Глава 9. Автоматизация Таблиц с помощью VBA

Это продолжение перевода книги Зак Барресс и Кевин Джонс. Таблицы Excel: Полное руководство для создания, использования и автоматизации списков и таблиц (Excel Tables: A Complete Guide for Creating, Using and Automating Lists and Tables by Zack Barresse and Kevin Jones. Published by: Holy Macro! Books. First printing: July 2014. – 161 p.). Visual Basic for Applications (VBA) – это язык программирования, который можно использовать для расширения стандартных возможностей Excel. VBA позволяет автоматизировать сложные или повторяющиеся задачи. Например, VBA можно использовать для форматирования листа, получаемого ежедневно из внешнего источника, извлечения данных с веб-страницы раз в неделю или построения сложной пользовательской функции листа.

Предыдущая глава Содержание Следующая глава

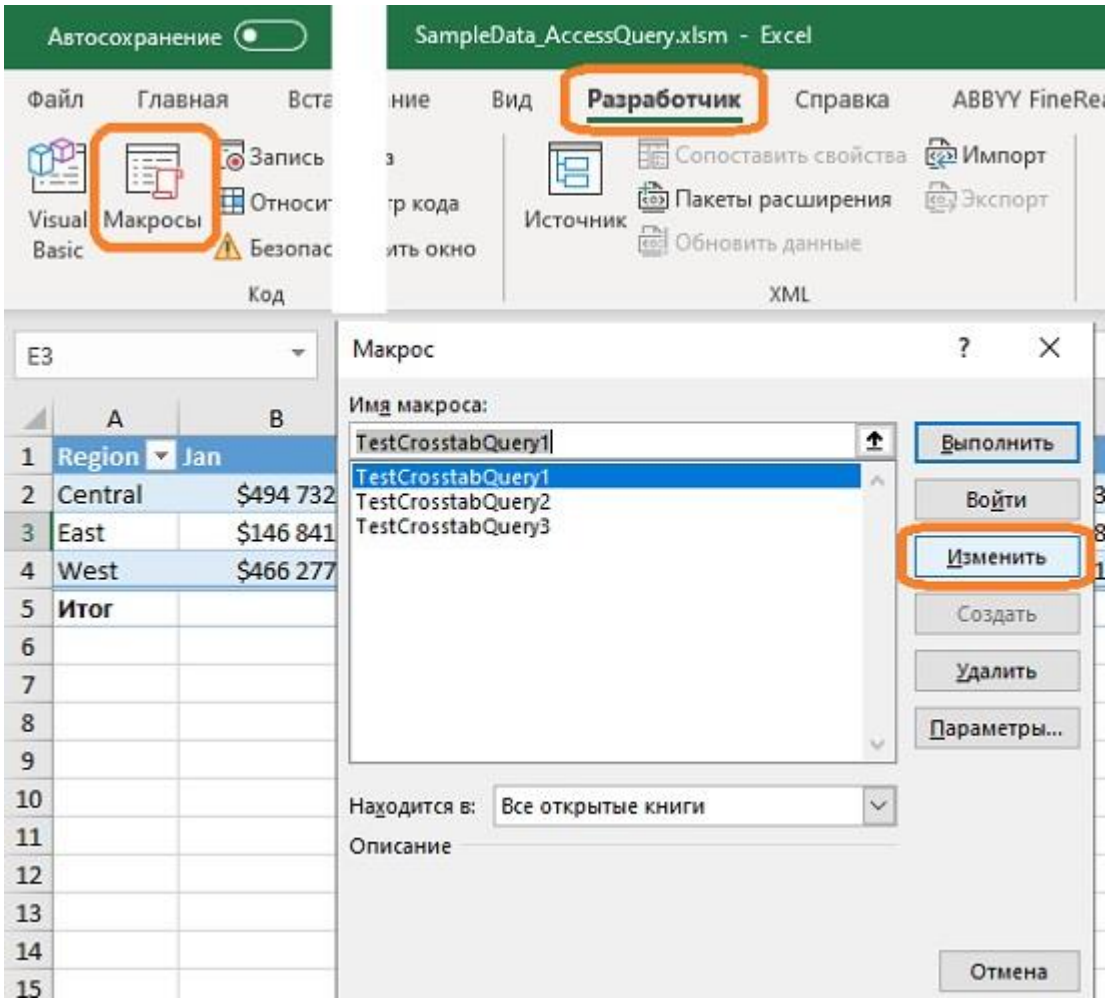

Рис. 9.1. Редактирование макросов

Вот некоторые примеры автоматизации Таблиц Excel:

- Вы создаете приложение для использования менее опытными пользователями, и хотите заблокировать Таблицу, предоставляя несколько простых функций для добавления, редактирования и удаления записей.
- Вы хотите предоставить функциональные возможности для редактирования групп записей в пользовательской форме, например, транзакции по одному заказу (аванс, финальная оплата, фрахт, пошлина и т.д.).
- Ваша Таблица поддерживает другие функции книги, такие как запись информации о событиях в журнал, и вы хотите вставлять строки в таблицу, обновлять существующие табличные данные, а также находить и извлекать строки из Таблицы.
- Вы хотите провести аудит Таблицы и отобразить ошибки, которые необходимо исправить.

Для работы с этой главой лучше всего, если у вас есть начальный уровень программирования, вы знакомы с объектной моделью и хотите расширить функциональность Таблиц Excel (см. Джон

Уокенбах. Excel 2010. Профессиональное программирование на VBA). Среда VBA не русифицирована.

Один из самых простых способов начать работать с VBA – это использовать запись макросов. При этом действия, которые вы совершаете в Excel, записываются в среде VBA в виде инструкций. Не забудьте выключить запись макросов, когда автоматизированные действия будут завершены. Вы можете просматривать и редактировать код VBA, пройдя по меню *Разработчик –> Макросы*, выбрать имя макроса и нажать кнопку *Изменить* (рис. 9.1). Код, сгенерированный при записи макросов, не самый эффективный, и вы часто можете улучшить его с помощью редактирования.

Если вкладка *Разработчик* не видна, включите ее, пройдя по меню *Файл –> Параметры –> Настроить ленту,* и установив флажок на вкладке *Разработчик*.

VBA, Excel и объекты

Этот раздел можно пропустить, если у вас уже есть опыт работы с объектами в среде Excel VBA.

В среде VBA приложение Excel представляет объекты, к которым можно получить доступ, в виде объектной модели Excel. Основные объекты Excel:

- *Application* – Приложение Excel.
- *Workbooks* – коллекция всех книг, открытых в данный момент в приложении Excel. В общем случае коллекция – набор объектов. В этом случае коллекция Workbooks представляет собой набор отдельных объектов Workbook.
- *Workbook* – одна книга.
- *Worksheets* – коллекция всех листов в рабочей книге.
- *Worksheet* – отдельный рабочий лист или вкладка в рабочей книге.
- *Cells* – совокупность всех ячеек на листе.
- *Range* – набор из одной или нескольких ячеек на одном листе. Клетки могут быть прерывистыми. Любая ссылка даже на одну ячейку является объектом Range.

Объекты имеют свойства. Каждый объект принадлежит родительскому объекту. Например, родителем объекта Workbook является объект Application, а родителем объекта Cell – объект Worksheet. Одним из исключений является объект Application, который не имеет родителя; это объект самого высокого уровня, доступный при просмотре объектной модели Excel. При описании контекста объекта или метода часто используются следующие термины:

- *Parent* – родительский объект. Например, родительский объект для Cell – это Worksheet.
- *Member* – член родительского объекта. Например, член объекта Worksheet – объект Cell. Член также может быть методом, который выполняет действие.

Когда вы ссылаетесь на объект, вы должны начать с самого высокого уровня. Чтобы найти каждый подчиненный объект, введите родительский объект, за которым следует точка, а затем дочерний элемент. Например, чтобы сослаться на значение ячейки A1 на листе *Sheet1* в книге *My Workbook.xlsx*, используйте следующий синтаксис:

```
Application.Workbooks("My Workbook.xlsm").Worksheets("Sheet1").Cells(1,1).Value
```

Ссылки в VBA как правило используют синтаксис R1C1, а не A1.

Объектная модель Excel предоставляет объекты по умолчанию в зависимости от того, какой элемент приложения в данный момент активен. Например, следующий синтаксис ссылается на ячейку A1 на листе, активном в момент выполнения кода:

```
Cells(1,1).Value
```

Хотя этот синтаксис работает, он не считается хорошей практикой. Мы рекомендуем использовать объект *ActiveSheet,* если вы действительно намеревались ссылаться на активный лист:

```
Application.ActiveSheet.Cells(1,1).Value
```

Единственный объект, который подразумевается во всей среде Excel VBA, – это объект Application, и его можно без проблем опустить. Поэтому следующие ссылки не являются неоднозначными в любом месте среды Excel VBA:

```
Workbooks("My Workbook.xlsm").Worksheets("Sheet1").Cells(1,1).Value
```

```
ActiveSheet.Cells(1,1).Value
```

Вы можете назначить ссылку на любой объект переменной, если эта переменная имеет тот же тип, что и объект, или определена как универсальный тип объекта. Основная причина для этого – удобство. Если вы ссылаетесь на объект повторно, то выделение переменной для ссылки на этот объект может привести к уменьшению объема кода, который будет легче читать и поддерживать. Например, вы можете назначить ссылку на *Sheet1* переменной:

```
Dim TargetWorksheet As Worksheet
Set TargetWorksheet = Workbooks("My Workbook.xlsm").Worksheets("Sheet1")
```

Теперь ссылка на A1 может быть записана в виде:

```
TargetWorksheet.Cells(1,1).Value
```

Ключевое слово *Set* присваивает ссылку. VBA требует использования ключевого слова Set при назначении ссылок на объекты. Кроме того, переменная является ссылкой на объект, а не копией объекта. Вы можете иметь любое количество переменных, содержащих ссылку на один и тот же экземпляр объекта.

Следует различать объекты, экземпляры объектов и ссылки на объекты. Объект включает в себя объектную модель (свойства и методы, которые ему принадлежат) и код, который управляет его поведением. Примером объекта является *Worksheet*. Экземпляр объекта является конкретным экземпляром этого объекта и включает в себя данные и значения свойств, связанные с этим экземпляром объекта. Листы *Sheet1* и *Sheet2* примеры экземпляров объектов. Переменная, ссылающаяся на объект, содержит ссылку на объект.

При определении переменных, ссылающихся на объекты, необходимо определить переменную того же типа, что и объект, на который ссылаются, или присвоить универсальный тип объекта. В общем случае универсальный тип объекта следует использовать только в том случае, если этой переменной будут присвоены ссылки на другие объекты различных типов; при использовании универсального типа объекта редактор VBA не может помочь вам с интеллектуальной подсказкой IntelliSense (IntelliSense представляет список свойств и методов объекта при вводе ссылки на объект, за которой следует точка.)

В дальнейшем описании мы используем следующие объекты:

- *ThisWorkbook* – рабочая книга, в которой выполняется код. Это удобно, когда единственная книга, на которую ссылаются, является той, в которой находится код. С этой рабочей книгой работать легче, чем с рабочими книгами (*My Workbook.xlsm*), особенно когда имя рабочей книги может меняться.
- *Me* – когда код находится в модуле кода листа, *Me* – это удобный способ ссылаться на объект листа, в котором находится код.

Excel Online открывает файлы с VBA (при просмотре в браузере), но не выполняет код. Код игнорируется при открытии книги в приложении Excel Online, но сам код сохраняется.

Объект ListObject

Excel использует объект *ListObject* для представления таблицы в объектной модели Excel. Он содержится в коллекции *ListObjects*, которая принадлежит объекту *Worksheet*. Используйте этот синтаксис для ссылки на Таблицу на листе:

```
ThisWorkbook.Worksheets("Sheet1").ListObjects("Table1")
```

Назовите Таблицу как-то иначе, чем имя по умолчанию. Это поможет вам как при написании кода, так и позже, когда вы смотрите на свой код, пытаясь понять, что вы сделали. Это также будет полезно для других пользователей, которые будут изучать ваш код.

Поскольку *ListObjects* – это набор таблиц, вы можете получить доступ к конкретной Таблице с помощью индекса:

```
ThisWorkbook.Worksheets("Sheet1").ListObjects(1)
```

Индекс (или позиция) объекта *ListObject* в коллекции *ListObjects* определяется порядком, в котором объекты ListObject были созданы на листе. Объект ListObject может быть назначен

переменной, которая была определена как тип ListObject. Объект ListObject имеет ряд свойств и методов, используемых для доступа к таблицам и управления ими.

Свойства объекта Таблица

Пять свойств представляют собой основные части таблицы. Каждое из этих свойств является объектом диапазона Range. Два дополнительных свойства или коллекции предоставляют доступ к строкам и столбцам Таблицы. Каждая коллекция предоставляет доступ ко всем объектам ListRow и ListColumn в Таблице.

Свойство Range

Свойство Range возвращает всю таблицу, включая заголовок и итоговые строки. Тип объекта – Range. Свойство не может быть установлено.

Свойство HeaderRowRange

Свойство *HeaderRowRange* возвращает строку заголовка таблицы. Тип объекта – Range. Свойство не может быть установлено. Диапазон всегда представляет собой одну строку – строку заголовка – и распространяется на все столбцы таблицы. Если строка заголовка отключена, этому свойству присваивается значение *Nothing*.

Свойство DataBodyRange

Свойство *DataBodyRange* возвращает тело таблицы. Тип объекта – Range. Свойство не может быть установлено. Диапазон – это каждая строка между заголовком и общей строкой и распространяется на все столбцы таблицы. Если таблица не содержит строк, свойство *DataBodyRange* не возвращает *Nothing* (и ListRows.Count возвращает 0). Это единственный случай, когда свойство InsertRowRange возвращает объект диапазона, который можно использовать для вставки новой строки. В этом состоянии таблица выглядит как одна строка без каких-либо значений. Как только одна ячейка в таблице получает значение, InsertRowRange получает значение Nothing, а DataBodyRange – значение строк данных в таблице.

Свойство TotalRowRange

Свойство *TotalRowRange* возвращает итоговую строку Таблицы. Тип объекта – Range. Свойство не может быть установлено. Диапазон всегда представляет собой одну строку – строку итогов – и распространяется на все столбцы Таблицы. Если строка итогов отключена, это свойство имеет значение Nothing.

Свойство InsertRowRange

Свойство *InsertRowRange* возвращает текущую строку вставки Таблицы. Тип объекта – Range. Свойство не может быть установлено. В Excel 2007 и более поздних версиях *InsertRowRange* возвращает только первую строку данных и только тогда, когда таблица не содержит никаких данных. В противном случае он ничего не возвращает и фактически бесполезен.

Свойство ListRows

Свойство *ListRows* возвращает коллекцию всех строк в таблице *DataBodyRange*. Это тип объекта ListRows, который ведет себя очень похоже на объект *Collection* и содержит коллекцию объектов *ListRow*. Свойство не может быть установлено. На строки ссылаются по индексу, отсчитываемому от единицы (one-based index).[1] В пустой таблице нет строк. Метод *Add* объекта *ListRows* используется для вставки одной новой строки за один раз.

Свойство ListColumns

Свойство ListColumns возвращает коллекцию всех столбцов таблицы. Это тип объекта ListColumns, который ведет себя очень похоже на объект *Collection* и содержит коллекцию объектов ListColumn. Свойство не может быть установлено. На столбцы ссылается one-based. Таблица всегда содержит хотя бы один столбец.

Свойства структуры Таблиц

Таких свойств несколько. Все они являются членами объекта ListObject.

---

[1] Различают три основных разновидности массивов: с отсчетом от нуля (zero-based), с отсчетом от единицы (one-based) и с отсчетом от специфического значения заданного программистом (n-based).

Свойство *ShowAutoFilter* возвращает или задает, включен ли Автофильтр. Свойство имеет логический тип. Если значение True, Автофильтр включен, False – отключен. Свойство представлено в пользовательском интерфейсе Excel, меню *Данные –> Сортировка и фильтр –> Фильтр*.

Свойство *ShowAutoFilterDropDown* возвращает или задает значение, указывающее отображается ли кнопка ниспадающего списка Автофильтра. Свойство имеет логический тип. Если значение True, кнопка отражается, False – не отражается. Если *ShowAutoFilter* имеет значение False, то свойство *ShowAutoFilterDropDown* изменить нельзя. Свойство представлено в пользовательском интерфейсе Excel, меню *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Кнопка фильтра*.

Свойство *ShowHeaders* возвращает или задает, включена ли строка заголовка таблицы. Свойство имеет логический тип. Если значение True, строка заголовка таблицы включена, False – отключена. Свойство представлено в пользовательском интерфейсе Excel, меню *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Строка заголовков*.

Свойство *ShowTotals* возвращает или задает включена ли строка итогов. Свойство имеет логический тип. Если значение True, строка итогов включена, False – отключена. Свойство представлено в пользовательском интерфейсе Excel, меню *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Строка итогов*.

Свойства стилей Таблиц

Этих свойств также несколько. Все они являются членами объекта ListObject.

Свойство *TableStyle* возвращает или задает имя стиля таблицы. Свойство имеет тип Variant. Чтобы изменить стиль таблицы, необходимо задать имя нужного стиля. При назначении стиля таблицы к таблице применяются только элементы стиля, определенные в этом стиле (подробнее см. Глава 7. Форматирование Таблиц Excel). Чтобы определить имя стиля, наведите курсор мыши на нужный стиль в галерее стилей таблиц, пока Excel не отобразит имя этого стиля:

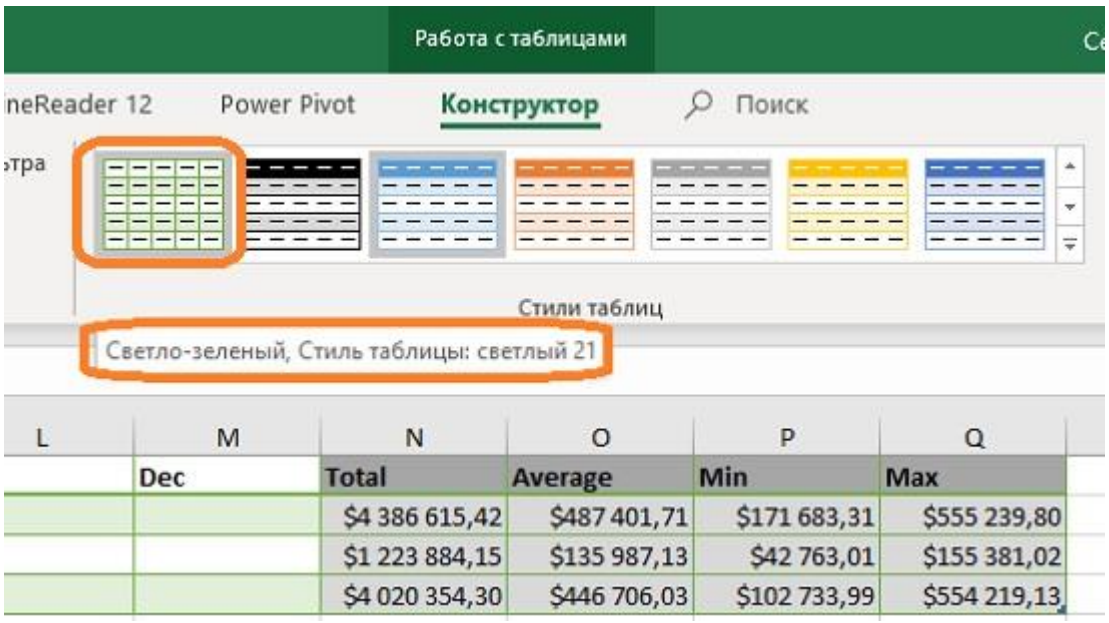

Рис. 9.2. Определение имени стиля таблицы

Чтобы присвоить это имя свойству *TableStyle*, используйте код:

```
ActiveSheet.ListObjects("qryCrosstab2").TableStyle = "TableStyleLight21"
```

Здесь *qryCrosstab2* – имя Таблицы. Внутренние имена всех встроенных стилей не содержат пробелов и английские, хотя имена, отображаемые на ленте, содержат пробелы и русифицированы. Чтобы узнать имя таблицы, которое нужно использовать в коде VBA, запустите запись макроса, присвойте стиль, завершите запись макроса, и посмотрите его код.

Свойство TableStyle представлено в пользовательском интерфейсе Excel в виде массива кнопок в группе *Работа с таблицами –> Конструктор –> Стили таблиц*.

Свойство *ShowTableStyleColumnStripes* возвращает или задает, форматируются ли нечетные столбцы таблицы иначе, чем четные столбцы. Свойство имеет логический тип. Если значение True, нечетные столбцы таблицы форматируются иначе, чем четные столбцы, как определено в заданном стиле таблицы. Нечетные столбцы форматируются с использованием параметров полосы первого столбца стиля таблицы, а четные строки форматируются с использованием параметров полосы второго столбца стиля таблицы. Если *ShowTableStyleColumnStripes* имеет значение False, столбцы таблицы не форматируются с использованием назначенного стиля таблицы. Свойство *ShowTableStyleColumnStripes* представлено в пользовательском интерфейсе Excel в виде флажка на ленте *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Чередующиеся столбцы*.

Обратите внимание, что по умолчанию количество столбцов в каждой полосе равно одному, но вы можете изменить его на большее число для элементов полосы первого столбца и полосы второго столбца независимо. Дополнительную информацию смотрите в главе 7.

Свойство *ShowTableStyleRowStripes* возвращает или задает, форматируются ли нечетные строки таблицы иначе, чем четные строки. Свойство имеет логический тип. Если значение True, нечетные строки таблицы форматируются иначе, чем четные строки, как определено в заданном стиле таблицы. Нечетные строки форматируются с использованием первой полосы строк стиля таблицы, а четные строки форматируются с использованием второй полосы строк стиля таблицы. Если свойству ShowTableStyleRowStripes присвоено значение False, строки таблицы не форматируются с использованием назначенного стиля таблицы. Это свойство эквивалентно флажку на ленте *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Чередующиеся строки*.

Свойство *ShowTableStyleFirstColumn* возвращает или задает, выделяется ли первый столбец. Свойство имеет логический тип. Если значение True, первый столбец таблицы форматируется так, как определено в настройках первого столбца назначенного стиля таблицы. Если установлено значение False, первый столбец таблицы не форматируется в соответствии с заданным стилем таблицы. Это свойство представлено в пользовательском интерфейсе Excel в виде флажка на ленте *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Первый столбец*.

Свойство *ShowTableStyleLastColumn* возвращает или задает, выделяется ли последний столбец. Свойство имеет логический тип. Если значение True, последний столбец таблицы форматируется так, как определено в настройках последнего столбца назначенного стиля таблицы. Если установлено значение False, последний столбец таблицы не форматируется в соответствии с заданным стилем таблицы. Это свойство представлено в пользовательском интерфейсе Excel в виде флажка на ленте *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Последний столбец*.

Другие свойства объекта Таблицы

В следующих разделах рассматриваются свойства объекта таблицы для использования в VBA. Они отличаются от методов.

Свойство *Active* возвращает True, если активная ячейка находится в пределах таблицы, включая заголовок и итоговые строки; в противном случае оно возвращает False. Свойство имеет логический тип. Свойство не может быть установлено.

Свойство *AlternativeText* возвращает или задает замещающий текст таблицы. Свойство имеет строковый тип. Установка этого свойства перезаписывает любое предыдущее значение. Это свойство представлено в пользовательском интерфейсе Excel в диалоговом окне Замещающий текст, доступ к которому можно получить, щелкнув правой кнопкой мыши в любом месте таблицы и пройдя по меню *Таблица –> Замещающий текст*. Это свойство отображается и редактируется в текстовом поле *Заголовок*.

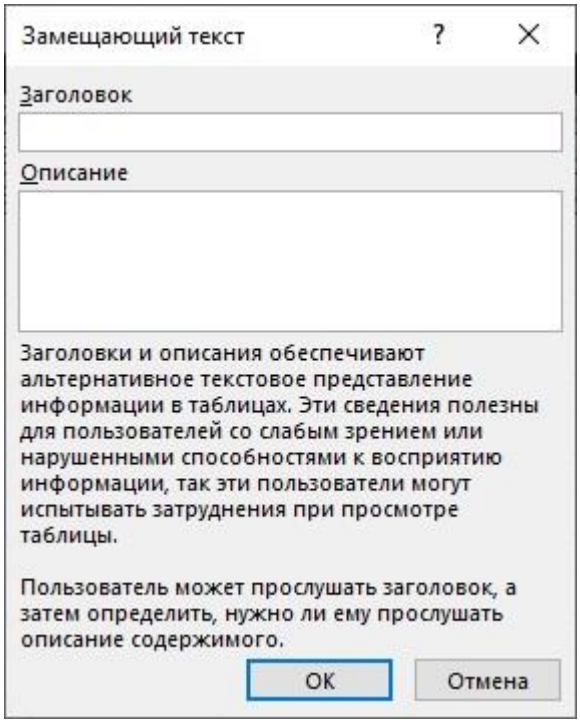

Рис. 9.3. Замещающий текст

Свойство *AutoFilter* – это объект AutoFilter со своими собственными свойствами и методами. Его можно использовать для проверки настроек Автофильтра и повторного применения или очистки настроек Автофильтра в таблице. Он не используется для установки фильтров; для этого используется метод AutoFilter объекта Range.

Свойство *Comment* возвращает или задает комментарий таблицы. Свойство имеет строковый тип. Это свойство, добавленное в Excel 2007, представлено в пользовательском интерфейсе Excel в диалоговом окне *Диспетчер имен*, доступ к которому можно получить, пройдя по меню *Формулы –> Определенные имена –> Диспетчер имен*. Комментарий (примечание) отображается в правом столбце, и вы можете изменить его, кликнув на кнопку *Изменить* (рис. 9.4). Напомним, поскольку все Таблицы именуются, их имена отражаются в окне *Диспетчер имен*.

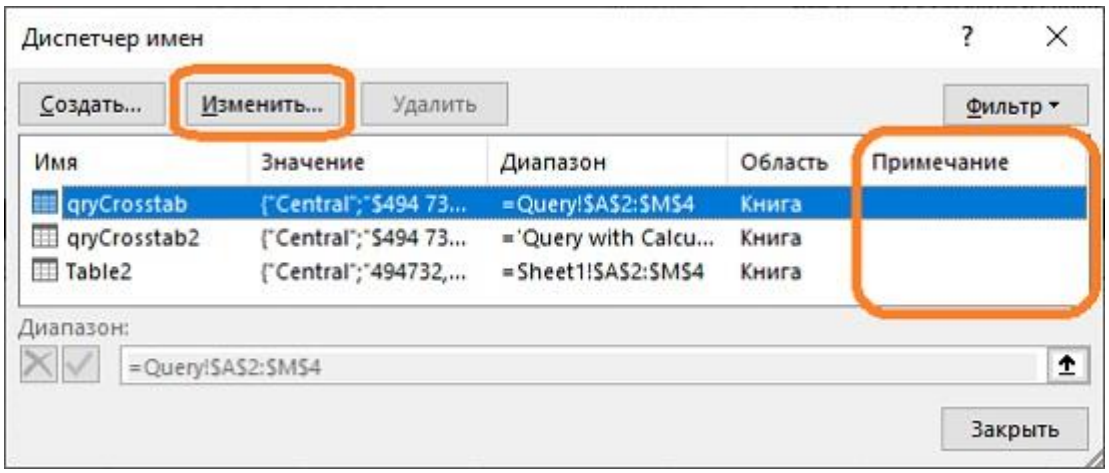

Рис. 9.4. Диспетчер имен

Свойство *DisplayName* возвращает или задает имя таблицы. Свойство имеет строковый тип. При назначении имени применяются те же ограничения, что и при изменении имени таблицы в пользовательском интерфейсе Excel; например, оно не может дублировать иное имя и не может содержать пробелов. Это свойство, добавленное в Excel 2007, ведет себя почти так же, как и свойство *Name*, но при использовании свойства DisplayName присваиваемое имя должно соответствовать ограничениям на имя таблицы, иначе возникает ошибка. Это свойство представлено в пользовательском интерфейсе Excel в виде поля ввода текста *Работа с таблицами –> Конструктор –> Свойства –> Имя таблицы*.

Свойство *Name* возвращает или задает имя таблицы. Свойство имеет строковый тип. В отличие от свойства *DisplayName*, когда вы присваиваете значение свойству *Name*, Excel изменяет имя так, чтобы оно соответствовало правилам имени таблицы. Например, он меняет пробелы на подчеркивания и, если имя уже существует, добавляет к нему символ подчеркивания, за которым следует число. Это свойство представлено в пользовательском интерфейсе Excel в виде поля ввода текста *Работа с таблицами –> Конструктор –> Свойства –> Имя таблицы*.

> Чтобы избежать проблем, используйте свойство *DisplayName* вместо свойства *Name* для присвоения имени таблице. Свойство *DisplayName* создает ошибку, если имя является незаконным или уже определено в другом месте. С другой стороны, Excel будет искажать значение, присвоенное свойству *Name*, чтобы сделать его законным, и таким образом имя может оказаться не совсем тем, что вы намеревались.

Свойство *Parent* возвращает родителя таблицы. Свойство имеет тип объекта и всегда возвращает Worksheet. Свойство не может быть установлено.

Свойство *QueryTable* возвращает объект QueryTable, который ссылается на сервер списков. Это тип объекта QueryTable. Свойство не может быть установлено. Объект QueryTable предоставляет свойства и методы, позволяющие управлять таблицей. Следующий код публикует Таблицу на сервере SharePoint и называет опубликованный список *Register*. Затем он восстанавливает объект QueryTable для таблицы и устанавливает свойству MaintainConnection Таблицы значение True:

```
Dim Table As ListObject
Dim QueryTable As QueryTable
Dim PublishTarget(4) As String
Dim ConnectionString As String
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
PublishTarget(0) = "0"
PublishTarget(1) = "http://myserver/myproject"
PublishTarget(2) = "1"
PublishTarget(3) = "Register"
ConnectionString = Table.Publish(PublishTarget, True)
Set QueryTable = Table.QueryTable QueryTable.MaintainConnection = True
```

Свойство *SharePointURL* возвращает URL-адрес списка SharePoint. Свойство имеет строковый тип. Это свойство устанавливается при создании или обслуживании подключения к SharePoint и не может быть изменено. Это свойство представлено в пользовательском интерфейсе Excel в виде кнопки *Работа с таблицами –> Конструктор –> Данные из внешней таблицы –> Экспорт –> Экспорт таблицы в список SharePoint*. Следующий код публикует существующую таблицу на сервере SharePoint с помощью *SharePointURL* и называет опубликованный список *Register*:

```
Dim Table As ListObject
Dim QueryTable As QueryTable
Dim PublishTarget(4) As String
Dim ConnectionString As String
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
PublishTarget(0) = "0"
PublishTarget(1) = Table.SharePointURL PublishTarget(2) = "1"
PublishTarget(3) = "Register"
ConnectionString = Table.Publish(PublishTarget, True)
```

Свойство *Slicers* возвращает коллекцию срезов, связанных с таблицей. Свойство имеет тип Slicers, который ведет себя очень похоже на объект Collection и содержит коллекцию объектов Slicer. Свойство не может быть установлено. Объект Slicers используется для добавления, управления и удаления срезов, связанных с таблицей. Каждый объект среза предоставляет свойства и методы, которые позволяют управлять срезом. Свойства срезов была добавлена в объект ListObject в Excel 2013.

В следующем примере срез добавляется и помещается на том же листе, что и Таблица. Чтобы добавить срез сначала нужно создать объект SlicerCache для каждого среза:

```
Dim Table As ListObject
Dim SlicerCache As SlicerCache
Dim Slicer As Slicer
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set SlicerCache = ThisWorkbook.SlicerCaches.Add(Table, "Category")
SlicerCache.RequireManualUpdate = False
Set Slicer = SlicerCache.Slicers.Add(Table.Parent, ,
        "tblRegisterCategory", "Category", 100, 400)
```

Объект SlicerCache привязан к таблице и фильтруемому столбцу. Сам срез является визуальным представлением кэша среза и имеет родителя, имя, заголовок и позицию; он также имеет размер, но в приведенном выше примере используется размер по умолчанию. Свойство *RequireManualUpdate* объекта SlicerCache имеет значение False, чтобы избежать появления в срезе сообщения *Устарело*.

Следующий срез настроен на отображение категории *Расходы* (Expense) и скрытие категории *Доходы* (Income):

```
Dim Table As ListObject
Dim Slicer As Slicer
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set Slicer = Table.Slicers("tblRegisterCategory")
With Slicer.SlicerCache
        .SlicerItems("Expense").Selected = True
        .SlicerItems("Income").Selected = False
End With
```

Следующий срез настроен для отображения только одной категории:

```
Dim Table As ListObject
Dim Slicer As Slicer
Dim SlicerItem As SlicerItem
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set Slicer = Table.Slicers("tblRegisterCategory")
With Slicer.SlicerCache
        .ClearManualFilter
        For Each SlicerItem In .SlicerItems
                If SlicerItem.Name <> "Expense" Then
                        SlicerItem.Selected = False
                End If
        Next SlicerItem
End With
```

В следующем примере происходит очистка фильтр среза:

```
Dim Table As ListObject
Dim Slicer As Slicer
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set Slicer = Table.Slicers("tblRegisterCategory")
Slicer.SlicerCache.ClearManualFilter
```

В следующем примере срез удаляется. Обратите внимание, что кэш среза также удаляется:

```
Dim Table As ListObject
Dim Slicer As Slicer
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set Slicer = Table.Slicers("tblRegisterCategory")
Slicer.Delete
Table.ShowAutoFilter = False
```

Обратите внимание, что свойство таблицы *ShowAutoFilter* имеет значение False, чтобы скрыть раскрывающийся список, который остается после удаления среза. Если Автофильтр таблицы был включен при создании среза, то этот шаг не требуется. Если Автофильтр таблицы не был включен до добавления среза, то после удаления среза раскрывающийся элемент управления Автофильтром остается только у столбца, для которого был удален срез.

Свойство *Sort* возвращает объект сортировки таблицы. Свойство имеет тип Sort object. Свойство не может быть установлено. В следующем примере Таблица сортируется по дате и описанию:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
With Table
        .Sort.SortFields.Add .ListColumns("Date").DataBodyRange,_
                xlSortOnValues, xlAscending
        .Sort.SortFields.Add .ListColumns("Description")._
                DataBodyRange, xlSortOnValues, xlAscending
        .Sort.Apply
        .Sort.SortFields.Clear
End With
```

На защищенном листе невозможно выполнить сортировку Таблицы, если не будут разблокированы все ячейки в заголовке таблицы и теле данных. Или снимите защиту со всего листа, выполните сортировку, и затем снова защитите лист.

Свойство *SourceType* возвращает текущий источник таблицы. Свойство имеет тип XlListObjectSourceType. Свойство не может быть установлено. Свойство может принимать следующие константы:

- xlSrcExternal или 0 – источник является внешним источником данных, таким как сайт Microsoft SharePoint.
- xlSrcModel или 4 – источником является модель Power Pivot.
- xlSrcQuery или 3 – источником является запрос Power Query.
- xlSrcRange или 1 – источником является объект Range.
- xlSrcXml или 2 – источником является XML.

Свойство *Summary* возвращает или задает текст, используемый для замещающего текста при публикации таблицы. Свойство имеет строковый тип, введенный в Excel 2010. Установка этого свойства перезаписывает любое предыдущее значение. Это свойство представлено в пользовательском интерфейсе Excel в диалоговом окне *Замещающий текст*, доступ к которому можно получить, щелкнув правой кнопкой мыши в любом месте Таблицы и выбрав *Таблица –> Замещающий текст* (см. рис. 9.3). Это свойство отображается и редактируется в текстовом поле *Описание*.

Свойство *TableObject* возвращает объект TableObject Таблицы. Свойство имеет тип объекта TableObject. Свойство не может быть установлено. Объект TableObject предоставляет свойства и методы, позволяющие управлять объектами таблицы. TableObject – это объект, построенный на основе данных, полученных из модели Power Pivot. Он был введен в Excel 2013.

Свойство *XmlMap* возвращает объект XmlMap Таблицы, который предоставляет карту XML Таблицы. Свойство имеет тип объекта XmlMap. Объект XmlMap предоставляет свойства и методы, позволяющие управлять XML-картой. Свойство не может быть установлено.

Другие свойства Таблиц

Свойство ListColumn – это элемент в свойстве или коллекции ListColumns. ListColumn – это объект с рядом полезных свойств, включая следующие:

- Range – ссылки на ячейки в столбце, включая заголовок и итоговые строки, если они включены.
- DataBodyRange – тип объекта Range, который ссылается на столбец, исключая строки заголовка и итогов. Это пересечение диапазонов, представленных свойством Range объекта ListObject и диапазоном DataBodyRange объекта ListObject.
- Index – относительный номер индекса столбца, представленного объектом ListColumn.

- Parent – объект ListObject, которому принадлежит столбец.

ListColumn также включает метод Delete – удаляет столбец из Таблицы.

Каждый объект *ListRow* в коллекции строк имеет три часто используемых свойства: Range, которое ссылается на ячейки в этой строке; Index, которое является относительным номером индекса этой строки, и Parent, которое ссылается на объект ListObject, содержащий строку. Объект *ListRow* также имеет один метод *Delete*, который удаляет строку из таблицы.

Методы объекта Таблицы

Метод *Delete* удаляет Таблицу, включая все ее значения, формулы и форматирование. Не путайте этот метод с методом *Unlist*, который преобразует таблицу в обычный диапазон ячеек.

Метод *ExportToVisio* экспортирует и открывает в Visio динамическую сводную диаграмму в новом документе. Этот метод был добавлен в Excel 2007. Ошибка возникает, если Visio не установлен у вас на ПК (см. также Глава 3. Работа с таблицами Excel).

Метод *Publish* публикует таблицу в службе SharePoint. Он возвращает URL-адрес опубликованного списка в SharePoint в виде строки:

```
expression.Publish(Target, LinkSource)
```

где: *expression* – переменная, представляющая объект ListObject; *Target* – одномерный массив типа Variant, содержащий два или три элемента: URL-адрес SharePoint server, отображаемое имя списка и, при необходимости, описание списка; *LinkSource* – логическое значение. Если оно равно True, создает новую ссылку на новый список SharePoint. Если False, хранит ссылку на текущий список SharePoint и заменяет этот список, или, если нет никакого текущего списка, создает новый список на SharePoint без ссылок на него.

Метод *Refresh* обновляет таблицу из внешнего источника, если Таблица имеет такую связь. Если Таблица не связана с внешним источником данных, возвращает ошибку. Все изменения, внесенные в таблицу после последнего обновления, будут потеряны.

Метод *Resize* изменяет диапазон Таблицы на указанный диапазон. Метод принимает один параметр Range, который определяет новый диапазон Таблицы. Если строка заголовка включена, новый диапазон должен включать по крайней мере одну ячейку строки заголовка. Если строка заголовка отключена, новый диапазон должен содержать по крайней мере одну ячейку в первой строке диапазона тела данных. Столбцы можно добавлять и удалять с обоих концов, а строки можно добавлять и удалять только снизу. При удалении строк и столбцов все существующие данные в удаленных строках и столбцах остаются, но теперь находятся вне Таблицы. При добавлении строк и столбцов в Таблицу добавляются любые данные из дополнительных ячеек. После изменения размера структурированные ссылки в формулах в ячейках, отсутствующих в Таблице, преобразуются в абсолютные стандартные ссылки на ячейки. Если Таблица связана со списком SharePoint, можно добавлять и удалять только строки. Попытка добавить или удалить столбцы в связанной таблице приводит к ошибке.

Метод *Unlink* удаляет любую внешнюю ссылку на данные, если она существует.

Метод *Unlist* преобразует Таблицу в обычный диапазон ячеек. В интерфейсе Excel это эквивалентно команде *Работа с таблицами –> Конструктор –> Параметры стилей таблицы –> Инструменты –> Преобразовать в диапазон.*

Другие методы

Метод *Add* объекта ListObject добавляет новую таблицу, используя существующий список данных или другой источник. При использовании существующего заголовка числовые значения преобразуются в текст, а все повторяющиеся заголовки становятся уникальными путем добавления целых значений. Вот синтаксис этого метода:

```
expression.Add(SourceType, Source, LinkSource,
    XlListObjectHasHeaders, Destination, TableStyleName)
```

где:

- expression – переменная, представляющая объект ListObjects,

- SourceType – передает константу XlListObjectSourceType, которая определяет тип источника, используемого для создания Таблицы. Необязательный параметр. Если опущен, предполагается xlSrcRange.
- Source – если SourceType = xlSrcRange, то передайте объект Range, представляющий диапазон значений данных для преобразования в Таблицу. Необязательный параметр. Если опущен, используется текущий выбор. Если SourceType = xlSrcExternal, передайте массив строковых значений, указывающих соединение с источником, где элементы массива: 0 – URL-адрес сайта SharePoint, 1 – имя списка SharePoint, 2 – код представления GUID (уникальный шестнадцатеричный идентификатор представления),
- LinkSource – логическое значение, указывающее, должен ли внешний источник данных быть связан с объектом ListObject. Если SourceType имеет значение xlSrcExternal, то значение по умолчанию равно True, и этот параметр не требуется передавать. Если значение передается и SourceType является xlSrcRange, генерируется ошибка.
- XlListObjectHasHeaders – передает тип константы xlYesNoGuess, которая может принимать три значения: xlYes, xlNo или xlGuess. Константа указывает, имеют ли импортируемые данные метки столбцов. Если источник не содержит заголовков, Excel автоматически создает заголовки. Необязательный параметр. Если опущено, значение по умолчанию = xlGuess. Имя этого параметра должно было быть HasHeaders, но, когда метод был реализован, разработчики ошибочно использовали XlListObjectHasHeaders.
- Destination – передает объект Range, который указывает ссылку на одну ячейку в качестве назначения для верхнего левого угла Таблицы. Ошибка генерируется, если диапазон относится к нескольким ячейкам. Этот параметр должен быть указан, если SourceType = xlSrcExternal. Он игнорируется, если SourceType имеет значение xlSrcRange. Диапазон назначения должен находиться на листе, содержащем коллекцию ListObjects, указанную в expression. Столбцы вставляются перед целевым диапазоном, чтобы соответствовать новому списку, предотвращая перезапись существующих данных. Необязательный параметр.
- TableStyleName – передает имя стиля, который будет применен к Таблице. Необязательный параметр. Если опущен, применяется стиль по умолчанию.

Метод *Add* объекта ListRows вставляет одну новую строку в таблицу в указанной позиции. Вот синтаксис этого метода:

expression.Add(Position, AlwaysInsert)

где:

- expression – переменная, представляющая объект ListRows.
- Position – целое число, определяющее относительное положение новой строки. Новая строка вставляется над текущей строкой в этой позиции. Необязательный параметр. Если опущен, новая строка добавляется в нижнюю часть Таблицы.
- AlwaysInsert – логическое значение, указывающее, следует ли всегда сдвигать данные в ячейках под последней строкой таблицы при вставке новой строки, независимо от того, является ли строка под таблицей пустой. При значении True ячейки под таблицей сдвигаются на одну строку вниз. При значении False, если строка под Таблицей пуста, Таблица расширяется, чтобы занять (добавить) эту строку без сдвига ячеек под ней; если строка под Таблицей содержит данные, эти ячейки сдвигаются вниз при вставке новой строки.

Метод *Add* возвращает объект ListRow, представляющий новую строку.

Метод *Delete* объекта ListRow удаляет строку Таблицы, представленную объектом ListRow.

Метод *Add* объекта ListColumns вставляет один новый столбец в Таблицу в указанной позиции:

expression.Add(Position)

- expression – переменная, представляющая объект ListColumns.
- Position – целое число, определяющее относительное положение нового столбца. Новый столбец вставляется перед текущим столбцом в этой позиции. Необязательный параметр. Если этот параметр опущен, новый столбец добавляется в правую часть Таблицы.

Этот метод возвращает объект ListColumn, представляющий новый столбец.

Метод *Delete* объекта ListColumn удаляет столбец Таблицы, представленный объектом ListColumn.

Метод Автофильтр объекта Range

Метод позволяет создать критерии Автофильтра для столбца, очистить критерии Автофильтра или переключить статус Автофильтра:

expression.AutoFilter(Field, Criteria1, Operator, Criteria2, VisibleDropDown)

где:

- expression – выражение, возвращающее объект Range.
- Field – целочисленное смещение поля, на котором будет основан фильтр, где крайнее левое поле = 1. Необязательный параметр. Если опущен, то состояние Автофильтра (включено/выключено) переключается для всего диапазона. Отключение Автофильтра удаляет раскрывающиеся элементы управления Автофильтра.
- Criteria1 – критерий в виде числа, строки или массива, например, 20 или "Расход". Можно использовать знаки равенства = и неравенства <>. Чтобы найти несколько строк, используйте функцию Array со значением оператора xlFilterValues, например, Array("Значение 1", "Значение 2"). Подстановочные знаки * (один или несколько символов) и ? (можно использовать любой отдельный символ). Если оператор xlTop10Items, то Criteria1 задает число элементов, например 10. Когда оператор xlFilterDynamic, Criteria1 – это константа xlDynamicFilterCriteria (описана ниже). Необязательный параметр. Если опущен, критерии для столбца очищаются.
- Operator – передает константу XlAutoFilterOperator, определяющую тип фильтра. Необязательный параметр. Если опущен, то принимается равным 0, и Crtiteria1 рассматривается как простое значение для поиска; в этом случае, если в Crtiteria1 передается массив значений, то используется только последнее значение. Если этот параметр опущен, а Crtiteria1 не указан, то Автофильтр для указанного столбца будет очищен.
- Criteria2 – второй критерий в виде числа, строки или массива. Он используется с Crtiteria1 и Operator для построения составных критериев или когда Operator = xlFilterValues, а фильтруются дата и время (описано ниже). Необязательный параметр.
- VisibleDropDown – передает True, чтобы отобразить стрелку раскрывающегося списка Автофильтра для отфильтрованного поля, и False, чтобы скрыть раскрывающийся список Автофильтра для отфильтрованного поля. Необязательный параметр. Если опущен, принимает значение True.

Если заданы критерии фильтрации, то возвращаемое значение является значением Field. При переключении статуса Автофильтра возвращается значение True.

Значения констант оператора XlAutoFilterOperator:

- xlAnd или 1 – фильтры с логическим И для Criteria1 and Criteria2. Оба критерия являются строками, задающими условия, например, ">0" и "<100".
- xlBottom10Items или 4 – находит отображаемые элементы с наименьшим значением, где число элементов задается как число или строка в Criteria1, например, 5 или "5", то есть пять наименьших элементов. Столбец, указанный в параметре Field, должен содержать хотя бы одно число, иначе произойдет ошибка.
- xlBottom10Percent или 6 – найти отображаемые элементы с наименьшим значением, где процент указан как число или строка в Criteria1, например, 20 или "20" для элементов в нижних 20%. Столбец, указанный в параметре Field, должен содержать хотя бы одно число, иначе произойдет ошибка.
- xlFilterCellColor или 8 – находит цвет ячейки, где цвет указан как значение RGB в Criteria1.
- xlFilterDynamic или 11 – динамический фильтр, где критерий фильтра задается как значение XlDynamicFilterCriteria в Criteria1. Динамический фильтр – это фильтр, изменяющийся в зависимости от какого-либо другого значения, например сегодняшней даты или среднего значения в столбце.
- xlFilterFontColor или 9 – находит цвет шрифта, где цвет указан как значение RGB в Criteria1.

- xlFilterIcon или 10 – находит значок, указанный в Criteria1. Значки извлекаются из свойства ActiveWorkbookIconSets. Свойство IconSets представляет собой набор объектов IconSets, в котором каждый набор иконок представляет собой набор из ряда иконок, таких как набор "3 стрелки (цветные)" или "3 Arrows (Colored)". В приведенном ниже примере извлекается первый значок в наборе "3 стрелки (цветные)":

  ActiveWorkbook.IconSets(xlIconSet.xl3Arrows).Item(1)

  Обратите внимание, что при вводе в редакторе VBA "xlIconSet." (включая точку) появится всплывающий список IntelliSense со всеми доступными ссылками на набор значков.

- xlFilterValues или 7 – находит несколько значений, заданных в виде массива в Criteria1 или Criteria2. Значения в массиве должны быть строковыми и точно соответствовать отображаемому значению, если оно числовое; например, если соответствующие значения отображаются как "$25.00", передаваемое значение должно быть"$25.00". Criteria2 используется при поиске дат и времени. При поиске дат и времен массив передается как массив пар значений, где первое значение каждой пары является типом поиска, а второе – датой. Типы поиска: 0 – находит элементы в том же году, что и следующее значение даты / времени; 1 – находит элементы в том же месяце, что и следующее значение даты / времени; 2 – находит элементы на ту же дату, что и следующее значение даты / времени; 3 – находит элементы в тот же час, что и следующее значение даты / времени; 4 – находит элементы в ту же минуту, что и следующее значение даты / времени; 5 – находит элементы в ту же секунду, что и следующее значение даты / времени.

  Может быть передано любое количество пар типа поиска и значения даты / времени. Все значения типа поиска должны находиться в нечетных позициях в массиве (1, 3, 5 и т.д.), а все значения даты/времени должны быть в четных позициях (2, 4, 6 и т.д.). Любое значение, переданное для типа поиска, которого нет в приведенном выше списке, создает ошибку. Любое значение, переданное для значения даты/времени, которое не является значением даты/времени, создает ошибку. Допустимо любое значение даты/времени, которое может быть введено в ячейку и распознано как дата. Некоторые примеры значений Criteria2:

  Показать все элементы в 2014 году: Array(0, "1/1/2014")
  Показать все элементы в 2014 году, когда текущий год 2014: Array(0, "1/1")
  Показать все элементы в январе 2014 года: Array(1, "1/1/2014")
  Показать все элементы 15 января 2014 года: Array(2, "15/1/2014")
  Показывать все элементы 15, 20 и 25 января 2014 года: Array(2, "15/1/2014", 2, "20/1/2014", 2, "25/1/2014")
  Показать все элементы в 2013 году и в январе 2014 года: Array(0, "1/1/2013", 1, "1/1/2014")
  Показать все элементы в в 3 часа дня 15 января 2014 года: Array(3, "15/1/2014 15:00") или Array(3, "15/1/2014 3 PM")
  Показать все элементы в 15:01 15 января 2014 года: Array (3, " 15/1/2014 15:01") или Array(3, "15/1/2014 3:01 PM")

- xlOr или 2 – логическим ИЛИ для Criteria1 and Criteria2. Оба критерия – это строки, задающие условие, например "<0" и ">100" для значений меньше нуля или больше 100.
- xlTop10Items или 3 – находит отображаемые элементы с наибольшим значением, где число элементов задается как число или строка в Criteria1, например 5 или "5", то есть пять самых больших элементов. Столбец, указанный в параметре Field, должен содержать хотя бы одно число, иначе произойдет ошибка.
- xlTop10Percent или 5 – находит отображаемые элементы с наибольшим значением, где процент указан как число или строка в Criteria1, например 20 или "20" для элементов в верхних 20%. Столбец, указанный в параметре Field, должен содержать хотя бы одно число, иначе произойдет ошибка.

Значения констант оператора XlDynamicFilterCriteria:

- xlFilterToday – фильтрует все значения дат, равные сегодняшнему дню.
- xlFilterYesterday – фильтрует все значения дат, равные вчерашнему дню.

- xlFilterTomorrow – фильтрует все значения дат, равные завтрашнему дню.
- xlFilterThisWeek – фильтрует все значения дат на текущей неделе.
- xlFilterLastWeek – фильтрует все значения дат за предыдущую неделю.
- xlFilterNextWeek – фильтрует все значения дат на следующей неделе.
- xlFilterThisMonth – фильтрует все значения дат в текущем месяце.
- xlFilterLastMonth – фильтрует все значения дат за предыдущий месяц.
- xlFilterNextMonth – фильтрует все значения дат в следующем месяце.
- xlFilterThisQuarter – фильтрует все значения дат в текущем квартале.
- xlFilterLastQuarter – фильтрует все значения дат в предыдущем квартале.
- xlFilterNextQuarter – фильтрует все значения дат в следующем квартале.
- xlFilterThisYear – фильтрует все значения дат в текущем году.
- xlFilterLastYear – фильтрует все значения дат за предыдущий год.
- xlFilterNextYear – фильтрует все значения, относящиеся к следующему году.
- xlFilterYearToDate – фильтрует все значения дат с начала года по сегодня.
- xlFMterAMDatesInPeriodQuarterl – фильтрует все значения дат в квартале 1.
- xlFMterAMDatesInPeriodQuarter2 – фильтрует все значения дат в квартале 2.
- xlFMterAMDatesInPeriodQuarter3 – фильтрует все значения дат в квартале 3.
- xlFilterAllDatesInPeriodQuarter4 – фильтрует все значения дат в квартале 4.
- xlFilterAllDatesInPeriodJanuary – фильтрует все значения дат в январе.
- xlFilterANDatesInPeriodFebruary – фильтрует все значения дат в феврале.
- xlFilterAllDatesInPeriodMarch – фильтрует все значения дат в марте.
- xlFilterAllDatesInPeriodApril – фильтрует все значения дат в апреле.
- xlFilterAllDatesInPeriodMay – фильтрует все значения дат в мае.
- xlFilterAllDatesInPeriodJune – фильтрует все значения дат в июне.
- xlFilterAllDatesInPeriodJuly – фильтрует все значения дат в июле.
- xlFilterAllDatesInPeriodAugust – фильтрует все значения дат в августе.
- xlFilterAllDatesInPeriodSeptember – фильтрует все значения дат в сентябре.
- xlFilterAllDatesInPeriodOctober – фильтрует все значения дат в октябре.
- xlFilterAllDatesInPeriodNovember – фильтрует все значения дат в ноябре.
- xlFilterAllDatesInPeriodDecember – фильтрует все значения дат в декабре.
- xlFilterAboveAverage – фильтрует все значения выше среднего.
- xlFilterBelowAverage – фильтрует все значения ниже среднего.

Доступ к элементам Таблицы

Хотя свойства ListObject, ListColumns, ListRows предоставляют доступ к основным частям таблицы, существуют и другие способы доступа к элементам таблицы с использованием синтаксиса структурированных ссылок, описанного в главе 4. Эти способы могут быть более удобными, в зависимости от стиля программирования и предпочтений. В приведенных ниже примерах предполагается, что есть Таблица с именем "tblRegister" на листе с именем "Register" с заголовками столбцов "Date", "Description", "Category" и "Amount":

Date	Description	Category	Amount
1/3/1900	Transaction 1	Expense	-100.00
1/15/2014	Transaction 4	Income	350.00
2/1/2014	Transaction 2	Income	150.00
3/1/2014	Transaction 3	Expense	-75.00
Total			**325.00**

Чтобы использовать структурированную ссылку, используйте объект Range для извлечения диапазона, описанного ссылкой. Объект Range является дочерним объектом многих объектов Excel, включая объекты Application и Worksheet. При использовании объекта Range с объектом Application ссылка должна иметь глобальную область действия. При использовании Range с Worksheet (или Sheet) ссылка может иметь глобальную или локальную (Worksheet) область.

> Область действия имени определяет, откуда к нему можно получить доступ. Ссылка с глобальной областью действия может быть доступна из любого объекта. На ссылку с

локальной областью действия можно ссылаться только из листа, на котором определена конкретная ссылка.

Имена таблиц определены глобально, то есть, вы можете получить к ним доступ из любого модуля кода, не уточняя ссылку на объект листа, в котором находится таблица. Например, следующие ссылки эквивалентны:

```
ThisWorkbook.Worksheets("Register").Range("tblRegister[Date]")

Application.Range("tblRegister[Date]")

Range("tblRegister[Date]")

[tblRegister[Date]]
```

Чтобы соответствовать этому правилу, Excel требует, чтобы каждая Таблица в книге имела уникальное имя. Это правило управляет всеми глобальными именами, а не только именами Таблиц.

Чтобы уменьшить вероятность коллизии имен, можно предварить все имена таблиц общим префиксом, часто называемым "венгерской нотацией", описанной в главе 2, например"tbl".

Excel 2003 не поддерживает структурированные ссылки. Существуют также некоторые различия между Excel 2007, 2010 и 2013 с точки зрения поддержки структурированных ссылок.

Создание и присвоение имени Таблице

Таблицы создаются с помощью метода *Add* объекта ListObjects. После создания новой Таблицы свойству DisplayName объекта ListObject присваивается имя новой Таблицы. Имя Таблицы, присваиваемое по умолчанию, зависит от источника Таблицы: xlSrcRange (диапазон данных на листе), xlSrcExternal (внешний источник данных), xlSrcModel (модель данных Power Pivot) и xlSrcQuery (запрос). Тип источника xlSrcXml (источник XML) не рассматривается, но показаны обходные пути.

Использование диапазона данных

В этом примере создается новая таблица, использующая существующий диапазон данных с заголовками. Параметр SourceType имеет значение xlSrcRange:

```
Dim TableRange As Range
Dim Table As ListObject
Set TableRange = ThisWorkbook.Worksheets("Register"). _
        Range("A1").CurrentRegion
Set Table = ThisWorkbook.Worksheets("Register"). _
        ListObjects.Add(xlSrcRange, TableRange, ,xlYes)
Table.DisplayName = "tblRegister"
```

Обратите внимание, что четвертый параметр, xlYes, сообщает Excel, что список данных уже содержит заголовки. В этом примере Таблица будет названа сразу же после ее создания; это поможет вам найти объект ListObject позже.

Использование модели данных Power Pivot

В этом примере для создания соединения с базой данных SQL Server используется объект TableObject. Таблица SQL "Product" добавляется в модель данных Power Pivot. Таблица помещается на рабочий лист "Sheet1" в ячейку A1. Поскольку взаимодействие происходит с моделью данных, то вместо объекта ListObject с xlSrcModel, переданного для SourceType, должен использоваться объект TableObject. Измените текст "YourServerName" на имя нужного SQL-сервера. Используется база данных AdventureWorks2012:

```
Dim SQLConnection As WorkbookConnection
Dim TargetWorksheet As Worksheet
Dim Table As TableObject
Dim ConnectionString As String
Set TargetWorksheet = ThisWorkbook.Worksheets("Sheet1")
ConnectionString = "OLEDB;Provider=SQLOLEDB.1;Integrated Security=SSPI;" _
```

```
        & "Initial Catalog=AdventureWorks2012;Data Source=YourServerName"
    Set SQLConnection = ActiveWorkbook.Connections.Add2("FriendlyName", _
        "Description", ConnectionString, "Product", 3, True)
    With TargetWorksheet
        Set Table = .ListObjects.Add(SourceType:=xlSrcModel, _
            Source:=SQLConnection, Destination:=.Range("A1")).NewTable
    End With
    Table.ListObject.DisplayName = "tblNewTable"
```

Константа xlSrcModel была добавлена в Excel 2013.

В следующем примере предполагается, что книга уже имеет соединение SQL Server с Таблицей в модели данных Power Pivot, и задача состоит в том, чтобы извлечь данные из таблицы модели данных в новую Таблицу Excel. Тип источника – xlSrcModel, и предполагается, что имя Таблицы модели данных – "Product". Этот пример работает только в Excel 2013:

```
Dim ModelSource As Model
Dim SourceTable As ModelTable
Dim TargetWorksheet As Worksheet
Dim Table As TableObject
Set TargetWorksheet = ThisWorkbook.Worksheets("Sheet1")
Set ModelSource = ThisWorkbook.Model
Set SourceTable = ModelSource.ModelTables("Product")
Set Table = TargetWorksheet.ListObjects.Add(SourceType:=xlSrcModel, _
    Source:=SourceTable.SourceWorkbookConnection, _
    LinkSource:=True, Destination:=DestinationSheet.Range("A1")).TableObject
Table.Refresh
```

Использование запроса Power Query

В этом примере объект QueryTable используется для создания соединения с базой данных SQL Server. Таблица "Product"добавляется на лист "Sheet1" в ячейку A1. Измените текст "YourServerName" на имя нужного SQL-сервера. Используется база данных AdventureWorks2012:

```
Dim TargetWorksheet As Worksheet
Dim Table As QueryTable
Dim ConnectionString As String
Set TargetWorksheet = ThisWorkbook.Worksheets("Sheet1")
ConnectionString = "OLEDB;Provider=SQLOLEDB.1;Integrated Security=SSPI;" _
    & "Initial Catalog=AdventureWorks2012;Data Source=YourServerName"
Set Table = TargetWorksheet.ListObjects.Add(SourceType:=xlSrcExternal, _
    Source:=ConnectionString, LinkSource:=True, _
    Destination:=DestinationSheet.Range("A1")).QueryTable
Table.CommandText = Array("""AdventureWorks2012"".""Production"".""Product""")
Table.CommandType = xlCmdTable
Table.Refresh BackgroundQuery:=False
Table.ListObject.DisplayName = "tblNewTable"
```

Константа xlSrcQuery была добавлена в Excel 2007.

При использовании xlSrcExternal необходимо указать параметр назначения. При использовании объекта QueryTable необходимо задать свойства CommandText и CommandType перед обновлением соединения.

В этом примере используется тип источника xlSrcExternal, который используется для любого внешнего подключения к данным. Передача xlSrcQuery для параметра SourceType приводит к тому же результату. Как правило, xlSrcQuery используется для подключения к базе данных, а xlSrcExternal используется для подключения к SharePoint.

Использование источника XML

По замыслу, метод *Add* объекта ListObjects с типом источника xlSrcXml должен создать объект ListObject, используя в качестве источника XML-файл. Однако этот метод ненадежен, и нет

известных рабочих примеров его использования. Для импорта исходного файла XML в Таблицу рекомендуется использовать два метода. Во-первых, необходимо импортировать XML-файл в новую пустую книгу:

```
Workbooks.OpenXML Filename:="C:\XML File Name.xml", _
        LoadOption:=xlXmlLoadImportToList
```

Во-вторых, необходимо импортировать XML-файл в существующий лист в указанном диапазоне.:

```
ActiveWorkbook.XmlImport URL:="C:\XML File Name.xml", _
        ImportMap:=Nothing, Overwrite:=True,
        Destination:=Range("A1")
```

В этих примерах, если указанный источник XML не ссылается на схему, Excel создает ее на основе того, что он найдет в указанном XML-файле.

Информация о Таблице

В следующих примерах предполагается, что DataBodyRange является допустимым объектом диапазона. Если в Таблице нет существующих строк (то есть если ListRows.Count равно 0), любая ссылка на DataBodyRange вернет ошибку.

Определение того, существует ли таблица

Это не простая задача, так как имя таблицы, которой нет, может использоваться с коллекцией ListObjects. В следующем коде показано, как использовать обработку ошибок для определения того, существует ли таблица:

```
Dim Table As ListObject
Set Table = Nothing
On Error Resume Next
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
On Error GoTo 0
If Table Is Nothing Then
        Debug.Print "Table does not exist"
Else
        Debug.Print "Table exists"
End If
```

Зачем устанавливать объектную переменную равную Nothing, прежде чем пытаться присвоить ей значение? В приведенном выше случае это не обязательно, поскольку VBA инициализирует каждую переменную, когда она определена с помощью оператора Dim. Но он включен выше в качестве примера написания надежного кода, потому что, если возникает ошибка, переменная не трогается и, если она уже содержит ссылку на другой объект, следующий тест не даст желаемого результата.

Определение адреса таблицы

В следующем примере выводится адрес Таблицы и адрес DataBodyRange Таблицы:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Debug.Print "Table's address: " & Table.Range.Address
Debug.Print "Table's data body range address: " & Table.DataBodyRange.Address
```

Определение количества строк

Количество строк в таблице определяется с помощью свойства Count объекта ListRows:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Debug.Print "Number of rows: " & Table.ListRows.Count
```

Свойство Count возвращает 0, если таблица пуста (то есть имеет одну строку, готовую для ввода данных, и нет данных ни в одной ячейке).

Определение количества столбцов

Количество столбцов в таблице определяется с помощью свойства Count объекта ListColumns:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Debug.Print "Number of columns: " & Table.ListColumns.Count
```

Определение того, существует ли столбец

Это также непростая задача. В следующем коде показано, как использовать обработку ошибок для определения того, существует ли столбец:

```
Dim Table As ListObject
Dim ListColumn As ListColumn
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set ListColumn = Nothing
On Error Resume Next
Set ListColumn = Table.ListColumns("Description")
On Error GoTo 0
If ListColumn Is Nothing Then
        Debug.Print "Column does not exist"
Else
        Debug.Print "Column exists"
End If
```

Добавление строк

Существует несколько способов добавления новых строк в Таблицу. Если вы добавляете одну строку, используйте метод *Add* объекта ListRows. Он возвращает объект ListRow, который затем можно использовать для добавления значений в эту новую строку:

```
Dim Table As ListObject
Dim NewRow As ListRow
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set NewRow = Table.ListRows.Add
With NewRow.Range
        .Columns(1).Value = #1/1/2015#
        .Columns(2).Value = "Transaction 20"
        .Columns(3).Value = "Expense"
        .Columns(4).Value = -75
End With
```

Обратите внимание, что в этом примере параметр Position не был передан методу Add, что привело к добавлению новой строки в конец таблицы. Чтобы вставить новую строку в определенную позицию таблицы, используйте параметр Position.

Чтобы добавить более одной строки в нижнюю часть таблицы, удобнее добавлять строки за один шаг, чем вызывать метод Add объекта ListRows несколько раз. В следующем примере строка итогов отключена, новые данные копируются в пустые ячейки непосредственно под Таблицей, после чего строка итогов включается. (Если функция TotalRow не отключена, Таблица не распознает новые строки и поэтому не расширяется для их включения.) Новые данные копируются из диапазона A2:D11 на листе Data:

```
Dim Table As ListObject
Dim NewValues As Variant
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
NewValues = ThisWorkbook.Worksheets("Data").Range("A2:D11").Value
Table.ShowTotals = False
With Table.DataBodyRange
        .Resize(10).Offset(.Rows.Count).Value = NewValues
End With
Table.ShowTotals = True
```

Чтобы вставить несколько строк в середину таблицы, используйте метод Insert объекта Range для вставки пустых ячеек, а затем эти ячейки заполняются новыми данными. В следующем примере 10 строк данных вставляются после существующей строки 2 (и перед строкой 3):

```
Dim Table As ListObject
Dim NewValues As Variant
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
NewValues = ThisWorkbook.Worksheets("Data").Range("A2:D11").Value
With Table.DataBodyRange
        .Resize(10).Offset(2).Insert Shift:=xlShiftDown
        .Resize(10).Offset(2).Value = NewValues
End With
```

Удаление строк

Метод, который вы используете, зависит от того, сколько строк вы хотите удалить.

Удаление одной строки

Для удаления одной строки используется метод *Delete* объекта ListRow:

```
Dim Table As ListObject
Dim ListRow as ListRow
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Set ListRow = Table.ListRows(3)
ListRow.Delete
```

Переменной ListRow присваивается третий объект ListRow в коллекции ListRows, а затем вызывается метод Delete объекта ListRow. Вот альтернативная, более короткая версия примера, которая не требует переменной ListRow:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Table.ListRows(3).Delete
```

Удаление нескольких строк

Удаление нескольких строк одновременно требует использования метода Delete объекта Range. В следующем примере удаляются 10 строк, начиная со строки 3:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Table.DataBodyRange.Resize(10).Offset(2).Delete
```

Удаление всех строк

В следующем примере удаляются все строки таблицы:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Table.DataBodyRange.Delete
```

В этом примере объекту DataBodyRange после завершения кода присваивается значение Nothing. Любые последующие ссылки на этот объект возвращают ошибку, если в Таблицу не добавлена хотя бы одна строка.

Циклы

В первом примере выполняется цикл по всем строкам Таблицы, добавляя в переменную TotalExpenses значения всех строк в столбце Expense (расходы) и выводя результат в окно Immediate:

```
Dim Table As ListObject
Dim ListRow As ListRow
Dim TotalExpenses As Double
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
For Each ListRow In Table.ListRows
        If ListRow.Range.Columns(3).Value = "Expense" Then
```

```
                TotalExpenses = TotalExpenses + ListRow.Range.Columns(4).Value
        End If
    Next ListRow
    Debug.Print "Total expenses: " & TotalExpenses
```

Ниже приведен альтернативный метод, использующий имена столбцов:

```
    Dim Table As ListObject
    Dim ListRow As ListRow
    Dim TotalExpenses As Double
    Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
    For Each ListRow In Table.ListRows
        If Intersect(ListRow.Range, Table.ListColumns("Category").Range) _
            .Value = "Expense" Then
        TotalExpenses = TotalExpenses + Intersect(ListRow.Range, Table.
        ListColumns("Amount").Range).Value
    End If
    Next ListRow
    Debug.Print "Total expenses: " & TotalExpenses
```

Во втором примере выполняется цикл по столбцам Таблицы, выводя имя каждого столбца на экран с помощью коллекции ListColumns и оператора For / Each:

```
    Dim Table As ListObject
    Dim ListColumn As ListColumn
    Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
    For Each ListColumn In Table.ListColumns
        Debug.Print ListColumn.Name
    Next ListColumn
```

Фильтрация

Одной из самых мощных особенностей Таблиц является их способность фильтровать строки. Объектная модель Excel предоставляет объект AutoFilter (дочерний элемент объекта ListObject) и метод AutoFilter (дочерний элемент объекта Range), что позволяет полностью контролировать процесс фильтрации в VBA. Используйте объект ListObject.AutoFilter для проверки текущих настроек Автофильтра, обновления Автофильтра и очистки Автофильтра. Используйте метод Range.AutoFilter для задания критериев Автофильтра.

Включение и выключение Автофильтра

Вы включаете и выключаете Автофильтр, задавая свойству ShowAutoFilter значения True (вкл.) и False (выкл.). В следующем примере показано, как включить Автофильтр:

```
    Dim Table As ListObject
    Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
    Table.ShowAutoFilter = True
```

Поскольку Автофильтр – это объект, который не имеет значения при отключенном Автофильтре, любой код, ссылающийся на какие-либо свойства и методы объекта Автофильтра, будет генерировать ошибку, если Автофильтр отключен. Чтобы избежать ошибок, убедитесь, что Автофильтр включен, и получите доступ к свойствам и методам объекта Автофильтра только в этом случае. Примеры ниже показывают, как осуществить эту проверку.

Вы также можете включать и отключать Автофильтр, повторно вызывая метод Range.AutoFilter без каких-либо параметров. Использование этого метода просто переключает состояние Автофильтра.

Определение состояния фильтрации

Объект Автофильтр используется для определения того, включен ли Автофильтр:

```
    Dim Table As ListObject
    Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
    If Table.ShowAutoFilter Then
        Debug.Print "AutoFilter is on"
```

```
Else
        Debug.Print "AutoFilter is off"
End If
```

Если Автофильтр включен, вы используете свойство *FilterMode* объекта Автофильтра, чтобы определить, установлены ли критерии фильтрации:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
If Table.ShowAutoFilter Then
        If Table.AutoFilter.FilterMode Then
                Debug.Print "Filtering is active"
        Else
                Debug.Print "Filtering is inactive"
        End If
Else
        Debug.Print "AutoFilter is off"
End If
```

Определение того, фильтруется ли столбец

Если Автофильтр включен, можно использовать свойство *On* объекта Filter, чтобы определить, имеет ли столбец активный критерий фильтра:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
If Table.ShowAutoFilter Then
        If Table.AutoFilter.Filters(3).On Then
                Debug.Print "Column 3 is being filtered"
        End If
Else
        Debug.Print "AutoFilter is off"
End If
```

Создание фильтров

Вы создаете (применяете) фильтры по одному столбцу за раз. Для добавления и удаления критериев фильтрации используется метод AutoFilter объекта Range. При применении критерия Автофильтра строка заголовка включается автоматически. В примерах ниже предполагается, что Таблица не имеет активных критериев фильтрации: Ниже приведены примеры, каждый из которых начинается с этих двух строк кода. Предполагается, что Таблица не имеет активных критериев фильтрации:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
```

В следующем примере фильтруются строки со значением Expense в третьем столбце.:

```
Table.Range.AutoFilter Field:=3, Criteria1:="Expense"
```

А здесь фильтруются строки с Expense или Income в третьем столбце.:

```
Table.Range.AutoFilter Field:=3, _
        Criteria1:=Array("Expense", "Income"), Operator:=xlFilterValues
```

Ниже показаны только строки со значениями во втором столбце, которые начинаются с Transaction:

```
Table.Range.AutoFilter Field:=2, Criteria1:="Transaction*"
```

Строки в четвертом столбце со значениями больше нуля:

```
Table.Range.AutoFilter Field:=4, Criteria1:=">0"
```

Строки с Income в третьем столбце и значениями более 100 – в четвертом:

```
Table.Range.AutoFilter Field:=3, Criteria1:="Income"
Table.Range.AutoFilter Field:=4, Criteria1:=">100"
```

Повторное применение критериев активного фильтра

Видимость строк в отфильтрованной таблице может не совпадать с критериями фильтра при изменении данных и добавлении новых строк. Эту ситуацию можно исправить, повторно применив критерии активного фильтра:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
If Table.ShowAutoFilter Then
        Table.AutoFilter.ApplyFilter
End If
```

Очистка фильтра одного столбца

Вы можете очистить фильтр одного столбца, используя метод AutoFilter и указав только параметр Field. В следующем примере очищаются критерии фильтра третьего столбца:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
Table.Range.AutoFilter Field:=3
```

Очистка всех фильтров

Вы можете очистить критерии фильтрации для всех столбцов за один шаг, не отключая Автофильтр, вызвав метод ShowAllData:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
If Table.ShowAutoFilter Then
        Table.AutoFilter.ShowAllData
End If
```

Скрытие раскрывающихся элементов управления по столбцам

Вы можете скрыть раскрывающиеся элементы управления Автофильтра в определенных столбцах. В следующем примере показано, как скрыть раскрывающийся элемент управления AutoFilter во втором столбце Таблицы:

```
Dim Table As ListObject
Set Table = ThisWorkbook.Worksheets("Register").ListObjects("tblRegister")
If Table.ShowAutoFilter Then
        Table.Range.AutoFilter Field:=2, VisibleDropDown:=False
End If
```

Пользовательские процедуры

В следующих разделах приведены некоторые пользовательские процедуры. Более надежные версии этих программ и ряд других программ, а также полезные утилиты и библиотеки доступны по адресу http://exceltables.com/.

Делаем объемную вставку

Следующая функция вставляет массив значений в Таблицу и возвращает новые строки в виде диапазона. Если задана строка, то значения вставляются выше этой строки; в противном случае значения добавляются в нижнюю часть Таблицы. Функция также сопоставляет столбцы значений со столбцами Таблицы с помощью параметра ColumnAssignmentS. Дополнительные сведения о параметрах см. в комментариях к процедуре.

```
Public Function BulkInsertIntoTable( _
        ByVal Table As ListObject, _
        ByVal Values As Variant, _
        Optional ByVal Position As Long = -1, _
        Optional ByVal ColumnAssignments As Variant _
) As Range
' Insert an array of values into a Table. Optionally specify the row before
' which the new rows are inserted. Optionally specify how the columns are
' assigned. The new rows in the Table are returned as a Range.
```

```
' Syntax
' BulkInsertIntoTable(Table, Values, Position, ColumnAssignments)
' Table - A Table object.
' Values - A single value, a single dimension array of values, or a two
' dimension array of values.
' Position - The row number before which the new rows are inserted. Optional.
' If omitted then the new rows are appended to the end of the Table.
' ColumnAssignments - A single dimension array of integer values specifying
' which Table column receives what column of values in the Values parameter.
' Each element in the array is a column number in the Table. The position of
' the element in the array corresponds to the column in the Values array.
' Optional. If omitted then the values are placed in column order starting in
' the first Table column. For example, passing Array(2,3,1) results in this
' column mapping:
' Values column 1 is placed in Table column 2.
' Values column 2 is placed in Table column 3.
' Values column 3 is placed in Table column 1.
Dim Calculation As XlCalculation
Dim ScreenUpdating As Boolean
Dim Result As Long
Dim TwoDimensionArray As Boolean
Dim WorkArray As Variant
Dim Column As Long
Dim SourceColumn As Long
Dim TargetColumn As Long
Dim ShowTotals As Boolean
Dim InsertRange As Range

' Exit if no values to insert
If IsEmpty(Values) Then Exit Function

Calculation = Application.Calculation
Application.Calculation = xlCalculationManual
ScreenUpdating = Application.ScreenUpdating
Application.ScreenUpdating = False

' Normalize Values parameter - must be a two-dimension array
On Error Resume Next
Result = LBound(Values, 2)
TwoDimensionArray = Err.Number = 0
On Error GoTo 0
If Not TwoDimensionArray Then
        If Not IsArray(Values) Then
                Values = Array(Values)
        End If
        ReDim WorkArray(1 To 1, 1 To UBound(Values) - LBound(Values) + 1)
        For Column = 1 To UBound(WorkArray, 2)
                WorkArray(1, Column) = Values(Column - 1 + LBound(Values))
        Next Column
        Values = WorkArray
End If

' Normalize Position parameter
If Position < 0 Then
        Position = Table.ListRows.Count
End If
```

```
Position = Application.Max(1, Application.Min(Position, _
        Table.ListRows.Count+ 1))

' Save total row setting and disable total
ShowTotals = Table.ShowTotals
Table.ShowTotals = False

' Insert the new rows
If Table.ListRows.Count > 0 And Position <= Table.ListRows.Count Then
        Table.DataBodyRange.Resize(UBound(Values)). _
                Offset(Position - 1).InsertShift:=xlShiftDown
End If
If Table.ListRows.Count > 0 Then
        Set InsertRange = Table.DataBodyRange.Resize _
                (UBound(Values)).Offset(Position - 1)
Else
        Set InsertRange = Table.InsertRowRange.Resize(UBound(Values))
End If
If IsEmpty(ColumnAssignments) Or IsMissing(ColumnAssignments) Then
        InsertRange.Value = Values
Else
        For TargetColumn = LBound(ColumnAssignments) To _
                UBound(ColumnAssignments)
                SourceColumn = TargetColumn - LBound(ColumnAssignments) + 1
                If ColumnAssignments(TargetColumn) >= 1 And _
                        ColumnAssignments(TargetColumn) <= _
                        Table.ListColumns.Count Then
                InsertRange.Columns(ColumnAssignments(TargetColumn)) _
                        .Value = Application.Index(Values, , SourceColumn)
                End If
        Next TargetColumn
End If
Set BulkInsertIntoTable = InsertRange

' Restore the total row setting
Table.ShowTotals = ShowTotals

Application.Calculation = Calculation
Application.ScreenUpdating = ScreenUpdating


End Function
```

Восстановление форматирования и формул

Обычно Таблица поддерживает одинаковое форматирование и формулы во всех строках, исключая строки заголовка и итоговые строки. При изменении форматирования или формулы в ячейке столбца Таблицы Excel применяет это новое форматирование или формулу ко всему столбцу. Форматирование и формулы автоматически применяются к новым строкам по мере их добавления.

Форматирование или формулы Таблицы могут стать несогласованными, если вы вручную редактируете форматирование или применяете разные формулы в одном столбце Таблицы. Вы можете исправить столбец Таблицы, повторно применив форматирование ко всему столбцу плюс одна дополнительная строка внизу (строка итогов должна быть отключена, чтобы выполнить эту корректировку). Вы можете исправить формулу (преобразовать столбец обратно в вычисляемый столбец), применив формулу ко всему столбцу.

Процедура ниже выполняет это с помощью метода изменения размера Таблицы. Сначала Таблица изменяется, чтобы быть только одной строкой. Затем форматирование и формулы удаляются из всех строк таблицы от строки 2 до последней строки плюс одна строка. Наконец, диапазон Таблицы возвращается к тому, что было. На этом заключительном этапе Excel должен применить форматирование и формулы в первой строке ко всем строкам ниже первой строки. В результате получается последовательно отформатированная таблица, использующая первую строку данных в качестве шаблона для всех остальных строк. Код предполагает, что в Таблице есть по крайней мере одна строка данных:

```
Public Sub RepairTable( _
                ByVal Table As ListObject _
        )
' Repair the Table's formatting and formulas by making them consistent down the
' entire length of each column.
'
' Syntax
'
' RepairTable(Table)
'
' Table - A Table object (ListObject object).

Dim RowCount As Long
Dim ListColumn As ListColumn
Dim ShowTotals As Boolean

RowCount = Table.ListRows.Count
If RowCount < 2 Then Exit Sub

With Table
        ShowTotals = .ShowTotals
        .ShowTotals = False
        .Resize .HeaderRowRange.Resize(2)
        For Each ListColumn In .ListColumns
                With ListColumn.DataBodyRange.Resize( _
                        Application.Max(RowCount, 1)).Offset(1)
                        If Left(.Rows(1).Formula, 1) = "=" Then
                                .Cells.Clear
                        Else
                                .Cells.ClearFormats
                        End If
                End With
        Next ListColumn
        .Resize .HeaderRowRange.Resize(1 + RowCount)
        .ShowTotals = ShowTotals
End With


End Sub
```

Копирование стиля Таблицы в новую книгу

Нет простого способа скопировать стиль Таблицы из одной книги в другую. Следующий пример кода копирует стиль, присвоенный Таблице с именем "tblRegister", в книгу "Destination Workbook. xlsx":

```
Sub ExportTableStyle()
        Dim Source As Workbook
        Dim Target As Workbook
        Dim Table As ListObject
```

```
    Set Source = ThisWorkbook
    Set Target = Workbooks("Destination Workbook.xlsx")
    Set Table = Source.Worksheets("Register").ListObjects("tblRegister")
    Target.Worksheets.Add Before:=Target.Worksheets(1)
    Table.Range.Copy Target.Worksheets(1).Range("A1")
    Target.Worksheets(1).Delete
End Sub
```