, с которыми не был знаком ранее. Например, если нужно скопировать содержимое (значение или формулу) ячейки в находящуюся ниже активную ячейку, нажмите <Ctrl+D>.

Можно щелкнуть правой кнопкой мыши на строке состояния и выбрать тип просматриваемой информации (рис. 1).

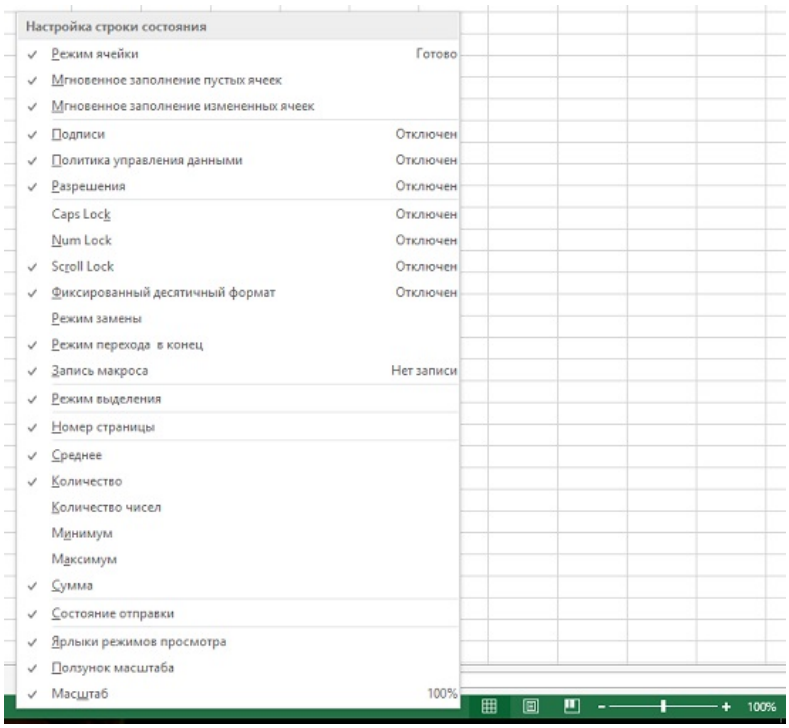

Рис. 1. Настройка информации, показываемой в строке состояния

Введение в VBA

Секрет использования VBA заключается в правильном понимании объектной модели в каждом отдельном приложении. Следует отметить, что VBA всего лишь управляет объектами, а каждый программный продукт (Excel, Word, Access, PowerPoint и т.п.) имеет свою объектную модель. Приложением можно управлять программным образом только с помощью объектов, которые представлены в этом приложении.

Объекты имеют свойства. *Свойство* может рассматриваться как настройка объекта. Например, объект диапазона имеет свойства Value и Name. Сослаться на свойство можно, записав объект и свойство, разделенные точкой. Например, сослаться на значение в ячейке A1 листа Лист1 можно так: `Worksheets("Лист1").Range("A1").Value`

Все объекты имеют методы. *Метод* — это действие, которое выполняется над объектом. Например, один из методов объекта Range называется `ClearContents`. Например, для удаления содержимого ячейки A1 с активного рабочего листа используется следующая конструкция: `Range ("A1").ClearContents.`

Некоторые объекты распознают определенные *события*, причем можно создать код VBA, который вызывается при наступлении определенных событий. Например, открытие рабочей книги приведет к вызову события `Workbook_Open`. Изменение ячейки в рабочей книге приведет к вызову события `Worksheet_Change`.

Все программы на языке VBA создаются с помощью редактора Visual Basic (Visual Basic Editor — VBE; подробнее см. Настройка среды Visual Basic Editor).

На ленте Excel изначально вкладка *Разработчик* не отображается. Чтобы ее отразить:

1. Щелкните правой кнопкой мыши на ленте и в контекстном меню выберите *Настройка ленты*.
2. Откроется окно *Параметры Excel*, раздел *Настройка ленты*.
3. В правой части окна установите флажок возле позиции *Разработчик*.
4. Кликните Ok.

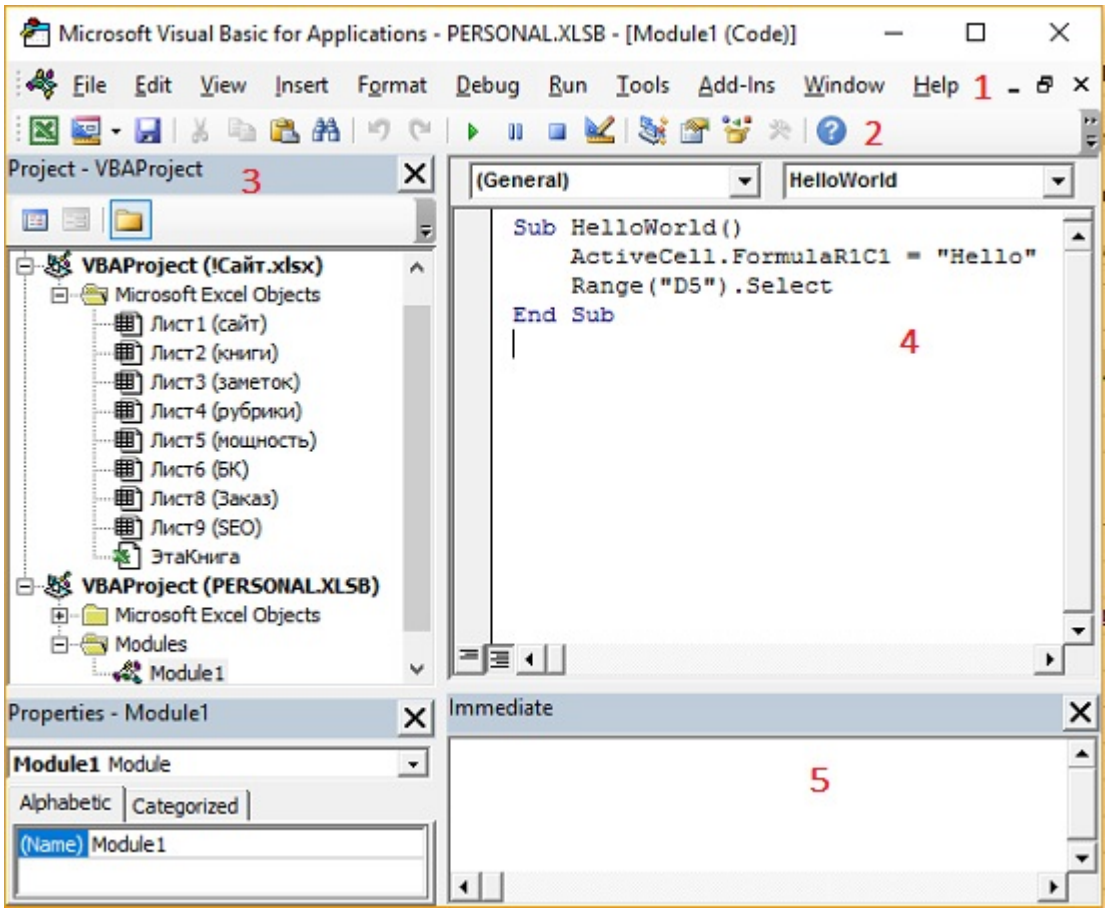

Рис. 2. Окно редактора Visual Basic Editor

Окно VBE (рис. 2) включает (1) строку меню, (2) стандартную панель инструментов, которая по умолчанию находится под строкой меню (это одна из шести панелей инструментов, используемых в VBE), (3) окно Project Explorer, в котором отображается древовидная структура всех открытых в данный момент в Excel рабочих книг, (если окно Project Explorer не отображено, нажмите <Ctrl+R>), (4) окно кода VBA (чтобы просмотреть код объекта, дважды щелкните мышью на этом объекте в окне Project Explorer), (5) окно отладки, предназначенное для непосредственного выполнения операторов VBA, тестирования операторов и отладки кода (если окна отладки нет на экране, нажмите <Ctrl+G>).

Чтобы добавить в проект новый модуль VBA, выделите название проекта в окне Project Explorer и выберите команду *Insert –> Module* (подробнее о VBE см. Настройка среды Visual Basic Editor).

Отдельная инструкция в VBA может иметь произвольную длину. Однако для обеспечения удобочитаемости кода длинные инструкции лучше разбить на две или более строк. Для этого следует в конце строки ввести пробел и символ подчеркивания, а затем нажать <Enter> и продолжить инструкцию в следующей строке.

Одним из способов создания кода модуля VBA является запись последовательности действий с помощью специальной функции записи макросов Excel. Запись действий — это наилучший способ изучить VBA. Если у вас возникают проблемы с введением кода, воспользуйтесь функцией записи действий. Даже если вы получаете совсем не то, чего ожидали, результирующий код укажет правильное направление.

Средство записи макросов

Команда записи макросов не может генерировать код, который включает циклические структуры (т.е. повторяющиеся операторы), а также присваивает переменные, выполняет условные операторы, отображает диалоговые окна и т.д. Средство записи макросов всегда создает процедуры типа Sub. С ее помощью невозможно создать процедуру типа Function.

При записи последовательности действий Excel обычно использует абсолютные ссылки на ячейки (подробнее см. Относительные, абсолютные и смешанные ссылки на ячейки в Excel). Для реализации относительной формы записи воспользуйтесь командой *Разработчик –> Код –> Относительные ссылки*. Эта кнопка играет роль переключателя. Как только она окрашивается в другой цвет, это служит признаком записи кода с относительными ссылками. Если кнопка окрашена в стандартный цвет, записывается код с абсолютными ссылками. Метод записи может быть изменен в любой момент, даже в середине процесса записи.

По умолчанию Excel помещает записанный макрос в модуль активной рабочей книги. По желанию можно записать его в личной книге макросов, имеющей имя Personal.xlsb. Этот файл не существует, пока не будет записан первый макрос (подробнее см. Создание личной книги макросов).

На заметку. Находясь в окне кода VBE, переместите курсор в любое место процедуры и нажмите клавишу <F5> для ее выполнения. Кроме того, убедитесь, что код выполняется в правильном контексте. Например, если оператор ссылается на лист `Лист1`, удостоверьтесь, что в рабочей книге действительно есть лист с названием `Лист1`. Код может состоять из единственного оператора. В этом случае используется *Окно отладки* VBE (Immediate). Оно применяется для «немедленного» выполнения операторов — без создания процедуры. Введите в окно отладки оператор VBA и нажмите клавишу <Enter>. Чтобы проверить выражение в окне отладки, введите перед ним знак вопроса (?) — символ вызова команды *Print*. Например, в окно отладки можно ввести следующий код:

```
? Range("A1").Value
```

Результат выполнения выражения отображается в следующей строке окна отладки.

Об объектах и коллекциях

Коллекция — это группа объектов одного класса (и сама коллекция также является объектом). Например, `Workbooks` — это коллекция всех открытых в данный момент объектов `Workbook`. `Worksheets` – коллекция всех объектов `Worksheet`, которые содержатся в конкретном объекте `Workbook`. Можно одновременно управлять целой коллекцией объектов или отдельным

объектом этой коллекции. Чтобы сослаться на один объект из коллекции, введите название или номер объекта в скобках после названия коллекции.

`Worksheets("Лист1")` или `Worksheets(1)`

Если вы обращаетесь к объекту в VBA, то в ссылке на него вводятся названия всех расположенных выше в иерархической структуре объектов, разделенных точкой (оператор-точка). Что делать, если в Excel открыты две рабочие книги и в обеих имеется рабочий лист с названием `Лист1`? В этом случае в ссылке необходимо указать контейнер требуемого объекта.

```
Workbooks("Книга1").Worksheets("Лист1")
```

В Excel отсутствует объект отдельной ячейки. Отдельная ячейка представляет собой объект `Range`, состоящий из одного элемента. Например, `Range("A1")`.

Все объекты обладают свойствами. Например, объект `Range` (Диапазон) обладает свойством `Value` (Значение). Следующая процедура, использующая функцию VBA `MsgBox` для отображения окна, в котором представлено значение ячейки A1 листа `Лист1` активной рабочей книги:

```
Sub ShowValue()
     Msgbox Worksheets("Лист1").Range("A1").Value
End Sub
```

Многие объекты имеют свойство, заданное по умолчанию. Для объекта `Range` свойством по умолчанию является `Value`. Следовательно, выражение `.Value` в предыдущем коде можно опустить, и ничего не произойдет. Однако лучше включать ссылку на свойство, даже если оно используется по умолчанию.

Свойство Formula доступно в режиме «только для чтения», поэтому определить формулу можно с помощью кода VBA.

```
Range("D12").Formula = "=RAND()*100"
```

Многие методы получают аргументы, определяющие выполняемые над объектом действия более детально. Например, обратимся к методу `Protect` объекта рабочей книги. Метод `Protect` имеет три аргумента: пароль, структуру и окна. Эти аргументы соответствуют параметрам диалогового окна *Защита книги* (рис. 3).

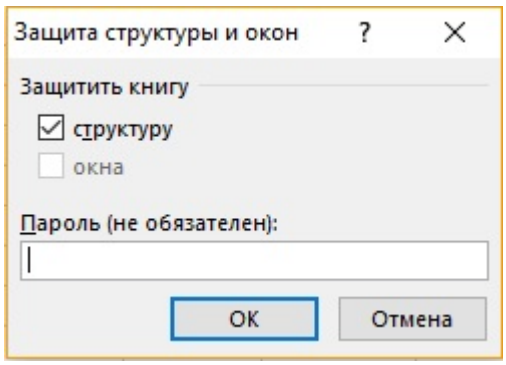

Рис. 3. Диалоговое окно Excel *Защита структуры и окон*

Если нужно защитить рабочую книгу `MyBook.xlsx`, используйте следующий оператор:

```
Workbooks("MyBook.xlsx").Protect "xyzzy", True, False
```

В данном случае рабочая книга защищена паролем (аргумент 1). Также защищена структура (аргумент 2), но не окна (аргумент 3). Если вы не хотите применять пароль, используйте следующий оператор:

```
Workbooks ("MyBook.xlsx").Protect , True, False
```

Обратите внимание, что первый аргумент пропущен, а его место обозначено запятой. Существует и другой подход (причем в этом случае программу будет удобнее читать) — использование именованных аргументов:

```
Workbooks("MyBook.xlsx").Protect Structure:=True, Windows:=False
```

Для свойств (и методов), использующих аргументы, последние указываются в скобках. Например, свойство Address объекта Range имеет пять аргументов — все необязательные. Показанный ниже оператор некорректен, поскольку пропущены скобки.

```
MsgBox Range("Al").Address False ' Некорректно
```

Правильный синтаксис для этого оператора выглядит так:

```
MsgBox Range("Al").Address(False)
```

Кроме того, оператор можно записать с помощью именованного аргумента.

```
MsgBox Range("Al").Address(rowAbsolute:=False)
```

Самый простой способ получить справку о конкретном объекте, свойстве или методе— ввести ключевое слово в окно кода и нажать клавишу <F1>. Если ключевое слово трактуется неоднозначно, появляется диалоговое окно выбора темы.

Управление свойствами сначала кажется сложной задачей, потому что некоторые из них возвращают объекты. Предположим, необходимо определить цвет фона конкретного примечания на листе Лист1. Просмотрев список свойств объекта Comment, вы не найдете ничего, что относится к определению цвета. Выполните следующие действия.

1. Используйте свойство Shape объекта Comment, возвращающее объект Shape, который содержится в примечании.
2. Используйте свойство Fill объекта Shape, возвращающее объект FillFormat.
3. Используйте свойство ForeColor объекта FillFormat, возвращающее объект ColorFormat.
4. Используйте свойство RGB (или свойство SchemeColor) объекта ColorFormat, чтобы задать цвет.

Иначе говоря, получение цвета фона объекта Comment связано с доступом к другим объектам, которые в нем содержатся. Ниже описана иерархия задействованных объектов.

```
Application (Excel)
      Объект Workbook
            Объект Worksheet
                  Объект Comment
                        Объект Shape
                              Объект FillFormat
                                    Объект ColorFormat
```

Данная иерархия объектов выглядит весьма запутанной! Но в качестве примера «элегантности» VBA посмотрите, как код для изменения цвета примечания можно записать с помощью одного оператора.

```
Worksheets("Лист1").Comments(l).Shape.Fill. _
      ForeColor.RGB = RGB(0,255,0)
```

Некоторые полезные свойства объекта Application

При работе в Excel активной одновременно может быть только одна рабочая книга. И если вы управляете рабочим листом, то активна на нем только одна ячейка (даже если выделен диапазон). Объект Application обладает рядом свойств (рис. 4; подробнее см. здесь), например, свойством ActiveCell, возвращающим ссылку на активную ячейку. Следующая инструкция присваивает значение 1 активной ячейке:

```
ActiveCell.Value = 1
```

Свойство	Возвращаемый объект
ActiveCell	Активная ячейка
ActiveChart	Активный лист диаграмм или объект диаграммы на рабочем листе (ChartObject). Если диаграмма не активна, то свойство равно Nothing
ActiveSheet	Активный лист (рабочий лист или лист диаграммы)
ActiveWindow	Активное окно
ActiveWorkbook	Активная рабочая книга
Selection	Выделенный объект (объект Range, Shape, ChartObject и т.д.)
ThisWorkbook	Рабочая книга, содержащая выполняемую процедуру VBA

Рис. 4. Некоторые свойства объекта Application

Работа с объектами Range

Объект Range содержится в объекте Worksheet и состоит из одной ячейки или диапазона ячеек на отдельном рабочем листе. Существует три способа задания ссылки на объекты Range:

- свойство Range объекта класса Worksheet или Range;
- свойство Cells объекта Worksheet;
- свойство Offset объекта Range.

Будьте осторожны при работе с объединенными ячейками. Если вы планируете писать VBA-код, ячейки на листе Excel лучше не объединять.

Свойство Range поддерживает имена, определенные в рабочих книгах. Поэтому, если ячейка называется Ввод, то для ввода значения в нее может использоваться следующий оператор:

```
Worksheets("Лист1").Range("Ввод").Value = 100
```

В следующем примере в диапазон из двадцати ячеек на активном листе вводится одинаковое значение. Если активный лист не является рабочим, то отображается сообщение об ошибке.

```
ActiveSheet.Range("A1:B10").Value = 2 или Range("A1", "B10") = 2
```

В следующем примере применяется оператор пересечения Excel (пробел). Пересечением является одна ячейка — C6. Следовательно, данный оператор вводит значение 3 в ячейку C6.

```
Range("C1:C10 A6:E6") = 3
```

До сих пор использовалось свойство Range объекта Worksheet. Вы также можете использовать свойство Range объекта Range. Ниже показан пример, в котором объектом Range является активная ячейка. Другими словами, полученная ссылка является относительной для верхнего левого угла объекта Range. Таким образом, следующий оператор вводит значение 5 в ячейку, расположенную справа внизу от активной ячейки.

```
ActiveCell.Range("B2") = 5
```

Свойство Cells также может использоваться в объектах Worksheet и Range. Имеется три варианта синтаксиса:

```
объект.Cells(номер_строки, номер_столбца)
объект.Cells(номер_строки)
объект.Cells
```

Например, следующий оператор введет значение 7 в ячейку D3 (пересечение строки 3 и столбца 4) активного рабочего листа:

```
ActiveSheet.Cells(3,4) = 7
```

Можно также использовать свойство Cells объекта Range. При этом объект Range, который возвращается свойством Cells, задается относительно левой верхней ячейки диапазона Range, на который мы ссылаемся. Следующая инструкция вводит значение 5 в ячейку, которая находится под активной:

```
ActiveCell.Cells(2,1) = 5
```

Еще один синтаксис свойства Cells использует один аргумент, который задается в диапазоне от 1 до 17 179 869 184. Второе число равно количеству ячеек на рабочем листе Excel. Ячейки нумеруются, начиная с A1 вправо, затем вниз и вправо вдоль следующей строки. Например, ячейка 16384 — это XFD1, а 16385 — A2.

Для отображения значения последней ячейки рабочего листа воспользуйтесь следующим оператором:

```
MsgBox ActiveSheet.Cells(17179869184)
```

Этот синтаксис можно использовать и с объектом Range. В таком случае будет получена ячейка по отношению к указанному объекту Range. Например, если объект Range — это диапазон A1:D10 (40 ячеек), то свойство Cells может иметь аргумент от 1 до 40 и возвращать одну из ячеек объекта Range. В следующем примере значение 2000 вводится в ячейку A2, так как A2 является пятой ячейкой (считая сверху направо и вниз) в указанном диапазоне:

```
Range("A1:D10").Cells(5) = 2000
```

Третий пример синтаксиса свойства Cells возвращает все ячейки на указанном рабочем листе. В приведенном ниже примере будет удалено содержимое каждой ячейки на рабочем листе:

```
ActiveSheet.Cells.ClearContents
```

Получение информации из ячейки

Наиболее часто используются следующие свойства.

Свойство Formula возвращает формулу в случае, когда она находится в ячейке. Если ячейка не содержит формулу, возвращается находящееся в ней значение. Свойство Formula может как считываться, так и изменяться пользователем.

Свойство Value возвращает исходное неотформатированное значение ячейки. Это свойство может считываться и изменяться пользователем.

Свойство Text возвращает текст, отображаемый в ячейке. Если ячейка содержит числовое значение, это свойство включает все имеющееся форматирование, включая запятые и денежные символы. Свойство Text предназначено только для чтения.

Свойство Offset

Свойство Offset также возвращает объект Range. Свойство Offset применяется только к объекту Range. Данное свойство использует единственный синтаксис:

```
объект.Offset(смещение_строки, смещение_столбца)
```

Два аргумента свойства Offset соответствуют смещению относительно левой верхней ячейки указанного диапазона Range. Эти аргументы могут быть положительными (сдвиг вниз или вправо), отрицательными (вверх или влево) или нулевыми. В приведенном ниже примере значение 12 вводится в ячейку, которая находится под активной:

```
ActiveCell.Offset(1,0).Value = 12
```

Свойство Offset особо эффективно при использовании переменных в цикле.

Узнайте больше об объектах и свойствах

Используйте браузер объектов – *Object Browser*, который можно открыть, например, нажав <F2> (рис. 5).

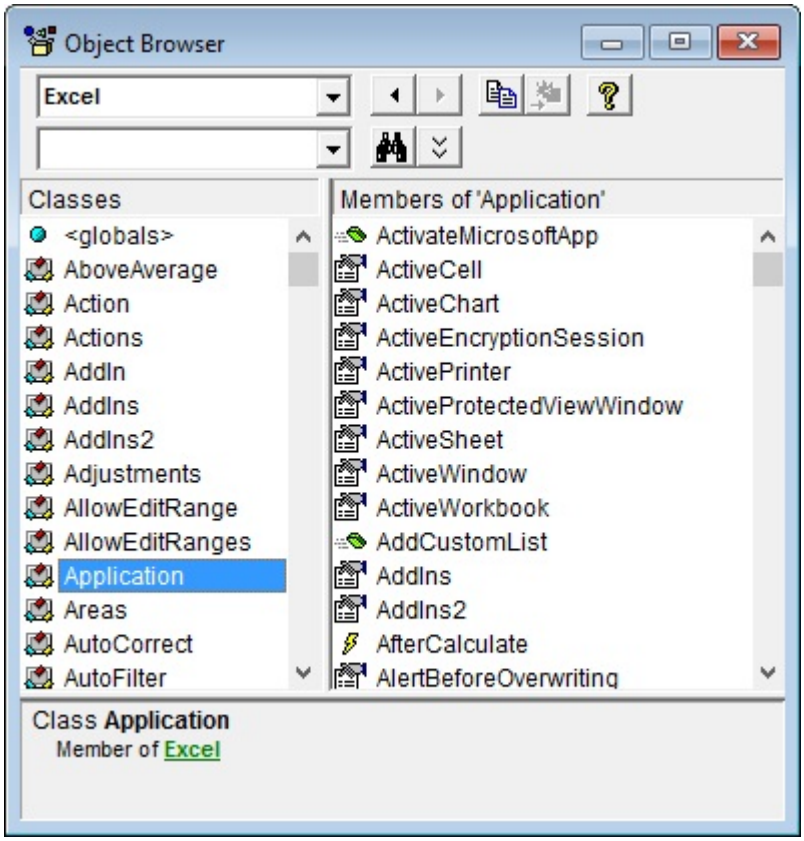

Рис. 5. В окне *Object Browser* можно найти много полезных сведений об объектах

Раскрывающийся список в левом верхнем углу окна содержит перечень всех библиотек объектов, к которым у вас есть доступ. Вас, в первую очередь интересует собственно Excel. Ваш выбор в этом списке определяет, что отображается в разделе *Classes* (Классы), а выбор в разделе Classes обусловливает появление определенных компонентов в поле *Members of* (Включены в).

После выбора библиотеки можно осуществить поиск конкретной строки текста, чтобы получить список свойств и методов, содержащих данный текст. Это можно сделать, введя текст во втором раскрывающемся списке и щелкнув на значке с изображением бинокля. Предположим, вы работаете над проектом, обрабатывающим примечания в ячейках (рис. 6).

1. Выберите интересующую вас библиотеку. Если вы не знаете, какую именно библиотеку выбрать, укажите вариант <All Libraries>.
2. Введите *Comment* в раскрывающемся списке под списком библиотек.
3. Щелкните на значке в виде бинокля, чтобы начать поиск текста.

В области *Search Results* (Результаты поиска) отображается текст, соответствующий фрагменту для поиска. Выберите один объект, чтобы отобразить его классы в разделе *Classes*. Укажите класс, чтобы отобразить его члены (свойства, методы и константы). Обратите внимание на нижнюю часть окна, где приведена дополнительная информация об объекте. Можно нажать <F1>, чтобы перейти к необходимому разделу справочной системы в Интернете.

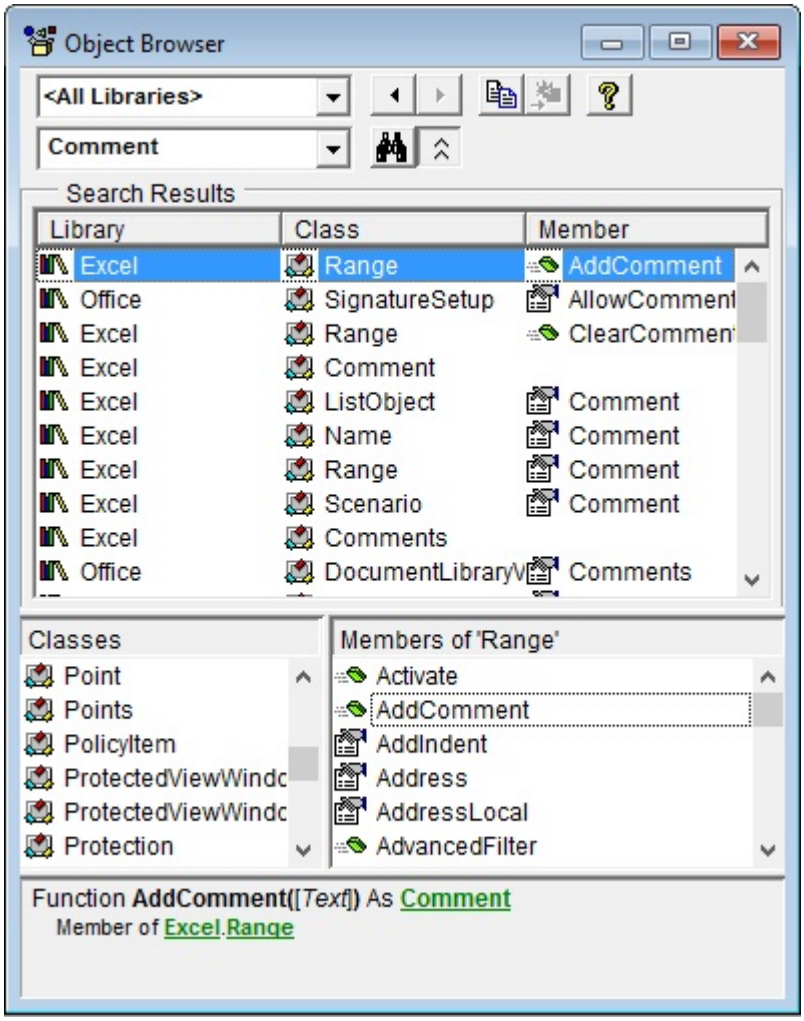

Рис. 6. Результаты поиска *Comment* среди объектов, свойств и методов

Экспериментируйте с окном отладки. Оно используется для тестирования операторов и проверки разных выражений VBA. Рекомендуется всегда отображать окно отладки на экране, так как оно часто используется для проверки выражений и при отладке кода.