OpenOffice.org

Программирование на языке OpenOffice.org BASIC

Авторские права

Перевод

Дмитрий Чернов

Благодарности

Данное руководство является переводом *StarOffice 8 Programming Guide for BASIC* фирмы Sun Microsystems.

Содержание

Глава 1. Введение

Это руководство знакомит с программированием на языке OpenOffice.org Basic (далее OOo Basic) и указывает возможные применения, обеспечиваемые использованием OOo Basic в OpenOffice.org. Чтобы получить максимальный эффект от этой книги, Вы должны быть знакомыми с другими языками программирования.

Подготовленные исчерпывающие примеры помогут Вам быстро разработать ваши собственные программы на OOo Basic.

О OpenOffice.org Basic

Язык программирования OOo Basic был специально разработан для OpenOffice.org и сильно интегрирован в офисный пакет.

Поскольку название подсказывает, OOo Basic – язык программирования из семейства Basic. Любой, кто раньше работал с другими языками Basic — в особенности с Visual Basic или Visual Basic for Applications (VBA) от Microsoft — быстро привыкнет к OOo Basic. Многие базовые конструкции OOo Basic совместимы с Visual Basic.

Язык программирования OOo Basic может быть разделен на четыре компонента:

- **Язык OOo Basic**: Определяет элементарные лингвистические конструкции, например, для определения переменных, циклов и функций;
- **Библиотека времени выполнения**: Обеспечивает стандартные функции, которые не имеют никакой прямой связи с OpenOffice.org, например, функции для редактирования чисел, строк, значений данных, и файлов;
- **OOo API (Интерфейс прикладного программирования)**: обеспечивает доступ к документам OpenOffice.org и позволяет их создавать, сохранять, изменять, и печатать;
- **Редактор Диалогов**: Позволяет создавать диалоговые окна и обеспечивает возможность добавления элементов управления и обработчиков событий.

Примечание	Совместимость между OOo Basic и VBA затрагивает как язык OOo Basic, так и Библиотеки времени выполнения. OOo API и Редактор Диалога не совместимы с VBA (стандартизация этих интерфейсов сделала многие из концепций предусмотренных в OpenOffice.org невыполнимыми)

Для кого предназначен OpenOffice.org Basic

Возможности заявленные для OOo Basic начинаются там, где заканчиваются стандартные функции OpenOffice.org. Обычные задачи могут быть поэтому автоматизированы в OOo

Basic, могут быть установлены связи с другими программам – например с сервером базы данных – и могут быть выполнены сложные действия при нажатии на кнопку с использованием предопределенных сценариев.

OOo Basic предлагает полный доступ ко всем функциям OpenOffice.org, поддерживает все функции, изменяет типы документов, и обеспечивает возможность создания персональных диалоговых окон.

Использование OpenOffice.org Basic

OOo Basic может использоваться любым пользователем OpenOffice.org без любых дополнительных программ или вспомогательных средств. Даже в стандартной установке OOo Basic имеет все компоненты, необходимые для создания своих макросов Basic, включая

- **интегрированную среду разработки (IDE)**, которая предоставляет редактор для создания и отладки макросов;
- **интерпретатор**, который необходим для выполнения макросов OOo Basic;
- **интерфейсы** к различным приложениям OpenOffice.org, которые предоставляют прямой доступ к документам Office.

Дополнительная Информация

Компоненты OOo API , которые обсуждаются в этом руководстве, были выбраны на основе их практической полезности для программиста OOo Basic. Вообще, здесь обсуждается только часть интерфейсов. Для более детальной картины, см. справочную информацию по API, которая является доступной в Интернете по адресу: http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html

Руководство Разработчика описывает программный интерфейс приложений OpenOffice.org более подробно чем это руководство, но прежде всего предназначено для программистов на Java и C++. Любой, кто уже знаком с программированием на OOo Basic, может найти дополнительную информацию в Руководстве разработчика на OOo Basic и программировании для OpenOffice.org. Вы можете загрузить Руководство разработчика в Интернете по адресу: http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html

Программисты, которые хотят работать непосредственно на Java или C++, а не на OOo Basic, должны консультироваться с Руководством Разработчика OpenOffice.org вместо этого руководства. Программирование для OpenOffice.org на Java или C++ значительно более сложный процесс чем программирование на OOo Basic.

Глава 2. Язык OpenOffice.org Basic

OOo Basic принадлежит семье языков Basic. Многие части OOo Basic идентичны Microsoft Visual Basic for Applications и Microsoft Visual Basic. Любой, кто уже работал с этими языками, может быстро привыкнуть к OOo Basic.

Программисты других языков – таких как Java, C++ или Delphi – должны также при знакомстве найти легким для себя OOo Basic. OOo Basic полноценно-развитый процедурный язык программирования и больше не использует элементарные структуры управления, такие как `GoTo` и `GoSub`.

Вы можете также извлечь выгоду из преимуществ объектно-ориентированного программирования так как интерфейс в OOo Basic позволяет Вам использовать внешние библиотеки объектов. Весь OpenOffice.org API основан на этих интерфейсах, которые описаны более подробно в следующих главах этого документа.

Эта глава дает краткий обзор ключевых элементов и конструкций языка OOo Basic, а так же как структуры, посредством которых приложения и библиотеки связываются с OOo Basic.

Краткий обзор программ OpenOffice.org Basic

OOo Basic – интерпретируемый язык. В отличие от C++ или Turbo Pascal, компилятор OOo Basic не создает выполняемые или саморазворачивающиеся файлы, которые автоматически способны к выполнению. Вместо этого Вы можете выполнить программу OOo Basic по нажатию кнопки. Код сначала проверяется на наличие очевидных ошибок и затем выполняется строка за строкой.

Строка программы

Реализация построчного выполнения интерпретатором Basic – одно из ключевых различий между Basic и другими языками программирования. Принимая во внимание, что положение разрывов строки в исходном тексте программ Java, C++, или Delphi является несущественным, каждая строка в программе Basic формирует отдельный элемент. Вызовы функции, математические выражения, и другие лингвистические элементы, типа функции и заголовков циклов, должны заканчиваться на той же самой строке, на которой они начинаются.

Если недостаточно места, или если это приводит к образованию слишком длинных строк, то несколько строк могут быть объединены добавлением символа подчеркивания _. Следующий пример показывает, как четыре строки математического выражения могут быть связаны:

```
LongExpression = (Expression1 * Expression2) + _
    (Expression3 * Expression4) + _
```

```
(Expression5 * Expression6) + _
(Expression7 * Expression8)
```

Примечание	Подчеркивание всегда должно быть последним символом в связанной строке и не может сопровождаться пробелом или табуляцией, иначе код вызовет ошибку.

В дополнение к соединению отдельных строк, Вы можете использовать двоеточия, чтобы разделить одну строку на несколько секций так, чтобы было достаточное место для нескольких выражений. Так присваивания

```
a = 1
a = a + 1
a = a + 1
```

могут быть записаны следующим образом:

```
a = 1 : a = a + 1 : a = a + 1
```

Комментарии

В дополнение к коду программы, который будет выполняться, программа OOo Basic может также содержать комментарии, которые объясняют отдельные участки программы и дают важную информацию, которая может быть полезной в дальнейшем.

OOo Basic предоставляет два метода вставки комментариев в код программы:

- Все символы, которые следуют за апострофом, рассматриваются как комментарий:

  ```
  Dim A ' Это комментарий переменной A
  ```

- Ключевое слово `Rem`, сопровождает комментарий:

  ```
  Rem Этот комментарий введен ключевым словом Rem.
  ```

Комментарий обычно включает все символы до конца строки. OOo Basic тогда интерпретирует следующую строку как очередную инструкцию. Если комментарии занимают несколько строк, каждая строка должна идентифицироваться как комментарий:

```
Dim B      ' Этот комментарий для переменной B достаточно
           ' длинный и разбит на несколько строк.
           ' Символ комментария должен поэтому повторяться
           ' на каждой строке.
```

Идентификаторы

Программа OOo Basic может содержать множество, сотни, или даже тысячи идентификаторов, которые являются именами переменных, констант, функций, и так далее. Когда Вы выбираете имя для идентификатора, применяются следующие правила:

- Идентификаторы могут содержать только латинские буквы, числа, и символ подчеркивания (_);
- Первый символ идентификатора должен быть буквой или символом подчеркивания;
- Идентификаторы не могут содержать специальные символы, такие как ä â î ß;
- Максимальная длина идентификатора - 255 символов;
- Не делается никаких различий между заглавными и строчными символами.

Идентификатор `OneTestVariable`, например, определяет ту же самую переменную что `onetestVariable` и `ONETESTVARIABLE`.

Есть, однако, одно исключение из этого правила: сделано различие между заглавными и строчными символами для констант UNO-API. Более подробная информация об UNO дается в Главе 4.

Примечание Правила построения идентификаторов отличаются в OOo Basic и VBA. Например, OOo Basic разрешает использование специальных символов в идентификаторах только при использовании `Option Compatible`, так как они могут вызвать проблемы в международных проектах.

Вот несколько примеров правильных и неправильных идентификаторов:

```
Surname         ' Правильный
Surname5        ' Правильный (5 не первая цифра)
First Name      ' Неправильный (пробел не допускается)
DéjàVu          ' Неправильный (буквы, такие как é, à не допускаются)
5Surnames       ' Неправильный (первый символ не должен быть цифрой)
First,Name      ' Неправильный (запятые и точки не допускаются)
```

Работа с переменными

Неявное объявление переменных

Языки Basic разработаны, чтобы быть простыми в использовании. В результате OOo Basic позволяет создавать переменные через простое использование и без явного объявления. Другими словами, переменная существует с момента, когда Вы включаете ее в ваш код. В зависимости от переменных, которые уже присутствуют, следующий пример объявляет до трех новых переменных:

```
a = b + c
```

Неявное объявление переменных – плохая практика программирования, потому что она может привести к небрежному введению новой переменной через, например, опечатку. Вместо того, чтобы выдать сообщение об ошибке, интерпретатор создаст новую переменную, не связанную с существующим именем, со значением 0. Может быть очень трудно определить местонахождение ошибок этого вида в вашем коде.

Явное объявление переменных

Чтобы предотвращать ошибки, вызванные неявным объявлением переменных, OOo Basic предоставляет переключатель объявлений:

```
Option Explicit
```

Он должен быть записан в первой строке программы каждого модуля и гарантирует, что появится сообщение об ошибке, если одна из используемых переменных не объявлена. Переключатель `Option Explicit` должен быть включен во все Basic модули.

В своей самой простой форме, команда для явного объявления переменной следующая:

```
Dim MyVar
```

Этот пример объявляет переменную с именем `MyVar` и типом variant. Variant –

универсальная переменная, которая может делать запись всех мыслимых значений, включая строковые, целочисленные, вещественные и логические значения. Вот несколько примеров различных переменных:

```
MyVar = "Hello World"          ' присвоение строки
MyVar = 1                      ' присвоение целого числа
MyVar = 1.0                    ' присвоение вещественного числа
MyVar = True                   ' присвоение логического значения
```

Переменные, объявленные в предыдущем примере могут использоваться даже для различных типов переменных в одной и той же программе. Хотя это обеспечивает значительную гибкость, лучше ограничивать переменную определенным типом. Тип переменной определяет то, что она может хранить. Когда OOo Basic встречает переменную с неправильно объявленным типом в соответствующем контексте, появляется сообщение об ошибке.

Используйте следующий стиль, когда Вы объявляете переменную определенного типа:

```
Dim MyVar As Integer ' объявление переменной типа integer
```

Переменная объявлена как тип integer и может делать запись значений целых чисел. Вы можете также использовать следующий стиль для объявления переменной типа integer:

```
Dim MyVar%           ' объявление переменной типа integer
```

Инструкция `Dim` может содержать несколько объявлений переменных:

```
Dim MyVar1, MyVar2
```

Если Вы хотите назначить переменным постоянный тип, Вы должны сделать отдельные назначения для каждой переменной:

```
Dim MyVar1 As Integer, MyVar2 As Integer
```

Если Вы не объявляете тип для переменной, OOo Basic назначает переменной тип `Variant`. Например, в следующем объявлении переменных, `MyVar1` становится переменной типа `Variant`, а `MyVar2` становится переменной типа `Integer`:

```
Dim MyVar1, MyVar2 As Integer
```

Следующие разделы перечисляют типы переменных, которые являются доступными в OOo Basic и описывают, как они могут использоваться и быть объявлены.

Строки

Строки вместе с числами, формируют самые важные базовые типы OOo Basic. Строка состоит из последовательности упорядоченных отдельных символов. Компьютер хранит строки как последовательность чисел, где каждое число представляет один определенный символ.

От набора символов ASCII к Unicode

Наборы символов приводят в соответствие символ в строке с определенным кодом (числа и символы) в таблице, которая описывает, как компьютер должен показывать строку.

Набор символов ASCII

Набор символов ASCII – ряд кодов, которые представляют числа, символы, и специальные

символы одним байтом. Коды ASCII от 0 до 127 соответствуют алфавиту и общим символам (такие как точка, круглые скобки и запятая), а так же некоторые специальные коды управления экрана и принтера. Набор символов ASCII обычно используется как стандартный формат для передачи текстовых данные между компьютерами.

Однако, этот набор символов не включает диапазон специальных символов, используемых в Европе, таких как â, ä, и î, так же как символов других форматов, таких как Кириллица.

Набор символов ANSI

Microsoft базировала свой продукт Windows на наборе символов Американского Национального Института Стандартов (ANSI), который был постепенно расширен, чтобы включить символы, отсутствующие в наборе символов ASCII.

Кодовые Страницы

Наборов символов ISO 8859 предусмотрен международным стандартом. Первые 128 символов набора символов ISO соответствуют набору символов ASCII. Стандарт ISO вводит новые наборы символов (*кодовые страницы*) так, чтобы могло быть правильно отображено большинство языков. Однако, в результате те же самые символьные значения могут представить различные символы в различных языках.

Unicode

Unicode увеличивает длину символа до четырех байт и объединяет различные наборы символов для создания стандарта, который может отображать столько много из всемирных языков насколько это возможно. Версия 2.0 Unicode теперь поддержана во многих программах – включая OpenOffice.org и OOo Basic.

Строковые переменные

OOo Basic хранит строки как строковые переменные в Unicode. Строковая переменная может содержать до 65535 символов. Внутренне, OOo Basic хранит соответствующие значения Unicode для каждого символа. Количество памяти, необходимое для строковой переменной зависит от длины строки. Символ объявления типа для строковой переменной – $.

Пример объявления строковой переменной:

```
Dim Variable As String
```

Вы можете также написать это объявление как:

```
Dim Variable$
```

Примечание	При переносе приложений VBA, гарантируйте, что максимальная возможная длина строки OOo Basic соблюдается.

Явное описание строки

Чтобы явно присвоить строку строковой переменной, заключите строку в кавычки (").

```
Dim MyString As String
MyString = "Это тест"
```

Чтобы разбить строку на две строки программы, добавьте знак плюс в конце первой строки программы:

```
Dim MyString As String
MyString = "Эта строка является такой длинной, что она" + _
           "была разбита на две строки программы."
```

Чтобы включить кавычку (") в строку, введите ее дважды в нужном месте:

```
Dim MyString As String
MyString = "a ""-quotation mark." ' результат a "-quotation mark
```

Числа

OOo Basic поддерживает пять основных типов для обработки чисел:

- Целочисленные переменные, типа Integer;
- Целочисленные переменные двойной точности, типа Long Integer;
- Вещественные переменные одинарной точности, типа Single;
- Вещественные переменные двойной точности, типа Double;
- Денежные переменные, типа Currency.

Переменные типа Integer

Переменные типа Integer (целочисленные) могут хранить любое целое число между -32768 и 32767. Целочисленная переменная занимает два байта памяти. Символ объявления типа для целочисленной переменной – %. Вычисления, которые используют целочисленные переменные, очень быстры и особенно полезны для счетчиков циклов. Если Вы присваиваете вещественное число целочисленной переменной, оно округляется вниз или вверх к ближайшему целому числу.

Пример объявления для целочисленных переменных:

```
Dim Variable As Integer
Dim Variable%
```

Переменные типа Long Integer

Переменные типа Long Integer (длинное целое) могут хранить любое целое число между -2147483648 и 2147483647. Переменная типа Long Integer занимает четыре байта памяти. Символ объявления типа для переменной типа Long Integer – &. Вычисления, которые используют переменные типа Long Integer, очень быстры и особенно полезны для счетчиков циклов. Если Вы присваиваете вещественное число переменной типа Long Integer, оно округляется вниз или вверх к ближайшему целому числу.

Пример объявления для переменных типа Long Integer:

```
Dim Variable as Long
```

```
Dim Variable&
```

Переменные типа Single

Переменные типа Single могут хранить любое положительное или отрицательное вещественное число между 3.402823×10^{38} и 1.401298×10^{-45}. Переменная типа Single занимает четыре байта памяти. Символ объявления типа для переменной типа Single – !.

Первоначально, переменные типа Single использовались, чтобы уменьшить время вычисления, требуемое для более точных переменных типа Double. Однако, эти рассуждения о скорости больше не применимы, что уменьшает потребность в переменных типа Single.

Пример объявления для переменных типа Single:

```
Dim Variable as Single
Dim Variable!
```

Переменные типа Double

Переменные типа Double могут хранить любые положительные или отрицательные вещественные числа между $1.79769313486232\times10^{308}$ и $4.94065645841247\times10^{-324}$. Переменная типа Double занимает восемь байтов памяти. Переменные типа Double являются подходящими для точных вычислений. Символ определения типа – #.

Пример объявления вещественных переменных двойной точности:

```
Dim Variable As Double
Dim Variable#
```

Переменные типа Currency

Переменные типа Currency отличаются от других переменных типов по тому, как они обращаются со значениями. Десятичная точка фиксирована и сопровождается четырьмя десятичными знаками. Переменная может содержать до 15 знаков перед десятичной точкой. Переменная типа Currency может хранить любое значение между -922337203685477.5808 и +922337203685477.5807 и занимает до восьми байтов памяти. Символ объявления типа для переменной типа Currency – @.

Пример объявления переменных типа Currency:

```
Dim Variable As Currency
Dim Variable@
```

Задание явных чисел

Числа могут быть представлены несколькими способами, например, десятичным форматом или в научной нотации, или даже с отличным от десятичной системы основанием. Следующие правила применяются к числовым символам в OOo Basic:

Целые Числа

Самый простой метод должен работать с целыми числами. Они перечисляются в исходном тексте без запятой – разделителя тысяч:

```
Dim A As Integer
```

```
Dim B As Float

A = 1210
B = 2438
```

Числам могут предшествовать знаки плюс (+) или минус (-) (с или без пробела между ними):

```
Dim A As Integer
Dim B As Float

A = + 121
B = - 243
```

Десятичные Числа

Когда Вы вводите десятичное число, используете точку (.) как десятичный разделитель. Это правило гарантирует, что исходные тексты могут быть переданы из одной страны в другую без преобразования.

```
Dim A As Integer
Dim B As Integer
Dim C As Float

A = 1223.53       ' округляется
B = - 23446.46    ' округляется
C = + 3532.76323
```

Вы можете также использовать знаки плюс (+) или минус (-) как префиксы для десятичных чисел (снова с или без пробела).

Если десятичное число присваивается переменной целочисленного типа, OOo Basic округляет число вверх или вниз.

Экспоненциальный стиль записи

OOo Basic позволяет определять числа в экспоненциальном стиле записи, например, Вы можете написать $1.5e^{-10}$ для числа 1.5×10^{-10} (0.00000000015). Буква "e" может быть строчной или прописной буквой с или без знака плюс (+) в качестве префикса.

Вот несколько правильных и неправильных примеров чисел в экспоненциальном формате:

```
Dim A As Double

A = 1.43E2    ' Правильно
A = + 1.43E2  ' Правильно (пробел между плюсом и основным числом)
A = - 1.43E2  ' Правильно ( пробел между минусом и основным числом)
A = 1.43E-2   ' Правильно (отрицательный показатель)

A = 1.43E -2  ' Incorrect (пробел не допускается в пределах числа)
A = 1,43E-2   ' Incorrect (запятые не разрешены как десятичный
              ' разделитель)
A = 1.43E2.2  ' Incorrect (показатель должен быть целым числом)
```

Заметьте, что в первом и третьем неправильных примерах, никакого сообщения об ошибке не выдается даже при том, что переменные возвращают неправильные значения. Выражение

```
A = 1.43E -2
```

интерпретируется как 1.43 минус 2, которое соответствует значению -0.57. Однако, значение 1.43×10^{-2} (соответствует 0.0143) было планируемым значением. В значении

```
A = 1.43E2.2
```

OOo Basic игнорирует часть показателя после десятичной точки и интерпретирует выражение как

```
A = 1.43E2
```

Шестнадцатеричные значения

В шестнадцатеричной системе счисления, число с двумя цифрами соответствует точно одному байту. Это позволяет обрабатывать числам способом, который более близко отражает архитектуру машины. В шестнадцатеричной системе, числа от 0 до 9 и буквы от A до F используются как числа. A соответствует десятичному числу 10, в то время как буква F представляет десятичное число 15. OOo Basic позволяет Вам использовать целые числовые шестнадцатеричные значения, когда им предшествует &H.

```
Dim A As Long

A = &HFF ' Шестнадцатиричное значение FF, соответствует
         ' десятичному значению 255
A = &H10 ' Шестнадцатиричное значение 10, соответствует
         ' десятичному значению 16
```

Восьмеричные значения

OOo Basic также понимает восьмеричную систему счисления, которая использует числа от 0 до 7. Вы должны использовать целые числа, которым предшествует &O.

```
Dim A As Long

A = &O77 ' Восьмеричному значению 77, соответствует
         ' десятичное значение 63
A = &O10 ' Восьмеричному значению 10, соответствует
         ' десятичное значение 8
```

True и False

Логические переменные

Логические переменные могут содержать только одно из двух значений: **True** (Истина) и **False** (Ложь). Они являются подходящими для двоичных спецификаций и могут принимать только одно из двух состояний. Логическое значение внутренне хранится как двухбайтовое значение целого числа, где 0 соответствует False, а любое другое значение True. Не существует символа объявления типа для логических переменных. Объявление может быть сделано только с использованием дополнения *As Boolean*.

Пример объявления логической переменной:

```
Dim Variable As Boolean
```

Подробнее о дате и времени

Переменные Даты

Переменные даты могут содержать значения времени и даты. Для сохранения значения даты, OOo Basic использует внутренний формат, который разрешает сравнение и математические операции со значениями времени и даты. Нет никакого символа объявления типа для переменных даты. Объявление может быть сделано только с использованием дополнения *As Date*.

Пример объявления переменной даты:

```
Dim Variable As Date
```

Наборы данных

В дополнение к простым переменным (скалярным), OOo Basic также поддерживает наборы данных (массивы). Наборы данных содержит несколько переменных, к которым обращаются по индексу.

Простые массивы

Объявление массива подобно объявлению обычной переменной. Однако, в отличие от объявления переменной, имя массива сопровождается круглыми скобками, в которых указывается число элементов. Выражение

```
Dim MyArray(3)
```

объявляет множество, которое имеет четыре переменные типа variant, а именно,

`MyArray(0), MyArray(1), MyArray(2) и MyArray(3).`

Вы можете также объявить в массиве переменные определенного типа. Например, следующая строка объявляет массив с четырьмя целочисленными переменными:

```
Dim MyInteger(3) As Integer
```

В предыдущих примерах, индекс для массива всегда начинается со стандартного начального значения ноль. Как альтернатива, достоверный диапазон со значениями начала и конца может быть определен в объявлении массива данных. Следующий пример объявляет набор данных, который имеет шесть целочисленных значений и к которому можно обратиться, используя индексы от 5 до 10:

```
Dim MyInteger(5 To 10) As Integer
```

Индексы не обязательно должны быть положительными значениями. Следующий пример показывает также правильное объявление, но с отрицательными пределами набора данных:

```
Dim MyInteger(-10 To -5) As Integer
```

В этом примере объявляется набор целочисленных данных с 6 значениями, к которым можно обращаться, используя индексы от -10 до -5.

Есть три предела, которые Вы должны соблюдать, когда Вы определяете индексы набора данных:

- наименьший индекс – -32768;
- наибольший индекс – 32767;
- максимальное число элементов (в пределах размера набора данных) – 16368.

Примечание Другие значения пределов иногда применяются для индексов набора данных в VBA. То же самое также применимо к максимальному числу элементов, допустимых в массиве. Значения, допустимые там могут быть найдены в соответствующей документации по VBA.

Задание значения для начального индекса

Начальный индекс набора данных обычно имеет значение 0. Вы можете изменить начальный индекс для всех объявлений наборов данных на значение 1 используя команду:

```
Option Base 1
```

Эта команда должна быть включена в заголовок модуля, если Вы хотите, чтобы она применялась ко всем объявлениям массивов в модуле. Однако, эта команда не затрагивает UNO массивы, которые определяются через OpenOffice.org API, индекс которых *всегда* начинается с 0. Чтобы не запутаться, Вы должны избегать использования `Option Base 1`.

Число элементов в массиве не изменяется, если Вы используете `Option Base 1`, только изменяется начальный индекс. Объявление

```
Option Base 1
' ...
Dim MyInteger(3) As Integer
```

создает 4 целочисленные переменные, которые могут быть описаны в выражениях `MyInteger(1)`, `MyInteger(2)`, `MyInteger(3)` и `MyInteger(4)`.

Примечание В OOo Basic выражение `Option Base 1` не затрагивает число элементов в массиве, как это делается в VBA. Это, скорее начальный индекс, который смещается в OOo Basic. В то время как объявление, `MyInteger (3)` создает три целочисленных значения в VBA с индексами 1-3, то же самое объявление в OOo Basic, создает четыре целочисленных значения с индексами 1-4. При использовании `Option Compatible`, OOo Basic ведет себя как VBA.

Многомерные наборы данных

В дополнение к одномерным наборам данных, OOo Basic также поддерживает работу с многомерными наборами данных. Соответствующие измерения отделены друг от друга запятыми. Например

```
Dim MyIntArray(5, 5) As Integer
```

определяет целочисленный массив с двумя измерениями, каждое с 6-ю индексами (можно обращаться через индексы от 0 до 5). Весь массив может осуществлять запись в общей сложности 6 × 6 = 36 целочисленных значений.

Динамическое изменение размерности наборов данных

Предыдущие примеры основаны на наборах данных определенной размерности. Вы можете также определять массивы, в которых динамически изменяется размерность наборов данных. Например, Вы можете определить массив, который содержит все слова в тексте, которые начинаются с буквы А. Поскольку число этих слов первоначально неизвестно, Вы должны быть в состоянии впоследствии изменить пределы набора. Чтобы выполнить это в OOo Basic, используйте следующую команду:

```
ReDim MyArray(10)
```

Примечание В отличие от VBA, где Вы можете изменять только динамические массивы объявленные при использовании `Dim MyArray()`, OOo Basic позволяет Вам изменять и статические и динамические массивы, используя `ReDim`.

Следующий пример изменяет размерность изначального массива так, чтобы можно было сделать запись 11 или 21 значений:

```
Dim MyArray(4) As Integer ' Объявление с пятью элементами
' ...
ReDim MyArray(10) As Integer ' Увеличение до 11 элементов
' ...
ReDim MyArray(20) As Integer ' Увеличение до 21 элементов
```

Когда Вы восстанавливаете размерность массива, Вы можете использовать любой из вариантов, указанных в предыдущих разделах. Они включают объявление многомерных наборов данных и определения явного начального и конечного значения. Когда размерность набора данных изменяется, все содержимое теряется. Если Вы хотите сохранить оригинальные значения, используйте команду `Preserve`:

```
Dim MyArray(10) As Integer ' Определение начальной
                           ' размерности
' ...
ReDim Preserve MyArray(20) As Integer ' Увеличение набора данных
                                      ' с сохранением содержимого
```

Когда Вы используете команду `Preserve`, гарантируйте, что число измерений и тип переменных останется тем же самым.

Примечание	В отличие от VBA, где только верхний предел последнего измерения набора данных может быть изменен через команду `Preserve`, OOo Basic позволяет Вам изменять также и другие измерения.

Область действия и время жизни переменных

Переменная в OOo Basic имеет ограниченное время жизни и ограниченную область действия, в которых она может быть прочитана и использоваться в других фрагментах программы. Время, в течение которого переменная сохраняется, так же как откуда к ней можно получить доступ, зависит от ее определенного местоположения и типа.

Локальные переменные

Переменные, которые объявлены в функции или процедуре, называют локальными переменными:

```
Sub Test
  Dim MyInteger As Integer

  ' ...

End Sub
```

Локальные переменные остаются действительными только пока функция или процедура выполняются, и затем сбрасываются в ноль. Каждый раз при вызове функции, значения, вычисленные прежде не доступны.

Чтобы сохранить предыдущее значение, Вы должны определить переменную как Статическую:

```
Sub Test
  Static MyInteger As Integer

  ' ...
```

```
End Sub
```

Примечание В отличие от VBA, OOo Basic гарантирует, что имя локальной переменной не используется одновременно как глобальная (`global`) и/или частная (`private`) переменная в заголовке модуля. Когда Вы преобразуете приложение VBA в OOo Basic, Вы должны изменить любые двойные имена переменных.

Общедоступные переменные

Общедоступные переменные определены в заголовке модуля ключевым словом `Dim`. Эти переменные доступны для всех модулей в их библиотеке:

Модуль A:

```
Dim A As Integer
Sub Test
  Flip
  Flop
End Sub

Sub Flip
  A = A + 1
End Sub
```

Модуль B:

```
Sub Flop
  A = A - 1
End Sub
```

Значение переменной `A` не изменяется функцией `Test`, оно увеличивается на единицу в функции `Flip` и уменьшается на единицу в функции `Flop`. Оба из этих изменений для переменной глобальны.

Вы можете также использовать ключевое слово `Public` вместо `Dim`, для объявления общедоступной переменной:

```
Public A As Integer
```

Общедоступная переменная доступна только пока связанный макрос выполняется, и затем переменная сбрасывается.

Глобальные Переменные

В отношении функции, глобальные переменные подобны общедоступным переменным, за исключением того, что их значение сохраняется даже после того, как связанный макрос выполнился. Глобальные переменные объявляются в заголовке модуля, с использованием ключевого слова `Global`:

```
Global A As Integer
```

Частные Переменные

Частные переменные доступны только в модуле, в котором они определены. Используйте ключевое слово `Private` для определения переменной:

```
Private MyInteger As Integer
```

Если несколько модулей содержат частную переменную с одинаковым именем, OOo Basic

создает различные переменные для каждого местонахождения имени. В следующем примере, модули A и B имеют частную переменную по имени C. Функция `Test` сначала устанавливает частную переменную в модуле A, а затем частную переменную в модуле B.

Модуль A:

```
Private C As Integer

Sub Test
  SetModuleA ' Устанавливаем переменную C из модуля A
  SetModuleB ' Устанавливаем переменную C из модуля B

  ShowVarA   ' Отображаем переменную C из модуля A (= 10)
  ShowVarB   ' Отображаем переменную C из модуля B (= 20)
End Sub

Sub SetmoduleeA
  A = 10
End Sub

Sub ShowVarA
  MsgBox C   ' Отображаем переменную C из модуля A.
End Sub
```

Модуль B:

```
Private C As Integer

Sub SetModuleB
  A = 20
End Sub

Sub ShowVarB
  MsgBox C   ' Отображаем переменную C из модуля B.
End Sub
```

Константы

Константа – это область памяти, которая имеет неизменяемое значение во время выполнения программы. Константы бывают числовыми, строковыми и логическими. OOo Basic имеет две предопределенные логические константы: `True` (Истина) и `False` (Ложь) и числовую константу `PI`.

В OOo Basic, используется ключевое слово `Const` для объявления констант.

```
Const A = 10
```

Вы можете также определить тип константы в объявлении:

```
Const B As Double = 10
```

Операции

OOo Basic понимает общие математические, логические и операции сравнения.

Математические операции

Математические операции могут применяться ко всем типам чисел, тогда как оператор + может также использоваться для связывания строк.

+	Сложение чисел и значений дат, объединение строк

-	Вычитание чисел и значений дат
*	Умножение чисел
/	Деление чисел
\	Деление чисел с целочисленным результатом (округлением)
^	Возведение чисел в степень
MOD	Вычисление остатка от деления двух операндов

Логические операции

Логические операции позволяют Вам связывать элементы согласно правилам Булевой алгебры. Если операция применяется к Булевым значениям, соединение обеспечивает результат, требуемый непосредственно. Если используется в соединении со значениями целых чисел и длинных целых чисел, соединение осуществляется на уровне битов.

AND	И соединение (Логическое умножение)
OR	ИЛИ соединение (Логическое сложение)
XOR	Соединение исключающее ИЛИ
NOT	Отрицание
EQV	Тест на эквивалентность (обе части, `True` или `False`)
IMP	Импликация (если первое выражение `True`, то второе должно также быть `True`)

Операции сравнения

Операции сравнения могут применяться ко всем элементарным типам переменных (числа, элементы даты, строки и логические значения).

=	Равенство чисел, значений дат и строк
<>	Неравенство чисел, значений дат и строк
>	Проверка больше чем для чисел, значений дат и строк
>=	Проверка больше чем или равно для чисел, значений дат и строк
<	Проверка меньше чем для чисел, значений дат и строк
<=	Проверка меньше чем или равно для чисел, значений дат и строк

Примечание OOo Basic не поддерживает VBA операцию сравнения `Like`.

Строковые операции

Строковая операция только одна – конкатенация (сцепление) строк. Она во многом

используется также, как операция сложения. Однако, в отличие от последнего, результатом операции сцепления строк всегда является строка, которая состоит из строк-операндов. В качестве знака операции используется &.

Приоритет операций

При работе со сложными выражениями, содержащими множество операций, OpenOffice.org должен определить, какую из операций выполнять первой, какую второй и т.д. Для этого вводится понятие *приоритета операций*. Операции имеющие одинаковый приоритет выполняются слева направо в порядке записи. Порядок выполнения операций может быть изменен с использованием скобок.

Приоритет	Операции
4	^
3	*, /, MOD
2	+, -, &
1	=, <, >, <>, <=, >=
0	NOT, AND, OR, XOR, EQV, IMP

Ветвление

Используйте операторы ветвления для ограничения выполнения блока кода пока конкретное условие выполняется.

If...Then...Else

Самый общий выполняющий ветвление оператор – оператор `If` как показано в следующем примере:

```
If A > 3 Then
  B = 2
End If
```

Присваивание `B = 2` происходят только тогда, когда значение переменной `A` больше чем три. Вариант оператора `If` - оператор `If/Else`:

```
If A > 3 Then
  B = 2
Else
  B = 0
End If
```

В этом примере, переменной `B` присваивается значение 2, когда `A` больше чем 3, в противном случае `B` присваивается значение 0.

Для более сложных утверждений, Вы можете применять каскадирование оператора `If`, например:

```
If A = 0 Then
  B = 0
ElseIf A < 3 Then
  B = 1
```

```
Else
  B = 2
End If
```

Если значение переменной A равняется нулю, B присваивается значение 0. Если A - меньше чем 3 (но не равно нулю), то B присваивается значение 1. Во всех других случаях (то есть, если A больше чем или равен 3), B присваивается значение 2.

Select...Case

Инструкция Select...Case альтернатива каскадному оператору If и используется, когда Вы должны проверить значение на различные условия:

```
Select Case DayOfWeek
  Case 1:
    NameOfWeekday = "Понедельник"
  Case 2:
    NameOfWeekday = "Вторник"
  Case 3:
    NameOfWeekday = "Среда"
  Case 4:
    NameOfWeekday = "Четверг"
  Case 5:
    NameOfWeekday = "Пятница"
  Case 6:
    NameOfWeekday = "Суббота"
  Case 7:
    NameOfWeekday = "Воскресенье"
End Select
```

В этом примере, название дня недели присваивается переменной NameOfWeekday в зависимости от значения переменной DayOfWeek. Если оно равно 1 то переменной NameOfWeekday присваивается значение Понедельник, если 2 значение Вторник, и так далее.

Команда Select не ограничивается простым назначением 1:1 – Вы можете также определить операции сравнения или списки выражений в ветви Case.

- в большинстве случаев используется константа, типа `Case 4` или `Case "Привет"`;
- Несколько значений могут быть заданы, разделенными запятыми: `Case 3, 5, 7`;
- Если Вы хотите проверить диапазон значений, есть ключевое слово To

  `Case 5 To 10`;
- Открытые диапазоны могут быть проверены следующим образом

  `Case < 10` или с использованием ключевого слова Is – `Case Is < 10`;
- Если никакое условие не выполняется, то будет работать дополнительный `Case Else`.

Внимание Будьте осторожны, используя диапазон в утверждении Case. Помощь онлайн неоднократно содержала неправильные примеры, такие как `Case i > 2 AND i < 10`. Это трудно понять и закодировать правильно.

Эндрю Питоняк в своей книге *Useful Macro Information for OpenOffice* утверждает, что подобная конструкция работает некорректно.

Следующий пример перечисляет самые важные варианты синтаксиса:

```
Select Case Var
  Case 1 To 5
    ' ... Var между числами 1 и 5
  Case 6, 7, 8
```

```
    ' ... Var – 6, 7 или 8
  Case Is > 8
    ' ... Var больше чем 8
  Case Else
    ' ... все другие случаи
End Select
```

Циклы

Цикл выполняет кодовый блок для определенного числа проходов. Вы можете также иметь циклы с неопределенным числом проходов.

For...Next

Цикл For...Next имеет заранее известное число проходов. Число итераций отслеживается с помощью переменной – счетчика цикла. В следующем примере,

```
Dim I
For I = 1 To 10
  ' ... Внутренняя часть цикла
Next I
```

переменная `I` – счетчик цикла, с начальным значением 1. Счетчик увеличивается на 1 в конце каждого прохода. Когда переменная `I` равняется 10, цикл останавливается.

Если Вы хотите увеличить счетчик цикла на значение отличное от 1 в конце каждого прохода, используйте ключевое слово `Step`:

```
Dim I
For I = 1 To 10 Step 0.5
  ' ... Внутренняя часть цикла
Next I
```

В предыдущем примере, счетчик увеличивается на 0.5 в конце каждого прохода, и цикл выполняется 19 раз.

Вы можете также использовать отрицательное значение шага:

```
Dim I
For I = 10 To 1 Step -1
  ' ... Внутренняя часть цикла
Next I
```

В этом примере, счетчик начинается с 10 и уменьшается на 1 в конце каждого прохода, пока счетчик не станет равным 1.

Инструкция `Exit For` позволяет Вам завершать цикл `For` преждевременно. В следующем примере, цикл заканчивается во время пятого прохода:

```
Dim I
For I = 1 To 10
  If I = 5 Then
    Exit For
  End If
  ' ... Внутренняя часть цикла
Next I
```

Примечание Вариант цикла `For Each...Next` из VBA не поддерживается в OOo Basic

Do...Loop

Цикл `Do...Loop` не связан с заданным заранее числом проходов. Вместо этого цикл `Do...Loop` выполняется, пока определенное условие не удовлетворено. Есть четыре варианта цикла `Do...Loop` (в следующих примерах, `A> 10` представляет любое условие):

1. Вариант `Do While...Loop`

   ```
   Do While A > 10
      ' ... тело цикла
   Loop
   ```

   проверяет, выполняется ли условие перед каждым проходом и только тогда выполняет цикл.

2. Вариант `Do Until...Loop`

   ```
   Do Until A > 10
      ' ... тело цикла
   Loop
   ```

   выполняет цикл, пока условие *не выполняется*.

3. Вариант `Do...Loop While`

   ```
   Do
      ' ... тело цикла
   Loop While A > 10
   ```

   проверяет условие только после того, как первый проход цикла выполнен, и продолжает выполнение цикла, если это условие *выполняется*.

4. Вариант `Do...Loop Until`

   ```
   Do
      ' ... тело цикла
   Loop Until A > 10
   ```

   также проверяет условие после первого прохода, но продолжает выполнение цикла, пока условие *не выполняется*.

Как и в цикле `For...Next`, цикл `Do...Loop` также имеет команду завершения. Команда `Exit Do` может завершить цикл в любой точке в пределах цикла.

```
Do
  If A = 4 Then
    Exit Do
  End If
  ' ... тело цикла
While A > 10
```

While...Wend

Нет ничего специального в конструкции `While...Wend`, она имеет следующую форму:

```
While A > 10
   ' ... тело цикла
Wend
```

Цикл выполняется до тех пор, пока условие истинно. В примере цикл выполняется до тех пор пока значение переменной `A` больше 10.

Эта конструкция имеет ограничения, которых не существует в конструкции `Do While...Loop` и не предлагает никаких других выгод. Вы не можете использовать команду `Exit`, не можете осуществить выход при помощи `GoTo`.

Пример программирования: Сортировка с вложенными циклами

Есть много способов использования циклов, например, поиск по спискам, возвращение значения, или выполнение сложных математических задач. Следующий пример - алгоритм, который использует цикл для сортировки списка имен.

```
Sub Sort
  Dim Entry(1 To 10) As String
  Dim Count As Integer
  Dim Count2 As Integer
  Dim Temp As String

  Entry(1) = "Patty"
  Entry(2) = "Kurt"
  Entry(3) = "Thomas"
  Entry(4) = "Michael"
  Entry(5) = "David"
  Entry(6) = "Cathy"
  Entry(7) = "Susie"
  Entry(8) = "Edward"
  Entry(9) = "Christine"
  Entry(10) = "Jerry"

  For Count = 1 To 10
    For Count2 = Count + 1 To 10
      If Entry(Count) > Entry(Count2) Then
        Temp = Entry(Count)
        Entry(Count) = Entry(Count2)
        Entry(Count2) = Temp
      End If
    Next Count2
  Next Count

  For Count = 1 To 10
    Print Entry(Count)
  Next Count
End Sub
```

Пары обмениваются значениями несколько раз, пока они наконец не отсортированы в порядке возрастания. Как пузыри, переменные постепенно перемещаются к правильному положению. Поэтому этот алгоритм также известен как *пузырьковая сортировка*.

Процедуры и Функции

Процедуры и функции формируют основные элементы в структуре программы. Они обеспечивают структуру для деления сложной задачи на несколько подзадач.

Процедуры

Процедура выполняет действие, не обеспечивая явного возврата значений. Ее синтаксис

```
Sub Test
  ' ... здесь реальный код процедуры
End Sub
```

Пример определяет процедуру по имени `Test`, которая содержит код, к которому можно получить доступ из любой точки в программе. Запрос делается, вводом имени процедуры в нужной точке программы:

```
Test
```

Функции

Функция, точно так же как процедура, объединяет блок программы, который будет выполнен как одна логическая единица. Однако, в отличие от процедуры, функция обеспечивает возврат значения.

```
Function Test
  ' ... здесь реальный код функции
  Test = 123
End Function
```

Возвращаемое значение определяется с использованием простого присваивания. Присваивание не обязательно должно помещаться в конце функции, оно может быть сделано где-нибудь внутри функции.

Предыдущую функцию можно вызвать в пределах программы следующим образом:

```
Dim A

A = Test
```

Код определяет переменную `A` и присваивает ей результат, возвращаемый функцией `Test`.

Возвращаемое значение может переприсваиваться в пределах функции несколько раз. Как с классическим именем переменной, функция в этом примере возвращает значение, которое было присвоено ей последним.

```
Function Test
  Test = 12
  ' ...
  Test = 123
End Function
```

В этом примере, возвращаемое значение функции – 123.

Если присваивания не происходит, функция возвращает нулевое значение (число 0 для числового значения и пустую строку для строк).

Возвращаемое значение функции может быть любым типом. Тип объявляется таким же образом как объявление переменной:

```
Function Test As Integer
  ' ... здесь реальный код функции
End Function
```

Если явное объявление типа значения отсутствует, тип возвращаемого значения объявляется как `Variant`.

Преждевременное завершение процедур и функций

В OOo Basic Вы можете использовать команды `Exit Sub` и `Exit Function` для преждевременного завершения процедуры или функции, например, при обработке ошибок. Эти команды останавливают выполнение процедуры или функции и возвращают управление программы в точку, из которой была вызвана процедура или функция.

Следующий пример показывает процедуру, которая заканчивает выполнение, когда переменная `ErrorOccured` имеет значение `True`.

```
Sub Test
  Dim ErrorOccured As Boolean
  ' ...
  If ErrorOccured Then
    Exit Sub
  End If
  ' ...
```

```
End Sub
```

Передача параметров

Функции и процедуры могут получать один или более параметров. Основные параметры должны быть заключены в круглые скобки после имени процедуры или функции. Например

```
Sub Test (A As Integer, B As String)
End Sub
```

определяет процедуру, которая ожидает в качестве параметров целочисленное значение `A` и строку `B`.

Параметры в OOo Basic обычно передаются *по ссылке*. Изменения, сделанные с переменным сохраняются при выходе из процедуры или функции:

```
Sub Test
  Dim A As Integer
  A = 10
  ChangeValue(A)
  ' Параметр A теперь имеет значение 20
End Sub

Sub ChangeValue(TheValue As Integer)
  TheValue = 20
End Sub
```

В этом примере, значение A, которое определяется в функции `Test`, передается как параметр в функцию `ChangeValue`. Значение здесь изменяется на 20 и передается в `TheValue`, которое сохраняется, когда происходит выход из функции.

Вы можете также передать параметр *по значению*, если Вы не хотите, чтобы последующие изменения параметра затронули значение, которое было передано первоначально. Чтобы определить, что параметр нужно передать по значению, удостоверьтесь, что ключевое слово `ByVal` предшествует объявлению переменной в заголовке функции.

В предыдущем примере, если мы заменяем функцию `ChangeValue` на функцию

```
Sub ChangeValue(ByVal TheValue As Integer)
  TheValue = 20
End Sub
```

тогда переменная A остается незатронутой этим изменением. После вызова функции `ChangeValue`, переменная A сохраняет значение 10.

Примечание Метод для передачи параметров процедурам и функциям в OOo Basic фактически идентичен VBA. По умолчанию, параметры передаются по ссылке. Чтобы передавать параметры по значению, используйте ключевое слово `ByVal`. В VBA, Вы можете также использовать ключевое слово `ByRef`, чтобы вынудить параметр быть переданным по ссылке. OOo Basic не поддерживает это ключевое слово, потому что оно уже используется по умолчанию в процедурах OOo Basic.

Необязательные параметры

Функции и процедуры можно вызвать, только если все необходимые параметры передаются во время вызова. OOo Basic позволяет Вам определять параметры как *необязательные*, то есть, если соответствующие значения не включены в запрос, OOo Basic передает пустой параметр. В примере

```
Sub Test(A As Integer, Optional B As Integer)

End Sub
```

параметр `A` обязателен, тогда как параметр `B` является необязательным.

Функция `IsMissing` проверяет, передан ли параметр или нет.

```
Sub Test(A As Integer, Optional B As Integer)
  Dim B_Local As Integer
  ' Проверка, присутствует ли параметр B фактически
  If Not IsMissing (B) Then
    B_Local = B ' Параметр B присутствует
  Else
    B_Local = 0 ' Параметр B отсутствует -> значение по умолчанию 0
  End If

  ' ... Фактическое начало функции
End Sub
```

Пример сначала проверяет, передали ли параметр `B` и, в случае необходимости, передает этот параметр внутренней переменной `B_Local`. Если соответствующий параметр отсутствует, то значение по умолчанию (в этом случае, значение 0) передается `B_Local`, а не переданному параметру.

Примечание Ключевое слово `ParamArray` присутствующее в VBA не поддерживается в OOo Basic.

Рекурсия

Рекурсия теперь возможна в OOo Basic. Рекурсивная процедура или функция – та, которая имеет способность вызывать себя, пока она не обнаруживает, что некоторое основное условие удовлетворяется. Когда функцию вызывают с основным условием, возвращается результат.

Следующий пример использует рекурсивную функцию для вычисления факториала чисел `42`, `-42`, и `3.14`:

```
Sub Main
  Msgbox CalculateFactorial( 42 )    ' отображает 1,40500611775288E+51
  Msgbox CalculateFactorial( -42 )   ' Отображает "Неверное число для
                                     ' факториала!"
  Msgbox CalculateFactorial( 3.14 ) ' Отображает "Неверное число для
                                     ' факториала!"
End Sub

Function CalculateFactorial(Number)
  If Number < 0 Or Number <> Int(Number) Then
    CalculateFactorial = "Неверное число для факториала!"
  ElseIf Number = 0 Then
    CalculateFactorial = 1
  Else
    ' Это рекурсивный вызов:
    CalculateFactorial = Number * CalculateFactorial(Number - 1)
  Endif
End Function
```

Пример возвращает факториал числа `42`, рекурсивно вызывая функцию `CalculateFactorial`, пока она не достигает основного условия `0! = 1`.

Примечание Глубины рекурсии установлены на различных уровнях, основанных на

платформе программного обеспечения. Для Windows глубина рекурсии - 5800. Для Solaris и Linux, выполняется оценка stacksize, и вычисляется глубина рекурсии.

Обработка ошибок

Правильная обработка ошибочных ситуаций – одна из наиболее трудоемких задач программирования. OOo Basic обеспечивает набор инструментов для упрощения обработки ошибок.

Инструкция On Error

Инструкция `On Error` – ключ к любой обработке ошибок:

```
Sub Test
  On Error Goto ErrorHandler
  ' ... выполнение задачи, в которой может произойти ошибка
  Exit Sub

ErrorHandler:
  ' ... индивидуальный код для обработки ошибок
End Sub
```

Строка `On Error Goto ErrorHandler` определяет как OOo Basic продолжает выполнение в случае возникновения ошибки. `Goto ErrorHandler` гарантирует, что OOo Basic прерывает выполнение текущей строки программы и затем выполняет код `ErrorHandler:`.

Команда Resume

Команда `Resume Next` продолжает программу со строки, которая следует за строкой в программе, где произошла ошибка, после того, как был выполнен код обработчика ошибок:

```
ErrorHandler:
  ' ... индивидуальный код для обработки ошибок
  Resume Next
```

Используйте команду `Resume Proceed`, чтобы определить точку перехода для продолжения программы после обработки ошибок:

```
ErrorHandler:
  ' ... индивидуальный код для обработки ошибок
  Resume Proceed

Proceed:
  ' ... программа продолжается здесь после ошибки
```

Для продолжения программы без сообщения об ошибке, в случае возникновения ошибки, используйте следующий формат:

```
Sub Test
  On Error Resume Next
  ' ... выполнение задачи, в течение которой может произойти ошибка
End Sub
```

Используйте команду `On Error Resume Next` с осторожностью, поскольку ее эффект глобален. За дополнительной информацией, см. “Подсказки для структурированного обработчика ошибок” на странице 27.

Вопросы относительно информации об ошибке

При обработке ошибок, полезно иметь описание ошибки и знать, где и почему произошла ошибка:

- переменная `Err` содержит номер ошибки, которая произошла;
- переменная `Error$` содержит описание ошибки;
- переменная `Erl` содержит номер строки, где произошла ошибка.

Вызов

```
MsgBox "Ошибка " & Err & ": " & Error$ & " (строка : " & Erl & ")"
```

показывает, как информация об ошибке может отображаться в окне сообщения.

Примечание Принимая во внимание, что VBA суммирует сообщения об ошибках в статистическом объекте по имени `Err`, OOo Basic обеспечивает переменные `Err`, `Error$`, и `Erl`.

Информация состояния остается действительной, пока программа не сталкивается с командами `Resume` или `On Error`, после чего информация очищается.

Примечание В VBA, метод `Err.Clear` объекта `Err` сбрасывает статус ошибки после того, как ошибка происходит. В OOo Basic, это достигается командами `On Error` или `Resume`.

Подсказки для структурированного обработчика ошибок

И команда определения обработчика, `On Error`, и команда возвращения, `Resume`, являются вариантами конструкции `Goto`.

Если Вы хотите аккуратно структурировать ваш код для предотвращения появления ошибок, когда Вы используете эти конструкции, Вы не должны использовать команды перехода, не контролируя их.

Будте осторожны, когда Вы используете команду `On Error Resume Next`, поскольку она сбрасывает все открытые сообщения об ошибках.

Лучшее решение состоит в том, чтобы использовать только один обработчик ошибок в пределах программы – держите обработчик ошибок отдельно от реального кода программы и не возвращайтесь назад к оригинальному коду после того, как происходит ошибка.

Далее – пример процедуры обработки ошибок:

```
Sub Example
   ' Определение обработчика ошибок в начале функции
  On Error Goto ErrorHandler
   ' ... фактический код программы
   ' Выключение обработчика ошибок
  On Error Goto 0
   ' Завершение нормального выполнения программы
  Exit Sub

' Начало обработчика ошибок
ErrorHandler:
   ' Проверка ожидаемых ошибок
  If Err = ExpectedErrorNo Then
```

```
    ' ... Технологическая ошибка
  Else
    ' ... Предупреждение о непредвиденной ошибке
  End If
  On Error Goto 0 ' Выключение обработчика ошибок
End Sub
```

Эта процедура начинается с определения обработчика ошибок, сопровождающего фактический код программы. В конце кода программы, обработка ошибок отключается вызовом `On Error Goto 0` и выполнение процедуры заканчивается командой `Exit Sub` (не перепутайте с `End Sub`).

Пример сначала проверяет, соответствует ли код ошибки ожидаемому числу (которое сохранено в воображаемой константе `ExpectedErrorNo`) и затем обрабатывает ошибку соответственно. Если происходит другая ошибка, система выводит предупреждение. Важно проверить код ошибки так, чтобы могли быть обнаружены непредвиденные ошибки.

Вызов `On Error Goto 0` в конце кода сбрасывает информацию о состоянии ошибки (код ошибки в системной переменной `Err`) так, чтобы ошибка, произошедшая позднее могла быть ясно распознана.

Глава 3. Библиотека времени выполнения OOo Basic

Следующие разделы представляют центральные функции библиотеки времени выполнения.

Функции преобразования

Во многих ситуациях, возникают обстоятельства, в которых переменная одного типа должна быть преобразована в переменную другого типа.

Неявное и явное преобразования типов

Самый легкий способ изменять переменную с одного типа на другой состоит в том, чтобы использовать присваивание.

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

В этом примере, переменная `A` – строка, а переменная `B` – целое число. OOo Basic гарантирует, что переменная `B` преобразуется в строку во время присваивания переменной `A`. Это преобразование намного более сложно, чем это кажется: целое число `B` остается в рабочей памяти в форме числа длиной два байта. `A`, с другой стороны, строка, и компьютер экономит один - или два байта длины значения для каждого символа (каждого числа). Поэтому, перед копированием содержимого из `B` в `A`, `B` должно быть преобразовано во внутренний формат `A`.

В отличие от большинства других языков программирования, Basic выполняет преобразование типа автоматически. Однако, это может иметь фатальные последствия. После более близкого рассмотрения, следующей последовательности кода

```
Dim A As String
Dim B As Integer
Dim C As Integer

B = 1
C = 1
A = B + C
```

который на первый взгляд кажется прямым, в конечном счете оказывается ловушкой. Интерпретатор Basic сначала вычисляет результат процесса сложения, а затем преобразовывает его в строку, которая, в результате, порождает строку 2.

Если, с другой стороны, Интерпретатор Basic сначала преобразует начальные значения `B` и `C`

в строки и применяет оператор плюс к результату, что порождает строку 11.

То же самое относится к использованию различных переменных:

```
Dim A
Dim B
Dim C

B = 1
C = "1"
A = B + C
```

Так как переменные типа Variant могут содержать и числа и строки, неясно, присваивается ли переменной `A` число 2 или строка 11.

Источник ошибок, вызванных неявным преобразованием типов можно избежать только осторожным программированием; например, не используя тип данных Variant.

Для предотвращения других ошибок, возникающих при неявном преобразовании типов, OOo Basic предлагает набор функций преобразования, которые Вы можете использовать для определения, когда тип данных операции должен быть преобразован:

- **CStr(Var)** – преобразует любой тип данных в строку. Логическое значение преобразуется в “True” или “False”. Дата преобразуется в строку с датой и временем. Значение `Null` вызывает ошибку времени выполнения. Значение `Empty` преобразуется в пустую строку. Числа преобразуются в соответствующее строковое значение, нулевые младшие разряды справа от десятичного разделителя отбрасываются;
- **CInt(Var)** – преобразует любые типы данных в целочисленное значение. Строки должны быть форматированы в соответствии с региональными настройками;
- **CLng (Var)** – преобразует любые типы данных в длинное целое число. Строки должны быть форматированы в соответствии с региональными настройками;
- **CSng(Var)** – преобразует любые типы данных в значение типа Single. Строки должны быть форматированы в соответствии с региональными настройками;
- **CDbl(Var)** – преобразует любые типы данных в значение типа Double. Строки должны быть форматированы в соответствии с региональными настройками. В США, “12.34” будет работать, но это может вызывать ошибку в другом месте;
- **CBool(Var)** - преобразует любые типы данных в логическое значение. Если Var – число, 0 отображается как `False`, любое другое значение как `True`. Если Var определяется как строка, то “true” и “false” (независимо от регистра) отображаются как `True` и `False`. Строки с любым другим значением вызывают ошибку времени выполнения;
- **CDate(Var)** - преобразует любые типы данных в значение даты. Числовые выражения содержат дату, начиная с 31 декабря 1899, слева от десятичного разделителя и время справа от десятичного разделителя. Строковые выражения должны быть форматированы в соответствии с соглашениями для функций `DateValue` и `TimeValue`. Другими словами, форматирование строки – зависит от региональных настроек.

Вы можете использовать эти функции преобразования, чтобы определить, как OOo Basic должен выполнять эти операции преобразования типа:

```
Dim A
Dim B
Dim C

B = 1
C = "1"
A = CStr(B + C)        ' B и C сначала складываются вместе, затем
```

```
                          ' преобразуются (порождая число 2)
A = CStr(B) + CStr(C)     ' B и C преобразуются в строки, затем
                          ' объединяются (порождая строку "11")
```

В первом дополнении в примере, OOo Basic сначала суммирует целочисленные переменные и затем преобразует результат в цепь символов. `A` присваивается строка `2`. Во втором случае, целочисленные переменные сначала преобразуются в две строки и затем связываются друг с другом посредством присваивания. Поэтому `A` присваивается строка `11`.

Числовые функции преобразования `CSng` и `CDbl` также принимают десятичные числа. Символ, определенный в соответствующих определенных для страны параметрах настройки должен использоваться как символ десятичной точки. Наоборот, CStr использует выбранные в настоящее время определенные параметры настройки для страны, при форматировании чисел, частей даты и времени.

Функция `Val` отличается от методов `Csng`, `Cdbl` и `Cstr`. Она преобразует строку в число; однако она всегда ожидает, что точка используется как символ десятичной точки.

```
Dim A As String
Dim B As Double

A = "2.22"
B = Val(A) ' Преобразуется правильно независимо от определенных
           ' для страны параметров настройки
```

Проверка содержимого переменных

В некоторых случаях, данные не могут быть преобразованы:

```
Dim A As String
Dim B As Date

A = "test"
B = A ' вызывает сообщение об ошибке
```

В приведенном примере, не имеет смысла присваивание строки `test` к переменной типа Date, таким образом интерпретатор Basic сообщает об ошибке. То же самое происходит, при попытке присвоить строку логической переменной:

```
Dim A As String
Dim B As Boolean

A = "test"
B = A ' вызывает сообщение об ошибке
```

Снова, интерпретатор Basic сообщает об ошибке.

Этих ошибочных сообщений можно избежать, проверяя программу перед присваиванием, чтобы установить является ли содержимое переменной, которая будет присвоена подходящим по типу целевой переменной. OOo Basic предоставляет с этой целью следующие функции проверки:

- **IsNumeric(Value)** - проверяет, является ли значение числом;
- **IsDate(Value)** - проверяет, является ли значение датой;
- **IsArray(Value)** - проверяет, является ли значение массивом.

Эти функции особенно полезны, при проверке правильности пользовательского ввода. Например, Вы можете проверить, ввел ли пользователь действительное число или дату.

```
If IsNumeric(UserInput) Then
  ValidInput = UserInput
Else
  ValidInput = 0
```

```
  MsgBox "Error message."
End If
```

В предыдущем примере, если переменная `UserInput` содержит действительное числовое значение, то оно присваивается переменной `ValidInput`. Если `UserInput` не содержит действительное число, `ValidInput` присваивается значение `0` и возвращается сообщение об ошибке.

В то время как в OOo Basic существуют тестовые функции для проверки чисел, элементов даты и массивов, соответствующей функции для проверки логических значений не существует. Функциональным возможностям, однако, можно подражать при использовании функции IsBoolean:

```
Function IsBoolean(Value As Variant) As Boolean
  On Error Goto ErrorIsBoolean
  Dim Dummy As Boolean


  Dummy = Value

  IsBoolean = True
  On Error Goto 0
  Exit Sub

ErrorIsBoolean:
  IsBoolean = False
  On Error Goto 0
End Function
```

Функция `IsBoolean` определяет внутреннюю вспомогательную переменную `Dummy` типа `Boolean` и пробует присвоить ей переданное значение. Если присваивание успешно, функция возвращает `True`. Если она терпит неудачу, порождается ошибка времени выполнения, которая перехватывается тестовой функцией, чтобы возвратить ошибку.

Примечание Если строка в OOo Basic содержит нечисловое значение и если оно присваивается числовой переменной, OOo Basic не порождает сообщение об ошибке, но передает переменной значение 0. Эта процедура отличается от VBA. Там вызывается ошибка и выполнение программы заканчивается, если выполняется соответствующее присваивание.

Строки

Работа с наборами символов

Управляя строками, OOo Basic использует набор символов Unicode. Функции `Asc` и `Chr` позволяют установить значение Unicode, принадлежащее символу и/или соответствующий символ находящийся по значению Unicode.

```
Code = Asc("A")   ' Латинская буква A (Unicode-значение 65)
Code = Asc("€")   ' Символ Euro (Unicode- значение 8364)
Code = Asc("л")   ' Кириллическая буква л (Unicode- значение 1083)
```

Наоборот, выражение

```
MyString = Chr(13)
```

гарантирует, что строка `MyString` инициализируется со значением числа 13, которое соответствует жесткому разрыву строки.

Функция `Chr` часто используется в языках Basic, чтобы вставить управляющие символы в строку. Присваивание

```
MyString = Chr(9) + "This is a test" + Chr(13)
```

гарантирует поэтому, что тексту предшествует символ табуляции (Unicode-значение 9), и что жесткий разрыв строки (Unicode-значение 13) добавляется после текста.

Функция `LCase` преобразует все буквы в строке к строчным буквам. Только заглавные буквы в пределах строки преобразуются. Все строчные буквы и небуквенные символы остаются неизменными.

Функция `UCase` преобразует символы в строке к прописным буквам. Только строчные буквы в строке затрагиваются. Заглавные буквы и все другие символы остаются неизменными.

```
Dim sVar As String

sVar = "Санкт-Петербург"

Print LCase(sVar) ' выводит "санкт-петербург"
Print UCase(sVar) ' выводит "САНКТ-ПЕТЕРБУРГ"
```

Для удаления пробелов в начале и конце строки используются следующие функции:

- **LTrim(MyString)** – возвращает строку удаляя из переданной ей строки `MyString` начальные пробелы;
- **RTrim(MyString)** – возвращает строку удаляя из переданной ей строки `MyString` пробелы в конце строки;
- **Trim(MyString)** – возвращает строку удаляя из переданной ей строки `MyString` пробелы в начале и конце строки.

Доступ к частям строк

OOo Basic предоставляет четыре функции, которые возвращают части строки:

- **Left(MyString, Length)** – возвращает первые `Length` символов `MyString`;
- **Right(MyString, Length)** – возвращает последние `Length` символов `MyString`;
- **Mid(MyString, Start, Length)** – возвращает первые `Length` символов `MyString` начиная с позиции `Start`;
- **Len(MyString)** – возвращает число символов в `MyString`.

Вот - несколько примеров вызова названных функций:

```
Dim MyString As String
Dim MyResult As String
Dim MyLen As Integer

MyString = "This is a small test"

MyResult = Left(MyString,5)     ' выдает строку "This "
MyResult = Right(MyString, 5)   ' выдает строку " test"
MyResult = Mid(MyString, 8, 5)  ' выдает строку " a sm"
MyLen    = Len(MyString)        ' выдает значение 21
```

Поиск и замена

OOo Basic предоставляет функцию `InStr` для поиска строки в пределах другой строки:

```
ResultString = InStr(SearchString, MyString)
```

Параметр `SearchString` определяет строку, которая разыскивается в пределах `MyString`. Функция возвращает число, которое содержит положение, в котором начало `SearchString` появляется в пределах `MyString`. Если Вы хотите найти другие вхождения строки, функция также обеспечивает возможность определить дополнительное положение начала, с которого OOo Basic начинает поиск. В этом случае, синтаксис функции:

```
ResultString = InStr(StartPosition, MyString, SearchString)
```

В предыдущих примерах, InStr игнорирует различие между заглавными и строчными символами. Чтобы изменить поиск так, чтобы функция `InStr` была чувствительна к регистру символов, добавьте параметр 0, как показано в следующем примере:

```
ResultString = InStr(MyString, SearchString, 0)
```

Используя предыдущие функции для редактирования строк, программисты могут написать функцию для поиска и замены одной строки в другой строке:

```
Function Replace(Source As String, Search As String, NewPart As String)
  Dim Result As String
  Dim StartPos As Long
  Dim CurrentPos As Long


  Result = ""
  StartPos = 1
  CurrentPos = 1

  If Search = "" Then
    Result = Source
  Else
    Do While CurrentPos <> 0
      CurrentPos = InStr(StartPos, Search, Source)
      If CurrentPos <> 0 Then
        Result = Result + Mid(Source, StartPos, _
        CurrentPos - StartPos)
        Result = Result + NewPart
        StartPos = CurrentPos + Len(Search)
      Else
        Result = Result + Mid(Source, StartPos, Len(Source))
      End If ' Position <> 0
    Loop
  End If
  Replace = Result
End Function
```

Функция ищет переданную строку `Search` в цикде посредством `InStr` в оригинальной строке `Source`. Если она находит элемент поиска, она берет часть перед ним и дописывает его к буферу возврата `Result`. Она добавляет секцию NewPart в точке поиска. Если соответствий больше не найдено для элемента поиска, функция устанавливает все еще остающейся часть строки и добавляет ее к буферу возврата. Она возвращает строку, произведенную таким образом как результат процесса замены.

Так как замены части последовательности символов – одна из наиболее часто используемых функций, функция `Mid` OOo Basic была расширена так, чтобы эта задача выполнялась автоматически. Следующий пример

```
Dim MyString As String

MyString = "This was my text"
Mid(MyString, 6, 3, "is")
```

заменяет три символа строки с шестой позиции строки `MyString`.

Форматирование строк

Функция `Format` преобразует число в строку, форматированную согласно дополнительной строке формата. Несколько форматов могут быть включены в единственную строку формата. Каждый отдельный формат отделяется “;”. Первый формат используется для положительных чисел, второй - для отрицательных чисел, и третий - для нуля. Если присутствует только один код формата, он обращается ко всем числам.

Нулевой символ в пределах шаблона гарантирует, что число всегда помещается в соответствующей позиции. Если число отсутствует, 0 отображается на его месте.

Точка поддерживает символ десятичного разделителя, определенный операционной системой в определенных для страны региональных параметрах настройки.

Пример ниже показывает, как символы `нуль` и *точка* могут определить цифры после десятичного разделителя в выражении:

```
MyFormat = "0.00"

MyString = Format(-1579.8, MyFormat) ' Получается "-1579,80"
MyString = Format(1579.8, MyFormat)  ' Получается "1579,80"
MyString = Format(0.4, MyFormat)     ' Получается "0,40"
MyString = Format(0.434, MyFormat)   ' Получается "0,43"
```

Таким же образом, нули могут быть добавлены перед числом, чтобы добиться требуемой длины:

```
MyFormat = "0000.00"

MyString = Format(-1579.8, MyFormat) ' Получается "-1579,80"
MyString = Format(1579.8, MyFormat)  ' Получается "1579,80"
MyString = Format(0.4, MyFormat)     ' Получается "0000,40"
MyString = Format(0.434, MyFormat)   ' Получается "0000,43"
```

Запятая представляет символ, который операционная система использует для разделителя тысяч, а признак фунта поддерживает цифру или место, которое показывается только, если этого требует входная строка.

```
MyFormat = "#,##0.00"

MyString = Format(-1579.8, MyFormat) ' Получается "-1.579,80"
MyString = Format(1579.8, MyFormat)  ' Получается "1.579,80"
MyString = Format(0.4, MyFormat)     ' Получается "0,40"
MyString = Format(0.434, MyFormat)   ' Получается "0,43"
```

Вместо знака доллара, функция `Format` показывает соответствующий символ валюты, определенный системой:

```
MyFormat = "#,##0.00 $"

MyString = Format(-1579.8, MyFormat) ' Получается "-1.579,80 €"
MyString = Format(1579.8, MyFormat)  ' Получается "1.579,80 €"
MyString = Format(0.4, MyFormat)     ' Получается "0,40 €"
MyString = Format(0.434, MyFormat)   ' Получается "0,43 €"
```

Код	Описание
0	Если Число имеет цифру в положении 0 в коде формата, цифра показывается; иначе появляется ноль. Это означает, что ведущие и замыкающие нули отображаются, ведущие цифры не обрезаются, и замыкающие десятичные разряды округляются.
#	Это работает как этот 0, но ведущие и замыкающие нули не отображаются.

Код	Описание
.	Позиция десятичной точки определяет число десятичных знаков слева и справа от десятичного разделителя.
%	Число умножается на 100 и вставляется знак процента (%), где он появляется в коде формата.
E- E+ e- e+	Если код формата содержит по крайней мере одну цифровой символ-заполнитель (0 или #) справа от символа, число форматируется в научной нотации. Буква E или e вставляется между основанием и показателем. Число символов-заполнителей для цифр справа от символа определяет число цифр в показателе. Если показатель отрицателен, знак минус отображается непосредственно перед показателем. Если показатель положителен, знак плюс отображается только перед показателем с E+ или e+.
,	Запятая – символ-заполнитель для разделителя тысяч. Она отделяет тысячи от сотен в числе по крайней мере с четырьмя цифрами. Разделитель тысяч показан, если коде формата содержит символ-заполнитель, окруженный цифровыми символами-заполнителями (0 или #).
- + $ ( ) пробел	Плюс (+), минус (-), доллар ($), пробел, или скобки, введенные непосредственно в код формата отображается как непосредственный символ.
\	Обратный слэш показывает следующий символ в коде формата. Другими словами, это препятствует следующему символу быть воспринятым как специальный символ. Обратный слэш не отображается, если Вы не вводите двойной обратный слэш (\\) в код формата. Символы, которым должен предшествовать обратный слэш в коде формата, чтобы быть показанным как буквальные символы, символы форматирования даты и времени (a, c, d, h, м., n, p, q, s, t, w, y,/ :), символы форматирования чисел (#, 0, %, E, e, *запятая*, *точка*) и символы форматирования строки (@, &, <, >, !). Вы можете также заключить символы в двойные кавычки.
General Number	Числа отображается как оно введено.
Currency	Знак долларовый помещается перед числом; отрицательные числа заключены в круглых скобках. Отображаются два десятичных знака. (Фактически, определяется региональными настройками),
Fixed	По крайней мере одна цифра отображается перед десятичным разделителем. Отображаются два десятичных знака.
Standard	Числа отображается с разделителем тысяч определенным в соответствии с региональными настройками. Отображаются два десятичных знака.
Scientific	Число отображается в научной нотации. Отображаются два десятичных знака.

```
Sub ExampleFormat
  MsgBox Format(6328.2, "##,##0.00")          ' = 6,328.20
  MsgBox Format(123456789.5555, "##,##0.00")  ' = 123,456,789.56
  MsgBox Format(0.555, ".##")                 ' .56
  MsgBox Format(123.555, "#.##")              ' 123.56
```

```
  MsgBox Format(0.555, "0.##")                ' 0.56
  MsgBox Format(0.1255555, "%#.##")           ' %12.56
  MsgBox Format(123.45678, "##E-####")        ' 12E1
  MsgBox Format(.0012345678, "0.0E-####")     ' 1.2E3 (нарушенный)
  MsgBox Format(123.45678, "#.e-###")         ' 1e2
  MsgBox Format(.0012345678, "#.e-###")       ' 1e3 (нарушенный)
  MsgBox Format(123.456789, "#.## is ###")    ' 123.45 is 679 (странный)
  MsgBox Format(8123.456789, "General Number") ' 8123.456789
  MsgBox Format(8123.456789, "Fixed")         ' 8123.46
  MsgBox Format(8123.456789, "Currency")      ' 8,123.46$ (нарушенный)
  MsgBox Format(8123.456789, "Standard")      ' 8,123.46
  MsgBox Format(8123.456789, "Scientific")    ' 8.12E03
  MsgBox Format(0.00123456789, "Scientific") ' 1.23E03 (нарушенный)
End Sub
```

Примечание Инструкции форматирования, используемые в VBA для форматирования элементов даты и времени не поддерживаются в OOo Basic.

Никакого преобразования не происходит, если параметр не число (то есть, если это – строка), и возвращается пустая строка.

Дата и Время

OOo Basic обеспечивает тип данных `Date`, который сохраняет подробности даты и времени в двоичном формате.

Описание подробностей Даты и Времени в пределах кода программы

Вы можете присвоить дату переменной типа `Date` через присваивание простой строки:

```
Dim MyDate As Date

MyDate = "1.1.2002"
```

Это присваивание может функционировать должным образом, потому что OOo Basic автоматически преобразует значение даты, определенное как строка в переменную типа `Date`. Этот тип присваивания, однако, может вызвать ошибки, значения даты и времени определяются и отображаются по-разному в различных странах.

С тех пор как OOo Basic использует определенные для страны региональные настройки операционной системы, преобразовывая строку в значение даты, выражение, показанное ранее функционируют правильно только, если определенные для страны параметры настройки соответствуют строковому выражению.

Чтобы избежать этой проблемы, должна использоваться функция `DateSerial` для присваивания заданного значения переменной типа `Date`:

```
Dim MyDate As Date

MyDate = DateSerial(2001, 1, 1)
```

Параметры функции должны быть в последовательности: год, месяц, день. Функция гарантирует, что переменной на самом деле присваивается правильное значение независимо от определенных для страны параметров настройки.

Функция `TimeSerial` форматирует подробности времени таким же образом, что функция `DateSerial` форматирует дату:

```
Dim MyDate As Date

MyDate = TimeSerial(11, 23, 45)
```

Ее параметры должны быть определены в последовательности: часы, минуты, секунды.

Извлечение подробностей Даты и Времени

Следующие функции образуют дополнение функциям `DateSerial` и `TimeSerial`:

- **Day(MyDate)** – возвращает день месяца из `MyDate`;
- **Month(MyDate)** – возвращает месяц из `MyDate`;
- **Year(MyDate)** – возвращает год из `MyDate`;
- **Weekday(MyDate)** – возвращает номер дня недели из `MyDate`;
- **Hour(MyTime)** – возвращает часы из `MyTime`;
- **Minute(MyTime)** - возвращает минуты из `MyTime`;
- **Second(MyTime)** - возвращает секунды из `MyTime`;

Эти функции извлекают секции даты или времени из указанной переменной `Date`. Пример

```
Dim MyDate As Date

' ... Initialization of MyDate
If Year(MyDate) = 2003 Then

  ' ... Specified date is in the year 2003

End If
```

проверяет, находится ли дата, сохранная в `MyDate` в 2003 году. Таким же образом, пример

```
Dim MyTime As Date
' ... Initialization of MyTime

If Hour(MyTime) >= 12 And Hour(MyTime) < 14 Then

  ' ... Specified time is between 12 and 14 hours

End If
```

проверяет, попадает ли `MyTime` между 12 и 14 часами.

Функция `Weekday` возвращает номер дня недели для переданной даты:

```
Dim MyDate As Date
Dim MyWeekday As String
' ... инициализация MyDate

Select Case WeekDay(MyDate)
  case 1
    MyWeekday = "Воскресенье"
  case 2
    MyWeekday = "Понедельник"
  case 3
    MyWeekday = "Вторник"
  case 4
    MyWeekday = "Среда"
  case 5
    MyWeekday = "Четверг"
  case 6
    MyWeekday = "Пятница"
  case 7
    MyWeekday = "Суббота"
End Select
```

Примечание Воскресенье считается первым днем недели.

Извлечение системной даты и времени

Следующие функции доступны в OOo Basic для извлечения системного времени и системной даты:

- **Date** - возвращает или изменяет текущую системную дату;
- **Time** - возвращает текущее время;
- **Now** - возвращает текущий момент времени (дата и время как объединенное значение).

Преобразование даты и времени

В контексте набора инструментов UNO элементов управления есть две других функции. Метод управления полем даты `com.sun.star.awt.XDateField:setDate()` ожидает, что дата будет передана как длинное целое значение в специальном ISO формате, а метод `com.sun.star.awt.XDateField:getDate()` возвращает дату в этом формате.

Функция времени выполнения Basic `CDateToIso` преобразует дату из внутреннего формата даты Basic в формата даты требуемый ISO. Так как строковый формат даты, возвращаемый функцией `Date` преобразуется к внутреннему формату даты Basic автоматически, Дата может использоваться непосредственно как входной параметр для `CDateToIso`:

```
IsoDate = CDateToIso(Date)
TextField.setDate(IsoDate)
```

Функция времени выполнения `CDateToIso` представляет обратную операцию и преобразует дату из формата даты ISO во внутренний формата даты Basic.

```
Dim aDate as Date
aDate = CDateFromIso(IsoDate)
```

Файлы и каталоги

Работа с файлами - одна из основных задач приложения. OOo API предоставляет Вам целый набор объектов, с помощью которых Вы можете создавать, открывать и изменять офисные документы. Они представлены подробно в Главе 4. Независимо от этого, в некоторых случаях Вы должны будете получить непосредственный доступ к файловой системе, осуществлять поиск в каталогах или редактировать текстовые файлы. Библиотека времени выполнения OOo Basic обеспечивает несколько основных функций для этих задач.

Примечание Некоторые DOS-ориентированные функции работы с файлами и каталогами больше не поддерживаются в OpenOffice.org 2, или их функция только ограничена. Например, поддержка функций `ChDir`, `ChDrive` и `CurDir` не обеспечивается. Некоторые DOS-ориентированные свойства больше не используются в функциях, которые ожидают свойства файла как параметры (например, для определения скрытых и системных файлов). Эти изменения были вызваны необходимостью гарантировать максимально возможный уровень независимости от платформы для OpenOffice.org.

Управление Файлами

Поиск по каталогам

Функция `Dir` в OOo Basic отвечает за поиск по каталогам файлов и подкаталогов. Предварительно запрошенная строка, содержащая путь каталогов, в которых будет осуществляться поиск, должна передаваться `Dir` в качестве первого параметра. URL нотация воспринемается. Второй параметр `Dir` определяет файл или каталог, который ищется. OOo Basic возвращает название первого найденного элемента каталогов. Чтобы извлечь следующий элемент, функцию `Dir` нужно повторно вызвать без параметров. Если функция `Dir` не находит больше записей, она возвращает пустую строку.

Следующий пример показывает, как функция `Dir` может использоваться для получения всех файлов, расположенных в одном каталоге.

```
Sub ShowFiles
  Dim NextFile As String
  Dim AllFiles As String

  AllFiles = ""
  NextFile = Dir("C:\", 0)

  While NextFile <> ""
    AllFiles = AllFiles & Chr(13) & NextFile
    NextFile = Dir
  Wend

  MsgBox AllFiles
End Sub
```

Нуль, используемый как второй параметр в функции `Dir` гарантирует, что `Dir` возвращает только имена файлов, а каталоги игнорируются. Здесь могут быть определены следующие параметры:

- **0**: возвращает обычные файлы;
- **16**: подкаталоги.

Следующий пример - фактически то же самое что и предыдущий пример, но функции `Dir` передается значение 16 как параметр, который требует возвращать подкаталоги папки, а не имена файлов.

```
Sub ShowDirs
  Dim NextDir As String
  Dim AllDirs As String

  AllDirs = ""
  NextDir = Dir("C:\", 16)

  While NextDir <> ""
    AllDirs = AllDirs & Chr(13) & NextDir
    NextDir = Dir
  Wend

  MsgBox AllDirs
End Sub
```

Примечание Когда требуется в OOo Basic, функция `Dir`, с использованием параметра 16, возвращает только подкаталоги папки. В VBA, функция также возвращает названия обычных файлов так, что требуется дальнейшая проверка, чтобы извлечь только каталоги. Используя функцию `CompatibilityMode(true)`, OOo Basic ведет себя как и функция VBA `Dir`, с использованием параметра 16, возвращая подкаталоги и обычные файлы.

Примечание Варианты, обеспечиваемые в VBA для того, чтобы искать в каталогах коннкретно для файлов со свойствами *скрытый*, *системный*, *архивный*, и *метка тома* не существуют в OOo Basic, потому что соответствующие функции файловой системы не доступны на всех операционных системах.

Примечание Спецификации пути, перечисленные в `Dir` могут использовать символы-заполнители * и ? и в VBA и в OOo Basic. В OOo Basic, символ-заполнитель * может однако быть только последним символом имени файла и/или расширения файла, что не имеет места в VBA.

Внимание Некоторые операционные системы включают каталоги "." и "..", которые обращаются к текущему и родительскому каталогам соответственно. Если Вы пишете код, который переходит по каталогам, Вы вероятно не хотите идти по ним, или Вы застрянете в бесконечном цикле.

Создание и удаление каталогов

OOo Basic предоставляет функцию `MkDir` для создания каталогов.

```
MkDir("C:\SubDir1")
```

Эта функция создает каталоги и подкаталоги. Все каталоги, необходимые в пределах иерархии также создаются, если требуется. Например, если существует только каталог `C:\SubDir1`, то запрос

```
MkDir ("C:\SubDir1\SubDir2\SubDir3\")
```

создает и каталог `C:\SubDir1\SubDir2` и каталог `C:\SubDir1\SubDir2\SubDir3`.

Функция `RmDir` удаляет каталоги.

```
RmDir ("C:\SubDir1\SubDir2\SubDir3\")
```

Если каталог содержит подкаталоги или файлы, они *также удаляются*. Вы, поэтому, должны быть осторожными, используя `RmDir`.

Примечание В VBA, функции `MkDir` и `RmDir` затрагивают только текущий каталог. С другой стороны в OOo Basic, `MkDir` и `RmDir` могут использоваться для создания или удаления иерархии каталогов.

Примечание В VBA, `RmDir` генерирует сообщение об ошибке, если каталог содержит файлы. В OOo Basic, удаляются каталог и все содержащиеся в нем файлы. Если Вы будете использовать функцию `CompatibilityMode(true)`, то OOo Basic будет вести себя как VBA.

Копирование, переименование, удаление и проверка существования файлов

Вызов

```
FileCopy(Source, Destination)
```

создает копию файла `Source` под именем `Destination`.

С помощью функции

```
Name OldName, NewName
```

Вы можете переименовать файл `OldName` в `NewName`.

Вызов

```
Kill(Filename)
```

удаляет файл `Filename`. Поддерживается любая файловая нотация, но подстановочные символы * и ? не поддерживаются. Если Вы хотите удалить каталог (включая его файлы), используют функцию `RmDir`.

Функция `FileExists` может использоваться для проверки, существует ли файл или каталог:

```
If FileExists(Filename) Then
  MsgBox "file exists."
End If
```

Чтение и изменение свойств файла

Работая с файлами, иногда важно иметь возможность установить свойства файла, время, когда файл последний раз изменялся и длину файла.

Вызов

```
Dim Attr As Integer
Attr = GetAttr(Filename)
```

возвращает некоторые свойства файла. Возвращаемое значение содержит битовую маску, в которой возможны следующие значения:

- **1**: файл только для чтения;
- **16**: имя каталога.

Пример

```
Dim FileMask As Integer
Dim FileDescription As String

FileMask = GetAttr("test.txt")
If (FileMask AND 1) > 0 Then
  FileDescription = FileDescription & " только для чтения "
End IF

If (FileMask AND 16) > 0 Then
  FileDescription = FileDescription & " каталог "
End IF

If FileDescription = "" Then
  FileDescription = " нормальный "
End IF

MsgBox FileDescription
```

определяет битовую маску файла `test.txt` и проверяет, только ли он для чтения, является ли он каталогом. Если ни один из них не используется, `FileDescription` присваивается строка “нормальный”.

Примечание Флаги, используемые в VBA для запроса файловых свойств *скрытый*, *системный*, *архивный* и *метка тома* не поддерживаются в OOo Basic, потому что они являются специфичными для Windows и не поддерживаются или поддерживаются только частично в других операционных системах.

Функция `SetAttr` изменяет свойства файла. Вызов

```
SetAttr("test.txt", 1)
```

может использоваться для задания файлу свойства только для чтения. Существующее свойство только для чтения может быть удалено следующим вызовом:

```
SetAttr("test.txt", 0)
```

Дату и время создания или последнего изменения файла можно получить функцией `FileDateTime`. Дата форматируется здесь в соответствии с региональными параметрами настройки, используемыми в системе. В имени файла подстановочные символы * и ? не поддерживаются.

```
FileDateTime("test.txt") ' Возвращает дату и время последнего
                         ' изменения файла
```

Функция `FileLen` определяет длину файла в байтах (как значение длинного целого числа).

```
FileLen("test.txt")      ' Возвращает длину файла в байтах
```

Для определения текущей длины открытого файла, используйте вместо этого функцию `Lof`.

Запись и чтение текстовых файлов

OOo Basic предоставляет широкий набор методов для чтения и записи файлов. Следующие объяснения касаются работы с текстовыми файлами (а не текстовыми документами).

Запись текстовых файлов

Прежде, чем к текстовому файлу получить доступ, он должен быть сначала открыт. Чтобы сделать это, необходим свободный файловый манипулятор, который явно идентифицирует файл для последующего доступа к нему.

Функция `FreeFile` используется для создания свободного файлового манипулятора. Манипулятор используется как параметр для инструкции `Open`, которая открывает файл. Для открытия файла, определенного как текстовый файл, вызывается Open:

```
Open Filename For Output As #FileNo
```

`Filename` – строка, содержащая имя файла. `FileNo` – манипулятор, созданный функцией `FreeFile`.

Как только файл открыт, инструкция Print может записывать строку за строкой:

```
Print #FileNo, "Это тестовая строка."
```

`FileNo` также означает здесь файловый манипулятор. Второй параметр определяет текст, который должен быть сохранен как строка текстового файла.

Как только запись будет закончена, файл должен быть закрыт, с использованием вызова `Close`:

```
Close #FileNo
```

Здесь снова, должен быть определен файловый манипулятор.

Следующий пример показывает, как текстовый файл открывается, в него осуществляется запись и он закрывается:

```
Dim FileNo As Integer
Dim CurrentLine As String
Dim Filename As String

Filename = "c:\data.txt" ' Определение имени файла
```

```
FileNo = Freefile ' Установление свободного файлового манипулятора
Open Filename For Output As #FileNo ' Открытие файла (в режиме на запись)
Print #FileNo, "Это строка текста"  ' сохранение строки
Print #FileNo, "Это другая строка текста" ' сохранение строки
Close #FileNo ' Закрытие файла
```

Чтение текстовых файлов

Текстовые файлы читаются таким же образом, каким они записываются. Инструкция `Open` используемая для открытия файл содержит выражение `For Input` вместо выражения `For Output`, а вместо команды `Print` для записи данных, должна использоваться инструкция `Line Input` для чтения данных.

Наконец, вызов для текстового файла инструкции

```
eof(FileNo)
```

используется, чтобы проверить, был ли достигнут конец файла.

Следующий пример показывает, как может быть прочитан текстовый файл:

```
Dim FileNo As Integer
Dim CurrentLine As String
Dim File As String
Dim Msg as String

' Определение имени файла
Filename = "c:\data.txt"

' Установление свободного файлового манипулятора
FileNo = Freefile

' Открытие файла (в режиме на чтение)
Open Filename For Input As #FileNo

' Проверка, был ли достигнут конец файла
Do While not eof(FileNo)
  ' Чтение строки
  Line Input #FileNo, CurrentLine
  If CurrentLine <> "" then
    Msg = Msg & CurrentLine & Chr(13)
  end if
Loop

' Закрытие файла
Close #FileNo

Msgbox Msg
```

Отдельные строки извлекаются в цикле `Do While`, сохраняется в переменной `Msg` и показывается в конце в окне сообщения.

Окна сообщений и ввода

OOo Basic предоставляет функции `MsgBox` и `InputBox` для базовой связи с пользователем.

Вывод сообщений

Функция MsgBox показывает базовое информационное окно, которое может иметь одну или более кнопок. Ее самый простой вариант вызова

```
MsgBox "Это информация!"
```

`MsgBox` содержит только текст и кнопку Ok.

Появление информационного окна может быть изменено с использованием параметра. Параметр обеспечивает выбор дополнительных кнопок, определение кнопки по умолчанию, и добавления информационного символа. Значения для выбора кнопки:

- **0** – кнопка Ok;
- **1** – кнопки Ok и Отмена;
- **2** – кнопки Отмена, Повторить и Пропустить;
- **3** - кнопки Да, Нет, и Отмена;
- **4** - кнопки Да и Нет;
- **5** - кнопки Повторить и Отмена.

Чтобы установить кнопку в качестве кнопки по умолчанию, добавьте одно из следующих значений к значению параметра из списка выбора кнопок.

- **0** – первая кнопка;
- **256** – вторая кнопка;
- **512** – третья кнопка.

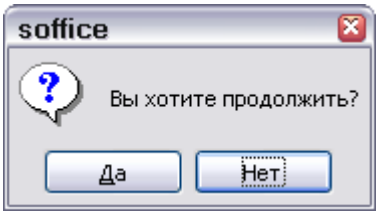
Наконец, следующие информационные символы доступны и могут также быть отображены, добавлением соответствующего значения к параметру:

- **16** - знак Стоп;
- **32** - Вопросительный знак;
- **48** - Восклицательный знак;
- **64** - знак Информация.

Вызов

```
MsgBox "Вы хотите продолжить?", 292
```

показывает информационное окно с кнопками Да и Нет (значение 4), из которых вторая кнопка (Нет) установлена как кнопка по умолчанию (значение 256), и которое также содержит вопросительный знак (значение 32), 4+256+32=292



Если информационное окно содержит несколько кнопок, то возвращаемое значение должно быть проверено для определения, какая кнопка была нажата. Следующие возвращаемые значения доступны в этом случае:

- **1** – кнопка Ok;
- **2** – кнопка Отмена;
- **4** – кнопка Повторить;
- **5** – кнопка Пропустить;
- **6** – кнопка Да;

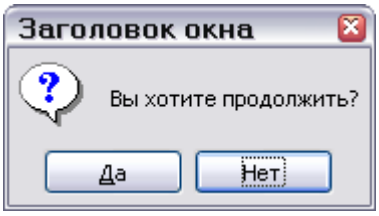
- **7** – кнопка Нет.

В предыдущем примере, проверка возвращаемого значения могла быть выполнена следующим образом:

```
If MsgBox ("Вы хотите продолжить?", 292) = 6 Then
   ' нажата кнопка Да
Else
   ' нажата кнопка Нет
End IF
```

В дополнение к информационному тексту и параметру для настройки информационного окна, `MsgBox` также разрешает третий параметр, который определяет текст в заголовке окна:

```
MsgBox "Вы хотите продолжить?", 292, "Заголовок окна"
```



Если никакое название окна не задано, по умолчанию – “soffice”.

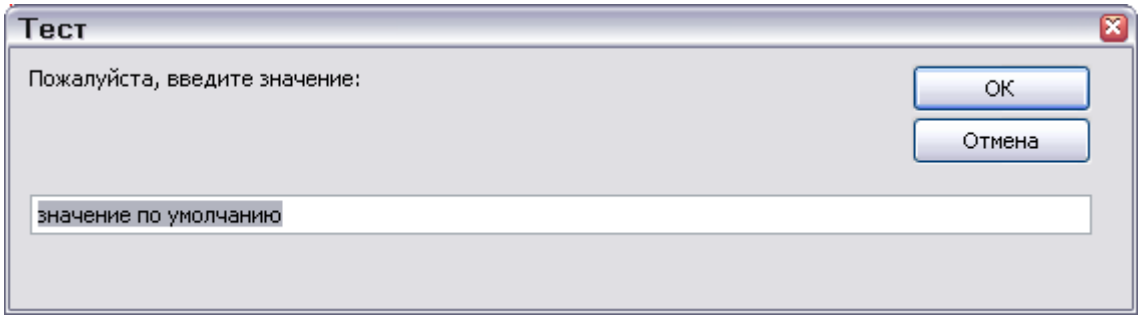
Окно ввода для запроса простых строк

Функция `InputBox` запрашивает простые строки от пользователя. Поэтому она простая альтернатива созданию диалогов. `InputBox` получает три стандартных параметра:

- информационный текст;
- заголовок окна;
- значение по умолчанию, которое может быть добавлено в область ввода.

```
InputVal = InputBox("Пожалуйста, введите значение:", "Тест", _
                    "значение по умолчанию")
```

В качестве возвращаемого значения, функция `InputBox` обеспечивает строку, введенную пользователем.



При нажатии на кнопку Отмена, возвращается пустая строка.

В качестве четвертого и пятого параметров могут указываться координаты положения окна ввода по горизонтали и вертикали. Если они не указываются, то диалог располагается в центре экрана.

```
InputVal = InputBox("Пожалуйста, введите значение:", "Тест", _
                    "значение по умолчанию", x, y)
```

Другие функции

Beep

Функция `Beep` заставляет систему проиграть звук, который может использоваться, для предупреждения пользователя относительно неправильного действия. `Beep` не имеет никаких параметров:

```
Beep ' создает информационный сигнал
```

Shell

Внешние программы могут быть запущены с использованием функции `Shell`.

```
Shell(Pathname, Windowstyle, Param)
```

`Pathname` определяет путь к программе, которая будет выполнена. `Windowstyle` определяет окно, в котором запускается программа. Возможны следующие значения:

- **0** - программа получает фокус и запускается в скрытом окне;
- **1** - программа получает фокус и запускается в окне нормального размера;
- **2** - программа получает фокус и запускается в минимизированном окне;
- **3** - программа получает фокус и запускается в развернутом окне;
- **4** - программа запускается в окне нормального размера, не получая фокуса;
- **6** - программа запускается в минимизированном окне, фокус остается в текущем окне;
- **10** - программа запускается в полноэкранном режиме.

Третий параметр, `Param`, разрешает параметрам командной строки быть переданными программе, которая будет запущена.

Wait

Функция Wait приостанавливает выполнение программы на указанный промежуток времени. Период ожидания определяется в миллисекундах. Команда

```
Wait 2000
```

определяет перерыв в 2 секунды (2000 миллисекунд).

Environ

Функция `Environ` возвращает переменные окружения операционной системы. В зависимости от системы и конфигурации, здесь сохраняются различные типы данных. Вызов

```
Dim TempDir

TempDir=Environ ("TEMP")
```

определяет переменную окружения временного каталога операционной системы.

Глава 4. Введение в OpenOffice.org API

OpenOffice.org API - универсальный программный интерфейс для доступа к OpenOffice.org. Вы можете использовать OpenOffice.org API для создания, открытия, изменения и распечатки документов OpenOffice.org. Он обеспечивает выбор расширенных функциональных возможностей OpenOffice.org через персональные макросы и позволяет писать персональные диалоги.

OpenOffice.org API может использоваться не только с OOo Basic, но также и с другими языками программирования, такими как Java и C++. Вызываемые методы *Универсальных сетевых объектов* (*Universal Network Objects*, UNO), которые обеспечивают интерфейс к различным языкам программирования, делают это возможным.

Эта глава сосредотачивается на том, как OpenOffice.org может использоваться в OOo Basic при помощи UNO. Она описывает основные понятия UNO с точки зрения программиста OOo Basic. Подробности относительно работы с различными частями OpenOffice.org API могут быть найдены в следующих главах.

Универсальные сетевые объекты (UNO)

OpenOffice.org предоставляет программный интерфейс в форме универсальных сетевых объектов (UNO). Это – объектно-ориентированный программный интерфейс, который OpenOffice.org подразделяет на различные объекты, которые для своих частей гарантируют программно-управляемый доступ к офисному пакету.

Так как OOo Basic – процедурный язык программирования, несколько лингвистических конструкций должны были быть добавлены к нему, чтобы позволить использование UNO.

Для использования универсальных сетевых объектов в OOo Basic, Вы будете нуждаться в объявлении переменной для дополнительных объектов. Объявление делается с использованием инструкции Dim (см. главу 2). Должно использоваться обозначение типа `Object` для объявления объектной переменной:

```
Dim Obj As Object
```

Этот вызов объявляет объектную переменную по имени `Obj`.

Созданная объектная переменная должна быть инициализирована так, чтобы она могла использоваться. Это может быть сделано с использованием функции `createUnoService`:

```
Obj = createUnoService("com.sun.star.frame.Desktop")
```

Этот вызов присваивает переменной `Obj` ссылку на недавно созданный объект. `com.sun.star.frame.Desktop` напоминает тип объекта; однако в терминологии UNO это называют "*сервисом*", а не типом. В соответствии с философией UNO, `Obj` описан как ссылка на объект, который поддерживается `com.sun.star.frame.Desktop`. Термин "*сервис*" используемый в OOo Basic, поэтому соответствует термину *тип* и *класс*, используемым в

других языках программирования.

Есть, однако, одно главное отличие: универсальный сетевой объект может поддерживать несколько сервисов одновременно. Некоторые сервисы UNO в свою очередь поддерживают другие сервисы так, чтобы, через один объект, Вам предоставлялся целый диапазон сервисов. Например, вышеупомянутый объект, который является основанным на `com.sun.star.frame.Desktop`, может также включать другие сервисы для загрузки документов и завершения программы.

Примечание Принимая во внимание, что структура объекта в VBA определяется классом, которому он принадлежит, в OOo Basic, структура определяется через сервисы, которые он поддерживает. Объект VBA всегда назначается точно на один единственный класс. Объект OOo Basic может, однако, поддерживать несколько сервисов.

Свойства и Методы

Объект в OOo Basic обеспечивает набор свойств и методов, к которым можно получить доступ посредством объекта.

Свойства

Свойства походят на свойства объекта; например, `Filename` и `Title` для объекта `Document`.

Свойства устанавливаются посредством простого присваивания:

```
Document.Title = "OpenOffice.org Basic. Руководство программиста"
Document.Filename = "progman.odt"
```

Свойство, точно так же как нормальная переменная, имеет тип, который определяет, какое значение оно может запись. Предыдущие свойства `Filename` и `Title` имеют строковый тип.

Реальные свойства и имитация свойств

Большинство свойств объекта в OOo Basic определено также в описании UNO сервиса. В дополнение к этим “реальным” свойствам, есть также свойства в OOo Basic, которые состоят из двух методов на уровне UNO. Один из них используется для запроса значения свойства, а другой, соответственно, для его установки (`get` и `set` методы). Свойство фактически имитируется двумя методами. Символьные объекты в UNO, например, предоставляют методы `getPosition` и `setPosition`, через которые связанную ключевую точку можно вызвать и изменить. Программист OOo Basic может получить доступ к значению через свойство `Position`. Независимо от этого, оригинальные методы также доступны (в нашем примере, `getPosition` и `setPosition`).

Методы

Методы могут быть поняты как функции, которые имеют непосредственное отношение с объектом и через которые вызывают этот объект. Предыдущий объект `Document` может например, предоставлять метод `Save`, который можно вызвать следующим образом:

```
Document.Save()
```

Методы, точно так же как функции, могут содержать параметры и возвращать значение. Синтаксис таких вызовов методов похож на вызов классических функций. Вызов

```
Ok = Document.Save(True)
```

также определяет параметр `True` для объекта `Document` при вызове метода `Save`. Как только метод выполнится, `Save` сохраняет возвращаемое значение в переменной `Ok`.

Модули, Сервисы и Интерфейсы

OpenOffice.org предоставляет сотни сервисов. Чтобы обеспечивать краткий обзор этих сервисов, они были объединены в модули. Модули не имеют никакого другого функционального значения для программистов OOo Basic. Задаваемое имя сервиса, это - только имя модуля, которое имеет какое-нибудь значение, потому что оно должно быть также перечислено в имени. Полное имя сервиса состоит из выражения `com.sun.star`, которое определяет, что это – сервис OpenOffice.org, сопровождаемое именем модуля, таким как `frame`, и наконец фактическое название сервиса, такое как `Desktop`. Полное имя в приведенном примере было бы:

```
com.sun.star.frame.Desktop
```

В дополнение к терминам модуль и сервис, UNO вводит термин "*интерфейс*". В то время как этот термин может быть знакомым Java программистам, он не используется в Basic.

Интерфейс объединяет несколько методов. В самом строгом смысле слова, сервис в UNO не поддерживает методы, а скорее интерфейсы, которые в свою очередь предоставляют различные методы. Другими словами, методы задаются (как комбинация) для сервисов в интерфейсах. Эти детали могут представлять интерес в особенности для Java или C++ программистов, так как на этих языках, интерфейс необходим для запроса метода. В OOo Basic, это является неподходящим. Здесь, методы вызываются прямо посредством существующего объекта.

Для понимания API, однако, полезно иметь назначение методов на различные удобные интерфейсы, так как множество интерфейсов используются в различных сервисах. Если Вы знакомы с интерфейсом, то Вы можете передать ваше знание от одного сервиса другому.

Некоторые центральные интерфейсы используются настолько часто, что их показывают снова в конце этой главы, посвященной различным сервисам.

Инструменты для работы с UNO

Вопрос остается, какие объекты – или сервисы, если мы собираемся оставаться с терминологией UNO – какие свойства, методы и интерфейсы поддерживаются и как они могут быть определены. В дополнение к этому руководству, Вы можете получить больше информации об объектах из следующих источников: метод `supportsService`, методы отладки, а также в Руководстве разработчика и Справочнике по API.

Метод supportsService

Множество объектов UNO поддерживают метод `supportsService`, с помощью которого Вы можете установить, поддерживает ли объект отдельный сервис. Вызов

```
Ok = TextElement.supportsService("com.sun.star.text.Paragraph")
```

например, определяет, поддерживает ли объект `TextElement` сервис `com.sun.star.text.Paragraph`.

Отладочные свойства

Каждый объект UNO в OOo Basic знает, какие свойства, методы и интерфейсы он уже содержит. Он предоставляет свойства, которые возвращают их в форме списка. Соответствующие свойства:

- **DBG_properties** – возвращает строку, содержащую все свойства объекта;
- **DBG_methods** – возвращает строку, содержащую все методы объекта;
- **DBG_SupportedInterfaces** – возвращает строку, содержащую все интерфейсы, которые поддерживает объект.

Следующая программный код показывает, как `DBG_properties` и `DBG_methods` могут использоваться в реальных приложениях. Он сначала создает сервис `com.sun.star.frame.Desktop` и затем показывает поддерживаемые свойства и методы в окнах сообщений.

```
Dim Obj As Object
Obj = createUnoService("com.sun.star.frame.Desktop")

MsgBox Obj.DBG_Propierties
MsgBox Obj.DBG_methods
```

Используя `DBG_properties`, отметьте, что функция возвращает все свойства, которые один определенный сервис может теоретически поддерживать. Не гарантируется, однако, предусматривается ли их использование рассматриваемым объектом. Перед вызовом свойства, Вы должны использовать функцию `IsEmpty`, чтобы проверить, доступно ли оно на самом деле.

Справочная информация по API

Больше информации о доступных сервисах и их интерфейсах, методах и свойствах может быть найдено в Справочнике по API для OpenOffice.org. Он может быть найден на www.openoffice.org:

`http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html`

Краткий обзор нескольких главных интерфейсов

Некоторые интерфейсы OpenOffice.org могут быть найдены во многих частях OpenOffice.org API. Они определяют наборы методов для абстрактных задач, которые могут применяться в различных сложных ситуациях. Здесь, Вы найдете краткий обзор самых общих из этих интерфейсов.

Происхождение объектов объясняется далее в этом руководства. Здесь, обсуждаются только некоторые из абстрактных строн объектов, для которых OpenOffice.org API обеспечивает некоторые основные интерфейсы.

Создание контекстно-зависимых объектов

OpenOffice.org API обеспечивает две альтернативы для создания объектов. Один может быть найден в функции `createUnoService`, упомянутой в начале этой главы.

`createUnoService` создает объект, который может использоваться универсально. Такие объекты и сервисы также известны как *контекстно-независимые сервисы*.

В дополнение к контекстно-независимым сервисам, имеются также *контекстно-зависимые сервисы*, объекты которых полезны только когда используется в связке с другим объектом. Объект рисунка для документа электронной таблицы, например, может поэтому существовать только в связке с этим документом.

Интерфейс com.sun.star.lang.XMultiServiceFactory

Контекстно-зависимые объекты обычно создаются посредством метода объекта, от которого зависит создаваемый объект. `createInstance` метод, который определен в интерфейсе `XMultiServiceFactory`, используется исключительно в объектах документа.

Вышеупомянутый объект рисунок может, например, быть создан, следующим образом с использованием объекта электронной таблицы:

```
Dim RectangleShape As Object

RectangleShape = _
   Spreadsheet.createInstance("com.sun.star.drawing.RectangleShape")
```

Стиль абзаца в текстовом документе создается таким же образом:

```
Dim Style as Object

Style = Textdocument.createInstance("com.sun.star.style.ParagraphStyle")
```

Именованный доступ для подчиненных Объектов

Интерфейсы `XNameAccess` и `XNameContainer` используются в объектах, которые содержат зависимые объекты, к которым можно обратиться с использованием естественно-языкового имени.

В то время как `XNamedAccess` разрешает доступ к отдельным объектам, `XNameContainer` осуществляет вставку, изменение и удаление элементов.

Интерфейс com.sun.star.container.XNameAccess

Пример использования `XNameAccess` подготавливает объект лист в электронной таблице. Он объединяет все страницы в пределах электронной таблицы. К отдельным страницам получают доступ с использованием метода `getByName` из `XNameAccess`:

```
Dim Sheets As Object
Dim Sheet As Object

Sheets = Spreadsheet.Sheets
Sheet = Sheets.getByName("Sheet1")
```

Метод `getElementNames` предоставляет краткий обзор имен всех элементов. В результате он возвращает набор данных, содержащий имена. Следующий пример показывает, как все имена элементов электронной таблицы могут быть определены таким образом и показаны в цикле:

```
Dim Sheets As Object
Dim SheetNames
Dim I As Integer
```

```
Sheets = Spreadsheet.Sheets
SheetNames = Sheets.getElementNames

For I=LBound(SheetNames) To UBound(SheetNames)
  MsgBox SheetNames(I)
Next I
```

Метод `hasByName` интерфейса `XNameAccess` показывает, существует ли зависимый объект с указанным названием в пределах основного объекта. Следующий пример, следовательно, показывает сообщение, которое информирует пользователя, содержит ли объект электронной таблицы страницу с именем `Sheet1`.

```
Dim Sheets As Object

Sheets = Spreadsheet.Sheets

If Sheets.HasByName("Sheet1") Then
  MsgBox "Sheet1 имеется"
Else
  MsgBox "Sheet1 отсутствует"
End If
```

Интерфейс com.sun.star.container.XNameContainer

Интерфейс `XNameContainer` осуществляет вставку, удаление и изменение зависимых элементов в основном объекте. Важные функции - `insertByName`, `removeByName` и `replaceByName`.

Далее – практический пример этого. Он вызывает текстовый документ, который содержит объект `StyleFamilies` и использует его, чтобы сделать стили абзаца документа (ParagraphStyles) доступными.

```
Dim StyleFamilies As Object
Dim ParagraphStyles As Object
Dim NewStyle As Object

StyleFamilies = Textdoc.StyleFamilies
ParagraphStyles = StyleFamilies.getByName("ParagraphStyles")

ParagraphStyles.insertByName("NewStyle", NewStyle)
ParagraphStyles.replaceByName("ChangingStyle", NewStyle)
ParagraphStyles.removeByName("OldStyle")
```

Строка `insertByName` вставляет стиль `NewStyle` под тем же самым именем в объект `ParagraphStyles`. Строка `replaceByName` изменяет объект `ChangingStyle` на `NewStyle`. Наконец, вызов `removeByName` удаляет объект `OldStyle` из `ParagraphStyles`.

Доступ по индексу для подчиненных Объектов

Интерфейсы `XIndexAccess` и `XIndexContainer` используются в объектах, которые содержат зависимые объекты и к которым можно обратиться с использованием индекса.

`XIndexAccess` предоставляет методы для получения доступа к отдельным объектам. `XIndexContainer` предоставляет методы для вставки и удаления элементов.

Интерфейс com.sun.star.container.XIndexAccess

`XIndexAccess` предоставляет методы `getByIndex` и `getCount` для вызова зависимых объектов. `getByIndex` предоставляет объект с указанным индексом. `getCount` возвращает количество доступных объектов.

```
Dim Sheets As Object
Dim Sheet As Object
```

```
Dim I As Integer

Sheets = Spreadsheet.Sheets

For I = 0 to Sheets.getCount() - 1
  Sheet = Sheets.getByIndex(I)
  ' Редактирование листа
Next I
```

Пример показывает цикл, который пробегает последовательно по всем листам один за другим и сохраняет ссылку на каждый в объектной переменной `Sheet`. При работе с индексами, отметьте, что `getCount` возвращает число элементов. Элементы в `getByIndex` нумеруются, начиная с 0. Счетчик цикла поэтому изменяется в диапазоне от `0` до `getCount()-1`.

Интерфейс com.sun.star.container.XIndexContainer

Интерфейс `XIndexContainer` предоставляет функции `removeByIndex` и `insertByIndex`. Параметры такие же, как и у соответствующих функций в `XNameContainer`.

Повторяющийся доступ для подчиненных объектов

В некоторых случаях, объект может содержать список зависимых объектов, к которым нельзя обратиться по имени или индексу. В этих ситуациях применимы интерфейсы `XEnumeration` и `XenumerationAccess`. Они обеспечивают механизм, через который все зависимые элементы объекта могут быть переданы, шаг за шагом, без необходимости использования прямой адресации.

Интерфейсы com.sun.star.container.XEnumeration и com.sun.star.container.XenumerationAccess

Базовый объект должен предоставлять интерфейс `XEnumerationAccess`, который содержит только метод `createEnumeration`. Он возвращает вспомогательный объект, который в свою очередь предоставляет интерфейс `XEnumeration` с методами `hasMoreElements` и `nextElement`. Через них, Вы имеете доступ к зависимым объектам.

Следующий пример перебирает все абзацы текста:

```
Dim ParagraphEnumeration As Object
Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

While ParagraphEnumeration.hasMoreElements()
  Paragraph = ParagraphElements.nextElement()
Wend
```

Пример сначала создает вспомогательный объект `ParagraphEnumeration`. Он последовательно возвращает отдельные абзацы текста в цикле. Цикл заканчивается, как только метод `hasMoreElements` возвращает значение `False`, сигнализируя, что был достигнут конец текста.

Глава 5. Работа с документами OpenOffice.org

OpenOffice.org API был структурирован так, чтобы многие из его частей, насколько это возможно, могли использоваться универсально для различных задач. Он включает интерфейсы и сервисы для создания, открытия, сохранения, преобразования и печати документов и для управления шаблонами. Так как эти функциональные области доступны во всех типах документов, они объясняются сначала в этой главе.

StarDesktop

При работе с документами, есть два сервиса, которые используются наиболее часто:

- Сервис `com.sun.star.frame.Desktop`, который является подобным основному сервису OpenOffice.org. Он предоставляет функции для объекта структуры OpenOffice.org, под которым классифицирует все окна документа. Документы также могут быть созданы, открыты и импортированы с использованием этого сервиса;
- Основные функциональные возможности для отдельных объектов документа обеспечиваются сервисом `com.sun.star.document.OfficeDocument`. Он предоставляет методы для сохранения, экспорта и печати документов.

Сервис `com.sun.star.frame.Desktop` открывается автоматически, когда OpenOffice.org запускается. Чтобы сделать это, OpenOffice.org создает объект, к которому может быть получен доступ посредством глобального имени `StarDesktop`.

Самый важный интерфейс `StarDesktop` – `com.sun.star.frame.XComponentLoader`. Он в основном содержит метод `loadComponentFromURL`, который является ответственным за создание, импортирование, и открытие документов.

Название объекта StarDesktop относится ко времени StarOffice 5, в котором все окна документа были вложены в одно общее приложение по имени StarDesktop. В существующей версии OpenOffice.org, больше не используется видимый StarDesktop. Имя StarDesktop было, однако, сохранено для структурного объекта OpenOffice.org, потому что оно ясно указывает, что это – основной объект для всего приложения.

Объект StarDesktop берет на себя положение преемника объекта `Application` StarOffice 5, который ранее применялся как корневой объект. В отличие от старого объекта `Application` он, тем не менее, прежде всего ответственен за открытие новых документов. Функции, свойственные старому объекту `Application` для управления экранным изображением StarOffice (например, `FullScreen`, `FunctionBarVisible`, `Height`, `Width`, `Top`, `Visible`) больше не используются.

Примечание Принимая во внимание, что к активному документу в Word получают доступ через `Application.ActiveDocument`, а в Excel через `Application.ActiveWorkbook`, в OpenOffice.org, `StarDesktop` ответственен за эту задачу. К активному объекту документа получают доступ в OpenOffice.org через свойство `StarDesktop.CurrentComponent`.

Основная информация о документах в OpenOffice.org

Работая с документами OpenOffice.org, полезно иметь дело с некоторыми из основных вопросов управления документами в OpenOffice.org. Они включают метод, в котором имена файлов представлены для документов OpenOffice.org, так же как формат, в котором сохранены файлы.

Имена файлов в URL нотации

Так как OpenOffice.org – платформо-независимое приложение, оно использует для имен файлов URL нотацию (которая является независимой от любой операционной системы), как определено в интернет-стандарте RFC 1738. Стандартные имена файлов, использующие эту систему начинаются с префикса

`file:///`

сопровождаемый локальным путем. Если имя файла включает подкаталоги, то они разделяются одним *слешем* (/), а не обратным слешем, обычно используемым в Windows. Следующий путь ссылается на файл test.sxw в каталоге doc на устройстве C:.

`file:///C:/doc/test.sxw`

Для преобразования локального имени файла в URL, OpenOffice.org предоставляет функцию `ConvertToUrl`. Для преобразования URL в локальное имя файла, OpenOffice.org предоставляет функцию ConvertFromUrl:

```
MsgBox ConvertToUrl("C:\doc\test.sxw")
      ' выдает file:///C:/doc/test.sxw

MsgBox ConvertFromUrl("file:///C:/doc/test.sxw")
      ' выдает (под Windows) c:\doc\test.sxw
```

Пример преобразует локальное имя файла в URL и отображает его в окне сообщения. Потом преобразует URL в локальное имя файла и также отображает его.

Интернет-стандарт RFC 1738, на котором это основано, разрешает использование символов `0-9`, `a-z`, и `A-Z`. Все другие символы вставляются в URL как Esc-коды. Чтобы сделать это, они преобразуются в свое шестнадцатеричное значение по набору символов ISO 8859-1 (ISO-Latin) и предваряются знаком процента. Пробел в локальном имени файла поэтому, например, становится `%20` в URL.

Файловый формат XML

Начиная с Версии 1.0, OpenOffice.org поддерживает файловый формат основанный на XML. Благодаря использованию XML, пользователь имеет также возможность открывать и редактировать файлы в других программах.

Сжатие файлов

Так как XML основан на стандартных текстовых файлах, результирующие файлы являются обычно очень большими. OpenOffice.org поэтому сжимает файлы и сохраняет их как ZIP файл. Посредством выбора метода `storeAsURL`, пользователь может сохранить

оригинальные XML файлы непосредственно. См. “Параметры метода storeAsURL” на странице 61.

Создание, открытие и импорт документов

Документы открываются, импортируются и создаются с использованием метода

```
StarDesktop.loadComponentFromURL(URL, Frame, _
            SearchFlags, FileProperties)
```

Первый параметр `loadComponentFromURL` определяет URL связанного файла.

В качестве второго параметра, `loadComponentFromURL` ожидает имя для фреймового объекта окна, которое OpenOffice.org создает внутри для его управления. Здесь обычно определяется предопределенное имя `_blank`, и оно гарантирует, что OpenOffice.org создает новое окно. Альтернативно, может также быть определено `_hidden`, и оно гарантирует, что соответствующий документ будет загружен, но останется невидимым.

Используя эти параметры, пользователь может открыть документ OpenOffice.org, так как двум последним параметрам могут быть присвоены символы-заполнители (фиктивные значения):

```
Dim Doc As Object
Dim Url As String
Dim Dummy()

Url = "file:///C:/test.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "_blank", 0, Dummy())
```

Предыдущий вызов открывает файл `text.sxw` и показывает его в новом окне.

Любое число документов может быть открыто таким образом в OOo Basic и затем отредактировано с использованием полученного объекта документ (переменная `Doc`).

Примечание `StarDesktop.loadComponentFromURL` заменяет методы `Documents.Add` и `Documents.Open` старого StarOffice API.

Замена содержимого окна документа

Названные значения `_blank` и `_hidden` для параметра `Frame` гарантируют, что OpenOffice.org создает новое окно для каждого запроса из `loadComponentFromURL`. В небольшом количестве ситуаций, полезно заменить содержимое существующего окна. В этом случае, объект фрейма окна должен содержать явное имя. Отметьте, что это имя не должно начинаться со знака подчеркивания. Кроме того, параметр `SearchFlags` должен быть установлен так, чтобы соответствующая структура была создана, если она еще не существует. Соответствующая константа для SearchFlags:

```
SearchFlags = com.sun.star.frame.FrameSearchFlag.CREATE + _
      com.sun.star.frame.FrameSearchFlag.ALL
```

Следующий пример показывает, как содержание открытого окна может быть заменено при помощи параметра `Frame` и `SearchFlags`:

```
Dim Doc As Object
Dim Dummy()
Dim Url As String
Dim SearchFlags As Long

SearchFlags = com.sun.star.frame.FrameSearchFlag.CREATE + _
        com.sun.star.frame.FrameSearchFlag.ALL
```

```
Url = "file:///C:/test.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "MyFrame", _
      SearchFlags, Dummy)

MsgBox "Press OK to display the second document."

Url = "file:///C:/test2.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "MyFrame", _
      SearchFlags, Dummy)
```

Пример сначала открывает файл `test.sxw` в новом окне с именем фрейма `MyFrame`. Как только окно сообщения было подтверждено, содержание окна заменяется файлом `test2.sxw`.

Параметры метода loadComponentFromURL

Четвертый параметр функции `loadComponentFromURL` – набор данных `PropertyValue`, который предоставляет OpenOffice.org различные параметры для открытия и создания документов. Набор данных должен обеспечить структуру `PropertyValue` для каждого параметра, в котором название параметра сохраняется как строка, а также соответствующее значение.

`loadComponentFromURL` поддерживает следующие параметры:

- **AsTemplate (Boolean)** – если `true`, загружает новый, документ без названия из указанного URL. Если `false`, файлы шаблона загружаются для редактирования;
- **CharacterSet (String)** – определяет набор символов документа;
- **FilterName (String)** – определяет специальный фильтр для функции loadComponentFromURL. Имена доступных фильтров, определены в файле `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`;
- **FilterOptions (String)** – определяет дополнительные параметры для фильтров;
- **JumpMark (String)** – если документ уже открывался, переход к позиции, определенной в `JumpMark`;
- **Password (String)** – передает пароль для защищенного файла;
- **ReadOnly (Boolean)** – загружает документ в режиме только для чтения.

Следующий пример показывает, как текстовый файл, разделенный запятыми может быть открыт в OpenOffice.org Calc, с использованием параметра `FilterName`.

```
Dim Doc As Object
Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
Dim Url As String

Url = "file:///C:/csv.doc"

FileProperties(0).Name = "FilterName"
FileProperties(0).Value ="scalc: Text - txt - csv (StarOffice Calc)"

Doc = StarDesktop.loadComponentFromURL(Url, "_blank", 0, FileProperties())
```

Набор данных `FileProperties` содержит именно одно значение, потому что делается запись одного параметра. Свойство `Filtername` определяет, что OpenOffice.org использует текстовый фильтр OpenOffice.org Calc для открытия файла.

Создание новых документов

OpenOffice.org автоматически создает новый документ, если документ, определенный в URL - шаблон.

Альтернативно, если необходим только пустой документ без какой-нибудь адаптации, может быть определен `private:factory` URL:

```
Dim Dummy()
Dim Url As String
Dim Doc As Object

Url = "private:factory/swriter"
Doc = StarDesktop.loadComponentFromURL(Url, "_blank", 0, Dummy())
```

Вызов создает пустой документ OpenOffice.org Writer.

Объекты документа

Функция `loadComponentFromURL`, введенная в предыдущем разделе возвращает объект документ. он поддерживает сервис `com.sun.star.document.OfficeDocument`, который в свою очередь предоставляет два центральных интерфейса:

- Интерфейс `com.sun.star.frame.XStorable`, который является ответственным за сохранение документов;
- Интерфейс `com.sun.star.view.XPrintable`, который содержит методы для печати документов.

Примечание Переходя к OpenOffice.org 2, Вы найдете, что функциональные возможности объектов документа остались тем же самым в большей части. Объекты документа, например, все еще обеспечивают методы для сохранения и печатати документов. Имена и параметры методов, однако, изменились.

Сохранение и экспорт документов

Документы OpenOffice.org сохраняются непосредственно через объект документа. С этой целью доступен метод `store` интерфейса `com.sun.star.frame.XStorable`:

```
Doc.store()
```

Этот запрос функционирует при условии, что документу уже было назначено место в памяти. Дело обстоит не так для новых документов. В этом случае, используется метод `storeAsURL`. Этот метод также определен в `com.sun.star.frame.XStorable` и может использоваться для определения местоположения документа:

```
Dim URL As String
Dim Dummy()

Url = "file:///C:/test3.sxw"

Doc.storeAsURL(URL, Dummy())
```

В дополнение к предыдущим методам, `com.sun.star.frame.XStorable` также предоставляет некоторые вспомогательные методы, которые являются полезными при сохранении документа. Вот они:

- **hasLocation()** – определяет, был ли документу уже назначен URL;
- **isReadonly()** – определяет, имеет ли документ защиту только для чтения;
- **isModified()** – определяет, был ли документ изменен, после того как было выполнено его последнее сохранение.

Код для сохранения документа может быть расширен этими опциями так, чтобы документ

сохранялся только, если объект был фактически изменен, и имя файла запрашивалось только, если это действительно необходимо:

```
If (Doc.isModified) Then
  If (Doc.hasLocation And (Not Doc.isReadOnly)) Then
    Doc.store()
  Else
    Doc.storeAsURL(URL, Dummy())
  End If
End If
```

Пример сначала проверяет, был ли соответствующий документ изменен, после его последнего сохранения. Процесс сохранения выполняется только если это так. Если документу уже был назначен URL и документ не является только для чтения, он сохраняется под существующим URL. Если он не имеет URL или был открыт в режиме только для чтения, он сохраняется под новым URL.

Параметры метода storeAsURL

Как и с методом `loadComponentFromURL`, некоторые параметры могут быть определены в форме набора данных `PropertyValue` используемого методом `storeAsURL`. Они определяют процедуру, используемую OpenOffice.org при сохранении документа. `storeAsURL` имеет следующие параметры:

- **CharacterSet (String)** – определяет набор символов документа;
- **FilterName (String)** – определяет специальный фильтр для функции `loadComponentFromURL`. Имена доступных фильтров, определены в файле `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`;
- **FilterOptions (String)** – определяет дополнительные параметры для фильтров;
- **Overwrite (Boolean)** – позволяет уже существующий файл перезаписать без вопроса;
- **Password (String)** – передает пароль для защищенного файла;
- **Unpacked (Boolean)** – сохраняет документ (без сжатия) в подкаталогах.

Следующий пример показывает, как параметр `Overwrite` может использоваться в связке с `storeAsURL`:

```
Dim Doc As Object
Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
Dim sUrl As String

' ... инициализация Doc
sUrl = "file:///c:/test3.sxw"

FileProperties(0).Name = "Overwrite"
FileProperties(0).Value = True

Doc.storeAsURL(sUrl, mFileProperties())
```

Пример сохраняет Doc под указанным именем файла в том случае, если уже существует файл с данным именем.

Печать документов

Подобно сохранению, документы печатаются прямо посредством объекта документа. Метод `Print` интерфейса `com.sun.star.view.Xprintable` предназначен для этой цели. В его самой простой форме, вызов печати:

```
Dim Dummy()

Doc.print(Dummy())
```

Как и в случае метода `loadComponentFromURL`, параметр `Dummy` - набор данных `PropertyValue`, посредством которого OpenOffice.org может определить несколько параметров для печати.

Параметры метода print

Метод `print` ожидает набор данных `PropertyValue` как параметр, который отражает параметры настройки диалога печати OpenOffice.org:

- **CopyCount (Integer)** – определяет число копий, которые будут напечатаны;
- **FileName (String)** – печатает документ в указанный файл;
- **Collate (Boolean)** – советует принтеру складывать в нужном порядке страницы копий;
- **Sort (Boolean)** – осуществляет сортировку страниц при печати нескольких копий (`CopyCount > 1`);
- **Pages (String)** – содержат список страниц, которые будут напечатаны (синтаксис как определено в диалоге печати).

Следующий пример показывает, как несколько страниц документа могут быть распечатаны с использованием параметра `Pages`:

```
Dim Doc As Object
Dim PrintProperties(0) As New com.sun.star.beans.PropertyValue

PrintProperties(0).Name="Pages"
PrintProperties(0).Value="1-3; 7; 9"

Doc.print(PrintProperties())
```

Выбор принтера и параметров настройки

Интерфейс `com.sun.star.view.XPrintable` обеспечивает свойство `Printer`, которое позволяет осуществлять выбор принтера. Это свойство получает набор данных `PropertyValue` со следующими параметрами настройки:

- **Name (String)** – определяет имя принтера;
- **PaperOrientation (Enum)** – определяет ориентацию бумаги (значение `com.sun.star.view.PaperOrientation.PORTRAIT` для книжной ориентации, `com.sun.star.view.PaperOrientation.LANDSCAPE` для альбомной ориентации);
- **PaperFormat (Enum)** – определяет формат бумаги (например, `com.sun.star.view.PaperFormat.A4` для DIN A4 или `com.sun.star.view.PaperFormat.Letter` для US Letters);
- **PaperSize (Size)** – определяет размер бумаги в сотых долях миллиметра.

Следующий пример показывает, как может быть изменен принтер и установлен размер бумаги с помощью свойства `Printer`.

```
Dim Doc As Object
Dim PrinterProperties(1) As New com.sun.star.beans.PropertyValue
Dim PaperSize As New com.sun.star.awt.Size

PaperSize.Width = 20000 ' соответствует 20 см
PaperSize.Height = 20000 ' соответствует 20 см

PrinterProperties(0).Name="Name"
PrinterProperties(0).Value="My HP Laserjet"

PrinterProperties(1).Name="PaperSize"
PrinterProperties(1).Value=PaperSize
```

```
Doc.Printer = PrinterProperties()
```

Пример определяет объект по имени `PaperSize` типа `com.sun.star.awt.Size`. Это необходимо для определения размера бумаги. Кроме того, он создает набор данных для двух записей `PropertyValue` по имени `PrinterProperties`. Этот набор данных инициализируется значениями, которые присваиваются свойству Printer. С точки зрения UNO, принтер не реальное свойство, а имитируемое.

Стили

Стилями называют списки, содержащие атрибуты форматирования. Они используются во всех приложениях OpenOffice.org и помогают значительно упрощать форматирование. Если пользователь изменяет один из атрибутов стиля, то OpenOffice.org автоматически настраивает все разделы документа в зависимости от атрибута. Пользователь может, например, изменить тип шрифта заголовков всех уровней посредством центральной модификации в документе. В зависимости от соответствующих типов документов, OpenOffice.org распознает целый диапазон различных типов стилей.

OpenOffice.org Writer поддерживает:

- стили символа;
- стили абзаца;
- стили врезок;
- стили страницы;
- стили списка.

OpenOffice.org Calc поддерживает:

- стили ячейки;
- стили страницы.

OpenOffice.org Impress поддерживает:

- стили символьных элементов;
- стили презентаций.

В терминологии OpenOffice.org, различные типы стилей вызывают `StyleFamilies` в соответствии с сервисом `com.sun.star.style.StyleFamily`, на котором они базируются. К `StyleFamilies` получают доступ посредством объекта документ:

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim CellStyles As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
CellStyles = StyleFamilies.getByName("CellStyles")
```

Пример использует свойство `StyleFamilies` документа электронной таблицы для создания списка, содержащего все доступные стили ячейки.

К отдельным стилям можно получить доступ напрямую посредством индекса:

```
Dim Doc As Object
```

```
Dim Sheet As Object
Dim StyleFamilies As Object
Dim CellStyles As Object
Dim CellStyle As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
CellStyles = StyleFamilies.getByName("CellStyles")

For I = 0 To CellStyles.Count - 1
  CellStyle = CellStyles(I)
  MsgBox CellStyle.Name
Next I
```

Цикл, добавленный к предыдущему примеру отображает имена всех стилей ячейки один за другим в окне сообщения.

Подробности о различных вариантах форматирования

Каждый тип стиля обеспечивает целый набор отдельных свойств форматирования. Вот - краткий обзор самых важных свойств форматирования и мест, в которых они объясняются:

- свойства символа, Глава 6, Текстовые документы, сервис `com.sun.star.style.CharacterProperties`;
- свойства абзаца, Глава 6, Текстовые документы, сервис `com.sun.star.text.Paragraph`;
- свойства ячейки, Глава 7, Документы электронной таблицы, сервис `com.sun.star.table.CellProperties`;
- свойства страницы, Глава 7, Документы электронной таблицы, сервис `com.sun.star.style.PageStyle`;
- свойства символьных элементов, Глава 7, Документы электронной таблицы, Различные сервисы.

Свойство форматирования ни в коем случае не ограничены приложением, в котором они объясняются, но вместо этого может использоваться универсально. Например, большинство свойств страницы, описанных в Главе 7, могут использоваться не только в OpenOffice.org Calc, но также и в OpenOffice.org Writer.

Более подробная информация о работе со стилями может быть найдена в разделе *Значения по умолчанию для свойств символов и абзацев* в Главе 6.

Глава 6. Текстовые документы

В дополнение к чистым строкам, текстовые документы также содержат информацию о форматировании. Она может появиться в любом месте в тексте. Структура более того, может усложняться таблицами. Они включают не только одномерные строки, но также и двумерные наборы. Большинство программ обработки текстов теперь, наконец, обеспечивают выбор размещения графических объектов, текстовых врезок и других объектов в пределах текста. Они могут быть вне потока текста и могут быть размещены где-нибудь на странице.

Эта глава представляет основные интерфейсы и сервисы текстовых документов. Первый раздел имеет дело с анатомией текстовых документов и концентрируется на том, как программа OOo Basic может использоваться для выполнения повторяющихся операций в документах OpenOffice.org. Он сосредотачивается на абзацах, частях абзаца и их форматировании.

Второй раздел сосредоточен на эффективной работе с текстовыми документами. С этой целью, OpenOffice.org предоставляет несколько вспомогательных объектов, типа объекта `TextCursor`, которые выходят за пределы определенные в первом разделе.

Третий раздел выходит за пределы работы с текстами. Он концентрируется на таблицах, текстовых врезках, текстовых полях, закладках, оглавлениях и т.п.

Информация о том, как создавать, открывать, сохранять и печатать документы описана в Главе 5, потому что эти операции могут использоваться не только для текстовых документов, но также и для других типов документов.

Структура текстовых документов

Текстовый документ по существу может содержать четыре типа информации:

- фактический текст;
- стили для форматирования символов, абзацев, и страниц;
- нетекстовые элементы, такие как таблицы, диаграммы и изображения;
- общие параметры настройки для текстового документа.

В этот разделе рассматриваются текст и соответствующие варианты форматирования.

Примечание Проект OpenOffice.org 2 API для OpenOffice.org Writer отличается от предыдущей версии. Старая версия API концентрировалась на работе с объектом `Selection`, была плохо пригодна к идее пользовательского интерфейса для конечных пользователей, работа которых основана на

управляемом мышью выделении.

OpenOffice.org 2 API заменил эти связи между пользовательским интерфейсом и интерфейсом программиста. Программист поэтому имеет параллельный доступ ко всем частям приложения и может работать с объектами из различных подразделов документа одновременно. Старый объект Selection больше не доступен.

Абзацы и части абзаца

Основная часть текстового документа состоит из последовательности абзацев. Они не именуются, ни вносятся в указатель и поэтому нет никакого возможного пути прямого обращения к отдельным абзацам. Абзацы, однако, могут последовательно перебираться с помощью объекта `Enumeration`, описанного в Главе 4. Это позволяет редактировать абзацы.

При работе с объектом `Enumeration`, должен, однако, быть отмечен один специальный сценарий: он возвращает не только абзацы, но также и таблицы (строго говоря, в OpenOffice.org Writer, таблица - специальный тип абзаца). Перед обращением к возвращенному объекту, Вы должны поэтому проверить, поддерживает ли возвращенный объект сервис `com.sun.star.text.Paragraph` для абзаца или сервис `com.sun.star.text.TextTable` для таблицы.

Следующий пример перебирает в цикле содержимое текстового документа и использует сообщение в каждом случае, чтобы проинформировать пользователя, является ли рассматриваемый объект абзацем или таблицей.

```
Dim Doc As Object
Dim Enum As Object
Dim TextElement As Object

' создание объекта document
Doc = StarDesktop.CurrentComponent

' создание объекта enumeration
Enum = Doc.Text.createEnumeration

' цикл по всем текстовым элементам
While Enum.hasMoreElements
  TextElement = Enum.nextElement

  If TextElement.supportsService("com.sun.star.text.TextTable") Then
    MsgBox "The current block contains a table."
  End If

  If TextElement.supportsService("com.sun.star.text.Paragraph") Then
    MsgBox "The current block contains a paragraph."
  End If
Wend
```

Пример создает объект документ `Doc`, который ссылается на текущий документ OpenOffice.org. При помощи `Doc`, пример создает объект `Enumeration`, который перебирает отдельные части текста (абзацы и таблицы) и присваивает текущий элемент объекту `TextElement`. Пример использует метод `supportsService` для проверки, является ли `TextElement` абзацем или таблицей.

Абзацы

Сервис `com.sun.star.text.Paragraph` предоставляет доступ к содержимому абзаца. Текст в абзаце может быть извлечен и изменен с использованием свойства `String`:

```
Dim Doc As Object
```

```
Dim Enum As Object
Dim TextElement As Object

Doc = StarDesktop.CurrentComponent
Enum = Doc.Text.createEnumeration

While Enum.hasMoreElements
  TextElement = Enum.nextElement

  If TextElement.supportsService("com.sun.star.text.Paragraph") Then
    TextElement.String = Replace(TextElement.String, "you", "U")
    TextElement.String = Replace(TextElement.String, "too", "2")
    TextElement.String = Replace(TextElement.String, "for", "4")
  End If
Wend
```

Пример открывает текущий текстовый документ и пробегает по нему с помощью объекта `Enumeration`. Он использует свойство TextElement.String во всех абзацах, чтобы получить доступ к содержимому абзаца и заменяет строки `you`, `too` и `for` на символы `U`, `2` и `4` соответственно. Функция `Replace`, используемая для замены не находится в пределах стандартных лингвистических возможностей OOo Basic. Это случай использования функции примера, описанной в Главе 3 в разделе “Поиск и замена” на стр. 33.

Примечание Содержание процедуры, описанной здесь для получения доступа к абзацам текста сопоставимо со списком абзацов, используемым в VBA, который предоставляется объектами `Range` и `Document`, доступных там. Принимая во внимание, что в VBA к абзацам получают доступ по их номеру (например, в соответствии с запросом `Paragraph(1)`), в OOo Basic должен использоваться объект `Enumeration`, описанный ранее.

Нет никакого прямого двойника в OOo Basic для списков `Characters`, `Sentences` и `Words` предоставляемых в VBA. Вы, однако, имеете параметр переключения для `TextCursor`, который учитывает навигацию на уровне символов, предложений и слов (обратитесь к *TextCursor*).

Части абзаца

Предыдущий пример может изменить текст как указано, но он может в некоторых случаях также нарушить форматирование.

Это потому, что абзац в свою очередь состоит из индивидуальных подобъектов. Каждый из этих подобъектов содержит свою собственную информацию о форматировании. Если в середине абзаца, например, содержится слово, напечатанное жирным, то он будет представлен в OpenOffice.org тремя частями абзаца: часть перед жирным шрифтом, слово выделенное жирным, и наконец часть после жирного шрифта, которая снова изображается как нормальный текст.

Если текст абзаца изменяется с использованием свойства абзаца `String`, то OpenOffice.org сначала удаляет старые части абзаца и вставляет новую часть абзаца. Форматирование предыдущих разделов после этого теряется.

Чтобы предотвратить этот эффект, пользователь может получить доступ к соответствующим частям абзаца, а не ко всему абзацу. Абзацы, с этой целью, обеспечивают их собственным объектом Enumeration. Следующий пример показывает двойной цикл, который осуществляет перебор всех абзацев текстового документа и частей абзаца, которые они содержат, и применяет процессы замены из предыдущего примера:

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim TextElement As Object
```

```
Dim TextPortion As Object

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' цикл по всем абзацам
While Enum1.hasMoreElements
  TextElement = Enum1.nextElement
  If TextElement.supportsService("com.sun.star.text.Paragraph") Then
    Enum2 = TextElement.createEnumeration

    ' цикл по всем под-абзацам
    While Enum2.hasMoreElements
      TextPortion = Enum2.nextElement
      MsgBox "'" & TextPortion.String & "'"
      TextPortion.String = Replace(TextPortion.String, "you", "U")
      TextPortion.String = Replace(TextPortion.String, "too", "2")
      TextPortion.String = Replace(TextPortion.String, "for", "4")
    Wend
  End If
Wend
```

Пример осуществляет перебор текстового документа в двойном цикле. Внешний цикл обращается к текстовым абзацам. Внутренний цикл обрабатывает части в этих абзацах. Код примера изменяет содержание в каждой из этих частей абзаца, используя строковое свойство `String`, как имеет место в предыдущем примере для абзацев. Так как части абзаца редактируются непосредственно, их информация о форматирования сохраняется при замене строк.

Форматирование

Есть различные способы форматирования текста. Самый легкий путь состоит в том, чтобы назначить свойства форматирования непосредственно на текстовую последовательность. Это называют непосредственным форматированием. Прямое форматирование используется в особенности в небольших документах, потому что форматирование может быть назначено пользователем с помощью мыши. Вы можете, например, выделить определенное слово в пределах текста, используя жирный шрифт или выровнять по центру строку.

В дополнении к непосредственному форматированию, Вы можете также форматировать текст с использованием стилей. Это называют косвенным форматированием. При косвенном форматировании, пользователь назначает предопределенный стиль соответствующей части текста. Если разметка текста изменяется позднее, пользователь должен изменить только стиль. OpenOffice.org тогда изменяет все части текста, которые используют этот стиль для отображения.

Примечание В VBA, свойства форматирования объекта обычно распространяется на диапазон подобъектов (например, `Range.Font`, `Range.Borders`, `Range.Shading`, `Range.ParagraphFormat`). К свойствам получают доступ посредством каскадных выражений (например, `Range.Font.AllCaps`). В OOo Basic, с другой стороны, свойства форматирования доступны непосредственно, с использованием соответствующих объектов (`TextCursor`, `Paragraph`, и так далее). Вы найдете краткий обзор свойств символов и абзацев доступных в OpenOffice.org в следующих двух разделах.

Примечание В старом StarOffice API, текст был по существу форматирован с использованием объекта `Selection` и его зависимых объектов (например, `Selection.Font`, `Selection.Paragraph` и `Selection.Border`). В новом OpenOffice.org API, свойства форматирования могут быть найдены в каждом

объекте (`Paragraph`, `TextCursor`, и так далее) и могут быть применены непосредственно. Список доступных свойств символов и абзацев может быть найден в следующих абзацах.

Свойства символа

Те свойства форматирования, которые относятся к отдельным символам, описываются как свойства символа. Они включают жирный шрифт и тип шрифта. Объекты, которые позволяют устанавливать свойства символа, должны поддерживать интерфейс `com.sun.star.style.CharacterProperties`. OpenOffice.org предоставляет целый набор сервисов, которые поддерживают этот сервис. Он включает ранее описанный сервис `com.sun.star.text.Paragraph` для абзацев, а также сервис `com.sun.star.text.TextPortion` для частей абзаца.

Сервис `com.sun.star.style.CharacterProperties` не предоставляет никаких интерфейсов, но вместо этого предлагает диапазон свойств, через которые свойства символа могут определяться и вызываться. Полный список всех свойств символа может быть найден в справочнике по OpenOffice.org API. Следующий список описывает самые важные свойства:

- **CharFontName (String)** – имя выбранного типа шрифта;
- **CharColor (Long)** – цвет текста;
- **CharHeight (Float)** – высота символа в пунктах (pt);
- **CharUnderline (Constant group)** – тип подчеркивания (константы в соответствии с `com.sun.star.awt.FontUnderline`);
- **CharWeight (Constant group)** – вес шрифта (константы в соответствии с `com.sun.star.awt.FontWeight`);
- **CharBackColor (Long)** – фоновый цвет;
- **CharKeepTogether (Boolean)** – подавление автоматического разрыва строк;
- **CharStyleName (String)** – имя стиля символа.

Свойства абзаца

Информация о форматировании, которая относится не к отдельным символам, а ко всему абзацу, как полагают, является свойством абзаца. Она включает расстояние абзаца от края страницы, а также межстрочный интервал. Свойства абзаца доступны через сервис `com.sun.star.style.ParagraphProperties`.

Как раз свойства абзаца доступны в различных объектах. Все объекты, которые поддерживают сервис `com.sun.star.text.Paragraph` также обеспечивают поддержку для свойств абзаца в `com.sun.star.style.ParagraphProperties`.

Полный список свойств абзаца может быть найден в справочнике по OpenOffice.org API. Самые общие свойства абзаца:

- **ParaAdjust (enum)** – вертикальная ориентация текста (константы в соответствии с `com.sun.star.style.ParagraphAdjust`);
- **ParaLineSpacing (struct)** – межстрочный интервал (структура в соответствии с `com.sun.star.style.LineSpacing`);

- **ParaBackColor (Long)** – фоновый цвет;
- **ParaLeftMargin (Long)** – левое поле в сотых долях миллиметра;
- **ParaRightMargin (Long)** – правое поле в сотых долях миллиметра;
- **ParaTopMargin (Long)** – верхнее поле в сотых долях миллиметра;
- **ParaBottomMargin (Long)** – нижнее поле в сотых долях миллиметра;
- **ParaTabStops (Array of struct)** – тип и положение позиций табуляции (массив структур типа `com.sun.star.style.TabStop`);
- **ParaStyleName (String)** – имя стиля абзаца.

Пример: простой экспорт в HTML

Следующий пример демонстрирует, как работать с информацией о форматировании. Он повторяет от начала до конца текстовый документ и создает простой файл HTML. С этой целью каждый параграф записывается в своем собственном элементе HTML <P>. Части параграфа, отображаемые жирным шрифтом отмечаются с использованием элемента HTML <B> при экспорте.

```
Dim FileNo As Integer, Filename As String, CurLine As String
Dim Doc As Object
Dim Enum1 As Object, Enum2 As Object
Dim TextElement As Object, TextPortion As Object

Filename = "c:\text.html"
FileNo = Freefile
Open Filename For Output As #FileNo
Print #FileNo, "<HTML><BODY>"

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' цикл по всем абзацам
While Enum1.hasMoreElements
  TextElement = Enum1.nextElement
  If TextElement.supportsService("com.sun.star.text.Paragraph") Then
    Enum2 = TextElement.createEnumeration
    CurLine = "<P>"

    ' цикл по всем частям абзаца
    While Enum2.hasMoreElements
      TextPortion = Enum2.nextElement
      If TextPortion.CharWeight = com.sun.star.awt.FontWeight.BOLD THEN
        CurLine = CurLine & "<B>" & TextPortion.String & "</B>"
      Else
        CurLine = CurLine & TextPortion.String
      End If
    Wend

    ' вывод строки
    CurLine = CurLine & "</P>"
    Print #FileNo, CurLine

  End If
Wend

' запись завершения HTML
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

Основная структура примера основана на примере для перебора частей текстовых абзацев, уже обсуждавшегося ранее. Были добавлены функции записи в HTML файл, а так же код, который проверяет вес шрифта соответствующих частей текста и обеспечивает части абзаца с жирном шрифтом соответствующим HTML признаком.

Значения по умолчанию для свойств символа и абзаца

Прямое форматирование всегда берет приоритет над *косвенным* форматированием. Другими словами, форматированию с использованием стилей назначается более низкий приоритет чем прямому форматированию в тексте.

Установить, был ли раздел документа форматирован прямо или косвенно, не легко. Символьная панель инструментов, предоставляемая OpenOffice.org показывает общие свойства текста, такие как тип шрифта, вес и размер. Однако неясно, основаны соответствующие параметры текста на стиле, или используют прямое форматирование.

OOo Basic предоставляет метод `getPropertyState`, которым программисты могут проверить, как определенное свойство было форматировано. Как параметр, он берет имя свойства и возвращает константу, которая обеспечивает информацию о происхождении форматирования. Следующие ответы, которые определены в списке `com.sun.star.beans.PropertyState`, являются возможными:

- **com.sun.star.beans.PropertyState.DIRECT_VALUE** – свойство определено непосредственно в тексте (прямое форматирование);
- **com.sun.star.beans.PropertyState.DEFAULT_VALUE** – свойство определено стилем (косвенное форматирование);
- **com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE** – свойство неясно. Этот статус возникает, например, при запросе свойства жирного шрифта абзаца, который включает и слова, отображаемые жирным и слова, отображаемые нормальным шрифтом.

Следующий пример показывает, как свойства форматирования могут быть отредактированы в OpenOffice.org. Он ищет текст частей абзаца, который отображается жирным шрифтом, с использованием прямого форматирования. Если он наталкивается на соответствующую частью абзаца, он удаляет прямое форматирование, используя метод `setPropertyToDefault` и назначает стиль символа MyBold соответствующей части абзаца.

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim TextElement As Object
Dim TextPortion As Object

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' цикл по всем абзацам
While Enum1.hasMoreElements
  TextElement = Enum1.nextElement
  If TextElement.supportsService("com.sun.star.text.Paragraph") Then
    Enum2 = TextElement.createEnumeration

    ' цикл по всем частям абзаца
    While Enum2.hasMoreElements
      TextPortion = Enum2.nextElement
      If TextPortion.CharWeight = _
        com.sun.star.awt.FontWeight.BOLD AND _
        TextPortion.getPropertyState("CharWeight") = _
        com.sun.star.beans.PropertyState.DIRECT_VALUE Then

        TextPortion.setPropertyToDefault("CharWeight")
        TextPortion.CharStyleName = "MyBold"
      End If
    Wend
  End If
Wend
```

Редактирование текстовых документов

Предыдущий раздел уже обсудил целый набор параметров для редактирования текстовых документов, сосредотачиваясь на сервисах `com.sun.star.text.TextPortion` и `com.sun.star.text.Paragraph`, которые предоставляют доступ к частям абзаца, а также к абзацам. Эти сервисы являются соответствующими для приложений, в которых содержание текста должно быть отредактировано в одном проходе цикла. Однако, их не достаточно для многих задач. OpenOffice.org предоставляет сервис `com.sun.star.text.TextCursor` для более сложных задач, включая перемещение назад в пределах документа или перемещение, основанное на предложениях из слов, а не на `TextPortions`.

TextCursor

`TextCursor` в OpenOffice.org API сопоставим с видимым курсором, используемым в документе OpenOffice.org. Он отмечает определенную точку в пределах текстового документа и может передвигаться в различных направлениях с помощью команд. Объект `TextCursor`, доступный в OOo Basic не надо путать с видимым курсором. Это две совсем разные вещи.

Примечание

Терминология отличается от используемого в VBA: по функциональным возможностям, объект `Range` из VBA может быть сравним с объектом `TextCursor` в OpenOffice.org, а не - поскольку название возможно предполагает - с объектом `Range` в OpenOffice.org.

Объект `TextCursor` в OpenOffice.org, например, предоставляет методы для перемещения по тексту и изменения текста, которые включены в объект `Range` в VBA (например, `MoveStart`, `MoveEnd`, `InsertBefore`, `InsertAfter`). Соответствующие расширения объекта `TextCursor` в OpenOffice.org описаны в следующих секциях.

Перемещение в пределах Текста

Объект `TextCursor` в OOo Basic действует независимо от видимого курсора в текстовом документе. Управляемое программой изменение положения объекта `TextCursor` не имеет вообще никакого воздействия на видимый курсор. Несколько объектов `TextCursor` могут быть открыты для одного и того же документа и использоваться в различных положениях, которые являются независимыми друг от друга.

Объект `TextCursor` создается с использованием вызова `createTextCursor`:

```
Dim Doc As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = TextDocument.Text.createTextCursor()
```

Объект `Cursor`, созданный таким образом поддерживает сервис `com.sun.star.text.TextCursor`, который в свою очередь предоставляет целый набор методов для перемещения в пределах текстовых документов. Следующий пример сначала перемещает `TextCursor` на десять символов влево, а затем на три символа вправо:

```
Cursor.goLeft(10, False)
Cursor.goRight(3, False)
```

`TextCursor` может выделить всю область. Это может быть сравнимо с выделением в тексте с использованием мыши. Параметр `False` в предыдущем вызове функции определяет,

выделяется ли область во время перемещения курсора. Например, `TextCursor` в следующем примере

```
Cursor.goLeft(10, False)
Cursor.goRight(3, True)
```

сначала перемещается на десять символов влево без выделения, и затем перемещается назад на три символа и выделяет их. Область, выделенная `TextCursor` поэтому начинается после седьмого символа в тексте и заканчивается после десятого символа.

Вот основные методы сервиса `com.sun.star.text.TextCursor` предусматривающие навигацию:

- **goLeft (Count, Expand)** – переход на `Count` символов влево;
- **goRight (Count, Expand)** – переход на `Count` символов вправо;
- **gotoStart (Expand)** – переход к началу текстового документа;
- **gotoEnd (Expand)** – переход к концу текстового документа;
- **gotoRange (TextRange, Expand)** – переход к указанному `TextRange`-объекту;
- **gotoStartOfWord (Expand)** – переход к началу текущего слова;
- **gotoEndOfWord (Expand)** – переход к концу текущего слова;
- **gotoNextWord (Expand)** – переход к началу следующего слова;
- **gotoPreviousWord (Expand)** – переход к началу предыдущего слова;
- **isStartOfWord ()** – возвращает `True`, если `TextCursor` в начале слова;
- **isEndOfWord ()** – возвращает `True`, если `TextCursor` в конце слова;
- **gotoStartOfSentence (Expand)** – переход к началу текущего предложения;
- **gotoEndOfSentence (Expand)** – переход к концу текущего предложения;
- **gotoNextSentence (Expand)** – переход к началу следующего предложения;
- **gotoPreviousSentence (Expand)** – переход к началу предыдущего предложения;
- **isStartOfSentence ()** – возвращает `True`, если `TextCursor` в начале предложения;
- **isEndOfSentence ()** – возвращает `True`, если `TextCursor` в конце предложения;
- **gotoStartOfParagraph (Expand)** – переход к началу текущего абзаца;
- **gotoEndOfParagraph (Expand)** – переход к концу текущего абзаца;
- **gotoNextParagraph (Expand)** – переход к началу следующего абзаца;
- **gotoPreviousParagraph (Expand)** – переход к началу предыдущего абзаца;
- **isStartOfParagraph ()** – возвращает `True`, если `TextCursor` в начале абзаца;
- **isEndOfParagraph ()** – возвращает `True`, если `TextCursor` в конце абзаца.

Текст разделяется на предложения на основе символов предложения. Точка, например, интерпретируются как символ, указывающий конец предложения.

Параметр `Expand` – логическое значение, которое определяет должна ли область, пройденная во время перемещения быть выделена. Все навигационные методы кроме того возвращают параметр, который определяет, было ли перемещение успешным или было ли

действие закончено из-за отсутствия текста.

В следующем списке нескольких методов для редактирования выделенной области с использованием `TextCursor`, которые также поддерживаются сервисом `com.sun.star.text.TextCursor`:

- **collapseToStart ()** – сбрасывает выделение и помещает `TextCursor` в начало ранее выделенной области;
- **collapseToEnd ()** – сбрасывает выделение и помещает `TextCursor` в конец ранее выделенной области;
- **isCollapsed ()** – возвращает `True`, если `TextCursor` в настоящее время не осуществляет никакого выделения.

Форматирование текста с TextCursor

Сервис `com.sun.star.text.TextCursor` поддерживает весе свойства символа и абзаца, которые были представлены в начале этой главы.

Следующий пример показывает, как они могут использоваться совместно с `TextCursor`. Он проходит через весь документ и форматирует первое слово каждого предложения жирным шрифтом.

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do
  Cursor.gotoEndOfWord(True)
  Cursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
  Proceed = Cursor.gotoNextSentence(False)
  Cursor.gotoNextWord(False)
Loop While Proceed
```

Пример сначала создает объект документ для текста, который был только что открыт. После этого он проходит через весь текст, предложение за предложением, выделяет каждое первое слово и форматирует его жирным.

Восстановление и изменение текстового содержания

Если `TextCursor` содержит выделенную область, то этот текст доступен через свойство `String` объекта `TextCursor`. Следующий пример используя свойство `String` отображает первые слова предложения в окне сообщения:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do
  Cursor.gotoEndOfWord(True)
  MsgBox Cursor.String
  Proceed = Cursor.gotoNextSentence(False)
  Cursor.gotoNextWord(False)
Loop While Proceed
```

Первое слово каждого предложения может быть изменено, таким же образом с использованием свойства `String`:

```
Dim Doc As Object
```

```
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do
  Cursor.gotoEndOfWord(True)
  Cursor.String = "Ups"
  Proceed = Cursor.gotoNextSentence(False)
  Cursor.gotoNextWord(False)
Loop While Proceed
```

Если `TextCursor` содержит выделенную область, присваивание свойству `String` заменяет выделение новым текстом. Если нет выделенной области, текст вставляется в текущее положение `TextCursor`.

Вставка управляющих кодов

В некоторых ситуациях, это не фактический текст документа, а скорее его структурные элементы, которые нуждаются в изменении. OpenOffice.org предоставляет для этой цели управляющие коды. Они вставляются в текст и влияют на его структуру. Управляющие коды определены в группе констант `com.sun.star.text.ControlCharacter`. Следующие управляющие коды доступны в OpenOffice.org:

- **PARAGRAPH_BREAK** – разрыв абзаца;
- **LINE_BREAK** – разрыв строки в пределах абзаца;
- **SOFT_HYPHEN** – возможная точка для вставки переноса;
- **HARD_HYPHEN** – обязательная точка для вставки переноса;
- **HARD_SPACE** – неразрывный пробел, который не растягивается или сжимается в выровненном тексте.

Для вставки управляющих кодов, Вы нуждаетесь не только в курсоре но также и в связанных объектах текстового документа. Следующий пример вставляет абзац после 20-ого символа текста:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Cursor.goRight(20, False)
Doc.Text.insertControlCharacter(Cursor, _
com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
```

Параметр `False` в вызове метода `insertControlCharacter` гарантирует, что область, выделенная в настоящее время `TextCursor` останется после операции вставки. Если здесь передают параметр `True`, то `insertControlCharacter` заменяет текущий текст.

Поиск частей текста

Во многих случаях имеет место ситуация, в которой текст должен быть найден по заданному элементу, и соответствующее место должно быть отредактировано. Все документы OpenOffice.org предоставляют с этой целью особый интерфейс, и этот интерфейс всегда функционирует в соответствии одними и теми же принципами: перед процессом поиска, то, что обычно упоминается как `SearchDescriptor`, должно быть

сначала создано. Он определяет то, что OpenOffice.org ищет в документе. `SearchDescriptor` – объект, который поддерживает сервис `com.sun.star.util.SearchDescriptor` и может быть создан посредством метода документа `createSearchDescriptor`:

```
Dim SearchDesc As Object
SearchDesc = Doc.createSearchDescriptor
```

Как только `SearchDescriptor` создан, он получает текст, который должен быть найден:

```
SearchDesc.searchString="any text"
```

По функциональным возможностям, `SearchDescriptor` является лучшим по сравнению с диалогом поиска OpenOffice.org. Подобным способу через окно поиска, параметры настройки, необходимые для поиска, могут быть установлены в объекте `SearchDescriptor`.

Свойства, предоставляемые сервисом `com.sun.star.util.SearchDescriptor`:

- **SearchBackwards (Boolean)** – осуществляет поиск текста в обратном порядке, а не вперед;
- **SearchCaseSensitive (Boolean)** – учитывает регистр символов во время поиска;
- **SearchRegularExpression (Boolean)** – рассматривает выражение поиска как регулярное выражение;
- **SearchStyles (Boolean)** – осуществляет поиск текста для указанного стиля абзаца;
- **SearchWords (Boolean)** – ищет только полные слова.

Функция OpenOffice.org `SearchSimilarity` (или “нечеткое соответствие”) также доступна в OOo Basic. С этой функцией, OpenOffice.org ищет выражение, которое может быть подобно, но не точно соответствовать выражению поиска. Число добавленных, удаленных и измененных символов для этих выражений определяется по отдельности. Вот соответствующие свойства сервиса `com.sun.star.util.SearchDescriptor`:

- **SearchSimilarity (Boolean)** – выполняет поиск подобия;
- **SearchSimilarityAdd (Short)** – число символов, которые могут быть добавлены при поиске подобия;
- **SearchSimilarityExchange (Short)** – число символов, которые могут отличаться при поиске подобия;
- **SearchSimilarityRemove (Short)** – число символов, которые могут отсутствовать при поиске подобия;
- **SearchSimilarityRelax (Boolean)** – принимать во внимание все правила отклонения одновременно для выражения поиска.

Как только `SearchDescriptor` будет готов согласно запросу, он может быть применен к текстовому документу. Документы OpenOffice.org предоставляют с этой целью методы `findFirst` и `findNext`:

```
Found = Doc.findFirst(SearchDesc)

Do Until IsNull(Found)
  ' Suchergebnis bearbeiten
  Found = Doc.findNext(Found.End, SearchDesc)
Loop
```

Пример находит все совпадения в цикле и возвращает объект `TextRange`, который обращается к найденным местам текста.

Пример: Поиск подобного

Этот пример показывает, как в тексте может быть найдено слово “товарооборот” и результаты, отформатированы жирным шрифтом. Поиск подобия используется для того, чтобы не только слово “товарооборот”, но также были найдены и множественная форма “товарообороты” и другие формы, типа “товарооборота”. Ищущиеся выражения отличаются до двух букв от выражения поиска:

```
Dim SearchDesc As Object
Dim Doc As Object

Doc = StarDesktop.CurrentComponent

SearchDesc = Doc.createSearchDescriptor
SearchDesc.SearchString="товарооборот"
SearchDesc.SearchSimilarity = True
SearchDesc.SearchSimilarityAdd = 2
SearchDesc.SearchSimilarityExchange = 2
SearchDesc.SearchSimilarityRemove = 2
SearchDesc.SearchSimilarityRelax = False

Found = Doc.findFirst(SearchDesc)

Do Until IsNull(Found)
  Found.CharWeight = com.sun.star.awt.FontWeight.BOLD
  Found = Doc.findNext(Found.End, SearchDesc)
Loop
```

Примечание Основная идея поиска и замены в OpenOffice.org, совместима с используемой в VBA. Оба интерфейса предоставляют Вам объект, через который могут быть определены свойства для поиска и замены. Этот объект применяется к указанной области текста для выполнения действия. Примите во внимание, что ответственный вспомогательный объект в VBA может быть достигнут через свойство `Find` объекта `Range`, в OOo Basic, он создается вызовом `createSearchDescriptor` или `createReplaceDescriptor` объекта документ. Даже доступные свойства и методы поиска отличаются.

Как и в старом API от StarOffice, операция поиска и замена текста в новом API также выполняется с использованием объекта документ. Примите во внимание, что ранее был объект по имени `SearchSettings` специально для определения параметров поиска, новый объект поиска теперь выполняется с использованием объектов `SearchDescriptor` или `ReplaceDescriptor` для автоматической замены текста. Эти объекты содержат не только параметры, но также и текущий ищущийся текст и, в случае необходимости, связанный заменяющий текст. Дескрипторные объекты создаются с использованием объекта документ, заполняются в соответствии с установленным запросом, и затем передаются назад объекту документ как параметр для методов поиска.

Замена частей текста

Так же как и функция поиска, функция замены OpenOffice.org также доступна в OOo Basic. Эти две функции идентичны в управлении. Специальный объект, который делает запись параметров для процесса, также сначала необходим для процесса замены. Он называется `ReplaceDescriptor` и поддерживается сервисом `com.sun.star.util.ReplaceDescriptor`. Все свойства `SearchDescriptor`, описанного в предыдущем разделе также поддерживаются `ReplaceDescriptor`. Например, во время процесса замены, чувствительность к регистру символов может также быть включена и выключена, и может быть выполнен поиск подобного.

Следующий пример демонстрирует использование ReplaceDescriptors для поиска в пределах документа OpenOffice.org.

```
Dim I As Long
Dim Doc As Object
Dim Replace As Object
Dim BritishWords(5) As String
Dim USWords(5) As String

BritishWords() = Array("colour", "neighbour", "centre", "behaviour", _
    "metre", "through")

USWords() = Array("color", "neighbor", "center", "behavior", _
    "meter", "thru")

Doc = StarDesktop.CurrentComponent
Replace = Doc.createReplaceDescriptor

For I = 0 To 5
  Replace.SearchString = BritishWords(I)
  Replace.ReplaceString = USWords(I)
  Doc.replaceAll(Replace)
Next  I
```

Выражения для поиска и замены устанавливаются с использованием свойств `SearchString` и `ReplaceString ReplaceDescriptors`. Фактический процесс замены осуществляется с использованием метода `replaceAll` объекта документ, который заменяет все найденные участки, соответствующие выражению поиска.

Пример: поиск и замена текста с использованием регулярных выражений

Функция замены OpenOffice.org особенно эффективна когда используется в сочетании с регулярными выражениями. Они обеспечивают выбор определения переменного выражения поиска с подстановочными символами и специальными символами, а не фиксированного значения.

Регулярные выражения, поддерживаемые OpenOffice.org описаны подробно в разделе онлайн справки для OpenOffice.org. Вот - несколько примеров:

- точка в пределах выражения поиска определяет любой символ, кроме разрыва строки или конца абзаца. Выражение поиска `sh.rt` поэтому может использоваться и для `shirt` и для `short`;
- символ ^ отмечает начало абзаца. Все появления имени `Петя`, которые встречаются в начале абзаца, могут, поэтому, быть найдены с использованием выражения поиска `^Петя`;
- символ $ отмечает конец абзаца. Все появления имени `Петя`, которые встречаются в конце абзаца, могут, поэтому, быть найдены с использованием выражения поиска `Петя$`;
- * указывает, что предыдущий символ может быть повторен любое количество раз. Он может быть объединен с точкой в качестве подстановочного символа для любого символа. Выражение `temper.*e`, например, может найти выражение `temperance` и `temperature`.

Следующий пример показывает, как все пустые строки в текстовом документе могут быть удалены с помощью регулярного выражения ^$:

```
Dim Doc As Object
Dim Replace As Object

Doc = StarDesktop.CurrentComponent
Replace = Doc.createReplaceDescriptor
```

```
Replace.SearchRegularExpression = True
Replace.SearchString = "^$"
Replace.ReplaceString = ""

Doc.replaceAll(Replace)
```

Текстовые документы: Больше чем только текст

Пока, эта глава имела дело только с текстовыми абзацами и их частями. Но текстовые документы могут также содержать другие объекты. Они включают таблицы, рисунки, текстовые поля и справочники. Все эти объекты могут быть размещены в любом месте в пределах текста.

Благодаря этим общим особенностям, все эти объекты в OpenOffice.org поддерживают общий базовый сервис, именуемый `com.sun.star.text.TextContent`. Он предоставляет следующие свойства:

- **AnchorType (Enum)** – определяет якорный тип объекта `TextContent` (значения по умолчанию в соответствии с перечнем `com.sun.star.text.TextContentAnchorType`);
- **AnchorTypes (sequence of Enum)** – перечень всех `AnchorTypes`, которые поддерживают специальный объект `TextContent`;
- **TextWrap (Enum)** – определяет тип обтекания текста вокруг объекта `TextContent` (значения по умолчанию в соответствии с перечнем `com.sun.star.text.WrapTextMode`).

Объекты `TextContent` также разделяют некоторые методы - в частности, для создания, вставки и удаления объектов.

- новый объект `TextContent` **создается** с использованием метода `createInstance` объекта документ;
- объект **вставляется** с использованием метода `insertTextContent` объекта `text`;
- объекты `TextContent` **удаляются** с использованием метода `removeTextContent`.

Вы найдете набор примеров, которые используют эти методы в следующих разделах.

Таблицы

Следующий пример создает таблицу с помощью метода `createInstance`, описанного ранее.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)
```

После создания, в таблице устанавливается требуемое число строк и столбцов с использованием вызова `initialize` и затем она вставляется в текстовый документ, с использованием `insertTextContent`.

Как можно заметить в примере, метод `insertTextContent` ожидает не только, объект

`Content`, который будет вставлен, но и два других параметра:

- объект `Cursor`, который определяет позицию вставки;
- Логическая переменная, которая определяет, заменяет ли объект `Content` текущее выделение курсора (значение `True`) или вставляется перед текущим выделением в тексте (`False`).

Примечание При создании и вставке таблицы в текстовом документе, объекты, подобные доступным в VBA используются в OOo Basic: объект документ и объект `TextCursor` в OOo Basic или объект `Range` в VBA соответственно. Принимая во внимание, что метод `Document.Tables.Add` берет задачу создания и настройки таблицы в VBA, она создается в OOo Basic в соответствии с предыдущим примером с использованием `createInstance`, инициализируется и вставляется в документ через `insertTextContent`.

Таблицы, вставленные в текстовый документ могут быть определены с использованием простого цикла. С этой целью используется метод `getTextTables()` объекта текстовый документ:

```
Dim Doc As Object
Dim TextTables As Object
Dim Table As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
TextTables = Doc.getTextTables()
For I = 0 to TextTables.count - 1
  Table = TextTables(I)
  ' Редактирование таблицы
Next I
```

Примечание Текстовые таблицы доступны в OpenOffice.org 2 через список `TextTables` объекта документ. Он происходит от прежнего списка таблиц, предоставляемого в объекте `Selection`. Предыдущий пример показывает, как текстовая таблица может быть создана. Варианты получения доступа к текстовым таблицам описаны в следующем разделе.

Редактирование таблиц

Таблица состоит из отдельных строк. Они в свою очередь содержат отдельные ячейки. Строго говоря, в OpenOffice.org нет столбцов таблицы. Они порождаются неявно при упорядочении строк (один под другим) рядом с друг другом. Чтобы упростить доступ к таблицам, OpenOffice.org, однако, предоставляет некоторые методы, которые используются для управления столбцами. Они полезны, если ячейки не были объединены в таблице.

Позвольте нам сначала начать непосредственно со свойств таблицы. Они определены в сервисе `com.sun.star.text.TextTable`. Вот список самых важных свойств объекта таблица:

- **BackColor (Long)** – цвет фона таблицы;
- **BottomMargin (Long)** – нижнее поле в сотых долях миллиметра;
- **LeftMargin (Long)** – левое поле в сотых долях миллиметра;

- **RightMargin (Long)** – правое поле в сотых долях миллиметра;
- **TopMargin (Long)** – верхнее поле в сотых долях миллиметра;
- **RepeatHeadline (Boolean)** – повторять заголовок таблицы на каждой странице;
- **Width (Long)** – абсолютная ширина таблицы в сотых долях миллиметра.

Строки

Таблица состоит из списка, содержащего строки. Следующий пример показывает, как строки таблицы могут быть извлечены и отформатированы.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object
Dim Rows As Object
Dim Row As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)
Rows = Table.getRows
For I = 0 To Rows.getCount() - 1
  Row = Rows.getByIndex(I)
  Row.BackColor = &HFF00FF
Next
```

Пример сначала создает список, содержащий все строки, используя запрос `Table.getRows`. Методы `getCount` и `getByIndex` позволяют в дальнейшем обработать список и принадлежат интерфейсу `com.sun.star.table.XtableRows`. Метод `getByIndex` возвращает объект строка, который поддерживает сервис `com.sun.star.text.TextTableRow`.

Вот основные методы интерфейса `com.sun.star.table.XtableRows`:

- **getByIndex(Integer)** – возвращает объект строка для указанного индекса;
- **getCount()** – возвращает количество объектов строка;
- **insertByIndex(Index, Count)** – вставляет `Count` строк в таблицу в позиции `Index`;
- **removeByIndex(Index, Count)** – удаляет `Count` строк из таблицы в позиции `Index`.

Примите во внимание, что методы `getByIndex` и `getCount` доступны во всех таблицах, методы `insertByIndex` и `removeByIndex` могут использоваться только в таблицах, которые не содержат объединенные ячейки.

Сервис `com.sun.star.text.TextTableRow` предоставляет следующие свойства:

- **BackColor (Long)** – цвет фона строки;
- **Height (Long)** – высота строки в сотых долях миллиметра;
- **IsAutoHeight (Boolean)** – высота строки таблицы динамически подстраивается под содержимое;
- **VertOrient (const)** – вертикальная ориентация текстового блока - подробно описывает вертикальную ориентацию текста в пределах таблицы (значение в соответствии с `com.sun.star.text.VertOrientation`).

Столбцы

К столбцам получают доступ таким же образом как к строкам, используя методы `getByIndex`, `getCount`, `insertByIndex`, и `removeByIndex` объекта `Column`, который может быть получен через `getColumns`. Они могут, однако, использоваться только в таблицах, которые не содержат объединенных ячеек. Ячейки не могут быть отформатированы через столбец в OOo Basic. Чтобы сделать это, должен использоваться метод форматирования отдельных ячеек таблицы.

Ячейки

Каждая ячейка документа OpenOffice.org имеет уникальное имя. Если курсор OpenOffice.org находится в ячейке, то имя этой ячейки можно заметить в строке состояния. Верхняя левая ячейка обычно имеет имя A1, а нижняя правая обычно имеет имя Xn, где X местоположение в буквах вершины колонки, а n номер последней строки. Объекты ячейка доступны через метод `getCellByName()` объекта таблица. Следующий пример показывает цикл, который перебирает все ячейки таблицы и вводит соответствующий номер строки и столбца в ячейки.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object
Dim Rows As Object
Dim RowIndex As Integer
Dim Cols As Object
Dim ColIndex As Integer
Dim CellName As String
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)

Rows = Table.getRows
Cols = Table.getColumns

For RowIndex = 1 To Rows.getCount()
  For ColIndex = 1 To Cols.getCount()
    CellName = Chr(64 + ColIndex) & RowIndex
    Cell = Table.getCellByName(CellName)
    Cell.String = "row: " & CStr(RowIndex) + ", column: " & CStr(ColIndex)
  Next
Next
```

Ячейка таблицы сопоставима со стандартным текстом. Она поддерживает интерфейс `createTextCursor` для создания связанного объекта `TextCursor`.

```
CellCursor = Cell.createTextCursor()
```

Поэтому автоматически доступны все варианты форматирования для отдельных символов и абзацев.

Следующий пример перебирает все таблицы текстового документа и применяет формат выравнивания вправо ко всем ячейкам с числовыми значениями посредством соответствующего свойства абзаца.

```
Dim Doc As Object
Dim TextTables As Object
Dim Table As Object
Dim CellNames
Dim Cell As Object
Dim CellCursor As Object
Dim I As Integer
```

```
Dim J As Integer

Doc = StarDesktop.CurrentComponent
TextTables = Doc.getTextTables()

For I = 0 to TextTables.count - 1
  Table = TextTables(I)
  CellNames = Table.getCellNames()
  For J = 0 to UBound(CellNames)
    Cell = Table.getCellByName(CellNames(J))
    If IsNumeric(Cell.String) Then
      CellCursor = Cell.createTextCursor()
      CellCursor.paraAdjust = com.sun.star.style.ParagraphAdjust.RIGHT
    End If
  Next
Next
```

Пример создает список `TextTables`, содержащий все таблицы в тексте, которые перебираются в цикле. OpenOffice.org тогда создает список связанных имен ячеек для каждой из этих таблиц. Они в свою очередь перебираются в цикле. Если ячейка содержит числовое значение, то пример изменяет форматирование соответственно. Чтобы сделать это, он сначала создает объект `TextCursor`, который устанавливает ссылку на содержимое ячейки таблицы и затем применяет свойства абзаца к ячейке таблицы.

Текстовые врезки

Текстовые врезки, рассматриваются как объекты `TextContent`, точно так же как таблицы и диаграммы. Они, по существу, состоят из обычного текста, но могут размещаться в любом месте на странице и не включаются в поток текста.

Как со всеми объектами `TextContent`, для текстовых врезок делается различие между фактическим созданием и вставкой в документ.

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Frame = Doc.createInstance("com.sun.star.text.TextFrame")
Doc.Text.insertTextContent(Cursor, Frame, False)
```

Текстовая врезка создается с использованием метода `createInstance` объекта документ. Текстовая врезка, созданная таким образом может быть вставлена в документ с использованием метода `insertTextContent` объекта `Text`. Таким образом, имя собственного сервиса `com.sun.star.text.TextFrame` должно быть определено.

Позиция вставки текстовой врезки определяется объектом `Cursor`, который также должен быть создан когда осуществляется вставка.

Примечание Текстовые врезки OpenOffice.org соответствуют фреймам, используемым в Word. Поскольку VBA использует метод Document.Frames.Add с этой целью, создание в OpenOffice.org выполняется с использованием предыдущей процедуры при помощи `TextCursor`, а также метода `createInstance` объекта документ.

Объекты Текстовая врезка предоставляют набор свойств, которыми можно влиять на положение и поведение врезки. Большинство этих свойств определено в сервисе `com.sun.star.text.BaseFrameProperties`, который также поддерживается каждым сервисом `TextFrame`. Основные свойства:

- **BackColor (Long)** – цвет фона текстовой врезки;
- **BottomMargin (Long)** – нижнее поле в сотых долях миллиметра;
- **LeftMargin (Long)** – левое поле в сотых долях миллиметра;
- **RightMargin (Long)** – правое поле в сотых долях миллиметра;
- **TopMargin (Long)** – верхнее поле в сотых долях миллиметра;
- **Height (Long)** – высота текстовой врезки в сотых долях миллиметра;
- **Width (Long)** – ширина текстовой врезки в сотых долях миллиметра;
- **HoriOrient (const)** – горизонтальная ориентация текстовой врезки (в соответствии с `com.sun.star.text.HoriOrientation`);
- **VertOrient (const)** – вертикальная ориентация текстовой врезки (в соответствии с `com.sun.star.text.VertOrientation`).

Следующий пример создает текстовую врезку, используя свойства, описанные ранее:

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Cursor.gotoNextWord(False)
Frame = Doc.createInstance("com.sun.star.text.TextFrame")

Frame.Width = 3000
Frame.Height = 1000
Frame.AnchorType = com.sun.star.text.TextContentAnchorType.AS_CHARACTER
Frame.TopMargin = 0
Frame.BottomMargin = 0
Frame.LeftMargin = 0
Frame.RightMargin = 0
Frame.BorderDistance = 0
Frame.HoriOrient = com.sun.star.text.HoriOrientation.NONE
Frame.VertOrient = com.sun.star.text.VertOrientation.LINE_TOP

Doc.Text.insertTextContent(Cursor, Frame, False)
```

Пример создает `TextCursor` как ориентир для вставки текстовой врезки. Она помещается между первым и вторым словом текста. Текстовая врезка создается с использованием `Doc.createInstance`. Свойства объекта текстовая врезка устанавливаются в требуемые начальные значения.

Здесь должно быть отмечено взаимодействие между свойствами `AnchorType` (от сервиса `TextContent`) и `VertOrient` (от сервиса `BaseFrameProperties`). `AnchorType` получает значение `AS_CHARACTER`. Текстовая врезка, поэтому, вставляется непосредственно в поток текста и ведет себя как символ. Она может, например, быть перемещена в следующую строку, если происходит разрыв строки. Значение `LINE_TOP` свойства `VertOrient` гарантирует, что верхний край текстовой врезки находится на той же самой высоте что и верхний край символа.

Как только инициализация выполнена, текстовая врезка, наконец, вставляется в текстовый документ, используя вызов `insertTextContent`.

Для редактирования содержания текстовой врезки, пользователь использует `TextCursor`, который был уже упомянут много раз и также доступен для текстовых врезок.

```
Dim Doc As Object
Dim TextTables As Object
```

```
Dim Cursor As Object
Dim Frame As Object
Dim FrameCursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Frame = Doc.createInstance("com.sun.star.text.TextFrame")

Frame.Width = 3000
Frame.Height = 1000

Doc.Text.insertTextContent(Cursor, Frame, False)

FrameCursor = Frame.createTextCursor()
FrameCursor.charWeight = com.sun.star.awt.FontWeight.BOLD
FrameCursor.paraAdjust = com.sun.star.style.ParagraphAdjust.CENTER
FrameCursor.String = "Это - маленький Тест!"
```

Пример создает текстовую врезку, вставляет ее в текущий документ и открывает `TextCursor` для текстовой врезки. Этот курсор используется для установки шрифта врезки жирным и задания выравнивания абзаца по центру. Текстовой врезке, наконец, присваивается строка “Это - маленький тест!”.

Текстовые поля

Текстовые поля – объекты `TextContent`, потому что они предоставляют дополнительную логику, выходящую за пределы чистого текста. Текстовые поля могут быть вставлены в текстовый документ с использованием тех же самых методов, которые используются для других объектов `TextContent`:

```
Dim Doc As Object
Dim DateTimeField As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

DateTimeField = Doc.createInstance("com.sun.star.text.TextField.DateTime")
DateTimeField.IsFixed = False
DateTimeField.IsDate = True

Doc.Text.insertTextContent(Cursor, DateTimeField, False)
```

Пример вставляет текстовое поле с текущей датой в начале текущего текстового документа. Значение `True` свойства `IsDate` приводит к тому, что отображается только дата без времени. Значение `False` для `IsFixed` гарантирует, что дата будет автоматически обновляться при открытии документа.

Примечание В то время как тип поле в VBA определяется параметром метода Document.Fields.Add, имя сервиса, который является ответственным за тип поля в запросе определяется в OOo Basic.

В прошлом к областям текста получали доступ, используя целый набор методов, которые StarOffice делал доступным в старом объекте `Selection` (например `InsertField`, `DeleteUserField`, `SetCurField`).

В OpenOffice.org 2, полями управляют, используя объектно-ориентированную концепцию. Для создания текстового поля, текстовое поле требуемого типа должно быть сначала создано и проинициализировано с использованием требуемых свойств. Текстовое поле после этого вставляется в документ с использованием метода `insertTextContent`. Соответствующий исходный текст можно видеть в предыдущем примере. Самые важные

типы полей и их свойства описаны в следующих разделах.

В дополнение к вставке текстовых полей, поиск поля в документе может также быть важной задачей. Следующий пример показывает, как все текстовые поля текстового документа могут быть перебраны в цикле и проверены на соответствие их типа.

```
Dim Doc As Object
Dim TextFieldEnum As Object
Dim TextField As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
TextFieldEnum = Doc.getTextFields.createEnumeration

While TextFieldEnum.hasMoreElements()
  TextField = TextFieldEnum.nextElement()

  If TextField.supportsService("com.sun.star.text.TextField.DateTime") Then
    MsgBox "Дата/время"
  ElseIf TextField.supportsService("com.sun.star.text.TextField.Annotation") Then
    MsgBox "Примечание"
  Else
    MsgBox "неизвестное"
  End If
Wend
```

Отправная точка для поиска текстового поля – имеющийся список `TextFields` объекта документ. Пример создает объект `Enumeration` на основе этого списка, через который все текстовые поля могут быть опрошены, в свою очередь, в цикле. Найденные текстовые поля проверяются на поддержку сервиса с использованием метода `supportsService`. Если поле оказывается полем даты/времени или примечания, то соответствующий тип поля отображается в окне сообщения. Если, с другой стороны, пример сталкивается с другим полем, то он отображает сообщение “неизвестное”.

Ниже, Вы найдете список самых важных текстовых полей и их связанных свойств. Полный список всех текстовых полей приводится в Справочном руководстве по API в модуле `com.sun.star.text.TextField`. (при перечислении имен сервисов текстовых полей, заглавные и строчные символы должны использоваться в OOo Basic, как в предыдущем примере.)

Число страниц, слов и символов

Текстовые поля

- `com.sun.star.text.TextField.PageCount`;
- `com.sun.star.text.TextField.WordCount`;
- `com.sun.star.text.TextField.CharacterCount`.

возвращают число страниц, слов, или символов текста. Они поддерживают следующие свойства:

- **NumberingType (const)** – числовой формат (рекомендации в соответствии с константами из `com.sun.star.style.NumberingType`).

Текущая страница

Номер текущей страницы может быть вставлен в документ с использованием текстового поля `com.sun.star.text.TextField.PageNumber`. Могут быть определены следующие свойства:

- **NumberingType (const)** – числовой формат (рекомендации в соответствии с

константами из `com.sun.star.style.NumberingType`);

- **Offset (short)** – смещение, добавляемое к номеру страницы (указание отрицательного значения, также возможно).

Следующий пример показывает, как номер страницы может быть вставлен в нижний колонтитул документа.

```
Dim Doc As Object
Dim DateTimeField As Object
Dim PageStyles As Object
Dim StdPage As Object
Dim FooterCursor As Object
Dim PageNumber As Object

Doc = StarDesktop.CurrentComponent

PageNumber = Doc.createInstance("com.sun.star.text.TextField.PageNumber")
PageNumber.NumberingType = com.sun.star.style.NumberingType.ARABIC

PageStyles = Doc.StyleFamilies.getByName("PageStyles")

StdPage = PageStyles("Обычный")
StdPage.FooterIsOn = True

FooterCursor = StdPage.FooterTextLeft.Text.createTextCursor()
StdPage.FooterTextLeft.Text.insertTextContent(FooterCursor, PageNumber, False)
```

Пример сначала создает текстовое поле, которое поддерживает сервис `com.sun.star.text.TextField.PageNumber`. Так как верхний и нижний колонтитулы определены как часть стиля страницы OpenOffice.org, он первоначально устанавливается с использованием списка всех стилей `PageStyles`.

Чтобы гарантировать, что нижний колонтитул является видимым, свойство `FooterIsOn` устанавливается в `True`. Текстовое поле, тогда, вставляется в документ, используя связанный объект текст левого нижнего колонтитула.

Примечания

Поле примечания (`com.sun.star.text.TextField.Annotation`), может быть замечено посредством маленького желтого символа в тексте. Нажатие на этот символ открывает текстовое поле, в котором может быть записан комментарий к текущей точке в тексте. Поле примечания имеет следующие свойства.

- **Author (String)** – имя автора;
- **Content (String)** – текст комментария;
- **Date (Date)** - дата, когда написано примечание.

Дата / Время

Поле даты / времени (`com.sun.star.text.TextField.DateTime`), отображает текущую дату или текущее время. Оно поддерживает следующие свойства:

- **IsFixed (Boolean)** – если `True`, детали вставленного времени остаются неизменными, если `False`, они обновляются каждый раз, при открытии документа;
- **IsDate (Boolean)** – если `True`, поле показывает текущую дату, в противном случае текущее время;
- **DateTimeValue (struct)** – текущее содержимое поля (структура `com.sun.star.util.DateTime`);

- **NumberFormat (const)** – формат, в котором отображаются время или дата.

Название / Номер главы

Название текущей главы доступно через текстовое поле типа `com.sun.star.text.TextField.Chapter`. Форма может быть определена с использованием двух свойств.

- **ChapterFormat (const)** – определяет, отображается ли название главы или номер главы (в соответствии с `com.sun.star.text.ChapterFormat`);
- **Level (Integer)** – определяет уровень главы, номер и/или названия которой должно быть показано. Значение 0 означает самый высокий доступный уровень.

Закладки

Закладки (сервис `com.sun.star.text.Bookmark`) – объекты `TextContent`. Закладки создаются и вставляются с использованием концепции, уже описанной ранее:

```
Dim Doc As Object
Dim Bookmark As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Bookmark = Doc.createInstance("com.sun.star.text.Bookmark")
Bookmark.Name = "My bookmarks"

Doc.Text.insertTextContent(Cursor, Bookmark, True)
```

Пример создает `Cursor`, который отмечает положение вставки закладки и затем фактический объект закладка (`Bookmark`). Закладке присваивается имя и она вставляется в документ через `insertTextContent` в позиции курсора.

К закладкам текста получают доступ через вызов списка `Bookmarks`. К закладкам можно получить доступ по их номеру или по их имени.

Следующий пример показывает, как закладка может быть найдена в пределах текста, и текст вставлен в ее позиции.

```
Dim Doc As Object
Dim Bookmark As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent

Bookmark = Doc.Bookmarks.getByName("My bookmarks")

Cursor = Doc.Text.createTextCursorByRange(Bookmark.Anchor)
Cursor.String = "Здесь - закладка"
```

В этом примере, метод `getByName` используется для поиска требуемой закладки по ее имени. Вызов `createTextCursorByRange` создает `Cursor`, который помещается в позицию привязки закладки. После чего курсор вставляет текст, требуемый в этом месте.

Глава 7. Электронные таблицы

OOo Basic предоставляет исчерпывающий интерфейс для программно-управляемого создания и редактирования электронных таблиц. Эта глава описывает, как управлять соответствующими сервисами, методами и свойствами документов электронных таблиц.

Первый раздел обращается к базовой структуре документов электронных таблиц и показывает Вам как получить доступ и редактировать содержимое отдельных ячеек.

Второй раздел концентрируется на эффективном редактировании электронной таблицы, сосредотачиваясь на областях ячеек и параметрах поиска и замены содержимого ячеек.

Примечание Объект `Range` позволяет Вам обратиться к любой области таблицы и был расширен в новом API.

Структура документов на основе таблиц (Электронных таблиц)

Объект документ электронной таблицы основан на сервисе `com.sun.star.sheet.SpreadsheetDocument`. Каждый из этих документов может содержать несколько электронных таблиц. В этом руководстве, *документ на основе таблицы* или *документ электронная таблица* - весь документ, тогда как *электронная таблица* (или *лист* для краткости) - лист (таблица) в документе.

Примечание в VBA и OOo Basic используется различная терминология для электронных таблиц и их содержания. Примите во внимание, что объект документ в VBA называют *Workbook*, а его отдельные страницы *Worksheets*, в OOo Basic их называют *SpreadsheetDocument* и *Sheet*.

Электронные таблицы

Вы можете получить доступ к отдельным листам документа электронной таблицы через список `Sheets`.

Следующие примеры показывают Вам, как получить доступ к листу или по его номеру или по его имени.

Пример 1: доступ по номеру (нумерация начинается с 0)

```
Dim Doc As Object
Dim Sheet As Object
```

```
Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
```

Пример 2: доступ по имени

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Лист1")
```

В первом примере, к листу получают доступ по его номеру (счет начинается с 0). Во втором примере, к листу получают доступ по его имени и с использованием метода `getByName`.

Объект `Sheet`, который получен методом `getByName`, поддерживает сервис `com.sun.star.sheet.Spreadsheet`. В дополнение к предоставлению нескольких интерфейсов для редактирования содержимого, этот сервис предоставляет следующие свойства:

- **IsVisible (Boolean)** – видимость электронной таблицы;
- **PageStyle (String)** – имя стиля страницы для электронной таблицы.

Создание, удаление и переименование листов

Список `Sheets` для документа `spreadsheet` также используется для создания, удаления и переименования отдельных листов. Следующий пример использует метод `hasByName`, чтобы проверить, существует ли лист по имени *MySheet*. Если это так, метод определяет соответствующую ссылку на объект, используя метод `getByName`, и затем сохраняет ссылку в переменной `Sheet`. Если соответствующий лист не существует, он создается вызовом `createInstance` и вставляется в документ электронной таблицы методом `insertByName`.

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

If Doc.Sheets.hasByName("MySheet") Then
  Sheet = Doc.Sheets.getByName("MySheet")
Else
  Sheet = Doc.createInstance("com.sun.star.sheet.Spreadsheet")
  Doc.Sheets.insertByName("MySheet", Sheet)
End If
```

Методы `getByName` и `insertByName` интерфейса `com.sun.star.container.XnameContainer` описаны в Главе 4.

Строки и столбцы

Каждый лист содержит список своих строк и столбцов. Они доступны через свойства `Rows` и `Columns` объекта электронная таблица и поддерживают сервисы `com.sun.star.table.TableColumns` и/или `com.sun.star.table.TableRows`.

Следующий пример создает два объекта, которые ссылаются на первую строку и первый столбец листа, и сохраняет ссылки в объектных переменных `FirstCol` и `FirstRow`.

```
Dim Doc As Object
Dim Sheet As Object
Dim FirstRow As Object
Dim FirstCol As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
```

```
FirstCol = Sheet.Columns(0)
FirstRow = Sheet.Rows(0)
```

Объекты столбец поддерживают сервис `com.sun.star.table.TableColumn`, который имеет следующие свойства:

- **Width (long)** – ширина столбца в сотых долях миллиметра;
- **OptimalWidth (Boolean)** – устанавливает оптимальную ширину столбца;
- **IsVisible (Boolean)** – отображение столбца;
- **IsStartOfNewPage (Boolean)** – при печати, вставить разрыв страницы перед столбцом.

Ширина столбца оптимизируется только когда свойство `OptimalWidth` устанавливается в `True`. Если ширина отдельной ячейки изменяется, ширина столбца, который содержит ячейку, не изменяется. В терминах функциональных возможностей, `OptimalWidth` – скорее метод чем свойство.

Объекты строка основаны на сервисе `com.sun.star.table.RowColumn`, который имеет следующие свойства:

- **Height (long)** – высота строки в сотых долях миллиметра;
- **OptimalHeight (Boolean) -** устанавливает оптимальную высоту строки;
- **IsVisible (Boolean)** – отображение строки;
- **IsStartOfNewPage (Boolean)** – при печати, вставить разрыв страницы перед строкой.

Если свойство `OptimalHeight` строки устанавливается в True, высота строки изменяется автоматически, когда высота ячейки в строке изменяется. Автоматическая оптимизация продолжается, пока строке не установлена абсолютная высота через свойство `Height`.

Следующий пример активизирует автоматическую оптимизацию высоты для первых пяти строк на листе и делает второй столбец невидимым.

```
Dim Doc As Object
Dim Sheet As Object
Dim Row As Object
Dim Col As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

For I = 0 To 4
  Row = Sheet.Rows(I)
  Row.OptimalHeight = True
Next I

Col = Sheet.Columns(1)
Col.IsVisible = False
```

Примечание К спискам `Rows` и `Columns` можно получить доступ через индекс в OOo Basic. В отличие от VBA, первая колонка имеет индекс 0 а не 1.

Вставка и удаление строк и столбцов

Объекты `Rows` и `Columns` листа могут получать доступ к существующим строкам и столбцам, а так же вставлять и удалять их.

```
Dim Doc As Object
Dim Sheet As Object
Dim NewColumn As Object
Doc = StarDesktop.CurrentComponent

Sheet = Doc.Sheets(0)

Sheet.Columns.insertByIndex(3, 1)
Sheet.Columns.removeByIndex(5, 1)
```

Этот пример использует метод `insertByIndex`, для вставки нового столбца в четвертой позиции столбца на листе (индекс 3 – нумерация начинается с 0). Второй параметр определяет число столбцов, которые будут вставлены (в этом примере: один).

Метод `removeByIndex` удаляет шестой столбец (индекс 5). Снова, второй параметр определяет число столбцов, которые Вы хотите удалить.

Методы для вставки и удаления строк используют функции объекта `Rows` таким же образом как методы, показанные для редактирования столбцов, используют объект `Columns`.

Ячейки

Электронная таблица состоит из двумерного списка, содержащего ячейки. Каждая ячейка определяется ее X и Y-положением относительно верхней левой ячейки, которая имеет положение (0, 0).

Следующий пример создает объект, который ссылается на верхнюю левую ячейку и вставляет текст в ячейку:

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(0, 0)
Cell.String = "Тест"
```

В дополнение к числовым координатам, каждая ячейка в листе имеет имя, например, верхнюю левую ячейку (0, 0) из электронной таблицы называют A1. Буква A для столбца и номер 1 для строки. Важно не перепутать *имя* и *положение* ячейки, потому что счет строк в имени начинается с 1, но счет в положении начинается с 0.

В OpenOffice.org, ячейка таблицы может быть пустой или содержать текст, числа, или формулы. Тип ячейки не определяется содержанием, которое сохранено в ячейке, а скорее свойствами объекта, который использовался для ее ввода. Числа могут быть вставлены и получены с использованием свойства `Value`, текст – свойства `String`, а формулы – свойства `Formula`.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(0, 0)
Cell.Value = 100

Cell = Sheet.getCellByPosition(0, 1)
Cell.String = "Тест"

Cell = Sheet.getCellByPosition(0, 2)
Cell.Formula = "=A1"
```

Пример вставляет одно число, один текст, и одну формулу в поле с A1 до A3.

Примечание Свойства `Value`, `String` и `Formula` заменяют метод `PutCell` для установки значения ячейки таблицы.

OpenOffice.org рассматривает содержимое ячейки, которое введено с использованием свойства `String` как текст, даже если содержимое – число. Число в ячейке выравнивается по левому краю вместо выравнивания по правому краю. Вы должны также отметить различие между текстом и числами, когда Вы используете формулы:

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(0, 0)
Cell.Value = 100

Cell = Sheet.getCellByPosition(0, 1)
Cell.String = 1000

Cell = Sheet.getCellByPosition(0, 2)
Cell.Formula = "=A1+A2"

MsgBox Cell.Value
```

Хотя ячейка, A1 содержит значение 100, а ячейка A2, содержит значение 1000, формула A1+A2 возвращает значение 100. Это потому, что содержимое ячейки A2 было введено как строка, а не как число.

Чтобы проверять, содержит содержимое ячейки число или строку, используйте свойство `Type`:

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(1,1)
Cell.Value = 1000

Select Case Cell.Type
  Case com.sun.star.table.CellContentType.EMPTY
    MsgBox "Содержимое: Пусто"

  Case com.sun.star.table.CellContentType.VALUE
    MsgBox "Содержимое: Число"

  Case com.sun.star.table.CellContentType.TEXT
    MsgBox "Содержимое: Текст"

  Case com.sun.star.table.CellContentType.FORMULA
    MsgBox "Содержимое: Формула"
End Select
```

Свойство `Cell.Type` возвращает значение из списка `com.sun.star.table.CellContentType`, которое идентифицирует тип содержимого ячейки. Возможные значения:

- **EMPTY** – пусто, нет значения;
- **VALUE** – число;
- **TEXT** – строка;

- **FORMULA** – формула.

Вставка, удаление, копирование и перемещение ячеек

В дополнение к прямому изменению содержимого ячейки, OpenOffice.org Calc также предоставляет интерфейс, который позволяет Вам вставлять, удалять, копировать, или объединять ячейки. Интерфейс (`com.sun.star.sheet.XRangeMovement`), доступен через объект электронная таблица и обеспечивает четыре метода для изменения содержимого ячейки.

Метод `insertCell` используется для вставки ячейки в лист.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

Sheet.insertCells(CellRangeAddress, com.sun.star.sheet.CellInsertMode.DOWN)
```

Этот пример вставляет область ячеек, размером в две строки и два столбца после второго столбца и строки (каждый имеет номер 1) первого листа (номер 0) в электронной таблице. Любые существующие значения в указанной области ячеек перемещаются вниз, под вставляемую область.

Для определения области ячеек, которые Вы хотите вставить, используйте структуру `com.sun.star.table.CellRangeAddress`. Следующие значения включены в эту структуру:

- **Sheet (short)** – номер листа (нумерация начинается с 0);
- **StartColumn (long)** – первый столбец в области ячеек (нумерация начинается с 0);
- **StartRow (long)** – первая строка в области ячеек (нумерация начинается с 0);
- **EndColumn (long)** – последний столбец в области ячеек (нумерация начинается с 0);
- **EndRow (long)** – последняя строка в области ячеек (нумерация начинается с 0).

Заполненную структуру `CellRangeAddress` нужно передать как первый параметр методу `insertCells`. Второй параметр `insertCells` содержит значение из списка `com.sun.star.sheet.CellInsertMode` и определяет то, что должно быть сделано со значениями, которые расположены перед точкой вставки. Список CellInsertMode предоставляет следующие значения:

- **NONE** – текущие значения не остаются в их существующем положении;
- **DOWN** – ячейки в и под позицией вставки перемещаются вниз;
- **RIGHT** – ячейки в и справа от позиции вставки перемещаются вправо;
- **ROWS** – строки после позиции вставки перемещаются вниз;
- **COLUMNS** – столбцы после позиции вставки перемещаются вправо.

Метод `removeRange` является дополнением методу `insertCells`. Этот метод удаляет область, которая определена в структуре `CellRangeAddress` из листа.

```
Dim Doc As Object
```

```
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

Sheet.removeRange(CellRangeAddress, com.sun.star.sheet.CellDeleteMode.UP)
```

Этот пример удаляет область ячеек `B2:C3` из листа и затем перемещает расположенные под ней ячейки вверх на две строки. Тип удаления определяется одним из следующих значений из списка `com.sun.star.sheet.CellDeleteMode`:

- **NONE** – текущие значения не остаются в их существующем положении;
- **UP** – ячейки, расположенные под удаляемой областью перемещаются вверх;
- **LEFT** – ячейки, расположенные справа от удаляемой области, перемещаются влево;
- **ROWS** – строки, расположенные ниже удаляемой области, перемещаются вверх;
- **COLUMNS** – столбцы, расположенные справа от удаляемой области, перемещаются влево.

Интерфейс `XRangeMovement` предоставляет два дополнительных метода для перемещения (`moveRange`) или копирования (`copyRange`) областей ячеек. Следующий пример перемещает область `B2:C3` так, чтобы она начиналась в позиции `A6`:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress
Dim CellAddress As New com.sun.star.table.CellAddress

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2
CellAddress.Sheet = 0
CellAddress.Column = 0
CellAddress.Row = 5

Sheet.moveRange(CellAddress, CellRangeAddress)
```

В дополнение к структуре `CellRangeAdress`, метод `moveRange` ожидает структуру `com.sun.star.table.CellAddress` для определения начала целевой области перемещения. Метод `CellAddress` предусматривает следующие значения:

- **Sheet (short)** – номер электронной таблицы (нумерация начинается с 0);
- **Column (long)** – номер адресного столбца (нумерация начинается с 0);
- **Row (long)** – номер адресной строки (нумерация начинается с 0).

Содержимое ячеек в целевой области всегда перезаписывается методом `moveRange`. В отличие от метода `InsertCells`, параметр для выполнения автоматического перемещения не предусмотрен в методе `removeRange`.

Метод `copyRange` функционирует таким же образом, как метод `moveRange`, за исключением того, что `copyRange` вставляет копию области ячеек вместо ее перемещения.

Примечание По функциональным возможностям, методы OOo Basic `insertCell`, `removeRange` и `copyRange` сопоставимы с методами VBA `Range.Insert`, `Range.Delete` и `Range.Copy`. Примите во внимание, что в VBA, методы применяются к соответствующему объекту `Range`, в OOo Basic, они применяются к взаимодействующему объекту `Sheet`.

Форматирование

Документ электронная таблица предоставляет свойства и методы для форматирования ячеек и страниц.

Свойства ячейки

Есть многочисленные параметры для форматирования ячейки, такие как определения типа шрифта и размера для текста. Каждая ячейка поддерживает сервисы `com.sun.star.style.CharacterProperties` и `com.sun.star.style.ParagraphProperties`, главные свойства которых описаны в Главе 6. Специальное форматирование ячеек обрабатывается сервисом `com.sun.star.table.CellProperties`. Главные свойства этого сервиса описаны в следующих разделах.

Вы можете применить все названные свойства к отдельным ячейкам и к областям ячеек.

Примечание Объект `CellProperties` в OpenOffice.org API сопоставим с объектом `Interior` в VBA, который также определяет характерные для ячейки свойства.

Фоновый цвет и тени

Сервис `com.sun.star.table.CellProperties` предоставляет следующие свойства для определения фонового цвета и тени:

- **CellBackColor (Long)** – цвет фона ячейки таблицы;
- **IsCellBackgroundTransparent (Boolean)** – устанавливает цвет фона в прозрачный;
- **ShadowFormat (struct)** – определяет тень для ячеек (структура в соответствии с `com.sun.star.table.ShadowFormat`).

Структура `com.sun.star.table.ShadowFormat` и подробное описание для теней ячейки имеют следующую структуру:

- **Location (enum)** – положение тени (значение из списка `com.sun.star.table.ShadowLocation`);
- **ShadowWidth (Short)** – размер тени в сотых долях миллиметра;
- **IsTransparent (Boolean)** – устанавливает цвет тени в прозрачный;
- **Color (Long)** – цвет тени.

Следующий пример записывает число 1000 в ячейку B2, изменяет цвет фона на красный, используя свойство `CellBackColor`, и затем создает легкую серую тень для ячейки, которая имеет размер 1 мм и помещается слева и внизу.

```
Dim Doc As Object
Dim Sheet As Object
```

```
Dim Cell As Object
Dim ShadowFormat As New com.sun.star.table.ShadowFormat

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
Cell = Sheet.getCellByPosition(1, 1)
Cell.Value = 1000

Cell.CellBackColor = RGB(255, 0, 0)

ShadowFormat.Location = com.sun.star.table.ShadowLocation.BOTTOM_RIGHT
ShadowFormat.ShadowWidth = 100
ShadowFormat.Color = RGB(160, 160, 160)

Cell.ShadowFormat = ShadowFormat
```

Выравнивание

OpenOffice.org предоставляет различные функции, которые позволяют Вам изменять выравнивание текста в ячейке таблицы.

Следующие свойства определяют горизонтальное и вертикальное выравнивание текста:

- **HoriJustify (enum)** – горизонтальное выравнивание текста (значение из `com.sun.star.table.CellHoriJustify`);
- **VertJustify (enum)** – вертикальное выравнивание текста (значение из `com.sun.star.table.CellVertJustify`);
- **Orientation (enum)** – ориентация текста (значение в соответствии с `com.sun.star.table.CellOrientation`);
- **IsTextWrapped (Boolean)** – разрешает автоматический перенос строк в пределах ячейки;
- **RotateAngle (Long)** – угол вращения текста в сотых долях градуса.

Следующий пример показывает, как Вы можете “сложить” содержимое ячейки так, чтобы отдельные символы были напечатаны один под другим в верхнем левом углу ячейки. Символы не вращаются.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(1, 1)

Cell.Value = 1000

Cell.HoriJustify = com.sun.star.table.CellHoriJustify.LEFT
Cell.VertJustify = com.sun.star.table.CellVertJustify.TOP
Cell.Orientation = com.sun.star.table.CellOrientation.STACKED
```

Форматы чисел, дат и текста

OpenOffice.org предоставляет целый набор предопределенных форматов даты и времени. Каждый из этих форматов имеет внутренний номер, который используется для назначения формата ячейке с использованием свойства NumberFormat. OpenOffice.org предоставляет методы `queryKey` и `addNew`, чтобы Вы могли получить доступ к существующим форматам чисел, а так же создавать ваши собственные форматы чисел. К методам получают доступ через следующий вызов объекта:

```
NumberFormats = Doc.NumberFormats
```

Формат определяется с используя строки формата, которая структурирована подобно используемой в функции формата OOo Basic. Однако есть одно основное отличие: тогда как функция формат ожидает английские аббревиатуру и десятичную точку или символ разделителя тысяч, определяемая региональными настройками аббревиатура должна использоваться для структуры команды формата объекта `NumberFormats`.

Следующий пример форматирует ячейку `B2` так, чтобы числа были показаны с тремя десятичными знаками и использует запятые в качестве разделителя тысяч.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object
Dim NumberFormats As Object
Dim NumberFormatString As String
Dim NumberFormatId As Long
Dim LocalSettings As New com.sun.star.lang.Locale

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
Cell = Sheet.getCellByPosition(1, 1)

Cell.Value = 23400.3523565

LocalSettings.Language = "en"
LocalSettings.Country = "us"

NumberFormats = Doc.NumberFormats
NumberFormatString = "#,##0.000"

NumberFormatId = NumberFormats.queryKey(NumberFormatString, LocalSettings, True)
If NumberFormatId = -1 Then
  NumberFormatId = NumberFormats.addNew(NumberFormatString, LocalSettings)
End If

MsgBox NumberFormatId
Cell.NumberFormat = NumberFormatId
```

Диалог Формат ячейки в OpenOffice.org Calc предоставляет краткий обзор различных вариантов форматирования для ячеек.

Свойства Страницы

Свойства страницы – параметры форматирования, которые размещают содержимое документа на странице, а также визуальные элементы, которые повторяются страницей за страницей. Они включают:

- формат бумаги;
- поля страницы;
- верхние и нижние колонтитулы.

Процедура для определения форматов страницы отличается от других форм форматирования. Принимая во внимание, что ячейка, абзац, и элементы символа могут быть отформатированы непосредственно, форматы страницы также могут быть определены и косвенно применены с использованием стилей страницы. Например, верхние и нижние колонтитулы добавляются вместе со стилем страницы.

Следующие разделы описывают основные варианты форматирования для страниц электронной таблицы. Многие из стилей, которые описаны, также доступны для текстовых документов. Свойства страницы, которые являются действительными для обоих типов документов, определяются в сервисе `com.sun.star.style.PageProperties`. Свойства страницы, которые применяются только к документам электронной таблицы, определяются в сервисе `com.sun.star.sheet.TablePageStyle`.

Примечание Свойства страницы (поля страницы, границы, и так далее) для документа Microsoft Office определяются посредством объекта `PageSetup` на уровне объектеа `Worksheet` (Excel) или Document (Word). В OpenOffice.org, эти свойства определяются с использованием стиля страницы, который в свою очередь связан с взаимодействующим документом.

Фон страницы

Сервис `com.sun.star.style.PageProperties` определяет следующие свойства для фона страниц:

- **BackColor (long)** – цвет фона;
- **BackGraphicURL (String)** – URL фонового изображения, которое Вы хотите использовать;
- **BackGraphicFilter (String)** – имя фильтра для интерпретации фонового изображения;
- **BackGraphicLocation (Enum)** – положение фонового изображения (значение согласно списка);
- **BackTransparent (Boolean)** – делает фон прозрачным.

Формат страницы

Формат страницы определяется с использованием следующих свойства сервиса `com.sun.star.style.PageProperties`:

- **IsLandscape (Boolean)** – альбомный формат;
- **Width (long)** – ширина страницы в сотых долях миллиметра;
- **Height (long)** – высота страницы в сотых долях миллиметра;
- **PrinterPaperTray (String)** – имя лотка бумаги принтера, который Вы хотите использовать.

Следующий пример устанавливает размер страницы стиля страницы “Обычный” в DIN A5 альбомный формат (высота 14.8 см, ширина 21 см):

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Обычный")

DefPage.IsLandscape = True
DefPage.Width = 21000
DefPage.Height = 14800
```

Поля страницы, граница, и тень

Сервис `com.sun.star.style.PageProperties` предоставляет следующие свойства для регулирования полей страницы, в так же границ и тени:

- **LeftMargin (long)** – ширина левого поля страницы в сотых долях миллиметра;

- **RightMargin (long)** – ширина правого поля страницы в сотых долях миллиметра;
- **TopMargin (long)** – ширина верхнего поля страницы в сотых долях миллиметра;
- **BottomMargin (long)** – ширина нижнего поля страницы в сотых долях миллиметра;
- **LeftBorder (struct)** – подробное описание для левой линии границы страницы (структура `com.sun.star.table.BorderLine`);
- **RightBorder (struct)** – подробное описание для правой линии границы страницы (структура `com.sun.star.table.BorderLine`);
- **TopBorder (struct)** – подробное описание для верхней линии границы страницы (структура `com.sun.star.table.BorderLine`);
- **BottomBorder (struct)** – подробное описание для нижней линии границы страницы (структура `com.sun.star.table.BorderLine`);
- **LeftBorderDistance (long)** – расстояние между левой границей страницы и содержанием страницы в сотых долях миллиметра;
- **RightBorderDistance (long)** – расстояние между правой границей страницы и содержанием страницы в сотых долях миллиметра;
- **TopBorderDistance (long)** – расстояние между верхней границей страницы и содержанием страницы в сотых долях миллиметра;
- **BottomBorderDistance (long)** – расстояние между нижней границей страницы и содержанием страницы в сотых долях миллиметра;
- **ShadowFormat (struct)** – подробное описание для тени области содержимого страницы (структура `com.sun.star.table.ShadowFormat`).

Следующий пример устанавливает левое и правое поля стиля страницы “Обычный” в 1 сантиметр.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Обычный")

DefPage.LeftMargin = 1000
DefPage.RightMargin = 1000
```

Верхний и нижний колонтитулы

Верхний и нижний колонтитулы документа являются частью свойств страницы и определяются с использованием сервиса `com.sun.star.style.PageProperties`. Свойства для форматирования верхнего колонтитула:

- **HeaderIsOn (Boolean)** – верхний колонтитул включен;
- **HeaderLeftMargin (long)** – расстояние между верхним колонтитулом и левым краем страницы в сотых долях миллиметра;
- **HeaderRightMargin (long)** – расстояние между верхним колонтитулом и правым краем страницы в сотых долях миллиметра;

- **HeaderBodyDistance (long)** – расстояние между верхним колонтитулом и основным полем документа в сотых долях миллиметра;
- **HeaderHeight (long)** – высота верхнего колонтитула в сотых долях миллиметра;
- **HeaderIsDynamicHeight (Boolean)** – высота верхнего колонтитула автоматически устанавливается в зависимости от содержимого;
- **HeaderLeftBorder (struct)** – подробное описание левой границы структуры вокруг верхнего колонтитула (структура `com.sun.star.table.BorderLine`);
- **HeaderRightBorder (struct)** – подробное описание правой границы структуры вокруг верхнего колонтитула (структура `com.sun.star.table.BorderLine`);
- **HeaderTopBorder (struct)** – подробное описание верхней границы структуры вокруг верхнего колонтитула (структура `com.sun.star.table.BorderLine`);
- **HeaderBottomBorder (struct)** – подробное описание нижней границы структуры вокруг верхнего колонтитула (структура `com.sun.star.table.BorderLine`);
- **HeaderLeftBorderDistance (long)** – расстояние между левой границей и содержимым верхнего колонтитула в сотых долях миллиметра;
- **HeaderRightBorderDistance (long)** – расстояние между правой границей и содержимым верхнего колонтитула в сотых долях миллиметра;
- **HeaderTopBorderDistance (long)** – расстояние между верхней границей и содержимым верхнего колонтитула в сотых долях миллиметра;
- **HeaderBottomBorderDistance (long)** – расстояние между нижней границей и содержимым верхнего колонтитула в сотых долях миллиметра;
- **HeaderIsShared (Boolean)** – верхние колонтитулы на четных и нечетных страницах имеют одинаковое содержимое (обратитесь к `HeaderText`, `HeaderTextLeft` и `HeaderTextRight`);
- **HeaderBackColor (long)** – цвет фона верхнего колонтитула;
- **HeaderBackGraphicURL (String)** – URL фонового изображения, которое Вы хотите использовать;
- **HeaderBackGraphicFilter (String)** – имя фильтра для интерпретации фонового изображения верхнего колонтитула;
- **HeaderBackGraphicLocation (Enum)** – положение фонового изображения верхнего колонтитула (значение согласно списка `com.sun.star.style.GraphicLocation`);
- **HeaderBackTransparent (Boolean)** – показывает фон верхнего колонтитула как прозрачный;
- **HeaderShadowFormat (struct)** – подробное описание тени верхнего колонтитула (структура `com.sun.star.table.ShadowFormat`).

Свойства для форматирования нижнего колонтитула:

- **FooterIsOn (Boolean)** – нижний колонтитул включен;
- **FooterLeftMargin (long)** – расстояние между нижним колонтитулом и левым краем страницы в сотых долях миллиметра;
- **FooterRightMargin (long)** – расстояние между нижним колонтитулом и правым краем

страницы в сотых долях миллиметра;

- **FooterBodyDistance (long)** – расстояние между нижним колонтитулом и основным полем документа в сотых долях миллиметра;
- **FooterHeight (long)** – высота нижнего колонтитула в сотых долях миллиметра;
- **FooterIsDynamicHeight (Boolean)** – высота нижнего колонтитула автоматически устанавливается в зависимости от содержимого;
- **FooterLeftBorder (struct)** – подробное описание левой границы структуры вокруг нижнего колонтитула (структура com.sun.star.table.BorderLine);
- **FooterRightBorder (struct)** – подробное описание правой границы структуры вокруг нижнего колонтитула (структура com.sun.star.table.BorderLine);
- **FooterTopBorder (struct)** – подробное описание верхней границы структуры вокруг нижнего колонтитула (структура com.sun.star.table.BorderLine);
- **FooterBottomBorder (struct)** – подробное описание нижней границы структуры вокруг нижнего колонтитула (структура com.sun.star.table.BorderLine);
- **FooterLeftBorderDistance (long)** – расстояние между левой границей и содержимым нижнего колонтитула в сотых долях миллиметра;
- **FooterRightBorderDistance (long)** – расстояние между правой границей и содержимым нижнего колонтитула в сотых долях миллиметра;
- **FooterTopBorderDistance (long)** – расстояние между верхней границей и содержимым нижнего колонтитула в сотых долях миллиметра;
- **FooterBottomBorderDistance (long)** – расстояние между нижней границей и содержимым нижнего колонтитула в сотых долях миллиметра;
- **FooterIsShared (Boolean)** – нижние колонтитулы на четных и нечетных страницах имеют одинаковое содержимое (обратитесь к `FooterText`, `FooterTextLeft` и `FooterTextRight`);
- **FooterBackColor (long)** – цвет фона нижнего колонтитула;
- **FooterBackGraphicURL (String)** – URL фонового изображения, которое Вы хотите использовать;
- **FooterBackGraphicFilter (String)** – имя фильтра для интерпретации фонового изображения нижнего колонтитула;
- **FooterBackGraphicLocation (Enum)** – положение фонового изображения нижнего колонтитула (значение согласно списка `com.sun.star.style.GraphicLocation`);
- **FooterBackTransparent (Boolean)** – показывает фон нижнего колонтитула как прозрачный;
- **FooterShadowFormat (struct)** – подробное описание тени нижнего колонтитула (структура `com.sun.star.table.ShadowFormat`).

Изменение Текста верхнего и нижнего колонтитулов

К содержимому верхнего и нижнего колонтитулов в электронной таблице получают доступ через следующие свойства:

- **LeftPageHeaderContent (Object)** – содержимое верхнего колонтитула для четных

страниц (сервис `com.sun.star.sheet.HeaderFooterContent`);

- **RightPageHeaderContent (Object)** – содержимое верхнего колонтитула для нечетных страниц (сервис `com.sun.star.sheet.HeaderFooterContent`);
- **LeftPageFooterContent (Object)** - содержимое нижнего колонтитула для четных страниц (сервис `com.sun.star.sheet.HeaderFooterContent`);
- **RightPageFooterContent (Object)** - содержимое нижнего колонтитула для нечетных страниц (сервис `com.sun.star.sheet.HeaderFooterContent`).

Если Вам не требуется различить верхние или нижние колонтитулы для четных и нечетных страниц (свойство `FooterIsShared` равно `False`), то устанавливают свойства для верхних и нижних колонтитулов на нечетных страницах.

Все названные объекты возвращают объект, который поддерживает сервис `com.sun.star.sheet.HeaderFooterContent`. Посредством (фиктивных) свойств `LeftText`, `CenterText`, и `RightText`, этот сервис предоставляет три элемента текста для верхних и нижних колонтитулов OpenOffice.org Calc.

Следующий пример записывает значение “Объективный тест.” в левое текстовое поле верхнего колонтитула из стиля “Обычный”.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object
Dim HContent As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies

PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Обычный")

DefPage.HeaderIsOn = True
HContent = DefPage.RightPageHeaderContent
HText = HContent.LeftText
HText.String = "Объективный тест."

DefPage.RightPageHeaderContent = HContent
```

Отметьте последнюю строку в примере: Как только текст изменен, объект `TextContent` должен быть снова присвоен верхнему колонтитулу так, чтобы изменение имело силу.

Другой механизм для изменения текста верхних и нижних колонтитулов доступен для текстовых документов (OpenOffice.org Writer), потому что они состоят из единственного блока текста. Следующие свойства определены в сервисе `com.sun.star.style.PageProperties`:

- **HeaderText (Object)** – текстовый объект с содержанием верхнего колонтитула (сервис `com.sun.star.text.XText`);
- **HeaderTextLeft (Object)** – текстовый объект с содержанием верхнего колонтитула на левых страницах (сервис `com.sun.star.text.XText`);
- **HeaderTextRight (Object)** – текстовый объект с содержанием верхнего колонтитула на правых страницах (сервис `com.sun.star.text.XText`);
- **FooterText (Object)** – текстовый объект с содержанием нижнего колонтитула (сервис `com.sun.star.text.XText`);

- **FooterTextLeft (Object)** – текстовый объект с содержанием нижнего колонтитула на левых страницах (сервис `com.sun.star.text.XText`);
- **FooterTextRight (Object)** – текстовый объект с содержанием нижнего колонтитула на правых страницах (сервис `com.sun.star.text.XText`).

Следующий пример создает верхний колонтитул для стиля страницы “Обычный” для текстовых документов и добавляет текст “Объективный тест.” к верхнему колонтитулу.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies

PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Обычный")

DefPage.HeaderIsOn = True
HText = DefPage.HeaderText

HText.String = "Объективный тест."
```

В этом случае, доступ обеспечивается непосредственно через свойство `HeaderText` стиля страницы, а не объект `HeaderFooterContent`.

Выравнивание по центру (только электронные таблицы)

Сервис `com.sun.star.sheet.TablePageStyle` используется только в стилях страницы OpenOffice.org Calc и позволяет диапазоны ячеек, которые Вы хотите напечатать, выровнять по центру на странице. Этот сервис предоставляет следующие свойства:

- **CenterHorizontally (Boolean)** – содержимое таблицы выравнивается по центру по горизонтали;
- **CenterVertically (Boolean)** – содержимое таблицы выравнивается по центру по вертикали.

Определение элементов для печати (только электронные таблицы)

Когда Вы форматируете листы, Вы можете определить, видимы ли элементы страницы. С этой целью, сервис `com.sun.star.sheet.TablePageStyle` предоставляет следующие свойства:

- **PrintAnnotations (Boolean)** – печатать комментарии ячейки;
- **PrintGrid (Boolean)** – печатать линии сетки ячейки;
- **PrintHeaders (Boolean)** – печатать заголовки столбцов и строк;
- **PrintCharts (Boolean)** – печатать диаграммы, содержащиеся на листе;
- **PrintObjects (Boolean)** – печатать внедренные объекты;
- **PrintDrawing (Boolean)** – печатать рисованные объекты;
- **PrintDownFirst (Boolean)** – если содержимое листа располагается на нескольких страницах, они сначала печатаются в вертикально нисходящем порядке, и затем ниже правая сторона;

- **PrintFormulas (Boolean)** – печатать формулы вместо вычисленных значений;
- **PrintZeroValues (Boolean)** – печатать нулевые значения.

Эффективное редактирование документов электронных таблиц

Принимая во внимание, что предыдущий раздел описал основную структуру документов электронных таблиц, этот раздел описывает сервисы, которые позволяют Вам легко получить доступ к ячейкам или областям ячеек.

Области Ячеек

В дополнение к объекту для отдельных ячеек (сервис `com.sun.star.table.Cell`), OpenOffice.org также предоставляет объекты, которые представляют области ячеек. Такие объекты `CellRange` создаются с использованием вызова `getCellRangeByName` объекта электронная таблица:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Лист1")

CellRange = Sheet.getCellRangeByName("A1:C15")
```

Двоеточие (:) используется для определения области ячеек в документе электронной таблицы. Например, A1:C15 представляет все ячейки в строках 1 - 15 и в столбцах A, B и C.

Местоположение отдельных ячеек в области ячеек может быть определено с использованием метода `getCellByPosition`, где координаты верхней левой ячейки в области ячеек – (0, 0). Следующий пример использует этот метод, чтобы создать объект из ячейки C3.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Лист1")

CellRange = Sheet.getCellRangeByName("B2:D4")
Cell = CellRange.GetCellByPosition(1, 1)
```

Форматирование областей ячеек

Точно так же как отдельные ячейки, Вы можете применить форматирование к областям ячеек, используя сервис `com.sun.star.table.CellProperties`. За дополнительной информацией и примерами использования этого сервиса, см. раздел *Форматирование*.

Вычисления над областями ячеек

Вы можете использовать метод `computeFunction` для выполнения математических операций над областями ячеек. computeFunction ожидает константу в качестве параметра, описывающую математическую функцию, которую Вы хотите использовать. Соответствующие константы определены в списке `com.sun.star.sheet.GeneralFunction`.

Следующие значения доступны:

- **SUM** – сумма всех числовых значений;
- **COUNT** – общее количество всех значений (включая нечисловые значения);
- **COUNTNUMS** – общее количество всех числовых значений;
- **AVERAGE** – среднее арифметическое всех числовых значений;
- **MAX** – наибольшее числовое значение;
- **MIN** – наименьшее числовое значение;
- **PRODUCT** – произведение всех числовых значений;
- **STDEV** - стандартное отклонение;
- **VAR** – дисперсия;
- **STDEVP** - стандартное отклонение, основанное на генеральной совокупности;
- **VARP** - дисперсия, основанная на генеральной совокупности.

Следующий пример вычисляет среднее арифметическое области `A1:C3` и выводит результат в окне сообщения:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Лист1")

CellRange = Sheet.getCellRangeByName("A1:C3")

MsgBox CellRange.computeFunction(com.sun.star.sheet.GeneralFunction.AVERAGE)
```

Удаление содержимого ячейки

Метод `clearContents` упрощает процесс удаления содержимого ячейки и области ячеек, в которых он удаляет один определенный тип содержимого из области ячеек.

Следующий пример удаляет все строки и непосредственное форматирование информации из области `B2:C3`.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Flags As Long

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
CellRange = Sheet.getCellRangeByName("B2:C3")

Flags = com.sun.star.sheet.CellFlags.STRING + _
        com.sun.star.sheet.CellFlags.HARDATTR

CellRange.clearContents(Flags)
```

Флаги, определенные в `clearContents` происходят из списка констант `com.sun.star.sheet.CellFlags`. Этот список предоставляет следующие элементы:

- **VALUE** – числовые значения, которые не форматированы как дата или время;
- **DATETIME** – числовые значения, которые форматированы как дата или время;
- **STRING** – строки;

- **ANNOTATION** – комментарии, связанные с ячейками;
- **FORMULA** – формулы;
- **HARDATTR** – непосредственное форматирование ячеек;
- **STYLES** – косвенное форматирование;
- **OBJECTS** - рисованные объекты, которые связаны с ячейками;
- **EDITATTR** - символ форматирования, который применяется только к частям ячеек.

Вы можете также сложить константы вместе для удаления различной информации, используя вызов `clearContents`.

Поиск и замена содержимого ячейки

Документы электронных таблиц, как и текстовые документы, предоставляют функцию для поиска и замены.

Дескрипторные объекты для поиска и замены в документах электронных таблиц не создаются непосредственно через объект документ, а скорее через список `Sheets`. Следующий пример процесса поиска и замены:

```
Dim Doc As Object
Dim Sheet As Object
Dim ReplaceDescriptor As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

ReplaceDescriptor = Sheet.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"

For I = 0 to Doc.Sheets.Count - 1
  Sheet = Doc.Sheets(I)
  Sheet.ReplaceAll(ReplaceDescriptor)
Next I
```

Этот пример использует первый лист документа для создания `ReplaceDescriptor` и затем применяет его ко всем листам в цикле.

Глава 8. Рисунки и Презентации

Эта глава обеспечивает введение в управляемое макросами создание и редактирование рисунков. Первый раздел описывает структуру рисунков, включая основные элементы, которые содержат рисунки. Второй раздел обращается к более сложным функциям редактирования, таким как группировка, вращение и масштабирование объектов.

Информация о создании, открытии, и сохранении рисунков может быть найдена в Главе 5, Работа с документами OpenOffice.org.

Структура рисунков

OpenOffice.org не ограничивает число страниц в документе рисунка. Вы можете проектировать каждую страницу отдельно. Нет также никакого ограничения на количество элементов рисунка, которые Вы можете добавить к странице.

Эта картина немного усложняется присутствием *слоев*. По умолчанию, каждый документ рисунка содержит слои *Расположение*, *Средства управления*, и *Размерные линии* и все элементы рисунка добавляются к слою *Расположение*. Вы также имеете возможность добавлять новые слои. См. Руководство разработчика OpenOffice.org для получения дополнительной информации о слоях рисунка.

Страницы

Страницы документа рисунок доступны через список `DrawPages`. Вы можете получить доступ к отдельным страницам или по их номеру или по их имени. Если документ имеет одну страницу, и она называется *Страница 1*, то следующие примеры идентичны.

Пример 1:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
```

Пример 2:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages.getByName("Страница 1")
```

В примере 1, к странице обращается по ее номеру (нумерация начинается с 0). Во втором примере, к странице получают доступ по ее имени используя метод `getByName`.

Предыдущий вызов возвращает объект страница, который поддерживает сервис

`com.sun.star.drawing.DrawPage`. Сервис распознает следующие свойства:

- **BorderLeft (Long)** – левая граница в сотых долях миллиметра;
- **BorderRight (Long)** – правая граница в сотых долях миллиметра;
- **BorderTop (Long)** – верхняя граница в сотых долях миллиметра;
- **BorderBottom (Long)** – нижняя граница в сотых долях миллиметра;
- **Width (Long)** – ширина страницы в сотых долях миллиметра;
- **Height (Long)** – высота страницы в сотых долях миллиметра;
- **Number (Short)** – номер страницы (нумерация начинается с 1), только для чтения;
- **Orientation (Enum)** – ориентация страницы (в соответствии с списком `com.sun.star.view.PaperOrientation`)

Если эти параметры изменяются, то затрагиваются *все* страницы в документе.

Следующий пример устанавливает размер страницы документа рисунок, который был только что открыт в 20 x 20 сантиметров с полем 0.5 сантиметра:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

Page.BorderLeft = 500
Page.BorderRight = 500
Page.BorderTop = 500
Page.BorderBottom = 500
Page.Width = 20000
Page.Height = 20000
```

Элементарные свойства рисованных объектов

Рисованные объекты включают фигуры (прямоугольники, круги, и так далее), линии, и текстовые объекты. Все они разделяют множество общих свойств и поддерживают сервис `com.sun.star.drawing.Shape`. Этот сервис определяет свойства `Size` и `Position` рисованного объекта.

OOo Basic также предлагает несколько других сервисов, через которые Вы можете изменить такие свойства, как форматирование или применить заливку. Параметры форматирования, которые являются доступными, зависят от типа рисованного объекта.

Следующий пример создает и вставляет прямоугольник в документ рисунок:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
```

```
RectangleShape.Position = Point

Page.add(RectangleShape)
```

Этот пример использует вызов `StarDesktop.CurrentComponent` для определения, какой документ является открытым. Объект документ устанавливает путь возвращающий первую страницу из рисунка через вызов `drawPages(0)`.

Структуры `Point` и `Size` тогда устанавливают начальную точку (левый край) и размер объекта рисунок. Расстояния определяются в сотых долях миллиметра.

Код программы тогда использует вызов Doc.createInstance для создания прямоугольного рисованного объекта как определено сервисом `com.sun.star.drawing.RectangleShape`. В конце, рисованный объект присваивается странице, с использованием вызова `Page.add`.

Свойства заполнения

Этот раздел описывает четыре сервиса, и в каждом случае типовой код программы использует фигуру прямоугольник, которая объединяет несколько типов форматирования. Свойства заполнения объединены в сервисе `com.sun.star.drawing.FillProperties`.

OpenOffice.org признает четыре главных типа форматирования для заполнения области. Самый простой вариант – заполнение единственным цветом. Параметры для определения цветовых градиентов и штриховок позволяют Вам создавать другие цвета в переливах. Четвертым вариантом параметра выступает существующее изображение в заполнении области.

Режим заполнения рисованного объекта определяется с использованием свойства `FillStyle`. Допустимые значения определяются в `com.sun.star.drawing.FillStyle`.

Заполнение единственным цветом

Основное свойство для заполнения единственным цветом

- **FillColor (Long)** – цвет заполнения области.

Для использования режима заполнения, Вы должны свойству `FillStyle` назначить режим заполнения `SOLID`.

Следующий пример создает фигуру прямоугольник и заполняет его красным (RGB значение 255, 0, 0):

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.SOLID
RectangleShape.FillColor = RGB(255, 0, 0)

Page.add(RectangleShape)
```

Цветовой градиент

Если Вы устанавливаете значение `FillStyle` в `GRADIENT`, Вы можете применить цветовой градиент к любому заполнению области документа OpenOffice.org.

Если Вы хотите применить один из предопределенных цветовых градиентов, Вы можете назначить соответствующее имя свойству `FillTransparenceGradientName`. Для определения вашего собственного цветового градиента, Вы должны заполнить структуру `com.sun.star.awt.Gradient`, чтобы присвоить свойству `FillGradient`. Это свойство предоставляет следующие параметры:

- **Style (Enum)** – тип градиента, например, линейный или радиальный (по умолчанию значение в соответствии с `com.sun.star.awt.GradientStyle`);
- **StartColor (Long)** – начальный цвет цветового градиента;
- **EndColor (Long)** – конечный цвет цветового градиента;
- **Angle (Short)** – угол цветового градиента в десятых долях градуса;
- **XOffset (Short)** – X-координата, в которой начинается цветовой градиент, определяется в сотых долях миллиметра;
- **YOffset (Short)** – Y-координата, в которой начинается цветовой градиент, определяется в сотых долях миллиметра;
- **StartIntensity (Short)** – интенсивность `StartColor` как процент (в OOo Basic Вы можете также определить значение больше чем 100 процентов);
- **EndIntensity (Short)** – интенсивность `EndColor` как процент (в OOo Basic Вы можете также определить значение больше чем 100 процентов);
- **StepCount (Short)** – число цветовых делений, которые OpenOffice.org должен вычислить для градиента.

Следующий пример демонстрирует использование цветовых градиентов при помощи структуры `com.sun.star.awt.Gradient`:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim Gradient As New com.sun.star.awt.Gradient

Point.x = 1000
Point.y = 1000

Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")

RectangleShape.Size = Size
RectangleShape.Position = Point

Gradient.Style = com.sun.star.awt.GradientStyle.LINEAR
Gradient.StartColor = RGB(255, 0, 0)
Gradient.EndColor = RGB(0, 255, 0)
Gradient.StartIntensity = 150
Gradient.EndIntensity = 150
Gradient.Angle = 450
Gradient.StepCount = 100
```

```
RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
RectangleShape.FillGradient = Gradient

Page.add(RectangleShape)
```

Этот пример создает линейный цветовой градиент (`Style = LINEAR`). Градиент начинается с красного (`StartColor`) в верхнем левом углу, и простирается под углом 45 градусов (`Angle`) к зеленому (`EndColor`) в нижнем правом углу. Цветовая интенсивность начального и конечного цветов - 150 процентов (`StartIntensity` и `EndIntensity`), которая приводит к цветам, кажущимся более ярким чем значения, определенные в свойствах `StartColor` и `EndColor`. Цветовой градиент изображается с использованием ста делений отдельных цветов (`StepCount`).

Штриховки

Для создания штриховки, свойство `FillStyle` должно быть установлено в `HATCH`. Код программы для определения штриховки очень подобен коду для цветовых градиентов. Снова используется вспомогательная структура, в этом случае `com.sun.star.drawing.Hatch`, для определения появления штриховки. Структура для штриховки имеет следующие свойства:

- **Style (Enum)** – тип штриховки: простой, перекрестный или тройной (по умолчанию значение в соответствии с `com.sun.star.awt.HatchStyle`);
- **Color (Long)** – цвет линий;
- **Distance (Long)** - расстояние между линиями в сотых долях миллиметра;
- **Angle (Short)** - угол штриховки в десятых долях градуса.

Следующий пример демонстрирует использование структуры штриховки:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim Hatch As New com.sun.star.drawing.Hatch

Point.x = 1000
Point.y = 1000

Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point
RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.HATCH

Hatch.Style = com.sun.star.drawing.HatchStyle.SINGLE
Hatch.Color = RGB(64, 64, 64)
Hatch.Distance = 20
Hatch.Angle = 450

RectangleShape.FillHatch = Hatch

Page.add(RectangleShape)
```

Этот код создает структуру простой штриховки (`HatchStyle = SINGLE`), чьи линии повернуты на 45 градусов (`Angle`). Линии темно-серые (`Color`) и с интервалом 0.2 миллиметра (`Distance`).

Растровое изображение

Чтобы использовать проекцию растрового изображения в качестве заполнения, Вы должны установить свойство `FillStyle` в `BITMAP`. Если растровое изображение уже доступно в OpenOffice.org, Вы должны только определить его имя в свойстве `FillBitMapName` и его стиль отображения (простой, мозаика, или растягивание) в свойстве `FillBitmapMode` (по умолчанию значения в соответствии с `com.sun.star.drawing.BitmapMode`).

Если Вы хотите использовать внешний файл растрового изображения, Вы можете определить его URL в свойстве `FillBitmapURL`.

Следующий пример создает прямоугольник и заполняет его мозаикой растровым изображением Небо, которое является доступным в OpenOffice.org для заполнения области прямоугольника:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point
RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
RectangleShape.FillBitmapName = "Sky"
RectangleShape.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT

Page.add(RectangleShape)
```

Прозрачность

Вы можете настроить прозрачность любого заполнения, которое Вы применили. Самый простой способ изменять прозрачность элемента рисунка состоит в том, чтобы использовать свойство `FillTransparence`.

Следующий пример создает красный прямоугольник с прозрачностью 50 процентов.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.SOLID
RectangleShape.FillTransparence = 50
RectangleShape.FillColor = RGB(255, 0, 0)

Page.add(RectangleShape)
```

Чтобы сделать заполнение прозрачным, установите свойство `FillTransparence` в 100.

В дополнение к свойству `FillTransparence`, сервис `com.sun.star.drawing.FillProperties` также предоставляет свойство `FillTransparenceGradient`. Оно используется для определения градиента, который определяет прозрачность заполнения области.

Свойства линии

Все объекты рисунка, которые могут иметь границу, поддерживают сервис `com.sun.star.drawing.LineStyle`. Некоторые из свойств, которые этот сервис предоставляет:

- **LineStyle (Enum)** – тип линии (по умолчанию значение в соответствии с `com.sun.star.drawing.LineStyle`);
- **LineColor (Long)** – цвет линии;
- **LineTransparence (Short)** – прозрачность линии;
- **LineWidth (Long)** – толщина линии в сотых долях миллиметра;
- **LineJoint (Enum)** - переходы в соединяемых точках (по умолчанию значение в соответствии с `com.sun.star.drawing.LineJoint`).

Следующий пример создает прямоугольник со сплошной границей (`LineStyle = SOLID`), 5 толщиной миллиметров (`LineWidth`) и прозрачностью 50 процентов. Правые и левые концы линии продолжаются до их точки пересечения друг с другом (`LineJoint = MITER`) для формирования прямого угла.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.LineColor = RGB(128, 128, 128)
RectangleShape.LineTransparence = 50
RectangleShape.LineWidth = 500
RectangleShape.LineJoint = com.sun.star.drawing.LineJoint.MITER

RectangleShape.LineStyle = com.sun.star.drawing.LineStyle.SOLID

Page.add(RectangleShape)
```

В дополнение к перечисленным свойствам, сервис `com.sun.star.drawing.LineStyle` предоставляет параметры для рисования точечных и пунктирных линий.

Свойства текста (рисованные объекты)

Сервисы `com.sun.star.style.CharacterProperties` и `com.sun.star.style.ParagraphProperties` могут форматировать текст в рисованном объекте. Эти сервисы касаются отдельных символов и абзацев и подробно описаны в

Главе 6 (*Текстовые документы*).

Следующий пример вставляет текст в прямоугольник и форматирует шрифт с использованием сервиса `com.sun.star.style.CharacterProperties`.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)

RectangleShape.String = "Это тест"
RectangleShape.CharWeight = com.sun.star.awt.FontWeight.BOLD
RectangleShape.CharFontName = "Arial"
```

Этот код использует свойство `String` прямоугольника для вставки текста и свойства `CharWeight` и `CharFontName` сервиса `com.sun.star.style.CharacterProperties` для форматирования шрифта текста.

Текст может быть вставлен только после того, как рисованный объект был добавлен на страницу рисунка. Вы можете также использовать сервис `com.sun.star.drawing.Text` для размещения и форматирования текст в рисованном объекте. Далее приведены некоторые из важных свойств этого сервиса:

- **TextAutoGrowHeight (Boolean)** – подстраивает высоту элемента рисунка к тексту, который он содержит;
- **TextAutoGrowWidth (Boolean)** – подстраивает ширину элемента рисунка к тексту, который он содержит;
- **TextHorizontalAdjust (Enum)** – горизонтальное положение текста в пределах элемента рисунка (по умолчанию значение в соответствии с `com.sun.star.drawing.TextHorizontalAdjust`);
- **TextVerticalAdjust (Enum)** - вертикальное положение текста в пределах элемента рисунка (по умолчанию значение в соответствии с `com.sun.star.drawing.TextVerticalAdjust`);
- **TextLeftDistance (Long)** – расстояние между левым краем элемента рисунка и текстом в сотых долях миллиметра;
- **TextRightDistance (Long)** – расстояние между правым краем элемента рисунка и текстом в сотых долях миллиметра;
- **TextUpperDistance (Long)** – расстояние между верхним краем элемента рисунка и текстом в сотых долях миллиметра;
- **TextLowerDistance (Long)** - расстояние между нижним краем элемента рисунка и текста в сотых долях миллиметра.

Следующий пример демонстрирует использование названных свойств.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)

RectangleShape.String = "Это тест" ' Может иметь место только после Page.add!

RectangleShape.TextVerticalAdjust = com.sun.star.drawing.TextVerticalAdjust.TOP
RectangleShape.TextHorizontalAdjust = com.sun.star.drawing.TextHorizontalAdjust.LEFT

RectangleShape.TextLeftDistance = 300
RectangleShape.TextRightDistance = 300
RectangleShape.TextUpperDistance = 300
RectangleShape.TextLowerDistance = 300
```

Этот код вставляет элемент рисунка на страницу и затем добавляет текст в верхний левый угол рисованного объекта, используя свойства `TextVerticalAdjust` и `TextHorizontalAdjust`. Минимальное расстояние между текстом и краем рисованного объекта устанавливается в три миллиметра.

Свойства тени

Вы можете добавить тень к большинству объектов рисунка при помощи сервиса `com.sun.star.drawing.ShadowProperties`. Свойства этого сервиса:

- **Shadow (Boolean)** – активизирует тень;
- **ShadowColor (Long)** – цвет тени;
- **ShadowTransparence (Short)** – прозрачность тени;
- **ShadowXDistance (Long)** – вертикальное расстояние тени от рисованного объекта в сотых долях миллиметра;
- **ShadowYDistance (Long)** – горизонтальное расстояние тени от рисованного объекта в сотых долях миллиметра.

Следующий пример создает прямоугольник с тенью, которая смещена по вертикали и горизонтали от прямоугольника на 2 миллиметра. Тень предоставлена темно-серым цветом с 50-процентной прозрачностью.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
```

```
RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.Shadow = True
RectangleShape.ShadowColor = RGB(192, 192, 192)
RectangleShape.ShadowTransparence = 50
RectangleShape.ShadowXDistance = 200
RectangleShape.ShadowYDistance = 200

Page.add(RectangleShape)
```

Краткий обзор различных рисованных объектов

Прямоугольные фигуры

Объекты прямоугольных фигур (`com.sun.star.drawing.RectangleShape`), поддерживают следующие сервисы для форматирования объектов:

- **Fill properties** – `com.sun.star.drawing.FillProperties`;
- **Line properties** – `com.sun.star.drawing.LineProperties`;
- **Text properties** – `com.sun.star.drawing.Text` (с `com.sun.star.style.CharacterProperties` и `com.sun.star.style.ParagraphProperties`);
- **Shadow properties** – `com.sun.star.drawing.ShadowProperties`;
- **CornerRadius (Long)** – радиус для скругления углов в сотых долях миллиметра.

Окружности и Эллипсы

Сервис `com.sun.star.drawing.EllipseShape` отвечает за круги и эллипсы и поддерживает следующие сервисы:

- **Fill properties** – `com.sun.star.drawing.FillProperties`;
- **Line properties** – `com.sun.star.drawing.LineProperties`;
- **Text properties** – `com.sun.star.drawing.Text` (с `com.sun.star.style.CharacterProperties` и `com.sun.star.style.ParagraphProperties`);
- **Shadow properties** – `com.sun.star.drawing.ShadowProperties`.

В дополнение к указанным сервисам, окружности и эллипсы также предоставляют следующие свойства:

- **CircleKind (Enum)** – тип круга или эллипса (по умолчанию значение в соответствии с `com.sun.star.drawing.CircleKind`);
- **CircleStartAngle (Long)** – начальный угол в сотых долях градуса (только для сегмента окружности или эллипса);
- **CircleEndAngle (Long)** – конечный угол в сотых долях градуса (только для сегмента окружности или эллипса).

Свойство `CircleKind` определяет, является ли объект полным кругом, сектор круга или сегмент круга. Следующие значения доступны:

- **com.sun.star.drawing.CircleKind.FULL** – полный круг или полный эллипс;
- **com.sun.star.drawing.CircleKind.CUT** – сегмент круга (частичный круг, концы которого связаны непосредственно друг с другом);
- **com.sun.star.drawing.CircleKind.SECTION** – сектор круга;
- **com.sun.star.drawing.CircleKind.ARC** – угол (не включая линию окружности).

Следующий пример создает сектор круга с 70 градусным углом (произведенный от различия между начальным углом 20 градусов и конечным углом 90 градусов)

```
Dim Doc As Object
Dim Page As Object
Dim EllipseShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

EllipseShape = Doc.createInstance("com.sun.star.drawing.EllipseShape")
EllipseShape.Size = Size
EllipseShape.Position = Point
EllipseShape.CircleStartAngle = 2000
EllipseShape.CircleEndAngle = 9000
EllipseShape.CircleKind = com.sun.star.drawing.CircleKind.SECTION

Page.add(EllipseShape)
```

Линии

OpenOffice.org предоставляет сервис `com.sun.star.drawing.LineShape` для объектов линия. Объекты линия поддерживают все общие сервисы форматирования за исключением областей. Далее приведены все свойства, которые связаны с сервисом `LineShape`:

- **Line properties** – `com.sun.star.drawing.LineProperties`;
- **Text properties** – `com.sun.star.drawing.Text` (с `com.sun.star.style.CharacterProperties` и `com.sun.star.style.ParagraphProperties`);
- **Shadow properties** – `com.sun.star.drawing.ShadowProperties`;

Следующий пример создает и форматирует линию с помощью названных свойств. Начало линии определяется свойством `Location`, тогда как координаты, перечисленные в свойстве `Size` определяют точку окончания линии.

```
Dim Doc As Object
Dim Page As Object
Dim LineShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

LineShape = Doc.createInstance("com.sun.star.drawing.LineShape")
```

```
LineShape.Size = Size
LineShape.Position = Point

Page.add(LineShape)
```

Многоугольники

OpenOffice.org также поддерживает сложные многоугольные фигуры через сервис `com.sun.star.drawing.PolyPolygonShape`. Строго говоря, *Многомногоугольник* не просто многоугольник, а множественный многоугольник. Несколько независимых списков, содержащих точки углов могут быть определены и объединены для формирования полного объекта.

Как с фигурами прямоугольник, все свойства форматирования рисованных объектов также предоставляются для многомногоугольников:

- **Fill properties** – `com.sun.star.drawing.FillProperties`;
- **Line properties** – `com.sun.star.drawing.LineProperties`;
- **Text properties** – `com.sun.star.drawing.Text` (с `com.sun.star.style.CharacterProperties` и `com.sun.star.style.ParagraphProperties`);
- **Shadow properties** – `com.sun.star.drawing.ShadowProperties`.

Сервис `PolyPolygonShape` также имеет свойство, которое позволяет Вам определять координаты многоугольника:

- **PolyPolygon (Array)** – массив, содержащий координаты многоугольника (двойной массив с точками типа `com.sun.star.awt.Point`).

Следующий пример показывает, как Вы можете определить треугольник с помощью сервиса `PolyPolygonShape`.

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Coordinates(2) As New com.sun.star.awt.Point

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")
' Page.add должен быть вызван перед заданием координат
Page.add(PolyPolygonShape)

Coordinates(0).x = 1000
Coordinates(1).x = 7500
Coordinates(2).x = 10000
Coordinates(0).y = 1000
Coordinates(1).y = 7500
Coordinates(2).y = 5000

PolyPolygonShape.PolyPolygon = Array(Coordinates())
```

Так как вершины многоугольника определяются как абсолютные значения, Вы не должны определять размер или положение начала многоугольника. Вместо этого Вы должны создать массив вершин, упаковать этот массив во втором массиве (используя вызов `Array(Coordinates())`, и затем присвоить это множество многоугольнику. Прежде, чем соответствующий вызов может быть сделан, многоугольник должен быть вставлен в документ.

Двойное массив в определении позволяет Вам создавать сложные формы, объединяя

несколько многоугольников. Вы можете создать прямоугольник и затем вставить в него другой прямоугольник, чтобы создать отверстие в исходном прямоугольнике:

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Square1(3) As New com.sun.star.awt.Point
Dim Square2(3) As New com.sun.star.awt.Point
Dim Square3(3) As New com.sun.star.awt.Point

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")

' Page.add должен быть вызван перед заданием координат
Page.add(PolyPolygonShape)

Square1(0).x = 5000
Square1(1).x = 10000
Square1(2).x = 10000
Square1(3).x = 5000
Square1(0).y = 5000
Square1(1).y = 5000
Square1(2).y = 10000
Square1(3).y = 10000

Square2(0).x = 6500
Square2(1).x = 8500
Square2(2).x = 8500
Square2(3).x = 6500
Square2(0).y = 6500
Square2(1).y = 6500
Square2(2).y = 8500
Square2(3).y = 8500

Square3(0).x = 6500
Square3(1).x = 8500
Square3(2).x = 8500
Square3(3).x = 6500
Square3(0).y = 9000
Square3(1).y = 9000
Square3(2).y = 9500
Square3(3).y = 9500

PolyPolygonShape.PolyPolygon = Array(Square1(), Square2(), Square3())
```

Что касается того, какие области заполняются и какие области являются отверстиями, OpenOffice.org применяет простое правило: край внешней фигуры всегда внешняя граница многомногоугольника. Следующая внутренняя линия – внутренний край фигуры и отмечает переход к первому отверстию. Если есть другая внутренняя линия, она отмечает переход к заполненной области.

Графика

Последние из элементов рисунка, представленных здесь – графические объекты, которые основанны на сервисе `com.sun.star.drawing.GraphicObjectShape`. Они могут использоваться с любой графикой в пределах OpenOffice.org, появление которой может быть приспособлено с использованием целого диапазона свойств.

Графические объекты поддерживают два из общих свойств форматирования:

- **Свойство Text** – `com.sun.star.drawing.Text` (с `com.sun.star.style.CharacterProperties` и `com.sun.star.style.ParagraphProperties`);
- **Свойство Shadow** – `com.sun.star.drawing.ShadowProperties`.

Дополнительные свойства, которые поддерживаются графическими объектами:

- **GraphicURL (String)** – URL изображения;
- **AdjustLuminance (Short)** – яркость цветов, как процент (отрицательные значения также разрешены);
- **AdjustContrast (Short)** – контраст как процент (отрицательные значения также разрешены);
- **AdjustRed (Short)** – значение красного как процент (отрицательные значения также разрешены);
- **AdjustGreen (Short)** – значение зеленого как процент (отрицательные значения также разрешены);
- **AdjustBlue (Short)** – значение синего как процент (отрицательные значения также разрешены);
- **Gamma (Short)** – значение гаммы изображения;
- **Transparency (Short)** – прозрачность изображения как процент;
- **GraphicColorMode (enum)** – режим цвета, например, стандартный, градации серого, черно-белое (по умолчанию значение в соответствии с `com.sun.star.drawing.ColorMode`).

Следующий пример показывает, как вставить графический объект на страницу.

```
Dim Doc As Object
Dim Page As Object
Dim GraphicObjectShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

' спецификации, незначащие, потому что связываются последние координаты
Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

GraphicObjectShape = Doc.createInstance("com.sun.star.drawing.GraphicObjectShape")

GraphicObjectShape.Size = Size
GraphicObjectShape.Position = Point

GraphicObjectShape.GraphicURL = "file:///c:/test.jpg"
GraphicObjectShape.AdjustBlue = -50
GraphicObjectShape.AdjustGreen = 5
GraphicObjectShape.AdjustBlue = 10
GraphicObjectShape.AdjustContrast = 20
GraphicObjectShape.AdjustLuminance = 50
GraphicObjectShape.Transparency = 40
GraphicObjectShape.GraphicColorMode = com.sun.star.drawing.ColorMode.STANDARD

Page.add(GraphicObjectShape)
```

Этот код вставляет изображение `test.jpg` и настраивает его вид, используя регулирующиеся свойства. В этом примере, отображение изображения с 40 процентами прозрачности без других цветовых преобразований не имеет места (`GraphicColorMode = STANDARD`).

Редактирование объектов рисунка

Группировка Объектов

Во многих ситуациях, полезно собрать в группу несколько отдельных объектов рисунка так, чтобы они вели себя как единый большой объект.

Следующий пример объединяет два объекта рисунка:

```
Dim Doc As Object
Dim Page As Object
Dim Square As Object
Dim Circle As Object
Dim Shapes As Object
Dim Group As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim NewPos As New com.sun.star.awt.Point
Dim Height As Long
Dim Width As Long

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

Point.x = 3000
Point.y = 3000
Size.Width = 3000
Size.Height = 3000

' создание квадратного элемента рисунка
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255, 128, 128)

Page.add(Square)

' создание круглого элемента рисунка
Circle = Doc.createInstance("com.sun.star.drawing.EllipseShape")
Circle.Size = Size
Circle.Position = Point
Circle.FillColor = RGB(0, 255, 0)

Page.add(Circle)

' объединение квадратного и круглого элементов рисунка
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)

Group = Page.group(Shapes)

' центрирование объединенного элемента рисунка
Height = Page.Height
Width = Page.Width
NewPos.X = Width / 2
NewPos.Y = Height / 2
Height = Group.Size.Height
Width = Group.Size.Width
NewPos.X = NewPos.X - Width / 2
NewPos.Y = NewPos.Y - Height / 2

Group.Position = NewPos
```

Этот код создает прямоугольник и круг и вставляет их на страницу. После этого создается объект, который поддерживает сервис `com.sun.star.drawing.ShapeCollection` и используется метод `Add` для добавления прямоугольника и круга к этому объекту. `ShapeCollection` добавляется к странице с использованием метода `Group` и возвращает реальный объект `Group`, который может редактироваться как отдельная фигура.

Если Вы хотите форматировать отдельные объекты группы, применяйте форматирование прежде, чем Вы добавляете их к группе. Вы не можете изменить объекты, как только они находятся в группе.

Вращение и сдвиг объектов рисунка

Все объекты рисунка, которые описаны в предыдущих секциях, могут также вращаться и сдвигаться с использованием сервиса `com.sun.star.drawing.RotationDescriptor`.

Этот сервис предоставляет следующие свойства:

- **RotateAngle (Long)** – угол поворота в сотых долях градуса;
- **ShearAngle (Long)** – угол сдвига в сотых долях градуса.

Следующий пример создает прямоугольник и поворачивает его на 30 градусов, используя свойство `RotateAngle`:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.RotateAngle = 3000

Page.add(RectangleShape)
```

Следующий пример создает тот же самый прямоугольник как в предыдущем примере, но вместо этого сдвигает его на 30 градусов, используя свойство `ShearAngle`.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.ShearAngle = 3000

Page.add(RectangleShape)
```

Поиск и Замена

Как в текстовых документах, рисованные документы обеспечивают функцию для поиска и

замены. Эта функция подобна той, которая используется в текстовых документах как описано в Главе 6, *Текстовые документы*. Однако, в рисованном документе дескрипторные объекты для поиска и замены не создаются непосредственно через объект документа, а скорее через связанный уровень символа. Следующий пример намечает в общих чертах процесс замены в пределах рисунка:

```
Dim Doc As Object
Dim Page As Object
Dim ReplaceDescriptor As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

ReplaceDescriptor = Page.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"

For I = 0 to Doc.drawPages.Count - 1
  Page = Doc.drawPages(I)
  Page.ReplaceAll(ReplaceDescriptor)
Next I
```

Этот код использует первую страницу документа `DrawPage` для создания `ReplaceDescriptor` и затем применяет этот описатель в цикле ко всем страницам в документе рисунка.

Презентации

Презентации OpenOffice.org основаны на документах рисунка. Каждая страница в презентации – слайд. Вы можете получить доступ к слайдам таким же образом, как к стандартному рисунку получают доступ через список `DrawPages` объекта документ. Сервис `com.sun.star.presentation.PresentationDocument`, ответственный за документы презентаций, также обеспечивает полный сервис `com.sun.star.drawing.DrawingDocument`.

Работа с презентациями

В дополнение к функциям рисунка, которые обеспечиваются свойством `Presentation`, документ презентации имеет объект презентация, который обеспечивает доступ к основным свойствам и механизмам управления для презентаций. Например, этот объект обеспечивает метод `start`, который запускает презентацию.

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation

Presentation.start()
```

Код, используемый в этом примере создает объект `Doc`, для ссылки на текущий документ презентации и устанавливает связанный объект презентации. Метод `start()` объекта используется для запуска примера и управления демонстрацией презентации.

Следующие методы предоставляются объектом презентации:

- **start** – запускает презентацию;
- **end** – завершает презентацию;

- **rehearseTimings** – начинает презентацию с начала и устанавливает ее время выполнения.

Также доступны следующие свойства:

- **AllowAnimations (Boolean)** – управляет анимацией в презентации;
- **CustomShow (String)** – позволяет Вам определять название презентации так, чтобы Вы могли ссылаться на название в презентации;
- **FirstPage (String)** – название слайда, с которого Вы хотите начать презентацию;
- **IsAlwaysOnTop (Boolean)** – всегда показывает окно презентации как первое окно на экране;
- **IsAutomatic (Boolean)** – автоматически запускает презентацию;
- **IsEndless (Boolean)** – повторно запускает презентацию с начала, как только она заканчивается;
- **IsFullScreen (Boolean)** – автоматически запускает презентацию в полноэкранном режиме;
- **IsMouseVisible (Boolean)** – отображает курсор мыши во время презентации;
- **Pause (long)** – время, в течение которого отображается пустой экран в конце презентации;
- **StartWithNavigator (Boolean)** – отображает окно навигатора, когда начинается презентация;
- **UsePn (Boolean)** – отображает указатель во время презентации.

Глава 9
Диаграммы

OpenOffice.org может отображать данные в виде диаграмм, которые создают графические связи между данными в форме гистограмм, круговых диаграмм, линий или других элементов. Данные могут быть показаны как 2-х или 3-х мерная графика, и появление элементов диаграммы может быть индивидуально подобрано подобным способом для процессов, используемых для рисованных элементов.

Если данные доступны в форме электронной таблицы, то она может быть динамически связана с диаграммой. Любые изменения, сделанные в исходных данных могут в этом случае быть немедленно отражены в заданной диаграмме. Эта глава предоставляет краткий обзор программного интерфейса для модулей диаграмм OpenOffice.org и сосредотачивается на использовании диаграмм в пределах документов электронных таблиц.

Использование диаграмм в электронных таблицах

Диаграммы не рассматривают как независимые документы в OpenOffice.org, но как объекты, которые являются вложенными в существующий документ.

В то время как диаграммы в текстовых и рисованных документах остаются изолированными от содержимого документа, когда они используется в документах электронных таблицы, обеспечивается механизм, который позволяет установить связи между данными документа и вложенными диаграммами. Следующий пример объясняет взаимодействие между документом электронной таблицы и диаграммой:

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
```

Хотя код, используемый в примере, может показаться сложным, основные процессы ограничены тремя строками: первая основная строка создает переменную документа `Doc`,

которая ссылается на текущий документ электронной таблицы (строка `Doc = StarDesktop.CurrentComponent`). Код, используемый в примере далее создает список, содержащий все диаграммы первой электронной таблицы (строка `Charts = Doc.Sheets(0).Charts`). Наконец, новая диаграмма добавляется к последней строке этого списка, с использованием метода `addNewByName`. Эта новая диаграмма тогда становится видимой пользователю.

Последняя строка инициализирует вспомогательные структуры `Rect` и `RangeAddress`, которые метод `addNewByName` также предусматривает как параметр. `Rect` определяет положение диаграммы в пределах электронной таблицы. `RangeAddress` определяет диапазон, данные которого должны быть связаны с диаграммой.

Предыдущий пример создает столбчатую диаграмму. Если необходим отличный тип графики, то столбчатая диаграмма должна быть явно изменена:

```
Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

Первая строка определяют соответствующий объект диаграммы. Вторая строка заменяет текущий тип диаграммы новым - в этом примере, линейной диаграммой.

Примечание В Excel, различие сделано между диаграммами, которые были вставлены как отдельная страница в документе Excel и диаграммами, которые являются вложенными в табличную страницу. Соответственно, там для диаграмм определены два различных метода доступа. Это различие не делается в OOo Basic, потому что диаграммы в OpenOffice.org Calc всегда создаются как внедренные объекты табличной страницы. К диаграммам всегда получают доступ, используя список `Charts` связанного объекта `Sheet`.

Структура Диаграмм

Структура диаграммы – и, поэтому, список сервисов и интерфейсов, поддерживаемых ею - зависит от ее типа. Методы и свойства Оси Z, например, доступны только в трехмерных диаграммах, но не в двумерных. В круговых диаграммах нет никаких интерфейсов для работы с осями.

Отдельные элементы диаграммы

Заголовок, подзаголовок и легенда

Заголовок, подзаголовок и легенда являются частью основных элементов каждой диаграммы. Диаграммы обеспечивают свои собственные объекты для каждого из этих элементов. Объект Диаграмма предоставляет следующие свойства для управления этими элементами:

- **HasMainTitle (Boolean)** – активизирует заголовок;
- **Title (Object)** – объект с подробной информацией о заголовке диаграммы (поддерживается сервисом `com.sun.star.chart.ChartTitle`);
- **HasSubTitle(Boolean)** – активизирует подзаголовок;
- **Subtitle (Object)** – объект с подробной информацией о подзаголовке диаграммы

(поддерживается сервисом `com.sun.star.chart.ChartTitle`).

- **HasLegend (Boolean)** – активизирует легенду;
- **Legend (Object)** – объект с подробной информацией о легенде к диаграмме (поддерживается сервисом `com.sun.star.chart.ChartLegendPosition`).

Во многих отношениях, определенные элементы соответствуют элементу рисунка. Вследствие этого, оба сервиса `com.sun.star.chart.ChartTitle` и `com.sun.star.chart.ChartLegendPosition` поддерживают сервис `com.sun.star.drawing.Shape`, который формирует техническую основу программы для рисованных элементов.

Пользователи поэтому имеют возможность определить положение и размер элемента, используя свойства `Size` и `Position`.

Другие свойства заполнения и линии (сервисы `com.sun.star.drawing.FillProperties` и `com.sun.star.drawing.LineStyle`), так же как свойства символа (сервис `com.sun.star.style.CharacterProperties`), предоставляются для форматирования элементов.

Сервис `com.sun.star.chart.ChartTitle` содержит не только имена свойств форматирования, но также и два других свойства:

- **TextRotation (Long)** – угол поворота текста в сотых долях градуса;
- **String (String)** – текст для отображения как заголовок или подзаголовок

Легенда диаграммы (сервис `com.sun.star.chart.ChartLegend`), содержит следующее дополнительное свойство:

- **Alignment (Enum)** – положение, в котором появляется легенда (значение по умолчанию в соответствии с `com.sun.star.chart.ChartLegendPosition`).

Следующий пример создает диаграмму и присваивает ей заголовок “Тест”, подзаголовок “Тест 2” и легенду. Легенда имеет серый цвет фона, размещается у основания диаграммы, и имеет размер символов 7 пунктов.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts
Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Chart.HasMainTitle = True
Chart.Title.String = "Test"

Chart.HasSubTitle = True
Chart.Subtitle.String = "Test 2"
```

```
Chart.HasLegend = True
Chart.Legend.Alignment = com.sun.star.chart.ChartLegendPosition.BOTTOM
Chart.Legend.FillStyle = com.sun.star.drawing.FillStyle.SOLID
Chart.Legend.FillColor = RGB(210, 210, 210)
Chart.Legend.CharHeight = 7
```

Фон

Каждая диаграмма имеет фоновую область. Каждая область имеет объект, к которому можно получить доступ, используя следующие свойства объекта диаграмма:

- **Area (Object)** – фоновая область диаграммы (поддерживается сервисом `com.sun.star.chart.ChartArea`).

Фон диаграммы охватывает ее полную область, включая область под заголовком, подзаголовком и легендой диаграммы. Соответствующий сервис `com.sun.star.chart.ChartArea` поддерживает свойства линии и заполнения и не обеспечивает никаких более обширных свойств.

Стены и основание диаграммы

Хотя фон диаграммы покрывает всю область диаграммы, обратная стена диаграммы ограничена областью непосредственно позади области данных.

Для трехмерных диаграмм обычно существуют две стены диаграммы: одна позади области данных и одна как левая граница по оси Y. Трехмерные диаграммы обычно также имеют основание.

- **Floor (Object)** – панель основания диаграммы (только для трехмерных диаграмм, поддерживается сервисом `com.sun.star.chart.ChartArea`);
- **Wall (Object)** – стены диаграммы (только для трехмерных диаграмм, поддерживается сервисом `com.sun.star.chart.ChartArea`).

Указанные объекты поддерживаются сервисом `com.sun.star.chart.ChartArea`, который в свою очередь предоставляет обычные свойства заполнение и линии (сервисы `com.sun.star.drawing.FillProperties` и `com.sun.star.drawing.LineStyle`, обратитесь к Главе 8).

К стенам и основанию диаграммы получают доступ через объект `Chart`, который в свою очередь является частью объекта диаграмма:

```
Chart.Area.FillBitmapName = "Sky"
```

Следующий пример показывает, как изображение (называемое Небо) уже содержавшийся в OpenOffice.org может использоваться как фон для диаграммы.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12
```

```
Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Chart.Area.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
Chart.Area.FillBitmapName = "Sky"
Chart.Area.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

Оси

OpenOffice.org признает пять различных осей, которые могут использоваться в диаграмме. В самом простом сценарии, это оси X и Y. Когда работают с трехмерными диаграммами, также иногда предоставляется ось Z. Для диаграмм, в которых значения различных рядов данных значительно отклоняются от друг друга, OpenOffice.org предоставляет дополнительные оси X и Y для дополнительных операций масштабирования.

Первичные Оси X, Y и Z

В дополнение к фактической оси, для каждой из первичных осей X, Y, и Z, имеются также заголовок, описание, сетка и вспомогательная сетка. Имеется возможность выбора для отображения и скрытия всех этих элементов. Объект диаграмма предоставляет следующие свойства для управления этими возможностями (приведен пример оси X; свойства для осей Y и Z структурированы таким же образом):

- **HasXAxis (Boolean)** – активизирует ось X;
- **XAxis (Object)** – объект с подробной информацией об оси X (поддерживает сервис `com.sun.star.chart.ChartAxis`);
- **HasXAxisDescription (Boolean)** – активизирует описание для оси X;
- **HasXAxisGrid (Boolean)** – активизирует основную сетку для оси X;
- **XMainGrid (Object)** – объект с подробной информацией об основной сетке для оси X (поддерживает сервис `com.sun.star.chart.ChartGrid`);
- **HasXAxisHelpGrid (Boolean)** – активизирует вспомогательную сетку для оси X;
- **XHelpGrid (Object)** – объект с подробной информацией о вспомогательной сетке для оси X (поддерживает сервис `com.sun.star.chart.ChartGrid`);
- **HasXAxisTitle (Boolean)** – активизирует заголовок оси X;
- **XAxisTitle (Object)** – объект с подробной информацией о заголовке оси X (поддерживает сервис `com.sun.star.chart.ChartTitle`).

Вторичные оси X и Y

Следующие свойства доступны для вторичных осей X и Y (приведен пример свойств вторичной оси X):

- **HasSecondaryXAxis (Boolean)** – активизирует вторичную ось X;
- **SecondaryXAxis (Object)** – объект с подробной информацией о вторичной оси X (поддерживает сервис `com.sun.star.chart.ChartAxis`);
- **HasSecondaryXAxisDescription (Boolean)** – активизирует описание вторичной оси X.

Свойства осей

Объекты ось диаграммы OpenOffice.org поддерживают сервис `com.sun.star.chart.ChartAxis`. В дополнение к свойствам для символов (сервис `com.sun.star.style.CharacterProperties`, обратитесь к Главе 6) и линиям (сервис `com.sun.star.drawing.LineStyle`, обратитесь к Главе 8), он предоставляет следующие свойства:

- **Max (Double)** – максимальное значение для оси;
- **Min (Double)** – минимальное значение для оси;
- **Origin (Double)** – точка пересечения для перекрещивающихся осей;
- **StepMain (Double)** – интервал между двумя первичными линиями оси;
- **StepHelp (Double)** – интервал между двумя вторичными линиями оси;
- **AutoMax (Boolean)** – автоматическое определение максимального значения для оси;
- **AutoMin (Boolean)** – автоматическое определение минимального значения для оси;
- **AutoOrigin (Boolean)** – автоматическое определение точки пересечения для перекрещивающихся осей;
- **AutoStepMain (Boolean)** – автоматическое определение интервала между первичными линиями оси;
- **AutoStepHelp (Boolean)** – автоматическое определение интервала между вторичными линиями оси;
- **Logarithmic (Boolean)** – логарифмический масштаб оси (а не линейный);
- **DisplayLabels (Boolean)** - активизирует текстовую надпись для осей;
- **TextRotation (Long)** - угол поворота текстовой надписи осей в сотых долях градуса;
- **Marks (Const)** – константа, которая определяет, должны ли первичные линии оси быть внутри или снаружи области диаграммы (значения по умолчанию в соответствии с `com.sun.star.chart.ChartAxisMarks`);
- **HelpMarks (Const)** – константа, которая определяет, должны ли вторичные линии оси быть внутри и/или снаружи области диаграммы (значения по умолчанию в соответствии с `com.sun.star.chart.ChartAxisMarks`);
- **Overlap (Long)** – процент, который определяет степень, до которой могут накладываться столбцы различных наборов данных (при 100 %, столбцы отображаются с полным перекрыванием, при -100 %, имеется расстояние шириной одного столбца между ними);
- **GapWidth (long)** – процент, который определяет расстояние, которое может быть между различными группами столбцов диаграммы (при 100 %, имеется расстояние, соответствующее ширине одного столбца);
- **ArrangeOrder (enum)** – подробности положения надписи; в дополнение к расположению в линию, есть также выбор разбиения надписи последовательно на две линии (значение по умолчанию согласно `com.sun.star.chart.ChartAxisArrangeOrderType`);
- **TextBreak (Boolean)** - разрешает разрывы строки;

- **TextCanOverlap (Boolean)** - разрешает наложения текста;
- **NumberFormat (Long)** – формат числа (обратитесь к разделу “Форматы чисел, дат и текста” на странице 97).

Свойства сетки оси

Объект для сетки оси основан на сервисе `com.sun.star.chart.ChartGrid`, который в свою очередь поддерживает свойства линии сервиса `com.sun.star.drawing.LineStyle` (обратитесь к Главе 8).

Свойства заголовка оси

Объекты для форматирования заголовка оси основаны на сервисе `com.sun.star.chart.ChartTitle`, который также используется для заголовков диаграммы.

Пример

Следующий пример создает линейную диаграмму. Цвет для задней стены диаграммы устанавливается белым. Обе оси X и Y имеют серую вспомогательную сетку для визуальной ориентации. Минимальное значение по оси Y установлено в 0, а максимальное значение установлено в 100 так, чтобы разрешение диаграммы сохранялось, даже если значения изменятся.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)

Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
Chart.Diagram.Wall.FillColor = RGB(255, 255, 255)

Chart.Diagram.HasXAxisGrid = True
Chart.Diagram.XMainGrid.LineColor = RGB(192, 192, 192)
Chart.Diagram.HasYAxisGrid = True
Chart.Diagram.YMainGrid.LineColor = RGB(192, 192, 192)
Chart.Diagram.YAxis.Min = 0
Chart.Diagram.YAxis.Max = 100
```

Трехмерные Диаграммы

Большинство диаграмм в OpenOffice.org может также быть показано с использованием трехмерной графики. Все типы диаграммы, которые обеспечивают этот вариант,

поддерживают сервис com.sun.star.chart.Dim3DDiagram. Сервис предоставляет только одно свойство:

- **Dim3D (Boolean)** – включение трехмерного отображения.

Многослойные диаграммы

Многослойные диаграммы – диаграммы, которые организованы с несколькими отдельными значениями одно над другим, для создания полного значения. Это представление отображает не только отдельные значения, но также дает общее представление обо всех значениях.

В OpenOffice.org, различные типы диаграмм могут быть отображены в многослойной форме. Все эти диаграммы поддерживают сервис com.sun.star.chart.StackableDiagram, который в свою очередь предоставляет следующие свойства:

- **Stacked (Boolean)** – активизирует многослойный режим представления;
- **Percent (Boolean)** – отображаются не абсолютные значения, а их процентное распределение.

Типы диаграмм

Линейные диаграммы

Линейные диаграммы (сервис com.sun.star.chart.LineDiagram), поддерживают одну ось X, две оси Y и одну ось Z. Они могут быть показаны как двумерная или трехмерная графика (сервис com.sun.star.chart.Dim3Ddiagram). Линейные диаграммы могут отображаться в виде многослойных диаграмм (com.sun.star.chart.StackableDiagram).

Линейные диаграммы предоставляют следующие свойства:

- **SymbolType (const)** – символ для отображения точек данных (константа в соответствии с com.sun.star.chart.ChartSymbolType);
- **SymbolSize (Long)** – размер символа для отображения точек данных в сотых долях миллиметра;
- **SymbolBitmapURL (String)** – имя файла графики для отображения точек данных;
- **Lines (Boolean)** – связывать точки данных посредством линий;
- **SplineType (Long)** – функция сплайна для сглаживания линии (0: никакой функции сплайна, 1: кубический сплайн, 2: B сплайны);
- **SplineOrder (Long)** – полиномиальный вес для сплайнов (только для B сплайнов);
- **SplineResolution (Long)** - число поддерживаемых точек для вычисления сплайна.

Диаграммы Области

Диаграммы области (сервис com.sun.star.chart.AreaDiagram), поддерживают одну ось X, две оси Y и одну ось Z. Они могут быть показаны как двумерная или трехмерная графика

(сервис `com.sun.star.chart.Dim3Ddiagram`). Диаграммы области могут отображаться в виде многослойных диаграмм (`com.sun.star.chart.StackableDiagram`).

Столбчатые диаграммы

Столбчатые диаграммы (сервис `com.sun.star.chart.BarDiagram`), поддерживают одну ось X, две оси Y и одну ось Z. Они могут быть показаны как двумерная или трехмерная графика (сервис `com.sun.star.chart.Dim3Ddiagram`). Столбчатые диаграммы могут отображаться в виде многослойных диаграмм (`com.sun.star.chart.StackableDiagram`).

Они предоставляют следующие свойства:

- **Vertical (Boolean)** – столбцы отображаются вертикально, в противном случае они отображаются горизонтально;
- **Deep (Boolean)** – в трехмерном режиме отображения, помещать бруски позади друг друга, а не рядом с друг другом;
- **StackedBarsConnected (Boolean)** - связывает соответствующие столбцы в многослойной диаграмме посредством линий (доступный только у горизонтальных диаграмм);
- **NumberOfLines (Long)** - число линий, которые будут показаны в многослойной диаграмме как линии, а не столбцы.

Круговые диаграммы

Круговые диаграммы (сервис `com.sun.star.chart.PieDiagram`), не содержат никаких осей и не могут быть отображены в виде многослойных диаграмм. Они могут быть показаны как двумерная или трехмерная графика (сервис `com.sun.star.chart.Dim3Ddiagram`).

Глава 10
Доступ к базам данных

OpenOffice.org имеет интегрированный интерфейс баз данных (независимый от любых систем) названный Star Database Connectivity (SDBC). Цель развития этого интерфейса состояла в том, чтобы обеспечить доступ к максимально возможному количеству различных источников данных.

Чтобы сделать это возможным, к источникам данных получают доступ через драйверы. Источники, из которых драйверы берут данные, являются несущественными для пользователя SDBC. Некоторый драйверы получают доступ к базам данных на основе файлов и берут данные непосредственно из них. Другие используют стандартные интерфейсы, такие как JDBC или ODBC. Есть, однако, также специальные драйверы, которые получают доступ к записной книжке MAPI, справочникам LDAP или электронным таблицам OpenOffice.org как к источникам данных.

Так как драйверы основаны на компонентах UNO, могут быть разработаны другие драйверы и следовательно делаются доступным новые источники данных. Вы можете найти детальную информацию об этом в Руководстве разработчика OpenOffice.org.

Примечание В терминах понятий, SDBC сопоставим с библиотеками ADO и DAO, доступными в VBA. Он разрешает доступ высокого уровня к базам данных, независимо от лежащего в основе механизма базы данных.

Примечание Интерфейс баз данных OpenOffice.org появился с выпуском OpenOffice.org 2. Хотя в прошлом к базам данных прежде всего получали доступ, используя набор методов объекта `Application`, интерфейс в OpenOffice.org 1.1 подразделяется на несколько объектов. `DatabaseContext` используется как корневой объект для функций базы данных.

SQL: Язык запросов

Язык SQL предусмотрен как язык запросов для пользователей SDBC. Чтобы сгладить различия между разными диалектами SQL, компоненты SDBC от OpenOffice.org имеют свой собственный анализатор SQL. Он использует окно запроса, чтобы проверить введенные команды SQL и исправляет простые синтаксические ошибки, такие как связанные с заглавными и строчными символами.

Если драйвер разрешает доступ к источнику данных, который не поддерживает SQL, то он должно независимо преобразовать переданные команды SQL в родные, необходимые для доступа.

Примечание Выполнение SQL от SDBC ориентируется на SQL-ANSI-Стандарт. Определенные Microsoft расширения, такие как конструкции `INNER JOIN` не поддерживаются. Они должны быть заменены стандартными командами (`INNER JOIN`, например должен быть заменен соответствующим условием `WHERE`).

Типы доступа к базам данных

Интерфейс базы данных OpenOffice.org доступен в приложениях OpenOffice.org Writer и OpenOffice.org Calc, а также в формах базы данных.

В OpenOffice.org Writer, стандартная рассылка писем может быть создана с помощью источников данных SDBC, и она тогда может быть распечатана. Есть также выбор для перемещения данных из окна базы данных в текстовые документы, с использованием функции drag-and-drop.

Если пользователь перемещает таблицу базы данных в электронную таблицу, OpenOffice.org создает область таблицы, которая может быть обновлена щелчком мыши, если оригинальные данные будут изменены. Наоборот, данные электронной таблицы можно переместить в таблицу базы данных и выполнить импорт в базу данных.

Наконец, OpenOffice.org обеспечивает механизм для форм, основанный на базах данных. Для этого, пользователь сначала создает стандартную форму OpenOffice.org Writer или StarOffice Calc и затем связывает поля с базой данных.

Все варианты, определенные здесь основаны на пользовательском интерфейсе OpenOffice.org. Никакого знания программирования не требуется, чтобы использовать соответствующие функции.

Эта глава, однако, предоставляет немного информации об определенных функциях, но вместо этого концентрируется на программном интерфейсе SDBC, который позволяет автоматизировать запросы к базе данных и поэтому дает возможность использовать намного больший диапазон приложений.

Однако необходимы фундаментальные знания принципов функционирования баз данных и языка запросов SQL, чтобы полностью понять следующие разделы.

Источники Данных

База данных включенная в OpenOffice.org, создает то, что обычно упоминается как *источник данных*. Пользовательский интерфейс обеспечивает соответствующий выбор для создания источников данных в **дополнительном** меню. Однако, Вы также можете создавать источники данных и работать с ними используя OOo Basic.

Объект контекста базы данных, который создан с использованием функции `createUnoService`, служит отправной точкой для получения доступа к источнику данных. Он основан на сервисе `com.sun.star.sdb.DatabaseContext` и является корневым объектом для всех операций с базами данных.

Следующий пример показывает, как контекст базы данных создается и затем используется, для определения имен всех доступных источников данных. Он показывает имена в окне сообщения.

```
Dim DatabaseContext As Object
Dim Names
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")

Names = DatabaseContext.getElementNames()
For I = 0 To UBound(Names())
  MsgBox Names(I)
Next I
```

Отдельные источники данных основаны на сервисе `com.sun.star.sdb.DataSource` и могут быть определены из контекста базы данных с использованием метода `getByName`:

```
Dim DatabaseContext As Object
Dim DataSource As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Клиенты")
```

Пример создает объект `DataSource` для источника данных по имени *Клиенты*.

Источники данных предоставляют набор свойств, которые в свою очередь предоставляют общую информацию о происхождении данных и информацию о методах доступа. Свойства:

- **Name (String)** – имя источника данных;
- **URL (String)** – URL источника данных в форме *jdbc:subprotocol:subname* или *sdbc:subprotocol:subname*;
- **Info (Array)** – массив, содержащее пары `PropertyValue` с параметрами связи (обычно, по крайней мере имя пользователя и пароль);
- **User (String)** – имя пользователя;
- **Password (String)** – пользовательский пароль (не сохраняется);
- **IsPasswordRequired (Boolean)** – пароль необходим и в интерактивном режиме требуется от пользователя;
- **IsReadOnly (Boolean)** – разрешает доступ к базе данных в режиме только для чтения;
- **NumberFormatsSupplier (Object)** – объект, содержащий форматы чисел, доступные для базы данных (поддерживает интерфейс `com.sun.star.util.XNumberFormatsSupplier`, обратитесь к разделу “Форматы чисел, дат и текста” на странице 97);
- **TableFilter (Array)** – список имен таблиц для отображения;
- **TableTypeFilter (Array)** – список типов таблиц для отображения. Доступные значения - `TABLE`, `VIEW` и `SYSTEM TABLE`;
- **SuppressVersionColumns (Boolean)** – подавляет отображение столбцов, которые используются для управления версиями.

Примечание Источники данных от OpenOffice.org несовместимы 1:1 с источниками данных в ODBC. Примите во внимание, что источник данных ODBC содержит только информацию о происхождении данных, источник данных в OpenOffice.org также включает набор информации о том, как данные отображаются в пределах окон базы данных OpenOffice.org.

Запросы

Предопределенные запросы могут быть назначены на источник данных. OpenOffice.org записывает команды запросов SQL так, чтобы они были всегда доступны. Запросы используются для упрощения работы с базами данных, потому что они могут быть открыты с помощью простого щелчка мышью и также обеспечивают пользователям без знания SQL с параметрами выдачу команд SQL.

Объект, который поддерживает сервис `com.sun.star.sdb.QueryDefinition` скрыт позади запроса. К запросам получают доступ посредством метода `QueryDefinitions` источника данных.

Следующий пример отображает список установленных имен запросов источника данных в окне сообщения.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Клиенты")
QueryDefinitions = DataSource.getQueryDefinitions()

For I = 0 To QueryDefinitions.Count() - 1
  QueryDefinition = QueryDefinitions(I)
  MsgBox QueryDefinition.Name
Next I
```

В дополнение к свойству Имя, используемому в примере, сервис `com.sun.star.sdb.QueryDefinition` предоставляет целый набор других свойств. Вот они:

- **Name (String)** – имя запроса;
- **Command (String)** – команда SQL (обычно команда `SELECT`);
- **UpdateTableName (String)** – для запросов, которые основаны на нескольких таблицах: имя таблицы, в которой модификация значений является возможной;
- **UpdateCatalogName (String)** – имя каталогов обновления таблиц;
- **UpdateSchemaName (String)** – название схем обновления таблиц.

Следующий пример показывает, как объект запрос может быть создан программно-управляемым способом и может быть назначен источнику данных.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Клиенты")
QueryDefinitions = DataSource.getQueryDefinitions()

QueryDefinition = createUnoService("com.sun.star.sdb.QueryDefinition")
QueryDefinition.Command = "SELECT * FROM Customer"

QueryDefinitions.insertByName("NewQuery", QueryDefinition)
```

Объект запроса сначала создается с использованием вызова `createUnoService`, затем инициализируется, и затем вставляется в объект `QueryDefinitions` посредством `insertByName`.

Связи с формами баз данных

Чтобы упростить работу с источниками данных, OpenOffice.org предоставляет параметры для связи источников данных с формами баз данных. Связи доступны через метод `getBookmarks()`. Он возвращает именованный контейнер (`com.sun.star.sdb.DefinitionContainer`), который содержит все связи источника данных. К закладкам можно получить доступ через `Name` или `Index`.

Следующий пример определяет URL закладки *MyBookmark*.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Bookmarks As Object
Dim URL As String

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Клиенты")
Bookmarks = DataSource.Bookmarks()

URL = Bookmarks.getByName("MyBookmark")
MsgBox URL
```

Доступ к базам данных

Соединение с базой данных необходимо для доступа к базе данных. Это канал передачи, который разрешает прямую коммуникацию с базой данных. В отличие от источников данных, представленных в предыдущем разделе, соединение с базой данных должно вновь устанавливаться каждый раз, когда программа повторно запускается.

OpenOffice.org обеспечивает различные способы установки соединения с базой данных. Вот объяснение метода, основанного на существующем источнике данных.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Клиенты")

If Not DataSource.IsPasswordRequired Then
  Connection = DataSource.GetConnection("", "")
Else
  InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
  Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If
```

Код, используемый в примере сначала проверяет, является ли база данных защищенной паролем. Если нет, он создает требуемое соединение с базой данных, используя вызов `GetConnection`. Две пустых строки в команде обозначают имя пользователя и пароль.

Если база данных защищена паролем, пример создает `InteractionHandler` и открывает соединение с базой данных, используя метод `ConnectWithCompletion`.`InteractionHandler` гарантирует, что OpenOffice.org запрашивает у пользователя необходимые регистрационные данные.

Повторение таблиц

К таблице обычно получают доступ в OpenOffice.org через объект `ResultSet`. `ResultSet` - тип маркера, который указывает текущий набор данных в пределах объема результатов, полученных с использованием команды SELECT.

Пример показывает, как ResultSet может использоваться, для запроса значений из таблицы базы данных.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object
Dim Statement As Object
Dim ResultSet As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Клиенты")

If Not DataSource.IsPasswordRequired Then
  Connection = DataSource.GetConnection("", "")
Else
  InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
  Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If

Statement = Connection.createStatement()
ResultSet = Statement.executeQuery("SELECT CustomerNumber FROM Customer")

If Not IsNull(ResultSet) Then
  While ResultSet.next
    MsgBox ResultSet.getString(1)
  Wend
End If
```

Как только соединение с базой данных установлено, код, используемый в первом примере использует запрос Connection.createObject для создания объекта Statement. Этот объект Statement используя вызов executeQuery возвращает фактический ResultSet. Программа теперь проверяет, существует ли ResultSet фактически и перебирает записи данных, используя цикл. Возвращая требуемые значения (в примере, это поле CustomerNumber) ResultSet с использованием метода getString, в соответствии с параметром 1 решает, что запрос имеет отношение к значениям первого столбца.

Примечание Объект ResultSet от SDBC сопоставим с объектом Recordset от DAO и ADO, так как он также обеспечивает повторяющийся доступ к базе данных.

Примечание К базе данных фактически получают доступ в OpenOffice.org 2 через объект ResultSet. Он отражает содержание таблицы или результат команды SQL-SELECT. В прошлом объект ResultSet обеспечивал резидентные методы в объекте Application для навигации в пределах данных, например, DataNextRecord).

Зависимые от типа методы для извлечения значений

Как можно заметить в примере из предыдущего раздела, OpenOffice.org предоставляет метод getString для получения доступа к содержимому таблицы. Метод возвращает результат в форме строки. Следующее get методы доступны:

- getByte() – поддерживает типы данных SQL для чисел, символов и строк;
- getShort() – поддерживает типы данных SQL для чисел, символов и строк;
- getInt() – поддерживает типы данных SQL для чисел, символов и строк;
- getLong() – поддерживает типы данных SQL для чисел, символов и строк;

- `getFloat()` – поддерживает типы данных SQL для чисел, символов и строк;
- `getDouble()` – поддерживает типы данных SQL для чисел, символов и строк;
- `getBoolean()` – поддерживает типы данных SQL для чисел, символов и строк;
- `getString()` – поддерживает все типы данных SQL;
- `getBytes()` – поддерживает типы данных SQL для двоичных значений;
- `getDate()` – поддерживает типы данных SQL для чисел, строк, дат и временных отметок;
- `getTime()` – поддерживает типы данных SQL для чисел, строк, дат и временных отметок;
- `getTimestamp()` – поддерживает типы данных SQL для чисел, строк, дат и временных отметок;
- `getCharacterStream()` – поддерживает типы данных SQL для чисел, строк и двоичных значений;
- `getUnicodeStream()` – поддерживает типы данных SQL для чисел, строк и двоичных значений;
- `getBinaryStream()` – двоичные значения;
- `getObject()` – поддерживает все типы данных SQL.

Во всех случаях, номер столбца должен быть передан как параметр, чье значение запрашивается.

Варианты ResultSet

Доступ к базам данных – зачастую вопрос критичный к скорости. OpenOffice.org поэтому обеспечивает несколько способов оптимизации работы `ResultSets` и таким образом управляет скоростью доступа. Чем больше функций, которые предоставляет ResultSet, тем более сложным является его выполнение и поэтому медленнее функции.

Простой `ResultSet`, такой как был представлен в разделе “Повторение таблиц”, обеспечивает минимальные возможности доступных функций. Он позволяет только повторно быть примененным в будущем, и запрашивать значения. Более широкие навигационные параметры, такие как возможности изменения значений, поэтому не включены.

Объект `Statement`, используемый для создания `ResultSet` предоставляет некоторые свойства, которые позволяют воздействовать на функции `ResultSet`:

- **ResultSetConcurrency (const)** – подробности относительно возможности изменения данных (подробности в соответствии с `com.sun.star.sdbc.ResultSetConcurrency`);
- **ResultSetType (const)** – подробности относительно типа `ResultSets` (подробности в соответствии с `com.sun.star.sdbc.ResultSetType`).

Значения, определенные в `com.sun.star.sdbc.ResultSetConcurrency`:

- **UPDATABLE** – `ResultSet` разрешает изменение значений;
- **READ_ONLY** – `ResultSet` не разрешает изменений.

Группа констант `com.sun.star.sdbc.ResultSetConcurrency` обеспечивает следующие подробности:

- **FORWARD_ONLY** – `ResultSet` разрешает только навигацию вперед;
- **SCROLL_INSENSITIVE** – `ResultSet` разрешает любой тип навигации, однако, изменения в оригинальных данных не видны;
- **SCROLL_SENSITIVE** – `ResultSet` разрешает любой тип навигации, изменения в оригинальных данных воздействует на `ResultSet`.

Примечание `ResultSet`, содержащий свойства `READ_ONLY` и `SCROLL_INSENSITIVE` соответствует набору записей типа `Snapshot` в ADO и DAO.

При использовании свойств `UPDATEABLE` и `SCROLL_SENSITIVE`, возможности функции `ResultSet` сопоставимы с набором записей типа `Dynaset` в ADO и DAO.

Методы для навигации в ResultSets

Если `ResultSet` типа `SCROLL_INSENSITIVE` или `SCROLL_SENSITIVE`, он поддерживает целый набор методов для навигации в подготовленных данных. Центральные методы:

- **next()** – перемещение к следующей записи данных;
- **previous()** – перемещение к предыдущей записи данных;
- **first()** – перемещение к первой записи данных;
- **last()** – перемещение к последней записи данных;
- **beforeFirst()** – перемещение к позиции перед первой записью данных;
- **afterLast()** – перемещение к позиции после последней записи данных.

Все методы возвращают логический параметр, который определяет, было ли перемещение успешным.

Чтобы определить текущее положение курсора, предоставляются следующие методы проверки и все они возвращают логическое значение:

- **isBeforeFirst()** – `ResultSet` перед первой записью данных;
- **isAfterLast()** – `ResultSet` после последней записи данных;
- **isFirst()** – `ResultSet` первая запись данных;
- **isLast()** – `ResultSet` - последняя запись данных.

Изменение записей данных

Если `ResultSet` был создан со значением `ResultSetConcurrency = UPDATEABLE`, то его содержание может быть отредактировано. Это применимо только, пока команда SQL позволяет перезаписывать данные в базу данных (зависит от принципа). Это, например, не возможно со сложными командами SQL со связанными столбцами или накопительными значениями.

Объект `ResultSet` предоставляет методы `Update` для изменения значений, которые структурированы таким же образом как методы `get` для извлечения значений. Метод `updateString`, например, позволяет строке быть записанной.

После модификации, значения должны быть переданы в базу данных с использованием метода `updateRow()`. Вызов должен иметь место перед следующей командой перемещения, иначе значения будут потеряны.

Если произошла ошибка во время модификации, модификация может быть отменена, применением метода `cancelRowUpdates()`. Этот вызов доступен только при условии, что данные не могут быть перезаписаны в базу данных с использованием `updateRow()`.

Глава 11
Диалоги

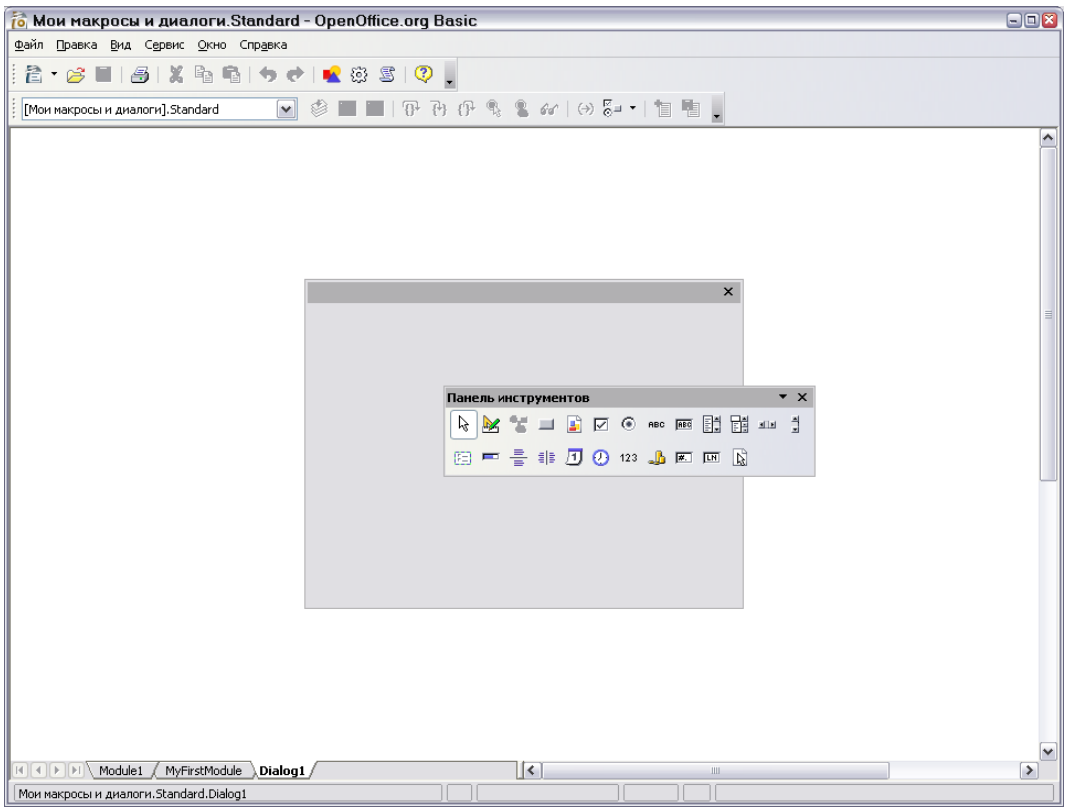
Вы можете добавлять сделанные диалоговые окна и формы к документам OpenOffice.org. Они в свою очередь могут быть связаны с макросами OOo Basic, что значительно расширяет диапазон использования OOo Basic. Диалоги, например, могут отображать информацию из базы данных или вести пользователей через постепенный процесс создания нового документа в форме Автопилота.

Работа с диалогами

Диалоги OOo Basic состоят из окна диалога, которое может содержать текстовые поля, списки, поля со списками, кнопки с зависимой фиксацией и другие элементы управления.

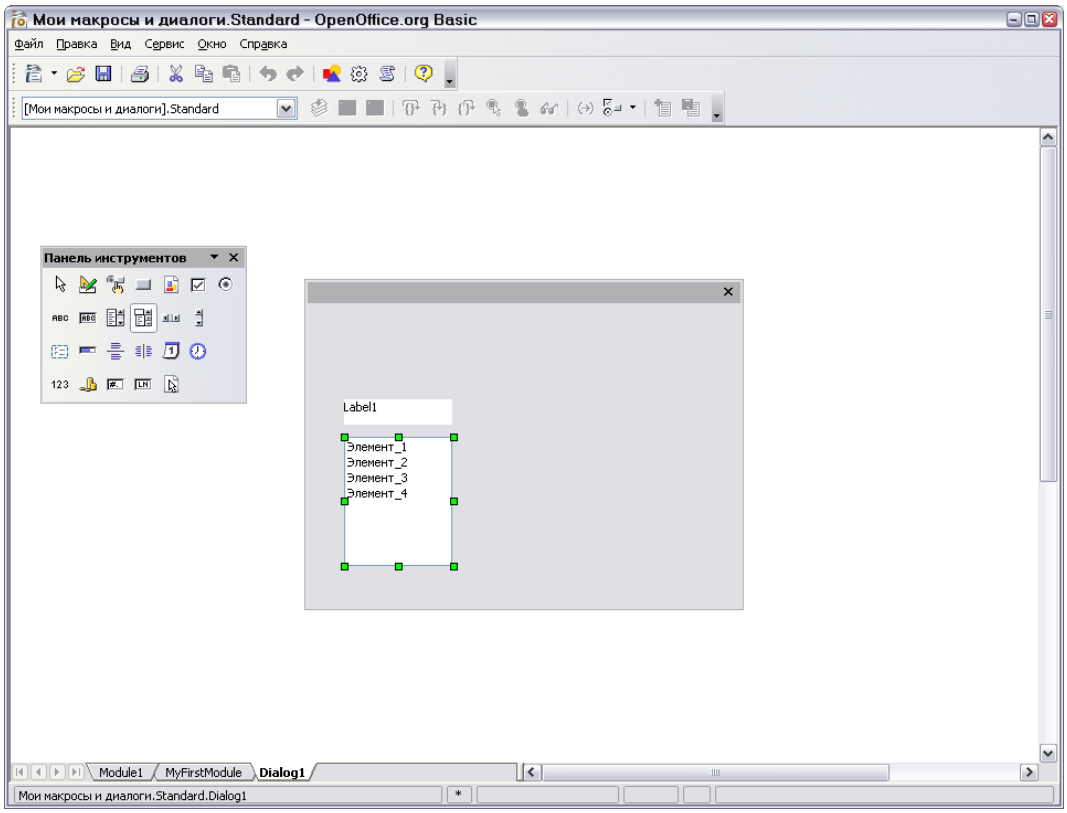
Создание диалогов

Вы можете создавать и интегрировать диалоги, используя редактора диалогов OpenOffice.org, который Вы можете использовать таким же образом как OpenOffice.org Draw:



По существу, Вы переносите элементы управления, которые Вы хотите из панели инструментов (справа) в область диалога, где Вы можете определить их положение и размер.

Пример показывает диалог, который содержит надпись и поле списка.



Вы можете открыть диалог с использованием следующего кода:

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` создает объект по имени `Dlg`, который ссылается на связанный диалог. Прежде, чем Вы можете создать диалог, Вы должны гарантировать, что библиотека, которую он использует (в этом примере, библиотека `Standard`) загружена. В противном случае метод `LoadLibrary` выполняет эту задачу.

Как только объект диалог `Dlg` проинициализировался, Вы можете использовать метод `Execute` для отображения диалога. Диалоги такие, как этот описываются как модальные, потому что они не разрешают никакого другого действия программы, пока они не закрыты. В то время как этот диалог открыт, программа остается в запросе `Execute`.

Метод `dispose` в конце кода освобождает ресурсы, используемые диалогом однажды при завершении программы.

Закрытие Диалогов

Закрытие кнопками Ок или Отмена

Если диалог содержит кнопки OK или Отмена, диалог автоматически закрывается, когда Вы

нажимаете одну из этих кнопок. Более подробная информации о работе с этими кнопками обсуждается в разделе “Элементы управления диалога подробно” этой главы.

Если Вы закрываете диалог, нажимая кнопку **OK**, метод `Execute` возвращает значение 1, в противном случае возвращается значение 0.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
  Case 1
    MsgBox "нажата Ok"
  Case 0
    MsgBox "нажата Отмена"
End Select
```

Закрытие кнопкой Закрыть в заголовке окна

Если Вы хотите, Вы можете закрыть диалог, нажав кнопку закрыть в заголовке окна диалога. В этом случае, метод `Execute` диалога возвращает значение 0, то же самое как тогда, когда Вы нажимаете кнопку Отмена.

Закрытие явным запросом программы

Вы можете также закрыть открытое окно диалога методом `endExecute`:

```
Dlg.endExecute()
```

Доступ к отдельным элементам управления

Диалог может содержать любое количество элементов управления. Вы можете получить доступ к этим элементам через метод `getControl`, который возвращает имя элемента управления.

```
Dim Ctl As Object

Ctl = Dlg.getControl("MyButton")
Ctl.Label = "Новая надпись"
```

Этот код определяет объект для элемента управления `MyButton` и затем инициализирует объектную переменную `Ctl` ссылкой на элемент. Наконец код устанавливает свойство `Label` элемента управления в значение `Новая надпись`.

Примечание	OOo Basic различает заглавные и строчные символы в именах элементов управления.

Работа с *моделью* диалогов и элементов управления

Разделение между видимыми элементами программы (*Вид*) и данными или документами позади них (*Модель*) происходит во многих местах в OpenOffice.org API. В дополнение к методам и свойствам элементов управления, и диалог и объекты элементов управления имеют подчиненный объект `Model`. Этот объект позволяет Вам получить непосредственный доступ к содержимому диалога или элемента управления.

В диалогах, различие между данными и описанием не всегда столь же ясно как в других областях OpenOffice.org API. Элементы API доступны и через Вид и через Модель.

Свойство `Model` обеспечивает программно-управляемый доступ к модели диалога и объектам элементов управления.

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

Этот пример отключает кнопку `cmdNext` в диалоге `Dlg` при помощи объекта модели `cmdNext`.

Свойства

Имя и заголовок

Каждый элемент управления имеет свое собственное имя, которое может быть запрошено с использованием следующего свойства модели:

- **Model.Name (String)** – имя элемента управления.

Вы можете определить заголовок, который появляется в заголовке диалога через следующее свойство модели:

- **Model.Title (String)** – заголовок диалога (применяется только к диалогам).

Положение и Размер

Вы можете запросить размер и положение элемента управления, используя следующие свойства объекта модель:

- **Model.Height (long)** – высота элемента управления (в единицах ma);
- **Model.Width (long)** – ширина элемента управления (в единицах ma);
- **Model.PositionX (long)** – координата X элемента управления, измеренная от левого внутреннего края диалога (в единицах ma);
- **Model.PositionY (long)** – координата Y элемента управления, измеренная от верхнего внутреннего края диалога (в единицах ma).

Чтобы гарантировать независимость от платформы для внешнего вида диалогов, OpenOffice.org использует внутренную единицу *Map AppFont* (*ma*) для определения положения и размера в пределах диалогов. Единица ma определена как одна восьмая средней высоты символа системного шрифта, определенного операционной системой и одной четверти его ширины. При использовании единицы ma, OpenOffice.org гарантирует, что диалог выглядит одинаково на различных системах при различных параметрах настройки системы.

Если Вы хотите изменить размер или положение элементов управления во время выполнения, определите полный размер диалога и регулируйте значения для элементов управления в соответствующем отношении частей.

Примечание Map AppFont (ma) заменяет единицу Twips, чтобы достигнуть лучшей независимости от платформы.

Фокус и последовательность обхода

Вы можете перемещаться между элементами управления в любом диалоге, нажимая клавишу Tab. Следующие свойства доступны в этом контексте в модели элементов управления:

- **Model.Enabled (Boolean)** – активизирует элемент управления;
- **Model.Tabstop (Boolean)** – позволяет элементу управления быть достижимым через клавишу Tab;
- **Model.TabIndex (Long)** – положение элемента управления в последовательности обхода.

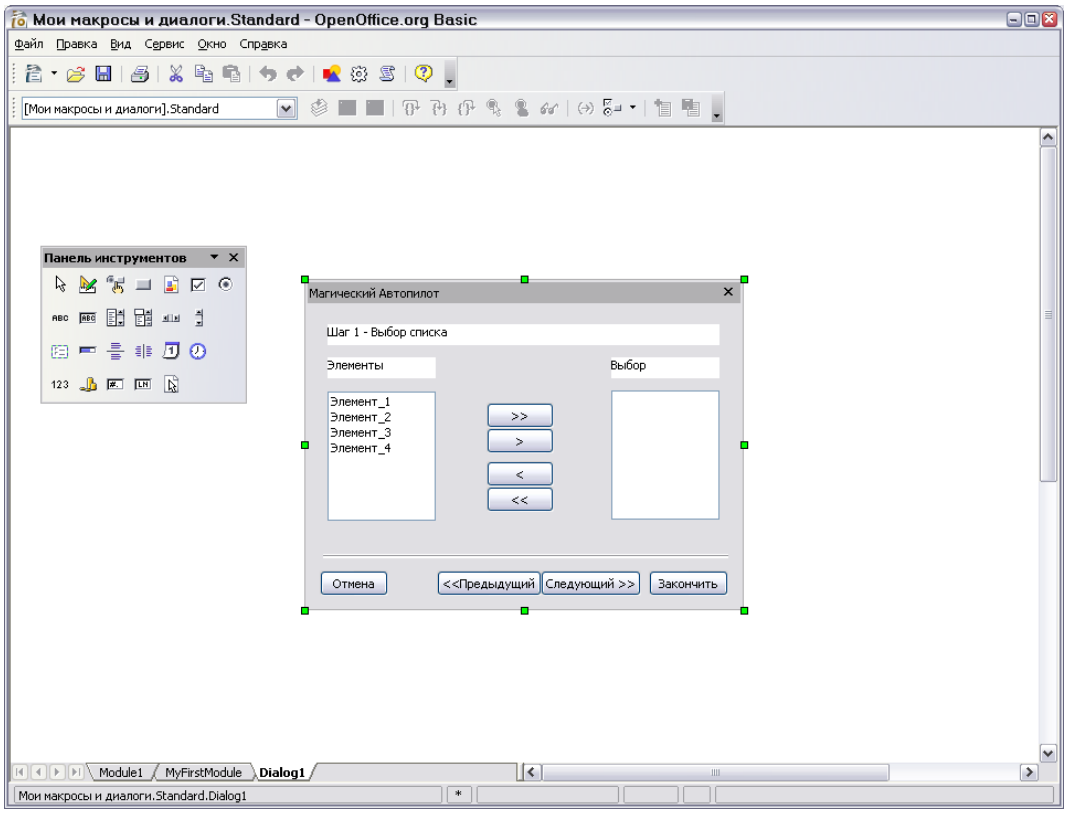
Наконец, элемент управления предоставляет метод `getFocus`, который гарантирует, что указанный элемент управления получает фокус:

- **getFocus** – элемент управления получает фокус (только для диалогов).

Многостраничные диалоги

Диалог в OpenOffice.org может иметь больше чем одну страницу. Свойство `Step` диалога определяет текущую страницу диалога, тогда как свойство `Step` для элемента управления определяет страницу, на которой элемент управления должен отображаться.

Значение `Step` равное 0 - специальный случай. Если Вы устанавливаете это значение в ноль для диалога, все элементы управления видимы независимо от их значения `Step`. Точно так же, если Вы устанавливаете это значение в ноль для элемента управления, элемент отображается на всех страницах в диалоге.



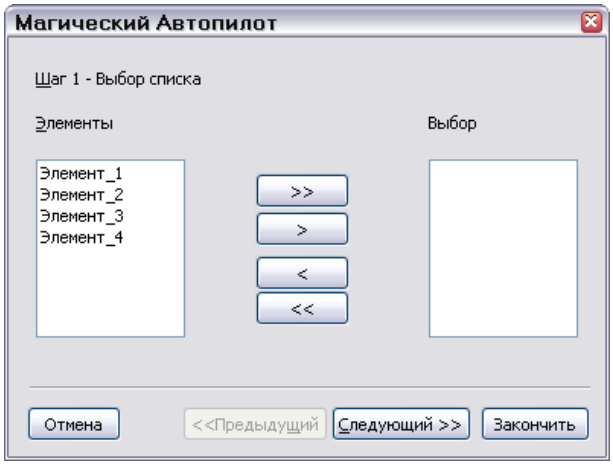
В предыдущем примере, Вы можете также присвоить значению Step 0 для разделительной линии, а так же кнопкам Отмена, Предыдущий, Следующий, и Закончить и отображать эти

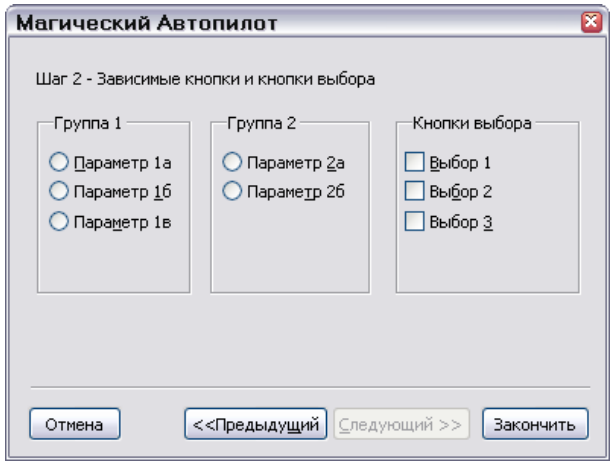
элементы на всех страницах. Вы можете также назначить элементы на отдельную страницу (например страница 1).

Следующий код программы показывает, как значение `Step` может быть увеличено или уменьшено в обработчиках событий кнопок `Следующий` и `Предыдущий` и изменен статус кнопок.

```
Sub cmdNext_Initiated
  Dim cmdNext As Object
  Dim cmdPrev As Object

  cmdPrev = Dlg.getControl("cmdPrev")
  cmdNext = Dlg.getControl("cmdNext")

  cmdPrev.Model.Enabled = True
  cmdNext.Model.Enabled = False

  Dlg.Model.Step = Dlg.Model.Step + 1
End Sub

Sub cmdPrev_Initiated
  Dim cmdNext As Object
  Dim cmdPrev As Object

  cmdPrev = Dlg.getControl("cmdPrev")
  cmdNext = Dlg.getControl("cmdNext")

  cmdPrev.Model.Enabled = False
  cmdNext.Model.Enabled = True

  Dlg.Model.Step = Dlg.Model.Step - 1
End Sub
```

Переменная `Dlg`, которая ссылается на открытый диалог, должна быть включена как глобальная, чтобы сделать этот пример возможным. Диалог тогда изменяет свой внешний вид следующим образом:





События

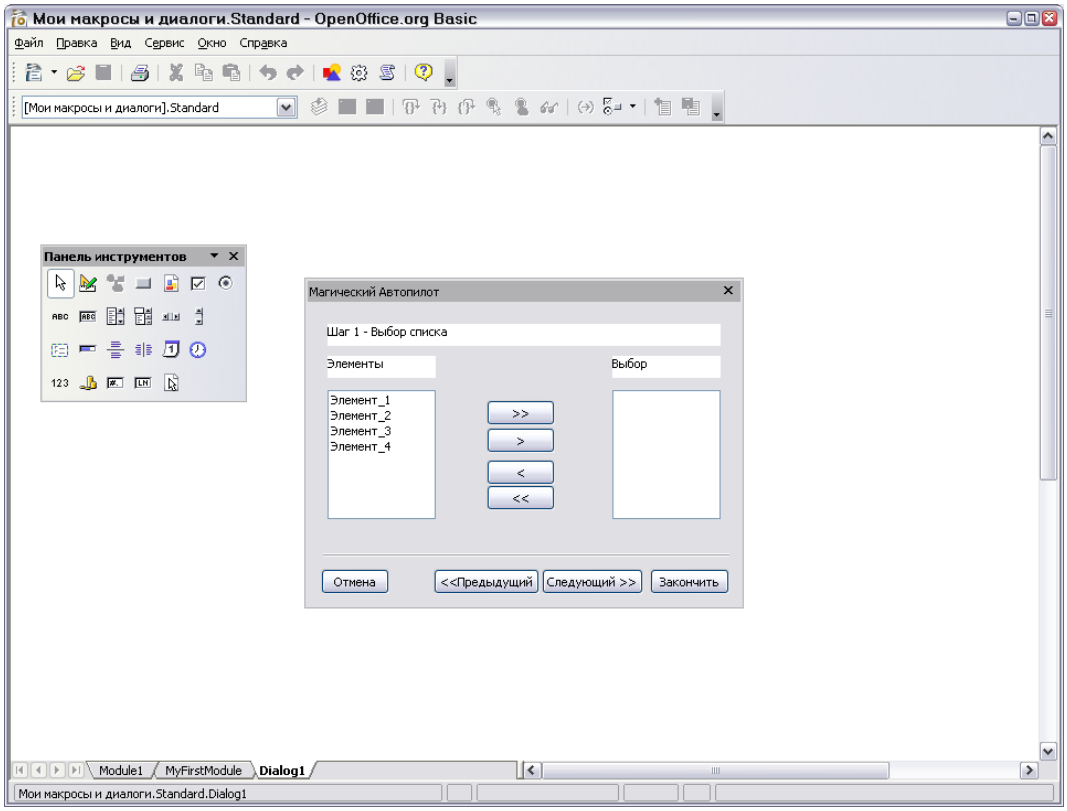
Диалоги и формы OpenOffice.org основаны на ориентируемую на события модель программировании, где Вы можете назначить обработчики событий элементам управления. Обработчик события выполняет предопределенную процедуру когда происходит определенное действие. Вы можете редактировать документы или открывать базы данных с помощью обработки событий так же как получать доступ к другим элементам управления.

Элементы управления OpenOffice.org опознают различные типы событий, которые могут быть вызваны в различных ситуациях. Эти типы событий могут быть разделены на четыре

группы:

- **Управление от мыши**: События, которые соответствуют действиям мыши (например, просто движениям мыши или нажатиям в определенном месте экрана);
- **Управление от клавиатуры**: События, которые вызваны нажатиями клавиатуры;
- **Изменения фокуса**: События, которые OpenOffice.org выполняет, когда элементы управления получают или теряют фокус;
- **Управление определенными для элемента событиями**: События, которые происходят только относительно определенных элементов управления.

Когда Вы работаете с событиями, гарантируйте, что Вы создаете взаимодействующий диалог в среде разработки OpenOffice.org и что он содержит необходимые элементы управления или документы (если Ваши события применяются к форме).



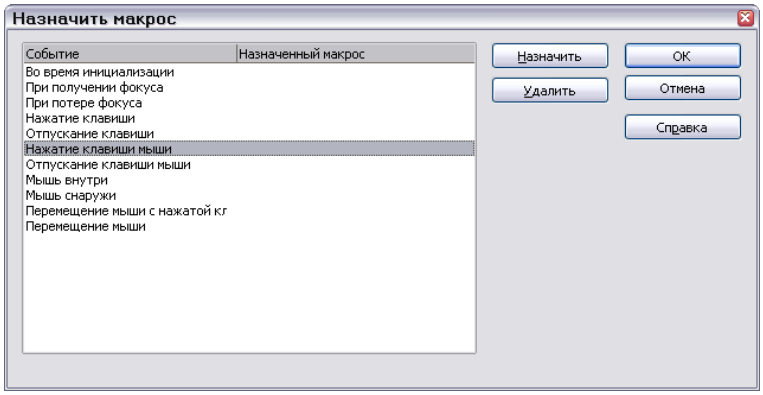
Предыдущее изображение показывает среду разработки OOo Basic с окном диалога, которое содержит два списка. Вы можете перемещать данные из одного списка в другой используя кнопки между списками.

Если Вы хотите отобразить расположение на экране, то Вы должны создать связанные процедуры OOo Basic так, чтобы их можно были вызывать обработчиками событий. Даже при том, что Вы можете использовать эти процедуры в любом модуле, лучше ограничивать их использование двумя модулями. Чтобы сделать ваш код легче читаемым, Вы должны назначить значащие имена этим процедурам. Переход непосредственно к общей процедуре программы из макроса может привести к неясному коду. Вместо того чтобы упростить поддержку кода и поиск неисправностей, Вы должны создать другую процедуру, которая будет служить точкой входа для обработки события – даже если она выполняет только единственный вызов целевой процедуры.

Код в следующем примере перемещает элемент из левого списка диалога в правый список.

```
Sub cmdSelect_Initiated
  Dim objList As Object

  lstEntries = Dlg.getControl("lstEntries")
  lstSelection = Dlg.getControl("lstSelection")

  If lstEntries.SelectedItem > 0 Then
    lstSelection.AddItem(lstEntries.SelectedItem, 0)
    lstEntries.removeItems(lstEntries.SelectItemPos, 1)
  Else
    Beep
  End If
End Sub
```

Если эта процедура была создана в OOo Basic, Вы можете назначить ее на требуемое событие, используя окно свойств редактора диалога.



Диалог назначения перечисляет все события OOo Basic. Чтобы назначать событию обработчик, выберите событие, и затем нажмите Назначить.

Параметры

Возникновение специфического события – не всегда достаточно для соответствующего ответа. Может требоваться дополнительная информация. Например, чтобы обработать нажатие кнопки мыши, Вы, возможно, нуждаетесь в позиции экрана, где была нажата кнопка мыши.

В OOo Basic Вы можете использовать параметры объекта, чтобы предоставить дополнительную информацию о событии для процедуры, например:

```
Sub ProcessEvent(Event As Object)

End Sub
```

Подробность, с которой объект `Event` структурирован и его свойства, зависит от типа события, которое инициирует вызов процедуры. Следующие разделы подробно описывают типы событий.

Независимо от типа события, все объекты обеспечивают доступ к соответствующему элементу управления и его модели. Элемент управления может быть достигнут с использованием

```
Event.Source
```

а его используемая модель

```
Event.Source.Model
```

Вы можете использовать эти свойства для инициирования события в пределах обработчика события.

События Мыши

OOo Basic распознает следующие события мыши:

- **перемещение мыши** – пользователь перемещает мышь;
- **перемещение мыши с нажатой кнопкой** – пользователь тянет мышь, удерживая в нужном состоянии кнопку;
- **нажатие кнопки мыши** – пользователь нажимает кнопку мыши;
- **отпускание кнопки мыши** – пользователь отпускает кнопку мыши;
- **мышь снаружи** – пользователь перемещает мышь наружу текущего окна.

Структура связанных объектов события определена в структуре `com.sun.star.awt.MouseEvent`, которая предоставляет следующую информацию:

- **Buttons (short)** – нажатая кнопка (одна или более констант в соответствии с `com.sun.star.awt.MouseButton`);
- **X (long)** – координата X мыши, измеренная в пикселах от верхнего левого угла элемента управления;
- **Y (long)** – координата Y мыши, измеренная в пикселах от верхнего левого угла элемента управления;
- **ClickCount (long)** – число щелчков связанное с событием мыши (если OpenOffice.org может ответить достаточно быстро, `ClickCount` – также 1 для двойного щелчка, потому что инициируется только отдельное событие).

Константы, определенные в `com.sun.star.awt.MouseButton` для кнопок мыши:

- **LEFT** – левая кнопка мыши;
- **RIGHT –** правая кнопка мыши;
- **MIDDLE** – средняя кнопка мыши.

Следующий примера выводит положение мыши, а также кнопку мыши, которая была нажата:

```
Sub MouseUp(Event As Object)
  Dim Msg As String
  Msg = "Keys: "
  If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
    Msg = Msg & "LEFT "
  End If
  If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
    Msg = Msg & "RIGHT "
  End If
  If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
    Msg = Msg & "MIDDLE "
  End If

  Msg = Msg & Chr(13) & "Position: "
  Msg = Msg & Event.X & "/" & Event.Y
  MsgBox Msg
End Sub
```

Примечание События VBA `Click` и `Doubleclick` не доступны в OOo Basic. Вместо этого используйте событие `MouseUp` OOo Basic для события `click` и имитируйте событие `Doubleclick`, изменяя прикладную логику .

События Клавиатуры

Следующие события клавиатуры доступны в OOo Basic:

- **нажатие клавиши** – пользователь нажимает клавишу;
- **отпускание клавиши** – пользователь отпускает клавишу.

Оба события касаются *логических* действий, а не относятся к *физическим* действиям. Если пользователь нажимает несколько клавиш для получения одного символа (например, чтобы добавить ударение к символу), то OOo Basic создает только одно событие.

Одиночное нажатие клавиш модификации, таких как клавиша SHIFT или клавиша Alt не создает самостоятельного события.

Информация о нажатой клавише предоставляется объектом `Event` OOo Basic передаваемым процедуре обработки события. Он содержит следующие свойства:

- **KeyCode (short)** – код нажатой клавиши (значение по умолчанию в соответствии с `com.sun.star.awt.Key`);
- **KeyChar (String)** – символ, который введен (берется в соответствии с клавишами модификации).

Следующий пример использует свойство `KeyCode`, чтобы установить, что были нажаты клавиша Enter, клавиша Tab, или одна из других клавиш управления. Если одна из этих клавиш была нажата, возвращается название клавиши, иначе возвращается символ, который был напечатан:

```
Sub KeyPressed(Event As Object)
  Dim Msg As String
  Select Case Event.KeyCode
    Case com.sun.star.awt.Key.RETURN
      Msg = "Return pressed"
    Case com.sun.star.awt.Key.TAB
      Msg = "Tab pressed"
    Case com.sun.star.awt.Key.DELETE
      Msg = "Delete pressed"
    Case com.sun.star.awt.Key.ESCAPE
      Msg = "Escape pressed"
    Case com.sun.star.awt.Key.DOWN
      Msg = "Down pressed"
    Case com.sun.star.awt.Key.UP
      Msg = "Up pressed"
    Case com.sun.star.awt.Key.LEFT
      Msg = "Left pressed"
    Case com.sun.star.awt.Key.RIGHT
      Msg = "Right pressed"
    Case Else
      Msg = "Character " & Event.KeyChar & " entered"
  End Select
  MsgBox Msg
End Sub
```

Информация о других клавиатурных константах может быть найдена в Справочном руководстве по API под группой констант `com.sun.star.awt.Key`.

События фокуса

События фокуса показывают, получает ли элемент управления или теряет фокус. Вы можете использовать эти события для, например, определения, закончил ли пользователь обрабатывать элемент управления, чтобы Вы могли обновить другие элементы диалога. Доступны следующие события фокуса:

- **При получении фокуса** – элемент управления получает фокус;
- **При потере фокуса** - элемент управления теряет фокус.

Объекты `Event` для событий фокуса структурированы следующим образом:

- **FocusFlags (short)** – причина изменения фокуса (по умолчанию значение в соответствии с `com.sun.star.awt.FocusChangeReason`);
- **NextFocus (Object)** – объект, который получает фокус (только для случая потери фокуса);
- **Temporary (Boolean)** – фокус временно потерян.

События определяемые элементами управления

В дополнение к предыдущим событиям, которые поддерживаются всеми элементами управления, есть также некоторые события определяемые элементами управления, которые определены только для определенных элементов управления. Самые важные из этих событий:

- **При изменении элемента** – значение элемента управления изменено;
- **Состояние изменено** – состояние элемента управления изменено;
- **Текст изменен** – текст элемента управления изменен;
- **Во время инициализации** - действие, которое может быть выполнено, когда элемент управления вызывается (например, кнопка нажата).

Когда Вы работаете с событиями, отметьте, что некоторые события, такие как событие **Во время инициализации**, могут быть запущены каждый раз, когда Вы нажимаете мышью на некоторых элементах управления (например, на кнопках выбора). Никакое действие не выполняется, для проверки, изменился ли статус элемента управления фактически. Чтобы избегать таких “слепых событий”, сохраните старое значение элемента управления в глобальной переменной, и затем проверьте, чтобы видеть, изменилось ли значение, когда событие выполняется.

Свойства для события **Состояние изменено**:

- **Selected (long)** – в настоящее время выбранный элемент;
- **Highlighted (long)** – в настоящее время выделенный элемент;
- **ItemId (long)** – идентификатор элемента.

Элементы управления диалогов подробно

OOo Basic предоставляет набор элементов управления, которые могут быть разделены на следующие группы:

Поля ввода:

- текстовые поля;
- поля ввода даты;
- поля ввода времени;

- числовые поля;
- поля ввода валюты;
- поля, принимающие любой формат.

Кнопки:

- стандартные кнопки;
- переключатели;
- флажки.

Списки выбора:

- список;
- поле со списком.

Прочие элементы управления:

- полосы прокрутки (горизонтальная и вертикальная);
- группа;
- индикатор выполнения;
- разделительные линии (горизонтальная и вертикальная);
- графический элемент;
- поле выбора файла.

Самые важные из этих элементов управления представлены ниже.

Кнопки

Кнопка выполняет действие, когда Вы нажимаете на нее.

Самый простой сценарий для кнопки, вызвать событие `Во время инициализации`, когда на нее нажимает пользователь. Вы можете также связать другое действие с кнопкой для открытого диалога, используя свойство `PushButtonType`. Когда Вы нажимаете не кнопку, для которой установили эту свойство в значение 0, диалог остается незатронутым. Если Вы нажимаете кнопку, для которой установили эту свойство в значение 1, диалог закрывается, и метод диалога `Execute` возвращает значение 1 (последовательность диалога была завершена правильно). Если `PushButtonType` имеет значение 2, диалог закрывается, и метод диалога `Execute` возвращает значение 0 (диалог закрывается).

Далее приведены все свойства, которые являются доступными через модель кнопки:

- **Model.BackgroundColor (long)** – цвет фона;
- **Model.DefaultButton (Boolean)** – кнопка используется как значение по умолчанию и отвечает на клавишу Enter, если она не имеет фокуса;
- **Model.FontDescriptor (struct)** – структура, которая определяет детали используемого шрифта (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **Model.ImageAlign (short)** – определяет выравнивание изображения на кнопке, (0 по левому краю, 1 – по верху, 2 – по правому краю и 3 – по низу);

- **Model.ImageURL (String)** – содержит путь к графическому файлу, которое отображается на кнопке (поддерживаются все стандартные графические форматы, такие как *.gif*, *.jpg*, *.tif*, *.wmf* и *.bmp*), если размер изображения превышает размер кнопки, изображение не масштабируется автоматически, а отключается;
- **ImagePosition (short)** – определяет положение изображения, если таковое имеются, относительно текста (если это свойство присутствует, оно заменяет свойство `ImageAlign`, по умолчанию значение в соответствии со списком `com.sun.star.awt.ImagePosition`);
- **Model.Label (String)** – надпись, которая отображается на кнопке;
- **Model.Printable (Boolean)** – элемент управления может быть напечатан;
- **Model.TextColor (Long)** – цвет текста элемента управления;
- **Model.HelpText (String)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления;
- **Model.PushButtonType (short)** – действие, которое связано с кнопкой (0: отсутствие действия, 1: Ок, 2: Отмена).

Переключатели

Эти кнопки используются в группах и позволяют Вам выбирать один из нескольких вариантов. Когда Вы выбираете переключатель, все другие переключатели в группе дезактивируются. Это гарантирует, что в любой момент времени установлен только один переключатель.

Элемент управления переключатель предоставляет два свойства:

- **State (Boolean)** – состояние переключателя;
- **Label (String)** – надпись, которая отображается рядом с переключателем.

Вы можете также использовать следующие свойства из модели переключателей:

- **Model.FontDescriptor (struct)** – структура, которая определяет детали шрифта, который используется (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **Model.Label (String)** – надпись, которая отображается на элементе управления;
- **Model.Printable (Boolean)** – элемент управления может быть напечатан;
- **Model.State (Short)** – если это свойство равно 1, переключатель активирован, иначе он дезактивирован;
- **Model.TextColor (Long)** – цвет текста элемента управления;
- **Model.HelpText (String)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Для объединения несколько переключателей в группу, Вы должны поместить их один за другим в последовательность активации без промежутков (свойство `Model.TabIndex`, описанное как Порядок в редакторе диалога). Если последовательность активации прервана другим элементом управления, то OpenOffice.org автоматически начинает новую группу элементов управления, которая может активироваться независимо от первой группы элементов управления.

Примечание	В отличие от VBA, Вы не можете вставить переключатели в группу элементов управления в OOo Basic. Группировка элементов управления в OOo Basic используется только для обеспечения визуального разделения, рисует рамку вокруг элементов управления.

Флажки

Флажки используются для записи значений Да или Нет и в зависимости от режима, они могут принимать два или три состояния. В дополнение к состояниям Да и Нет, флажок может иметь промежуточное состояние, если соответствующий статус Да или Нет имеет более чем одно значение или неясен.

Флажки используются в группах, для отображения различных параметров так, чтобы пользователь мог выбрать один или более параметров. Когда флажок выбран, он отображает контрольную метку. Флажки работают независимо друг от друга, таким образом отличаются от переключателей. Пользователь может выбрать любое количество флажков одновременно.

Флажки предоставляют следующие свойства:

- **State (Short)** – состояние флажка (0: нет, 1: да, 2: промежуточное состояние);
- **Label (String)** – надпись для элемента управления;
- **enableTriState (Boolean)** – в дополнение к активированному и дезактивированному состояниям, Вы можете также использовать промежуточное состояние.

Модель объекта флажок обеспечивает следующие свойства:

- **Model.FontDescriptor (struct)** – структура, которая определяет детали шрифта, который используется (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **Model.Label (String)** – надпись, которая отображается на элементе управления;
- **Model.Printable (Boolean)** – элемент управления может быть напечатан;
- **Model.State (Short)** – состояние флажка (0: нет, 1: да, 2: промежуточное состояние);
- **Model.Tabstop (Boolean)** – элемент управления может быть достигнут при помощи клавиши Tab;
- **Model.TextColor (Long)** – цвет текста элемента управления;
- **Model.HelpText (String)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Текстовые поля

Текстовые поля позволяют пользователям вводить числа и текст. Сервис `com.sun.star.awt.UnoControlEdit`. формирует основание для текстовых полей.

Текстовое поле может содержать одну или более строк и может редактироваться или быть заблокированным для пользовательского ввода. Текстовые поля могут также использоваться как поля ввода специальных денежных и числовых данных, а также как поля экрана для специальных задач. Поскольку эти элементы управления основаны на UNO сервисе `UnoControlEdit`, их программно-управляемая обработка подобна.

Текстовые поля предоставляют следующие свойства:

- **Text (String)** – текущий текст;
- **SelectedText (String)** – выделенный в настоящее время текст;
- **Selection (Struct)** – подробности выделения только для чтения (структура в соответствии с `com.sun.star.awt.Selection`, со свойствами `Min` и `Max`, определяющими начало и конец текущего выделения);
- **MaxTextLen (short)** – максимальное количество символов, которые Вы можете ввести в поле;
- **Editable (Boolean)** – `True` активизирует возможность ввода текста, `False` блокирует возможность ввода (свойство нельзя вызвать непосредственно, а только через `IsEditable`);
- **IsEditable (Boolean)** – содержимое элемента управления может быть изменено, только для чтения.

Текстовые поля предоставляют следующие методы:

- **insertText (Sel, Text)** – вставляет текст, заданный строковой переменной `Text`, в положение, определяемое структурой `Sel`;
- **getSelectedText** – возвращает строку, содержащую выделенный в настоящее время текст;
- **setSelection (Selection)** – устанавливает пользовательское выделение в соответствии с информацией, содержащейся в структуре `Selection`;
- **setEditable (Editable)** – делает текст редактируемым для пользователя или только для чтения в зависимости от логической переменной `Editable`.

Кроме того, следующие свойства предоставляются через связанный объект модели:

- **Model.Align (short)** – выравнивание текста (0: выравнивание по левому краю, 1: выравнивание по центру, 2: выравнивание по правому краю);
- **Model.BackgroundColor (long)** – цвет фона элемента управления;
- **Model.Border (short)** – тип границы (0: нет границы, 1: трехмерная граница, 2: простая граница);
- **Model.EchoChar (short)** – код эхо-символа для полей ввода пароля;
- **Model.FontDescriptor (struct)** – структура с подробностями используемого шрифта (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **Model.HardLineBreaks (Boolean)** – автоматические разрывы строки постоянно

вставляются в текст элемента управления;

- **Model.HScroll (Boolean)** – текст имеет горизонтальную полосу прокрутки;
- **Model.MaxTextLen (Short)** – максимальная длина текста, где 0 соответствует отсутствию ограничения длины;
- **Model.MultiLine (Boolean)** – разрешает ввод в объеме нескольких строк;
- **Model.Printable (Boolean)** – элемент управления может быть напечатан;
- **Model.ReadOnly (Boolean)** – содержимое элемента управления только для чтения;
- **Model.Tabstop (Boolean)** – элемент управления может быть достигнут при помощи клавиши Tab;
- **Model.Text (String)** – текст связанный с элементом управления;
- **Model.TextColor (Long)** – цвет текста элемента управления;
- **Model.VScroll (Boolean)** – текст имеет вертикальную полосу прокрутки;
- **Model.HelpText (String)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Когда текстовое поле получает фокус, по нажатию клавиши **Tab**, отображаемый текст по умолчанию выбран и выделен. Положение курсора по умолчанию в пределах текстового поля справа от существующего текста. Если пользователь начинает вводить, в то время как блок текста выбран, выбранный текст заменяется. В некоторых случаях, пользователь может изменить поведение выделения по умолчанию и установить выделение вручную.

```
Dim sText As String
Dim Sel As New com.sun.star.awt.Selection

' получение элемента управления
txtField = Dlg.getControl("TextField1")

' установка отображаемого текста
sText = "Displayed Text"
txtField.setText(sText)

' задание выделения
Sel.Min = 0
Sel.Max = Len(sText)
txtField.setSelection(Sel)
```

Элемент управления текстовое поле также может использоваться для ввода паролей. Свойство `EchoChar` определяет символ, который отображается в текстовом поле, когда пользователь вводит в пароль. В этом контексте, свойство `MaxTextLen` используется для ограничения числа вводимых символов:

```
txtField = Dlg.getControl("TextField1")
txtField.Model.EchoChar = Asc("*")
txtField.Model.MaxTextLen = 8
```

Пользователь может вводить в текстовое поле любой вид данных, такие как значения чисел и дат. Эти значения всегда сохраняются как строка в свойстве `Text`, таким образом приводя к проблемам при оценке пользовательского ввода. Поэтому, рассмотрите использование поля даты, поля времени, числового поля, поля валюты или форматированного поля вместо текстового поля.

Поля даты

Элемент управления поле даты (сервис `com.sun.star.awt.UnoControlDateField`) расширяет элемент управления текстовое поле и используется для отображения и ввода даты. Дата, отображаемая в поле даты управляется свойством `Date`. Значение даты имеет тип `Long` и должно быть определено в формате `YYYYMMDD`, например, дата 30-ого сентября 2002 устанавливается в следующем формате:

```
oDateField = Dlg.getControl("DateField1")
oDateField.Model.Date = 20020930
```

Текущая дата устанавливается при использовании функций времени выполнения `Date` и `CDateToIso`:

```
oDateField.Model.Date = CDateToIso(Date())
```

Минимальная и максимальная даты, в которые пользователь может ввести, определяются свойствами `DateMin` и `DateMax`. Формат отображения даты определяется свойствами `DateFormat` и `DateShowCentury`, но использование `DateShowCentury` не рекомендуется. Некоторые форматы зависят от параметров настройки системы. Если свойство `StrictFormat` установлено в True, дата, введенная пользователем проверяется во время ввода. Свойство `Dropdown` разрешает использование выпадающего календаря, который пользователь может вызвать для выбора даты.

Объект модели поля дата предоставляет следующие свойства:

- **Model.Date (long)** – определяет дату, отображаемую в элементе управления;
- **Model.DateFormat (short)** – определяет формат отображения даты;
- **Model.DateMax (long)** – определяет максимальную дату, которая может быть введена;
- **Model.DateMin (long)** – определяет минимальную дату, которая может быть введена;
- **Model.DateShowCentury (boolean)** – определяет, отображается ли столетие даты;
- **Model.Dropdown (boolean)** – определяет, имеет ли элемент управления кнопку, вызывающую выпадающий календарь;
- **Model.Repeat (boolean)** – определяет, должна ли мышь вызывать повторяющиеся события, то есть неоднократно вызвать действие, при удержании кнопки в нажатом положении;
- **Model.RepeatDelay (long)** – определяет задержку повторения для мыши, в миллисекундах (Когда пользователь нажимает кнопку мышь в области элемента управления, где это вызывает действие, тогда обычное выполнение элемента управления позволяет неоднократно вызывать это действие, без потребности отпустить кнопку мыши и нажимать ее снова. Задержка между двумя такими событиями определяется этим свойством.);
- **Model.Spin (boolean)** – определяет, что элемент управления имеет кнопки счетчика;
- **Model.StrictFormat (boolean)** – определяет, что дата проверяется во время пользовательского ввода.

Поля времени

Элемент управления поле времени (сервис `com.sun.star.awt.UnoControlDateField`) отображает и осуществляет ввод значения времени. Значение времени устанавливается и

извлекается через свойство `Time`. Значение времени имеет тип `Long` и определяется в формате `HHMMSShh`, где `HH` - часы, `MM` - минуты, `SS` - секунды, и `hh` - сотые секунды. Например, время 15:18:23 устанавливается следующим образом:

```
oTimeField = Dlg.getControl("TimeField1")
oTimeField.Model.Time = 15182300
```

Минимальное и максимальное значение времени, которое может быть введено, задается свойствами `TimeMin` и `TimeMax`. Формат отображения времени определяется свойством `TimeFormat`.

Значение времени проверяется во время ввода, установкой свойства `StrictFormat` в `True`.

Объект модели поля времени предоставляет следующие свойства:

- **Model.Repeat (boolean)** – определяет, должна ли мышь вызывать повторяющиеся события, то есть неоднократно вызвать действие, при удержании кнопки в нажатом положении;
- **Model.RepeatDelay (long)** – определяет задержку повторения для мыши, в миллисекундах (Когда пользователь нажимает кнопку мышь в области элемента управления, где это вызывает действие, тогда обычное выполнение элемента управления позволяет неоднократно вызывать это действие, без потребности отпустить кнопку мыши и нажимать ее снова. Задержка между двумя такими событиями определяется этим свойством.);
- **Model.Spin (boolean)** – определяет, что элемент управления имеет кнопки счетчика;
- **Model.StrictFormat (boolean)** – определяет, что дата проверяется во время пользовательского ввода;
- **Model.Time (long)** – определяет время, отображаемое в элементе управления;
- **Model.TimeFormat (long)** – определяет формат отображения времени;
- **Model.TimeMax (long)** – определяет максимальное время, которое может быть введено;
- **Model.TimeMin (long)** – определяет минимальное время, которое может быть введено;

Числовые поля

если пользовательский ввод ограничен числовыми значениями рекомендуют использовать элемент управления числовое поле (сервис `com.sun.star.awt.UnoControlNumericField`). Числовым значением управляет свойство `Value`, которая имеет тип `Double`. Минимальное и максимальное значение для пользовательского ввода определяется свойствами `ValueMin` и `ValueMax`. Десятичная точность числового значения определяется свойством `DecimalAccuracy`, например, значение 6 соответствует 6 десятичным знакам. Если свойство `ShowThousandsSeparator` устанавливается в `True`, отображается разделитель тысяч. Числовое поле также имеет встроенную кнопку счетчика, которая включается свойством `Spin`. Кнопка счетчика используется для инкремента и декремента отображаемого числового значения, щелчками мыши, тогда как шаг устанавливается свойством `ValueStep`.

```
numField = Dlg.getControl("NumericField1")
numField.Model.Value = 25.40
numField.Model.DecimalAccuracy = 2
```

Объект модели числового поля предоставляет следующие свойства:

- **Model.DecimalAccuracy (short)** – определяет десятичную точность;
- **Model.Repeat (boolean)** – определяет, должна ли мышь вызывать повторяющиеся события, то есть неоднократно вызвать действие, при удержании кнопки в нажатом положении;
- **Model.RepeatDelay (long)** – определяет задержку повторения для мыши, в миллисекундах (Когда пользователь нажимает кнопку мышь в области элемента управления, где это вызывает действие, тогда обычное выполнение элемента управления позволяет неоднократно вызывать это действие, без потребности отпустить кнопку мыши и нажимать ее снова. Задержка между двумя такими событиями определяется этим свойством.);
- **Model.ShowThousandsSeparator (boolean)** – определяет, должен ли отображаться разделитель тысяч;
- **Model.Spin (boolean)** – определяет, что элемент управления имеет кнопки счетчика;
- **Model.StrictFormat (boolean)** – определяет, что значение проверяется во время пользовательского ввода;
- **Model.Value (double)** – определяет значение, отображаемое в элементе управления;
- **Model.ValueMax (double)** – определяет максимальное значение, отображаемое в элементе управления;
- **Model.ValueMin (double)** – определяет минимальное значение, отображаемое в элементе управления;
- **Model.ValueStep (double)** – определяет шаг приращения значения, при использовании кнопок счетчика.

Поля валюты

Элемент управления поле валюты (сервис `com.sun.star.awt.UnoControlCurrencyField`) используется для ввода и отображения денежных значений. В дополнение к денежному значению, отображается символ валюты, который устанавливается свойством `CurrencySymbol`. Если свойство `PrependCurrencySymbol` устанавливается в `True`, символ валюты отображается перед денежным значением.

```
curField = Dlg.getControl("CurrencyField1")
curField.Model.Value = 500.00
curField.Model.CurrencySymbol = "€"
curField.Model.PrependCurrencySymbol = True
```

Объект модели поля валюты предоставляет следующие свойства:

- **Model.CurrencySymbol (string)** – определяет символ валюты;
- **Model.DecimalAccuracy (short)** – определяет десятичную точность;
- **Model.PrependCurrencySymbol (boolean)** – определяет, должен ли символ валюты находится перед числовым значением;
- **Model.Repeat (boolean)** – определяет, должна ли мышь вызывать повторяющиеся события, то есть неоднократно вызвать действие, при удержании кнопки в нажатом положении;
- **Model.RepeatDelay (long)** – определяет задержку повторения для мыши, в

миллисекундах (Когда пользователь нажимает кнопку мышь в области элемента управления, где это вызывает действие, тогда обычное выполнение элемента управления позволяет неоднократно вызывать это действие, без потребности отпустить кнопку мыши и нажимать ее снова. Задержка между двумя такими событиями определяется этим свойством.);

- **Model.ShowThousandsSeparator (boolean)** – определяет, должен ли отображаться разделитель тысяч;
- **Model.Spin (boolean)** – определяет, что элемент управления имеет кнопки счетчика;
- **Model.StrictFormat (boolean)** – определяет, что значение проверяется во время пользовательского ввода;
- **Model.Value (double)** – определяет значение, отображаемое в элементе управления;
- **Model.ValueMax (double)** – определяет максимальное значение, отображаемое в элементе управления;
- **Model.ValueMin (double)** – определяет минимальное значение, отображаемое в элементе управления;
- **Model.ValueStep (double)** – определяет шаг приращения значения, при использовании кнопок счетчика.

Поле форматирования

Элемент управления поле форматирования (сервис `com.sun.star.awt.UnoControlFormattedField`) определяет формат, который используется для форматирования вводимых и отображаемых данных. Источник числовых форматов может быть установлен в свойстве `FormatsSupplier`, а ключ формата для используемого формата должен быть определен в свойстве `FormatKey`. Рекомендуется использовать браузер свойств в редакторе диалога для установки этих свойств. Поддерживаемые форматы чисел – число, процент, валюта, дата, время, научное, дробь и логическое значения. Поэтому, поле форматирования может использоваться вместо поля даты, поля времени, числового поля или поля валюты.

Объект модели поля форматирования предоставляет следующие свойства:

- **Model.EffectiveDefault (any)** – определяет значение по умолчанию поля форматирования, оно может быть числовым значением (double) или строкой, в зависимости от поля форматирования;
- **Model.EffectiveMax (double)** – определяет максимальное значение, которое может быть введено, это свойство игнорируется, если формат поля не числовой;
- **Model.EffectiveMin (double)** – определяет минимальное значение, которое может быть введено, это свойство игнорируется, если формат поля не числовой;
- **Model.EffectiveValue (double)** – определяет текущее значение поля форматирования, оно может быть числовым значением (double) или строкой, в зависимости от поля форматирования;
- **Model.FormatKey (long)** – определяет формат, который используется полем форматирования для ввода и вывода, это значение является значащим только для свойства `FormatsSupplier`;

- **Model.FormatsSupplier** – предоставляет форматы, с которыми поле должно работать;
- **Model.Repeat (boolean)** – определяет, должна ли мышь вызывать повторяющиеся события, то есть неоднократно вызвать действие, при удержании кнопки в нажатом положении;
- **Model.RepeatDelay (long)** – определяет задержку повторения для мыши, в миллисекундах (Когда пользователь нажимает кнопку мышь в области элемента управления, где это вызывает действие, тогда обычное выполнение элемента управления позволяет неоднократно вызывать это действие, без потребности отпустить кнопку мыши и нажимать ее снова. Задержка между двумя такими событиями определяется этим свойством.);
- **Model.Spin (boolean)** – определяет, что элемент управления имеет кнопки счетчика;
- **Model.StrictFormat (boolean)** – определяет, что значение проверяется во время пользовательского ввода;
- **Model.TreatAsNumber (boolean)** – определяет, что текст рассматривается как число.

Поле с маской ввода

Элемент управления поле с маской ввода (сервис `com.sun.star.awt.UnoControlPatternField`) отображает и осуществляет ввод строки согласно указанной маске. Поля, в которые пользователь осуществляет ввод в поле с маской ввода, определяются в свойстве `EditMask` как специальный символьный код. Длина маски редактирования определяет число возможных позиций ввода. Если вводится символ, который не соответствует маске редактирования, ввод отклоняется. Например, в маске редактирования "NNLNNLLLLL" символ L имеет значение постоянного текста, и символ N означает, что могут быть введены только цифры от 0 до 9. Полный список действительных символов может быть найден в онлайн справке OpenOffice.org. Свойство `LiteralMask` содержит начальные значения, которые отображаются в поле с маской ввода. Длина буквальной маски всегда должна соответствовать длине маски редактирования. Пример буквальной маски, которая соответствует к вышеупомянутой маске редактирования, может быть “__.__.2002”. В этом случае, пользователь вводит только 4 цифры, при вводе даты.

```
PatternField = Dlg.getControl("PatternField1")
PatternField.Model.EditMask = "NNLNNLLLLL"
PatternField.Model.LiteralMask = "__.__.2002"
```

Объект модели поля с маской ввода предоставляет следующие свойства:

- **Model.EditMask (string)** – определяет маску редактирования;
- **Model.LiteralMask (string)** – определяет символьную маску;
- **Model.MaxTextLen (short)** – определяет максимальное число символов;
- **Model.StrictFormat (boolean)** – определяет, что значение проверяется во время пользовательского ввода.

Выбор файла

Элемент управления выбор файла (сервис `com.sun.star.awt.UnoControlFileControl`) имеет все свойства элемента управления текстовое поле, с дополнительной встроенной командной кнопкой. Когда нажимают кнопку, открывается диалог выбора файла. Каталог,

который первоначально показывает диалог выбора файла, устанавливается через свойство `Text`. Каталог нужно указывать как путь в системе, в настоящее время URL файла не работает. В OOo Basic Вы можете использовать функцию времени выполнения `ConvertToURL()`, для преобразования пути системы к URL.

```
FileControl = Dlg.getControl("FileControl1")
FileControl.Text = "D:\Programme\Office60"
```

Фильтры для диалога выбора файла не могут быть установлены или применены для элемента управления выбор файла. Альтернативный путь состоит в том, чтобы использовать текстовое поле и командную кнопку вместо элемента управления выбор файла и назначить макрос на кнопку, который иллюстрируется примерами диалога выбора файла com.sun.star.ui.dialogs.FilePicker во время выполнения. Пример приводится ниже.

```
Sub OpenFileDialog()
  Dim oFilePicker As Object, oSimpleFileAccess As Object
  Dim oSettings As Object, oPathSettings As Object
  Dim txtField As Object
  Dim sFileURL As String
  Dim sFiles As Variant

  ' файловый диалог
  oFilePicker = CreateUnoService("com.sun.star.ui.dialogs.FilePicker")

  ' установка фильтров
  oFilePicker.AppendFilter("All files (*.*)", "*.*")
  oFilePicker.AppendFilter("StarOffice 6.0 Text Text Document", "*.sxw")
  oFilePicker.AppendFilter("StarOffice 6.0 Spreadsheet", "*.sxc")
  oFilePicker.SetCurrentFilter("All files (*.*)")

  ' если URL файла не установлен, получаем параметры пути из конфигурации
  txtField = Dlg.GetControl("TextField1")
  sFileURL = ConvertToURL(txtField.Model.Text)
  If sFileURL = "" Then
    oSettings = CreateUnoService("com.sun.star.frame.Settings")
    oPathSettings = oSettings.getByName("PathSettings")
    sFileURL = oPathSettings.getPropertyValue("Work")
  End If

  ' установка отображаемого каталога
  oSimpleFileAccess = CreateUnoService("com.sun.star.ucb.SimpleFileAccess")
  If oSimpleFileAccess.exists(sFileURL) And _
          oSimpleFileAccess.isFolder(sFileURL) Then
    oFilePicker.setDisplayDirectory(sFileURL)
  End If

  ' выполнение файлового диалога
  If oFilePicker.execute() Then
    sFiles = oFilePicker.getFiles()
    sFileURL = sFiles(0)
    If oSimpleFileAccess.exists(sFileURL) Then
      ' установка пути файла в текстовом поле
      oTextField.SetText(ConvertFromURL(sFileURL))
    End If
  End If
End Sub
```

Списки

Элемент управления список (сервис `com.sun.star.awt.UnoControlListBox`) отображает список элементов, из которых пользователь может выбрать один или несколько. Если число элементов превышает то количество, которое может быть отображено в поле списка, в элементе управления автоматически появляются полосы прокрутки. Если свойство `Dropdown` установлено в `True`, список элементов отображается в раскрывающемся списке. В этом случае, максимальное количество строк в раскрывающемся списке определяется свойством `LineCount`. Фактическим списком элементов управляет свойство

`StringItemList`. Всеми выбранными пунктами управляет свойство `SelectedItems`. Если свойство `MultiSelection` установлено в `True`, может быть выбран более чем один элемент.

Списки поддерживают следующие свойства:

- **ItemCount (Short)** – число элементов, только для чтения;
- **SelectedItem (String)** – текст выделенного элемента, только для чтения;
- **SelectedItems (Array Of Strings)** – массив данных с выделенными записями (для списков, которые поддерживают множественный выбор), только для чтения;
- **SelectItemPos (Short)** – номер элемента, выделенного в настоящее время, только для чтения;
- **SelectItemsPos (Array of Short)** – массив данных с номерами выделенных записей (для списков, которые поддерживают множественный выбор), только для чтения;
- **MultipleMode (Boolean)** – `True` активизирует возможность для множественного выбора записей, `False` блокирует множественный выбор (свойство нельзя вызвать непосредственно, но только через `IsMultipleMode`);
- **IsMultipleMode (Boolean)** – разрешает множественный выбор в пределах списка, только для чтения.

Списки предоставляют следующие методы:

- **addItem (Item, Pos)** – вводит строку, определенную в `Item` в список в позиции `Pos`;
- **addItems (ItemArray, Pos)** – вводит записи, перечисленные в строковом массиве данных `ItemArray` в список в позиции `Pos`;
- **selectItem (Item, SelectMode)** – активизирует или дезактивирует выделение для элемента, определенного в строке `Item` в зависимости от логической переменной `SelectMode`;
- **makeVisible (Pos)** – прокручивает область списка так, чтобы элемент, определенный параметром `Pos` был видим.

Объект модели списка предоставляет следующие свойства:

- **Model.BackgroundColor (long)** – цвет фона элемента управления;
- **Model.Border (short)** – тип границы (0: нет границы, 1: трехмерная граница, 2: простая граница);
- **Model.FontDescriptor (struct)** – структура с подробностями используемого шрифта (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **Model.LineCount (Short)** – число строк в элементе управления;
- **Model.MultiSelection (Boolean)** – разрешает множественный выбор записей;
- **Model.SelectedItems (Array of Strings)** – список выделенных записей;
- **Model.StringItemList (Array of Strings)** – список всех записей;
- **Model.Printable (Boolean)** – элемент управления может быть напечатан;
- **Model.ReadOnly (Boolean)** – содержимое элемента управления только для чтения;
- **Model.Tabstop (Boolean)** – элемент управления может быть достигнут при помощи

клавиши Tab;

- **Model.TextColor (Long)** – цвет текста элемента управления;
- **Model.HelpText (String)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Примечание Параметр VBA для выделения элемента списка с дополнительным числовым значением (`ItemData`) не существует в OOo Basic. Если Вы хотите управлять числовым значением (например удостоверение личности из базы данных) в дополнение к тексту естественного языка, Вы должны создать вспомогательное поле данных, которое управляется параллельно со списком.

При работе со списками элемент может быть добавлен к списку в определенном положении с помощью метода `addItem`. Например, элемент добавляется в конец списка:

```
Dim nCount As Integer

lstBox = Dlg.getControl("ListBox1")
nCount = lstBox.getItemCount()
lstBox.addItem("Новый элемент", nCount)
```

Несколько элементов добавляются с помощью метода `addItems`. Метод `removeItems` используется для удаления элементов из списка. Например, первый элемент в списке удаляется:

```
Dim nPos As Integer, nCount As Integer

nPos = 0
nCount = 1
lstBox.removeItems(nPos, nCount)
```

Элемент списка может быть заранее выбран методами `selectItemPos`, `selectItemsPos` и `selectItem`. Например, первый элемент в списке может быть выбран:

```
lstBox.selectItemPos(0, True)
```

Поле со списком

Элемент управления поле со списком (сервис `com.sun.star.awt.UnoControlComboBox`) представляет для пользователя список выбора. Дополнительно, он содержит текстовое поле, позволяя пользователю ввести вариант, который отсутствует в списке. Поле со списком используется, когда имеется только список предложенных выборов, тогда как список используется, когда пользовательский ввод ограничен только списком.

Возможности и свойства поля со списком и списка подобны. Также в поле со списком список элементов может быть показан в раскрывающемся списке, если свойство Dropdown установлено в True. Фактический список элементов доступен через свойство StringItemList. Текстом, отображаемым в текстовом поле поля со списком управляет свойство Text.

Поля со списком поддерживают следующие свойства:

- **ItemCount (Short)** – число элементов, только для чтения;
- **Text (String)** – текущий текст;
- **MaxTextLen (short)** – максимальное количество символов, которые Вы можете ввести

в поле;

Поля со списком предоставляют следующие методы:

- **addItem (Item, Pos)** – вводит строку, определенную в `Item` в список в позиции `Pos`;
- **addItems (ItemArray, Pos)** – вводит записи, перечисленные в строковом массиве данных `ItemArray` в список в позиции `Pos`;
- **getDropDownLineCount ()** –возвращает число видимых линий в выпадающем списке поля со списком;
- **getItemCount ()** – возвращает число элементов в поле со списком;
- **getItem (Pos)** – возвращает элемент в указанной позиции `Pos`;
- **getItems ()** – возвращает все элементы поля со списком;
- **removeItems (Pos, Count)** – удаляет `Count` элементов в позиции `Pos`;
- **setDropDownLineCount (Lines)** – устанавливает число отображаемых линий в выпадающем списке поля со списком;

Объект модели списка предоставляет следующие свойства:

- **Model.Align (short)** – определяет горизонтальное выравнивание текста в элементе управления;
- **Model.Autocomplete (boolean)** – определяет, разрешено ли автоматическое завершение текста;
- **Model.BackgroundColor (long)** – цвет фона элемента управления;
- **Model.Border (short)** – тип границы (0: нет границы, 1: трехмерная граница, 2: простая граница);
- **Model.Dropdown (boolean)** – определяет, имеет ли элемент управления кнопку раскрывающегося списка;
- **Model.FontDescriptor (struct)** – структура с подробностями используемого шрифта (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **Model.HelpText (string)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (string)** – URL страницы онлайн справки для соответствующего элемента управления;
- **Model.HideInactiveSelection (boolean)** – определяет, должен ли вариант выбора в элементе управления быть скрыт, когда элемент управления не активен (не в фокусе);
- **Model.LineCount (short)** – число строк в элементе управления;
- **Model.MaxTextLen (short)** – определяет максимальное количество символов;
- **Model.StringItemList (Array of Strings)** – список всех записей;
- **Model.Printable (Boolean)** – элемент управления может быть напечатан;
- **Model.ReadOnly (Boolean)** – содержимое элемента управления только для чтения;
- **Model.Tabstop (Boolean)** – элемент управления может быть достигнут при помощи клавиши Tab;

- **Model.Text (String)** – текст связанный с элементом управления;
- **Model.TextColor (Long)** – цвет текста элемента управления;

Если пользователь выбирает элемент из списка, выбранный элемент отображается в текстовом поле и может быть получен через свойство `Text`:

```
Function GetSelectedItem( oComboBox As Object ) As String
  GetSelectedItem = oComboBox.Model.Text
End Function
```

Когда пользователь вводит текст в текстовое поле поля со списком, автоматическое завершение слова – полезная возможность и включается установкой свойства `Autocomplete` в `True`. Рекомендуется использовать интерфейс `com.sun.star.awt.XComboBox`, для получения доступа к элементам поля со списком:

```
Dim oComboBox As Object
Dim nCount As Integer
Dim sItems As Variant

' получение элемента управления
oComboBox = Dlg.getControl("ComboBox1")

' сначала удаляем все старые элементы из списка
nCount = oComboBox.getItemCount()
oComboBox.removeItems(0, nCount)

' добавляем новые элементы в список
sItems = Array("Item1", "Item2", "Item3", "Item4", "Item5")
oComboBox.addItems(sItems, 0)
```

Надписи

Элемент управления надпись (сервис `com.sun.star.awt.UnoControlFixedText`) отображает на экране текст, который пользователь не может редактировать. Например, надписи используется, для добавления описательных подписей к текстовым полям, спискам, и полям со списком.

Надпись используется для определения клавиши быстрого выбора для элементов управления без ярлыков. Когда пользователь нажимает символьную клавишу одновременно с клавишей ALT, элемент управления автоматически получает фокус. Чтобы назначать клавишу быстрого выбора элементу управления без надписи, например, текстовому полю, используется надпись. Поставьте тильду перед соответствующим символом в свойстве Label надписи. Поскольку надпись не может получить фокус, фокус автоматически перемещается в следующий элемент управления в последовательности табуляции. Поэтому, важно, чтобы надпись и текстовое поле имели последовательные индексы табуляции.

```
Dim lblOne As Object

lblOne = Dlg.getControl("Label1")
lblOne.Model.Label = "Enter ~Text"
```

Объект модели надписи предоставляет следующие свойства:

- **Model.Align (Short)** – определяет горизонтальное выравнивание текста в элементе управления;
- **Model.BackgroundColor (Long)** – определяет фоновый цвет (RGB) элемента управления;
- **Model.Border (Short)** – определяет стиль границы элемента управления;
- **Model.BorderColor (Long)** – определяет цвет границы, если она присутствует;

- **Model.Enabled (Boolean)** – активизирует элемент управления;
- **Model.FontDescriptor (struct)** – структура, которая определяет детали шрифта, который используется элементом управления (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **Model.HelpText (String)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.
- **Model.Label (String)** – надпись, которая отображается на элементе управления;
- **Model.MultiLine (Boolean)** – определяет, что текст может отображаться на более чем одной строке;
- **Model.Printable (Boolean)** – элемент управления может быть напечатан;
- **Model.TextColor (Long)** – цвет текста элемента управления;
- **Model.VerticalAlign (Long)** – определяет вертикальное выравнивание текста в элементе управления (по умолчанию значение в соответствии с `com.sun.star.style.VerticalAlignment`).

Индикатор выполнения

Элемент управления индикатор выполнения `com.sun.star.awt.UnoControlProgressBar` показывает увеличение или уменьшение полосы, чтобы дать пользовательскую обратную связь о течении операции, например, завершении длинной задачи. Минимальное и максимальное значение выполнения элемента управления устанавливаются свойствами `ProgressValueMin` и `ProgressValueMax`. Значением выполнения управляет свойство `ProgressValue`. По умолчанию, индикатор выполнения является синим, но цвет заполнения может быть изменен, установкой свойства `FillColor`.

Объект модели элемента управления индикатор выполнения предоставляет следующие свойства:

- **Model.BackgroundColor (long)** – определяет цвет фона (RGB) элемента управления;
- **Model.Border (short)** – определяет стиль элемента управления;
- **Model.BorderColor (long)** – определяет цвет границы (RGB) элемента управления, если она присутствует;
- **Model.FillColor (long)** – определяет цвет заполнения (RGB) элемента управления;
- **Model.HelpText (string)** – текст всплывающей подсказки, которая появляется, когда Вы перемещаете курсор мыши над элементом управления;
- **Model.HelpURL (string)** – URL страницы онлайн справки для соответствующего элемента управления.
- **Model.Printable (boolean)** – элемент управления может быть напечатан;
- **Model.ProgressValue (long)** – определяет значение выполнения элемента управления;
- **Model.ProgressValueMax (long)** – определяет максимальное значение выполнения элемента управления;

- **Model.ProgressValueMix (long)** – определяет минимальное значение выполнения элемента управления;

Функциональные возможности индикатора выполнения демонстрируются в следующем примере:

```
Sub ProgressBarDemo()
  Dim oProgressBar As Object
  Dim oCancelButton As Object
  Dim oStartButton As Object
  Dim ProgressValue As Long

  ' настройка индикатора выполнения
  Const ProgressValueMin = 0
  Const ProgressValueMax = 40
  Const ProgressStep = 4

  ' установка минимального и максимального значений выполнения
  oProgressBar.Model = Dlg.getControl("ProgressBar1")
  oProgressBar.Model.ProgressValueMin = ProgressValueMin
  oProgressBar.Model.ProgressValueMax = ProgressValueMax

  ' отключение кнопок отмена и старт
  oCancelButton = Dlg.getControl("CommandButton1")
  oCancelButton.Model.Enabled = False
  oStartButton = Dlg.getControl("CommandButton2")
  oStartButton.Model.Enabled = False

  ' отображение индикатора выполнения
  oProgressBar.setVisible(True)

  ' увеличение значения выполнения каждую секунду
  For ProgressValue = ProgressValueMin To ProgressValueMax Step ProgressStep
    oProgressBar.Model.ProgressValue = ProgressValue
    Wait 1000
  Next ProgressValue

  ' скрытие индикатора выполнения
  oProgressBar.setVisible(False)

  ' включение кнопок отмена и старт
  oCancelButton.Model.Enabled = True
  oStartButton.Model.Enabled = True
End Sub
```

Горизонтальная/Вертикальная полосы прокрутки

Если видимая область в диалоге является меньшей чем визуализируемое содержимое, элемент управления полоса прокрутки `com.sun.star.awt.UnoControlScrollBar` обеспечивает навигацию по содержимому, обеспечивая прокрутку по горизонтали или вертикали. Кроме того, элемент управления полоса прокрутки используется для обеспечения прокрутки в элементах управления, которые не имеют встроенных полос прокрутки.

Ориентация полосы прокрутки определяется свойством `Orientation` и может быть горизонтальной или вертикальной. Полоса прокрутки имеет указатель (ползунок), который пользователь может тянуть мышью к любому положению на полосе прокрутки. Положением указателя управляет свойство `ScrollValue`. Для горизонтальной полосы прокрутки, крайнее левое положение соответствует минимальному значению прокрутки 0, а самому правому положению максимальное значение прокрутки, определяемое свойством `ScrollValueMax`. Полоса прокрутки также имеет стрелки на своих концах, когда на них нажимают или удерживают, указатель перемещается по полосе прокрутки с приращением, чтобы увеличить или уменьшить значение прокрутки. Изменение значения прокрутки при нажатии мышью на стрелку, определяется свойством `LineIncrement`. При нажатии на

полосе прокрутки в области между указателем и стрелками, значение прокрутки увеличивается или уменьшения на значение устанавливаемое свойством `BlockIncrement`. Положение указателя представляет часть визуализируемого содержания, которое является в настоящее время видимым в диалоге. Видимый размер указателя устанавливается свойством `VisibleSize` и представляет процент от видимого в настоящее время содержания и полного визуализируемого содержания.

```
oScrollBar = Dlg.getControl("ScrollBar1")
oScrollBar.Model.ScrollValueMax = 100
oScrollBar.Model.BlockIncrement = 20
oScrollBar.Model.LineIncrement = 5
oScrollBar.Model.VisibleSize = 20
```

Элемент управления полоса прокрутки использует событие регулирования `com.sun.star.awt.AdjustmentEvent`, для управления движением указателя по полосе прокрутки. В обработчике события для событий регулирования разработчик может изменить положение видимого содержания на диалоге как функция свойства `ScrollValue`.

Объект модели элемента управления полоса прокрутки предоставляет следующие свойства:

- **Model.BlockIncrement (long)** – определяет приращение для блочного перемещения;
- **Model.LineIncrement (long)** – определяет приращение для перемещения на одну строку;
- **Model.LiveScroll (boolean)** – определяет поведение прокручивания элемента управления (если `True`, то, когда пользователь перемещает ползунок в полосе прокрутки, содержимое окна обновляется немедленно; если `False`, то окно обновляется только после того, как пользователь отпустил кнопку мыши);
- **Model.Orientation (long)** – определяет ориентацию полосы прокрутки (в соответствии со списком `com.sun.star.awt.ScrollBarOrientation`);
- **Model.RepeatDelay (long)** – определяет задержку повторения для мыши, в миллисекундах (Когда пользователь нажимает кнопку мышь в области элемента управления, где это вызывает действие, тогда обычное выполнение элемента управления позволяет неоднократно вызывать это действие, без потребности отпустить кнопку мыши и нажимать ее снова. Задержка между двумя такими спусковыми механизмами определена этим свойством.);
- **Model.ScrollValue (long)** – определяет значение прокрутки элемента управления;
- **Model.ScrollValueMin (long)** – определяет минимальное значение прокрутки элемента управления, если это дополнительное свойство отсутствует, минимальное значение прокрутки принимается равным 0;
- **Model.ScrollValueMax (long)** – определяет максимальное значение прокрутки элемента управления;
- **Model.VisibleSize (long)** – определяет видимый размер полосы прокрутки.

В следующем примере, размер области надписи превышает размер диалога. Каждый раз, когда пользователь нажимает на полосу прокрутки, вызывается макрос `AdjustmentHandler()`, и положение поля надписи в диалоге изменяется согласно значениям прокрутки.

```
Sub AdjustmentHandler()
  Dim oLabel As Object
  Dim oScrollBar As Object
  Dim ScrollValue As Long, ScrollValueMax As Long
  Dim VisibleSize As Long
  Dim Factor As Double
```

```
  Static bInit As Boolean
  Static PositionX0 As Long
  Static Offset As Long

  ' получение модели надписи
  oLabel = Dlg.getControl("Label1")

  ' при инициализации запоминаем положение надписи и вычисляем смещение
  If bInit = False Then
    bInit = True
    PositionX0 = oLabel.Model.PositionX
    OffSet = PositionX0 + oLabel.Model.Width - (Dlg.Model.Width - Border)
  End If

  ' получение модели полосы прокрутки
  oScrollBar = Dlg.getControl("ScrollBar1")

  ' получение фактического значения свитка
  ScrollValue = oScrollBar.Model.ScrollValue

  ' вычисление и установка нового положения надписи
  ScrollValueMax = oScrollBar.Model.ScrollValueMax
  VisibleSize = oScrollBar.Model.VisibleSize
  Factor = Offset / (ScrollValueMax - VisibleSize)
  oLabel.Model.PositionX = PositionX0 - Factor * ScrollValue
End Sub
```

Графический элемент управления

Если пользователь хочет показать изображение без функциональных возможностей кнопки, выбирается графический элемент управления (сервис `com.sun.star.awt.UnoControlImageControl`). Местоположение графического файла для элемента управления устанавливается свойством `ImageURL`. Обычно, размер изображения не соответствует размеру элемента управления, поэтому графический элемент управления автоматически масштабирует изображение к размеру элемента управления, установкой свойства `ScaleImage` в `True`.

```
imgControl = Dlg.getControl("ImageControl1")
imgControl.Model.ImageURL = "file:///D:/Office60/share/gallery/photos/beach.jpg"
imgControl.Model.ScaleImage = True
```

Объект модели графичекого элемента управления предоставляет следующие свойства:

- **Model.ImageURL (string)** – определяет URL изображения для использования в элементе управления;
- **Model.ScaleImage (boolean)** – определяет, масштабируется ли изображение автоматически по размеру элемента управления.

Группы

Элемент управления группа (сервис `com.sun.star.awt.UnoControlGroupBox`) создает рамку, чтобы визуально собрать в группу другие элементы управления, такие как переключатели и флажки. Отметьте, что элемент управления группа не обеспечивает никаких контейнерных функциональных возможностей для других элементов управления, он имеет только визуальные функциональные возможности. Для получения дополнительной информации, см. раздел “Переключатели” на стр. 156.

Группа содержит надпись, внедренную в пределах границы и устанавливаемую с помощью свойства `Label`. В большинстве случаев, элемент управления группа используется пассивно.

Горизонтальные/Вертикальные линии

Элемент управления линия (сервис `com.sun.star.awt.UnoControlFixedLine`) создает простые линии в диалоге. В большинстве случаев, элемент управления линия используется для визуального разделения диалога. Элемент управления линия может иметь горизонтальную или вертикальную ориентацию, которая определяется свойством `Orientation`. Надпись элемента управления линия устанавливается свойством `Label`. Отметьте, что надпись отображается только если элемент управления имеет горизонтальную ориентацию.

Глава 12 Формы

Во многих отношениях, структура форм OpenOffice.org соответствует диалогам, обсужденным в предыдущей главе. Есть, однако, несколько ключевых различий:

- диалоги появляются в форме одного единственного окна диалога, которое отображается над документом и не разрешает никаких действий кроме обработки диалога, пока диалог не завершен. Формы, с другой стороны, отображаются непосредственно в документе, точно так же как рисованные элементы;
- редактор диалога предоставляется для создания диалогов, и он может быть найден в среде разработки OOo Basic. Формы создаются с использованием панели инструментов Элементы управления непосредственно в пределах документа;
- примате во внимание, что функции диалога доступны во всех документах OpenOffice.org, полные возможности функций формы доступны только в текстовых документах и электронных таблицах;
- элементы управления формы могут быть связаны с внешней таблицей базы данных. Эта функция не доступна в диалогах;
- элементы управления диалогов и форм отличаются по нескольким аспектам.

Пользователи, которые хотят предоставить своим формам их собственные методы для обработки событий, должны обратиться к Главе 11, Диалоги. Механизмы, объясняемые там идентичны аналогичным для форм.

Работа с Формами

Формы OpenOffice.org могут содержать текстовые поля, списки, переключатели и набор других элементов управления, которые вставляются непосредственно в текст или электронную таблицу. Панель инструментов **Элементы управления** используется для редактирования форм.

Форма OpenOffice.org может принимать один из двух режимов: режим проектирования и режим отображения. В режиме проектирования, может быть изменено положение элементов управления и их свойства могут быть отредактированы с использованием окна свойств.

Панель инструментов **Элементы управления** также используется для переключения между режимами.

Определение объектов форм

OpenOffice.org помещает элементы управления форм на уровень графического объекта. К

фактическому объекту формы можно получить доступ через список Форм на графическом уровне. К объектам получают доступ следующим образом в текстовых документах:

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

Метод `GetByIndex` возвращает форму с индексом 0.

При работе с электронными таблицами, необходима промежуточная стадия – список Листов, потому что графический уровень располагается не непосредственно в документе, а в отдельных листах:

```
Dim Doc As Object
Dim Sheet As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.GetByIndex(0)
DrawPage = Sheet.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

Как уже подсказывает название метода `GetByIndex`, документ может содержать несколько форм. Это полезно, например, если содержание различных баз данных отображается в пределах одного документа, или если отношения базы данных один ко многим отображаются в пределах формы. Вариант создания подформ также обеспечивается с этой целью.

Три аспекта элементов управления форм

Элемент управления формы имеет три аспекта:

- Изначально, есть *Модель* элемента управления. Это ключевой объект для программиста OOo Basic, при работе с элементами управления форм.
- Противоположная сторона этому - *Представление* элемента управления, которое управляет отображением информации.
- Так как элементы управления форм в пределах документов управляются как специальные графические элементы, есть также объект *Shape*, который отражает определенные для элемента графические свойства элемента управления (в особенности его положение и размер).

Доступ к модели элементов управления форм

Модель элементов управления формы доступна через метод `GetByName` объекта форма:

```
Dim Doc As Object
Dim Form As Object
Dim Ctl As Object

Doc = StarDesktop.CurrentComponent
Form = Doc.DrawPage.Forms.GetByIndex(0)
Ctl = Form.getByName("MyListBox")
```

Пример определяет модель элемента управления `MyListBox`, который расположен в первой форме открытого в настоящее время текстового документа.

Если Вы не уверены относительно формы элемента управления, Вы можете использовать параметр для поиска по всем формам для требуемого элемента управления:

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
  Form = Forms.GetbyIndex(I)
  If Form.HasByName("MyListBox") Then
    Ctl = Form.GetbyName("MyListBox")
    Exit Function
  End If
Next I
```

Пример использует метод `HasByName`, чтобы проверить все формы текстового документа для определения, содержат ли они модель элемента управления по имени `MyListBox`. Если соответствующая модель найдена, то ссылка на нее сохраняется в переменной `Ctl` и поиск прекращается.

Доступ к представлению элементов управления форм

Чтобы получить доступ к представлению элемента управления формы, сначала необходима связанная модель. Представление элемента управления может быть определено с помощью модели и использования диспетчера документа.

```
Dim Doc As Object
Dim DocCrl As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim CtlView As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
DocCrl = Doc.getCurrentControler()
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
  Form = Forms.GetbyIndex(I)
  If Form.HasByName("MyListBox") Then
    Ctl = Form.GetbyName("MyListBox")
    CtlView = DocCrl.GetControl(Ctl)
    Exit Function
  End If
Next I
```

Код, приведенный в примере очень подобен коду, приведенному в предыдущем примере для определить модели элемента управления. Он использует не только объект документа `Doc`, но также и объект диспетчера документа `DocCrl`, который создает ссылку на текущее окно документа. С помощью этого объекта диспетчера и модели элемента управления, он тогда использует метод `GetControl` для определения представления (переменная `CtlView`) элемента управления формы.

Доступ к объекту Shape элементов управления форм

Метод для доступа к объектам Shape элемента управления также использует соответствующий графический уровень документа. Для определения отдельного элемента

управления, все графические элементы графического уровня должны быть просмотрены.

```
Dim Doc As Object
Dim Shape as Object
Dim I as integer

Doc = StarDesktop.CurrentComponent

For I = 0 to Doc.DrawPage.Count - 1
  Shape = Doc.DrawPage(I)
  If HasUnoInterfaces(Shape, "com.sun.star.drawing.XControlShape") Then
    If Shape.Control.Name = "MyListBox" Then
      Exit Function
    End If
  End If
Next
```

Пример проверяет все графические элементы, чтобы определить, поддерживают ли они интерфейс `com.sun.star.drawing.XControlShape`, необходимый для элемента управления формы. Если дело обстоит так, тогда проверяется свойство `Control.Name`, является ли имя элемента управления `MyListBox`. Если это верно, функция заканчивает поиск.

Определение размера и положения элементов управления

Как уже было упомянуто, размер и положение элементов управления можно определить с использованием связанного объекта `shape`. Форма элемента управления, как все другие объекты `shape`, предоставляет с этой целью свойства `Size` и `Position`:

- **Size (struct)** – размер элемента управления (структура данных `com.sun.star.awt.Size`);
- **Position (struct)** - положение элемента управления (структура данных `com.sun.star.awt.Point`).

Следующий пример показывает, как могут быть заданы положение и размер элемента управления с использованием связанного объекта `shape`:

```
Dim Shape As Object

Point.x = 1000
Point.y = 1000

Size.Width = 10000
Size.Height = 10000

Shape.Size = Size
Shape.Position = Point
```

Элементы управления форм подробно

Элементы управления, доступные в формах подобны таковым из диалогов. Выбор простирается от простых текстовых полей через списки и поля со списками до различных кнопок.

Ниже, Вы найдете список самых важных свойств для элементов управления форм. Все свойства являются частью соответствующих моделей объектов.

В дополнение к стандартным элементам управления, табличные элементы управления также доступны для форм, которые позволяют полное вхождение таблиц баз данных. Они описаны в разделе “Формы баз данных” на странице 183 в Главе 12.

Кнопки

Модель объекта формы кнопка предоставляет следующие свойства:

- **BackgroundColor (long)** – цвет фона;
- **DefaultButton (Boolean)** – кнопка служит значением по умолчанию. В этом случае, она также отвечает на клавишу Enter, если нет никакого фокуса;
- **Enabled (Boolean)** – элемент управления может быть активизирован;
- **Tabstop (Boolean)** – элемент управления может быть достигнут через клавишу Tab;
- **TabIndex (Long)** – положение элемента управления в последовательности активации;
- **FontName (String)** – имя типа шрифта;
- **FontHeight (Single)** – высота символа в пунктах (pt);
- **Tag (String)** – строка, содержащая дополнительную информацию, которая может быть сохранена в элементе управления для программно-управляемого доступа;
- **TargetURL (String)** – заданный URL для кнопок типа URL;
- **TargetFrame (String)** – имя окна (или фрейма), в котором `TargetURL` должен быть открыт при активизации кнопки (для кнопок типа URL);
- **Label (String)** – надпись на кнопке;
- **TextColor (Long)** – цвет текста элемента управления;
- **HelpText (String)** – текст подсказки, который отображается, если курсор мыши находится над элементом управления;
- **HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления;
- **ButtonType (Enum)** – действие, которое связано с кнопкой (значение по умолчанию из `com.sun.star.form.FormButtonType`).

Через свойство `ButtonType`, Вы имеете возможность определить действие, которое автоматически выполняется, когда кнопка нажата. Связанная группа констант `com.sun.star.form.FormButtonType` обеспечивает следующие значения:

- **PUSH** – стандартная кнопка;
- **SUBMIT** – завершение ввода в форму (особенно уместно для HTML форм);
- **RESET** - сброс всех значений в пределах формы к их начальным значениям;
- **URL** - вызов URL, определенного в `TargetURL` (открывается в пределах окна, которое было определено через `TargetFrame`).

Типы кнопок **ОК** и **Отмена**, предусмотренные в диалогах не поддерживаются в формах.

Переключатели

Следующие свойства переключателя доступны через его объект модель:

- **Enabled (Boolean)** – элемент управления может быть активирован;

- **Tabstop (Boolean)** – элемент управления может быть достигнут через клавишу Tab;
- **TabIndex (Long)** – положение элемента управления в последовательности активации;
- **FontName (String)** – имя типа шрифта;
- **FontHeight (Single)** – высота символа в пунктах (pt);
- **Tag (String)** – строка, содержащая дополнительную информацию, которая может быть сохранена в элементе управления для программно-управляемого доступа;
- **Label (String)** – надпись на элементе управления;
- **Printable (Boolean)** - элемент управления может быть напечатан;
- **State (Short)** - если 1, переключатель активирован, иначе он дезактивирован;
- **RefValue (String)** - строка для сохранения дополнительной информации (например, для управления идентификатором записи данных);
- **TextColor (Long)** - цвет текста элемента управления;
- **HelpText (String)** – текст подсказки, который отображается, если курсор мыши находится над элементом управления;
- **HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Механизм для группировки переключателей различается для элементов управления диалогов и форм. Принимая во внимание, что элементы управления, появляющиеся один за другим в диалогах автоматически объединяются, чтобы сформировать группу, группировка в формах выполняется на основе имен. Чтобы сделать это, все переключатели группы должны содержать одинаковое имя. OpenOffice.org объединяет группы элементов управления в массив так, чтобы отдельные кнопки из программы OOo Basic могли быть достигнуты таким же образом как и ранее.

Следующий пример показывает, как может быть определена модель группы элементов управления.

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
  Form = Forms.GetbyIndex(I)
  If Form.HasByName("MyOptions") Then
    Ctl = Form.GetGroupbyName("MyOptions")
    Exit Function
  End If
Next I
```

Код соответствует предыдущему примеру для определения модели простого элемента управления. Он в цикле просматривает все формы в текущем текстовом документе и использует метод `HasByName` для проверки, содержит ли соответствующая форма элемент с именем `MyOptions`, который он ищет. Если дело обстоит так, то к модели массива получают доступ, используя метод `GetGroupByName` (а не метод `GetByName`, для определения простой модели).

Флажки

Модель объекта формы флажок предоставляет следующие свойства:

- **Enabled (Boolean)** – элемент управления может быть активирован;
- **Tabstop (Boolean)** – элемент управления может быть достигнут через клавишу Tab;
- **TabIndex (Long)** – положение элемента управления в последовательности активации;
- **FontName (String)** – имя типа шрифта;
- **FontHeight (Single)** – высота символа в пунктах (pt);
- **Tag (String)** – строка, содержащая дополнительную информацию, которая может быть сохранена в элементе управления для программно-управляемого доступа;
- **Label (String)** – надпись на элементе управления;
- **Printable (Boolean)** - элемент управления может быть напечатан;
- **State (Short)** - если 1, флажок активирован, иначе он дезактивирован;
- **RefValue (String)** - строка для сохранения дополнительной информации (например, для управления идентификатором записи данных);
- **TextColor (Long)** - цвет текста элемента управления;
- **HelpText (String)** – текст подсказки, который отображается, если курсор мыши находится над элементом управления;
- **HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Текстовые поля

Модель объекта формы текстовое поле предоставляет следующие свойства:

- **Align (short)** – выравнивание текста (0: выровненный по левому краю, 1: выровненный по центру, 2: выровненный по правому краю);
- **BackgroundColor (long)** – цвет фона;
- **Border (short)** – тип границы (0: нет границы, 1: трехмерная граница, 2: простая граница);
- **EchoChar (String)** - эхо-символ для полей ввода пароля;
- **FontName (String)** – имя типа шрифта;
- **FontHeight (Single)** – высота символа в пунктах (pt);
- **HardLineBreaks (Boolean)** – автоматические разрывы строк постоянно вставляются в текст элемента управления;
- **HScroll (Boolean)** – текст имеет горизонтальную полосу прокрутки;
- **MaxTextLen (Short)** – максимальная длина текста; если определен 0, нет никаких ограничений;
- **MultiLine (Boolean)** – разрешает многострочный ввод;

- **Printable (Boolean)** – элемент управления может быть напечатан;
- **ReadOnly (Boolean)** – содержимое элемента управления только для чтения;
- **Enabled (Boolean)** - элемент управления может быть активирован;
- **Tabstop (Boolean)** – элемент управления может быть достигнут через клавишу Tab;
- **TabIndex (Long)** – положение элемента управления в последовательности активации;
- **FontName (String)** – имя типа шрифта;
- **FontHeight (Single)** – высота символа в пунктах (pt);
- **Text (String)** – текст элемента управления;
- **TextColor (Long)** – цвет текста элемента управления;
- **VScroll (Boolean)** – текст имеет вертикальную полосу прокрутки;
- **HelpText (String)** – текст подсказки, который отображается, если курсор мыши находится над элементом управления;
- **HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Списки

Модель объекта список предоставляет следующие свойства:

- **BackgroundColor (long)** – цвет фона элемента управления;
- **Border (short)** – тип границы (0: нет границы, 1: трехмерная граница, 2: простая граница);
- **FontDescriptor (struct)** – структура с подробной информацией об используемом шрифте (в соответствии со структурой `com.sun.star.awt.FontDescriptor`);
- **LineCount (Short)** – число строк элемента управления;
- **MultiSelection (Boolean)** – разрешает множественный выбор элементов;
- **SelectedItems (Array of Strings)** – список выделенных элементов;
- **StringItemList (Array of Strings)** – список всех элементов;
- **ValueItemList (Array of Variant)** – список, содержащий дополнительную информацию для каждого элемента (например, для управления идентификаторами записей данных);
- **Printable (Boolean)** – элемент управления может быть напечатан;
- **ReadOnly (Boolean)** – содержимое элемента управления только для чтения;
- **Enabled (Boolean)** – элемент управления может быть активирован;
- **Tabstop (Boolean)** – элемент управления может быть достигнут через клавишу Tab;
- **TabIndex (Long)** – положение элемента управления в последовательности активации;
- **FontName (String)** – имя типа шрифта;

- **FontHeight (Single)** – высота символа в пунктах (pt);
- **Tag (String)** – строка, содержащая дополнительную информацию, которая может быть сохранена в элементе управления для программно-управляемого доступа;
- **TextColor (Long)** – цвет текста элемента управления;
- **HelpText (String)** – текст подсказки, который отображается, если курсор мыши находится над элементом управления;
- **HelpURL (String)** – URL страницы онлайн справки для соответствующего элемента управления.

Примечание Через свойство `ValueItemList`, списки предоставляют эквивалент свойства VBA, `ItemData`, через который Вы можете управлять дополнительной информацией для отдельных элементов списка.

Кроме того, следующие методы предоставляются через объект представления списка:

- **addItem (Item, Pos)** – вставляет строку, определенную в `Item` в позицию `Pos` в списке;
- **addItems (ItemArray, Pos)** – вставляет элементы, перечисленные в строковом массиве данных `ItemArray` в список в позицию `Pos`;
- **removeItems (Pos, Count)** – удаляет `Count` элементов в позиции `Pos`;
- **selectItem (Item, SelectMode)** – активирует или дезактивирует выделение для элемента, определенного в строке `Item` в зависимости от переменной `SelectMode`;
- **makeVisible (Pos)** – прокручивает область списка так, чтобы элемент, определенный `Pos` был видим.

Формы баз данных

Формы OpenOffice.org могут быть непосредственно связаны с базой данных. Формы, созданные таким образом обеспечивают все функции полного пользовательского интерфейса баз данных, не требуя независимой работы программиста.

Пользователь имеет возможность поиска и выбора таблиц и запросов, а так же изменения записей данных и вставки новых записей данных. OpenOffice.org автоматически гарантирует, что важные данные извлекаются из базы данных, и что любые сделанные изменения будут записаны обратно в базу данных.

Форма базы данных в основном соответствует стандартной форме OpenOffice.org. В дополнение к стандартным свойствам, следующие определенные для базы данных свойства также должны быть установлены в форме:

- **DataSourceName (String)** – имя источника данных (обратитесь к Главе 10; источник данных должен быть глобально создан в OpenOffice.org);
- **Command (String)** - имя таблицы, запроса, или команда выбора SQL, с которой должна быть установлена связь;
- **CommandType (Const)** - определяет, является ли команда таблицей, запросом или командой SQL (значение из списка `com.sun.star.sdb.CommandType`).

Список `com.sun.star.sdb.CommandType` содержит следующие значения:

- **TABLE** – таблица;
- **QUERY** – запрос;
- **COMMAND** – команда SQL.

Поля базы данных назначаются отдельным элементам управления через это свойство:

- **DataField (String)** – имя связанного поля базы данных.

Таблицы

Другой элемент управления предоставляется для работы с базами данных: табличный элемент управления. Он представляет содержание полной таблицы или запроса базы данных. В самом простом сценарии, табличный элемент управления связывается с базой данных с использованием формы автопилота, которая связывает все столбцы с соответствующими полями базы данных в соответствии с пользовательскими спецификациями. Поскольку соответствующий API относительно сложен, мы не будем здесь приводить полное описание API.