**ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»**

# Настольные приложения «МойОфис Текст», «МойОфис Таблица»

# 26.1.0-h

# Справочник макрокоманд на языке программирования Lua

**На 515 листах**

**Версия документа: 1**

**Дата публикации: 23.04.2026**

**Москва**
**2026**

Все упомянутые в этом документе названия продуктов, логотипы, торговые марки и товарные знаки принадлежат их владельцам.

Товарные знаки «МойОфис» и «MyOffice» принадлежат ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ».

Ни при каких обстоятельствах нельзя истолковывать любое содержимое настоящего документа как прямое или косвенное предоставление лицензии или права на использование товарных знаков, логотипов или знаков обслуживания, приведенных в нем. Любое несанкционированное использование этих товарных знаков, логотипов или знаков обслуживания без письменного разрешения их правообладателя строго запрещено.

## СОДЕРЖАНИЕ



## ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе используются следующие сокращения (см. таблицу 1).

Таблица 1 - Сокращения и расшифровки

| Сокращение | Расшифровка |
|---|---|
| ОС | Операционная система |
| ПО МойОфис | Программное обеспечение «МойОфис Стандартный. Домашняя версия» |
| Объектная модель | Совокупность структур данных для управления содержимым текстового или табличного документа |
| EULA | End User License Agreement (пользовательское соглашение) |
| SDK | Software Development Kit (комплект для разработки программного обеспечения) |

# 1 ОБЩИЕ СВЕДЕНИЯ

В данном разделе представлены общие сведения о макрокомандах для редакторов МойОфис.

## 1.1 Назначение

В данном справочнике описываются шаги по созданию и запуску макрокоманд. Также в нем перечислены функциональные возможности вместе с таблицами и функциями на языке программирования Lua, которые могут использоваться для создания макрокоманд в следующих приложениях:

- «МойОфис Текст» – редактор для быстрого и удобного создания и форматирования текстовых документов любой сложности;
- «МойОфис Таблица» – редактор для создания электронных таблиц, ведения расчетов, анализа данных, формирования сводных отчетов и автоматизации обработки данных с использованием макрокоманд.

Подробное описание возможностей этих приложений приведено в соответствующем документе «Функциональные возможности».

## 1.2 Макрокоманды ПО МойОфис

Макрокоманды ПО МойОфис представляют собой программы небольшого размера, с помощью которых автоматизируется выполнение продолжительных или часто встречающихся операций.

При работе с документом периодически возникают ситуации, когда приходится повторять одну и ту же последовательность действий. Для оптимизации работы можно создать макрокоманду, которая будет автоматически выполнять эти действия. Для разработки макрокоманд в ПО МойОфис используется язык программирования Lua.

Справочное руководство по языку программирования Lua опубликовано по ссылкам: https://lua.org.ru (ru), https://devdocs.io (en).

В языке программирования Lua таблицы – это единственная структура данных. Все структуры, которые предлагают другие языки программирования, в том числе массивы, объекты и другие, представлены в языке программирования Lua в виде таблиц.

Структура данных, представляющая текущий открытый документ в ПО МойОфис, в терминах языка программирования Lua является таблицей `DocumentAPI.Document` со своим

набором методов. Иные структуры данных для работы с отдельными ячейками, областями, диаграммами, свойствами текста и т. д. также являются таблицами с необходимым набором полей и методов.

Для управления содержимым документа используется объектная модель документа ПО МойОфис. В данном случае термин «объектная модель» обозначает всю совокупность структур данных для управления содержимым текстового или табличного документа ПО МойОфис.

Для управления текстовым или табличным документом ПО МойОфис используются одни и те же методы объектной модели. К примеру, объект `DocumentAPI.Cell` позволяет управлять как отдельной ячейкой электронной таблицы, так и ячейкой таблицы в текстовом документе.

Для работы с текстом макрокоманды используется редактор макрокоманд в составе текстового или табличного редактора ПО МойОфис. Редактор макрокоманд также предоставляет возможность исполнения макрокоманд и доступ к информации об ошибках их исполнения.

Текст макрокоманды сохраняется в текущий открытый документ ПО МойОфис. Макрокоманда в ПО МойОфис может быть сохранена в документы с форматами DOCX, XODT, ODT, XLSX, XODS, ODS.

### 1.3 Перечень эксплуатационной документации

Настоящий документ содержит описание объектной модели документа ПО МойОфис и примеры ее использования, а также является справочником по возможностям объектной модели редакторов ПО МойОфис.

Вся необходимая информация по использованию макрокоманд в ПО МойОфис приведена в настоящем документе.

# 2 РАБОТА С МАКРОКОМАНДАМИ

В данном разделе описаны действия по созданию, выполнению и отладке макрокоманд в редакторе документов МойОфис.

## 2.1 Редактор макрокоманд

Данный раздел содержит описание окна редактора макрокоманд и доступные в нем действия.

### 2.1.1 Окно редактора макрокоманд

Для работы с макрокомандами используется редактор макрокоманд. Чтобы открыть окно редактора, выберите пункт **Редактор макрокоманд** вкладки **Макрокоманды**.

Окно редактора макрокоманд содержит следующие области (см. Рисунок 1):

1. Список макрокоманд документа.
2. Область ввода текста макрокоманд.
3. Кнопки для создания + и удаления − макрокоманд.
4. Кнопки выполнения (см. раздел Выполнение макрокоманд) и отладки (см. раздел Отладка макрокоманд) макрокоманд. Кнопки становятся активными, изменяя цвет, после ввода текста макрокоманд в области **2** (см. раздел Редактирование макрокоманд).
5. Область вывода результата выполнения макрокоманд, а также отображения информации в процессе отладки макрокоманд.

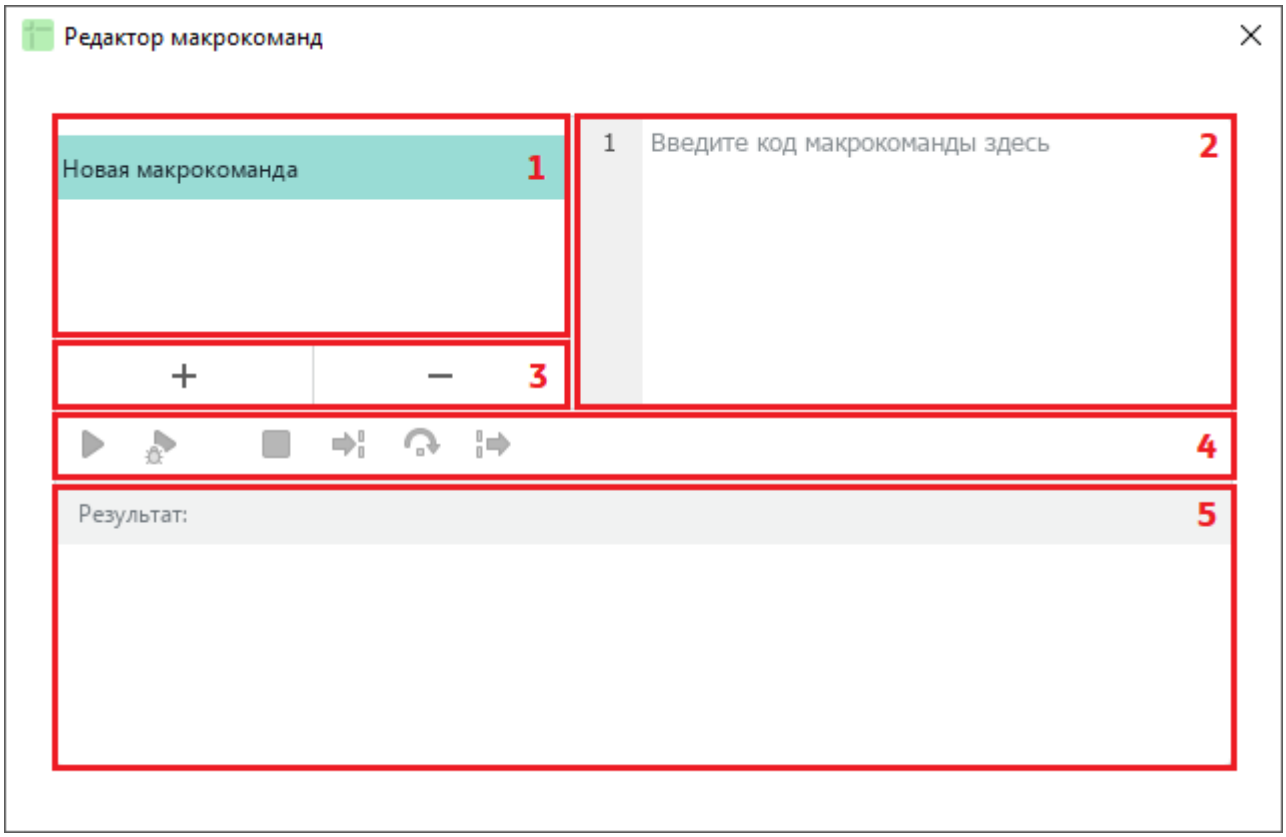



Рисунок 1 – Окно редактора макрокоманд

### 2.1.2 Создание макрокоманд

Для создания макрокоманды выполните следующие действия (см. Рисунок 1):

1. Нажмите кнопку + в области **3**.
2. Введите наименование макрокоманды в соответствующей строке перечня макрокоманд в области **1**. Чтобы сохранить название, нажмите клавишу **Enter** или щелкните мышью по любой области окна **Редактор макрокоманд.**
3. Введите текст макрокоманды в области ввода **2**.

### 2.1.3 Выполнение макрокоманд

Чтобы выполнить макрокоманду, выберите ее в перечне макрокоманд и нажмите кнопку **Выполнить** (см. Рисунок 1).

Результат выполнения макрокоманды отображается в области **5**.

### 2.1.4 Редактирование макрокоманд

Чтобы редактировать макрокоманду, выберите ее в перечне макрокоманд и внесите необходимые изменения в ее текст в области **2** (см. Рисунок 1).

### 2.1.5 Удаление макрокоманд

Чтобы удалить макрокоманду, выберите ее в перечне макрокоманд и нажмите кнопку — в области **3** (см. Рисунок 1).

### 2.1.6 Отладка макрокоманд

Для отладки макрокоманды выполните следующие действия:

1. Выберите наименование макрокоманды в перечне макрокоманд.
2. Установите (при необходимости) в тексте макрокоманд точки останова отладчика, щелкнув мышью справа от номера строки макрокоманды. Строка точки останова будет отмечена значком . Для удаления точки останова щелкните мышью на значок .
3. Нажмите кнопку **Отладить**. Запустится режим отладки и окно редактора макрокоманд изменит свой вид (см. Рисунок 2).

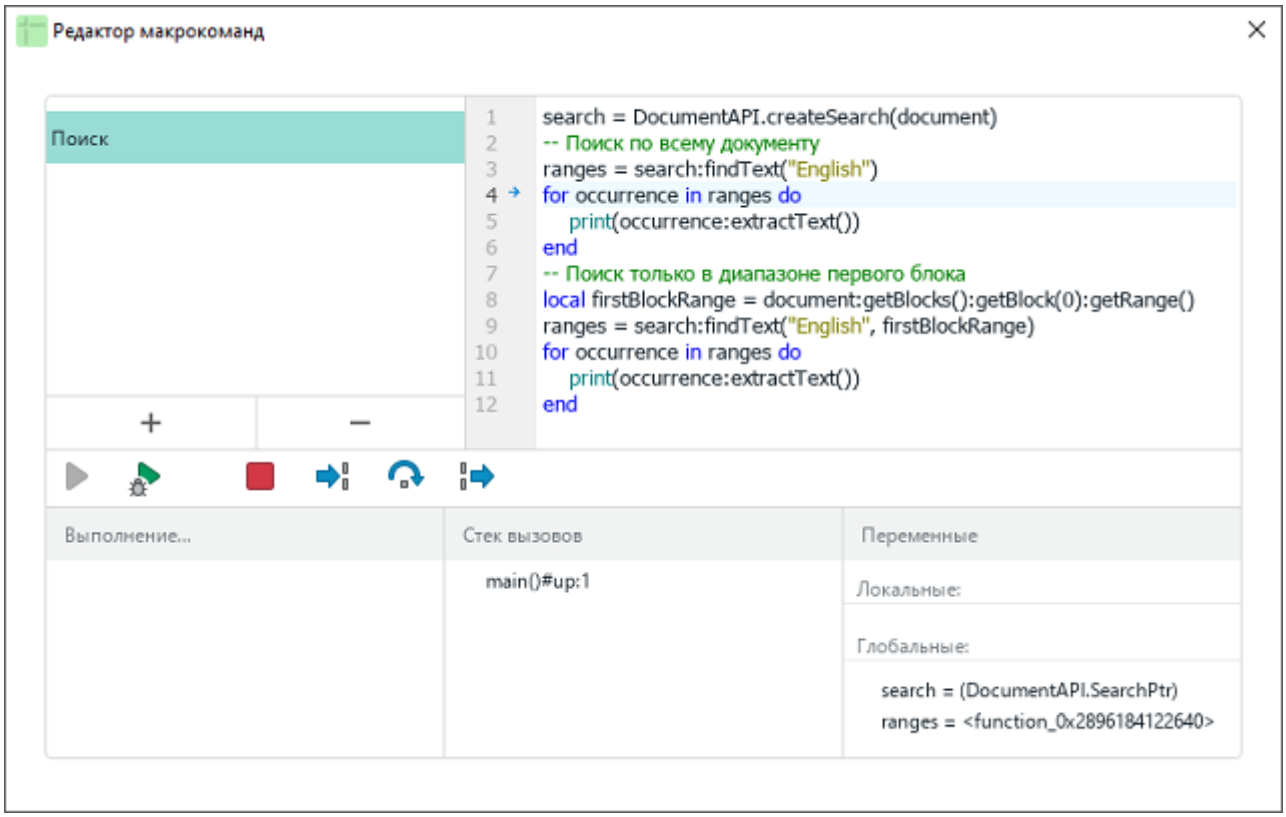



Рисунок 2 – Окно редактора макрокоманд в начале отладки

Процесс отладки макрокоманды остановится на первой точке останова. Если точки останова отсутствуют, то отладка начнется с остановки на первой строке макрокоманды.

Для продолжения отладки нажмите одну из следующих кнопок: (см. Рисунок 2):

– – для выполнения одного шага отладки или захода в тело функции, если таковая есть в текущей позиции отладки;

– – для выполнения одного шага отладки без захода в тело функции;

– – для продолжения выполнения макрокоманды до момента выхода из функции, в которой отладчик находится в текущей позиции.

Для прерывания процесса отладки нажмите кнопку (см. Рисунок 2), отладка прервется и на экран будет выведено сообщение: «Выполнение макрокоманды прервано пользователем».

В процессе отладки в нижней части окна редактора макрокоманд отображаются следующие области (см. Рисунок 3):

- **Выполнение...** – окно для вывода сообщений во время отладки, например, командой print;
- **Стек вызовов** – окно стека вызовов;
- **Переменные** – окно вывода значений локальных и глобальных переменных, доступных на текущем шаге выполнения макрокоманды. Если отображаемая переменная представляет из себя таблицу или массив, то при нажатии кнопки ▸, расположенной рядом с именем переменной, доступен просмотр содержимого переменной в развернутом виде.

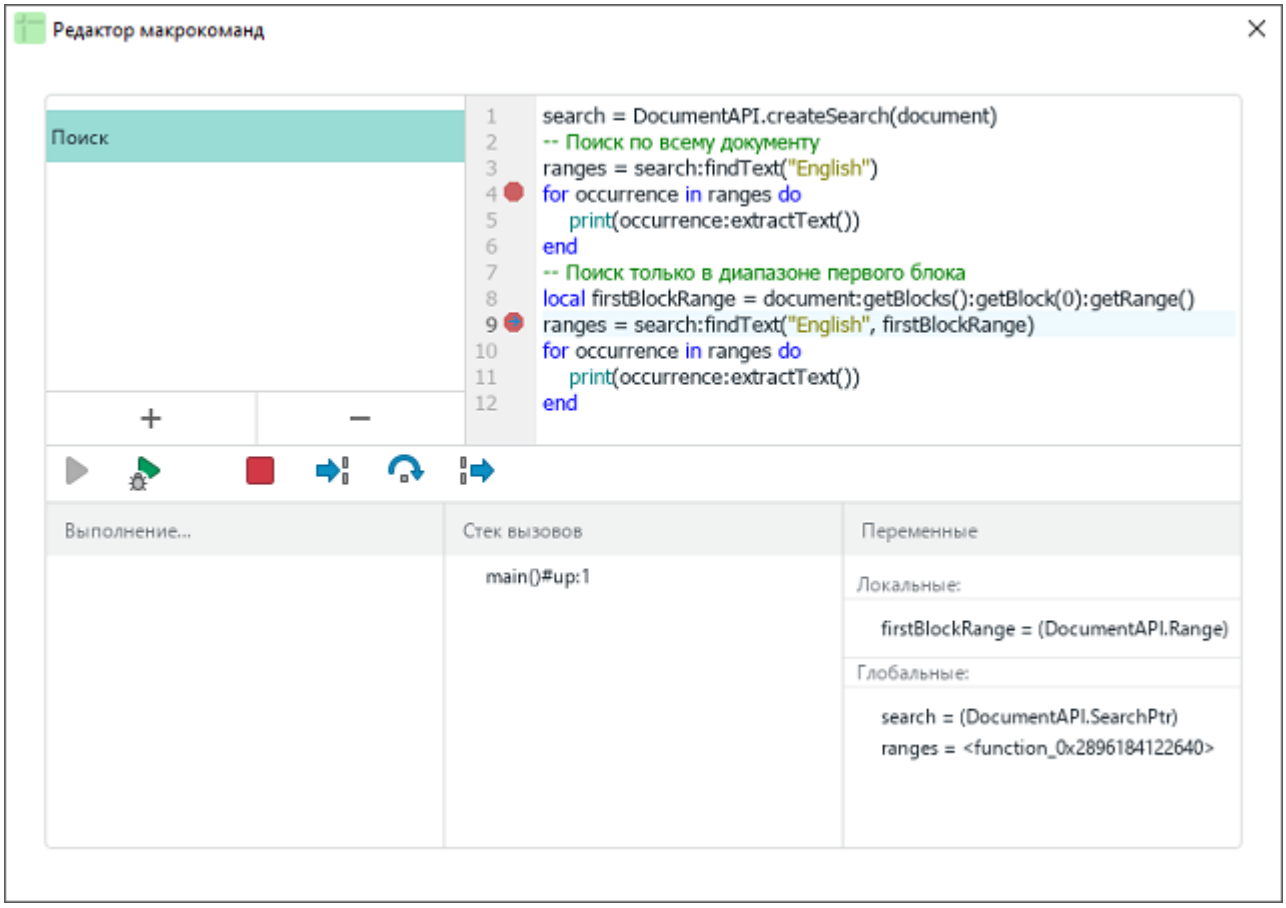



Рисунок 3 – Окно редактора макрокоманд в процессе отладки макрокоманд

Отладка завершается при достижении конца макрокоманды.

### 2.2 Пример подготовки и запуска макрокоманды

В данном разделе представлен подробный пример создания и запуска макрокоманды.

### 2.2.1 Редактирование и запуск макрокоманды в текстовом документе

Следующий пример описывает создание и запуск макрокоманды, которая выводит в первой строке текстового документа строку «*HELLO, WORLD!*».

Чтобы создать и запустить макрокоманду, необходимо выполнить следующие действия:

1. Запустить приложение «МойОфис Текст» и открыть редактор макрокоманд. Для этого выбрать пункт **Редактор макрокоманд** вкладки **Макрокоманды**.
2. Нажать кнопку + для создания новой макрокоманды. Указать имя макрокоманды. По умолчанию новой макрокоманде присваивается имя «Без имени».
3. Ввести текст макрокоманды:

```
range = document:getRange()
startPos = range:getBegin()

textProp = range:getTextProperties()
textProp.italic = true
textProp.allCapitals = true
range:setTextProperties(textProp)
startPos:insertText("Hello, World!")
```

4. Запустить макрокоманду нажатием на кнопку **Выполнить**. В случае успешного завершения работы макрокоманды редактор макрокоманд выведет сообщение «Макрос выполнен успешно».
5. Закрыть окно редактора макрокоманд, чтобы увидеть изменения в текстовом документе. В первой строке документа отобразится текст «*HELLO, WORLD!*».
6. Сохранить текстовый документ. Для этого выбрать в командном меню пункт **Файл** > **Сохранить** / **Файл** > **Сохранить как** или нажать сочетание клавиш **Ctrl**+**S**.

При сохранении необходимо выбрать тип файла «Текстовый документ» одного из форматов: DOCX, XODT, ODT (для электронных таблиц - XLSX, XODS, ODS).

### 2.2.2 Описание примера работы макрокоманды

Ниже приведено описание макрокоманды, текст которой приведен в разделе Редактирование и запуск макрокоманды в текстовом документе.

С помощью последовательности вызовов устанавливается курсор в начало документа:

```
range = document:getRange()
startPos = range:getBegin()
```

Таблица DocumentAPI.Document представляет текущий открытый текстовый документ.

Таблица DocumentAPI.Range используется для того, чтобы предоставить доступ к любой части (фрагменту) содержимого документа.

В данном случае переменная range содержит весь документ целиком. Вызов range:getBegin() устанавливает курсор в начало фрагмента, а в данном случае – в начало самого документа.

Следующая последовательность вызовов настраивает форматирование для документа:

```
textProp = range:getTextProperties()
textProp.italic = true
textProp.allCapitals = true
range:setTextProperties(textProp)
```

В результате выполнения range:getTextProperties переменной textProp присваивается экземпляр TextProperties, содержащий настройки форматирования текущего фрагмента документа.

Таблица DocumentAPI.TextProperties позволяет управлять такими характеристиками как наименование и размер шрифта, цвет, начертание и т.п.

В данном примере устанавливаются две настройки форматирования:

- свойство textProp.italic принимает значение **true**, что равносильно нажатию кнопки K (**Курсив**) в пользовательском интерфейсе текстового редактора;
- свойство textProp.allCapitals принимает значение **true**, что равносильно нажатию кнопки AB (**Все прописные**) в пользовательском интерфейсе текстового редактора.

Следующий вызов range:setTextProperties(textProp) применяет новые настройки форматирования для документа. Теперь эти настройки форматирования будут применяться автоматически для вводимого текста.

Последний вызов вставляет в начало документа текст «Hello, World!»:

```
startPos:insertText("Hello, World!")
```

При вставке текста автоматически применяются настройки форматирования, и итоговый текст отображается как «*HELLO, WORLD!*» прописными буквами курсивом.

### 2.3 Преобразование макрокоманд на языке программирования VBA

Макрокоманды для пакета Microsoft Office, написанные на языке программирования VBA, предназначены для выполнения в пакете Microsoft Office под управлением операционной системы Microsoft Windows и несовместимы с макросами редакторов МойОфис.

Однако, большинство макрокоманд на языке программирования VBA возможно реализовать на языке программирования Lua с использованием объектной модели ПО МойОфис.

ПО МойОфис является кроссплатформенным решением (решением не только для работы в операционных системах семейства Microsoft Windows), поэтому при реализации макрокоманд на основе языка программирования VBA следует принимать во внимание следующие ограничения, связанные с операционными системами семейства Microsoft Windows:

- невозможность обращения к внешним приложениям с помощью технологий Component Object Model (COM);
- невозможность использования внешних динамических библиотек DLL.

Также в настоящее время в редакторе макрокоманд ПО МойОфис существует временное ограничение по работе с визуальными элементами, такими как выпадающие списки, переключатели и некоторыми другими.

В редакторе МойОфис Таблица представлена бета-версия VBA интерпретатора, который позволяет вам конвертировать код VBA макрокоманд на язык программирования Lua. Подробную информацию о работе конвертора и существующих ограничениях вы можете найти в разделе **Средства автоматизации > Макрокоманды > Макрокоманды на VBA** документа о работе в приложении МойОфис Таблица.

## 3 ОБЪЕКТНАЯ МОДЕЛЬ МОЙОФИС

МойОфис SDK предоставляет разработчику возможности для управления содержимым текстового и табличного документа. Функции управления сосредоточены в следующей группе таблиц:

- `DocumentAPI` – содержит таблицы и функции для представления всех элементов документа, которые поддерживает МойОфис: абзацы, таблицы, рисунки, колонтитулы, операции для работы с текстом, цветом и т.д;
- `EditorAPI` – содержит функции для управления редакторами МойОфис Текст и МойОфис Таблица;

Вышеописанные таблицы составляют объектную модель МойОфис SDK (см. Рисунок 4).

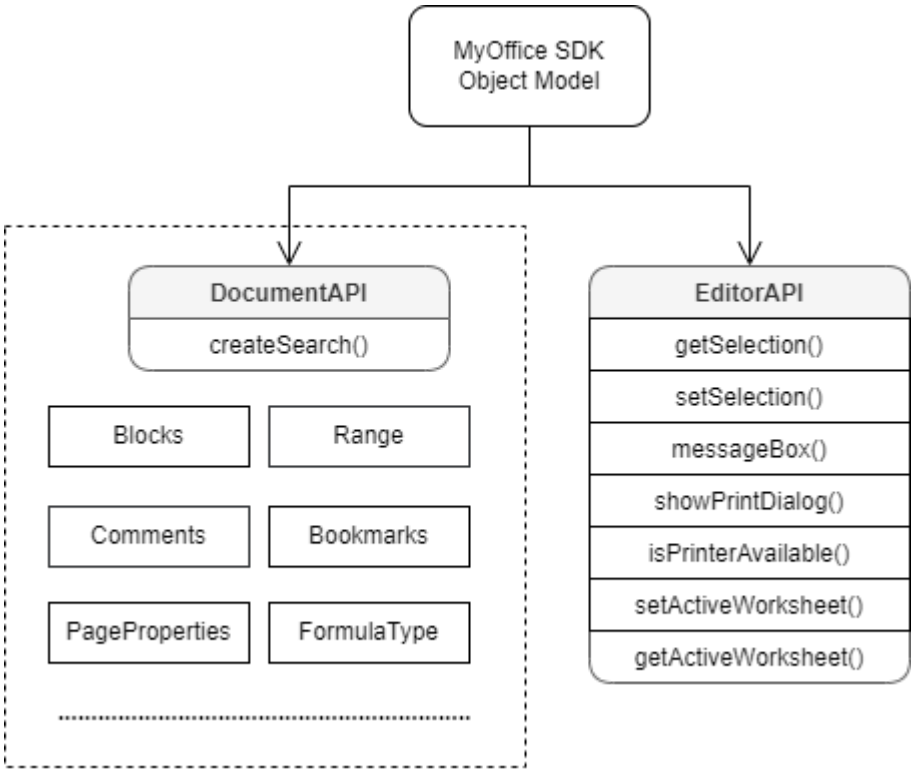



Рисунок 4 – Объектная модель МойОфис SDK.

# 4 РАБОТА С ДОКУМЕНТАМИ

В данном разделе представлены возможные сценарии использования макрокоманд для взаимодействия с текстовыми и табличными документами.

## 4.1 Работа с текстовым документом

Данный раздел содержит специфичные для текстовых документов действия, доступные с помощью макрокоманд.

### 4.1.1 Разделы (секции) документа

На рисунке 5 изображена объектная модель таблиц, относящихся к работе с секциями текстового документа.

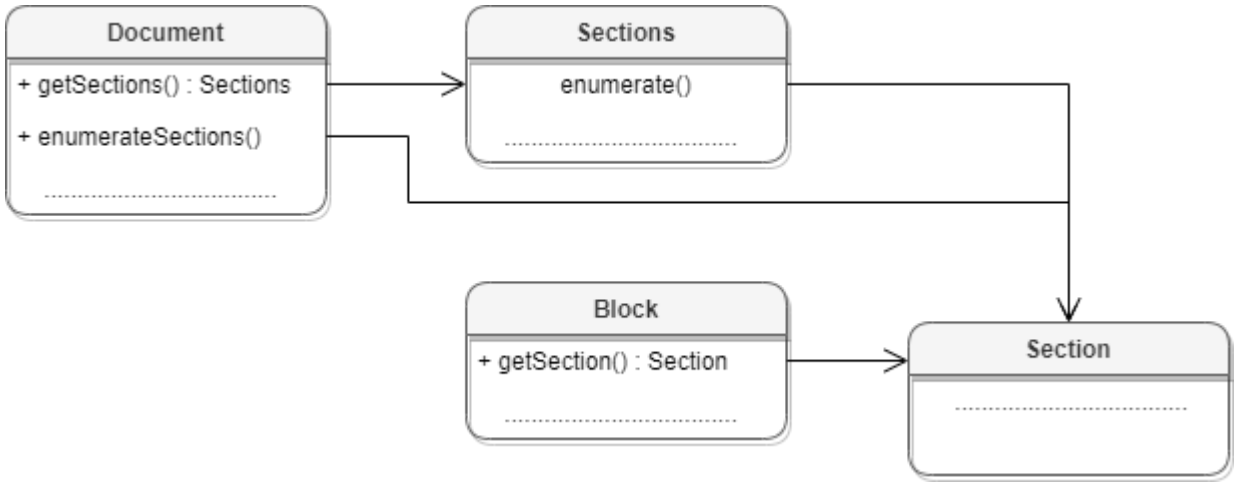



Рисунок 5 – Объектная модель таблиц для работы с секциями

Секция в текстовом документе - это раздел, который содержит страницы с одинаковыми параметрами, а также одинаковыми верхними и нижними колонтитулами.

Доступ к секциям текстового документа может быть осуществлен одним из следующих способов:

- получение таблицы `DocumentAPI.Sections` с помощью вызова `document:getSections()`;
- перечисление всех доступных секций `DocumentAPI.Section` с помощью вызова `document:enumerateSections()`;

– получение секции DocumentAPI.Section вызовом метода Block.getSection() для блока, который входит в секцию.

**Примеры**

```
local sections = document:getSections()
for section in sections:enumerate() do
    local properties = section:getPageProperties()
    print(properties.width)
    print(properties.height)
end
```

```
local sections = document:enumerateSections()
for section in sections do
  print(section:getPageProperties().width)
end
```

```
local section = document:getBlocks():getBlock(0):getSection()
local properties = section:getPageProperties()
print(properties.width)
```

#### 4.1.1.1 Работа с колонтитулами раздела

Для получения колонтитулов раздела следует использовать методы Section::getHeaders() или Section::getFooters().

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    if (header:getType() == DocumentAPI.HeaderFooterType_Header) then
       print("Header") else print("Footer")
    end
end
local footers = section:getFooters()
for footer in footers:enumerate() do
    if (footer:getType() == DocumentAPI.HeaderFooterType_Header) then
       print("Header") else print("Footer")
    end
end
```

#### 4.1.1.2 Управление ориентацией и свойствами страниц раздела

Для установки ориентации страницы можно использовать метод Section::setPageOrientation() секции, полученной из блока документа.

```
local section = document:getBlocks():getBlock(0):getSection()
section:setPageOrientation(DocumentAPI.PageOrientation_Landscape)
local orientation = section:getPageOrientation()
print(orientation)
```

Установить необходимые значения высоты и ширины страниц раздела документа можно с помощью метода Section::setPageProperties(), задав необходимые значения в структуре PageProperties.

```
local section = document:getBlocks():getBlock(0):getSection()
local properties = section:getPageProperties()
properties.width = 100
properties.height = 200
properties.margins.left = 10
section:setPageProperties(properties)
```

Ориентация страниц может быть установлена для каждого раздела документа. Список разделов документа может быть получен посредством метода Document::enumerateSections().

```
local sections = document:enumerateSections()
for section in sections do
  print(section:getPageProperties().width)
end
```

Ориентация страниц объекта Section может быть получена с использованием метода Section::getPageOrientation().

```
local section = document:getBlocks():getBlock(0):getSection()
local orientation = section:getPageOrientation()
print(orientation)
```

Свойства страниц объекта Section могут быть получены с использованием метода Section::getPageProperties().

```
local section = document:getBlocks():getBlock(0):getSection()
local properties = section:getPageProperties()
print(properties.width)
print(properties.height)
```

```
print(properties.margins.left)
print(properties.margins.top)
```

### 4.1.2 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены на страницах документа. Доступ к объектам Table осуществляется из Blocks (см. Рисунок 6).

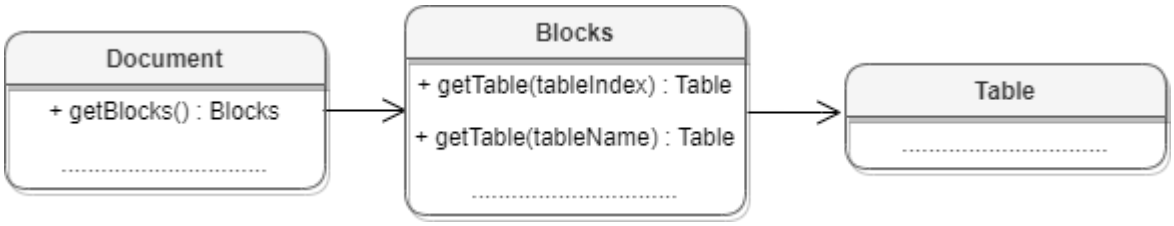



Рисунок 6 – Объектная модель для работы с таблицами

Для работы с таблицами доступны следующие операции:

– перечисление таблиц документа;
– получение таблицы документа;
– вставка таблицы в позицию документа;
– переименование таблицы;
– удаление таблицы.

Ниже приведены примеры работы с таблицами в текстовых документах:

#### Перечисление таблиц документа

Для перечисления таблиц текстового документа используется метод Blocks:enumerateTables().

```
for table in document:getBlocks():enumerateTables() do
    print(table:getName())
end
```

#### Получение таблицы текстового документа

Для получения таблицы текстового документа используется метод Blocks:getTable(). В качестве аргумента используется индекс или имя таблицы.

```
local table = document:getBlocks():getTable(0)
```

```
local table = document:getBlocks():getTable("Sheet1")
```

**Вставка таблицы в текстовый документ**

Для вставки таблицы в текстовый документ используется метод Position:insertTable(). В качестве аргументов передаются размеры и имя таблицы.

```
local rng = document:getRange()
local begin_pos = rng:getBegin()
t = begin_pos:insertTable(3, 3, "Table")
```

**Вставка текста в таблицу текстового документа**

Для вставки текста в таблицу текстового документа используется метод Position:insertText().

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A2")
local range = cell:getRange()
local pos = range:getBegin()
pos:insertText("Привет, Мир!")
```

**Переименование таблицы**

Для переименования таблицы используется метод Table:setName(). В текстовых документах наименование таблицы нигде не отображается, но в дальнейшем его можно использовать для доступа к таблице по имени.

```
local tbl = document:getBlocks():getTable(0)
tbl:setName("Первый")
```

**Удаление таблицы**

Для удаления таблицы используется метод Table:remove().

```
local tbl = document:getBlocks():getTable(0)
tbl:remove()
```

### 4.1.3 Работа с закладками

Основной таблицей для работы с закладками является DocumentAPI.Bookmarks. Список закладок документа возвращает метод document:getBookmarks(). Метод Bookmarks.getBookmarkRange() возвращает диапазон текста, метод Bookmarks.removeBookmark() удаляет закладку по имени. Для создания закладки используется метод Position.insertBookmark().

Доступны следующие операции с закладками:

- вставка закладки в указанное местоположение;
- удаление закладки с заданным именем;
- поиск закладки по имени;
- замена текстового содержимого закладки;
- вставка текста в закладку;
- удаление содержимого закладки;
- получение текстового содержимого закладки;
- вставка таблицы в закладку.

**Вставка закладки в указанное местоположение**

```
-- Вставка новой закладки с именем Signers в начало документа
local sig_pos = document:getRange():getBegin()
sig_pos:insertBookmark("Signers")
```

**Удаление закладки с заданным именем**

```
-- Удаление закладки "Signers"
document:getBookmarks():removeBookmark("Signers")
```

**Поиск закладки по имени**

```
-- Поиск закладки "Signers" по имени
local sig_rng = document:getBookmarks():getBookmarkRange("Signers")
```

**Замена текстового содержимого закладки**

```
-- Замена содержимого закладки на текст "Lua"
local bookmarks = document:getBookmarks()
local bookmarkRange = bookmarks:getBookmarkRange( "bm_1" )
bookmarkRange:replaceText("Lua")
```

**Вставка текста в закладку**

```
sig_rng:getBegin():insertText("Лист")
```

**Удаление содержимого закладки**

```
sig_rng:removeContent()
```

**Получение текстового содержимого закладки**

```
local msg = sig_rng:extractText()
print(msg)
```

**Вставка таблицы в закладку**

```
-- Вставка таблицы в закладку "Signers"
local tbl_id = sig_rng:getEnd():insertTable(3, 3, "signers_list")
```

### 4.1.4 Рецензирование документов

Средства рецензирования документа доступны в текстовом редакторе, они позволяют выполнять следующие действия:

- помечать изменения, вносимые пользователем в текстовый документ (DocumentAPI.TrackedChange);
- ассоциировать текстовый комментарий с фрагментом текстового документа (DocumentAPI.Comments).

Данные механизмы используются на стадии рецензирования или согласования документа с последующим внесением замечаний. Функции объектной модели для работы со средствами рецензирования позволяют получить детальную информацию о каждом изменении: автор изменения, дата внесения изменения, оригинальный текст, измененный текст.

Для включения или отключения режима рецензирования используется метод document:setChangesTrackingEnabled(). Для проверки текущего статуса данного режима используется метод document:isChangesTrackingEnabled().

**Пример**

```
document:setChangesTrackingEnabled(true)
print(document:isChangesTrackingEnabled())
```

Инструменты рецензирования применяются к диапазону документа, по этой причине методы доступа к ним находятся в таблице DocumentAPI.Range (см. Рисунок 7).

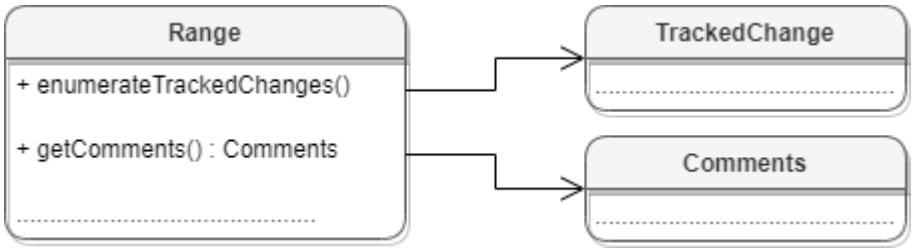



Рисунок 7 – Инструменты рецензирования документа

### 4.1.5 Работа с графическими объектами в текстовом документе

Редактор текста МойОфис поддерживает несколько типов графических объектов со схожим поведением: изображения (`DocumentAPI.Image`) и фигуры (`DocumentAPI.Shape`).

Объектная модель документа в части управления изображениями развивается и дополняется возможностями. Доступны следующие операции:

– Перечисление графических объектов, находящихся в документе, определение их типа и геометрических размеров.
– Вставка изображений в текстовый документ. Место вставки определяется типом `Position`.
– Перемещение графических объектов, изменение их размеров и масштаба.

**Перечисление графических объектов в текстовом документе**

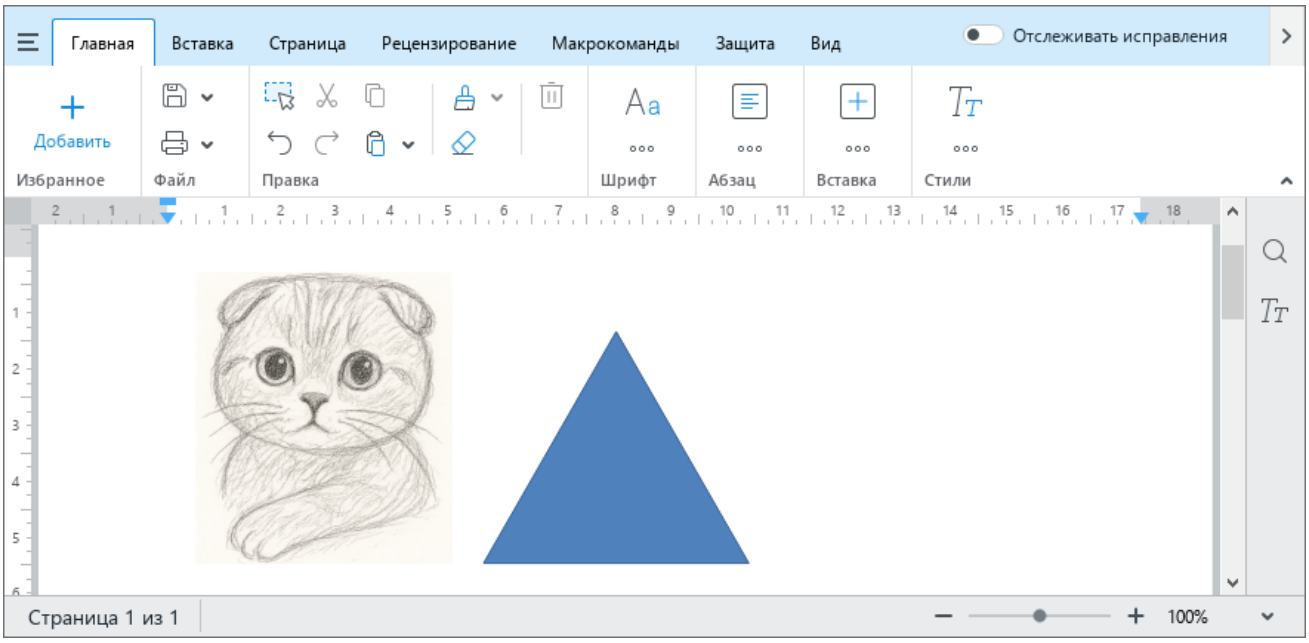



Рисунок 8 – Графические объекты в текстовом документе

**Вариант 1: перечисление графических объектов в текстовом документе**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
   image = mediaObject:toImage()
   if image then
     print("Image:", image)
   else
     print("Shape:", mediaObject)
   end
end
```

**Вывод**

```
> Image: <userdata of type 'CO::API::Document::Image *' ..... >
> Shape: <userdata of type 'CO::API::Document::MediaObject *' ..... >
```

**Вариант 2: перечисление изображений в текстовом документе**

```
local images = document:getRange():getImages()
for image in images:enumerate() do
   print("Image:", image)
end
```

**Вывод**

```
> Image: <userdata of type 'CO::API::Document::Image *' ..... >
```

**Перечисление графических объектов в таблицах текстового документа**

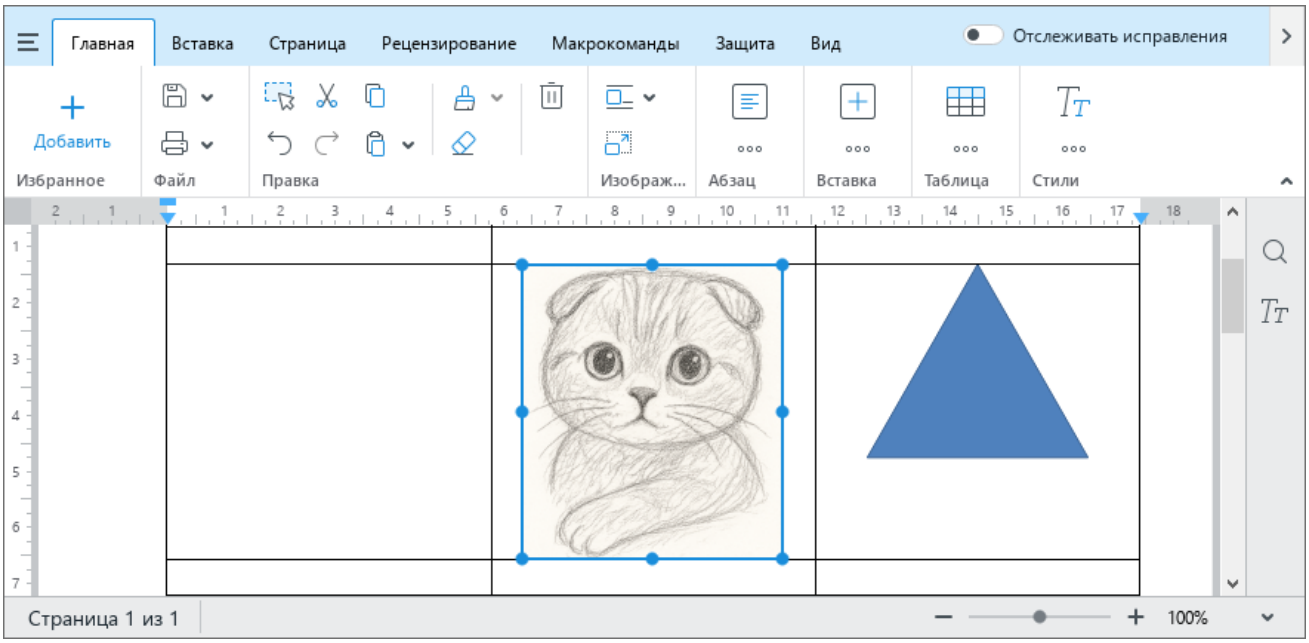



Рисунок 9 – Графические объекты в таблице текстового документа

**Вариант 1: перечисление графических объектов в таблице текстового документа**

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
   local image = mediaObject:toImage()
   if image then
      print("Image:", image)
   else
     print("Shape", mediaObject)
```

```
   end
end
```

**Вывод**

```
> Image: <userdata of type 'CO::API::Document::Image *' ..... >
> Shape: <userdata of type 'CO::API::Document::MediaObject *' ..... >
```

**Вариант 2: перечисление изображений в таблице текстового документа**

```
images = document:getBlocks():getTable(0):getImages()
for image in images:enumerate() do
    print("Image:", image)
end
```

**Вывод**

```
> Image: <userdata of type 'CO::API::Document::Image *' ..... >
```

Стоит обратить внимание на то, что графический объект обладает свойством frame, описывающим позицию, размеры и выравнивание. Данное свойство возвращается посредством методов `MediaObject:getFrame()` или `Image:getFrame()`. В текстовом документе данный метод возвращает тип `DocumentAPI.InlineFrame`, в табличном документе возвращается `DocumentAPI.AbsoluteFrame`.

**Вставка изображения в текстовый документ**

Для вставки изображения используется метод `Position:insertImage()`.

**Вариант 1: вставка изображения в позицию диапазона текстового документа**

```
local range = document:getRange()
local imageSize = DocumentAPI.SizeU(50, 50)
local image = range:getBegin():insertImage("C://Tmp//123.jpg", imageSize)
```

**Вариант 2: вставка изображения в ячейку таблицы текстового документа**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")
local range = cell:getRange()
local imageSize = DocumentAPI.SizeU(50, 50)
local image =
range:getBegin():insertImage("https://www.images.ru/images/fish.jpg", imageSize)
```

**Вставка изображения в колонтитулы текстового документа**

```
local sections = document:enumerateSections()
for section in sections do
    local footers = section:getFooters():enumerate()
    for footer in footers do
        local pos = footer:getRange():getBegin()
        pos:insertImage("logo.jpg", DocumentAPI.SizeU(100, 50))
    end
end
```

### 4.1.6 Работа с элементами управления

К элементам управления относятся следующие объекты: "Флажок" (`CheckBoxControl`), "Поле ввода" (`InputFieldControl`), "Выбор даты" (`DatePickerControl`) и "Выпадающий список" (`DropListControl`). Они могут быть расположены в текстовом документе или его шаблоне.

Используйте метод `Document:getContentControls()` чтобы получить элементы управления из текущего документа:

```
local controls = document:getContentControls()
```

Метод `ContentControls:findByTitle(string)` позволяет получить элемент управления по его названию:

```
local textField = controls:findByTitle("input")
```

Чтобы взаимодействовать со значениями элементов управления, преобразуйте полученный объект `ContentControl` в объект элемента управления. Это можно сделать с помощью методов `toCheckBox()`, `toInputField()`, `toDatePicker()` и `toDropList()`:

```
local checkBox = controls:findByTitle("check"):toCheckBox()
local inputField = controls:findByTitle("input"):toInputField()
local startDate = controls:findByTitle("date"):toDatePicker()
local comboBox = controls:findByTitle("select"):toDropList()
```

У каждого объекта элемента управления есть методы для получения и задания его значения (`getValue()` и `setValue()`):

```
checkBox:setValue(not(checkBox:getValue()))
```

Выпадающий список дополнительно содержит метод `DropListControl:getChoices()`, который возвращает элементы выпадающего списка.

### 4.2 Работа с табличным документом

Данный раздел содержит специфичные для табличных документов действия, доступные с помощью макрокоманд.

#### 4.2.1 Работа с текстом в табличном документе

**Вставка текста в табличный документ**

Для вставки текста в ячейку табличного документа используется метод `Position::insertText()`.

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A3")
local range = cell:getRange()
local pos = range:getBegin()
pos:insertText("Привет, Мир!")
```

#### 4.2.2 Копирование ячеек в табличном документе

Для копирования / переноса группы ячеек вместе с их содержимым и свойствами используются методы `CellRange.copyInto()` и `CellRange.moveInto()`.

Следующий пример копирует диапазон ячеек `"A1:B2"` между листами документа:

```
local leftTopCellPosition = DocumentAPI.CellPosition(0, 0)
local rightBottomCellPosition = DocumentAPI.CellPosition(1, 1)
local srcCellRangePosition = DocumentAPI.CellRangePosition(leftTopCellPosition,
rightBottomCellPosition)

local strTargetRange = "A1:B2"

local sourceList = document:getBlocks():getTable(0)
local targetList = document:getBlocks():getTable(1)
local sourceRange = sourceList:getCellRange(srcCellRangePosition)
local destRange = targetList:getCellRange(strTargetRange)
sourceRange:copyInto(destRange)
```

Для перемещения ячеек следует воспользоваться методом `CellRange.moveInto()`:

```
sourceRange:moveInto(destRange)
```

### 4.2.3 Диаграммы

Работа с диаграммами реализована только в табличных документах. На рисунке 10 изображена объектная модель таблиц, относящихся к работе с диаграммами.

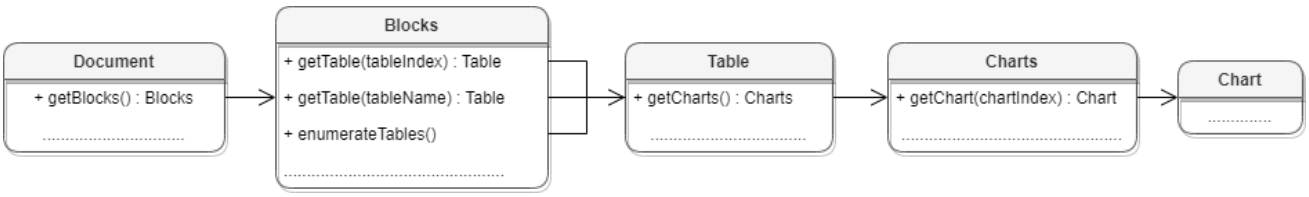



Рисунок 10 – Объектная модель таблиц для работы с диаграммами

Доступ к списку диаграмм производится через таблицу DocumentAPI.Table, соответствующую листу табличного документа.

**Пример**

```
local sheetDocumentPage = document:getBlocks():getTable(0)
local charts = sheetDocumentPage:getCharts()
print(charts:getChartsCount())
```

Создание и удаление диаграмм в текущей версии не поддерживается.

### 4.2.4 Работа с формулами

Метод Cell:setFormula позволяет поместить формулу в ячейку таблицы:

```
local firstSheet = document:getBlocks():getTable(0)
firstSheet:getCell("A3"):setFormula("=SUM(A1:A2)")
```

Также при создании формулы можно использовать именованные диапазоны для обозначения группы ячеек.

Из ячейки, в которой находится формула, можно получить текст формулы (Cell:getFormulaAsString) или результат вычисления (Cell:getRawValue или Cell:getFormattedValue):

```
firstSheet:getCell("B1"):getFormulaAsString()    -- =AVERAGE(B:B)
firstSheet:getCell("B1"):getRawValue()           -- 1.5
firstSheet:getCell("B1"):getFormattedValue()     -- 150.0%
```

По умолчанию формулы пересчитываются автоматически при изменении значений ячеек, указанных в формуле. Для увеличения производительности при работе с таблицами с большим объемом ячеек, можно отключить автоматический пересчет с помощью метода Document:setCalculationMode. Узнать текущее состояние автоматического пересчета можно используя метод Document:getCalculationMode:

```
if document:getCalculationMode() == DocumentAPI.CalculationMode_Auto then
    document:setCalculationMode(DocumentAPI.CalculationMode_Manual)
end
```

Также формулы пересчитываются при сохранении документа. Для того чтобы изменить это поведение, вы можете использовать метод Document:setCalculatedOnSave. Текущее его состояние можно узнать используя метод Document:isCalculatedOnSave:

```
if document:isCalculatedOnSave() then
    document:setCalculatedOnSave(false)
end
```

Если автоматический пересчет формул отключен, вы можете обновить значения всех формул в документе с помощью метода Document:calculateOutdatedFormulas или использовать метод `calculate()` объектов Table, CellRange или Cell для пересчета формул, находящихся в определенном диапазоне:

```
// пересчет всего документа:
document:calculateOutdatedFormulas()
// пересчет листа документа:
firstSheet:calculate()
// пересчет заданного диапазона:
firstSheet:getCellRange("A1:B3"):calculate()
// пересчет заданной ячейки:
firstSheet:getCell("B1"):calculate()
```

### 4.2.5 Проверка данных

Табличный редактор позволяет настроить проверку значений ячеек, чтобы разрешить ввод только корректных данных и исключить ошибки. Также вы можете проверить правильность уже введенных значений. Поддерживаются следующие виды проверок: проверка по списку допустимых значений и проверка данных в формате **Дата**.

### Проверка данных по списку значений

Данная проверка сравнивает значение ячейки с заранее определенным списком допустимых значений. А также позволяет показать выпадающий список с доступными значениями. Для применения проверки по списку, выполните следующие действия:

1. Создайте экземпляр таблицы `DataValidation`.
2. Вызовите метод `DataValidation:setType()` с параметром `DataValidationType_List`, чтобы создать проверку по списку значений.
3. Передайте список значений в метод `DataValidation:setFormula1()`. Метод принимает значения в виде строки, разделенной точкой с запятой (;), а также формулы, именованные диапазоны и адреса.
4. Вызовите метод `DataValidation:setShowDropDown()` с параметром `true` для показа выпадающего списка при вводе данных в ячейку.
5. Примените проверку данных к диапазону ячеек с помощью метода `CellRange:setDataValidation()`.

```
local dvList = DocumentAPI.DataValidation()
dvList:setType(DocumentAPI.DataValidationType_List)
dvList:setFormula1("UK; Italy; Germany; Austria; Brazil")
dvList:setShowDropDown(true)

cellRange:setDataValidation(dvList)
```

### Проверка данных в формате Дата

Данная проверка сравнивает введенные даты. Для применения проверки, выполните следующие действия:

1. Создайте экземпляр таблицы `DataValidation`.
2. Вызовите метод `DataValidation:setType()` с параметром `DataValidationType_Date` для создания проверки дат.
3. Используйте метод `DataValidation:setOperator()`, чтобы задать оператор сравнения.
4. Передайте дату для сравнения в метод `DataValidation:setFormula1()`. Метод принимает значения в виде строки в поддерживаемом формате, а также формулы, именованные диапазоны и адреса.
5. Если выбран оператор сравнения `Between` или `NotBetween`, задайте вторую дату для промежутка с помощью метода `DataValidation:setFormula2()`.

6. Используя метод CellRange:setDataValidation() примените проверку данных к диапазону ячеек.

```
local dvDate = DocumentAPI.DataValidation()
dvDate:setType(DocumentAPI.DataValidationType_Date)
dvDate:setOperator(DocumentAPI.DataValidationOperator_Between)
dvDate:setFormula1("01.06.2024")
dvDate:setFormula2("31.08.2024")

cellRange:setDataValidation(dvDate)
```

### Отображение сообщений об ошибках

Добавьте визуальное предупреждение о вводе недопустимых значений:

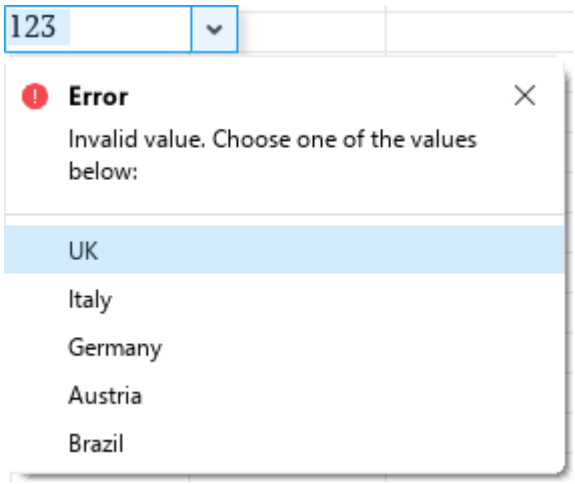



Рисунок 12 – Ошибка проверки данных

1. Вызовите метод DataValidation:setShowErrorMessage() с параметром `true` для показа сообщения об ошибке.
2. Задайте поведение редактора при вводе недопустимых значений с помощью метода DataValidation:setErrorStyle(): запретить ввод недопустимых значений (DataValidationErrorStyle_Stop) или разрешить ввод после показа предупреждения (DataValidationErrorStyle_Warning).
3. Передайте заголовок сообщения в метод DataValidation:setErrorTitle().
4. Используйте метод DataValidation:setErrorMessage() для задания сообщения об ошибке.

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки сообщения об ошибке:
validation:setShowErrorMessage(true)
validation:setErrorStyle(DocumentAPI.DataValidationErrorStyle_Stop)
```

```
validation:setErrorTitle("Error")
validation:setErrorMessage("Invalid value. Choose one of the values below:")

cellRange:setDataValidation(validation)
```

### Отображение подсказок

Добавьте сообщение, которое будет показываться во время редактирования ячейки с проверкой:

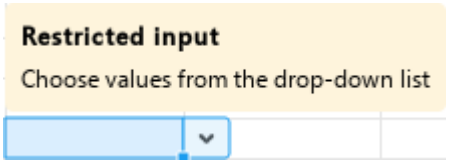



Рисунок 14 – Подсказка при вводе значения в ячейку с проверкой

1. Вызовите метод `DataValidation:setShowInputMessage()` с параметром `true` для показа подсказки.
2. Передайте заголовок подсказки в метод `DataValidation:setPromptTitle()`.
3. Используйте метод `DataValidation:setPrompt()` для задания сообщения подсказки.

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки подсказки:
validation:setShowInputMessage(true)
validation:setPromptTitle("Restricted input")
validation:setPrompt("Choose values from the drop-down list")

cellRange:setDataValidation(validation)
```

### Обработка результатов проверки

Для проверки данных в ячейке используйте метод `Cell:checkDataValidation()`. Данный метод возвращает объект `DataValidationResult`, который позволяет определить допустимость текущего значения и примененные настройки проверки данных.

```
if cell:getDataValidation() == nil then
    print("No validation applied")
elseif cell:checkDataValidation():isValid() then
    print("Validation passed")
else
    local validation = cell:checkDataValidation():getDataValidation()
```

```
    print(validation:getErrorMessage())
end
```

#### Получение настроек проверок

Чтобы получить настройки проверки данных для конкретной ячейки, используйте метод `Cell:getDataValidation()`.

#### Удаление проверок

Вы можете использовать метод `CellRange:clearDataValidations()`, чтобы убрать проверку данных из диапазона ячеек.

### 4.2.6 Работа с графическими объектами в табличном документе

Редактор таблиц МойОфис поддерживает несколько типов графических объектов со схожим поведением: изображения (`DocumentAPI.Image`) и фигуры (`DocumentAPI.Shape`).

Объектная модель документа в части управления изображениями развивается и дополняется возможностями. Доступны следующие операции:

- Перечисление графических объектов, находящихся в документе, определение их типа и геометрических размеров.
- Вставка изображений в табличный документ.
- Перемещение графических объектов, изменение их размеров и масштаба.

### Перечисление графических объектов в табличном документе

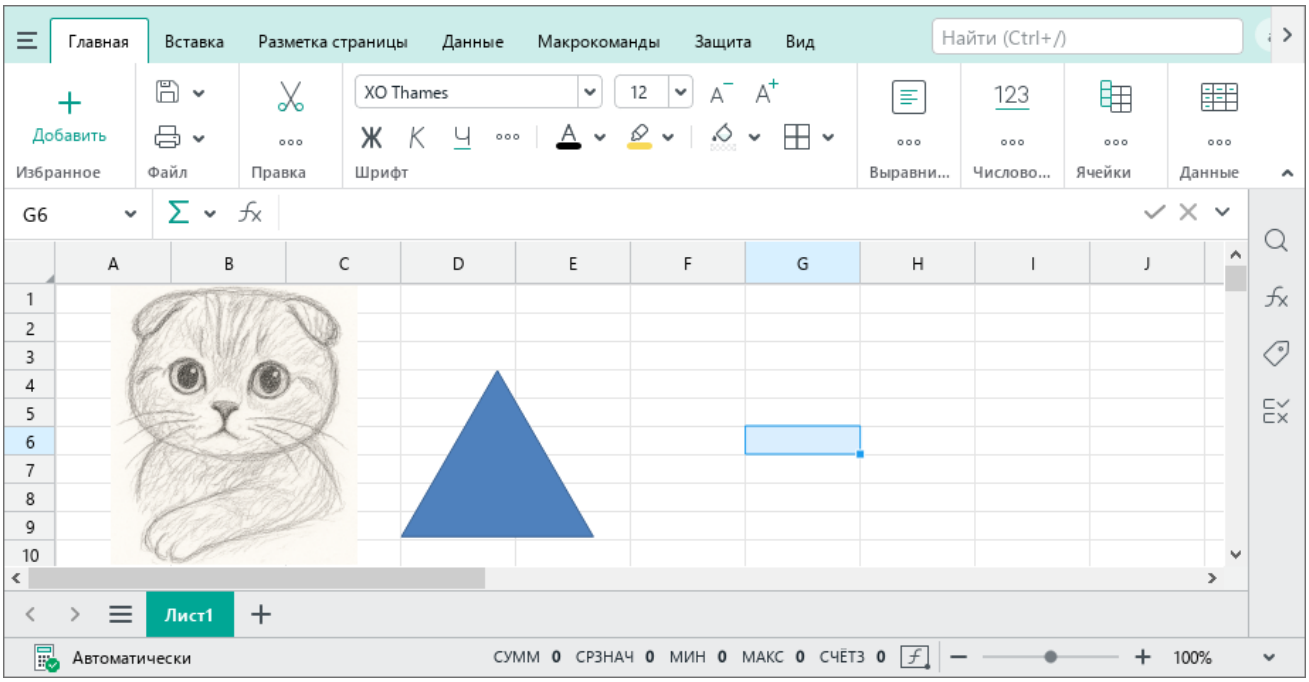



Рисунок 15 – Графические объекты в табличном документе

### Вариант 1: перечисление графических объектов в табличном документе

```
local tbl = document:getBlocks():getTable(0)
local mediaObjects = tbl:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
   image = mediaObject:toImage()
   if image then
      print("Image:", image)
   else
     chart = mediaObject:toChart()
     if chart then
       print("Chart:", chart)
     else
       print("Shape", mediaObject)
     end
   end
end
```

### Вывод

```
> Image: <userdata of type 'CO::API::Document::Image *' ..... >
> Shape: <userdata of type 'CO::API::Document::MediaObject *' ..... >
```

**Вариант 2: перечисление изображений в табличном документе**

```
images = document:getBlocks():getTable(0):getImages()
for image in images:enumerate() do
    print("Image:", image)
end
```

**Вывод**

```
> Image: <userdata of type 'CO::API::Document::Image *' ..... >
```

**Вставка изображения в табличный документ**

Для вставки изображения используется метод `Table:insertImage()`.

```
local sheet = document:getBlocks():getTable(0)
local rect = DocumentAPI.RectU()
rect.topLeft = DocumentAPI.PointU(100, 100)
rect.bottomRight = DocumentAPI.PointU(200, 150)
local insertedImage = sheet:insertImage("image.png", rect)
```

### 4.2.7 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 16).

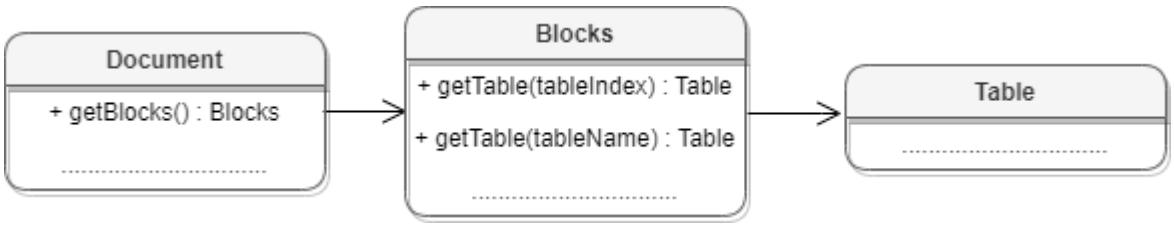



Рисунок 16 – Объектная модель для работы с таблицами

**Получение листа табличного документа**

Для получения таблицы применяется метод `Blocks::getTable()`. В качестве аргумента используется индекс или имя листа документа.

```
local table = document:getBlocks():getTable(0)
```

```
local table = document:getBlocks():getTable("Sheet1")
```

**Перечисление страниц табличного документа**

Для перечисления листов табличного документа используется метод Blocks::getTablesEnumerator().

```
for table in document:getBlocks():enumerateTables() do
    print(table:getName())
end
```

Также доступен вариант перечисления листов документа посредством использования метода Blocks::enumerate() с дальнейшим преобразованием блока в таблицу (Block::toTable()).

```
for block in document:getBlocks():enumerate() do
    local table = block:toTable()
    print(table:getName())
end
```

**Вставка страницы в табличный документ**

Для вставки листа (страницы) в табличный документ используется метод Position::insertTable(). В качестве аргументов передаются размеры и имя таблицы.

```
local range = document:getRange()
local end_pos = range:getEnd()
t = end_pos:insertTable(3, 3, "Table")
```

**Активация страницы, получение активной страницы**

Для переключения листа таблицы и получения активного листа используются методы EditorAPI.setActiveWorksheet(), EditorAPI.getActiveWorksheet().

```
EditorAPI.setActiveWorksheet("Table1")
activeWorksheet = EditorAPI.getActiveWorksheet()
print(activeWorksheet:getName())
```

**Переименование страницы**

Для переименования таблицы используется метод Table::setName().

```
local tbl = document:getBlocks():getTable(0)
tbl:setName("Первый")
```

**Скрытие и отображение страниц табличного документа**

Для скрытия / отображения листа документа используется метод `Table::setVisible()`.

```
local table = document:getBlocks():getTable(0)
table:setVisible(false)
```

**Копирование страницы**

Для создания копии страницы используется метод `Table::duplicate()`.

```
local table = document:getBlocks():getTable(0)
table:duplicate()
```

**Удаление страницы**

Для удаления таблицы используется метод `Table::remove()`.

```
local table = document:getBlocks():getTable(0)
table:remove()
```

### 4.2.8 Работа со сводными таблицами

Сводная таблица - инструмент обработки данных, служащий для их обобщения и удобства обработки. Схема взаимодействия объектов, связанных со сводными таблицами, приведена на рисунке 17.

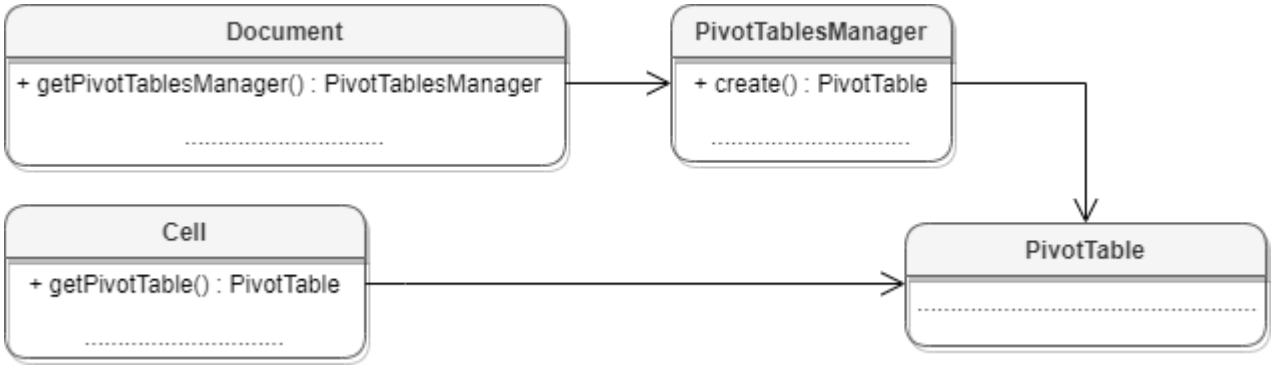



Рисунок 17 – Сводные таблицы

#### 4.2.8.1 Создание сводной таблицы

Метод `PivotTablesManager:create` используется для добавления сводной таблицы

в документ.

**Пример**

```
local pivotTablesManager = document:getPivotTablesManager()
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("A1:G6")
local pivotTable = pivotTablesManager:create(cellRange, sheet:getCell("I8"))
local tableEditor = pivotTable:createPivotTableEditor()
-- Для добавления полей в таблицу используется метод PivotTableEditor:addField:
tableEditor:addField("Product Name", DocumentAPI.PivotTableFieldCategory_Rows)
tableEditor:addField("Country", DocumentAPI.PivotTableFieldCategory_Columns)
tableEditor:addField("Total", DocumentAPI.PivotTableFieldCategory_Values)
-- Для задания заголовков сводной таблицы используется метод
PivotTableEditor:setCaptions:
local pivotCaptions = pivotTable:getPivotTableCaptions()
pivotCaptions.grandTotalCaption = "Итого"
pivotCaptions.rowHeaderCaption = "Продукт"
pivotCaptions.columnHeaderCaption = "Страна"
tableEditor:setCaptions(pivotCaptions)

tableEditor:apply()
```

#### 4.2.8.2 Получение диапазона исходных данных сводной таблицы

Для получения диапазона исходных данных сводной таблицы используется метод PivotTable:getSourceRange().

**Пример**

```
local tbl = document:getBlocks():getTable(0)
-- Получаем ячейку, находящуюся в диапазоне исходных данных сводной таблицы
local cell = tbl:getCell("L8")
-- Получаем сводную таблицу
local pivotTable = cell:getPivotTable()
-- Получаем диапазон исходных данных сводной таблицы
local cellRange = pivotTable:getSourceRange()
-- Для получения границ диапазона используем поля CellRange:
print(cellRange:getBeginRow(), cellRange:getBeginRow())
print(cellRange:getBeginRow(), cellRange:getBeginColumn())
print(cellRange:getBeginRow(), cellRange:getLastRow())
print(cellRange:getBeginRow(), cellRange:getLastColumn())
```

#### 4.2.8.3 Получение диапазона размещения сводной таблицы

Для получения диапазона размещения сводной таблицы используется метод PivotTable:getPivotRange().

**Пример**

```
-- Получаем диапазон размещения сводной таблицы
local cellRange = pivotTable:getPivotRange()
```

#### 4.2.8.4 Получение флагов отображения общих итогов для строк и колонок

Для получения флагов отображения общих итогов для строк и колонок используются методы PivotTable:isRowGrandTotalEnabled(), PivotTable:isColumnGrandTotalEnabled().

**Пример**

```
-- Получаем флаги отображения общих итогов для строк и колонок
isRowGrandTotalEnabled = pivotTable:isRowGrandTotalEnabled()
isColGrandTotalEnabled = pivotTable:isColumnGrandTotalEnabled()
```

#### 4.2.8.5 Получение заголовков сводной таблицы

Для получения заголовков сводной таблицы используется метод PivotTable:getPivotTableCaptions().

**Пример**

```
captions = pivotTable:getPivotTableCaptions()
-- Поля таблицы PivotTableCaptions:
print(captions.emptyCaption)
print(captions.errorCaption)
print(captions.rowHeaderCaption)
print(captions.columnHeaderCaption)
print(captions.valuesHeaderCaption)
```

#### 4.2.8.6 Получение и применение фильтра для сводной таблицы

Для работы с фильтрами сводной таблицы используются методы PivotTable:getFilter(), PivotTableEditor:setFilter().

**Пример**

```
-- По названию поля сводной таблицы получаем фильтр
filter = pivotTable:getFilter("Category")

-- Делаем элементы `Car` и `Technology` скрытыми
filter:setHidden("Car", true)
filter:setHidden("Technology", true)

-- Делаем элемент `Furniture` видимым
filter:setHidden("Furniture", false)

-- Применяем фильтр к сводной таблице
pivotTable:createPivotTableEditor():setFilter(filter):apply()
```

#### 4.2.8.7 Получение полей из области фильтров

Для получения полей из области фильтров используется метод PivotTable:getPageFields().

**Пример**

```
-- Получение полей из области фильтров
pageFields = pivotTable:getPageFields()
-- Перебираем все поля из области фильтров

for fieldIdx = 0, pageFields:size() - 1 do
    fieldProperties = pageFields[fieldIdx].fieldProperties
    print(fieldProperties.fieldName)
    print(fieldProperties.fieldAlias)
    print(fieldProperties.subtotalAlias)
end
```

#### 4.2.8.8 Получение полей из области значений

Для получения полей из области значений используется метод PivotTable:getValueFields().

**Пример**

```
-- Получение полей из области значений
valueFields = pivotTable:getValueFields()
-- Перебираем все поля из области значений
```

```
for fieldIdx = 0, valueFields:size() - 1 do
   print(valueFields[fieldIdx].baseFieldName)
   print(valueFields[fieldIdx].valueFieldName)
   print(valueFields[fieldIdx].cellNumberFormat)
   print(valueFields[fieldIdx].totalFunction)
end
```

#### 4.2.8.9 Получение полей из области строк

Для получения полей из области строк используется метод PivotTable:getRowFields().

**Пример**

```
-- Получение полей из области строк
rowFields = pivotTable:getRowFields();
-- Перебираем все поля из области строк
for fieldIdx = 0, rowFields:size() - 1 do
    fieldProperties = rowFields[fieldIdx].fieldProperties
    print(fieldProperties.fieldName)
    print(fieldProperties.fieldAlias)
    print(fieldProperties.subtotalAlias)
end
```

#### 4.2.8.10 Получение полей из области колонок

Для получения полей из области колонок используется метод PivotTable:getColumnFields().

**Пример**

```
-- Получение полей из области колонок
columnFields = pivotTable:getColumnFields()
-- Перебираем все поля из области колонок
for fieldIdx = 0, columnFields:size() - 1 do
    fieldProperties = columnFields[fieldIdx].fieldProperties
    subtotalFunctions = columnFields[fieldIdx].subtotalFunctions
    --  Далее используем поля структуры PivotTableCategoryField:
    print(fieldProperties.fieldName)
    print(fieldProperties.fieldAlias)
    print(fieldProperties.subtotalAlias)
end
```

#### 4.2.8.11 Получение настроек отображения сводной таблицы

Для получения настроек отображения сводной таблицы используется метод PivotTable:getPivotTableLayoutSettings().

**Пример**

```
layoutSettings = pivotTable:getPivotTableLayoutSettings()
--  Далее используем поля структуры PivotTableLayoutSettings:
print(layoutSettings.reportLayout)
print(layoutSettings.pageFieldOrder)
print(layoutSettings.useGridDropZones)
print(layoutSettings.pageFieldWrapCount)
print(layoutSettings.displayFieldCaptions)
print(layoutSettings.indentForCompactLayout)
print(layoutSettings.valueFieldsOrientation)
print(layoutSettings.isMergeAndCenterLabelsEnabled)
```

#### 4.2.8.12 Обновление сводной таблицы

Для обновления сводной таблицы используется метод PivotTable:update(). Метод возвращает значение типа PivotTableUpdateResult.

```
-- Пересчет и перезаполнение сводной таблицы в соответствии с исходными данными.
-- Обновление сводной таблицы приводит к потере всех неподдерживаемых свойств.
pivotTableUpdateResult = pivotTable:update()
```

### 4.2.9 Работа с фильтрами

Работа с фильтрами возможна только в табличном документе. Диаграмма взаимодействия объектов приведена на рисунке 18.

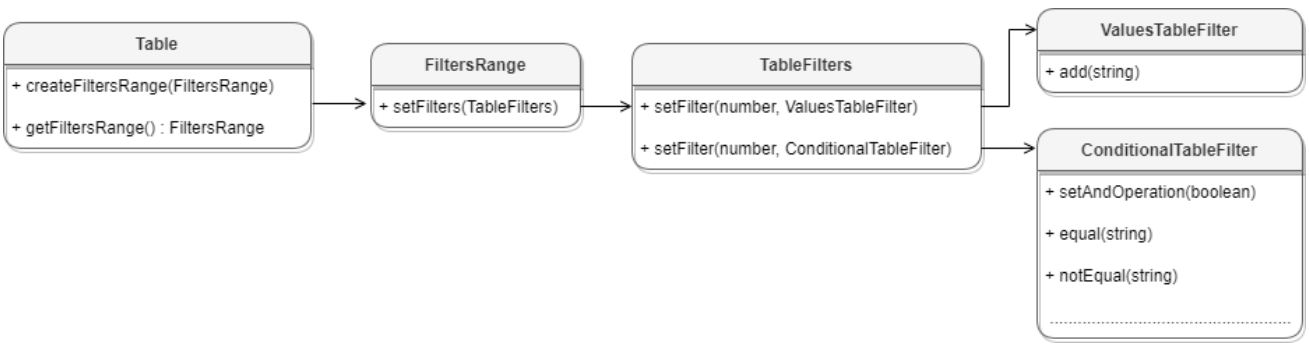



Рисунок 18 – Объектная модель таблиц для работы с фильтрами

Диапазон ячеек для фильтров `FiltersRange` формируется посредством метода `Table:createFiltersRange()`.

Далее создаются фильтры (возможные варианты: `ValuesTableFilter`, `ConditionalTableFilter`).

Фильтры помещаются в структуру `TableFilters`.

Далее фильтры `TableFilters` помещаются в диапазон `FiltersRange` посредством использования метода `FiltersRange:setFilters()`.

**Пример работы с фильтрами в табличном документе**

```
local tbl = EditorAPI.getActiveWorksheet()
local range = DocumentAPI.CellRangePosition(1, 1, 8, 2)
local filtersRange = tbl:createFiltersRange(range)

local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")

local songFilter = DocumentAPI.ConditionalTableFilter()
songFilter:setAndOperation(true)
songFilter:notEqual("")
songFilter:notBegins("TODO")

local tableFilters = DocumentAPI.TableFilters()
tableFilters:setFilter(0, johnPaulFilter)
tableFilters:setFilter(1, songFilter)

filtersRange:setFilters(tableFilters)
```

Пример фильтров, сформированных в результате работы данного примера, приведен на рисунке 19.

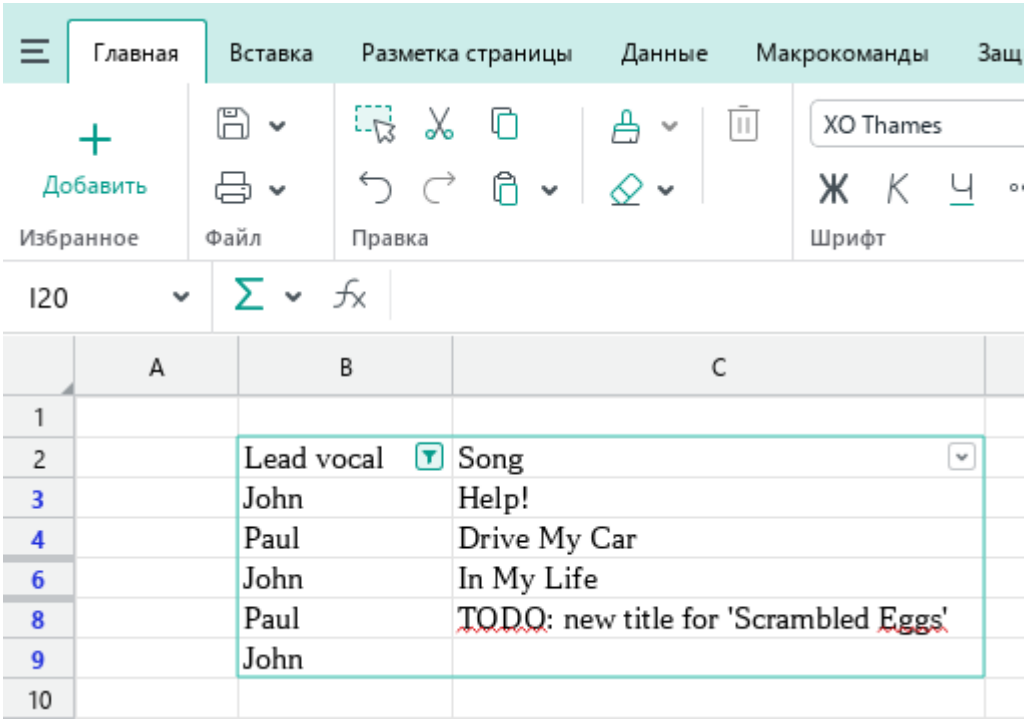



Рисунок 19 – Применение фильтров

### 4.2.10 Обработка событий табличного документа

Для редактора таблиц реализована возможность отслеживания событий, перечисленных в таблице 2. При открытии документа с макрокомандой подписки на события, пользователю выводится оповещение о ее присутствии с возможностью запуска макрокоманды или отказа от него.

> Внимание! Подписка на события возможна только из макрокоманды с названием **"Subscribe to events [MyOffice]"**.

Таблица 2 - События табличного документа

| Событие | Метод |
|---|---|
| Открытие документа | `EventsAPI.subscribeWorkbookOpen` |
| Событие перед сохранением документа | `EventsAPI.subscribeWorkbookBeforeSave` |
| Событие перед закрытием документа | `EventsAPI.subscribeWorkbookBeforeClose` |
| Изменение выделения ячеек | `EventsAPI.subscribeWorksheetSelectionChange` |

| Событие | Метод |
|---|---|
| Изменение содержимого ячеек | EventsAPI.subscribeWorksheetChange |
| Активация страницы документа | EventsAPI.subscribeWorksheetActivate |
| Деактивация страницы документа | EventsAPI.subscribeWorksheetDeactivate |

Пример подписки на событие:

```
EventsAPI.subscribeWorkbookBeforeSave(function(isSaveAs, toCancel)
    EditorAPI.messageBox(string.format('Macro "BeforeSave" is working! (isSaveAs=%s; toCancel=%s)', tostring(isSaveAs), tostring(toCancel)))
    return not toCancel
end)
```

Отменить подписку на событие можно передав в метод подписки `nil` в качестве аргумента. Методы `disableEvents()` и `enableEvents()` позволяют отключить и включить обработку событий для текущего документа.

На данный момент существуют следующие ограничения:

– обработка событий возможна только в табличном документе;
– обработка событий реализована только для надстроек и макрокоманд;
– сначала события обрабатываются надстройками, затем – макрокомандами;
– события вызываются только действиями пользователя, использование функций из макросов или изменения других пользователей при коллаборации не вызывает событий;
– Undo / redo, приводящие к изменению документа, не вызывают события.

### 4.3 Поиск в документе

Для поиска в текстовом или табличном документе необходимо создать экземпляр класса `Search` посредством вызова `DocumentAPI.createSearch(document)`, затем использовать метод `Search:findText` (см. Рисунок 20).

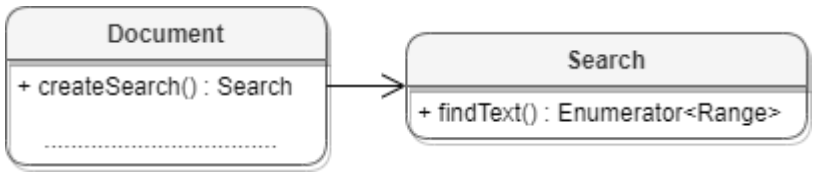



Рисунок 20 – Объектная модель для поиска в документе

**Примеры поиска в документе**

```
search = DocumentAPI.createSearch(document)
text = "English"

-- Поиск по всему документу
ranges = search:findText(text)

-- Поиск с учетом регистра
ranges = search:findText(text, DocumentAPI.CaseSensitive_No)

-- Поиск в диапазоне
local range = document:getBlocks():getBlock(0):getRange()
ranges = search:findText(text, range)

-- Поиск в диапазоне с учетом регистра
local range = document:getBlocks():getBlock(0):getRange()
ranges = search:findText(text, range, DocumentAPI.CaseSensitive_No)

-- Поиск в диапазоне ячеек
local table = document:getBlocks():getTable(0)
local cellRange = table:getCellRange("A1:B3")
ranges = search:findText(text, cellRange)

-- Поиск в диапазоне ячеек с учетом регистра
local table = document:getBlocks():getTable(0)
local cellRange = table:getCellRange("A1:B3")
ranges = search:findText(text, cellRange, DocumentAPI.CaseSensitive_Yes)

-- Поиск в таблице
local table = document:getBlocks():getTable(0)
ranges = search:findText(text, table)

-- Поиск в таблице с учетом регистра
local table = document:getBlocks():getTable(0)
```

```
ranges = search:findText(text, table, DocumentAPI.CaseSensitive_Yes)

-- Отображение результата поиска
for occurrence in ranges do
    print(occurrence:extractText())
end
```

Для поиска ячеек в табличном документе используйте методы Table:find и CellRange:find.

**Пример поиска ячеек в табличном документе**

```
local sheet = document:getBlocks():getTable(0)

local searchProps = DocumentAPI.TableSearchSettings()
searchProps.caseSensitive = DocumentAPI.CaseSensitive_No
searchProps.matchBehaviour = DocumentAPI.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = DocumentAPI.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = true

local results = sheet:find("*eye", searchProps)
for cell in results do
    print(cell:getFormattedValue()) -- Steeleye Stout
end
```

**Пример форматирования результатов поиска**

```
local sheet = document:getBlocks():getTable(0)
local search = DocumentAPI.createSearch(document)
local template = "МойОфис"
local ranges = search:findText(template, sheet)
for range in ranges do
    local props = range:getTextProperties()
    props.bold = true
    range:setTextProperties(props)
end
```

### 4.4 Работа с макрокомандами

Таблица DocumentAPI.Scripts предоставляет доступ к списку макрокоманд документа. На рисунке 21 изображена объектная модель таблиц, относящихся к работе с макрокомандами.

Таблица `DocumentAPI.Scripts` предназначена для доступа к списку макрокоманд, доступна через метод `document:getScripts()`, таблица `DocumentAPI.Scripting` служит для запуска макрокоманд, доступна через `DocumentAPI.createScripting(document)`.

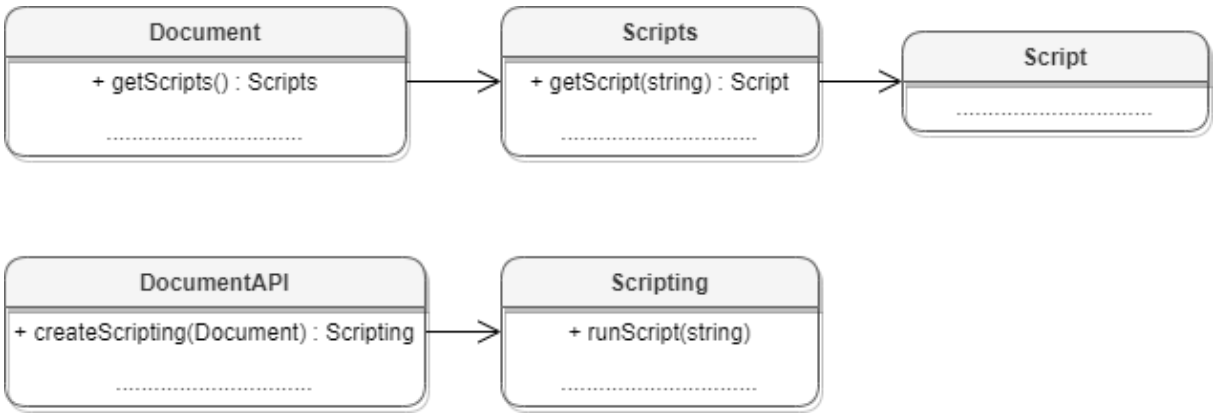



Рисунок 21 – Объектная модель таблиц для работы с макрокомандами

### 4.5 Работа с именованными диапазонами

Именованный диапазон – это диапазон ячеек или формула, которым присвоено имя. Преимуществом именованного диапазона является его информативность. Именованные диапазоны упрощают работу с ячейками, также их удобно использовать при работе с формулами. На данный момент доступна возможность работы с именованными диапазонами, представляющими собой ссылки на диапазоны ячеек. Доступ к именованным диапазонам осуществляется посредством методов `Document:getNamedExpressions()` и `Table:getNamedExpressions()` (см. Рисунок 22 ).

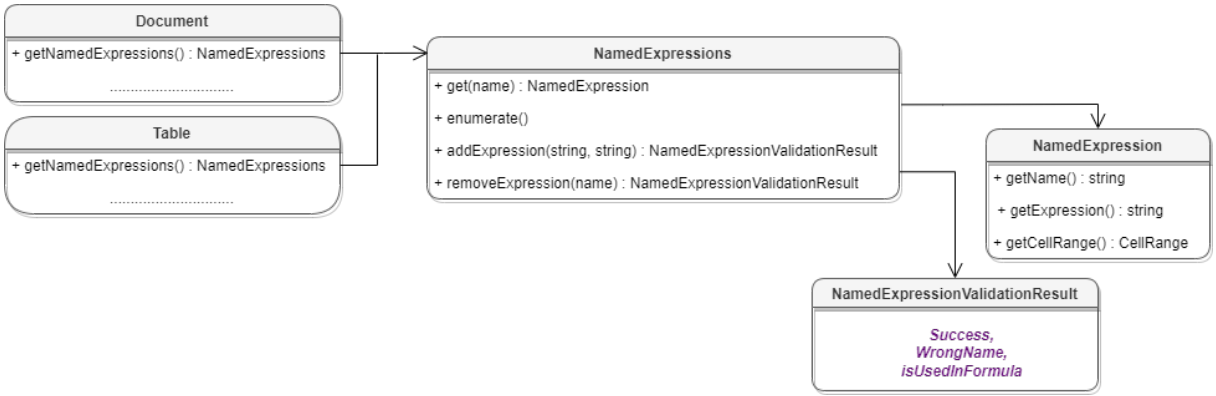



Рисунок 22 – Таблицы для работы с именованными диапазонами

### 4.5.1 Доступ к именованным диапазонам

Доступ к именованным диапазонам осуществляется посредством методов Document:getNamedExpressions() и Table:getNamedExpressions().

**Пример для текстового документа**

```
local namedExpressions = document:getNamedExpressions()
for namedExpression in namedExpressions:enumerate() do
   print(namedExpression)
end
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local namedExpressions = sheet:getNamedExpressions()
local namedExpression = namedExpressions:get("Продажи")
```

### 4.5.2 Получение коллекции именованных диапазонов

Для перечисления именованных диапазонов используется объект NamedExpressionsEnumerator, который может быть получен с помощью метода NamedExpressions:enumerate().

**Пример**

```
local namedExpressions = sheet:getNamedExpressions()
for namedExpression in namedExpressions:enumerate() do
   print(namedExpression)
end
```

### 4.5.3 Добавление именованного диапазона

Для добавления именованного диапазона используется метод NamedExpressions:addExpression().

**Пример**

```
local expressionName = "Покупки"
local expressionValue = "=Формула покупки!$E$6:$E$14"
local namedExpressions = document:getNamedExpressions()
namedExpressions:addExpression(expressionName, expressionValue)
```

#### 4.5.4 Получение параметров именованного диапазона

Для получения детальной информации об именованном диапазоне используются методы NamedExpression:getName, NamedExpression:getExpression, NamedExpression:getCellRange.

```
local name = namedExpression:getName()
local formula = namedExpression:getExpression()
local range = namedExpression:getCellRange()
```

#### 4.5.5 Переименование именованного диапазона

Для переименования именованного диапазона используется метод NamedExpression:setName().

```
local expressions = document:getNamedExpressions()
local expression = expressions:get("Prices")
expression:setName("Totals")
```

#### 4.5.6 Удаление именованного диапазона

Для удаления именованного диапазона используется метод NamedExpressions:removeExpression().

**Пример**

```
local namedExpression = namedExpressions:get(expressionName)
if (namedExpression) then
    namedExpressions:removeExpression(expressionName)
end
```

### 4.6 Работа со строками и столбцами таблиц

В данном разделе содержатся общие для текстовых и табличных документов методы взаимодействия со строками и столбцами таблиц.

#### 4.6.1 Группировка строк и колонок таблицы

Следующий набор методов позволяет группировать строки и колонки таблицы: Table::groupRows(), Table::ungroupRows(), Table::clearRowGroups(),

Table::groupColumns(), Table::ungroupColumns(), Table::clearColumnGroups().

Редактор дает возможность отображать группы в виде иерархии. Совместно с данными методами можно использовать методы Table::setColumnsVisible и Table::setRowsVisible чтобы раскрывать и закрывать фрагменты иерархии групп.

Методы могут вызвать исключения DocumentAPI::OutOfRangeError и DocumentAPI::IncorrectArgumentError в случае использования индексов, выходящих за рамки таблицы.

### 4.6.2 Управление видимостью строк / колонок

Метод Table:isRowVisible позволяет определять видимость строки с заданным индексом.

Метод Table:isColumnVisible позволяет определять видимость столбца с заданным индексом.

Вышеуказанные методы предназначены для работы как в текстовом, так и в табличном редакторе.

**Пример для текстового и табличного редактора**

```
local tbl = document:getBlocks():getTable(0)
print(tbl:isRowVisible(0))
print(tbl:isColumnVisible(1))
```

Метод Table:setColumnsVisible позволяет задавать видимость столбцов, начиная с заданного индекса (только для табличного редактора).

Метод Table:setRowsVisible позволяет задавать видимость строк, начиная с заданного индекса (только для табличного редактора).

**Пример для табличного редактора**

```
function setSelectionVisible(visibility)
    local selection = EditorAPI.getSelection()
    local tbl = selection:getTable()

    local beginRow = selection:getBeginRow()
    local lastRow = selection:getLastRow()
    local beginColumn = selection:getBeginColumn()
```

```
    local lastColumn = selection:getLastColumn()

    tbl:setRowsVisible(beginRow, lastRow - beginRow + 1, visibility)
    tbl:setColumnsVisible(beginColumn, lastColumn - beginColumn + 1, visibility)
end

setSelectionVisible(false)
```

### 4.7 Работа с ячейками таблиц

В данном разделе содержатся общие для текстовых и табличных документов методы взаимодействия с ячейками таблиц.

#### 4.7.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 23):

– непосредственно из таблицы, используя метод `Table.getCell()`;

– из диапазона ячеек методом перечисления `CellRange.enumerate()`.

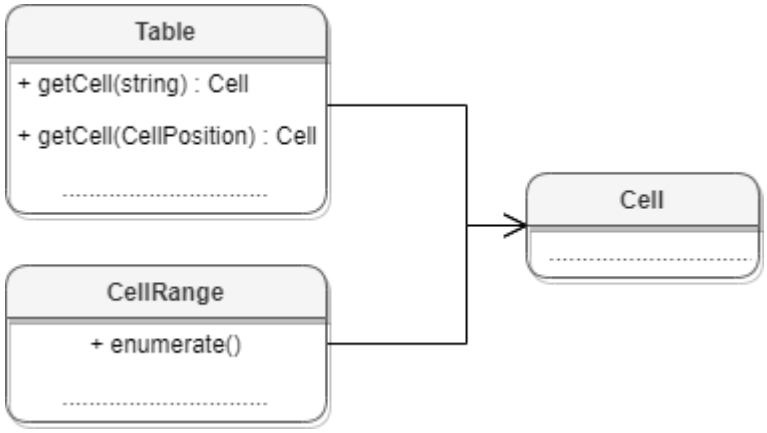



Рисунок 23 – Объектная модель для работы с ячейками таблиц

Для получения содержимого ячейки, заполнения данных, а также для форматирования ячейки используется объект `DocumentAPI.Cell`, представляющий ячейку таблицы с указанным адресом. Метод `Table.getCell()` возвращает экземпляр таблицы `Cell`.

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")
```

Второй вариант доступа к ячейке - перечисление диапазона ячеек методом CellRange.enumerate().

**Пример**

```
local table = document:getBlocks():getTable(0)
local rng = table:getCellRange("B3:C4")
for cell in rng:enumerate() do
    print(cell:getFormattedValue())
end
```

Для определения того, входит ли ячейка в указанный диапазон, используется метод CellRange:containsCell().

**Примеры**

```
local table1 = document:getBlocks():getTable(0)
local table2 = document:getBlocks():getTable(1)

local cellRange1 = table1:getCellRange("A1:C4")
local cellRange2 = table2:getCellRange("A1:C4")

local cell1 = table1:getCell("A1")
local cell2 = table1:getCell("C4")
local cell3 = table1:getCell("E4")

print(cellRange1:containsCell(cell1))
print(cellRange1:containsCell(cell2))
print(cellRange1:containsCell(cell3))

print(cellRange2:containsCell(cell1))
print(cellRange2:containsCell(cell2))
print(cellRange2:containsCell(cell3))
```

Для установки значений ячеек используются методы Cell:setText, Cell:setNumber, Cell:setFormula, Cell:setBool.

**Примеры**

```
local sheet = document:getBlocks():getTable("Лист2")

--setText, текстовое значение
sheet:getCell("A1"):setText("Текст")
```

```
--setNumber, числовое значение с фиксированной точкой
sheet:getCell("B2"):setNumber(10)

--setNumber, числовое значение с плавающей точкой
sheet:getCell("B3"):setNumber(1.0)

--setFormula, текст формулы
sheet:getCell("B4"):setFormula("=SUM(B2:B3)")

--setBool, логическое значение
sheet:getCell("B4"):setBool(false)
```

Для установки даты и времени используется функция Cell:setFormattedValue. Данная функция пытается определить тип значения, переданного в качестве аргумента (число, дата и т.д.) и применяет необходимое форматирование.

**Пример**

```
local sheet = document:getBlocks():getTable("Лист1")

--setFormattedValue, дата
sheet:getCell("B5"):setFormattedValue("22.07.2020")

--setFormattedValue, время
sheet:getCell("B6"):setFormattedValue("12:39")
```

При необходимости есть возможность явно указать формат вводимого значения DocumentAPI.CellFormat (процентный, денежный, экспоненциальный и т.д.), для этого используется функция Cell.SetFormat().

**Пример**

```
local sheet = document:getBlocks():getTable("Лист1")
local value = 12
local cell = sheet:getCell("B1")
-- Установка формата данных
cell:setFormat(DocumentAPI.CellFormat_Accounting)
cell:setNumber(value)
```

Для получения значения ячейки используется метод Cell.getFormattedValue().

**Пример**

```
local sheet = document:getBlocks():getTable("Лист1")
local value = sheet:getCell("B1"):getFormattedValue()
print(value)
```

### 4.7.2 Форматирование ячеек

При работе с ячейками таблиц можно использовать следующие варианты форматирования:

– форматирование параметров ячейки DocumentAPI.CellProperties, например, цвет фона, угол поворота текста;
– форматирование абзаца ячейки, например, отступы абзаца, межстрочный интервал текста;
– форматирование текста, например, цвет текста, начертание;
– задание параметров границ ячеек.

Содержимое ячейки (контент), вне зависимости от того является ли оно текстом, числовым значением или формулой, также описывается экземпляром класса DocumentAPI.Paragraph, и обладает свойствами DocumentAPI.ParagraphProperties. Это дает возможность управлять настройками отображения контента как отдельного абзаца, так и группы абзацев (например, если ячейка содержит несколько предложений текста). Для управления этими настройками используются методы Cell:getParagraphProperties() и Cell:setParagraphProperties() (CellRange:getParagraphProperties() и CellRange:setParagraphProperties()).

**Пример установки и получения свойств абзаца ячейки**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A2") --(DocumentAPI.CellPosition(1,0))

local paraProps = cell:getParagraphProperties()
paraProps.alignment = DocumentAPI.Alignment_Center
cell:setParagraphProperties(paraProps)
```

Вы можете управлять настройками текста ячейки (шрифт, цвет). Метод Cell:getTextProperties() позволяет получить экземпляр таблицы TextProperties, представляющий свойства текста. После изменения значения свойств их необходимо применить к тексту ячейки с помощью метода Cell:setTextProperties().

**Пример настроек текста ячейки**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell(DocumentAPI.CellPosition(0,1))

local textProps = cell:getTextProperties()
textProps.bold = true
textProps.italic = true
local rgba = DocumentAPI.ColorRGBA(121,112,212,255)
textProps.textColor = DocumentAPI.Color(rgba)
cell:setTextProperties(textProps)
```

### 4.7.3 Форматирование границ ячеек

Для оформления границ ячеек используется таблица DocumentAPI.Borders (см. Рисунок 24). Она описывает свойства полей, соответствующих границам и диагоналям ячейки: Left, Right, Top, Bottom, DiagonalDown, DiagonalUp, InnerHorizontal, InnerVertical. Каждая граница ячейки описывается таблицей DocumentAPI.LineProperties, которая, в свою очередь, обладает свойствами DocumentAPI.LineStyle, DocumentAPI.LineEndingProperties, DocumentAPI.Color, LineWidth.

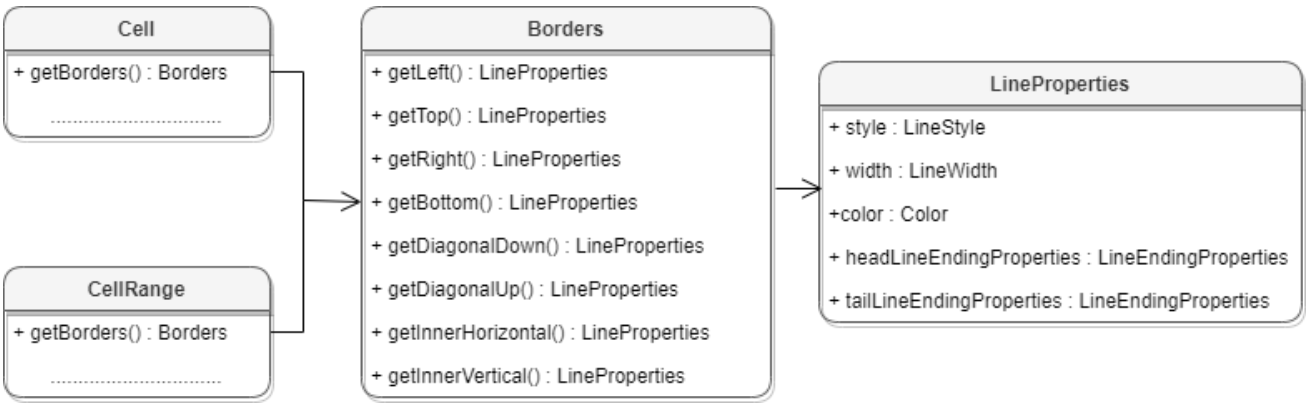



Рисунок 24 – Таблицы для работы с границами ячеек

Для оформления границ отдельной ячейки или группы ячеек необходимо выполнить следующие действия:

– получить ячейку DocumentAPI.Cell или область ячеек DocumentAPI.CellRange;
– настроить параметры для рисования линии границы с помощью экземпляра класса DocumentAPI.LineProperties;

– настроить свойства линии: левой границы, верхней границы и т.д. с помощью экземпляра класса DocumentAPI.Borders;
– установить границы ячеек с помощью Cell.setBorders() или CellRange.setBorders().

**Пример настройки границ ячеек**

```
local sheet = document:getBlocks():getTable("Лист2")
local cellRange = sheet:getCellRange("F3:H7")

--Настроить параметры для рисования линии
local lineProp = DocumentAPI.LineProperties()
lineProp.style = DocumentAPI.LineStyle_Solid
lineProp.width = 1.5
local lc = DocumentAPI.ColorRGBA(55, 146, 179, 200)
lineProp.color = DocumentAPI.Color(lc)

--Настроить положение линии – обводка по внешней границе области
local borders = DocumentAPI.RangeBorders()

--установка внешних границ
borders:setOuter(lineProp)

--Нарисовать границы области
cellRange:setBorders(borders)
```

### 4.7.4 Объединение и разделение ячеек таблицы

Допустимо объединение произвольного числа ячеек таблицы. При объединении указанный диапазон становится единой ячейкой. После завершения операции объединенная ячейка получает значение первой ячейки диапазона.

Для объединения нескольких ячеек используйте метод CellRange.merge().

**Пример**

```
-- Объединение ячеек A1 и A2 на первом листе табличного документа
local tbl = document:getBlocks():getTable(0)
tbl:getCellRange("A1:A2"):merge()
```

С помощью метода Cell:isInMergedRange() можно узнать, принадлежит ли ячейка объединенному диапазону. Метод Cell:getMergedRange() возвращает объединенный диапазон, который содержит ячейку.

**Пример**

```
local cell = tbl:getCell("A2")
if cell:isInMergedRange() then
    mergedRange = cell:getMergedRange()
end
```

Допустимо разъединение только тех ячеек, которые были объединены ранее. После завершения операции данные, содержавшиеся в объединенной ячейке, будут помещены в верхнюю левую ячейку диапазона.

Для разъединения ячеек используйте метод `Cell.unmerge()`.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
-- Ячейка A1 является результатом объединения диапазона A1:A2
tbl:getCell("A1"):unmerge()
```

### 4.8 Защита документов

Данный раздел содержит способы защиты содержимого текстовых и табличных документов:

- Защита диапазона текстового документа
- Защита листа табличного документа
- Защита структуры табличного документа

#### 4.8.1 Защита диапазона текстового документа

Вы можете защитить от изменений диапазон текстового документа. Данные в таком диапазоне нельзя модифицировать до снятия защиты.

Используйте метод `Range:lockContent()` для установки защиты:

```
-- Защита первого абзаца документа:
local textRange =
document:getRange():getBegin():getCurrentRange(DocumentAPI.TextUnit_Paragraph)
textRange:lockContent()
```

При попытке редактирования защищенного фрагмента возникает исключение `DocumentModificationError`:

```
local pos = textRange:getContentEnd():getPreviousPosition(10)
pos:insertText("My Text") -- DocumentModificationError
```

Чтобы определить, защищен ли диапазон документа, используйте метод Range:isContentLocked(). Он возвращает true, если текущий диапазон защищен целиком или содержит защищенные фрагменты:

```
print(textRange:isContentLocked()) -- true
print(document:getRange():isContentLocked()) -- true
```

Для снятия защиты, вызовите метод Range:unlockContent() защищенного фрагмента или диапазона текста, который его содержит:

```
textRange:unlockContent()
-- или
document:getRange():unlockContent()
```

Также вы можете снять защиту только с части диапазона:

```
-- Снятие защиты с последнего предложения защищенного абзаца:
textRange:getContentEnd():getPreviousRange(DocumentAPI.TextUnit_Sentence):unlockC
ontent()
```

### 4.8.2 Защита листа табличного документа

Вы можете защитить ячейки от редактирования и запретить выполнение определенных действий на листе табличного документа.

Для настройки параметров защиты ячеек используется объект CellProtectionProperties. Он позволяет настроить возможность редактирования ячеек и видимость формул на защищенном листе. Вызовите метод Cell:setProtectionProperties() или CellRange:setProtectionProperties(), чтобы применить эти параметры к ячейке или диапазону. Параметры ячеек должны быть заданы до установки защиты.

С помощью объекта TableProtectionProperties можно задать доступные на листе действия. Он используется в методе для установки защиты Table:setProtection(). Также метод позволяет установить пароль для снятия защиты.

```
local sheet = document:getBlocks():getTable(0)
-- Разрешает редактировать все ячейки листа:
local cells = sheet:getCellRange("A1:Z300")
local cellProtection = DocumentAPI.CellProtectionProperties()
cellProtection.lockedForChanges = false
cellProtection.formulasNotDisplayed = false
cells:setProtectionProperties(cellProtection)
-- Задает ячейки, которые будут защищены от изменения:
```

```
local protectedCells = sheet:getCellRange("A1:H8")
local cellProtection1 = DocumentAPI.CellProtectionProperties()
cellProtection1.lockedForChanges = true
cellProtection1.formulasNotDisplayed = false
protectedCells:setProtectionProperties(cellProtection1)
-- Запрещает вставку и удаление строк и столбцов:
local tableProtection = DocumentAPI.TableProtectionProperties()
tableProtection.deleteColumns = false
tableProtection.deleteRows = false
tableProtection.filterData = true
tableProtection.formatCells = true
tableProtection.formatColumns = true
tableProtection.formatRows = true
tableProtection.insertAndEditObjects = true
tableProtection.insertAndEditPivotTables = true
tableProtection.insertColumns = false
tableProtection.insertLinks = true
tableProtection.insertRows = false
tableProtection.selectProtectedCells = true
tableProtection.sortData = true
-- Устанавливает защиту на лист и пароль для ее снятия:
sheet:setProtection(tableProtection, "password")
```

При попытке редактирования защищенных ячеек или выполнения запрещенных действий возникает исключение `SpreadsheetProtectionError`.

Вы можете узнать, защищена ли ячейка или диапазон от редактирования, с помощью метода `Cell:isProtected()` или `CellRange:isProtected()`:

```
local cell = sheet:getCell("F6")
print(cell:isProtected()) -- true
```

Текущие настройки защиты ячеек возвращают методы `Cell:getProtectionProperties()` и `CellRange:getProtectionProperties()`.

Статус защиты листа документа можно получить используя метод `Table:isProtected()`:

```
print(sheet:isProtected()) -- true
```

Метод `Table:getProtectionProperties()` позволяет получить настройки защиты листа.

Для снятия защиты с листа, используется метод Table:removeProtection(). Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение IncorrectPasswordError.

```
sheet:removeProtection("password")
```

### 4.8.3 Защита структуры табличного документа

Защита структуры документа ограничивает взаимодействие с листами табличного документа. Пока защита установлена, вы не можете добавлять, удалять, скрывать, перемещать и переименовывать листы.

Для защиты структуры используется метод Document:setStructureProtection(). Опционально вы можете передать в качестве параметра пароль, который будет необходим для снятия защиты:

```
document:setStructureProtection("password")
```

При изменении листов в документе с защищенной структурой, возникает исключение SpreadsheetProtectionError. Также это исключение возникает при установке защиты структуры на уже защищенный документ.

```
local sheet = document:getBlocks():getTable(0)
sheet:remove() -- SpreadsheetProtectionError
```

Используйте метод Document:isStructureProtected() для получения состояния защиты структуры документа:

```
print(document:isStructureProtected()) -- true
```

Метод Document:removeStructureProtection() позволяет снять защиту структуры. В качестве параметра вы можете передать пароль, если он использовался при установке защиты. В случае несоответствия введенного пароля заданному при установке, возникает исключение IncorrectPasswordError.

```
document:removeStructureProtection("password")
```

# 5 СПРАВОЧНИК ТАБЛИЦ DOCUMENTAPI

В данном разделе приведено описание таблиц и методов пространства имен `DocumentAPI`.

## 5.1 Таблица DocumentAPI.AboveAverageConditionalFormatOperator

Таблица `AboveAverageConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил "Выше среднего" и "Ниже среднего". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

### Конструкторы

```
AboveAverageConditionalFormatOperator(ConditionalFormatAboveAverageCondition condition)
```

```
AboveAverageConditionalFormatOperator(ConditionalFormatAboveAverageCondition condition, number stdDev)
```

### Пример

| Количество |
|---|
| 2 |
| 5 |
| 10 |
| 1 |
| 3 |
| 4 |
| 6 |
| 15 |
| 20 |
| 2 |
| 7 |

Рисунок 25 – Пример создания правила "Выше среднего"

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local cellRange = sheet:getCellRange("F2:F12")
local cellRangePosition = cellRange:getTableRange()

local aboveStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill = DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(0,
```

```
255, 0, 150)))
aboveStyle.cellProperties = cellProperties

local aboveOperator =
DocumentAPI.createAboveAverageConditionalFormatOperator(DocumentAPI.ConditionalFo
rmatAboveAverageCondition_Above)

local aboveRule = DocumentAPI.ConditionalFormatRule(aboveOperator, aboveStyle,
cellRangePosition, false)
rules:addRule(aboveRule)
```

### 5.1.1 Метод AboveAverageConditionalFormatOperator:getCondition

Метод возвращает условие применения форматирования.

**Вызов**

```
ConditionalFormatAboveAverageCondition getCondition()
```

**Возвращает**

– условие применения форматирования, тип ConditionalFormatAboveAverageCondition.

### 5.1.2 Метод AboveAverageConditionalFormatOperator:getStdDev

Метод возвращает количество стандартных отклонений используемое для вычисления значений выше и ниже среднего.

**Вызов**

```
number getStdDev()
```

**Возвращает**

– количество стандартных отклонений, тип `number`.

– `nil`, если отклонения не заданы.

### 5.1.3 Метод AboveAverageConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

### 5.2 Таблица DocumentAPI.AbsoluteFrame

Таблица DocumentAPI.AbsoluteFrame описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 26). Предназначена для получения и изменения свойств позиции медиаобъектов. Используется в табличном документе.

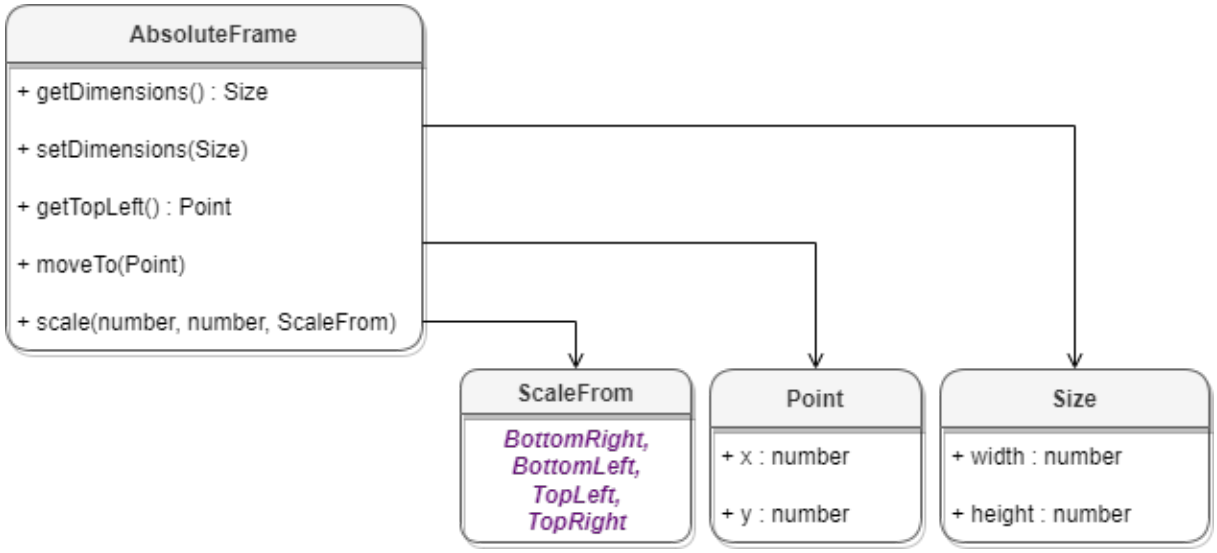



Рисунок 26 – Объектная модель таблицы DocumentAPI.AbsoluteFrame

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
   local absoluteFrame = mediaObject:getFrame()
   print(absoluteFrame:getDimensions())
   print(absoluteFrame:getTopLeft())
end
```

#### 5.2.1 Метод AbsoluteFrame:getDimensions

Возвращает размеры медиаобъекта, тип - DocumentAPI.SizeU.

**Пример**

```
-- Получение размеров всех медиаобъектов
sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
```

```
   print(mediaObject:getFrame():getDimensions())
end
```

### 5.2.2 Метод AbsoluteFrame:getTopLeft

Метод возвращает позицию верхней левой точки медиаобъекта, тип - DocumentAPI.PointU.

**Пример**

```
sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
   topLeftPosition = mediaObject:getFrame():getTopLeft()
   print("x=", topLeftPosition.x, "y=", topLeftPosition.y)
end
```

### 5.2.3 Метод AbsoluteFrame:moveTo

Метод перемещает объект в заданную позицию, тип аргумента - DocumentAPI.PointU.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
   local absoluteFrame = mediaObject:getFrame()
   newFramePosition = DocumentAPI.PointU(20, 20)
   absoluteFrame:moveTo(newFramePosition)
end
```

### 5.2.4 Метод AbsoluteFrame:scale

Метод `scale` изменяет размер объекта, масштабируя его по горизонтали и вертикали. Возможно изменение позиции объекта в соответствии со значением аргумента `scaleFrom`.

**Вызов**

```
scale(widthScale, heightScale, scaleFrom)
```

**Параметры**

- `widthScale` – коэффициент масштабирования по горизонтали, тип - числовой;
- `heightScale` – коэффициент масштабирования по вертикали, тип - числовой;

– `scaleFrom` – точка, сохраняющая позицию при масштабировании, тип - `DocumentAPI.ScaleFrom`.

**Пример**

```
-- Уменьшение масштаба всех медиаобъектов на 50%
sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
   mediaObject:getFrame():scale(0.5, 0.5, DocumentAPI.ScaleFrom_TopLeft)
end
```

### 5.2.5 Метод AbsoluteFrame:setDimensions

Метод задает размеры (изменяет размер) медиаобъекта.

**Вызов**

```
setDimensions(size)
```

**Параметры**

`size` – размеры встроенного объекта, тип - `DocumentAPI.SizeU`.

**Пример**

```
-- Изменение размера всех медиаобъектов
sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
   mediaObject:getFrame():setDimensions(DocumentAPI.SizeU(100, 100))
end
```

## 5.3 Таблица DocumentAPI.AccountingCellFormatting

Таблица содержит параметры финансового формата ячеек таблицы и используется в качестве аргумента метода `Cell:setFormat()`.

Таблица 3 – Описание полей таблицы `DocumentAPI.AccountingCellFormatting`

| Поле | Тип | Описание |
|---|---|---|
| `AccountingCellFormatting.decimalPlaces` | `number` | Количество десятичных позиций |
| `AccountingCellFormatting.symbol` | `string` | Символ денежной единицы |

| Поле | Тип | Описание |
|---|---|---|
| AccountingCellFormatting.localeCode | number | Идентификатор кода языка (MS-LCID) |
| AccountingCellFormatting.fillSymbol | string | Символ заполнения |
| AccountingCellFormatting.useThousandsSeparator | bool | Использовать разделитель для тысячных |
| AccountingCellFormatting.currencySignPlacement | CurrencySignPlacement | Тип размещения знака валюты |

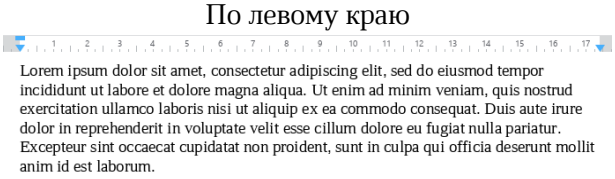
**Пример**

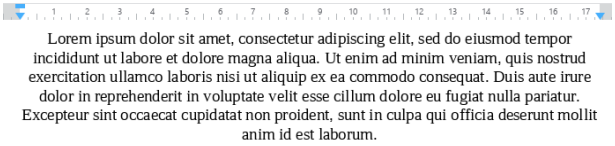
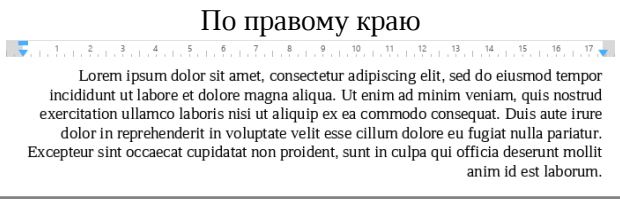
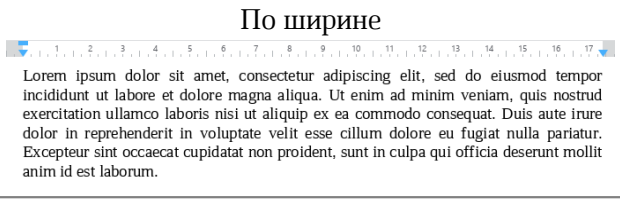
```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")

cell:setFormat(DocumentAPI.CellFormat_Accounting)
local accountingCellFormatting = DocumentAPI.AccountingCellFormatting()
accountingCellFormatting.decimalPlaces = 3
accountingCellFormatting.symbol = 'Руб'

cell:setFormat(accountingCellFormatting)
print(cell:getFormattedValue())
```

### 5.4 Таблица DocumentAPI.Alignment

В таблице 4 представлены варианты выравнивания текста по горизонтали в текстовом редакторе или содержимого ячеек в табличном редакторе.

Таблица 4 – Варианты выравнивания по горизонтали

| Значение | Описание |
|---|---|
| DocumentAPI.Alignment_Default | Выравнивание по умолчанию. |
| DocumentAPI.Alignment_Left | По левому краю  |
| DocumentAPI.Alignment_Center | По центру |

| Значение | Описание |
|---|---|
| |  |
| DocumentAPI.Alignment_Right |  |
| DocumentAPI.Alignment_Justify |  |

**Пример для текстового документа**

```
local para = document:getBlocks():getParagraph(0)
local props = para:getParagraphProperties()
props.alignment = DocumentAPI.Alignment_Center
para:setParagraphProperties(props)
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
cell = tbl:getCell("D3")
local props = cell:getParagraphProperties()
props.alignment = DocumentAPI.Alignment_Center
cell:setParagraphProperties(props)
```

### 5.5 Таблица DocumentAPI.BinaryConditionalFormatOperator

Таблица `BinaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил "Между" и "Не между". Используется при создании правила условного форматирования (`ConditionalFormatRule`). В качестве аргументов могут использоваться как значения, так и формулы.

**Конструктор**

```
	BinaryConditionalFormatOperator(ConditionalFormatBinaryCondition condition, string firstArgument, string secondArgument)
```

**Пример**

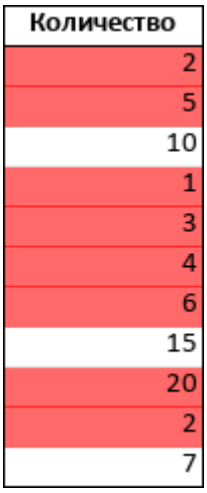



Рисунок 27 – Пример создания правила "Не между"

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local binaryStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
binaryStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("F2:F12")
local cellRangePosition = cellRange:getTableRange()

local binaryOperator =
DocumentAPI.createBinaryConditionalFormatOperator(DocumentAPI.ConditionalFormatBi
naryCondition_NotBetween, "=$F$12", "=$F$9")

local binaryRule = DocumentAPI.ConditionalFormatRule(binaryOperator, binaryStyle,
cellRangePosition, false)
rules:addRule(binaryRule)
```

### 5.5.1 Метод BinaryConditionalFormatOperator:getCondition

Метод возвращает условие применения форматирования.

**Вызов**

```
ConditionalFormatBinaryCondition getCondition()
```

**Возвращает**

– условие применения форматирования, тип ConditionalFormatBinaryCondition.

### 5.5.2 Метод BinaryConditionalFormatOperator:getFirstArgument

Метод возвращает первый аргумент условия.

**Вызов**

```
string getFirstArgument()
```

**Возвращает**

– первый аргумент, тип `string`.

### 5.5.3 Метод BinaryConditionalFormatOperator:getSecondArgument

Метод возвращает второй аргумент условия.

**Вызов**

```
string getSecondArgument()
```

**Возвращает**

– второй аргумент, тип `string`.

### 5.5.4 Метод BinaryConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип `ConditionalFormatOperatorType`.

## 5.6 Таблица DocumentAPI.Block

Таблица `DocumentAPI.Block` является базовой для всех блоков документа. От нее наследуются таблицы `Paragraph`, `Table`, `Shape`, `Field` (см. Рисунок 28).

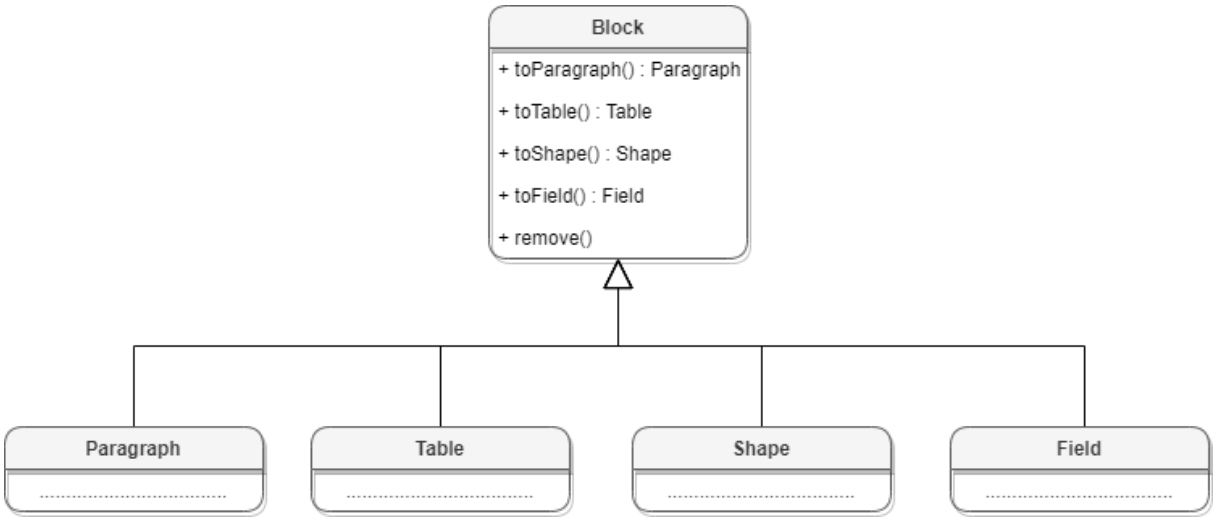



Рисунок 28 – Объектная модель таблицы `DocumentAPI.Block`

### 5.6.1 Метод Block:getRange

Возвращает диапазон `DocumentAPI.Range`, в котором содержится данный блок.

**Пример**

```
local range = document:getBlocks():getBlock(0):getRange()
print(range:extractText())
```

### 5.6.2 Метод Block:getSection

Метод возвращает раздел `DocumentAPI.Section`, содержащий блок.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local pageProperties = section:getPageProperties()
```

### 5.6.3 Метод Block:remove

Удаляет блок из документа. Текущий экземпляр объекта `DocumentAPI.Block` становится недействительным.

**Пример**

```
document:getBlocks():getBlock(0):remove()
```

### 5.6.4 Методы toParagraph, toTable, toShape, toField

Преобразует объект `DocumentAPI.Block` в объект соответствующего типа. В случае, если тип не совпадает, и преобразование не может быть выполнено, метод возвращает `nil`.

**Пример**

```
local paragraph = document:getBlocks():getBlock(0):toParagraph()
if (paragraph ~= nil) then
  local para_props = paragraph:getParagraphProperties()
end
```

### 5.7 Таблица DocumentAPI.Blocks

Таблица `DocumentAPI.Blocks` обеспечивает доступ к блокам `DocumentAPI.Block` документа или диапазона документа (см. Рисунок 29). Таблица `DocumentAPI.Blocks` может быть получена вызовом метода `Document:getBlocks` или `HeaderFooter:getBlocks`.

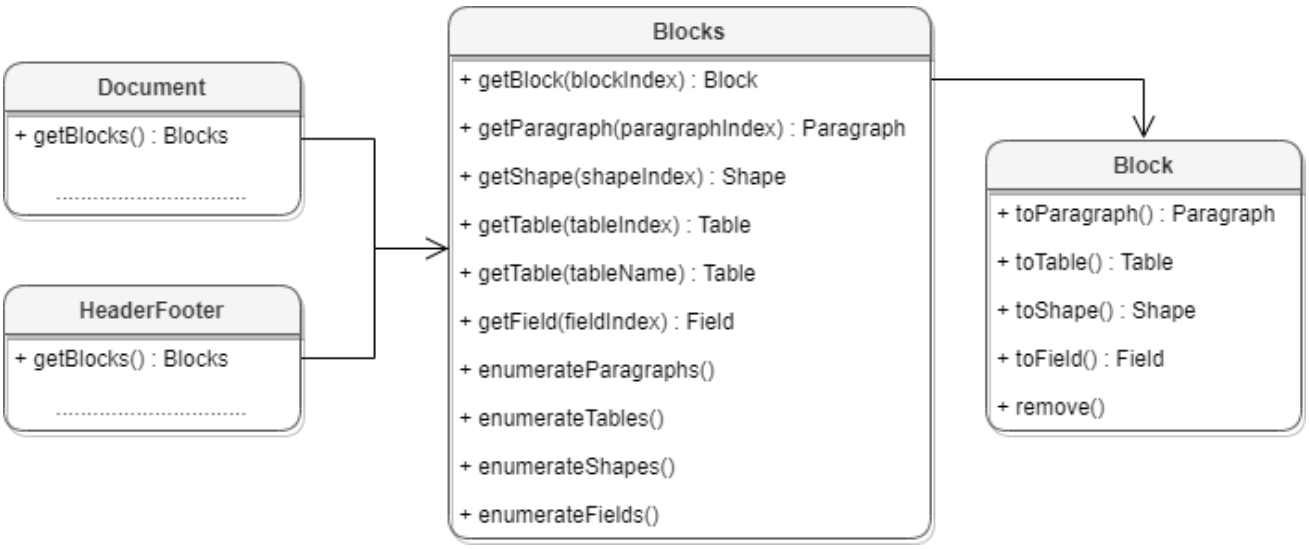



Рисунок 29 – Объектная модель таблицы `DocumentAPI.Blocks`

### 5.7.1 Метод Blocks:enumerate

Позволяет перечислить объекты типа `DocumentAPI.Block`.

**Пример**

```
for block in document:getBlocks():enumerate() do
    print(block:getRange():extractText())
end
```

### 5.7.2 Метод Blocks:enumerateParagraphs

Позволяет реализовать перечисление абзацев DocumentAPI.Paragraph.

**Пример**

```
for paragraph in document:getBlocks():enumerateParagraphs() do
    print(paragraph:getRange():extractText())
end
```

### 5.7.3 Метод Blocks:enumerateTables

Позволяет перечислить объекты типа DocumentAPI.Table.

**Пример**

```
for table in document:getBlocks():enumerateTables() do
    print(table:getName())
end
```

### 5.7.4 Метод Blocks:getBlock

Возвращает объект типа DocumentAPI.Block по заданному индексу. Нумерация индексов начинается с нуля.

**Пример**

```
local block = document:getBlocks():getBlock(0)
```

### 5.7.5 Метод Blocks:getField

Возвращает объект типа DocumentAPI.Field по заданному индексу.

**Пример**

```
local field = document:getBlocks():getField(0)
```

### 5.7.6 Метод Blocks:getParagraph

Возвращает абзац с указанным индексом. Нумерация индексов начинается с нуля.

**Пример**

```
local para = document:getBlocks():getParagraph(0)
```

### 5.7.7 Метод Blocks:getShape

Возвращает фигуру `DocumentAPI.Shape` по заданному индексу.

**Пример**

```
local shape = document:getBlocks():getShape(0)
```

### 5.7.8 Метод Blocks:getTable

Для табличного документа возвращает лист (`worksheet`), для текстового документа возвращает таблицу. Параметры поиска - индекс или имя таблицы. Нумерация листов начинается с нуля.

**Пример**

```
local table = document:getBlocks():getTable(0)
```

В качестве параметра метода также можно указать имя таблицы.

**Пример**

```
local table = document:getBlocks():getTable("Sheet1")
```

## 5.8 Таблица DocumentAPI.Bookmarks

Предоставляет доступ к операциям с закладками в текстовом документе. Закладки не поддерживаются в табличном документе.

### 5.8.1 Метод Bookmarks:getBookmarkRange

Возвращает объект `DocumentAPI.Range` для дальнейшей работы с содержимым закладки (`bookmark`). Если закладка не найдена. возвращается `nil`. Метод можно использовать только в текстовых документах.

**Пример**

```
local bookmarks = document:getBookmarks()
local bookmarkRange = bookmarks:getBookmarkRange("Bookmark")
if (bookmarkRange ~= nil) then
  bookmarkRange:replaceText("Lua")
end
```

### 5.8.2 Метод Bookmarks:removeBookmark

Удаляет закладку по ее названию.

**Пример**

```
document:getBookmarks():removeBookmark("Bookmark")
```

## 5.9 Таблица DocumentAPI.Borders

Таблица `DocumentAPI.Borders` предназначена для оформления границ отдельной ячейки таблицы (см. таблицу 5). Методы установки границ возвращают объект `Borders` и используют в качестве параметра объект `DocumentAPI.LineProperties`, который содержит такие настройки линии, как тип, толщина, цвет.

Таблица 5 – Описание методов таблицы `DocumentAPI.Borders`

| Метод | Описание |
|---|---|
| `Borders setLeft(lineProperties)` | Установка левой границы ячейки |
| `Borders setRight(lineProperties)` | Установка правой границы ячейки |
| `Borders setTop(lineProperties)` | Установка верхней границы ячейки |
| `Borders setBottom(lineProperties)` | Установка нижней границы ячейки |
| `Borders setDiagonalDown(lineProperties)` | Установка диагональной линии  |
| `Borders setDiagonalUp(lineProperties)` | Установка диагональной линии  |
| `Borders setOuter(lineProperties)` | Установка внешних границ ячейки |
| `Borders setDiagonals(lineProperties)` | Установка обеих типов диагональных линий одновременно |
| `Borders setInnerHorizontal(lineProperties)` | Установка внутренних горизонтальных границ ячейки |

| Метод | Описание |
|---|---|
| `Borders setInnerVertical(lineProperties)` | Установка внутренних вертикальных границ ячейки |
| `Borders setInner(lineProperties)` | Установка внутренних границ ячейки |
| `Borders setAll(lineProperties)` | Установка всех границ ячейки |
| `LineProperties getLeft()` | Получение левой границы ячейки |
| `LineProperties getRight()` | Получение правой границы ячейки |
| `LineProperties getTop()` | Получение верхней границы ячейки |
| `LineProperties getBottom()` | Получение нижней границы ячейки |
| `LineProperties getDiagonalDown()` | Получение диагональной линии |
| `LineProperties getDiagonalUp()` | Получение диагональной линии |
| `LineProperties getInnerHorizontal()` | Получение внутренних горизонтальных границ ячейки |
| `LineProperties getInnerVertical()` | Получение внутренних вертикальных границ ячейки |
| `bool isEmpty()` | Возвращает `true`, если не установлена ни одна граница ячейки |

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("C3")

LineProperties = DocumentAPI.LineProperties()
LineProperties.style = DocumentAPI.LineStyle_Dash
LineProperties.width = 1.5
LineProperties.color = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))

borders = DocumentAPI.Borders()
borders:setLeft(LineProperties):setRight(LineProperties)
borders:setTop(LineProperties):setBottom(LineProperties)

cell:setBorders(borders)
```

### 5.10 Таблица DocumentAPI.CalculationMode

Режимы пересчета формул в документе представлены в таблице 6. Таблица `DocumentAPI.CalculationMode` используется в методах `Document:getCalculationMode()` и `Document:setCalculationMode()`.

Таблица 6 – Режимы пересчета формул

| Значение | Описание |
|---|---|
| `DocumentAPI.CalculationMode_Auto` | Формулы пересчитываются автоматически при изменении данных. |
| `DocumentAPI.CalculationMode_Manual` | Формулы пересчитываются вручную. |

### 5.11 Таблица DocumentAPI.CaseSensitive

Таблица `DocumentAPI.CaseSensitive` используется для настройки параметров поиска в текстовом или табличном документе (см. метод `Search:FindText()` и поле `TableSearchSettings.caseSensitive`). Описание полей таблицы представлено в таблице 7.

Таблица 7 – Параметры регистра при поиске

| Значение | Описание |
|---|---|
| `DocumentAPI.CaseSensitive_Yes` | Поиск с учетом регистра |
| `DocumentAPI.CaseSensitive_No` | Поиск без учета регистра |

**Пример**

```
local search = DocumentAPI.createSearch(document)
for r in search:findText("Hello, world", DocumentAPI.CaseSensitive_Yes) do
    print(r:extractText())
end
```

### 5.12 Таблица DocumentAPI.Cell

Таблица `DocumentAPI.Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 30).

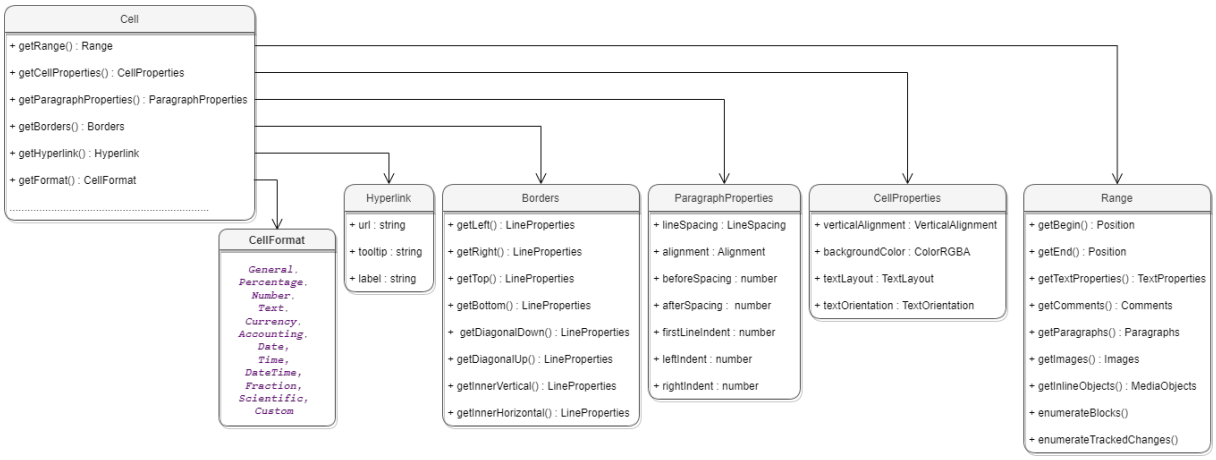



Рисунок 30 – Объектная модель ячейки таблиц

### 5.12.1 Метод Cell:calculate

Метод пересчитывает формулу в текущей ячейке.

**Вызов**

```
calculate()
```

### 5.12.2 Метод Cell:checkDataValidation

Метод проверяет значение ячейки согласно заданной в ней проверке данных.

**Вызов**

```
DataValidationResult checkDataValidation()
```

**Возвращает**

– результат проверки значения ячейки, тип DataValidationResult.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("C10")
local result = cell:checkDataValidation()
if not result:isValid() then
    print(result:getDataValidation():getErrorMessage())
end
```

### 5.12.3 Метод Cell:getBoolValue

Метод возвращает логическое значение ячейки.

**Вызов**

```
bool getBoolValue()
```

**Возвращает**

– логическое значение ячейки, тип `bool`.

– `nil`, если ячейка не содержит логического значения.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
sheet:getCell("D2"):setNumber(1)
sheet:getCell("E2"):setNumber(100)
sheet:getCell("C2"):setFormula("=D2*100=E2")

sheet:getCell("C2"):getBoolValue() -- true
```

### 5.12.4 Метод Cell:getBorders

Позволяет получить границы ячейки (тип `DocumentAPI.Borders`). Примеры использования приведены в разделе `DocumentAPI.Borders`.

**Пример**

```
local cell = document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0,
0))
local borders = cell:getBorders()
```

### 5.12.5 Метод Cell:getCellProperties

Позволяет получить свойства `DocumentAPI.CellProperties` ячейки.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell_props = tbl:getCell("A6"):getCellProperties()
```

### 5.12.6 Метод Cell:getColumnIndex

Метод возвращает индекс текущего столбца. Индексация столбцов начинается с нуля.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("A3")
```

```
print(cell:getColumnIndex()) -- 0
print(cell:getRowIndex()) -- 2
```

### 5.12.7 Метод Cell:getCurrentRegion

Метод возвращает заполненный диапазон ячеек, содержащий текущую ячейку. Возвращаемый диапазон ограничен пустыми строками и столбцами.

**Вызов**

```
CellRange getCurrentRegion()
```

**Возвращает**

– диапазон ячеек, тип CellRange.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("E6")
local region = cell:getCurrentRegion()
```

### 5.12.8 Метод Cell:getCustomFormat

Возвращает строку формата ячейки.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cust_format = tbl:getCell("A6"):getCustomFormat()
```

### 5.12.9 Метод Cell:getDataValidation

Метод возвращает настройки проверки данных для текущей ячейки.

**Вызов**

```
DataValidation getDataValidation()
```

**Возвращает**

– настройки проверки данных, тип DataValidation.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("E4")
local cellDV = cell:getDataValidation()
```

```
print(cellDV:getPrompt())
print(cellDV:getFormula1())
```

### 5.12.10 Метод Cell:getFormat

Метод возвращает формат ячейки. Список поддерживаемых форматов ячеек приведен в разделе DocumentAPI.CellFormat.

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")

local cellFormatting = DocumentAPI.PercentageCellFormatting()
cell:setFormat(cellFormatting)
print("Формат: ", cell:getFormat()) -- 1
```

### 5.12.11 Метод Cell:getFormattedValue

Метод позволяет получить значение ячейки в текущем формате. Список поддерживаемых форматов см. в разделе DocumentAPI.CellFormat.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getCell("A6"):getFormattedValue()) -- 21.6.1972
```

### 5.12.12 Метод Cell:getFormulaAsString

Возвращает текст формулы ячейки. Формула – это любое выражение в ячейке, которое начинается со знака равенства (=).

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local formula = tbl:getCell("A6"):getFormulaAsString()
```

### 5.12.13 Метод Cell:getHyperlink

Возвращает первый объект в ячейке типа DocumentAPI.Hyperlink.

```
local cell = document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0,
 0))
```

```
local hyperlink = cell:getHyperlink()
if (hyperlink ~= nil) then
   print(hyperlink)
end
```

### 5.12.14 Метод Cell:getMergedRange

Метод возвращает объединенный диапазон, который содержит текущую ячейку. Если ячейка не принадлежит объединенному диапазону, возвращает диапазон, состоящий из текущей ячейки.

**Вызов**

```
CellRange getMergedRange()
```

**Возвращает**

– диапазон ячеек, тип CellRange.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
sheet:getCellRange("D2:F6"):merge()
local cell = sheet:getCell("E4")
local mergedRange = cell:getMergedRange()
print(mergedRange:getAddress(DocumentAPI.CellRangeAddressSettings())) -- D2:F6
```

### 5.12.15 Метод Cell:getNote

Метод возвращает текст заметки для текущей ячейки.

**Вызов**

```
string getNote()
```

**Возвращает**

– текст заметки.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")
cell:setText(cell:getNote())
```

Методы Cell:setNote и Cell:removeNote позволяют добавить и удалить заметку.

### 5.12.16 Метод Cell:getNumberValue

Метод возвращает числовое значение ячейки.

**Вызов**

```
number getNumberValue()
```

**Возвращает**

– числовое значение ячейки, тип `number`.

– `nil`, если ячейка не содержит числового значения.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("D2")
cell:setNumber(0.1)

cell:getNumberValue()
```

### 5.12.17 Метод Cell:getParagraphProperties

Возвращает свойства абзаца `DocumentAPI.ParagraphProperties`, находящегося в ячейке.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A2") --(DocumentAPI.CellPosition(1,0))

local paraProps = cell:getParagraphProperties()
print(paraProps.alignment)
```

### 5.12.18 Метод Cell:getPivotTable

Возвращает сводную таблицу `DocumentAPI.PivotTable`, относящуюся к ячейке.

**Пример**

```
tbl = document:getBlocks():getTable(0)
cell = tbl:getCell("B4")
pivotTable = cell:getPivotTable()
```

### 5.12.19 Метод Cell:getProtectionProperties

Метод возвращает параметры защиты ячейки табличного документа (см. Защита листа табличного документа).

**Вызов**

```
CellProtectionProperties getProtectionProperties()
```

**Возвращает**

– `CellProtectionProperties`: свойства защиты ячейки (`nil`, если ячейка находится в сводной таблице).

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")

local cellProps = cell:getProtectionProperties()
cellProps.lockedForChanges = false
cellProps.formulasNotDisplayed = false

cell:setProtectionProperties(cellProps)
firstSheet:setProtection(tableProps)
```

### 5.12.20 Метод Cell:getRange

Метод возвращает объект `DocumentAPI.Range` для управления содержимым ячейки.

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")
local range = cell:getRange()
local pos = range:getBegin()
pos:insertText("Привет, Мир!")
```

### 5.12.21 Метод Cell:getRawValue

Возвращает значение ячейки в формате «Общий» (без форматирования).

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local val = tbl:getCell("A6"):getRawValue()
```

### 5.12.22 Метод Cell:getResolvedBorders

Метод возвращает границы ячейки с учетом границ окружающих ячеек.

**Вызов**

```
Borders getResolvedBorders()
```

**Возвращает**

– границы ячейки, тип Borders.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellBorders = sheet:getCell("B2")

local lineProperties = DocumentAPI.LineProperties()
lineProperties.style = DocumentAPI.LineStyle_Solid
lineProperties.width = 1.5
local borders = DocumentAPI.Borders()
borders:setLeft(lineProperties):setRight(lineProperties)
cellBorders:setBorders(borders)

local cell = sheet:getCell("A2")
print(cell:getBorders():getRight()) -- nil
print(cell:getResolvedBorders():getRight().width) -- 1,5
```

### 5.12.23 Метод Cell:getRowIndex

Метод возвращает индекс текущей строки. Индексация строк начинается с нуля.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("A3")
print(cell:getColumnIndex()) -- 0
print(cell:getRowIndex()) -- 2
```

### 5.12.24 Метод Cell:getTable

Метод возвращает таблицу Table, в которой находится текущая ячейка.

**Вызов**

```
Table getTable()
```

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("A3")
local table = cell:getTable()
table:getCell("A1"):setText("MyText")
```

### 5.12.25 Метод Cell:getTextProperties

Метод возвращает текущие настройки форматирования текста для ячейки.

**Вызов**

```
TextProperties getTextProperties()
```

**Возвращает**

– свойства форматирования текста, тип `TextProperties`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")

local props = cell:getTextProperties()
props.backgroundColor = DocumentAPI.ColorRGBA(0, 0, 255, 200)
props.textColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))

cell:setTextProperties(props)
```

Метод `Cell:setTextProperties` позволяет задать настройки форматирования текста в ячейке.

### 5.12.26 Метод Cell:isContentEmpty

Метод позволяет определить содержит ли ячейка данные.

**Вызов**

```
bool isContentEmpty()
```

**Возвращает**

– `true`, если ячейка не содержит данные, в ином случае – `false`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell1 = sheet:getCell("A1")
```

```
local cell2 = sheet:getCell("A2")
local cell3 = sheet:getCell("A3")
cell1:setText("123")
local props = DocumentAPI.TextProperties()
props.bold = true
cell2:setTextProperties(props)

print(cell1:isEmpty()) -- false
print(cell1:isContentEmpty()) -- false

print(cell2:isEmpty()) -- false
print(cell2:isContentEmpty()) -- true

print(cell3:isEmpty()) -- true
print(cell3:isContentEmpty()) -- true
```

### 5.12.27 Метод Cell:isEmpty

Метод позволяет определить содержит ли ячейка данные или настройки форматирования.

**Вызов**

```
bool isEmpty()
```

**Возвращает**

– `true`, если ячейка не содержит данные или настройки форматирования, в ином случае – `false`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell1 = sheet:getCell("A1")
local cell2 = sheet:getCell("A2")
local cell3 = sheet:getCell("A3")
cell1:setText("123")
local props = DocumentAPI.TextProperties()
props.bold = true
cell2:setTextProperties(props)

print(cell1:isEmpty()) -- false
print(cell1:isContentEmpty()) -- false

print(cell2:isEmpty()) -- false
```

```
print(cell2:isContentEmpty()) -- true

print(cell3:isEmpty()) -- true
print(cell3:isContentEmpty()) -- true
```

### 5.12.28 Метод Cell:isFormula

Метод позволяет определить, содержит ли текущая ячейка формулу.

**Вызов**

```
bool isFormula()
```

**Возвращает**

– `true`, если ячейка содержит формулу, в ином случае – `false`.

### 5.12.29 Метод Cell:isInMergedRange

Метод позволяет определить находится ли ячейка в объединенном диапазоне.

**Вызов**

```
bool isInMergedRange()
```

**Возвращает**

– `true`, если ячейка находится в объединенном диапазоне, в ином случае – `false`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
sheet:getCellRange("A1:C1"):merge()
local cell1 = sheet:getCell("B1")
local cell2 = sheet:getCell("B2")

print(cell1:isInMergedRange()) -- true
print(cell2:isInMergedRange()) -- false
```

### 5.12.30 Метод Cell:isPivotTableRoot

Метод позволяет определить является ли ячейка основанием сводной таблицы.

**Пример**

```
tbl = document:getBlocks():getTable(0)
cell = tbl:getCell("A1")
print("Is pivot table root: " .. tostring((cell:isPivotTableRoot())))
```

### 5.12.31 Метод Cell:isProtected

Метод возвращает статус защиты от редактирования ячейки в табличном документе (см. Защита листа табличного документа).

**Вызов**

```
bool isProtected()
```

### 5.12.32 Метод Cell:removeNote

Метод удаляет заметку из текущей ячейки.

**Вызов**

```
removeNote()
```

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")
if cell:getNote() ~= nil then
    cell:removeNote()
end
```

Методы `Cell:setNote` и `Cell:getNote` позволяют добавить заметку и получить её текст.

### 5.12.33 Метод Cell:setBool

Устанавливает для ячейки значение логического типа.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:getCell("A6"):setBool(true)
```

### 5.12.34 Метод Cell:setBorders

Метод предназначен для установки границ ячейки (тип DocumentAPI.Borders). Примеры использования приведены в разделе DocumentAPI.Borders.

### 5.12.35 Метод Cell:setCellProperties

Позволяет установить свойства ячейки DocumentAPI.CellProperties.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local props = tbl:getCell("A6"):getCellProperties()
props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
tbl:getCell("A6"):setCellProperties(props)
```

### 5.12.36 Метод Cell:setContent

Определяет и устанавливает соответствующую формулу или значение, а затем форматирует ячейку. Устанавливает текст, если автоопределение не удалось.

**Пример**

```
local cell = document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0,
0))
cell:setContent("=A2+A3")
```

### 5.12.37 Метод Cell:setCustomFormat

Устанавливает формат ячейки.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:getCell("A6"):setCustomFormat("0,00")
```

### 5.12.38 Метод Cell:setFormat

Метод устанавливает формат ячейки. Существуют несколько вариантов использования метода.

**Варианты вызова метода**

```
setFormat(cellFormat)
```

Где **cellFormat** – формат ячейки типа DocumentAPI.CellFormat.

```
setFormat(accountingCellFormatting)
```

Где **accountingCellFormatting** – формат ячейки типа DocumentAPI.AccountingCellFormatting.

```
setFormat(percentageCellFormatting)
```

Где **percentageCellFormatting** – формат ячейки типа DocumentAPI.PercentageCellFormatting.

```
setFormat(numberCellFormatting)
```

Где **numberCellFormatting** – формат ячейки типа DocumentAPI.NumberCellFormatting.

```
setFormat(currencyCellFormatting)
```

Где **currencyCellFormatting** – формат ячейки типа DocumentAPI.CurrencyCellFormatting.

```
setFormat(dateTimeCellFormatting, typeFormat)
```

Где **dateTimeCellFormatting** – формат ячейки типа DocumentAPI.DateTimeCellFormatting, **typeFormat** - формат даты/времени типа DocumentAPI.CellFormat.

```
setFormat(fractionCellFormatting)
```

Где **fractionCellFormatting** – формат ячейки типа DocumentAPI.FractionCellFormatting.

```
setFormat(scientificCellFormatting)
```

Где **scientificCellFormatting** – формат ячейки типа DocumentAPI.ScientificCellFormatting.

**Примеры использования**

```
local tbl = document:getBlocks():getTable(0)
local cellA1 = tbl:getCell("A1")

cellA1:setNumber(2.3)

-- Формат: Общий
```

```
cellA1:setFormat(DocumentAPI.CellFormat_General)
print("Формат Общий: ", cellA1:getFormat()) -- 0
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,3

-- Формат: Процентный
local a1CellFormatting = DocumentAPI.PercentageCellFormatting()
a1CellFormatting.decimalPlaces = 1
cellA1:setFormat(a1CellFormatting)
print("Формат Процентный: ", cellA1:getFormat()) -- 1
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 230,0%

-- Формат: Числовой
a1CellFormatting = DocumentAPI.NumberCellFormatting()
a1CellFormatting.decimalPlaces = 2
cellA1:setFormat(a1CellFormatting)
print("Формат Числовой: ", cellA1:getFormat()) -- 2
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30

-- Формат: Денежный
a1CellFormatting = DocumentAPI.CurrencyCellFormatting()
a1CellFormatting.symbol = '$'
cellA1:setFormat(a1CellFormatting)
print("Формат Денежный: ", cellA1:getFormat()) -- 4
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30$

-- Формат: Финансовый
a1CellFormatting = DocumentAPI.AccountingCellFormatting()
a1CellFormatting.symbol = '₽'
cellA1:setFormat(a1CellFormatting)
print("Формат Финансовый: ", cellA1:getFormat()) -- 5
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30₽

-- Формат: Дата / Время
a1CellFormatting = DocumentAPI.DateTimeCellFormatting()
a1CellFormatting.dateListID = DocumentAPI.DatePatterns_FullDate
a1CellFormatting.timeListID = DocumentAPI.TimePatterns_ShortTime
cellA1:setFormat(a1CellFormatting)
print("Формат Дата / Время: ", cellA1:getFormat()) -- 8
print("Значение ячейки: ", cellA1:getRange():extractText()) -- понедельник, 1
января 1900 г. 7:12

-- Формат: Дробный
```

```
a1CellFormatting = DocumentAPI.FractionCellFormatting()
a1CellFormatting.minNumeratorDigits = 2
cellA1:setFormat(a1CellFormatting)
print("Формат Экспоненциальный: ", cellA1:getFormat()) -- 9
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2 2/7

-- Формат: Научный
a1CellFormatting = DocumentAPI.ScientificCellFormatting()
a1CellFormatting.decimalPlaces = 5
cellA1:setFormat(a1CellFormatting)
print("Формат Научный: ", cellA1:getFormat()) -- 10
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30000E+00
```

### 5.12.39 Метод Cell:setFormattedValue

Анализирует переданное значение и автоматически устанавливает формат ячейки и ее значение. В случае, если распознать тип переданного значения не удается, то для ячейки устанавливается формат `DocumentAPI.CellFormat_Text`.

Список поддерживаемых форматов см. в разделе `DocumentAPI.CellFormat`. Пример использования метода см. в разделе Доступ к ячейкам.

### 5.12.40 Метод Cell:setFormula

Метод позволяет вставить формулу в ячейку табличного документа.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:getCell("A1"):setNumber(2.3)
tbl:getCell("A2"):setNumber(3.2)
tbl:getCell("A3"):setFormula("=SUM(A1:A2)") -- 5,5
```

### 5.12.41 Метод Cell:setNote

Метод добавляет заметку в текущую ячейку.

**Вызов**

```
setNote(noteText)
```

**Параметры**

– `noteText`: текст заметки, тип `string`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")
if cell:getNote() == nil then
    cell:setNote("New Note")
end
```

Методы Cell:getNote и Cell:removeNote позволяют получить текст заметки или удалить её.

### 5.12.42 Метод Cell:setNumber

Устанавливает для ячейки значение числового типа.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell(DocumentAPI.CellPosition(0, 0))
cell:setNumber(0.0001)
```

### 5.12.43 Метод Cell:setParagraphProperties

Устанавливает свойства абзаца DocumentAPI.ParagraphProperties, находящегося в ячейке.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A2") --(DocumentAPI.CellPosition(1,0))

local paraProps = cell:getParagraphProperties()
paraProps.alignment = DocumentAPI.Alignment_Center
cell:setParagraphProperties(paraProps)
```

### 5.12.44 Метод Cell:setProtectionProperties

Метод задает параметры защиты ячейки в табличном документе (см. Защита листа табличного документа).

**Вызов**

```
setProtectionProperties(protectionProps)
```

**Параметры**

– `protectionProps`: свойства защиты ячейки, тип `CellProtectionProperties`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")

local cellProps = cell:getProtectionProperties()
cellProps.lockedForChanges = false
cellProps.formulasNotDisplayed = false

cell:setProtectionProperties(cellProps)
firstSheet:setProtection(tableProps)
```

Метод `setProtectionProperties()` позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой защиты). Для установки защиты используйте метод `Table:setProtection()` после задания параметров защиты ячеек. Если метод `setProtectionProperties()` применяется к ячейке уже защищенного листа, возникает исключение `SpreadsheetProtectionError`.

Метод `setProtectionProperties()` не меняет свойства защиты для ячеек сводной таблицы и скрытых/отфильтрованных ячеек.

### 5.12.45 Метод Cell:setText

Устанавливает для ячейки значение строкового типа.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell(DocumentAPI.CellPosition(0, 0))
local trackingChanges = "Disabled"
if document:isChangesTrackingEnabled() then
   trackingChanges = "Enabled"
end
cell:setText(trackingChanges)
```

### 5.12.46 Метод Cell:setTextProperties

Метод задает настройки форматирования текста для ячейки.

**Вызов**

```
setTextProperties(props)
```

**Параметры**

– `props`: свойства форматирования текста, тип `TextProperties`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")

local props = cell:getTextProperties()
props.backgroundColor = DocumentAPI.ColorRGBA(0, 0, 255, 200)
props.textColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))

cell:setTextProperties(props)
```

Метод `Cell:getTextProperties` позволяет получить текущие настройки форматирования текста в ячейке.

### 5.12.47 Метод Cell:unmerge

Разъединяет несколько ячеек, которые были объединены ранее. Примеры объединения и разъединения ячеек см. в разделе Объединение и разделение ячеек таблицы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
-- Ячейка A1 является результатом объединения диапазона A1:A2
tbl:getCell("A1"):unmerge()
```

## 5.13 Таблица DocumentAPI.CellFormat

По умолчанию при создании документа всем ячейкам присваивается формат «Общий». Полный список форматов представлен в таблице 8. Таблица `CellFormat` используется в методах `Cell:getFormat()` и `Cell:setFormat()`, а также в поле `PivotTableValueField.cellNumberFormat`.

Таблица 8 – Поддерживаемые форматы ячеек таблицы

| Значение | Описание |
|---|---|
| `DocumentAPI.CellFormat_General` | Формат ячейки «Общий».<br><br>В этом формате в ячейке отображаются первые 9 символов числа, остальные доступны для просмотра в строке формул. Для дробных чисел в |

| Значение | Описание |
|---|---|
| | формате «Общий» незначащие нули в дробной части не отображаются.<br>Числа, состоящие более чем из 12 символов, переводятся в экспоненциальную форму после завершения ввода в ячейку. |
| `DocumentAPI.CellFormat_Percentage` | Формат ячейки «Процентный».<br><br>Этот формат используется для представления чисел как процентов. При применении формата «Процентный» введенное число умножается на 100 и обозначается знаком «%». |
| `DocumentAPI.CellFormat_Number` | Формат ячейки «Числовой».<br><br>Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя. |
| `DocumentAPI.CellFormat_Text` | Формат ячейки «Текстовый». |
| `DocumentAPI.CellFormat_Currency` | Формат ячейки «Денежный».<br><br>Этот формат используется для представления чисел со знаком или кодом валюты. |
| `DocumentAPI.CellFormat_Accounting` | Формат ячейки «Финансовый».<br><br>Этот формат применяется для чисел, используемых в бухгалтерских документах. В формате «Финансовый» введенное число автоматически дополняется названием валюты, которая соответствует настройкам системы компьютера.<br>Отрицательные числа в формате «Финансовый» заключаются в круглые скобки в ячейке, а в строке формул остаются в том виде, в котором они были введены. |
| `DocumentAPI.CellFormat_Date` | Формат ячейки «Дата».<br><br>Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ. |
| `DocumentAPI.CellFormat_Time` | Формат ячейки «Время».<br><br>Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ. |

| Значение | Описание |
|---|---|
| `DocumentAPI.CellFormat_DateTime` | Формат ячейки «Дата + Время» |
| `DocumentAPI.CellFormat_Fraction` | Формат ячейки «Дробный».<br><br>Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель). |
| `DocumentAPI.CellFormat_Scientific` | Формат ячейки «Экспоненциальный».<br><br>Экспоненциальный (или научный) формат используется для представления больших чисел в короткой форме. Все введенные числа длиной более 12 символов автоматически переводятся в этот формат.<br>В ячейке число в формате «Экспоненциальный» представлено следующим образом:<br>– целая часть, всегда состоящая из одной цифры;<br>– разделитель целой и дробной части;<br>– дробная часть, по умолчанию состоящая из двух цифр;<br>– показатель степени числа 10 в виде **E<знак показателя степени> <показатель степени>**. |
| `DocumentAPI.CellFormat_Custom` | Пользовательский формат |

Использование данных констант позволяет установить выбранный формат. При этом будет использованы параметры формата по умолчанию.

**Примеры использования**

```
local table = document:getBlocks():getTable(0)
local cell_B1 = table:getCell("B1")
cell_B1:setFormat(DocumentAPI.CellFormat_General)
local cell_B2 = table:getCell("B2")
cell_B2:setFormat(DocumentAPI.CellFormat_Percentage)
local cell_B3 = table:getCell("B3")
cell_B3:setFormat(DocumentAPI.CellFormat_Number)
```

Результат выполнения данного примера приведен на рисунке 31.

| | A | B |
|---|---|---|
| 1 | CellFormat.General | 1 |
| 2 | CellFormat.Percentage | 100,00% |
| 3 | CellFormat.Number | 1,00 |

Рисунок 31 – Результат установки формата

Пример форматирования ячейки также приведен в разделе, описывающем установку значений ячеек.

### 5.14 Таблица DocumentAPI.CellPosition

Таблица `DocumentAPI.CellPosition` позволяет задать координаты ячейки табличного документа или таблицы в составе текстового документа. Позиция ячейки A1 имеет координаты (0, 0).

**Конструкторы**

```
CellPosition()
```

```
CellPosition(rowArg, columnArg)
```

– `rowArg` – индекс строки ячейки (индексация начинается с нуля), тип `number`;

– `columnArg` – индекс колонки ячейки, тип `number`.

**Пример**

```
local table = document:getBlocks():getTable(0) -- первый лист документа
local cell = table:getCell(DocumentAPI.CellPosition(2, 0)) -- ячейка A3
```

#### 5.14.1 Поле CellPosition.column

Номер столбца в значении ячейки. Нумерация столбцов начинается с нуля.

**Пример**

```
cellPosition = DocumentAPI.CellPosition()
cellPosition.column = 1
```

#### 5.14.2 Поле CellPosition.row

Номер строки в позиции ячейки. Нумерация строк начинается с нуля.

**Пример**

```
cellPosition = DocumentAPI.CellPosition()
cellPosition.row = 1
```

#### 5.14.3 Метод CellPosition:toString

Возвращает координаты ячейки в формате `(row: R, column: C)`, где `R` и `C` - номер

строки и столбца соответственно.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local pos = DocumentAPI.CellPosition(0,0)
print(pos:toString()) --(row: 0, column: 0)
```

### 5.14.4 Метод CellPosition:__eq

Метод используется для определения эквивалентности двух объектов `CellPosition`.

**Пример**

```
local pos1 = DocumentAPI.CellPosition(0,0)
local pos2 = DocumentAPI.CellPosition(0,0)
print(pos1:__eq(pos2)) -- true
```

## 5.15 Таблица DocumentAPI.CellProperties

Таблица `DocumentAPI.CellProperties` предназначена для форматирования содержимого в ячейках таблицы. Описание полей таблицы `DocumentAPI.CellProperties` представлено в таблице 9.

Для задания свойств ячейки используется метод `Cell:setCellProperties()`. Для получения свойств ячейки используется метод `Cell:getCellProperties()`. Иерархия таблиц и полей `DocumentAPI.CellProperties` отображена на рисунке 32.

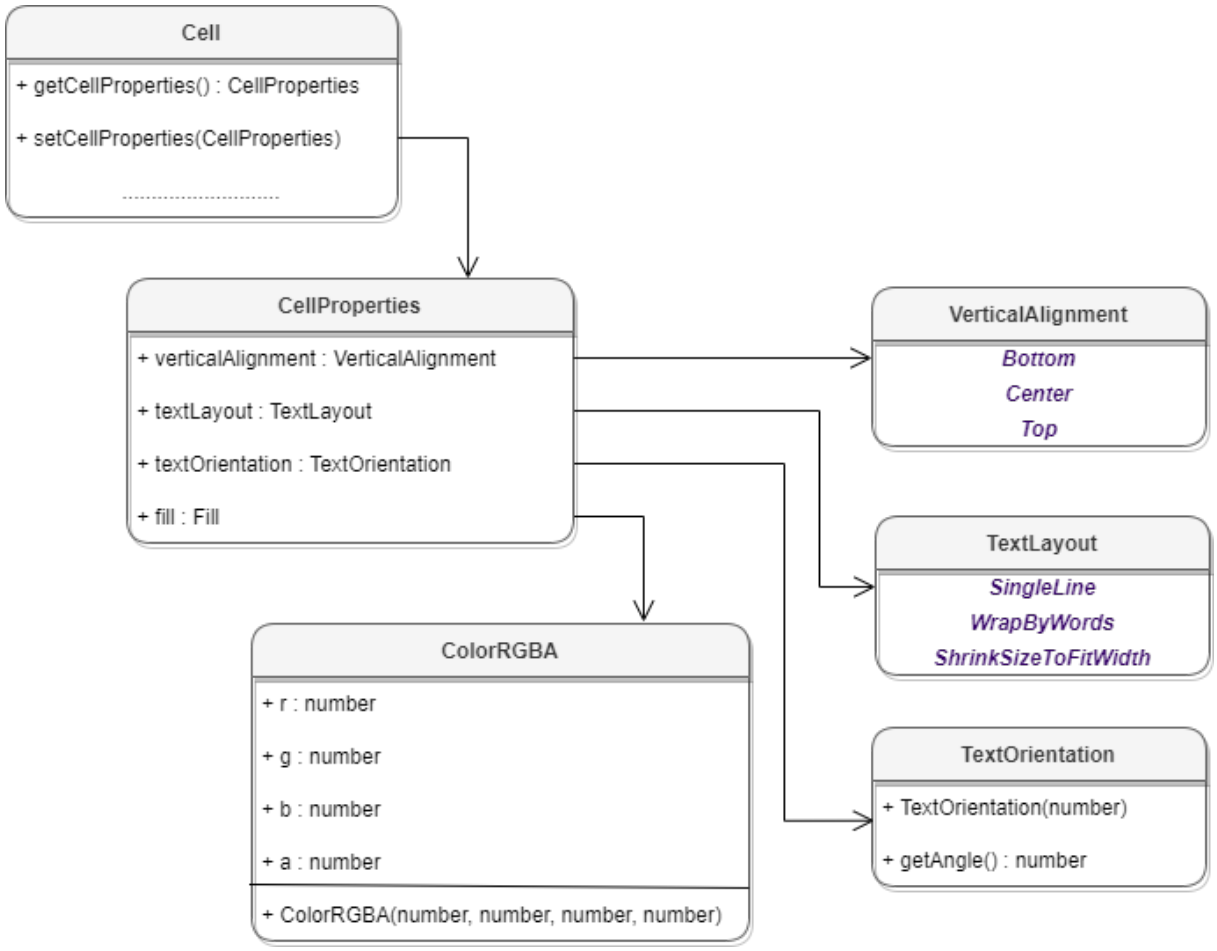



Рисунок 32 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 9 – Описание полей таблицы `DocumentAPI.CellProperties`

| Поле | Тип | Значение |
|---|---|---|
| `CellProperties.verticalAlignment` | `VerticalAlignment` | Вертикальное выравнивание в ячейке |
| `CellProperties.textLayout` | `TextLayout` | Способ отображения значения ячейки |
| `CellProperties.fill` | `Fill` | Заполнение фона ячейки |
| `CellProperties.textOrientation` | `TextOrientation` | Ориентация текста в ячейке (угол поворота) |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("D2") --(DocumentAPI.CellPosition(3,1))
local props = cell:getCellProperties()

props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
```

```
props.textLayout = DocumentAPI.TextLayout_ShrinkSizeToFitWidth
props.fill = DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 255, 0, 255)))
props.textOrientation = DocumentAPI.TextOrientation(45)

cell:setCellProperties(props)
```

### 5.15.1 Метод CellProperties:__eq

Метод используется для определения эквивалентности значений двух объектов `CellProperties`.

**Пример**

```
local cell1 = tbl:getCell("A1")
local cell2 = tbl:getCell("A2")
print("Eq: " ..
tostring(cell1:getCellProperties():__eq(cell2:getCellProperties())))
```

## 5.16 Таблица DocumentAPI.CellProtectionProperties

Таблица `DocumentAPI.CellProtectionProperties` предназначена для настройки параметров защиты ячеек в табличном документе (аналог раздела «Свойства ячеек» в меню «Управление защитой»). Данная таблица используется в методах `Cell:setProtectionProperties()`, `Cell:getProtectionProperties()`, `CellRange:setProtectionProperties()` и `CellRange:getProtectionProperties()`.

Таблица 10 – Описание полей таблицы `DocumentAPI.CellProtectionProperties`

| Поле | Значение по умолчанию | Описание |
|---|---|---|
| `CellProtectionProperties.lockedForChanges` | `true` | Запретить редактирование значения ячейки |
| `CellProtectionProperties.formulasNotDisplayed` | `false` | Отображать в строке формул только результат |

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cell = firstSheet:getCell("A3")

local cellProps = cell:getProtectionProperties()
```

```
cellProps.lockedForChanges = false
cellProps.formulasNotDisplayed = false

cell:setProtectionProperties(cellProps)
firstSheet:setProtection(tableProps)
```

### 5.16.1 Метод CellProtectionProperties:toString

Метод возвращает текстовое представление текущих параметров защиты ячеек в формате: `[lockedForChanges: #, formulasNotDisplayed: #]`.

**Вызов**

```
string toString()
```

**Возвращает**

– текстовое представление параметров защиты ячеек, тип `string`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("A1")
print(cell:getProtectionProperties():toString()) -- [lockedForChanges: true,
formulasNotDisplayed: false]
```

## 5.17 Таблица DocumentAPI.CellRange

Таблица `DocumentAPI.CellRange` описывает диапазон ячеек таблицы.

**Пример**

```
local cellRange = table:getCellRange("B3:C4")
```

### 5.17.1 Метод CellRange:autoFill

Метод `autoFill` заполняет диапазон ячеек, переданный в параметре `destination`, используя в качестве источника ячейки текущего диапазона. Результирующий диапазон формируется из начальной позиции текущего диапазона и последней позиции, определенной аргументом метода (`destination`).

Таким образом, целевой (результирующий) диапазон назначения содержит весь исходный диапазон ячеек. Метод подбирает алгоритм аппроксимации и использует его для экстраполяции исходных значений в результирующем диапазоне.

Форматирование ячейки распространяется на заполненные ячейки. Результат для текстового редактора может отличаться от результата для табличного редактора.

Метод возвращает `True`, если ячейки успешно заполнены, и `False` в других случаях (например, если диапазон ячеек назначения содержит формулу, сводную таблицу и т. д.).

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("A1:A2")
print(rng:autoFill(DocumentAPI.CellPosition(2, 0)))
```

### 5.17.2 Метод CellRange:calculate

Метод пересчитывает формулы в текущем диапазоне ячеек.

**Вызов**

```
calculate()
```

### 5.17.3 Метод CellRange:clearDataValidations

Метод убирает проверку данных из текущего диапазона ячеек.

**Вызов**

```
clearDataValidations()
```

**Пример**

```
local sheet = document:getBlocks():getTable(0)
sheet:getUsedRange():clearDataValidations()
```

### 5.17.4 Метод CellRange:clearFormat

Метод сбрасывает числовой формат диапазона ячеек на **Общий**.

**Вызов**

```
clearFormat(skipHiddenCells)
```

**Параметры**

– `skipHiddenCells`: (необязательный) `true`, не сбрасывать формат для отфильтрованных и скрытых ячеек диапазона, в ином случае – `false`.

### 5.17.5 Метод CellRange:clearStyle

Метод очищает форматирование диапазона ячеек.

**Вызов**

```
clearStyle(skipHiddenCells)
```

**Параметры**

– `skipHiddenCells`: (необязательный) `true`, не удалять форматирование для отфильтрованных и скрытых ячеек диапазона, в ином случае – `false`.

### 5.17.6 Метод CellRange:containsCell

Метод определяет принадлежность ячейки диапазону. В качестве параметра выступает тип `DocumentAPI.Cell`. Если ячейка находится в текущем диапазоне, метод возвращает `true`, в противном случае - `false`. Метод `CellRange:containsCell` может быть использован как для листов табличного документа, так и для таблиц текстового документа.

**Примеры**

```
local table = document:getBlocks():getTable(0)
local cellRange = table:getCellRange("A1:C4")
local cell = table:getCell("A1")

print(cellRange:containsCell(cell))
```

Дополнительный пример использования метода `CellRange:containsCell` приведен в разделе Доступ к ячейкам.

### 5.17.7 Метод CellRange:copyInto

Метод позволяет копировать (аналог **Ctrl+C**, **Ctrl+V** в редакторе таблиц) ячейки текущего диапазона в заданную позицию, представленную параметром типа `DocumentAPI.CellRange`.

Метод `CellRange:copyInto` реализован только в табличных документах и может использоваться в следующих вариантах:

– копирование диапазона ячеек в рамках одного листа табличного документа;
– копирование диапазона ячеек между листами табличного документа.

**Пример (только для табличного документа)**

```
local sourceRange = sheetList:getCellRange("A1:B2")
local destRange = sheetList:getCellRange("C3:D4")
sourceRange:copyInto(destRange)
```

При копировании ячеек в качестве новой позиции достаточно указать верхнюю левую ячейку нового диапазона, однако, если необходимо продублировать исходный блок ячеек, в качестве параметра следует использовать диапазон, превышающий размеры исходного диапазона, но кратный его размерам. Например, при копировании диапазона `"A1:B2"` (размер 2x2) в диапазон `"B5:E6"` (размер 2x4) блок исходных ячеек продублируется два раза (см. Рисунок 33).

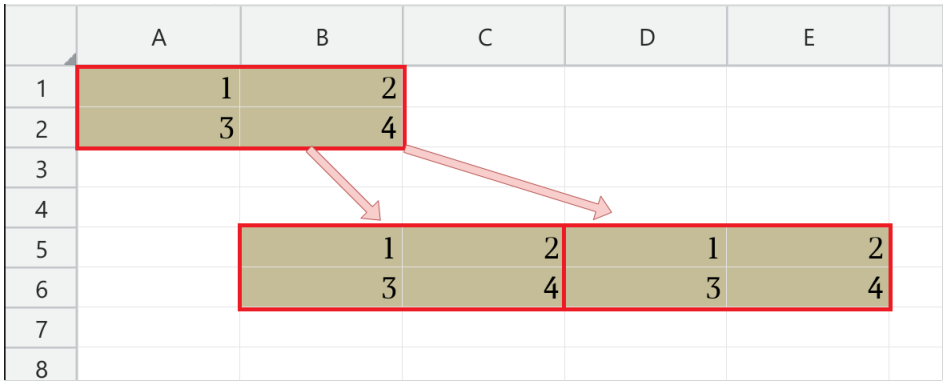



Рисунок 33 – Копирование ячеек табличного документа

Дополнительный пример использования приведен в разделе Копирование ячеек в табличном документе.

### 5.17.8 Метод CellRange:enumerate

Метод возвращает коллекцию ячеек в диапазоне.

**Пример**

```
-- Печать значений ячеек в диапазоне B3:C4
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
for cell in rng:enumerate() do
```

```
    print(cell:getFormattedValue())
end
```

### 5.17.9 Метод CellRange:find

Метод выполняет поиск ячеек, соответствующих заданному запросу, в текущем диапазоне.

**Вызов**

```
function find(string, settings)
```

**Параметры**

– `string`: поисковый запрос, тип `string`.

– `settings`: (необязательный) параметры поиска, тип `TableSearchSettings`.

**Возвращает**

– список ячеек, соответствующих поисковому запросу.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("B1:B20")

local searchProps = DocumentAPI.TableSearchSettings()
searchProps.caseSensitive = DocumentAPI.CaseSensitive_No
searchProps.matchBehaviour = DocumentAPI.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = DocumentAPI.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = true

local results = cellRange:find("*eye", searchProps)
for cell in results do
    print(cell:getFormattedValue()) -- Steeleye Stout
end
```

### 5.17.10 Метод CellRange:getAddress

Метод возвращает адрес диапазона ячеек в заданном формате. Схема адреса: `'[Название_Документа]Название_Листа'!Адрес_Диапазона`.

**Вызов**

```
string getAddress(addressSettings)
```

**Параметры**

– `addressSettings`: настройки форматирования адреса, тип `CellRangeAddressSettings`.

**Возвращает**

– адрес диапазона ячеек.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local range = sheet:getCellRange("A1:C11")

local addressSettings = DocumentAPI.CellRangeAddressSettings()
addressSettings.isFileNameRequired = true
addressSettings.isWorksheetNameRequired = true
addressSettings.addressFormat = DocumentAPI.CellRangeAddressFormat_A1
addressSettings.isAbsoluteRow = true
addressSettings.isAbsoluteColumn = false

print(range:getAddress(addressSettings)) -- '[sheet.xods]Data'!$A1:$C11
```

### 5.17.11 Метод CellRange:getBeginColumn

Метод возвращает индекс столбца первой ячейки диапазона. Нумерация столбцов начинается с нуля.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getBeginColumn()) -- 1
```

### 5.17.12 Метод CellRange:getBeginRow

Метод возвращает индекс строки первой ячейки диапазона. Нумерация строк начинается с нуля.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getBeginRow()) -- 2
```

### 5.17.13 Метод CellRange:getCellProperties

Метод возвращает набор свойств форматирования (DocumentAPI.CellProperties) для диапазона ячеек. Возвращаемая структура содержит свойства, общие для всех ячеек диапазона.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
local cellProperties = rng:getCellProperties()
local colorRGBA = cellProperties.backgroundColor
if (colorRGBA ~= nil) then
   print(colorRGBA.r)
end
```

### 5.17.14 Метод CellRange:getLastColumn

Метод возвращает индекс столбца последней ячейки диапазона. Нумерация столбцов начинается с нуля.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getLastColumn()) -- 2
```

### 5.17.15 Метод CellRange:getLastRow

Метод возвращает индекс строки последней ячейки диапазона. Нумерация строк начинается с нуля.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getLastRow()) -- 3
```

### 5.17.16 Метод CellRange:getParagraphProperties

Метод возвращает текущие настройки форматирования абзацев, находящихся в диапазоне ячеек.

**Вызов**

```
ParagraphProperties getParagraphProperties(skipHiddenCells)
```

**Параметры**

– `skipHiddenCells`: (необязательный, по умолчанию `true`) не учитывать настройки отфильтрованных и скрытых ячеек диапазона, тип `bool`.

**Возвращает**

– свойства форматирования абзацев, тип `ParagraphProperties`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cellRange = firstSheet:getCellRange("A1:C3")

local paragraphProperties = cellRange:getParagraphProperties()
paragraphProperties.alignment = DocumentAPI.Alignment_Center

cellRange:setParagraphProperties(paragraphProperties, false)
```

Свойства возвращаемого объекта `ParagraphProperties` могут быть `nil`, если выбранный диапазон содержит ячейки с разными параметрами. Метод `CellRange:setParagraphProperties` позволяет задать настройки форматирования абзацев, находящихся в диапазоне ячеек.

### 5.17.17 Метод CellRange:getProtectionProperties

Метод возвращает параметры защиты диапазона ячеек табличного документа (см. Защита листа табличного документа).

**Вызов**

```
CellProtectionProperties getProtectionProperties()
```

**Возвращает**

– `CellProtectionProperties`: свойства защиты диапазона ячеек (`nil`, если диапазон содержит только ячейки сводной таблицы).

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cellRange = firstSheet:getCellRange("A1:A3")

local cellProps = cellRange:getProtectionProperties()
cellProps.lockedForChanges = false
```

```
cellProps.formulasNotDisplayed = false

cellRange:setProtectionProperties(cellProps)
firstSheet:setProtection(tableProps)
```

Свойства возвращаемого объекта CellProtectionProperties могут быть `nil`, если текущий диапазон содержит ячейки с разными параметрами.

### 5.17.18 Метод CellRange:getTable

Метод возвращает таблицу (Table) для диапазона ячеек.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getTable())
```

### 5.17.19 Метод CellRange:getTableRange

Возвращает положение текущего диапазона ячеек в таблице (объект CellRangePosition).

### 5.17.20 Метод CellRange:getTextProperties

Метод возвращает текущие настройки форматирования текста для диапазона ячеек.

**Вызов**

```
TextProperties getTextProperties(skipHiddenCells)
```

**Параметры**

– `skipHiddenCells`: (необязательный, по умолчанию `true`) не учитывать настройки отфильтрованных и скрытых ячеек диапазона, тип `bool`.

**Возвращает**

– свойства форматирования текста, тип TextProperties.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cellRange = firstSheet:getCellRange("A1:C3")

local props = cellRange:getTextProperties()
```

```
props.backgroundColor = DocumentAPI.ColorRGBA(0, 0, 255, 200)
props.textColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))

cellRange:setTextProperties(props)
```

Свойства возвращаемого объекта `TextProperties` могут быть `nil`, если выбранный диапазон содержит ячейки с разными параметрами. Метод `CellRange:setTextProperties` позволяет задать настройки форматирования текста в диапазоне ячеек.

### 5.17.21 Метод CellRange:insert

Метод вставляет пустые ячейки на место текущего диапазона и сдвигает существующие ячейки диапазона в заданном направлении. Метод не позволяет сдвинуть защищенные ячейки и сводные таблицы, а также части диапазонов фильтрации, умных таблиц и объединенных ячеек.

**Вызов**

```
insert(shiftAxis)
```

**Параметры**

– `shiftAxis`: направление сдвига текущих ячеек диапазона (вправо или вниз), тип `CellShiftAxis`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("A1:C3")

cellRange:insert(DocumentAPI.CellShiftAxis_Vertical)
```

### 5.17.22 Метод CellRange:insertCurrentDateTime

Метод служит для установки значения даты/времени `DocumentAPI.DateTimeFormat` диапазона ячеек.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cellRange = tbl:getCellRange("A1:B2")
cellRange:insertCurrentDateTime(DocumentAPI.DateTimeFormat_Date)
```

### 5.17.23 Метод CellRange:intersect

Метод возвращает диапазон ячеек, который является пересечением текущего и заданного диапазонов ячеек.

**Вызов**

```
CellRange intersect(other)
```

**Параметры**

– `other`: диапазон ячеек, тип `CellRange`.

**Возвращает**

– диапазон пересечения, тип `CellRange`.

– `nil`, если диапазоны ячеек не пересекаются.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange1 = sheet:getCellRange("F2:F6")
local cellRange2 = sheet:getCellRange("A4:H6")
local cells = cellRange1:intersect(cellRange2)
print(cells:getAddress(DocumentAPI.CellRangeAddressSettings())) -- F4:F6
```

### 5.17.24 Метод CellRange:isContentEmpty

Метод позволяет определить содержат ли ячейки диапазона данные.

**Вызов**

```
bool isContentEmpty()
```

**Возвращает**

– `true`, если все ячейки диапазона не содержат данные, в ином случае – `false`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local range1 = sheet:getCellRange("A1:B1")
local range2 = sheet:getCellRange("A2:B2")
local range3 = sheet:getCellRange("A3:B3")
sheet:getCell("A1"):setText("123")
local props = DocumentAPI.TextProperties()
props.bold = true
range2:setTextProperties(props)
```

```
print(range1:isEmpty()) -- false
print(range1:isContentEmpty()) -- false

print(range2:isEmpty()) -- false
print(range2:isContentEmpty()) -- true

print(range3:isEmpty()) -- true
print(range3:isContentEmpty()) -- true
```

### 5.17.25 Метод CellRange:isEmpty

Метод позволяет определить содержат ли ячейки диапазона данные или настройки форматирования.

**Вызов**

```
bool isEmpty()
```

**Возвращает**

– `true`, если все ячейки диапазона не содержат данные или настройки форматирования, в ином случае – `false`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local range1 = sheet:getCellRange("A1:B1")
local range2 = sheet:getCellRange("A2:B2")
local range3 = sheet:getCellRange("A3:B3")
sheet:getCell("A1"):setText("123")
local props = DocumentAPI.TextProperties()
props.bold = true
range2:setTextProperties(props)

print(range1:isEmpty()) -- false
print(range1:isContentEmpty()) -- false

print(range2:isEmpty()) -- false
print(range2:isContentEmpty()) -- true

print(range3:isEmpty()) -- true
print(range3:isContentEmpty()) -- true
```

### 5.17.26 Метод CellRange:isProtected

Метод возвращает статус защиты от редактирования диапазона ячеек в табличном документе (см. Защита листа табличного документа).

**Вызов**

```
bool isProtected()
```

Метод `isProtected()` возвращает `nil`, если часть ячеек диапазона защищена от редактирования, а часть – нет.

### 5.17.27 Метод CellRange:merge

Метод объединяет несколько ячеек таблицы в одну. Группа ячеек (диапазон) формируется с помощью таблицы `CellRange`. Содержимое крайней левой ячейки диапазона помещается в объединенной ячейке. Примеры объединения и разъединения ячеек см. в разделе Объединение и разделение ячеек таблицы.

**Пример**

```
-- Объединение ячеек A1 и A2 на первом листе табличного документа
local tbl = document:getBlocks():getTable(0)
tbl:getCellRange("A1:A2"):merge()
```

### 5.17.28 Метод CellRange:moveInto

Метод позволяет переносить (аналог **Ctrl+X**, **Ctrl+V** в редакторе таблиц) ячейки текущего диапазона в заданную позицию, представленную параметром типа `DocumentAPI.CellRange`.

Метод `CellRange:moveInto` реализован только в табличных документах и может использоваться в следующих вариантах:

- перемещение диапазона ячеек в рамках одного листа табличного документа;
- перемещение диапазона ячеек между листами табличного документа.

**Пример (только для табличного документа)**

```
local sourceRange = sheetList:getCellRange("A1:B2")
local destRange = sheetList:getCellRange("C3:D4")
sourceRange:moveInto(destRange)
```

При перемещении ячеек в качестве новой позиции достаточно указать верхнюю левую ячейку нового диапазона, однако, при необходимости можно продублировать исходный блок ячеек в новом местоположении (см. подробности в разделе `CellRange.CopyInto`).

Дополнительный пример использования приведен в разделе Копирование ячеек в табличном документе.

### 5.17.29 Метод CellRange:remove

Метод удаляет ячейки текущего диапазона и сдвигает на их место оставшиеся ячейки в заданном направлении. Метод не позволяет сдвинуть защищенные ячейки и сводные таблицы, а также части диапазонов фильтрации, умных таблиц и объединенных ячеек.

**Вызов**

```
remove(shiftAxis)
```

**Параметры**

– `shiftAxis`: направление сдвига ячеек (влево или вверх), тип `CellShiftAxis`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("A1:C3")

cellRange:remove(DocumentAPI.CellShiftAxis_Horizontal)
```

### 5.17.30 Метод CellRange:removeContent

Метод очищает содержимое диапазона ячеек.

**Вызов**

```
removeContent(skipHiddenCells)
```

**Параметры**

– `skipHiddenCells`: (необязательный) `true`, не удалять содержимое отфильтрованных и скрытых ячеек диапазона, в ином случае – `false`.

### 5.17.31 Метод CellRange:setArrayFormula

Метод вставляет формулу массива в текущий диапазон ячеек. Формула массива – это формула, в процессе вычисления которой создается один или несколько массивов. С помощью таких формул вы можете выполнять операции с массивами и диапазонами данных.

**Вызов**

```
setArrayFormula(arrayFormula)
```

**Параметры**

– `arrayFormula`: формула массива, тип `string`.

**Примеры**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("B1:B7")
cellRange:setArrayFormula("=A1:A7 * {1;2;3;4;5;6;7}")
local cell = sheet:getCell("B3")
print(cell:getFormattedValue())  -- 3
print(cell:getFormulaAsString()) -- {=A1:A7*{1; 2; 3; 4; 5; 6; 7}}
```

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("C1:C7")
cellRange:setArrayFormula("=A1:A7 > 0")
local cell = sheet:getCell("C3")
print(cell:getFormattedValue())  -- ИСТИНА
print(cell:getFormulaAsString()) -- {=A1:A7>0}
```

### 5.17.32 Метод CellRange:setBorders

Метод предназначен для установки границ диапазона ячеек. Отдельные границы устанавливаются с помощью методов таблицы DocumentAPI.Borders.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("C3")
--
line_prop = DocumentAPI.LineProperties()
line_prop.style = DocumentAPI.LineStyle_Dash
line_prop.width = 1.5
line_prop.color = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))
--
newBorders = DocumentAPI.Borders()
newBorders = newBorders:setLeft(line_prop)
newBorders = newBorders:setRight(line_prop)
newBorders = newBorders:setTop(line_prop)
newBorders = newBorders:setBottom(line_prop)
--
local borders = cell:setBorders(newBorders)
```

### 5.17.33 Метод CellRange:setCellProperties

Метод предназначен для установки свойств DocumentAPI.CellProperties для всех ячеек диапазона.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
local props = DocumentAPI.CellProperties()
props.backgroundColor = DocumentAPI.ColorRGBA(55, 146, 179, 200)
rng:setCellProperties(props)
```

### 5.17.34 Метод CellRange:setDataValidation

Метод устанавливает проверку данных для текущего диапазона ячеек.

**Вызов**

```
setDataValidation(DataValidation dataValidation)
```

**Параметры**

– `dataValidation`: настройки проверки данных, тип DataValidation.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("E2:E10")
local dvList = DocumentAPI.DataValidation()
dvList:setType(DocumentAPI.DataValidationType_List)
dvList:setFormula1("UK; Italy; Germany; Austria; Brazil")
dvList:setShowDropDown(true)

cellRange:setDataValidation(dvList)
```

### 5.17.35 Метод CellRange:setParagraphProperties

Метод задает настройки форматирования абзацев, находящихся в диапазоне ячеек.

**Вызов**

```
setParagraphProperties(paraProperties, skipHiddenCells)
```

**Параметры**

– `paraProperties`: свойства форматирования абзацев, тип ParagraphProperties.

– `skipHiddenCells`: (необязательный, по умолчанию `true`) не применять настройки к отфильтрованным и скрытым ячейкам диапазона, тип `bool`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cellRange = firstSheet:getCellRange("A1:C3")

local paragraphProperties = cellRange:getParagraphProperties()
paragraphProperties.alignment = DocumentAPI.Alignment_Center

cellRange:setParagraphProperties(paragraphProperties, false)
```

Метод `CellRange:getParagraphProperties` позволяет получить текущие настройки форматирования абзацев, находящихся в диапазоне ячеек.

### 5.17.36 Метод CellRange:setProtectionProperties

Метод задает параметры защиты диапазона ячеек в табличном документе (см. Защита листа табличного документа).

**Вызов**

```
setProtectionProperties(protectionProps)
```

**Параметры**

– `protectionProps`: свойства защиты диапазона ячеек, тип `CellProtectionProperties`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cellRange = firstSheet:getCellRange("A1:A3")

local cellProps = cellRange:getProtectionProperties()
cellProps.lockedForChanges = false
cellProps.formulasNotDisplayed = false

cellRange:setProtectionProperties(cellProps)
firstSheet:setProtection(tableProps)
```

Метод `setProtectionProperties()` позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой защиты). Для установки защиты используйте метод `Table:setProtection()` после задания

параметров защиты ячеек. Если метод `setProtectionProperties()` применяется к ячейкам уже защищенного листа, возникает исключение `SpreadsheetProtectionError`.

Метод `setProtectionProperties()` не меняет свойства защиты для ячеек сводной таблицы и скрытых/отфильтрованных ячеек.

### 5.17.37 Метод CellRange:setTextProperties

Метод задает настройки форматирования текста для диапазона ячеек.

**Вызов**

```
setTextProperties(textProperties, skipHiddenCells)
```

**Параметры**

– `textProperties`: свойства форматирования текста, тип `TextProperties`.

– `skipHiddenCells`: (необязательный, по умолчанию `true`) не применять настройки к отфильтрованным и скрытым ячейкам диапазона, тип `bool`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
local cellRange = firstSheet:getCellRange("A1:C3")

local props = cellRange:getTextProperties()
props.backgroundColor = DocumentAPI.ColorRGBA(0, 0, 255, 200)
props.textColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))

cellRange:setTextProperties(props)
```

Метод `CellRange:getTextProperties` позволяет получить текущие настройки форматирования текста в диапазоне ячеек.

### 5.17.38 Метод CellRange:sort

Метод сортирует строки текущего диапазона в соответствии с заданными условиями.

**Вызов**

```
sort(sortingConditions)
```

**Параметры**

– `sortingConditions`: коллекция условий сортировки ячеек, тип `SortingConditions`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local range = sheet:getCellRange("A1:C11")

local conditions = DocumentAPI.SortingConditions()
conditions:add(0, DocumentAPI.SortingDirection_Descending)
conditions:add(1, DocumentAPI.SortingDirection_Ascending)

range:sort(conditions)
```

Метод вызывает `DocumentModificationError` в следующих случаях:

– Индекс столбца выходит за пределы текущего диапазона

– Диапазон содержит объединенные ячейки

– Диапазон содержит формулы

– В диапазоне присутствуют ячейки сводной таблицы

– Диапазон заблокирован от изменений

### 5.17.39 Метод CellRange:textToColumns

Метод разделяет текст в каждой ячейке диапазона на отдельные горизонтальные ячейки. Символы-разделители и настройки разделения задаются с помощью объекта `TextToColumnsSettings`. Данный метод можно использовать только у диапазона шириной в одну ячейку.

**Вызов**

```
textToColumns(settings)
```

**Параметры**

– `settings`: параметры разделения текста по ячейкам, тип `TextToColumnsSettings`.

**Пример**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | 1, Laptop, Moscow, 1500, 8, 'fragile, express' | | | | | |
| 2 | 2, UPS, 'Saint Petersburg', 600, 12, 'heavy, premium' | | | | | |
| 3 | 3, Mouse, Kazan, 200, 5, 'free shipping, standard' | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | 1 | Laptop | Moscow | 1500 | 8 | fragile, express |
| 2 | 2 | UPS | Saint Petersburg | 600 | 12 | heavy, premium |
| 3 | 3 | Mouse | Kazan | 200 | 5 | free shipping, standard |
| 4 | | | | | | |
| 5 | | | | | | |

Рисунок 35 – Разделение текста диапазона по ячейкам

```
local sheet = document:getBlocks():getTable(0)
local cell1 = sheet:getCell("A1")
local cell2 = sheet:getCell("A2")
local cell3 = sheet:getCell("A3")
cell1:setText("1, Laptop, Moscow, 1500, 8, 'fragile, express'")
cell2:setText("2, UPS, 'Saint Petersburg', 600, 12, 'heavy, premium'")
cell3:setText("3, Mouse, Kazan, 200, 5, 'free shipping, standard'")

local settings = DocumentAPI.TextToColumnsSettings()
settings.useCommaDelimiter = true
settings.useSpaceDelimiter = true
settings.treatMultipleDelimitersAsOne = true
settings.textQualifier =
DocumentAPI.TextToColumnsSettings.TextQualifier_SingleQuotes

local cellRange = sheet:getCellRange("A1:A3")
cellRange:textToColumns(settings)
```

### 5.18 Таблица DocumentAPI.CellRangeAddressFormat

Таблица `CellRangeAddressFormat` определяет формат отображения адреса диапазона ячеек. Используется в поле `CellRangeAddressSettings.addressFormat`.

Таблица 11 – Описание форматов отображения адреса

| Значение | Описание |
|---|---|
| `DocumentAPI.CellRangeAddressFormat_A1` | Отображение адреса в формате A1 (`A1:C11`) |
| `DocumentAPI.CellRangeAddressFormat_R1C1` | Отображение адреса в формате R1C1 (`R[0]C[0]:R[10]C[2]`) |

### 5.19 Таблица DocumentAPI.CellRangeAddressSettings

Таблица `CellRangeAddressSettings` предназначена для настройки параметров отображения адреса диапазона ячеек. Данная таблица используется в методе `CellRange:getAddress()`.

Таблица 12 – Описание полей таблицы `CellRangeAddressSettings`

| Поле | Тип | Описание |
|---|---|---|
| `CellRangeAddressSettings.addressFormat` | `CellRangeAddressFormat` | Формат адреса (A1 или R1C1) |
| `CellRangeAddressSettings.isAbsoluteColumn` | `bool` | Использовать абсолютные ссылки на столбцы |
| `CellRangeAddressSettings.isAbsoluteRow` | `bool` | Использовать абсолютные ссылки на строки |
| `CellRangeAddressSettings.isFileNameRequired` | `bool` | Добавить в адрес название документа |
| `CellRangeAddressSettings.isWorksheetNameRequired` | `bool` | Добавить в адрес название листа |

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local range = sheet:getCellRange("A1:C11")

local addressSettings = DocumentAPI.CellRangeAddressSettings()
addressSettings.isFileNameRequired = true
addressSettings.isWorksheetNameRequired = true
addressSettings.addressFormat = DocumentAPI.CellRangeAddressFormat_A1
addressSettings.isAbsoluteRow = true
addressSettings.isAbsoluteColumn = false

print(range:getAddress(addressSettings)) -- '[sheet.xods]Data'!$A1:$C11
```

### 5.20 Таблица DocumentAPI.CellRangePosition

Таблица `DocumentAPI.CellRangePosition` представляет положение диапазона ячеек в таблице. Объект `DocumentAPI.CellRangePosition` используется в методах `Table:getCellRange()`, `CellRange:getTableRange()` и `Chart:setRange()`. По умолчанию диапазон включает одну ячейку в позиции 0,0 что соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.

**Конструкторы**

```
CellRangePosition()
```

```
CellRangePosition(topLeftArg, bottomRightArg)
```

– `topLeftArg` – позиция левой верхней ячейки диапазона, тип `CellPosition`;

– `bottomRightArg` – позиция правой нижней ячейки диапазона, тип `CellPosition`.

```
CellRangePosition(topLeftRow, topLeftColumn, bottomRightRow, bottomRightColumn)
```

– `topLeftRow` – индекс верхней строки диапазона (индексация начинается с нуля), тип `number`;

– `topLeftColumn` – индекс левого столбца диапазона, тип `number`;

– `bottomRightRow` – индекс нижней строки диапазона, тип `number`;

– `bottomRightColumn` – индекс правого столбца диапазона, тип `number`.

Таблица 13 – Поля таблицы `DocumentAPI.CellRangePosition`

| Поле | Тип | Описание |
|---|---|---|
| `CellRangePosition.topLeft` | `CellPosition` | Позиция левой верхней ячейки таблицы прямоугольного диапазона. Значение 0,0 соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц. |
| `CellRangePosition.bottomRight` | `CellPosition` | Содержит позицию правой нижней ячейки таблицы прямоугольного диапазона. |

**Примеры**

```
local table = document:getBlocks():getTable(0)
local leftTopCellPosition = DocumentAPI.CellPosition(0, 0)
local rightBottomCellPosition = DocumentAPI.CellPosition(5, 5)
local cellRangePosition = DocumentAPI.CellRangePosition(leftTopCellPosition, rightBottomCellPosition)
local range = table:getCellRange(cellRangePosition)
```

```
local table = document:getBlocks():getTable(0)
local cellRangePosition = DocumentAPI.CellRangePosition(0, 0, 5, 5)
local range = table:getCellRange(cellRangePosition)
```

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local rangeInfo = charts:getChart(0):getRange(0)
```

```
local cellRangePosition = rangeInfo.tableRangeInfo.tableRange
print("topLeft=", cellRangePosition.topLeft.row,
cellRangePosition.topLeft.column)
print("bottomRight=", cellRangePosition.bottomRight.row,
cellRangePosition.bottomRight.column)
```

### 5.20.1 Метод CellRangePosition:toString

Возвращает информацию о диапазоне ячеек в виде строкового значения формата `(topLeft: <value>, bottomRight: <value>)`.

**Пример**

```
local cellRangePosition = DocumentAPI.CellRangePosition(0, 0, 5, 5)
print(cellRangePosition:toString()) -- [topLeft: (row: 0, column: 0),
bottomRight: (row: 5, column: 5)]
```

### 5.20.2 Метод CellRangePosition:__eq

Метод используется для определения эквивалентности двух объектов `CellRangePosition`.

**Пример**

```
local pos1 = DocumentAPI.CellRangePosition(0, 0, 5, 5)
local pos2 = DocumentAPI.CellRangePosition(0, 0, 5, 5)
print(pos1:__eq(pos2)) -- true
```

## 5.21 Таблица DocumentAPI.CellRanges

Таблица `DocumentAPI.CellRanges` представляет собой коллекцию диапазонов ячеек (DocumentAPI.CellRange).

**Конструктор**

```
DocumentAPI.CellRanges(document)
```

### 5.21.1 Метод CellRanges:addRange

Метод добавляет заданный диапазон ячеек в коллекцию.

**Вызов**

```
addRange(range)
```

**Параметры**

– `range`: диапазон ячеек, тип `CellRange`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellrange = sheet:getCellRange("A1:A3")
local cellrange1 = sheet:getCellRange("C1:C3")

ranges = DocumentAPI.CellRanges(document)
ranges:addRange(cellrange)
ranges:addRange(cellrange1)
```

### 5.21.2 Метод CellRanges:clear

Метод очищает текущую коллекцию диапазонов ячеек.

**Вызов**

```
clear()
```

### 5.21.3 Метод CellRanges:enumerate

Метод возвращает перечисление диапазонов ячеек.

**Вызов**

```
function enumerate()
```

**Возвращает**

– перечисление диапазонов ячеек.

**Пример**

```
local ranges = EditorAPI.getSelection()
for range in ranges:enumerate() do
    print(range:getTableRange():toString())
end
```

## 5.22 Таблица DocumentAPI.CellShiftAxis

Таблица `CellShiftAxis` определяет направление сдвига ячеек при вставке или удалении диапазона. Используется в методах `CellRange:insert()` и `CellRange:remove()`.

Таблица 14 – Описание направлений сдвига ячеек

| Значение | Описание |
|---|---|
| `DocumentAPI.CellShiftAxis_Horizontal` | Сдвигает ячейки по горизонтали (вправо при вставке, влево при удалении) |
| `DocumentAPI.CellShiftAxis_Vertical` | Сдвигает ячейки по вертикали (вниз при вставке, вверх при удалении) |

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("A1:C3")

cellRange:insert(DocumentAPI.CellShiftAxis_Vertical)
```

### 5.23 Таблица DocumentAPI.Chart

Таблица `DocumentAPI.Chart` представляет диаграмму в табличном документе и описывает все ее элементы (заголовок, легенда, тип, данные, диапазон и т.д). Объектная модель `DocumentAPI.Chart` приведена на рисунке 36.

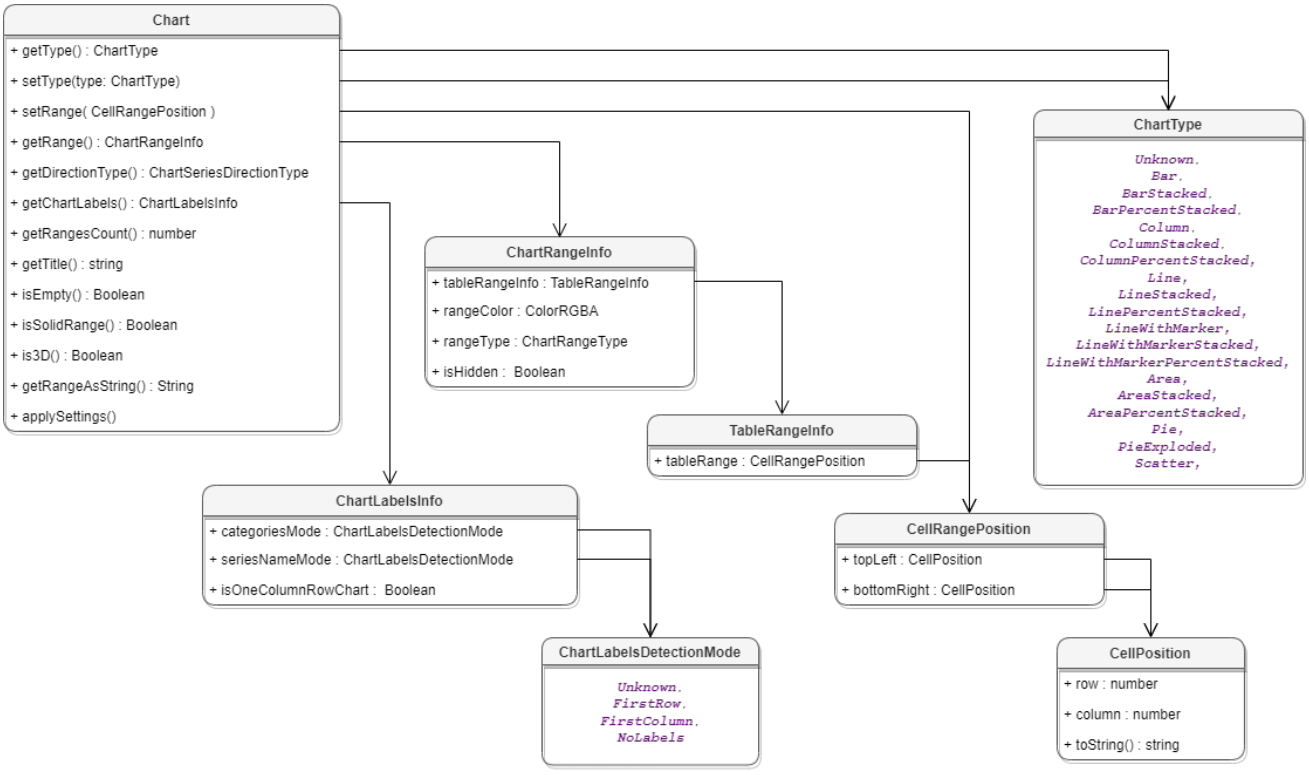



Рисунок 36 – Объектная модель таблицы `DocumentAPI.Chart`

### 5.23.1 Метод Chart:applySettings

Метод позволяет обновить параметры текущей выбранной диаграммы.

**Вызов**

```
applySettings(cellRange, directionType, title, labelsInfo)
```

**Параметры**

– `cellRange` – обновленный диапазон исходных данных диаграммы `DocumentAPI.CellRange`;

– `directionType` – направление серий `DocumentAPI.ChartSeriesDirectionType`;

– `title` – заголовок диаграммы (тип - строка);

– `labelsInfo` – информация о метках диаграммы `DocumentAPI.ChartLabelsInfo`.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()

local cellRange = tbl:getCellRange("B3:C4")
local directionType = DocumentAPI.ChartSeriesDirectionType_ByRow
local title = 'Title'
local chartLabelsInfo =
DocumentAPI.ChartLabelsInfo(DocumentAPI.ChartLabelsDetectionMode_FirstRow,
DocumentAPI.ChartLabelsDetectionMode_FirstRow, false)

charts:getChart(0):applySettings(cellRange, directionType, title,
chartLabelsInfo)
```

### 5.23.2 Метод Chart:getChartLabels

Метод возвращает коллекцию меток диаграммы типа `DocumentAPI.ChartLabelsInfo`.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local chart = charts:getChart(0)
local chartLabelsInfo = chart:getChartLabels()
print(chartLabelsInfo.categoriesMode, chartLabelsInfo.seriesNameMode,
chartLabelsInfo.isOneColumnRowChart)
```

### 5.23.3 Метод Chart:getDirectionType

Метод возвращает направление DocumentAPI.ChartSeriesDirectionType серий диаграммы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getDirectionType())
```

### 5.23.4 Метод Chart:getRange

Метод возвращает диапазон ячеек DocumentAPI.ChartRangeInfo с исходными данными диаграммы. Параметр `rangesIndex` – индекс диапазона.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRange(0).rangeType)
```

### 5.23.5 Метод Chart:getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRangeAsString())
```

### 5.23.6 Метод Chart:getRangesCount

Метод возвращает количество серий диаграммы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRangesCount())
```

### 5.23.7 Метод Chart:getTitle

Метод возвращает заголовок диаграммы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getTitle())
```

### 5.23.8 Метод Chart:getType

Метод возвращает тип диаграммы `DocumentAPI.ChartType`.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getType())
```

### 5.23.9 Метод Chart:is3D

Метод возвращает `true`, если диаграмма трехмерная.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):is3D())
```

### 5.23.10 Метод Chart:isEmpty

Метод возвращает `true`, если диаграмма не содержит значений.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):isEmpty())
```

### 5.23.11 Метод Chart:isSolidRange

Метод возвращает `true`, если диапазон исходных данных диаграммы может быть выделен одним прямоугольником и не имеет промежутков.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):isSolidRange())
```

### 5.23.12 Метод Chart:setRange

Метод задает диапазон DocumentAPI.CellRangePosition ячеек с исходными данными для диаграммы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local cellRangePosition = DocumentAPI.CellRangePosition(0, 0, 5, 5)
charts:getChart(0):setRange(cellRangePosition)
```

### 5.23.13 Метод Chart:setRect

Метод задает область расположения диаграммы, параметр `rect` – новая область.

Внимание ! Метод устаревший (deprecated), оставлен для обратной совместимости и не рекомендован к использованию.

### 5.23.14 Метод Chart:setType

Метод устанавливает тип диаграммы DocumentAPI.ChartType. Параметр `chartType` - новый тип диаграммы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
charts:getChart(0):setType(DocumentAPI.ChartType_LineStacked)
print(charts:getChart(0):getType())
```

## 5.24 Таблица DocumentAPI.ChartLabelsDetectionMode

Таблица `DocumentAPI.ChartLabelsDetectionMode` описывает режимы автоматического определения меток диаграмм. Используется в полях ChartLabelsInfo.categoriesMode и ChartLabelsInfo.seriesNameMode.

Таблица 15 – Описание полей таблицы `DocumentAPI.ChartLabelsDetectionMode`

| Значение | Описание |
|---|---|
| `DocumentAPI.ChartLabelsDetectionMode_Unknown` | Неопределенный тип |
| `DocumentAPI.ChartLabelsDetectionMode_FirstRow` | Метка на первой строке |
| `DocumentAPI.ChartLabelsDetectionMode_FirstColumn` | Метка на первой колонке |
| `DocumentAPI.ChartLabelsDetectionMode_NoLabels` | Не отрисовывать метки |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local chart = charts:getChart(0)
local chartLabels = chart:getChartLabels()
print(chartLabels.categoriesMode, chartLabels.seriesNameMode)
```

### 5.25 Таблица DocumentAPI.ChartLabelsInfo

Таблица `DocumentAPI.ChartLabelsInfo` описывает настройки автоматического определения меток диаграммы. Используется в методах `Chart:applySettings()` и `Chart:getChartLabels()`.

**Конструктор**

```
ChartLabelsInfo(categoriesMode, seriesNameMode, oneColumnRow)
```

- `categoriesMode` - режим автоматического определения меток для категорий, тип `DocumentAPI.ChartLabelsDetectionMode`;
- `seriesNameMode` - режим автоматического определения меток для серий, тип `DocumentAPI.ChartLabelsDetectionMode`;
- `oneColumnRow` - передается `true`, если диапазон диаграммы содержит только одну строку или одну колонку.

Таблица 16 – Описание полей таблицы `DocumentAPI.ChartLabelsInfo`

| Поле | Тип | Описание |
|---|---|---|
| `ChartLabelsInfo.categoriesMode` | `ChartLabelsDetectionMode` | Режим автоматического определения меток для категорий |
| `ChartLabelsInfo.seriesNameMode` | `ChartLabelsDetectionMode` | Режим автоматического определения меток для серий |
| `ChartLabelsInfo.isOneColumnRowChart` | `bool` | Поле содержит `true`, если диапазон диаграммы содержит только одну строку или одну колонку |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local chart = charts:getChart(0)
local chartLabelsInfo = chart:getChartLabels()
print(chartLabelsInfo.categoriesMode, chartLabelsInfo.seriesNameMode,
chartLabelsInfo.isOneColumnRowChart)
```

### 5.26 Таблица DocumentAPI.ChartRangeInfo

Таблица `DocumentAPI.ChartRangeInfo` описывает серию диаграммы. Используется в методе `Chart:getRange()`.

**Конструктор**

```
ChartRangeInfo(cellRange, color, hidden, type)
```

Таблица 17 – Описание полей таблицы `DocumentAPI.ChartRangeInfo`

| Поле | Тип | Описание |
|---|---|---|
| `ChartRangeInfo.cellRange` | `CellRange` | Диапазон ячеек диаграммы |
| `ChartRangeInfo.rangeColor` | `ColorRGBA` | Цвет для отрисовки серии |
| `ChartRangeInfo.isHidden` | `bool` | Задает видимость серии диаграммы |

| Поле | Тип | Описание |
|---|---|---|
| ChartRangeInfo.rangeType | ChartRangeType | Тип диапазона диаграммы |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local rangeInfo = charts:getChart(0):getRange(0)
print(rangeInfo.cellRange, rangeInfo.rangeColor, rangeInfo.isHidden,
rangeInfo.rangeType)
```

### 5.27 Таблица DocumentAPI.ChartRangeType

Таблица DocumentAPI.ChartRangeType описывает тип диапазона исходных данных диаграммы. Используется в поле ChartRangeInfo.rangeType.

Таблица 18 – Описание полей таблицы DocumentAPI.ChartRangeType

| Значение | Описание |
|---|---|
| DocumentAPI.ChartRangeType_Series | Серии |
| DocumentAPI.ChartRangeType_SeriesName | Имена серий |
| DocumentAPI.ChartRangeType_Categories | Категории |
| DocumentAPI.ChartRangeType_DataPoint | Разметка данных |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local rangeInfo = charts:getChart(0):getRange(0)

rangeTypes = {"Series", "SeriesName", "Categories", "DataPoint" }
print(rangeTypes[rangeInfo.rangeType + 1])
```

### 5.28 Таблица DocumentAPI.Charts

Таблица `DocumentAPI.Charts` обеспечивает доступ к списку диаграмм (см. Рисунок 37) табличного документа. Доступ к списку диаграмм осуществляется с помощью метода `Table.getCharts()`.

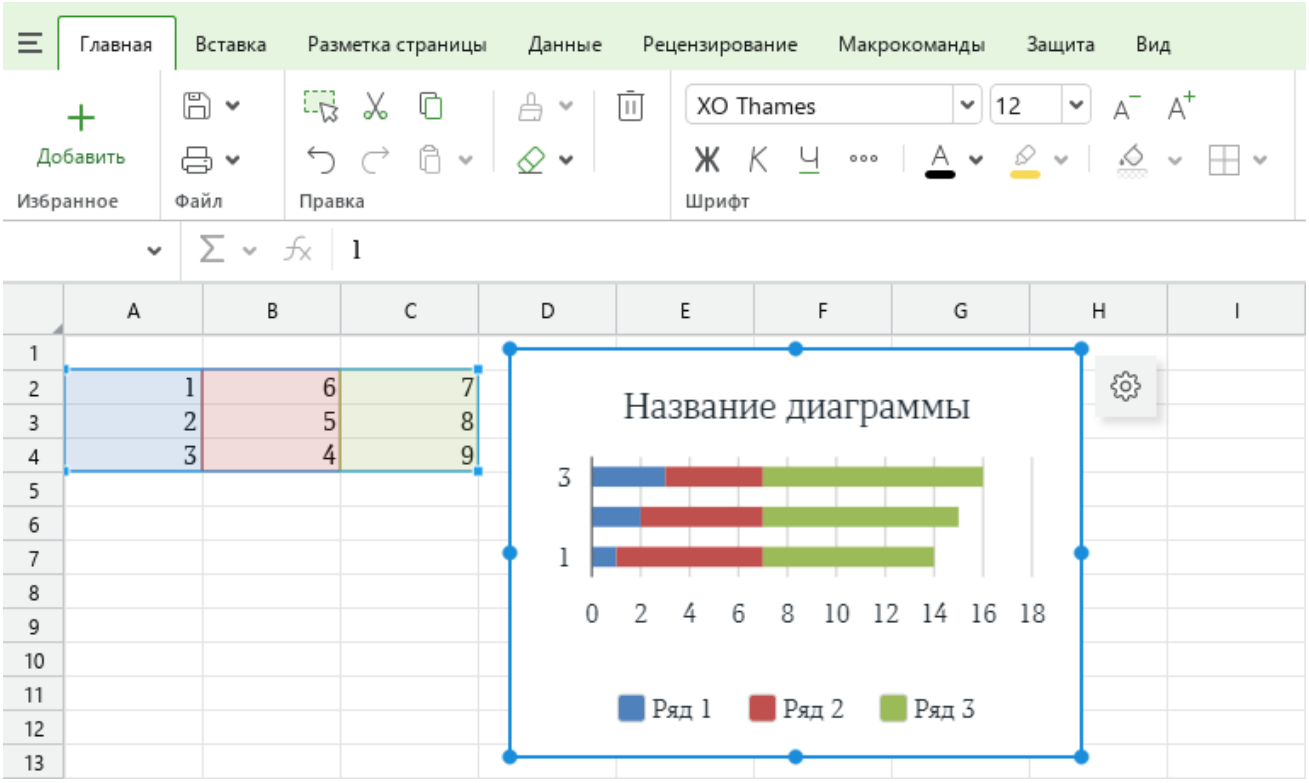



Рисунок 37 – Пример отображения диаграммы в приложении «МойОфис Таблица»

#### 5.28.1 Метод Charts:addChart

Метод добавляет новую диаграмму на лист.

**Вызов**

```
Chart addChart(dataRange, type, rect)
```

**Параметры**

– `dataRange`: диапазон ячеек с исходными данными для диаграммы, тип `CellRangePosition`.

– `type`: тип диаграммы, тип `ChartType`.

– `rect`: область расположения диаграммы, тип `RectU`.

**Возвращает**

– новая диаграмма, тип `Chart`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("B2:C18")
local charts = sheet:getCharts()
local newChart = charts:addChart(cellRange:getTableRange(),
DocumentAPI.ChartType_Doughnut, DocumentAPI.RectU(400, 200, 800, 600))
```

### 5.28.2 Метод Charts:getChart

Метод возвращает диаграмму DocumentAPI.Chart по индексу chartIndex в коллекции диаграмм.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRangeAsString())
```

### 5.28.3 Метод Charts:getChartIndexByDrawingIndex

Метод возвращает индекс диаграммы по индексу отрисовки drawingIndex.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChartIndexByDrawingIndex(0))
```

### 5.28.4 Метод Charts:getChartsCount

Метод возвращает общее количество диаграмм в табличном документе.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChartsCount())
```

### 5.28.5 Метод Charts:enumerate

Метод позволяет перечислить диаграммы листа.

**Вызов**

```
ChartsEnumerator enumerate()
```

**Возвращает**

– перечисление диаграмм в коллекции.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local charts = sheet:getCharts()
local enumerator = charts:enumerate()
for chart in enumerator do
      print(chart:getTitle());
end
```

### 5.29 Таблица DocumentAPI.ChartSeriesDirectionType

Таблица `DocumentAPI.ChartSeriesDirectionType` описывает направление серий диаграмм. Используется в методах `Chart:applySettings()` и `Chart:getDirectionType()`.

Таблица 19 – Описание полей таблицы `DocumentAPI.ChartSeriesDirectionType`

| Значение | Описание |
|---|---|
| `DocumentAPI.ChartSeriesDirectionType_Unknown` | Неопределенный тип |
| `DocumentAPI.ChartSeriesDirectionType_ByRow` | Серии направлены по строкам |
| `DocumentAPI.ChartSeriesDirectionType_ByColumn` | Серии направлены по колонкам |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getDirectionType())
```

### 5.30 Таблица DocumentAPI.ChartType

Таблица `DocumentAPI.ChartType` описывает все поддерживаемые типы диаграмм. Используется в методах `Charts:addChart()`, `Chart:getType()` и `Chart:setType()`.

Таблица 20 – Типы диаграмм

| Значение | Описание |
|---|---|
| `DocumentAPI.ChartType_Unknown` | Неопределенный тип |
| `DocumentAPI.ChartType_Bar` | Линейчатая диаграмма с группировкой |
| `DocumentAPI.ChartType_BarStacked` | Линейчатая диаграмма с накоплением |
| `DocumentAPI.ChartType_BarPercentStacked` | Линейчатая нормированная диаграмма с накоплением |
| `DocumentAPI.ChartType_Column` | Гистограмма с группировкой |
| `DocumentAPI.ChartType_ColumnStacked` | Гистограмма с накоплением |
| `DocumentAPI.ChartType_ColumnPercentStacked` | Нормированная гистограмма с накоплением |
| `DocumentAPI.ChartType_Line` | Стандартный график |
| `DocumentAPI.ChartType_LineStacked` | График с накоплением |
| `DocumentAPI.ChartType_LinePercentStacked` | Нормированный график с накоплением |
| `DocumentAPI.ChartType_LineWithMarker` | Стандартный график с маркерами |
| `DocumentAPI.ChartType_LineWithMarkerStacked` | График с накоплением и маркерами |
| `DocumentAPI.ChartType_LineWithMarkerPercentStacked` | Нормированный график с накоплением и маркерами |
| `DocumentAPI.ChartType_Area` | Стандартная диаграмма с областями |

| Значение | Описание |
|---|---|
| `DocumentAPI.ChartType_AreaStacked` | Диаграмма с областями с накоплением |
| `DocumentAPI.ChartType_AreaPercentStacked` | Нормированная диаграмма с областями с накоплением |
| `DocumentAPI.ChartType_Pie` | Круговая диаграмма |
| `DocumentAPI.ChartType_PieExploded` | Круговая диаграмма с отделенными секторами |
| `DocumentAPI.ChartType_Scatter` | Диаграмма рассеяния |
| `DocumentAPI.ChartType_Doughnut` | Кольцевая диаграмма |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getType())
```

### 5.31 Таблица DocumentAPI.CheckBoxControl

Представляет собой элемент управления "Флажок" в документе. Является наследником таблицы `ContentControl`. Методы `CheckBoxControl:getValue()` и `CheckBoxControl:setValue(bool)` позволяют получить и задать значение этого элемента управления.

**Пример**

```
local controls = document:getContentControls()
local checkBox = controls:findByTitle("check1"):toCheckBox()
checkBox:setValue(not(checkBox:getValue()))
```

### 5.32 Таблица DocumentAPI.Color

Таблица `DocumentAPI.Color` представляет либо цветовой объект RGBA, либо заданные цвета идентификатора темы. В качестве параметров конструктора используются

таблицы DocumentAPI.ColorRGBA, DocumentAPI.ThemeColorID.

**Пример**

```
local rgbaColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))
local themeColor = DocumentAPI.Color(DocumentAPI.ThemeColorID_Text1)
```

### 5.32.1 Метод Color:getRGBAColor

Метод возвращает цвет DocumentAPI.ColorRGBA.

**Пример**

```
local color = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))
local rgbaColor = color:getRGBAColor()
if rgbaColor then
  print(rgbaColor.r, rgbaColor.g, rgbaColor.b, rgbaColor.a)
end
```

### 5.32.2 Метод Color:getThemeColorID

Метод возвращает цвет идентификатора темы DocumentAPI.ThemeColorID.

**Пример**

```
local color = DocumentAPI.Color(DocumentAPI.ThemeColorID_Text1)
local themeColorID = color:getThemeColorID()
if themeColorID then
  print(themeColorID)
end
```

### 5.32.3 Метод Color:getTransforms

Метод возвращает примененные к цвету трансформации.

**Вызов**

```
ColorTransforms getTransforms()
```

**Возвращает**

– трансформации цвета, тип ColorTransforms.

– `nil`, если трансформации не заданы.

**Пример**

```
local transforms = color:getTransforms()
local enumerator = transforms:enumerate()
for transform in enumerator do
      print(transform:getType())
end
```

### 5.32.4 Метод Color:setTransforms

Метод применяет заданные трансформации к текущему цвету.

**Вызов**

```
setTransforms(transforms)
```

**Параметры**

– `transforms`: трансформации цвета, тип `ColorTransforms`.

**Пример**

```
local color = DocumentAPI.Color(DocumentAPI.ThemeColorID_FollowedHyperlink)
local colorTransforms = DocumentAPI.ColorTransforms()
colorTransforms:addTransform(DocumentAPI.ColorTransformType_RedModulation,
1.5):addTransform(DocumentAPI.ColorTransformType_Tint, 0.3)
color:setTransforms(colorTransforms)
```

### 5.32.5 Метод Color:__eq

Метод используется для определения эквивалентности двух значений цвета.

**Пример**

```
local color = DocumentAPI.Color(DocumentAPI.ThemeColorID_Text1)
if color:__eq(color) then
  print("Equals")
end
```

## 5.33 Таблица DocumentAPI.ColorRGBA

Таблица `DocumentAPI.ColorRGBA` предназначена для задания цвета текста, линии, фона и т.д. Используется четырехканальный формат, содержащий данные для красного (`r`), зеленого (`g`), голубого (`b`) цветов и альфа-канала (`a`).

**Конструкторы**

```
ColorRGBA()
ColorRGBA(number r, number g, number b, number a)
```

Описание полей таблицы `DocumentAPI.ColorRGBA` представлено в таблице 21.

Таблица 21 – Описание таблицы `DocumentAPI.ColorRGBA`

| Поле | Тип | Описание |
|---|---|---|
| `ColorRGBA.r` | `number` | Значение от 0 до 255 для установки интенсивности красного цвета. |
| `ColorRGBA.g` | `number` | Значение от 0 до 255 для установки интенсивности зеленого цвета. |
| `ColorRGBA.b` | `number` | Значение от 0 до 255 для установки интенсивности голубого цвета. |
| `ColorRGBA.a` | `number` | Значение от 0 до 255 для регулировки прозрачности.<br>Значение 255 соответствует полностью непрозрачному цвету. |

**Примеры использования**

```
local rgba = DocumentAPI.ColorRGBA()
rgba.r = 0
rgba.g = 0
rgba.b = 255
rgba.a = 200
print("r=", rgba.r, ", g=", rgba.g, ", b=", rgba.b, ", a=", rgba.a)  -- r=0, g=0, b=255, a=200
```

```
local rgba = DocumentAPI.ColorRGBA(55, 146, 179, 200)
print("r=", rgba.r, ", g=", rgba.g, ", b=", rgba.b, ", a=", rgba.a)  -- r=55, g=146, b=179, a=200
```

```
local line_prop = DocumentAPI.LineProperties()
line_prop.color = DocumentAPI.Color(rgba)
```

### 5.33.1 Метод ColorRGBA:__eq

Метод используется для определения эквивалентности двух значений цвета `DocumentAPI.ColorRGBA`.

**Пример**

```
colorRGBA = DocumentAPI.Color(DocumentAPI.ColorRGBA(55, 146, 179, 200))
if colorRGBA:__eq(colorRGBA) then
  print("Equals")
end
```

### 5.34 Таблица DocumentAPI.ColorScaleConditionalFormatOperator

Таблица `ColorScaleConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Цветовая шкала". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

**Конструктор**

```
ColorScaleConditionalFormatOperator(ConditionalFormatColorScaleEntries
entries)
```

**Пример**

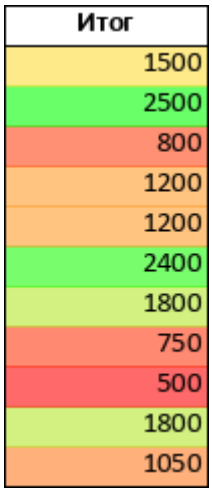



Рисунок 38 – Пример создания правила "Цветовая шкала" с тремя пороговыми значениями

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local entries = DocumentAPI.ConditionalFormatColorScaleEntries()
local value1 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Min, "0", false)
local entry1 = DocumentAPI.ConditionalFormatColorScaleEntry(value1,
DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
entries:addEntry(entry1)
local value2 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Percent, "50", false)
local entry2 = DocumentAPI.ConditionalFormatColorScaleEntry(value2,
DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 220, 56, 150)))
entries:addEntry(entry2)
local value3 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
```

```
Type_Max, "0", false)
local entry3 = DocumentAPI.ConditionalFormatColorScaleEntry(value3,
DocumentAPI.Color(DocumentAPI.ColorRGBA(0, 255, 0, 150)))
entries:addEntry(entry3)

local cellRange = sheet:getCellRange("G2:G12")
local cellRangePosition = cellRange:getTableRange()

local colorScaleOperator =
DocumentAPI.createColorScaleConditionalFormatOperator(entries)

local colorScaleRule = DocumentAPI.ConditionalFormatRule()
colorScaleRule:setRange(cellRangePosition)
colorScaleRule:setOperator(colorScaleOperator)
rules:addRule(colorScaleRule)
```

### 5.34.1 Метод ColorScaleConditionalFormatOperator:getEntries

Метод возвращает набор правил применения цветов для текущего оператора.

**Вызов**

```
ConditionalFormatColorScaleEntries getEntries()
```

**Возвращает**

– набор правил применения цветов, тип ConditionalFormatColorScaleEntries.

### 5.34.2 Метод ColorScaleConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

### 5.34.3 Метод ColorScaleConditionalFormatOperator:setEntries

Метод задает набор правил применения цветов для текущего оператора.

**Вызов**

```
setEntries(entries)
```

**Параметры**

– `entries`: набор правил применения цветов, тип `ConditionalFormatColorScaleEntries`.

### 5.35 Таблица DocumentAPI.ColorTransform

Класс `ColorTransform` представляет собой трансформацию цвета для объектов `ColorRGBA` и `Color`. Является элементом коллекции `ColorTransforms`.

**Пример**

```
local transforms = color:getTransforms()
local enumerator = transforms:enumerate()
for transform in enumerator do
       print(transform:getType())
end
```

#### 5.35.1 Метод ColorTransform:getType

Метод возвращает тип текущей трансформации цвета.

**Вызов**

```
ColorTransformType getType()
```

**Возвращает**

– тип трансформации цвета, тип `ColorTransformType`.

#### 5.35.2 Метод ColorTransform:getValue

Метод возвращает значение текущей трансформации цвета.

**Вызов**

```
number getValue()
```

**Возвращает**

– значение трансформации, тип `number`.

– `nil`, если значение не задано.

### 5.36 Таблица DocumentAPI.ColorTransforms

Класс `ColorTransforms` представляет собой коллекцию трансформаций цвета для объектов `ColorRGBA` и `Color` (см. методы `Color:setTransforms` и `Color:getTransforms`). Элементами коллекции являются объекты класса `ColorTransform`.

**Примеры**

```
local colorTransforms = DocumentAPI.ColorTransforms()
colorTransforms:addTransform(DocumentAPI.ColorTransformType_Tint,
0.7):addTransform(DocumentAPI.ColorTransformType_Complement)
local newColor = colorTransforms:apply(DocumentAPI.ColorRGBA(55, 146, 179, 200))
```

```
local color = DocumentAPI.Color(DocumentAPI.ThemeColorID_FollowedHyperlink)
local colorTransforms = DocumentAPI.ColorTransforms()
colorTransforms:addTransform(DocumentAPI.ColorTransformType_RedModulation,
1.5):addTransform(DocumentAPI.ColorTransformType_Tint, 0.3)
color:setTransforms(colorTransforms)
```

#### 5.36.1 Метод ColorTransforms:addTransform

Метод добавляет новую трансформацию цвета в коллекцию.

**Вызов**

```
ColorTransforms addTransform(transformType, transformValue)
```

**Параметры**

– `transformType`: тип трансформации, тип `ColorTransformType`.

– `transformValue`: (необязательный) значение трансформации, тип `number`. Допустимые значения для каждого типа трансформации перечислены в описании `ColorTransformType`.

**Возвращает**

– текущая коллекция трансформаций цвета, тип `ColorTransforms`.

**Пример**

```
local colorTransforms = DocumentAPI.ColorTransforms()
colorTransforms:addTransform(DocumentAPI.ColorTransformType_Tint,
0.7):addTransform(DocumentAPI.ColorTransformType_Complement)
local newColor = colorTransforms:apply(DocumentAPI.ColorRGBA(55, 146, 179, 200))
```

### 5.36.2 Метод ColorTransforms:apply

Метод применяет текущие трансформации к заданному цвету.

**Вызов**

```
ColorRGBA apply(color)
```

**Параметры**

– `color`: исходный цвет, тип ColorRGBA.

**Возвращает**

– цвет после применения трансформаций, тип ColorRGBA.

**Пример**

```
local colorTransforms = DocumentAPI.ColorTransforms()
colorTransforms:addTransform(DocumentAPI.ColorTransformType_Shade, 0.1)
colorTransforms:addTransform(DocumentAPI.ColorTransformType_Inverse)
local newColor = colorTransforms:apply(DocumentAPI.ColorRGBA(55, 146, 179, 200))
```

### 5.36.3 Метод ColorTransforms:clearTransforms

Метод очищает коллекцию трансформаций цвета.

**Вызов**

```
clearTransforms()
```

### 5.36.4 Метод ColorTransforms:enumerate

Метод позволяет перечислить элементы коллекции.

**Вызов**

```
ColorTransformsEnumerator enumerate()
```

**Возвращает**

– перечисление трансформаций цвета в коллекции.

**Пример**

```
local transforms = color:getTransforms()
local enumerator = transforms:enumerate()
for transform in enumerator do
      print(transform:getType())
end
```

### 5.37 Таблица DocumentAPI.ColorTransformType

Перечисление `ColorTransformType` содержит типы трансформаций цвета. Используется в методах `ColorTransforms:addTransform()` и `ColorTransform:getType()`.

Таблица 22 – Типы трансформаций цвета

| Значение | Описание | Значения параметра |
|---|---|---|
| `ColorTransformType_Alpha` | Задает значение Alpha-канала (прозрачности) для модели RGBA. Значение задается в процентах от 255. | 0,0-1,0 |
| `ColorTransformType_Red` | Задает значение красного канала для модели RGBA. Значение задается в процентах от 255. | 0,0-1,0 |
| `ColorTransformType_Green` | Задает значение зеленого канала для модели RGBA. Значение задается в процентах от 255. | 0,0-1,0 |
| `ColorTransformType_Blue` | Задает значение синего канала для модели RGBA. Значение задается в процентах от 255. | 0,0-1,0 |
| `ColorTransformType_AlphaModulation` | Умножает значение Alpha-канала на заданный коэффициент. Результат трансформации не может превысить 255. | >=0,0 |
| `ColorTransformType_RedModulation` | Умножает значение красного канала на заданный коэффициент. Результат трансформации не может превысить 255. | >=0,0 |
| `ColorTransformType_GreenModulation` | Умножает значение зеленого канала на заданный коэффициент. Результат трансформации не может превысить 255. | >=0,0 |
| `ColorTransformType_BlueModulation` | Умножает значение синего канала на заданный коэффициент. Результат трансформации не может превысить 255. | >=0,0 |
| `ColorTransformType_AlphaOffset` | Увеличивает или уменьшает значение Alpha-канала на заданный процент. | >=-1,0 |

| Значение | Описание | Значения параметра |
|---|---|---|
| | Результат трансформации не может быть меньше 0 и больше 255. | |
| ColorTransformType_RedOffset | Увеличивает или уменьшает значение красного канала на заданный процент. Результат трансформации не может быть меньше 0 и больше 255. | >=-1,0 |
| ColorTransformType_GreenOffset | Увеличивает или уменьшает значение зеленого канала на заданный процент. Результат трансформации не может быть меньше 0 и больше 255. | >=-1,0 |
| ColorTransformType_BlueOffset | Увеличивает или уменьшает значение синего канала на заданный процент. Результат трансформации не может быть меньше 0 и больше 255. | >=-1,0 |
| ColorTransformType_Hue | Задает положительный угол поворота на окружности цветового тона (Hue) в модели HSL. | 0,0-360,0 |
| ColorTransformType_HueModulation | Умножает значение тона (Hue) в модели HSL на заданный коэффициент. Результат трансформации не может привысить 360. | >=0,0 |
| ColorTransformType_HueOffset | Увеличивает или уменьшает угол поворота на окружности цветового тона (Hue) на заданное количество градусов. Результат трансформации не может быть меньше 0 и больше 360. | -360,0-360,0 |
| ColorTransformType_Luminance | Задает значение канала светлоты (Luminance) в модели HSL. Значение задается в процентах. | 0,0-1,0 |
| ColorTransformType_LuminanceModulation | Умножает значение канала светлоты (Luminance) в модели HSL на заданный коэффициент. Результат трансформации не может привысить 100. | >=0,0 |

| Значение | Описание | Значения параметра |
|---|---|---|
| ColorTransformType_LuminanceOffset | Увеличивает или уменьшает значение канала светлоты (Luminance) на заданный процент.<br>Результат трансформации не может быть меньше 0 и больше 100. | >=-1,0 |
| ColorTransformType_Saturation | Задает значение канала насыщенности (Saturation) в модели HSL. Значение задается в процентах. | 0,0-1,0 |
| ColorTransformType_SaturationModulation | Умножает значение канала насыщенности (Saturation) в модели HSL на заданный коэффициент. Результат трансформации не может привысить 100. | >=0,0 |
| ColorTransformType_SaturationOffset | Увеличивает или уменьшает значение канала насыщенности (Saturation) на заданный процент.<br>Результат трансформации не может быть меньше 0 и больше 100. | >=-1,0 |
| ColorTransformType_Shade | Умножает значение каждого канала (R, G, B) на заданный параметр, делая цвет темнее (0,0 - полностью черный, 1,0 - исходный цвет). | 0,0-1,0 |
| ColorTransformType_Tint | Делает цвет светлее, смешивая его с белым. Параметр задает процент осветления (0,0 - исходный цвет, 1,0 - полностью белый). | 0,0-1,0 |
| ColorTransformType_Complement | Сдвигает цветовой тон (Hue) на 180° (Меняет цвет на противоположный на цветовом круге). | - |
| ColorTransformType_Inverse | Инвертирует каждый RGB канал цвета. | - |
| ColorTransformType_Gamma | Преобразует цвет из линейного пространства в пространство sRGB. | - |
| ColorTransformType_InverseGamma | Преобразует цвет из гамма-пространства (sRGB) в линейное пространство. | - |

| Значение | Описание | Значения параметра |
|---|---|---|
| `ColorTransformType_Gray` | Преобразует цвет в оттенок серого, сохраняя яркость. | - |

### 5.38 Таблица DocumentAPI.Comment

Таблица `DocumentAPI.Comment` предоставляет доступ к следующим свойствам комментария:

- диапазон текста `DocumentAPI.Range`, который описывает комментарий;
- текст комментария;
- информация о комментарии `DocumentAPI.TrackedChangeInfo`;
- признак того, что комментарий принят;
- список ответов на комментарий `DocumentAPI.Comments`.

#### 5.38.1 Метод Comment:getInfo

Метод предоставляет доступ к информации о комментарии `DocumentAPI.TrackedChangeInfo` (автор изменения, дата и т. д).

**Пример**

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    local commentInfo = comment:getInfo()
    local name = commentInfo.author.name
    print("Автор комментария:", name)
end
```

#### 5.38.2 Метод Comment:getRange

Метод возвращает диапазон документа `DocumentAPI.Range`, которому соответствует комментарий.

**Пример**

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    print("Диапазон комментария: ", comment:getRange():extractText())
end
```

### 5.38.3 Метод Comment:getReplies

Метод предоставляет доступ к ответам на комментарии. Ответы находятся в такой же таблице `DocumentAPI.Comments`, как и сами комментарии документа.

**Пример**

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    local commentReplies = comment:getReplies()
    for reply in commentReplies:enumerate() do
        local name = reply.author.name
        print("Ответ на комментарий " .. name .. ": ", reply:getText())
    end
end
```

### 5.38.4 Метод Comment:getText

Метод возвращает текст комментария.

**Пример**

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    print("Текст комментария: ", comment:getText())
end
```

### 5.38.5 Метод Comment:isResolved

Метод возвращает значение `true`, если комментарий принят.

**Пример**

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    print("Комментарий принят:", comment:isResolved())
end
```

### 5.39 Таблица DocumentAPI.Comments

Таблица `DocumentAPI.Comments` содержит коллекцию комментариев диапазона (см. Рисунок 39).

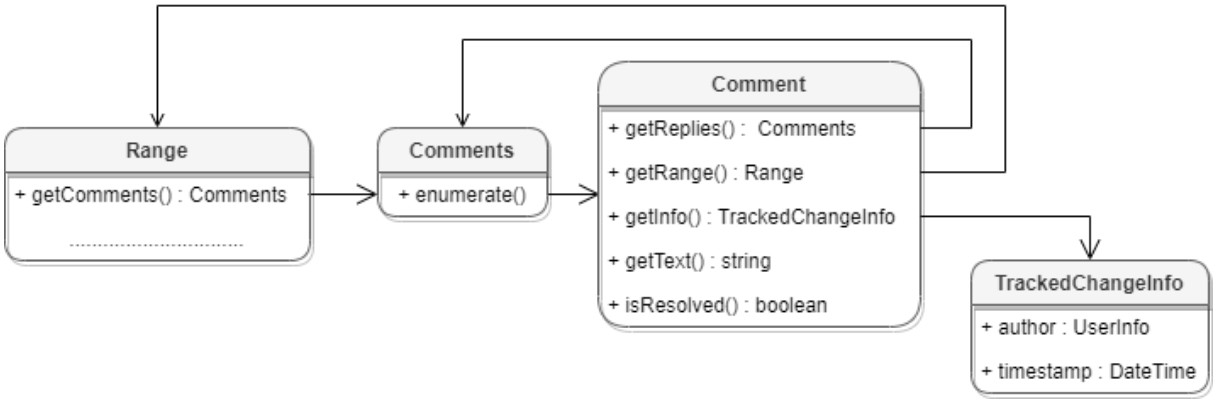



Рисунок 39 – Объектная модель таблиц для работы с комментариями

Для получения списка комментариев используется метод `Range.getComments()`.

**Пример**

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    local commentInfo = comment:getInfo()
    local name = commentInfo.author.name
    print("Комментарий " .. name .. ": ", comment:getText())
end
```

### 5.39.1 Метод Comments:enumerate

Метод возвращает коллекцию комментариев всего документа.

**Пример**

```
local comments = document:getComments()
for comment in comments:enumerate() do
  print(comment:getText())
end
```

## 5.40 Таблица DocumentAPI.ConditionalFormatAboveAverageCondition

Таблица `ConditionalFormatAboveAverageCondition` содержит условия применения форматирования для правил "Выше среднего" и "Ниже среднего". Используется в методе `AboveAverageConditionalFormatOperator:getCondition()`.

Таблица 23 – Условия применения форматирования

| Значение | Описание |
|---|---|
| ConditionalFormatAboveAverageCondition_Above | Выше среднего |
| ConditionalFormatAboveAverageCondition_EqualAbove | Выше среднего или равно |
| ConditionalFormatAboveAverageCondition_Below | Ниже среднего |
| ConditionalFormatAboveAverageCondition_EqualBelow | Ниже среднего или равно |

### 5.41 Таблица DocumentAPI.ConditionalFormatBinaryCondition

Таблица ConditionalFormatBinaryCondition содержит условия применения форматирования для правил "Между" и "Не между". Используется в методе BinaryConditionalFormatOperator:getCondition().

Таблица 24 – Условия применения форматирования

| Значение | Описание |
|---|---|
| ConditionalFormatBinaryCondition_Between | Между, включительно |
| ConditionalFormatBinaryCondition_NotBetween | Не между |

### 5.42 Таблица DocumentAPI.ConditionalFormatCellStyle

Таблица ConditionalFormatCellStyle предназначена для настройки параметров отображения ячеек, которые попадают под условие форматирования. Эти настройки применяются к правилам, созданным с помощью следующих операторов: AboveAverageConditionalFormatOperator, BinaryConditionalFormatOperator, NullaryConditionalFormatOperator, TextConditionalFormatOperator, TopBottomConditionalFormatOperator, UnaryConditionalFormatOperator и UniquenessConditionalFormatOperator. Данная таблица используется в методах ConditionalFormatRule:getStyle(), ConditionalFormatRule:setStyle(), ConditionalFormatRuleProxy:getStyle() и ConditionalFormatRuleProxy:setStyle().

Таблица 25 – Описание полей таблицы `ConditionalFormatCellStyle`

| Поле | Тип | Описание |
|---|---|---|
| `ConditionalFormatCellStyle.borders` | `Borders` | Настройки границ ячеек |
| `ConditionalFormatCellStyle.cellFormat` | `CellFormat` | Формат ячеек |
| `ConditionalFormatCellStyle.cellProperties` | `CellProperties` | Настройки форматирования содержимого ячеек |
| `ConditionalFormatCellStyle.textProperties` | `TextProperties` | Параметры текста в ячейках |

### 5.43 Таблица DocumentAPI.ConditionalFormatColorScaleEntries

Таблица `ConditionalFormatColorScaleEntries` представляет собой набор правил применения цветов для условного форматирования "Цветовая шкала". Правила в наборе должны быть расположены в порядке возрастания пороговых значений. Используется в методах `ColorScaleConditionalFormatOperator:getEntries()` и `ColorScaleConditionalFormatOperator:setEntries()`.

**Пример**

```
local entries = DocumentAPI.ConditionalFormatColorScaleEntries()
local value1 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Min, "0", false)
local entry1 = DocumentAPI.ConditionalFormatColorScaleEntry(value1,
DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
entries:addEntry(entry1)
-- ...
local colorScaleOperator =
DocumentAPI.ColorScaleConditionalFormatOperator(entries)
```

#### 5.43.1 Метод ConditionalFormatColorScaleEntries:addEntry

Метод добавляет правило применения цвета в текущий набор.

**Вызов**

```
addEntry(entry)
```

**Параметры**

– `entry`: правило применения цвета, тип `ConditionalFormatColorScaleEntry`.

### 5.43.2 Метод ConditionalFormatColorScaleEntries:getEntriesCount

Метод возвращает количество правил в текущем наборе.

**Вызов**

```
number getEntriesCount()
```

**Возвращает**

– количество правил, тип `number`.

### 5.43.3 Метод ConditionalFormatColorScaleEntries:getEntry

Метод возвращает правило по его индексу.

**Вызов**

```
ConditionalFormatColorScaleEntry getEntry(index)
```

**Параметры**

– `index`: индекс правила, индексация начинается с нуля, тип `number`.

**Возвращает**

– правило применения цвета, тип `ConditionalFormatColorScaleEntry`.

### 5.43.4 Метод ConditionalFormatColorScaleEntries:setEntry

Метод заменяет правило под заданным индексом.

**Вызов**

```
setEntry(index, entry)
```

**Параметры**

– `index`: индекс правила для замены, индексация начинается с нуля, тип `number`.

– `entry`: новое правило применения цвета, тип `ConditionalFormatColorScaleEntry`.

## 5.44 Таблица DocumentAPI.ConditionalFormatColorScaleEntry

Таблица `ConditionalFormatColorScaleEntry` представляет собой правило применения цвета для условного форматирования "Цветовая шкала". Используется в методах `ConditionalFormatColorScaleEntries:addEntry()`,

ConditionalFormatColorScaleEntries:getEntry() и ConditionalFormatColorScaleEntries:setEntry().

**Конструктор**

```
ConditionalFormatColorScaleEntry(ConditionalFormatValueObject valueObject, Color color)
```

**Пример**

```
local entries = DocumentAPI.ConditionalFormatColorScaleEntries()
local value1 = DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObjectType_Min, "0", false)
local entry1 = DocumentAPI.ConditionalFormatColorScaleEntry(value1, DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
entries:addEntry(entry1)
```

### 5.44.1 Метод ConditionalFormatColorScaleEntry:getColor

Метод возвращает цвет, который используется в текущем правиле.

**Вызов**

```
Color getColor()
```

**Возвращает**

– используемый цвет, тип Color.

### 5.44.2 Метод ConditionalFormatColorScaleEntry:getValueObject

Метод возвращает пороговое значение для текущего правила.

**Вызов**

```
ConditionalFormatValueObject getValueObject()
```

**Возвращает**

– пороговое значение для текущего правила, тип ConditionalFormatValueObject.

### 5.44.3 Метод ConditionalFormatColorScaleEntry:setColor

Метод позволяет задать цвет для текущего правила.

**Вызов**

```
setColor(color)
```

**Параметры**

– `color`: цвет для правила, тип `Color`.

### 5.44.4 Метод ConditionalFormatColorScaleEntry:setValueObject

Метод позволяет задать пороговое значение для текущего правила.

**Вызов**

```
setValueObject(valueObject)
```

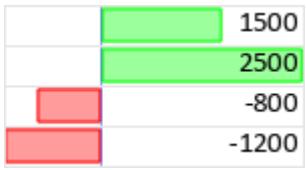
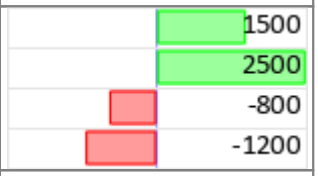
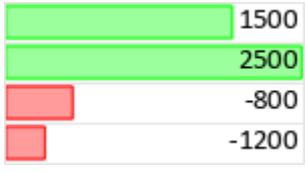
**Параметры**

– `valueObject`: пороговое значение для текущего правила, тип `ConditionalFormatValueObject`.

## 5.45 Таблица DocumentAPI.ConditionalFormatDataBarAxisPosition

Таблица `ConditionalFormatDataBarAxisPosition` содержит варианты расположения оси между положительными и отрицательными значениями гистограммы. Используется в поле `ConditionalFormatDataBarParams.axisPosition`.

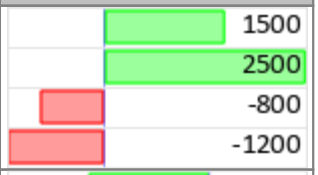
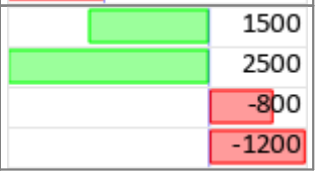
Таблица 26 – Варианты расположения оси

| Значение | Описание | Изображение |
|---|---|---|
| `ConditionalFormatDataBarAxisPosition_Auto` | Ось расположена в зависимости от соотношения максимального положительного и отрицательного значений |  |
| `ConditionalFormatDataBarAxisPosition_Middle` | Ось расположена посередине ячейки |  |
| `ConditionalFormatDataBarAxisPosition_None` | Ось отсутствует, положительные и отрицательные значения направлены в одну сторону |  |

### 5.46 Таблица DocumentAPI.ConditionalFormatDataBarDirection

Таблица `ConditionalFormatDataBarDirection` содержит варианты направления гистограммы в ячейке. Используется в поле `ConditionalFormatDataBarParams.direction`.

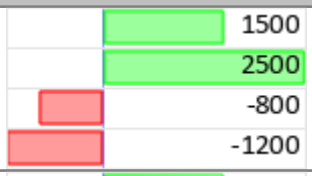
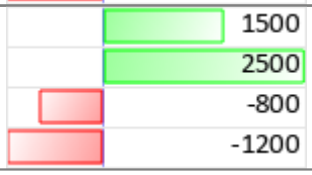
Таблица 27 – Варианты направления гистограммы

| Значение | Описание | Изображение |
|---|---|---|
| `ConditionalFormatDataBarDirection_LeftToRight` | Гистограмма для положительных значений рисуется слева направо |  |
| `ConditionalFormatDataBarDirection_RightToLeft` | Гистограмма для положительных значений рисуется справа налево |  |

### 5.47 Таблица DocumentAPI.ConditionalFormatDataBarFillType

Таблица `ConditionalFormatDataBarDirection` содержит варианты заливки гистограммы в ячейке. Используется в поле `ConditionalFormatDataBarParams.fillType`.

Таблица 28 – Варианты заливки гистограммы

| Значение | Описание | Изображение |
|---|---|---|
| `ConditionalFormatDataBarFillType_Solid` | Сплошная заливка |  |
| `ConditionalFormatDataBarFillType_Gradient` | Градиент |  |

### 5.48 Таблица DocumentAPI.ConditionalFormatDataBarParams

Таблица `ConditionalFormatDataBarParams` предназначена для настройки параметров гистограммы условного форматирования. Используется при создании экземпляра класса `DataBarConditionalFormatOperator`.

**Конструктор**

```
     ConditionalFormatDataBarParams(ConditionalFormatValueObject lowerThreshold,
ConditionalFormatValueObject upperThreshold)
```

Таблица 29 – Описание полей таблицы `ConditionalFormatDataBarParams`

| Поле | Тип | Описание |
|---|---|---|
| `ConditionalFormatDataBarParams.lowerThreshold` | `ConditionalFormatValueObject` | Нижнее пороговое значение |
| `ConditionalFormatDataBarParams.upperThreshold` | `ConditionalFormatValueObject` | Верхнее пороговое значение |
| `ConditionalFormatDataBarParams.barFill` | `Color` | Цвет заливки гистограммы для положительных значений |
| `ConditionalFormatDataBarParams.negativeBarFill` | `Color` | Цвет заливки гистограммы для отрицательных значений |
| `ConditionalFormatDataBarParams.axisColor` | `Color` | Цвет оси между положительными и отрицательными значениями |
| `ConditionalFormatDataBarParams.borderColor` | `Color` | Цвет границ гистограммы для положительных значений |
| `ConditionalFormatDataBarParams.negativeBorderColor` | `Color` | Цвет границ гистограммы для отрицательных значений |
| `ConditionalFormatDataBarParams.minLength` | `number` | Минимальная длина гистограммы, в процентах от ширины ячейки |
| `ConditionalFormatDataBarParams.maxLength` | `number` | Максимальная длина гистограммы, в процентах от ширины ячейки |
| `ConditionalFormatDataBarParams.axisPosition` | `ConditionalFormatDataBarAxisPosition` | Расположение оси между положительными и отрицательными значениями |

| Поле | Тип | Описание |
|---|---|---|
| ConditionalFormatDataBarParams.direction | ConditionalFormatDataBarDirection | Направление гистограммы |
| ConditionalFormatDataBarParams.fillType | ConditionalFormatDataBarFillType | Тип заливки гистограммы |
| ConditionalFormatDataBarParams.isValueVisible | bool | Показывать значение в ячейке с гистограммой |

### 5.49 Таблица DocumentAPI.ConditionalFormatDocumentRules

Таблица ConditionalFormatDocumentRules представляет собой коллекцию правил условного форматирования для табличного документа. Элементы коллекции представлены объектами ConditionalFormatRuleProxy. Используется в методе Document:getConditionalFormatRules().

#### 5.49.1 Метод ConditionalFormatDocumentRules:enumerate

Метод позволяет перечислить элементы коллекции.

**Вызов**

```
ConditionalFormatRuleProxysEnumerator enumerate()
```

**Возвращает**

– Перечисление правил условного форматирования в документе.

**Пример**

```
local rules = document:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_AboveAverage
then
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule:getOperator())
    end
end
```

#### 5.49.2 Метод ConditionalFormatDocumentRules:removeAllRules

Метод удаляет все правила условного форматирования из коллекции.

**Вызов**

```
    removeAllRules()
```

**Пример**

```
local rules = document:getConditionalFormatRules()
rules:removeAllRules()
```

### 5.50 Таблица DocumentAPI.ConditionalFormatIconSet

Таблица `ConditionalFormatIconSet` содержит наборы значков, которые используются в условном форматировании. Используется в методах `ConditionalFormatIconSetEntry:getIconSet()` и `ConditionalFormatIconSetEntry:setIconSet()`.

Таблица 30 – Наборы значков условного форматирования

| Значение | Набор значков |
|---|---|
| `ConditionalFormatIconSet_ThreeArrows` |  |
| `ConditionalFormatIconSet_ThreeArrowsGray` |  |
| `ConditionalFormatIconSet_ThreeFlags` |  |
| `ConditionalFormatIconSet_ThreeTrafficLights1` |  |
| `ConditionalFormatIconSet_ThreeTrafficLights2` |  |
| `ConditionalFormatIconSet_ThreeSigns` |  |
| `ConditionalFormatIconSet_ThreeSymbols` |  |
| `ConditionalFormatIconSet_ThreeSymbols2` |  |
| `ConditionalFormatIconSet_ThreeStars` |  |
| `ConditionalFormatIconSet_ThreeTriangles` |  |
| `ConditionalFormatIconSet_ThreeSmilies` | Не поддерживается |
| `ConditionalFormatIconSet_ThreeColorSmilies` | Не поддерживается |
| `ConditionalFormatIconSet_FourArrows` |  |
| `ConditionalFormatIconSet_FourArrowsGray` |  |
| `ConditionalFormatIconSet_FourRedToBlack` |  |
| `ConditionalFormatIconSet_FourRating` |  |
| `ConditionalFormatIconSet_FourTrafficLights` |  |

| Значение | Набор значков |
|---|---|
| ConditionalFormatIconSet_FiveArrows |  |
| ConditionalFormatIconSet_FiveArrowsGray |  |
| ConditionalFormatIconSet_FiveRating |  |
| ConditionalFormatIconSet_FiveQuarters |  |
| ConditionalFormatIconSet_FiveBoxes |  |
| ConditionalFormatIconSet_NoIcons | Пустой набор |

**Пример**

```
local entries = DocumentAPI.ConditionalFormatIconSetEntries()
local value1 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Percent, "0", false)
local entry1 = DocumentAPI.ConditionalFormatIconSetEntry(value1,
DocumentAPI.ConditionalFormatIconSet_FiveBoxes, 0)
entries:addEntry(entry1)
```

### 5.51 Таблица DocumentAPI.ConditionalFormatIconSetEntries

Таблица `ConditionalFormatIconSetEntries` представляет собой набор правил отображения значков для условного форматирования. Правила в наборе должны быть расположены в порядке возрастания пороговых значений. Используется в методах `IconSetConditionalFormatOperator:getEntries()` и `IconSetConditionalFormatOperator:setEntries()`.

**Пример**

```
local entries = DocumentAPI.ConditionalFormatIconSetEntries()
local value1 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Percent, "0", false)
local entry1 = DocumentAPI.ConditionalFormatIconSetEntry(value1,
DocumentAPI.ConditionalFormatIconSet_FiveBoxes, 0)
entries:addEntry(entry1)
-- ...
local iconSetOperator = DocumentAPI.IconSetConditionalFormatOperator(entries,
true)
```

### 5.51.1 Метод ConditionalFormatIconSetEntries:addEntry

Метод добавляет правило отображения значка в текущий набор.

**Вызов**

```
addEntry(entry)
```

**Параметры**

– `entry`: правило отображения значка, тип `ConditionalFormatIconSetEntry`.

### 5.51.2 Метод ConditionalFormatIconSetEntries:getEntriesCount

Метод возвращает количество правил в текущем наборе.

**Вызов**

```
number getEntriesCount()
```

**Возвращает**

– количество правил, тип `number`.

### 5.51.3 Метод ConditionalFormatIconSetEntries:getEntry

Метод возвращает правило по его индексу.

**Вызов**

```
ConditionalFormatIconSetEntry getEntry(index)
```

**Параметры**

– `index`: индекс правила, индексация начинается с нуля, тип `number`.

**Возвращает**

– правило отображения значка, тип `ConditionalFormatIconSetEntry`.

### 5.51.4 Метод ConditionalFormatIconSetEntries:setEntry

Метод заменяет правило под заданным индексом.

**Вызов**

```
setEntry(index, entry)
```

**Параметры**

– `index`: индекс правила для замены, индексация начинается с нуля, тип `number`.

– `entry`: новое правило отображения значка, тип `ConditionalFormatIconSetEntry`.

### 5.52 Таблица DocumentAPI.ConditionalFormatIconSetEntry

Таблица `ConditionalFormatIconSetEntry` представляет собой правило отображения значка для условного форматирования. Используется в методах `ConditionalFormatIconSetEntries:addEntry()`, `ConditionalFormatIconSetEntries:getEntry()` и `ConditionalFormatIconSetEntries:setEntry()`.

#### Конструкторы

```
ConditionalFormatIconSetEntry()
```

```
ConditionalFormatIconSetEntry(ConditionalFormatValueObject valueObject, ConditionalFormatIconSet iconSet, number iconIndex)
```

#### Пример

```
local entries = DocumentAPI.ConditionalFormatIconSetEntries()
local value1 = DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObjectType_Percent, "0", false)
local entry1 = DocumentAPI.ConditionalFormatIconSetEntry(value1, DocumentAPI.ConditionalFormatIconSet_FiveBoxes, 0)
entries:addEntry(entry1)
```

#### 5.52.1 Метод ConditionalFormatIconSetEntry:getIconIndex

Метод возвращает индекс используемого значка из текущего набора.

#### Вызов

```
number getIconIndex()
```

#### Возвращает

– индекс значка в наборе, индексация начинается с нуля, тип `number`.

#### 5.52.2 Метод ConditionalFormatIconSetEntry:getIconSet

Метод возвращает набор, значок из которого используется в текущем правиле.

#### Вызов

```
ConditionalFormatIconSet getIconSet()
```

#### Возвращает

– набор значков, тип ConditionalFormatIconSet.

### 5.52.3 Метод ConditionalFormatIconSetEntry:getValueObject

Метод возвращает пороговое значение для текущего правила.

**Вызов**

```
ConditionalFormatValueObject getValueObject()
```

**Возвращает**

– пороговое значение для текущего правила, тип ConditionalFormatValueObject.

### 5.52.4 Метод ConditionalFormatIconSetEntry:setIconIndex

Метод позволяет задать значок по его индексу в наборе.

**Вызов**

```
setIconIndex(iconIndex)
```

**Параметры**

– `iconIndex`: индекс значка в наборе, индексация начинается с нуля, тип `number`.

### 5.52.5 Метод ConditionalFormatIconSetEntry:setIconSet

Метод позволяет задать набор значков для использования в текущем правиле.

**Вызов**

```
setIconSet(iconSet)
```

**Параметры**

– `iconSet`: набор значков, тип ConditionalFormatIconSet.

### 5.52.6 Метод ConditionalFormatIconSetEntry:setValueObject

Метод позволяет задать пороговое значение для текущего правила.

**Вызов**

```
setValueObject(valueObject)
```

**Параметры**

– `valueObject`: пороговое значение для текущего правила, тип ConditionalFormatValueObject.

### 5.53 Таблица DocumentAPI.ConditionalFormatNullaryCondition

Таблица `ConditionalFormatNullaryCondition` содержит условия применения форматирования для правил без параметров. Используется в методе `NullaryConditionalFormatOperator:getCondition()`.

Таблица 31 – Условия применения форматирования

| Значение | Описание |
|---|---|
| `ConditionalFormatNullaryCondition_IsBlank` | Пустая ячейка |
| `ConditionalFormatNullaryCondition_IsNotBlank` | Непустая ячейка |
| `ConditionalFormatNullaryCondition_IsError` | Ячейка с ошибкой формулы |
| `ConditionalFormatNullaryCondition_IsNotError` | Ячейка без ошибки формулы |
| `ConditionalFormatNullaryCondition_Yesterday` | Вчера |
| `ConditionalFormatNullaryCondition_Today` | Сегодня |
| `ConditionalFormatNullaryCondition_Tomorrow` | Завтра |
| `ConditionalFormatNullaryCondition_InLast7Days` | Последние 7 дней |
| `ConditionalFormatNullaryCondition_LastWeek` | Прошлая неделя |
| `ConditionalFormatNullaryCondition_ThisWeek` | Эта неделя |
| `ConditionalFormatNullaryCondition_NextWeek` | Следующая неделя |
| `ConditionalFormatNullaryCondition_LastMonth` | Прошлый месяц |
| `ConditionalFormatNullaryCondition_ThisMonth` | Этот месяц |
| `ConditionalFormatNullaryCondition_NextMonth` | Следующий месяц |

### 5.54 Таблица DocumentAPI.ConditionalFormatOperator

Таблица `ConditionalFormatOperator` представляет собой базовую таблицу для операторов условного форматирования. Используется в методах `ConditionalFormatRule:getOperator()`, `ConditionalFormatRule:setOperator()`, `ConditionalFormatRuleProxy:getOperator()` и `ConditionalFormatRuleProxy:setOperator()`.

**Наследники**

- AboveAverageConditionalFormatOperator
- BinaryConditionalFormatOperator
- ColorScaleConditionalFormatOperator
- DataBarConditionalFormatOperator
- IconSetConditionalFormatOperator
- NullaryConditionalFormatOperator
- TextConditionalFormatOperator
- TopBottomConditionalFormatOperator
- UnaryConditionalFormatOperator
- UniquenessConditionalFormatOperator

### 5.54.1 Метод ConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

## 5.55 Таблица DocumentAPI.ConditionalFormatOperatorType

Таблица ConditionalFormatOperatorType содержит типы операторов условного форматирования. Используется в методах ConditionalFormatOperator:getType() и ConditionalFormatRuleProxy:getType().

Таблица 32 – Типы операторов условного форматирования

| Значение | Описание |
|---|---|
| ConditionalFormatOperatorType_Nullary | Оператор без аргументов |
| ConditionalFormatOperatorType_Unary | Больше, меньше, равно, не равно |
| ConditionalFormatOperatorType_Binary | Между и не между |
| ConditionalFormatOperatorType_Text | "Текст" |
| ConditionalFormatOperatorType_TopBottom | Наибольшие и наименьшие значения |

| Значение | Описание |
|---|---|
| ConditionalFormatOperatorType_AboveAverage | Выше и ниже среднего |
| ConditionalFormatOperatorType_Uniqueness | Уникальные и повторяющиеся значения |
| ConditionalFormatOperatorType_IconSet | "Значки" |
| ConditionalFormatOperatorType_ColorScale | "Цветовая шкала" |
| ConditionalFormatOperatorType_DataBar | "Гистограмма" |

### 5.56 Таблица DocumentAPI.ConditionalFormatRule

Таблица `ConditionalFormatRule` представляет собой правило условного форматирования. Используется в методе `ConditionalFormatTableRules:addRule()`.

**Конструкторы**

```
ConditionalFormatRule()
```

```
ConditionalFormatRule(ConditionalFormatOperator conditionalFormatOperator, ConditionalFormatCellStyle cellStyle, CellRangePosition cellRangePosition, bool stopCalculations)
```

```
ConditionalFormatRule(ConditionalFormatOperator conditionalFormatOperator, ConditionalFormatCellStyle cellStyle, CellRangePositions cellRangePositions, bool stopCalculations)
```

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local cellRange = sheet:getCellRange("F2:F12")
local cellRangePosition = cellRange:getTableRange()

local aboveStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill = DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(0, 255, 0, 150)))
aboveStyle.cellProperties = cellProperties

local aboveOperator = DocumentAPI.createAboveAverageConditionalFormatOperator(DocumentAPI.ConditionalFormatAboveAverageCondition_Above)

local aboveRule = DocumentAPI.ConditionalFormatRule(aboveOperator, aboveStyle,
```

```
cellRangePosition, false)
rules:addRule(aboveRule)
```

### 5.56.1 Метод ConditionalFormatRule:getOperator

Метод возвращает оператор условного форматирования.

**Вызов**

```
ConditionalFormatOperator getOperator()
```

**Возвращает**

– оператор условного форматирования, тип ConditionalFormatOperator.

### 5.56.2 Метод ConditionalFormatRule:getRanges

Метод возвращает диапазоны ячеек таблицы, к которым применяется текущее правило.

**Вызов**

```
CellRangePositions getRanges()
```

**Возвращает**

– диапазоны ячеек таблицы, тип `CellRangePositions`.

### 5.56.3 Метод ConditionalFormatRule:getStopCalculations

Метод позволяет определить, будут ли применяться другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

**Вызов**

```
bool getStopCalculations()
```

**Возвращает**

– `true`, если другие правила не применяются к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

### 5.56.4 Метод ConditionalFormatRule:getStyle

Метод возвращает настройки параметров отображения ячеек, которые попадают под условие форматирования.

**Вызов**

```
ConditionalFormatCellStyle getStyle()
```

**Возвращает**

– настройки параметров отображения ячеек, тип ConditionalFormatCellStyle.

### 5.56.5 Метод ConditionalFormatRule:getUUID

Метод возвращает уникальный идентификатор для текущего правила.

**Вызов**

```
string getUUID()
```

**Возвращает**

– уникальный идентификатор правила, тип string.

### 5.56.6 Метод ConditionalFormatRule:setOperator

Метод позволяет задать оператор условного форматирования.

**Вызов**

```
setOperator(conditionalFormatOperator)
```

**Параметры**

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local entries = DocumentAPI.ConditionalFormatIconSetEntries()
-- ...
local iconSetOperator =
DocumentAPI.createIconSetConditionalFormatOperator(entries, true)

local cellRange = sheet:getCellRange("G2:G12")
local cellRangePosition = cellRange:getTableRange()

local iconSetRule = DocumentAPI.ConditionalFormatRule()
iconSetRule:setOperator(iconSetOperator)
iconSetRule:setRange(cellRangePosition)
rules:addRule(iconSetRule)
```

### 5.56.7 Метод ConditionalFormatRule:setRange

Метод позволяет задать диапазон ячеек таблицы, к которому будет применено текущее правило.

**Вызов**

```
setRange(cellRangePosition)
```

**Параметры**

– `cellRangePosition`: диапазон ячеек таблицы, тип `CellRangePosition`.

### 5.56.8 Метод ConditionalFormatRule:setRanges

Метод позволяет задать диапазоны ячеек таблицы, к которым будет применено текущее правило.

**Вызов**

```
setRanges(cellRangePositions)
```

**Параметры**

– `cellRangePositions`: диапазоны ячеек таблицы, тип `CellRangePositions`.

### 5.56.9 Метод ConditionalFormatRule:setStopCalculations

Метод позволяет задать, применять ли другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

**Вызов**

```
setStopCalculations(stopCalculations)
```

**Параметры**

– `stopCalculations`: `true`, чтобы не применять другие правила к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

### 5.56.10 Метод ConditionalFormatRule:setStyle

Метод позволяет задать настройки параметров отображения ячеек, которые попадают под условие форматирования.

**Вызов**

```
setStyle(style)
```

**Параметры**

– `style`: настройки параметров отображения ячеек, тип `ConditionalFormatCellStyle`.

### 5.56.11 Метод ConditionalFormatRule:setUUID

Метод позволяет задать уникальный идентификатор для текущего правила.

**Вызов**

```
setUUID(uuid)
```

**Параметры**

– `uuid`: уникальный идентификатор правила, тип `string`.

## 5.57 Таблица DocumentAPI.ConditionalFormatRuleProxy

Таблица `ConditionalFormatRuleProxy` представляет собой правило условного форматирования, как элемент коллекции `ConditionalFormatDocumentRules` или `ConditionalFormatTableRules`. Используется в методе `ConditionalFormatTableRules:getRule()`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local firstRule = rules:getRule(0)
firstRule:remove()
```

### 5.57.1 Метод ConditionalFormatRuleProxy:getData

Метод возвращает объект `ConditionalFormatRule` для текущего правила.

**Вызов**

```
ConditionalFormatRule getData()
```

**Возвращает**

– правило условного форматирования, тип `ConditionalFormatRule`.

### 5.57.2 Метод ConditionalFormatRuleProxy:getIndex

Метод возвращает индекс правила в коллекции.

**Вызов**

```
number getIndex()
```

**Возвращает**

– индекс правила в коллекции, индексация начинается с нуля, тип `number`.

### 5.57.3 Метод ConditionalFormatRuleProxy:getOperator

Метод возвращает оператор условного форматирования.

**Вызов**

```
ConditionalFormatOperator getOperator()
```

**Возвращает**

– оператор условного форматирования, тип `ConditionalFormatOperator`.

### 5.57.4 Метод ConditionalFormatRuleProxy:getRanges

Метод возвращает диапазоны ячеек таблицы, к которым применяется текущее правило.

**Вызов**

```
CellRangePositions getRanges()
```

**Возвращает**

– диапазоны ячеек таблицы, тип `CellRangePositions`.

### 5.57.5 Метод ConditionalFormatRuleProxy:getStopCalculations

Метод позволяет определить, будут ли применяться другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

**Вызов**

```
bool getStopCalculations()
```

**Возвращает**

– `true`, если другие правила не применяются к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

### 5.57.6 Метод ConditionalFormatRuleProxy:getStyle

Метод возвращает настройки параметров отображения ячеек, которые попадают под условие форматирования.

**Вызов**

```
ConditionalFormatCellStyle getStyle()
```

**Возвращает**

– настройки параметров отображения ячеек, тип `ConditionalFormatCellStyle`.

### 5.57.7 Метод ConditionalFormatRuleProxy:getTableName

Метод возвращает название листа, который содержит текущее правило.

**Вызов**

```
string getTableName()
```

**Возвращает**

– название листа, тип `string`.

### 5.57.8 Метод ConditionalFormatRuleProxy:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип `ConditionalFormatOperatorType`.

### 5.57.9 Метод ConditionalFormatRuleProxy:remove

Метод удаляет текущее правило из коллекции.

**Вызов**

```
remove()
```

### 5.57.10 Метод ConditionalFormatRuleProxy:setOperator

Метод позволяет задать оператор условного форматирования.

**Вызов**

```
setOperator(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

### 5.57.11 Метод ConditionalFormatRuleProxy:setRange

Метод позволяет задать диапазон ячеек таблицы, к которому будет применено текущее правило.

**Вызов**

```
setRange(cellRangePosition)
```

**Параметры**

– `cellRangePosition`: диапазон ячеек таблицы, тип `CellRangePosition`.

### 5.57.12 Метод ConditionalFormatRuleProxy:setRanges

Метод позволяет задать диапазоны ячеек таблицы, к которым будет применено текущее правило.

**Вызов**

```
setRanges(cellRangePositions)
```

**Параметры**

– `cellRangePositions`: диапазоны ячеек таблицы, тип `CellRangePositions`.

### 5.57.13 Метод ConditionalFormatRuleProxy:setStopCalculations

Метод позволяет задать, применять ли другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

**Вызов**

```
setStopCalculations(stopCalculations)
```

**Параметры**

– `stopCalculations`: `true`, чтобы не применять другие правила к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

### 5.57.14 Метод ConditionalFormatRuleProxy:setStyle

Метод позволяет задать настройки параметров отображения ячеек, которые попадают под условие форматирования.

**Вызов**

```
setStyle(style)
```

**Параметры**

– `style`: настройки параметров отображения ячеек, тип `ConditionalFormatCellStyle`.

## 5.58 Таблица DocumentAPI.ConditionalFormatTableRules

Таблица `ConditionalFormatTableRules` представляет собой коллекцию правил условного форматирования для листа табличного документа. Элементы коллекции представлены объектами `ConditionalFormatRuleProxy`. Используется в методе `Table:getConditionalFormatRules()`.

### 5.58.1 Метод ConditionalFormatTableRules:addRule

Метод добавляет заданное правило в коллекцию.

**Вызов**

```
addRule(rule)
```

**Параметры**

– `rule`: правило условного форматирования, тип `ConditionalFormatRule`.

**Пример**

```
local aboveOperator =
DocumentAPI.createAboveAverageConditionalFormatOperator(DocumentAPI.ConditionalFo
rmatAboveAverageCondition_Above)
local aboveRule = DocumentAPI.ConditionalFormatRule(aboveOperator, aboveStyle,
cellRangePosition, false)
rules:addRule(aboveRule)
```

### 5.58.2 Метод ConditionalFormatTableRules:enumerate

Метод позволяет перечислить элементы коллекции.

**Вызов**

```
ConditionalFormatRuleProxysEnumerator enumerate()
```

**Возвращает**

– Перечисление правил условного форматирования на листе.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_AboveAverage
then
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule:getOperator())
    end
end
```

### 5.58.3 Метод ConditionalFormatTableRules:getRule

Метод возвращает правило по его индексу.

**Вызов**

```
ConditionalFormatRuleProxy getRule(index)
```

**Параметры**

– `index`: индекс правила, индексация начинается с нуля, тип `number`.

**Возвращает**

– Правило условного форматирования, тип `ConditionalFormatRuleProxy`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local rule = rules:getRule(rules:getRuleCount() - 1)
```

### 5.58.4 Метод ConditionalFormatTableRules:getRuleCount

Метод возвращает количество правил в коллекции.

**Вызов**

```
number getRuleCount()
```

**Возвращает**

– количество правил в коллекции, тип `number`.

### 5.58.5 Метод ConditionalFormatTableRules:removeAllRules

Метод удаляет все правила условного форматирования из коллекции.

**Вызов**

```
removeAllRules()
```

### 5.58.6 Метод ConditionalFormatTableRules:removeRulesFromRange

Метод удаляет правила условного форматирования, которые попадают в заданный диапазон ячеек.

**Вызов**

```
removeRulesFromRange(cellRangePosition)
```

**Параметры**

– `cellRangePosition`: диапазон ячеек таблицы, тип `CellRangePosition`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = sheet:getCellRange("A2:F12")
local cellRangePosition = cellRange:getTableRange()
local rules = sheet:getConditionalFormatRules()
rules:removeRulesFromRange(cellRangePosition)
```

## 5.59 Таблица DocumentAPI.ConditionalFormatTextCondition

Таблица `ConditionalFormatTextCondition` содержит условия применения форматирования для правила "Текст". Используется в методе `TextConditionalFormatOperator:getCondition()`.

Таблица 33 – Условия применения форматирования

| Значение | Описание |
|---|---|
| `ConditionalFormatTextCondition_ContainsText` | Содержит |
| `ConditionalFormatTextCondition_DoesNotContainText` | Не содержит |
| `ConditionalFormatTextCondition_BeginsWith` | Начинается с |
| `ConditionalFormatTextCondition_EndsWith` | Заканчивается на |

### 5.60 Таблица DocumentAPI.ConditionalFormatTopBottomCondition

Таблица `ConditionalFormatTopBottomCondition` содержит условия применения форматирования для правила "Наибольшие и наименьшие значения". Используется в методе `TopBottomConditionalFormatOperator:getCondition()`.

Таблица 34 – Условия применения форматирования

| Значение | Описание |
|---|---|
| `ConditionalFormatTopBottomCondition_Top` | Наибольшие значения |
| `ConditionalFormatTopBottomCondition_Bottom` | Наименьшие значения |

### 5.61 Таблица DocumentAPI.ConditionalFormatUnaryCondition

Таблица `ConditionalFormatUnaryCondition` содержит условия применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Используется в методе `UnaryConditionalFormatOperator:getCondition()`.

Таблица 35 – Условия применения форматирования

| Значение | Описание |
|---|---|
| `ConditionalFormatUnaryCondition_Equal` | Равно |
| `ConditionalFormatUnaryCondition_NotEqual` | Не равно |
| `ConditionalFormatUnaryCondition_Greater` | Больше |
| `ConditionalFormatUnaryCondition_GreaterOrEqual` | Больше или равно |
| `ConditionalFormatUnaryCondition_Less` | Меньше |
| `ConditionalFormatUnaryCondition_LessOrEqual` | Меньше или равно |
| `ConditionalFormatUnaryCondition_Expression` | Позволяет использовать формулу в качестве аргумента |

**Пример**

| Итог | Статус |
|---|---|
| 1500 | Доставлено |
| 2500 | |
| 800 | Доставлено |
| 1200 | Доставлено |
| 1200 | Доставлено |
| 2400 | |
| 1800 | Доставлено |
| 750 | |
| 500 | |
| 1800 | Доставлено |
| 1050 | |

Рисунок 40 – Пример создания правила с формулой

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local unaryStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
unaryStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("G2:G12")
local cellRangePosition = cellRange:getTableRange()

local unaryOperator =
DocumentAPI.createUnaryConditionalFormatOperator(DocumentAPI.ConditionalFormatUna
ryCondition_Expression, "=ISBLANK(H2)")

local unaryRule = DocumentAPI.ConditionalFormatRule(unaryOperator, unaryStyle,
cellRangePosition, false)
rules:addRule(unaryRule)
```

### 5.62 Таблица DocumentAPI.ConditionalFormatUniquenessCondition

Таблица `ConditionalFormatUniquenessCondition` содержит условия применения форматирования для правила "Уникальные и повторяющиеся значения". Используется в методе `UniquenessConditionalFormatOperator:getCondition()`.

Таблица 36 – Условия применения форматирования

| Значение | Описание |
|---|---|
| `ConditionalFormatUniquenessCondition_Duplicate` | Повторяющиеся значения |
| `ConditionalFormatUniquenessCondition_Unique` | Уникальные значения |

### 5.63 Таблица DocumentAPI.ConditionalFormatValueObject

Таблица `ConditionalFormatValueObject` представляет собой пороговое значение для применения подправил условного форматирования. Используется в правилах, созданных с помощью следующих операторов: `ColorScaleConditionalFormatOperator`, `DataBarConditionalFormatOperator` и `IconSetConditionalFormatOperator`.

**Конструкторы**

```
ConditionalFormatValueObject()
```

```
ConditionalFormatValueObject(ConditionalFormatValueObjectType type, string value, bool isStrictCompare)
```

**Пример**

```
local value1 = DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObjectType_Percent, "0", false)
local entry1 = DocumentAPI.ConditionalFormatIconSetEntry(value1, DocumentAPI.ConditionalFormatIconSet_FiveBoxes, 0)
```

#### 5.63.1 Метод ConditionalFormatValueObject:getType

Метод возвращает тип текущего порогового значения.

**Вызов**

```
ConditionalFormatValueObjectType getType()
```

**Возвращает**

– тип порогового значения, тип `ConditionalFormatValueObjectType`.

#### 5.63.2 Метод ConditionalFormatValueObject:getValue

Метод возвращает значение порога.

**Вызов**

```
string getValue()
```

**Возвращает**

– значение порога, тип `string`.

### 5.63.3 Метод ConditionalFormatValueObject:isStrictCompare

Метод позволяет определить, используется ли строгое сравнение в текущем пороговом значении (">" вместо ">=").

**Вызов**

```
bool isStrictCompare()
```

**Возвращает**

– `true`, если используется строгое сравнение, в ином случае – `false`.

### 5.63.4 Метод ConditionalFormatValueObject:setStrictCompare

Метод позволяет использовать строгое сравнение в текущем пороговом значении (">" вместо ">=").

**Вызов**

```
setStrictCompare(isStrictCompare)
```

**Параметры**

– `isStrictCompare`: `true`, чтобы использовать строгое сравнение, в ином случае – `false`.

### 5.63.5 Метод ConditionalFormatValueObject:setType

Метод позволяет задать тип текущего порогового значения.

**Вызов**

```
setType(type)
```

**Параметры**

– `type`: тип порогового значения, тип `ConditionalFormatValueObjectType`.

### 5.63.6 Метод ConditionalFormatValueObject:setValue

Метод позволяет задать значение порога.

**Вызов**

```
setValue(value)
```

**Параметры**

– `value`: значение порога, тип `string`.

### 5.64 Таблица DocumentAPI.ConditionalFormatValueObjectType

Таблица `ConditionalFormatValueObjectType` содержит типы пороговых значений, которые используются в правилах условного форматирования. Используется в методах `ConditionalFormatValueObject:getType()` и `ConditionalFormatValueObject:setType()`.

Таблица 37 – Типы пороговых значений

| Значение | Описание |
|---|---|
| `ConditionalFormatValueObjectType_Percentile` | Процентиль |
| `ConditionalFormatValueObjectType_Value` | Фиксированное число |
| `ConditionalFormatValueObjectType_Percent` | Процент от наибольшего числа |
| `ConditionalFormatValueObjectType_Formula` | Результат формулы |
| `ConditionalFormatValueObjectType_Min` | Наименьшее число |
| `ConditionalFormatValueObjectType_Max` | Наибольшее число |
| `ConditionalFormatValueObjectType_AutoMin` | Автоматически устанавливает нижнее значение (только для правила "Гистограмма") |
| `ConditionalFormatValueObjectType_AutoMax` | Автоматически устанавливает верхнее значение (только для правила "Гистограмма") |

Для корректного задания пороговых значений, при использовании типов `Min`, `Max`, `AutoMin` и `AutoMax`, необходимо дополнительно указать значение порога – 0.

**Примеры**

```
local value1 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Percent, "0", false)
local entry1 = DocumentAPI.ConditionalFormatIconSetEntry(value1,
DocumentAPI.ConditionalFormatIconSet_FiveBoxes, 0)
```

```
local value1 = DocumentAPI.ConditionalFormatValueObject()
value1:setType(DocumentAPI.ConditionalFormatValueObjectType_AutoMin)
```

```
value1:setValue("0")
local value2 = DocumentAPI.ConditionalFormatValueObject()
value2:setType(DocumentAPI.ConditionalFormatValueObjectType_AutoMax)
value2:setValue("0")

local dataBarParams = DocumentAPI.ConditionalFormatDataBarParams(value1, value2)
```

### 5.65 Таблица DocumentAPI.ConditionalTableFilter

Таблица `ConditionalTableFilter` реализует фильтр, содержащий предикат(ы) для фильтрации строк. Согласно схеме XML, можно использовать одно или два условия, которые объединяются с помощью логической операции AND или OR. На самом деле поддерживается больше критериев, но рекомендуется использовать только один или два. Если не было добавлено ни одного унарного условия, этот фильтр очищает любой другой фильтр, примененный к определенному столбцу. Этот фильтр сохраняется в документе, но редактор не полностью его поддерживает. Фильтр может быть загружен и применен редактором, но если документ изменяется, фильтр будет изменен на тип `ValuesTableFilter`. Если этот фильтр применяется через API, и документ не изменяется после применения фильтра, он будет сохранен как `ConditionalTableFilter`.

Конструктор по умолчанию, логическая операция фильтра, по умолчанию - OR:

```
ConditionalTableFilter()
```

Конструктор с параметром, задающим логическую операцию фильтра

```
ConditionalTableFilter(bool andOperation)
```

**Параметр**

- `andOperation` – логическая операция фильтра, по умолчанию - OR. В дальнейшем может быть изменена методом `ConditionalTableFilter:setAndOperation`.

Конструктор копирования:

```
ConditionalTableFilter(ConditionalTableFilter other)
```

Пример использования приведен в разделе Работа с фильтрами.

#### 5.65.1 Метод ConditionalTableFilter:setAndOperation

Метод `ConditionalTableFilter:setAndOperation` устанавливает логическую операцию AND. Логическая операция применяется, если определено более одного унарного

критерия. Логическая операция по умолчанию - OR.

**Пример**

```
local songFilter = DocumentAPI.ConditionalTableFilter(johnPaulFilter)
songFilter:setAndOperation(true)
```

### 5.65.2 Методы добавления условий

Эти методы добавляют в фильтр условия сравнения со значениями, которые передаются в качестве аргумента. Если значение ячейки не соответствует указанным критериям, строки будут скрыты при применении фильтра.

- Критерии **match** и **notMatch** не могут быть сохранены для документов формата OXML.
- Критерии **aboveAverage**, **belowAverage**, **topValues**, **bottomValues**, **topPercent** и **bottomPercent** могут быть сохранены для документов формата OXML только если они являются единственными в фильтре.

```
equal(string value)
notEqual(string value)
less(string value)
lessOrEqual(string value)
greater(string value)
greaterOrEqual(string value)
match(string value)
notMatch(string value)
begins(string value)
notBegins(string value)
ends(string value)
notEnds(string value)
contains(string value)
notContains(string value)
aboveAverage()
belowAverage()
topValues(number numValues)
bottomValues(number numValues)
topPercent(number percentage)
bottomPercent(number percentage)
```

**Пример**

```
local songFilter = DocumentAPI.ConditionalTableFilter(johnPaulFilter)
songFilter:notEqual("")
songFilter:notBegins("TODO")
```

Пример использования приведен в разделе Работа с фильтрами.

### 5.66 Таблица DocumentAPI.ContainingTableFilter

Таблица `ContainingTableFilter` содержит возможные типы фильтров столбцов таблицы. Используется в методе `TableFilters:getFilterType()`.

Таблица 38 – Типы фильтров

| Значение | Описание |
|---|---|
| `ContainingTableFilter_None` | Фильтр отсутствует |
| `ContainingTableFilter_Values` | Фильтр по значению (`ValuesTableFilter`) |
| `ContainingTableFilter_Conditional` | Фильтр по условию (`ConditionalTableFilter`) |
| `ContainingTableFilter_Color` | Не поддерживается |

**Пример**

```
if filters:getFilterType(0) == DocumentAPI.ContainingTableFilter_Values then
    valuesFilter = filters:getAsValueFilter(0)
end
```

### 5.67 Таблица DocumentAPI.ContentControl

Базовая таблица для элементов управления (см. Работа с элементами управления).

**Наследники**

- `CheckBoxControl`
- `DatePickerControl`
- `DropListControl`
- `InputFieldControl`

#### 5.67.1 Метод ContentControl:canEdit

Метод возвращает `true`, если значение элемента управления может быть изменено, и `false` в обратном случае.

### 5.67.2 Метод ContentControl:getTitle

Метод возвращает название элемента управления.

### 5.67.3 Методы toCheckBox, toInputField, toDatePicker, toDropList

Методы преобразуют объект `ContentControl` в объект соответствующего типа. Возвращают `nil`, если преобразование невозможно.

**Пример**

```
local controls = document:getContentControls()

local checkBox = controls:findByTitle("check"):toCheckBox()
local inputField = controls:findByTitle("input"):toInputField()
local startDate = controls:findByTitle("date"):toDatePicker()
local comboBox = controls:findByTitle("select"):toDropList()
```

## 5.68 Таблица DocumentAPI.ContentControls

Предоставляет доступ к операциям с элементами управления в документе (см. Работа с элементами управления). Метод `ContentControls:findByTitle(string)` позволяет получить элемент управления `ContentControl` по его названию.

**Пример**

```
local controls = document:getContentControls()
local textField = controls:findByTitle("input")
```

## 5.69 Таблица DocumentAPI.CurrencyCellFormatting

Таблица содержит параметры для денежного формата ячеек таблицы.

Таблица 39 – Описание полей таблицы `DocumentAPI.CurrencyCellFormatting`

| Поле | Тип | Описание |
|---|---|---|
| `CurrencyCellFormatting.decimalPlaces` | `number` | Количество десятичных позиций |

| Поле | Тип | Описание |
|---|---|---|
| CurrencyCellFormatting.symbol | string | Символ денежной единицы |
| CurrencyCellFormatting.localeCode | number | Идентификатор кода языка (MS-LCID) |
| CurrencyCellFormatting.useRedForNegative | bool | Использовать красный цвет для отрицательных значений |
| CurrencyCellFormatting.useBracketsForNegative | bool | Использовать скобки для отрицательных значений |
| CurrencyCellFormatting.hideSign | bool | Скрывать знак «минус» для отрицательных значений |
| CurrencyCellFormatting.useThousandsSeparator | bool | Использовать разделитель для тысячных |
| CurrencyCellFormatting.currencySignPlacement | CurrencySignPlacement | Варианты размещения знака валюты |

Данная таблица используется в качестве аргумента метода Cell:setFormat(), см. пример.

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")

local currencyCellFormatting = DocumentAPI.CurrencyCellFormatting()
currencyCellFormatting.decimalPlaces = 2
currencyCellFormatting.useThousandsSeparator = true
currencyCellFormatting.useRedForNegative = true
currencyCellFormatting.useBracketsForNegative = true
currencyCellFormatting.hideSign = false
currencyCellFormatting.currencySignPlacement =
DocumentAPI.CurrencySignPlacement_Suffix

cell:setFormat(currencyCellFormatting)
print(cell:getFormattedValue())
```

### 5.70 Таблица DocumentAPI.CurrencySignPlacement

Варианты размещения знака валюты представлены в таблице 40. Используется в полях `LocaleInfo.currencyFormat` и `CurrencyCellFormatting.currencySignPlacement`.

Таблица 40 – Описание вариантов расположения знаков валюты

| Значение | Описание |
|---|---|
| `DocumentAPI.CurrencySignPlacement_Prefix` | Символ валюты перед значением ($12.00) |
| `DocumentAPI.CurrencySignPlacement_Suffix` | Символ валюты после значения (12.00 Р) |

### 5.71 Таблица DocumentAPI.DataBarConditionalFormatOperator

Таблица `DataBarConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Гистограмма". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

**Конструктор**

```
DataBarConditionalFormatOperator(ConditionalFormatDataBarParams params)
```

**Пример**

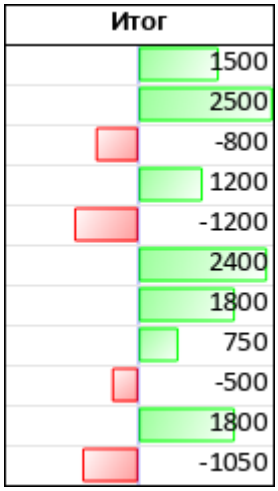



Рисунок 41 – Пример создания правила "Гистограмма"

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local cellRange = sheet:getCellRange("G2:G12")
local cellRangePosition = cellRange:getTableRange()

local value1 = DocumentAPI.ConditionalFormatValueObject()
value1:setType(DocumentAPI.ConditionalFormatValueObjectType_AutoMin)
```

```
value1:setValue("0")
local value2 = DocumentAPI.ConditionalFormatValueObject()
value2:setType(DocumentAPI.ConditionalFormatValueObjectType_AutoMax)
value2:setValue("0")

local dataBarParams = DocumentAPI.ConditionalFormatDataBarParams(value1, value2)
dataBarParams.barFill = DocumentAPI.Color(DocumentAPI.ColorRGBA(0, 255, 0, 100))
dataBarParams.borderColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(0, 255, 0,
255))
dataBarParams.negativeBarFill = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0,
0, 100))
dataBarParams.negativeBorderColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(255,
0, 0, 255))
dataBarParams.axisPosition =
DocumentAPI.ConditionalFormatDataBarAxisPosition_Middle
dataBarParams.axisColor = DocumentAPI.Color(DocumentAPI.ColorRGBA(0, 0, 255,
100))
dataBarParams.fillType = DocumentAPI.ConditionalFormatDataBarFillType_Gradient
dataBarParams.direction =
DocumentAPI.ConditionalFormatDataBarDirection_LeftToRight
dataBarParams.isValueVisible = true

local dataBarOperator =
DocumentAPI.createDataBarConditionalFormatOperator(dataBarParams)

local dataBarRule = DocumentAPI.ConditionalFormatRule()
dataBarRule:setRange(cellRangePosition)
dataBarRule:setOperator(dataBarOperator)
rules:addRule(dataBarRule)
```

### 5.71.1 Метод DataBarConditionalFormatOperator:getAxisColor

Метод возвращает цвет оси между положительными и отрицательными значениями.

**Вызов**

```
Color getAxisColor()
```

**Возвращает**

– цвет оси, тип Color.

### 5.71.2 Метод DataBarConditionalFormatOperator:getAxisPosition

Метод возвращает позицию оси между положительными и отрицательными значениями.

**Вызов**

```
ConditionalFormatDataBarAxisPosition getAxisPosition()
```

**Возвращает**

– расположение оси, тип ConditionalFormatDataBarAxisPosition.

### 5.71.3 Метод DataBarConditionalFormatOperator:getBarFill

Метод возвращает цвет заливки гистограммы для положительных значений.

**Вызов**

```
Color getBarFill()
```

**Возвращает**

– цвет заливки для положительных значений, тип Color.

### 5.71.4 Метод DataBarConditionalFormatOperator:getBorderColor

Метод возвращает цвет границ гистограммы для положительных значений.

**Вызов**

```
Color getBorderColor()
```

**Возвращает**

– цвет границ для положительных значений, тип Color.

### 5.71.5 Метод DataBarConditionalFormatOperator:getDirection

Метод возвращает направление гистограммы в ячейках.

**Вызов**

```
ConditionalFormatDataBarDirection getDirection()
```

**Возвращает**

– направление гистограммы, тип ConditionalFormatDataBarDirection.

### 5.71.6 Метод DataBarConditionalFormatOperator:getFillType

Метод возвращает тип заливки гистограммы.

**Вызов**

```
ConditionalFormatDataBarFillType getFillType()
```

**Возвращает**

– тип заливки гистограммы, тип `ConditionalFormatDataBarFillType`.

### 5.71.7 Метод DataBarConditionalFormatOperator:getLowerThreshold

Метод возвращает нижнее пороговое значение гистограммы.

**Вызов**

```
ConditionalFormatValueObject getLowerThreshold()
```

**Возвращает**

– нижнее пороговое значение гистограммы, тип `ConditionalFormatValueObject`.

### 5.71.8 Метод DataBarConditionalFormatOperator:getMaxLength

Метод возвращает максимальную длину гистограммы, в процентах от ширины ячейки.

**Вызов**

```
number getMaxLength()
```

**Возвращает**

– максимальная длина гистограммы, в процентах, тип `number`.

### 5.71.9 Метод DataBarConditionalFormatOperator:getMinLength

Метод возвращает минимальную длину гистограммы, в процентах от ширины ячейки.

**Вызов**

```
number getMinLength()
```

**Возвращает**

– минимальная длина гистограммы, в процентах, тип `number`.

### 5.71.10 Метод DataBarConditionalFormatOperator:getNegativeBarFill

Метод возвращает цвет заливки гистограммы для отрицательных значений.

**Вызов**

```
Color getNegativeBarFill()
```

**Возвращает**

– цвет заливки для отрицательных значений, тип Color.

### 5.71.11 Метод DataBarConditionalFormatOperator:getNegativeBorderColor

Метод возвращает цвет границ гистограммы для отрицательных значений.

**Вызов**

```
Color getNegativeBorderColor()
```

**Возвращает**

– цвет границ для отрицательных значений, тип Color.

### 5.71.12 Метод DataBarConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

### 5.71.13 Метод DataBarConditionalFormatOperator:getUpperThreshold

Метод возвращает верхнее пороговое значение гистограммы.

**Вызов**

```
ConditionalFormatValueObject getUpperThreshold()
```

**Возвращает**

– верхнее пороговое значение гистограммы, тип ConditionalFormatValueObject.

### 5.71.14 Метод DataBarConditionalFormatOperator:getValueVisibility

Метод позволяет определить показывается ли значение в ячейке с гистограммой.

**Вызов**

```
bool getValueVisibility()
```

**Возвращает**

– `true`, если значение показывается в ячейке с гистограммой, в ином случае – `false`.

### 5.71.15 Метод DataBarConditionalFormatOperator:setAxisColor

Метод позволяет задать цвет оси между положительными и отрицательными значениями.

**Вызов**

```
setAxisColor(color)
```

**Параметры**

– `color`: цвет оси, тип `Color`.

### 5.71.16 Метод DataBarConditionalFormatOperator:setAxisPosition

Метод позволяет задать позицию оси между положительными и отрицательными значениями.

**Вызов**

```
setAxisPosition(position)
```

**Параметры**

– `position`: расположение оси, тип `ConditionalFormatDataBarAxisPosition`.

### 5.71.17 Метод DataBarConditionalFormatOperator:setBarFill

Метод позволяет задать цвет заливки гистограммы для положительных значений.

**Вызов**

```
setBarFill(fill)
```

**Параметры**

– `fill`: цвет заливки для положительных значений, тип `Color`.

### 5.71.18 Метод DataBarConditionalFormatOperator:setBorderColor

Метод позволяет задать цвет границ гистограммы для положительных значений.

**Вызов**

```
setBorderColor(color)
```

**Параметры**

– `color`: цвет границ для положительных значений, тип `Color`.

### 5.71.19 Метод DataBarConditionalFormatOperator:setDirection

Метод позволяет задать направление гистограммы в ячейках.

**Вызов**

```
setDirection(direction)
```

**Параметры**

– `direction`: направление гистограммы, тип `ConditionalFormatDataBarDirection`.

### 5.71.20 Метод DataBarConditionalFormatOperator:setFillType

Метод позволяет задать тип заливки гистограммы.

**Вызов**

```
setFillType(fillType)
```

**Параметры**

– `fillType`: тип заливки гистограммы, тип `ConditionalFormatDataBarFillType`.

### 5.71.21 Метод DataBarConditionalFormatOperator:setLowerThreshold

Метод позволяет задать нижнее пороговое значение гистограммы.

**Вызов**

```
setLowerThreshold(lowerThreshold)
```

**Параметры**

– `lowerThreshold`: нижнее пороговое значение гистограммы, тип `ConditionalFormatValueObject`.

### 5.71.22 Метод DataBarConditionalFormatOperator:setMaxLength

Метод задает максимальную длину гистограммы, в процентах от ширины ячейки.

**Вызов**

```
setMaxLength(number maxLength)
```

**Параметры**

– `maxLength`: максимальная длина гистограммы, в процентах, тип `number`.

### 5.71.23 Метод DataBarConditionalFormatOperator:setMinLength

Метод задает минимальную длину гистограммы, в процентах от ширины ячейки.

**Вызов**

```
setMinLength(number minLength)
```

**Параметры**

– `minLength`: минимальная длина гистограммы, в процентах, тип `number`.

### 5.71.24 Метод DataBarConditionalFormatOperator:setNegativeBarFill

Метод позволяет задать цвет заливки гистограммы для отрицательных значений.

**Вызов**

```
setNegativeBarFill(fill)
```

**Параметры**

– `fill`: цвет заливки для отрицательных значений, тип `Color`.

### 5.71.25 Метод DataBarConditionalFormatOperator:setNegativeBorderColor

Метод позволяет задать цвет границ гистограммы для отрицательных значений.

**Вызов**

```
setNegativeBorderColor(color)
```

**Параметры**

– `color`: цвет границ для отрицательных значений, тип `Color`.

### 5.71.26 Метод DataBarConditionalFormatOperator:setUpperThreshold

Метод позволяет задать верхнее пороговое значение гистограммы.

**Вызов**

```
setUpperThreshold(upperThreshold)
```

**Параметры**

– `upperThreshold`: верхнее пороговое значение гистограммы, тип `ConditionalFormatValueObject`.

### 5.71.27 Метод DataBarConditionalFormatOperator:setValueVisibility

Метод задает видимость значения в ячейке с гистограммой.

**Вызов**

```
setValueVisibility(visible)
```

**Параметры**

– `visible`: `true`, чтобы показывать значение в ячейке с гистограммой, в ином случае – `false`.

## 5.72 Таблица DocumentAPI.DataValidation

Таблица `DataValidation` представляет собой настройки проверки данных (см. Проверка данных). Используется в методах `CellRange:setDataValidation()`, `Cell:getDataValidation()` и `DataValidationResult:getDataValidation()`.

### 5.72.1 Метод DataValidation:clear

Метод очищает все настройки текущей проверки данных.

**Вызов**

```
clear()
```

**Пример**

```
local cell = sheet:getCell("C4")
local validation = cell:getDataValidation()
if not validation:isEmpty() then
    validation:clear()
end
```

### 5.72.2 Метод DataValidation:getAllowBlank

Метод возвращает разрешен ли ввод пустых значений.

**Вызов**

```
bool getAllowBlank()
```

**Возвращает**

– `true`, если проверка данных разрешает ввод пустых значений, в ином случае – `false`.

### 5.72.3 Метод DataValidation:getErrorMessage

Метод возвращает текст сообщения об ошибке.

**Вызов**

```
string getErrorMessage()
```

**Возвращает**

– текст сообщения об ошибке, тип `string`.

### 5.72.4 Метод DataValidation:getErrorStyle

Метод возвращает значение, которое определяет поведение редактора при вводе недопустимых значений.

**Вызов**

```
DataValidationErrorStyle getErrorStyle()
```

**Возвращает**

– поведение редактора при вводе недопустимых значений, тип `DataValidationErrorStyle`.

### 5.72.5 Метод DataValidation:getErrorTitle

Метод возвращает заголовок сообщения об ошибке.

**Вызов**

```
string getErrorTitle()
```

**Возвращает**

– заголовок сообщения об ошибке, тип `string`.

### 5.72.6 Метод DataValidation:getFormula1

Метод возвращает первый аргумент проверки данных. При проверке по списку значений, первым и единственным аргументом может быть:

- формула, результатом которой является диапазон ячеек (`"=INDIRECT("'[Source.xods]Data'!$E$2:$E$6")"`);
- именованный диапазон (`"=Countries"`);
- адрес диапазона ячеек (`"='Data'!$E$2:$E$6"`);
- значения, разделенные точкой с запятой (`"UK; Italy; Germany; Austria; Brazil"`).

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()-7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=StartDate"`);
- адрес ячейки (`"='Data'!C2"`);
- дата в поддерживаемом формате (`"31.12.2024"`).

**Вызов**

```
string getFormula1()
```

**Возвращает**

– первый аргумент проверки данных, тип `string`.

### 5.72.7 Метод DataValidation:getFormula2

Метод возвращает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов `Between` и `NotBetween`. Второй аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()+7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=EndDate"`);
- адрес ячейки (`"='Data'!C2"`);
- дата в поддерживаемом формате (`"31.12.2024"`).

**Вызов**

```
string getFormula2()
```

**Возвращает**

– второй аргумент проверки данных, тип `string`.

### 5.72.8 Метод DataValidation:getOperator

Метод возвращает оператор сравнения введенного значения с аргументом/аргументами. Используется только при проверке дат.

**Вызов**

```
DataValidationOperator getOperator()
```

**Возвращает**

– оператор сравнения, тип DataValidationOperator.

### 5.72.9 Метод DataValidation:getPrompt

Метод возвращает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

**Вызов**

```
string getPrompt()
```

**Возвращает**

– текст подсказки, тип `string`.

### 5.72.10 Метод DataValidation:getPromptTitle

Метод возвращает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

**Вызов**

```
string getPromptTitle()
```

**Возвращает**

– заголовок подсказки, тип `string`.

### 5.72.11 Метод DataValidation:getShowDropDown

Метод возвращает значение, которое определяет показывать ли выпадающий список значений. Используется только при проверке данных по списку значений.

**Вызов**

```
bool getShowDropDown()
```

**Возвращает**

– `true`, если выпадающий список значений показывается, в ином случае – `false`.

### 5.72.12 Метод DataValidation:getShowErrorMessage

Метод возвращает значение, которое определяет показывать ли сообщение об ошибке при вводе недопустимых значений.

**Вызов**

```
bool getShowErrorMessage()
```

**Возвращает**

– `true`, если сообщение об ошибке показывается, в ином случае – `false`.

### 5.72.13 Метод DataValidation:getShowInputMessage

Метод возвращает значение, которое определяет показывать ли подсказку при вводе данных в ячейку с проверкой.

**Вызов**

```
bool getShowInputMessage()
```

**Возвращает**

– `true`, если подсказка показывается, в ином случае – `false`.

### 5.72.14 Метод DataValidation:getType

Метод возвращает тип проверки данных.

**Вызов**

```
DataValidationType getType()
```

**Возвращает**

– тип проверки данных, тип `DataValidationType`.

### 5.72.15 Метод DataValidation:isEmpty

Метод позволяет определить наличие заданных настроек проверки данных в текущем объекте `DataValidation`.

**Вызов**

```
bool isEmpty()
```

**Возвращает**

– `true`, если объект не содержит заданных настроек проверки данных, в ином случае – `false`.

**Пример**

```
local cell = sheet:getCell("C4")
local validation = cell:getDataValidation()
if not validation:isEmpty() then
    validation:clear()
end
```

### 5.72.16 Метод DataValidation:setAllowBlank

Метод позволяет разрешить ввод пустых значений.

**Вызов**

```
setAllowBlank(allowBlank)
```

**Параметры**

– `allowBlank`: `true`, чтобы разрешить ввод пустых значений, в ином случае – `false`.

### 5.72.17 Метод DataValidation:setErrorMessage

Метод задает текст сообщения об ошибке.

**Вызов**

```
setErrorMessage(errorMessage)
```

**Параметры**

– `errorMessage`: текст сообщения об ошибке, тип `string`.

**Пример**

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки сообщения об ошибке:
validation:setShowErrorMessage(true)
validation:setErrorStyle(DocumentAPI.DataValidationErrorStyle_Stop)
validation:setErrorTitle("Error")
validation:setErrorMessage("Invalid value. Choose one of the values below:")

cellRange:setDataValidation(validation)
```

### 5.72.18 Метод DataValidation:setErrorStyle

Метод задает поведение редактора при вводе недопустимых значений.

**Вызов**

```
setErrorStyle(errorStyle)
```

**Параметры**

– `errorStyle`: поведение редактора при вводе недопустимых значений, тип `DataValidationErrorStyle`.

**Пример**

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки сообщения об ошибке:
validation:setShowErrorMessage(true)
validation:setErrorStyle(DocumentAPI.DataValidationErrorStyle_Stop)
validation:setErrorTitle("Error")
validation:setErrorMessage("Invalid value. Choose one of the values below:")

cellRange:setDataValidation(validation)
```

### 5.72.19 Метод DataValidation:setErrorTitle

Метод задает заголовок сообщения об ошибке.

**Вызов**

```
setErrorTitle(errorTitle)
```

**Параметры**

– `errorTitle`: заголовок сообщения об ошибке, тип `string`.

**Пример**

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки сообщения об ошибке:
validation:setShowErrorMessage(true)
validation:setErrorStyle(DocumentAPI.DataValidationErrorStyle_Stop)
validation:setErrorTitle("Error")
validation:setErrorMessage("Invalid value. Choose one of the values below:")

cellRange:setDataValidation(validation)
```

### 5.72.20 Метод DataValidation:setFormula1

Метод задает первый аргумент проверки данных. При проверке по списку значений, первым и единственным аргументом может быть:

- формула, результатом которой является диапазон ячеек (`"=INDIRECT("'[Source.xods]Data'!$E$2:$E$6")"`);
- именованный диапазон (`"=Countries"`);
- адрес диапазона ячеек (`"='Data'!$E$2:$E$6"`);
- значения, разделенные точкой с запятой (`"UK; Italy; Germany; Austria; Brazil"`).

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()-7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=StartDate"`);
- адрес ячейки (`"='Data'!C2"`);
- дата в поддерживаемом формате (`"31.12.2024"`).

**Вызов**

```
setFormula1(formula1)
```

**Параметры**

– `formula1`: первый аргумент проверки данных, тип `string`.

**Пример**

```
local dvList = DocumentAPI.DataValidation()
dvList:setType(DocumentAPI.DataValidationType_List)
dvList:setFormula1("UK; Italy; Germany; Austria; Brazil")
dvList:setShowDropDown(true)

cellRange:setDataValidation(dvList)
```

### 5.72.21 Метод DataValidation:setFormula2

Метод задает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов `Between` и `NotBetween`. Второй аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()+7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=EndDate"`);

- адрес ячейки (`"='Data'!C2"`);
- дата в поддерживаемом формате (`"31.12.2024"`).

**Вызов**

```
setFormula2(formula2)
```

**Параметры**

– `formula2`: второй аргумент проверки данных, тип `string`.

**Пример**

```
local dvDate = DocumentAPI.DataValidation()
dvDate:setType(DocumentAPI.DataValidationType_Date)
dvDate:setOperator(DocumentAPI.DataValidationOperator_Between)
dvDate:setFormula1("01.06.2024")
dvDate:setFormula2("31.08.2024")

cellRange:setDataValidation(dvDate)
```

### 5.72.22 Метод DataValidation:setOperator

Метод задает оператор сравнения введенного значения с аргументом/аргументами. Используется только при проверке дат.

**Вызов**

```
setOperator(dataValidationOperator)
```

**Параметры**

– `dataValidationOperator`: оператор сравнения, тип `DataValidationOperator`.

**Пример**

```
local dvDate = DocumentAPI.DataValidation()
dvDate:setType(DocumentAPI.DataValidationType_Date)
dvDate:setOperator(DocumentAPI.DataValidationOperator_Between)
dvDate:setFormula1("01.06.2024")
dvDate:setFormula2("31.08.2024")

cellRange:setDataValidation(dvDate)
```

### 5.72.23 Метод DataValidation:setPrompt

Метод задает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

**Вызов**

```
setPrompt(prompt)
```

**Параметры**

– `prompt`: текст подсказки, тип `string`.

**Пример**

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки подсказки:
validation:setShowInputMessage(true)
validation:setPromptTitle("Restricted input")
validation:setPrompt("Choose values from the drop-down list")

cellRange:setDataValidation(validation)
```

### 5.72.24 Метод DataValidation:setPromptTitle

Метод задает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

**Вызов**

```
setPromptTitle(promptTitle)
```

**Параметры**

– `promptTitle`: заголовок подсказки, тип `string`.

**Пример**

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки подсказки:
validation:setShowInputMessage(true)
validation:setPromptTitle("Restricted input")
validation:setPrompt("Choose values from the drop-down list")

cellRange:setDataValidation(validation)
```

### 5.72.25 Метод DataValidation:setShowDropDown

Метод определяет показывать ли выпадающий список значений. Используется только при проверке данных по списку значений.

**Вызов**

```
setShowDropDown(showDropDown)
```

**Параметры**

– `showDropDown`: `true`, чтобы показывать выпадающий список значений, в ином случае – `false`.

**Пример**

```
local dvList = DocumentAPI.DataValidation()
dvList:setType(DocumentAPI.DataValidationType_List)
dvList:setFormula1("UK; Italy; Germany; Austria; Brazil")
dvList:setShowDropDown(true)

cellRange:setDataValidation(dvList)
```

### 5.72.26 Метод DataValidation:setShowErrorMessage

Метод определяет показывать ли сообщение об ошибке при вводе недопустимых значений.

**Вызов**

```
setShowErrorMessage(showErrorMessage)
```

**Параметры**

– `showErrorMessage`: `true`, чтобы показывать сообщение об ошибке, в ином случае – `false`.

**Пример**

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки сообщения об ошибке:
validation:setShowErrorMessage(true)
validation:setErrorStyle(DocumentAPI.DataValidationErrorStyle_Stop)
validation:setErrorTitle("Error")
validation:setErrorMessage("Invalid value. Choose one of the values below:")

cellRange:setDataValidation(validation)
```

### 5.72.27 Метод DataValidation:setShowInputMessage

Метод определяет показывать ли подсказку при вводе данных в ячейку с проверкой.

**Вызов**

```
setShowInputMessage(showInputMessage)
```

**Параметры**

– `showInputMessage`: `true`, чтобы показывать подсказку, в ином случае – `false`.

**Пример**

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки подсказки:
validation:setShowInputMessage(true)
validation:setPromptTitle("Restricted input")
validation:setPrompt("Choose values from the drop-down list")

cellRange:setDataValidation(validation)
```

### 5.72.28 Метод DataValidation:setType

Метод задает тип проверки данных.

**Вызов**

```
setType(type)
```

**Параметры**

– `type`: тип проверки данных, тип DataValidationType.

**Пример**

```
local dvDate = DocumentAPI.DataValidation()
dvDate:setType(DocumentAPI.DataValidationType_Date)
dvDate:setOperator(DocumentAPI.DataValidationOperator_Between)
dvDate:setFormula1("01.06.2024")
dvDate:setFormula2("31.08.2024")

cellRange:setDataValidation(dvDate)
```

## 5.73 Таблица DocumentAPI.DataValidationErrorStyle

Таблица `DataValidationErrorStyle` определяет поведение редактора при вводе недопустимых значений. Используется в методах DataValidation:getErrorStyle() и DataValidation:setErrorStyle().

Таблица 41 – Описание поведений редактора при вводе недопустимых значений

| Значение | Описание |
|---|---|
| `DataValidationErrorStyle_Stop` | Запрещает ввод в ячейку недопустимого значения |
| `DataValidationErrorStyle_Warning` | Позволяет ввести в ячейку недопустимое значение после соответствующего предупреждения |
| `DataValidationErrorStyle_Information` | Поведение идентично значению `Warning` |

### 5.74 Таблица DocumentAPI.DataValidationOperator

Таблица `DataValidationOperator` определяет оператор сравнения введенной даты с аргументом/аргументами. Используется в методах `DataValidation:getOperator()` и `DataValidation:setOperator()`.

Таблица 42 – Описание операторов сравнения дат

| Значение | Описание |
|---|---|
| `DataValidationOperator_Between` | Дата должна попадать в интервал между двумя аргументами включительно |
| `DataValidationOperator_NotBetween` | Дата должна быть вне интервала между двумя аргументами |
| `DataValidationOperator_Equal` | Дата должна совпадать со значением первого аргумента |
| `DataValidationOperator_NotEqual` | Дата не должна совпадать со значением первого аргумента |
| `DataValidationOperator_LessThan` | Дата должна быть раньше первого аргумента |
| `DataValidationOperator_LessThanOrEqual` | Дата должна быть раньше первого аргумента или совпадать с ним |
| `DataValidationOperator_GreaterThan` | Дата должна быть позже первого аргумента |
| `DataValidationOperator_GreaterThanOrEqual` | Дата должна быть позже первого аргумента или совпадать с ним |

### 5.75 Таблица DocumentAPI.DataValidationResult

Таблица `DataValidationResult` представляет собой результат проверки значения ячейки (см. Проверка данных). Используется в методе `Cell:checkDataValidation()`.

### 5.75.1 Метод DataValidationResult:getDataValidation

Метод возвращает настройки проверки данных, если ячейка содержит недопустимое значение.

**Вызов**

```
DataValidation getDataValidation()
```

**Возвращает**

– настройки проверки данных, тип DataValidation.

– `nil`, если ячейка содержит допустимое значение.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("C10")
local result = cell:checkDataValidation()
if not result:isValid() then
    print(result:getDataValidation():getErrorMessage())
end
```

### 5.75.2 Метод DataValidationResult:isValid

Метод возвращает является ли значение ячейки допустимым при текущих настройках проверки данных. При проверке данных по списку текстовых значений, необходимо установить формат ячейки **Текст**.

**Вызов**

```
bool isValid()
```

**Возвращает**

– `true`, если значение ячейки допустимо или проверка данных отсутствует, в ином случае – `false`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("C10")
local result = cell:checkDataValidation()
if not result:isValid() then
    print(result:getDataValidation():getErrorMessage())
end
```

### 5.76 Таблица DocumentAPI.DataValidationType

Таблица `DataValidationType` определяет тип проверки данных. Используется в методах `DataValidation:getType()` и `DataValidation:setType()`.

Таблица 43 – Описание типов проверки данных

| Значение | Описание |
|---|---|
| `DataValidationType_None` | Проверка данных отсутствует |
| `DataValidationType_Integer` | Не поддерживается |
| `DataValidationType_Fractional` | Не поддерживается |
| `DataValidationType_List` | Проверка данных по списку доступных значений |
| `DataValidationType_Date` | Проверка данных в формате **Дата** |
| `DataValidationType_Time` | Не поддерживается |
| `DataValidationType_TextLength` | Не поддерживается |
| `DataValidationType_Custom` | Не поддерживается |

### 5.77 Таблица DocumentAPI.DatePatterns

Форматы даты представлены в таблице 44. Пример использования см. в главе `DocumentAPI.DateTimeCellFormatting`.

Таблица 44 – Форматы даты

| Значение | Описание |
|---|---|
| `DatePatterns_DayMonthTextLongYearLong` | `'mmmm dd, yyyy'` для языка `en_US` |
| `DatePatterns_FullDate` | `'день недели, mmmm dd, yyyy'` для языка `en_US` |
| `DatePatterns_DayMonthNumberLongYearLong` | `'mm/dd/yyyy'` для языка `en_US` |
| `DatePatterns_DayMonthNumberLongYearShort` | `'mm/dd/yy'` для языка `en_US` |
| `DatePatterns_DayMonthNumberShortYearShort` | `'m/dd/yy'` для языка `en_US` |
| `DatePatterns_DayMonthTextShort` | `'dd-mmm'` для языка `en_US` |
| `DatePatterns_MonthTextShortYearShort` | `'mmm-yy'` для языка `en_US` |
| `DatePatterns_DayMonthTextShortYearShort` | `'mmm dd, yy'` для языка `en_US` |
| `DatePatterns_DayMonthYearLongNonLocalizableHyphen` | Нелокализуемый шаблон `'dd-mm-yyyy'` |

| Значение | Описание |
|---|---|
| `DatePatterns_DayMonthYearLongNonLocalizableSlash` | Нелокализуемый шаблон `'dd/mm/yyyy'` |

### 5.78 Таблица DocumentAPI.DatePickerControl

Представляет собой элемент управления "Выбор даты", используемый для ввода дат в документе. Является наследником таблицы `ContentControl`. Методы `DatePickerControl:getValue()` и `DatePickerControl:setValue(DateTime)` позволяют получить и задать значение этого элемента управления.

**Пример**

```
local controls = document:getContentControls()
local startDate = controls:findByTitle("startDate"):toDatePicker()
local endDate = controls:findByTitle("endDate"):toDatePicker()
local value = startDate:getValue()
value.year = value.year + 1
endDate:setValue(value)
```

### 5.79 Таблица DocumentAPI.DateTime

Таблица `DocumentAPI.DateTime` предоставляет дату и время с точностью до секунды. Используется для поля `TrackedChangeInfo.timeStamp`.

Таблица 45 – Описание полей таблицы `DocumentAPI.DateTime`

| Поле | Тип | Описание |
|---|---|---|
| `DateTime.year` | `number` | Год |
| `DateTime.month` | `number` | Месяц |
| `DateTime.day` | `number` | День |
| `DateTime.hour` | `number` | Часы |
| `DateTime.minute` | `number` | Минуты |
| `DateTime.second` | `number` | Секунды |

#### 5.79.1 Метод DateTime:__eq

Метод используется для определения эквивалентности двух значений времени.

### 5.80 Таблица DocumentAPI.DateTimeCellFormatting

Таблица содержит параметры для формата ячеек таблицы типа Дата и Время, используется в качестве аргумента метода `Cell:setFormat()`.

Таблица 46 – Описание полей таблицы `DocumentAPI.DateTimeCellFormatting`

| Поле | Тип | Описание |
|---|---|---|
| `DateTimeCellFormatting.dateListID` | `DatePatterns` | Формат даты |
| `DateTimeCellFormatting.timeListID` | `TimePatterns` | Формат времени |

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")

local dateTimeCellFormatting = DocumentAPI.DateTimeCellFormatting()
dateTimeCellFormatting.dateListID = DocumentAPI.DatePatterns_FullDate
dateTimeCellFormatting.timeListID = DocumentAPI.TimePatterns_LongTime

cell:setFormat(dateTimeCellFormatting)
print(cell:getFormattedValue())
```

### 5.81 Таблица DocumentAPI.DateTimeFormat

В таблице 47 представлены варианты форматирования даты и времени. Используется в качестве параметра метода `CellRange:insertCurrentDateTime()`.

Таблица 47 – Варианты форматирования даты и времени

| Значение | Описание |
|---|---|
| `DocumentAPI.DateTimeFormat_DateTime` | Дата и время |
| `DocumentAPI.DateTimeFormat_Date` | Дата |
| `DocumentAPI.DateTimeFormat_Time` | Время |

### 5.82 Таблица DocumentAPI.document

Таблица `DocumentAPI.Document` осуществляет доступ к содержимому открытого текстового или табличного документа.

**Пример**

```
local para = document:getBlocks():getParagraph(0)
```

#### 5.82.1 Метод document:areMirroredMarginsEnabled

Возвращает состояние режима зеркальных полей в документе (разрешены или запрещены).

**Пример**

```
document:setMirroredMarginsEnabled(true)
print(document:areMirroredMarginsEnabled())
```

#### 5.82.2 Метод document:calculateAllFormulas

Метод пересчитывает все формулы в документе. Используйте этот метод после изменения данных табличного документа, если функция автоматического пересчета формул отключена. Чтобы узнать текущее состояние этой функции, воспользуйтесь методом `Document:getCalculationMode()`.

Также вы можете использовать метод `Document:calculateOutdatedFormulas()` для пересчета только формул, данные которых были изменены, и метод `calculate()` объектов `Table`, `CellRange` или `Cell` для пересчета формул, находящихся в определенном диапазоне.

**Пример**

```
-- пересчет выделенного диапазона ячеек
EditorAPI.getSelection():calculate()

-- пересчет заданного диапазона ячеек
document:getBlocks():getTable(0):getCellRange("A2:A6"):calculate()

-- пересчет всего листа
document:getBlocks():getTable(0):calculate()
```

```
-- пересчет всего документа
document:calculateOutdatedFormulas()
```

### 5.82.3 Метод document:calculateOutdatedFormulas

Метод пересчитывает формулы, данные которых были изменены. Используйте этот метод после изменения данных табличного документа, если функция автоматического пересчета формул отключена. Чтобы узнать текущее состояние этой функции, воспользуйтесь методом `Document:getCalculationMode()`.

Также вы можете использовать метод `calculate()` объектов `Table`, `CellRange` или `Cell` для пересчета формул, находящихся в определенном диапазоне.

### 5.82.4 Метод document:disableEvents

Метод отключает обработку событий табличного документа (см. Обработка событий табличного документа).

**Вызов**

```
disableEvents()
```

Метод `enableEvents()` позволяет включить обработку событий документа. Вы можете использовать метод `isEventsEnabled()`, чтобы узнать текущее состояние обработки событий.

### 5.82.5 Метод document:enableEvents

Метод включает обработку событий табличного документа (см. Обработка событий табличного документа).

**Вызов**

```
enableEvents()
```

Метод `disableEvents()` позволяет отключить обработку событий документа. Вы можете использовать метод `isEventsEnabled()`, чтобы узнать текущее состояние обработки событий.

### 5.82.6 Метод document:enumerateSections

Возвращает таблицу объектов типа DocumentAPI.Section.

**Пример**

```
local sections = document:enumerateSections()
for section in sections do
  print(section:getPageProperties().width)
end
```

### 5.82.7 Метод document:getAbsoluteFilePath

Метод возвращает строку, содержащую абсолютный путь к текущему документу. Получаемый путь имеет ОС - зависимый формат (например, содержит символы "/" для Unix и "\" для Windows).

**Пример**

```
local documentPath = document:getAbsoluteFilePath()
print(documentPath)
```

Ограничения:

- Если документ был создан, но не сохранен, данный метод вернет пустую строку;
- Абсолютный путь может быть получен только для локальных файлов и не будет доступен для получения пути хранения облачного документа;
- В текущей реализации отсутствует возможность полноценного использования метода при совместном редактировании.

### 5.82.8 Метод document:getBlocks

Метод предоставляет доступ к таблице DocumentAPI.Blocks и далее к отдельным фрагментам (абзацам, таблицам и т. д.), из которых состоит документ.

**Пример**

```
local blocks = document:getBlocks()
```

### 5.82.9 Метод document:getBookmarks

Метод предоставляет доступ к таблице закладок DocumentAPI.Bookmarks. Используется только в текстовых документах.

**Пример**

```
local bookmarks = document:getBookmarks()
```

### 5.82.10 Метод document:getCalculationMode

Метод возвращает текущий режим пересчета формул в документе CalculationMode. Чтобы изменить этот режим, используйте метод Document:setCalculationMode().

### 5.82.11 Метод document:getComments

Метод обеспечивает доступ к комментариям документа, возвращает таблицу DocumentAPI.Comments. Используется только в текстовых документах.

**Пример**

```
local comments = document:getComments()
for comment in comments:enumerate() do
  print(comment:getRange())
  print(comment:getText())
  print(comment:getInfo().author)
  print(comment:getInfo().timeStamp)
  print(comment:isResolved())
  print(comment:getReplies())
end
```

### 5.82.12 Метод document:getConditionalFormatRules

Метод позволяет получить коллекцию правил условного форматирования для табличного документа.

**Вызов**

```
ConditionalFormatDocumentRules getConditionalFormatRules()
```

**Возвращает**

– коллекция правил условного форматирования для документа, тип ConditionalFormatDocumentRules.

**Пример**

```
local rules = document:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_AboveAverage
```

```
then
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule:getOperator())
    end
end
```

### 5.82.13 Метод document:getContentControls

Метод предоставляет доступ к списку элементов управления ContentControls, содержащихся в документе (см. Работа с элементами управления).

**Пример**

```
local controls = document:getContentControls()
```

### 5.82.14 Метод document:getFormulaType

Метод возвращает поддерживаемую адресацию ячеек DocumentAPI.FormulaType документа.

**Пример**

```
document:setFormulaType(DocumentAPI.FormulaType_R1C1)
print(document:getFormulaType())
```

### 5.82.15 Метод document:getNamedExpressions

Используется для получения списка именованных диапазонов DocumentAPI.NamedExpressions.

**Пример**

```
namedExpressions = document:getNamedExpressions()
print(namedExpressions)
```

### 5.82.16 Метод document:getPivotTablesManager

Возвращает объект DocumentAPI.PivotTablesManager, который используется для создания сводных таблиц. Метод может быть использован только в табличном редакторе.

**Пример**

```
local pivotTablesManager = document:getPivotTablesManager()
print(pivotTablesManager)
```

### 5.82.17 Метод document:getRange

Метод предоставляет доступ ко всему диапазону DocumentAPI.Range документа.

**Пример**

```
local range = document:getRange()
print(range:extractText())
```

### 5.82.18 Метод document:getScripts

Метод предоставляет доступ к таблице макрокоманд DocumentAPI.Scripts, содержащихся в документе.

**Пример**

```
local scripts = document:getScripts()
for script in document:getScripts():enumerate() do
    print(script:getName())
end
```

### 5.82.19 Метод document:getSections

Возвращает таблицу объектов типа DocumentAPI.Sections.

**Пример**

```
local sections = document:getSections()
for section in sections:enumerate() do
    local properties = section:getPageProperties()
    print(properties.width)
    print(properties.height)
end
```

### 5.82.20 Метод document:getTextStyles

Метод возвращает объект, который позволяет взаимодействовать со стилями абзацев в документе.

**Вызов**

```
TextStyles getTextStyles()
```

**Возвращает**

– объект для работы со стилями абзацев, тип TextStyles.

**Пример**

```
local styles = document:getTextStyles()
for style in styles:enumerate() do
    print(style:getName())
end
```

### 5.82.21 Метод document:isCalculatedOnSave

Метод возвращает состояние функции пересчета формул при сохранении документа. Используйте метод `Document:setCalculatedOnSave()`, чтобы включить или отключить эту функцию.

### 5.82.22 Метод document:isChangesTrackingEnabled

Метод возвращает текущее состояние отслеживания изменений в документе (`true` - включены). Используется только в текстовых документах.

**Пример**

```
local trackingChanges = "Disabled"
if document:isChangesTrackingEnabled() then
   trackingChanges = "Enabled"
end
print(trackingChanges)
```

### 5.82.23 Метод document:isEventsEnabled

Метод возвращает состояние обработки событий для текущего документа (см. Обработка событий табличного документа).

**Вызов**

```
bool isEventsEnabled()
```

Методы `disableEvents()` и `enableEvents()` позволяют отключить и включить обработку событий документа.

### 5.82.24 Метод document:isStructureProtected

Метод возвращает состояние защиты от изменения структуры табличного документа (см. Защита структуры табличного документа).

**Вызов**

```
bool isStructureProtected()
```

### 5.82.25 Метод document:removeStructureProtection

Метод снимает защиту от изменений структуры табличного документа (см. Защита структуры табличного документа).

**Вызов**

```
removeStructureProtection(password)
```

**Параметры**

– `password`: (необязательный) пароль для снятия защиты, тип `string`.

**Пример**

```
document:removeStructureProtection("password")
```

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

### 5.82.26 Метод document:setCalculatedOnSave

Метод позволяет включить и отключить функцию пересчета формул при сохранении документа. Используйте метод `Document.isCalculatedOnSave()`, чтобы узнать текущее состояние этой функции.

**Пример**

```
document:setCalculatedOnSave(false)
```

### 5.82.27 Метод document:setCalculationMode

Метод позволяет задать режим пересчета формул в документе `CalculationMode`. Используйте метод `Document:getCalculationMode()`, чтобы получить текущий режим пересчета.

**Пример**

```
document:setCalculationMode(DocumentAPI.CalculationMode_Manual)
```

### 5.82.28 Метод document:setChangesTrackingEnabled

Метод управляет состоянием отслеживания изменений в документе (включены или выключены). Используется только в текстовых документах.

**Пример**

```
if trackingChanges == "Disabled" then
  document:setChangesTrackingEnabled(true)
  if document:isChangesTrackingEnabled() then
     trackingChanges = "Enabled"
  end
end
```

### 5.82.29 Метод document:setFormulaType

Метод устанавливает поддерживаемую адресацию ячеек DocumentAPI.FormulaType документа.

**Пример**

```
document:setFormulaType(DocumentAPI.FormulaType_A1)
```

### 5.82.30 Метод document:setMirroredMarginsEnabled

Метод позволяет включать и выключать зеркальные поля в документе.

**Пример**

```
document:setMirroredMarginsEnabled(true)
print(document:areMirroredMarginsEnabled())
```

### 5.82.31 Метод document:setPageOrientation

Метод устанавливает альбомную, либо книжную ориентацию страниц в документе (см. DocumentAPI.PageOrientation).

**Пример**

```
document:setPageOrientation(DocumentAPI.PageOrientation_Landscape)
```

### 5.82.32 Метод document:setPageProperties

Метод устанавливает свойство DocumentAPI.PageProperties в документе.

**Пример**

```
local properties = DocumentAPI.PageProperties()
properties.width = 100
```

```
properties.height = 200
document:setPageProperties(properties)
```

#### 5.82.33 Метод document:setStructureProtection

Метод устанавливает защиту от изменений структуры табличного документа (см. Защита структуры табличного документа).

**Вызов**

```
setStructureProtection(password)
```

**Параметры**

– `password:` (необязательный) пароль для установки защиты, тип `string`.

**Пример**

```
document:setStructureProtection("password")
```

Если метод `setStructureProtection()` применяется к документу с уже защищенной структурой, возникает исключение `SpreadsheetProtectionError`.

### 5.83 Таблица DocumentAPI.DropListControl

Представляет собой элемент управления "Выпадающий список" в документе. Является наследником таблицы `ContentControl`. Методы `DropListControl:getValue()` и `DropListControl:setValue(int)` позволяют получить и задать значение этого элемента управления. Метод `DropListControl:getChoices()` возвращает коллекцию элементов, находящихся в выпадающем списке.

**Пример**

```
local controls = document:getContentControls()
local comboBox = controls:findByTitle("select"):toDropList()
comboBox:setValue(2)
```

### 5.84 Таблица DocumentAPI.Field

Таблица `Field` предназначена для реализации некоторых полей, например, содержания.

### 5.85 Таблица DocumentAPI.Fill

Таблица описывает свойства заполнения для `ShapeProperties`, `CellProperties`.

Варианты заполнения:

- без заполнения;
- заполнение цветом;
- фон задается путем к изображению фона.

**Примеры**

```
-- Без заполнения
shapeProperties.fill = DocumentAPI.Fill()
```

```
-- Заполнение цветом
shapeProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(55, 146, 179, 200)))
```

```
-- Заполнение шаблоном из url
shapeProperties.fill = DocumentAPI.Fill("https://fillpattern.url")
```

#### 5.85.1 Метод Fill:getColor

Метод возвращает цвет заполнения `DocumentAPI.Color`.

#### 5.85.2 Метод Fill:getUrl

Метод возвращает путь к изображению, которое используется в качестве заполнения, тип - строка.

#### 5.85.3 Метод Fill:isNoFill

Метод возвращает `true`, если заполнения нет.

### 5.86 Таблица DocumentAPI.FiltersRange

Таблица `FiltersRange` реализует диапазон таблицы, позволяющий манипулировать фильтрами столбцов. Пример использования приведен в разделе Работа с фильтрами.

### 5.86.1 Метод FIltersRange:clear

Метод `FiltersRange:clear()` удаляет автоматически отфильтрованный диапазон и фильтры.

**Пример**

```
local filtersRange = table:createFiltersRange(range)
.....................................
tableFilters:clear()
```

### 5.86.2 Метод FIltersRange:eraseFilters

Метод `FiltersRange:eraseFilters` удаляет фильтры из диапазона. Диапазон фильтрации остается нетронутым.

**Пример**

```
local filtersRange = table:createFiltersRange(range)
.....................................
tableFilters:eraseFilters()
```

### 5.86.3 Метод FiltersRange:getCellRange

Метод `FiltersRange:getCellRange` возвращает диапазон ячеек `CellRange`, содержащий текущий объект фильтрации. Возвращаемый диапазон включает строку заголовка.

**Пример**

```
local cellRange = filtersRange:getCellRange()
print(cellRange:getBeginRow() .. ", " .. cellRange:getLastRow())
```

Более подробный пример приведен в разделе Работа с фильтрами.

### 5.86.4 Метод FiltersRange:getFilters

Метод возвращает фильтры столбцов, которые находятся в текущем диапазоне фильтрации.

**Вызов**

```
TableFilters getFilters()
```

**Возвращает**

– фильтры столбцов, тип TableFilters.

– nil, если диапазон фильтрации недействителен.

**Пример**

```
local filtersRange = sheet:createFiltersRange(cellRange)
-- ...
local filters = filtersRange:getFilters()
```

### 5.86.5 Метод FiltersRange:setFilters

Метод FiltersRange:setFilters устанавливает фильтры TableFilters для столбцов диапазона. Фильтрация выполняется с использованием логической операции AND по столбцам. Нумерация столбцов начинается относительно левой позиции диапазона фильтрации. Если номер столбца в фильтрах превышает диапазон фильтрации, то фильтр будет успешно добавлен, но фильтрация для этого столбца будет пропущена. Такое поведение полезно, если необходимо изменить размер диапазона фильтрации при этом сохранить ранее определенные фильтры. Диапазон фильтрации должен существовать до вызова этого метода. В настоящее время DocumentAPI позволяет создавать диапазон фильтрации только для всего листа табличного документа. Для создания или изменения размера существующего диапазона используется метод Table:createFiltersRange().

**Пример**

```
local filtersRange = table:createFiltersRange(range)
.....................................
local tableFilters = DocumentAPI.TableFilters()
tableFilters:setFilter(0, johnPaulFilter)
tableFilters:setFilter(1, songFilter)
.....................................
filtersRange:setFilters(tableFilters)
```

Более подробный пример приведен в разделе Работа с фильтрами.

### 5.87 Таблица DocumentAPI.FootnoteEndnote

Таблица FootnoteEndnote представляет собой сноску в текстовом документе.

### 5.87.1 Метод FootnoteEndnote:getPosition

Метод возвращает позицию сноски в тексте.

**Вызов**

```
Position getPosition()
```

**Возвращает**

– позиция сноски в тексте, тип Position.

**Пример**

```
local range = document:getRange()
local notes = range:getFootnotesEndnotes()
for note in notes:enumerate() do
    note:getPosition():removeForward()
end
```

### 5.87.2 Метод FootnoteEndnote:getRange

Метод возвращает диапазон содержимого сноски.

**Вызов**

```
Range getRange()
```

**Возвращает**

– диапазон содержимого сноски, тип Range.

**Пример**

```
local range = document:getRange()
local notes = range:getFootnotesEndnotes()
for note in notes:enumerate() do
    print(note:getRange():extractText())
end
```

### 5.87.3 Метод FootnoteEndnote:getType

Метод возвращает тип текущей сноски.

**Вызов**

```
FootnoteEndnoteType getType()
```

**Возвращает**

– тип сноски, тип FootnoteEndnoteType.

### 5.88 Таблица DocumentAPI.FootnoteEndnoteType

Таблица `FootnoteEndnoteType` определяет тип сноски в текстовом документе. Используется в методе `FootnoteEndnote:getType()`.

Таблица 48 – Описание типов сносок

| Значение | Описание |
|---|---|
| `FootnoteEndnoteType_Footnote` | Сноска |
| `FootnoteEndnoteType_Endnote` | Концевая сноска |

### 5.89 Таблица DocumentAPI.FootnotesEndnotes

Таблица `FootnotesEndnotes` представляет собой коллекцию сносок и концевых сносок. Используется в методе `Range:getFootnotesEndnotes()`.

#### 5.89.1 Метод FootnotesEndnotes:enumerate

Метод возвращает перечисление сносок.

**Вызов**

```
FootnoteEndnotesEnumerator enumerate()
```

**Возвращает**

– перечисление сносок.

**Пример**

```
local range = document:getRange()
local notes = range:getFootnotesEndnotes()
for note in notes:enumerate() do
    print(note:getRange():extractText())
end
```

### 5.90 Таблица DocumentAPI.FormulaType

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 49. Используется в методах `document:getFormulaType()` и `document:setFormulaType()`.

Таблица 49 – Системы адресации ячеек в табличном документе

| Значение | Описание | Пример |
|---|---|---|
| `FormulaType_A1` | Наиболее распространенная система адресации ячеек, при которой столбцы задаются буквами, а строки – числами | `='Лист1'!$D$20:$F$25` |
| `FormulaType_R1C1` | Альтернативная система адресации ячеек, при которой столбцы и строки задаются числами | `='Лист1'!R20C4:R25C6` |
| `FormulaType_OpenFormula` | Система адресации ячеек в рамках стандарта ODF | `=['Лист1'.$D$20:.$F$25]` |

### 5.91 Таблица DocumentAPI.FractionCellFormatting

Таблица содержит параметры для дробного формата ячеек таблицы. Данная таблица используется в качестве аргумента метода `Cell:setFormat()`.

Таблица 50 – Описание полей таблицы `DocumentAPI.FractionCellFormatting`

| Поле | Тип | Описание |
|---|---|---|
| `FractionCellFormatting.minNumeratorDigits` | `number` | Количество позиций числителя |
| `FractionCellFormatting.minDenominatorDigits` | `number` | Количество позиций знаменателя |
| `FractionCellFormatting.denominatorValue` | `number` | Знаменатель |

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("B2")

local fractionCellFormatting = DocumentAPI.FractionCellFormatting()
fractionCellFormatting.minNumeratorDigits = 2
fractionCellFormatting.minDenominatorDigits = 3
fractionCellFormatting.denominatorValue = 22
```

```
cell:setFormat(fractionCellFormatting)
print(cell:getFormattedValue())
```

### 5.92 Таблица DocumentAPI.FrozenRangePosition

Таблица `DocumentAPI.FrozenRangePosition` представляет заблокированную область таблицы. Возвращается посредством метода `Table:getFrozenRange()`, устанавливается методом `Table:freeze()`.

#### 5.92.1 Конструкторы

Конструктор с параметрами по умолчанию.

```
FrozenRangePosition()
```

Конструктор, создающий диапазон ячеек. В качестве параметров используются координаты левой верхней и правой нижней точек области.

```
FrozenRangePosition(top, left, bottom, right)
```

**Примеры**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition()
print(frozenRangePosition:isRowsCols())

frozenRangePosition = DocumentAPI.FrozenRangePosition(0, 2, 5, 5)
print(frozenRangePosition:isRowsCols())
```

#### 5.92.2 Метод FrozenRangePosition:create

Создает объект заблокированной области таблицы `FrozenRangePosition`. В качестве параметров используются координаты левой верхней и правой нижней точек области.

**Вызов**

```
FrozenRangePosition create(top, left, bottom, right)
```

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.create(0, 2, 5, 5)
print(frozenRangePosition:isRowsCols())
```

### 5.92.3 Метод FrozenRangePosition:createFrozenArea

Создает объект заблокированной области таблицы `FrozenRangePosition`. Область содержит все ячейки прямоугольника `{0, 0, bottom, right}`.

**Вызов**

```
FrozenRangePosition createFrozenArea(bottom, right)
```

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenArea(0, 2)
print(frozenRangePosition:isArea())
```

### 5.92.4 Метод FrozenRangePosition:createFrozenCols

Создает объект заблокированной области таблицы `FrozenRangePosition`. Область содержит все колонки с `first` по `last`.

**Вызов**

```
FrozenRangePosition createFrozenCols(first, last)
```

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
print(frozenRangePosition:isRowsCols())
```

### 5.92.5 Метод FrozenRangePosition:createFrozenRows

Создает объект заблокированной области таблицы `FrozenRangePosition`. Область содержит все строки с `first` по `last`.

**Вызов**

```
FrozenRangePosition createFrozenRows(first, last)
```

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenRows(0, 2)
print(frozenRangePosition:isRows())
```

### 5.92.6 Метод FrozenRangePosition:isArea

Возвращает `true` если диапазон является непрерывной областью.

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenArea(2, 2)
print(frozenRangePosition:isArea())
```

### 5.92.7 Метод FrozenRangePosition:isCols

Возвращает `true` если диапазон состоит из колонок.

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
print(frozenRangePosition:isCols())
```

### 5.92.8 Метод FrozenRangePosition:isRows

Возвращает `true` если диапазон состоит из строк.

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenRows(0, 2)
print(frozenRangePosition:isRows())
```

### 5.92.9 Метод FrozenRangePosition:isRowsCols

Возвращает `true` если диапазон содержит строки и колонки.

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenArea(2, 2)
print(frozenRangePosition:isRowsCols())
```

### 5.92.10 Метод FrozenRangePosition:__eq

Метод используется для определения эквивалентности двух объектов `FrozenRangePosition`.

**Пример**

```
local pos1 = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local pos2 = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
print(pos1:__eq(pos2)) -- true
```

### 5.93 Таблица DocumentAPI.HeaderFooter

Таблица `DocumentAPI.HeaderFooter` определяет колонтитул текстового документа.

#### 5.93.1 Метод HeaderFooter:getBlocks

Метод предоставляет доступ к блокам (`DocumentAPI.Blocks`), которые содержатся в колонтитуле.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    for block in header:getBlocks():enumerate() do
        print(block:getRange():extractText())
    end
end
```

#### 5.93.2 Метод HeaderFooter:getRange

Метод предоставляет диапазон (`DocumentAPI.Range`) с содержанием верхнего или нижнего колонтитулов.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    print(header:getRange():extractText())
end
```

#### 5.93.3 Метод HeaderFooter:getType

Метод предоставляет информацию о типе колонтитула

(DocumentAPI.HeaderFooterType).

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    if (header:getType() == DocumentAPI.HeaderFooterType_Header) then
       print("Header") else print("Footer")
    end
end
```

### 5.94 Таблица DocumentAPI.HeaderFooterType

Типы колонтитулов представлены в таблице 51.

Таблица 51 – Типы колонтитулов

| Значение | Описание |
|---|---|
| DocumentAPI.HeaderFooterType_Header | Верхний колонтитул |
| DocumentAPI.HeaderFooterType_Footer | Нижний колонтитул |

### 5.95 Таблица DocumentAPI.HeadersFooters

Таблица DocumentAPI.HeadersFooters представляет коллекцию верхних и нижних колонтитулов раздела текстового документа (см. Рисунок 42). Доступ к колонтитулам осуществляется посредством методов Section:getHeaders(), Section:getFooters().

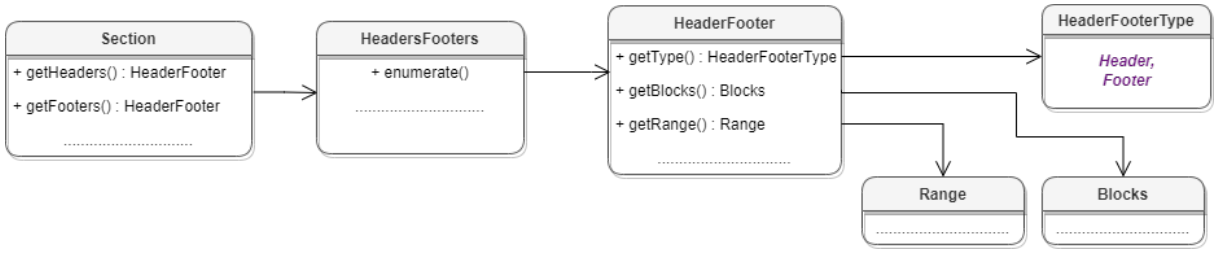



Рисунок 42 – Таблицы колонтитулов

### 5.95.1 Метод HeadersFooters:enumerate

Метод возвращает коллекцию колонтитулов.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    if (header:getType() == DocumentAPI.HeaderFooterType_Header) then
       print("Header") else print("Footer")
    end
end
```

## 5.96 Таблица DocumentAPI.HorizontalAnchorAlignment

В таблице 52 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали. Используется в DocumentAPI.HorizontalTextAnchoredPosition.

Таблица 52 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

| Значение | Описание |
|---|---|
| DocumentAPI.HorizontalAnchorAlignment_Left | По верхнему краю |
| DocumentAPI.HorizontalAnchorAlignment_Right | По нижнему краю |
| DocumentAPI.HorizontalAnchorAlignment_Center | По центру |
| DocumentAPI.HorizontalAnchorAlignment_Inside<br>DocumentAPI.HorizontalAnchorAlignment_Outside | По границам |

## 5.97 Таблица DocumentAPI.HorizontalRelativeTo

В таблице 53 представлены типы размещения объекта относительно закрепленной позиции по горизонтали. Используется в DocumentAPI.HorizontalTextAnchoredPosition.

Таблица 53 – Типы размещения объекта относительно закрепленной позиции по горизонтали

| Значение | Описание |
|---|---|
| DocumentAPI.HorizontalRelativeTo_Character | Символ |
| DocumentAPI.HorizontalRelativeTo_Column | Столбец |

| Значение | Описание |
|---|---|
| `DocumentAPI.HorizontalRelativeTo_ColumnLeftMargin` | Левое поле столбца |
| `DocumentAPI.HorizontalRelativeTo_ColumnRightMargin` | Правое поле столбца |
| `DocumentAPI.HorizontalRelativeTo_ColumnInsideMargin` | Внутреннее поле столбца |
| `DocumentAPI.HorizontalRelativeTo_ColumnOutsideMargin` | Внешнее поле столбца |
| `DocumentAPI.HorizontalRelativeTo_Page` | Страница |
| `DocumentAPI.HorizontalRelativeTo_PageContent` | Содержимое страницы |
| `DocumentAPI.HorizontalRelativeTo_PageLeftMargin` | Левое поле страницы |
| `DocumentAPI.HorizontalRelativeTo_PageRightMargin` | Правое поле страницы |
| `DocumentAPI.HorizontalRelativeTo_PageInsideMargin` | Внутреннее поле страницы |
| `DocumentAPI.HorizontalRelativeTo_PageOutsideMargin` | Внешнее поле страницы |

### 5.98 Таблица DocumentAPI.HorizontalTextAnchoredPosition

Таблица `DocumentAPI.HorizontalTextAnchoredPosition` (см. Рисунок 43) предназначена для управления относительным положением объекта со смещением или выравниванием по горизонтали. Пример использования см. в `InlineFrame:setPosition()`.

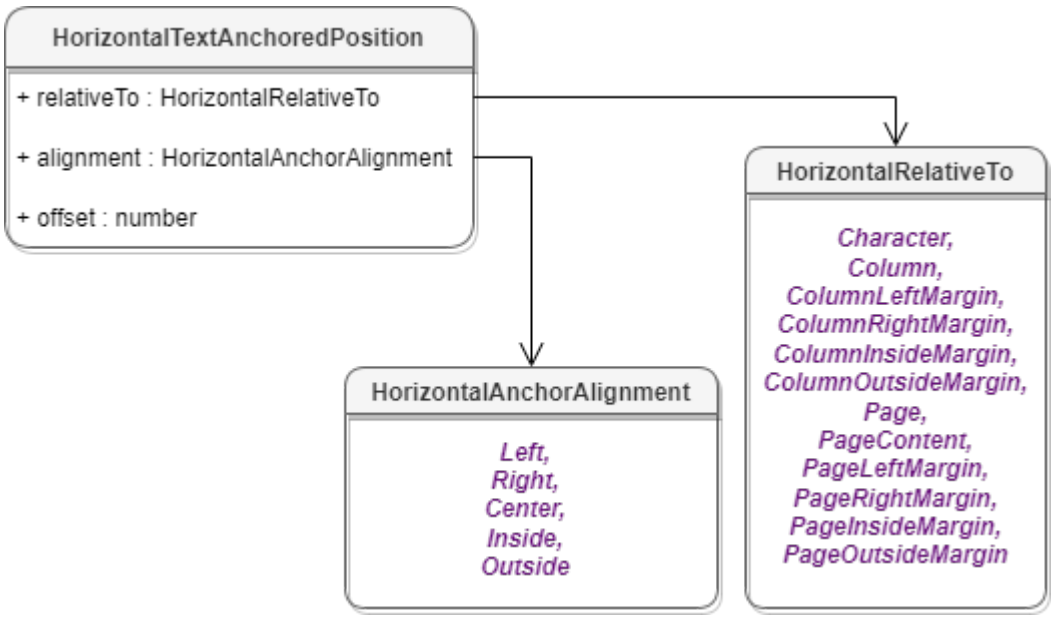



Рисунок 43 – Поля таблицы `DocumentAPI.HorizontalTextAnchoredPosition`

Таблица 54 – Описание полей таблицы `DocumentAPI.HorizontalTextAnchoredPosition`

| Поле | Тип | Описание |
|---|---|---|
| `HorizontalTextAnchoredPosition.relativeTo` | `HorizontalRelativeTo` | Тип размещения объекта относительно закрепленной позиции по горизонтали |
| `HorizontalTextAnchoredPosition.offset` | `number` | Смещение объекта |
| `HorizontalTextAnchoredPosition.alignment` | `HorizontalAnchorAlignment` | Тип выравнивания объекта относительно закрепленной позиции по горизонтали |

### 5.98.1 Метод HorizontalTextAnchoredPosition:__eq

Метод используется для определения эквивалентности двух положений объекта по горизонтали.

**Пример**

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos1.horizontal.offset = 1

local pos2 = DocumentAPI.TextAnchoredPosition()
pos2.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos2.horizontal.offset = 1

print(pos1.horizontal:__eq(pos2.horizontal))
```

### 5.99 Таблица DocumentAPI.Hyperlink

Таблица `DocumentAPI.Hyperlink` описывает свойства ссылки. Может быть получена посредством вызова метода `Cell:getHyperlink()`.

Таблица 55 – Описание полей таблицы `DocumentAPI.Hyperlink`

| Поле | Тип | Описание |
|---|---|---|
| `Hyperlink.url` | `string` | Адрес ссылки |
| `Hyperlink.tooltip` | `string` | Текст подсказки |
| `Hyperlink.label` | `string` | Текст описания |

**Пример**

```
local cell = document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0,
 0))
local hyperlink = cell:getHyperlink()
if (hyperlink ~= nil) then
  print(hyperlink.url, hyperlink.tooltip, hyperlink.label)
end
```

### 5.99.1 Метод Hyperlink:__eq

Метод используется для определения эквивалентности двух объектов `Hyperlink`.

**Пример**

```
table_0 = document:getBlocks():getTable(0)
cell_00 = table_0:getCell(DocumentAPI.CellPosition(0, 0))
cell_01 = table_0:getCell(DocumentAPI.CellPosition(0, 1))
local hyperlink_00 = cell_00:getHyperlink()
local hyperlink_01 = cell_01:getHyperlink()
if (hyperlink_00 and hyperlink_01) then
   print(hyperlink_00:__eq(hyperlink_01))
end
```

## 5.100 Таблица DocumentAPI.IconSetConditionalFormatOperator

Таблица `IconSetConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Значки". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

**Конструктор**

```
      IconSetConditionalFormatOperator(ConditionalFormatIconSetEntries   entries,
bool isValueShown)
```

**Пример**

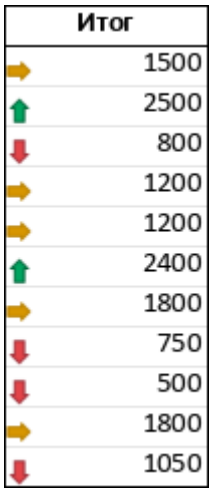



Рисунок 44 – Пример создания правила "Значки"

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local entries = DocumentAPI.ConditionalFormatIconSetEntries()
local value1 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Percent, "0", false)
local entry1 = DocumentAPI.ConditionalFormatIconSetEntry(value1,
DocumentAPI.ConditionalFormatIconSet_ThreeArrows, 0)
entries:addEntry(entry1)
local value2 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Percent, "33", false)
local entry2 = DocumentAPI.ConditionalFormatIconSetEntry(value2,
DocumentAPI.ConditionalFormatIconSet_ThreeArrows, 1)
entries:addEntry(entry2)
local value3 =
DocumentAPI.ConditionalFormatValueObject(DocumentAPI.ConditionalFormatValueObject
Type_Percent, "67", false)
local entry3 = DocumentAPI.ConditionalFormatIconSetEntry(value3,
DocumentAPI.ConditionalFormatIconSet_ThreeArrows, 2)
entries:addEntry(entry3)

local iconSetOperator =
DocumentAPI.createIconSetConditionalFormatOperator(entries, true)

local cellRange = sheet:getCellRange("G2:G12")
local cellRangePosition = cellRange:getTableRange()
```

```
local iconSetRule = DocumentAPI.ConditionalFormatRule()
iconSetRule:setOperator(iconSetOperator)
iconSetRule:setRange(cellRangePosition)
rules:addRule(iconSetRule)
```

### 5.100.1 Метод IconSetConditionalFormatOperator:getEntries

Метод возвращает набор правил отображения значков для текущего оператора.

**Вызов**

```
ConditionalFormatIconSetEntries getEntries()
```

**Возвращает**

– набор правил отображения значков, тип `ConditionalFormatIconSetEntries`.

### 5.100.2 Метод IconSetConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип `ConditionalFormatOperatorType`.

### 5.100.3 Метод IconSetConditionalFormatOperator:isValueShown

Метод позволяет определить, показывается ли значение ячейки при примененном условном форматировании.

**Вызов**

```
isValueShown()
```

**Возвращает**

– `true`, если значение ячейки показывается при примененном форматировании, в ином случае – `false`.

### 5.100.4 Метод IconSetConditionalFormatOperator:setEntries

Метод задает набор правил отображения значков для текущего оператора.

**Вызов**

```
setEntries(entries)
```

**Параметры**

– `entries`: набор правил отображения значков, тип `ConditionalFormatIconSetEntries`.

### 5.100.5 Метод IconSetConditionalFormatOperator:setValueShown

Метод позволяет настроить отображение значения ячейки при примененном условном форматировании.

**Вызов**

```
setValueShown(isValueShown)
```

**Параметры**

– `isValueShown`: `true`, чтобы показывать значение ячейки при примененном форматировании, в ином случае – `false`.

## 5.101 Таблица DocumentAPI.Image

Таблица `DocumentAPI.Image` представляет собой изображение, находящееся в текстовом или табличном документе.

### 5.101.1 Метод Image:getFrame

Метод аналогичен методу `MediaObject:getFrame()`, он возвращает свойства позиции изображения. В зависимости от текущего редактора метод возвращает разные типы рамок. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют тип `DocumentAPI.InlineFrame`, табличные документы работают с абсолютной позицией и используют тип `DocumentAPI.AbsoluteFrame`.

**Пример для текстового документа**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if (image) then
    print(image:getFrame()) -- <userdata of type
'CO::API::Document::InlineFrame'>
```

```
  end
end
```

**Пример для табличного документа**

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if (image) then
    print(image:getFrame()) -- <userdata of type
'CO::API::Document::AbsoluteFrame'>
  end
end
```

### 5.101.2 Метод Image:remove

Метод удаляет изображение из документа.

**Пример для текстового документа**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if image then
    image:remove()
    break
  end
end
```

### 5.101.3 Метод Image:replaceURL

Метод заменяет текущее изображение.

**Вызов**

```
replaceURL(newURL)
```

**Параметры**

– `newURL`: путь к новому файлу изображения, тип `string`.

**Пример**

```
local range = document:getRange()
local images = range:getImages()
for image in images:enumerate() do
    image:replaceURL("logo2025.png")
```

```
    image:getFrame():setDimensions(DocumentAPI.SizeU(150, 150))
end
```

### 5.102 Таблица DocumentAPI.Images

Таблица `DocumentAPI.Images` используется для доступа к коллекции изображений. Может быть получена вызовом методов `Table.getImages()`, `Range.getImages()`.

#### 5.102.1 Метод Images:enumerate

Метод позволяет перечислить коллекцию изображений.

**Пример для текстового документа**

```
for image in EditorAPI.getSelection():getImages():enumerate() do
  print(image:getFrame():getWrapType())
end
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local images = sheet:getImages()
for image in images:enumerate() do
  print(image:getFrame():getTopLeft().x)
end
```

### 5.103 Таблица DocumentAPI.InlineFrame

Таблица `DocumentAPI.InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 45). Предназначена для получения и изменения свойств позиции графических объектов. Используется в текстовом документе.

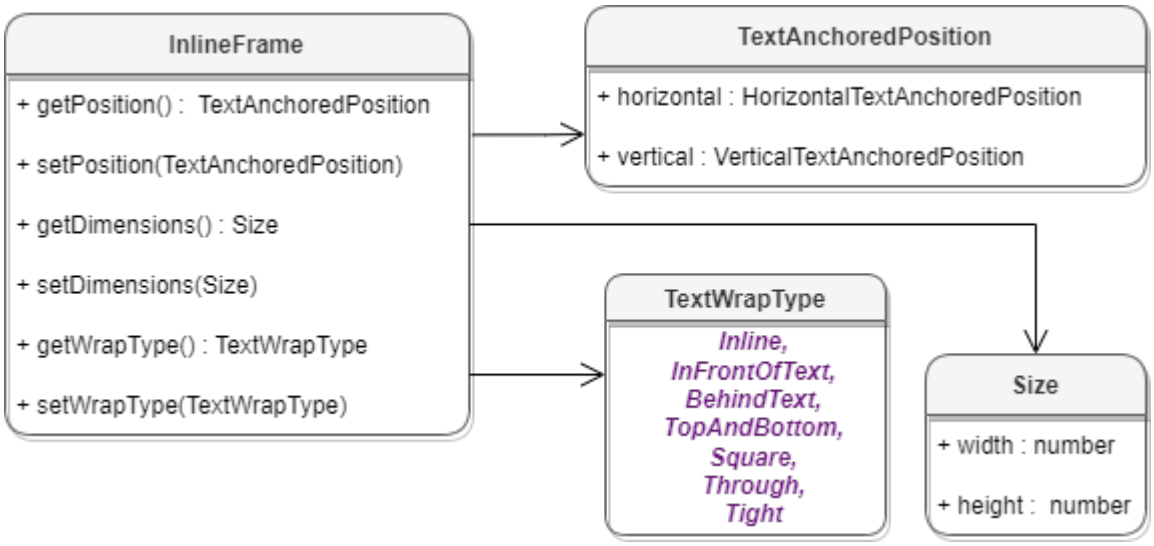



Рисунок 45 – Объектная модель таблицы `DocumentAPI.InlineFrame`

**Пример для текстового документа**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  local inlineFrame = mediaObject:getFrame()
  print(inlineFrame:getDimensions())
  print(inlineFrame:getWrapType())
  print(inlineFrame:getPosition())
end
```

#### 5.103.1 Метод InlineFrame:getDimensions

Метод возвращает задает размеры встроенного объекта, тип - `DocumentAPI.Size`.

**Пример**

```
local mediaObjects = document:getRange():getImages()
for mediaObject in mediaObjects:enumerate() do
  local inlineFrame = mediaObject:getFrame()
  local dimensions = inlineFrame:getDimensions()
  if (dimensions) then
```

```
    print(dimensions.width, dimensions.height)
  end
end
```

### 5.103.2 Метод InlineFrame:getPosition

Метод возвращает позицию встроенного объекта на странице в виде таблицы DocumentAPI.TextAnchoredPosition.

**Пример**

```
local mediaObjects = document:getRange():getImages()
for mediaObject in mediaObjects:enumerate() do
  local inlineFrame = mediaObject:getFrame()
  local textAnchoredPosition = inlineFrame:getPosition()
  if (textAnchoredPosition) then
    print(textAnchoredPosition.horizontal, textAnchoredPosition.vertical)
  end
end
```

### 5.103.3 Метод InlineFrame:getWrapType

Метод возвращает вариант обтекания текстом встроенного объекта (см. DocumentAPI.TextWrapType).

**Пример**

```
local mediaObjects = document:getRange():getImages()
for mediaObject in mediaObjects:enumerate() do
  print(mediaObject:getFrame():getWrapType())
end
```

### 5.103.4 Метод InlineFrame:setDimensions

Метод задает размер DocumentAPI.SizeU встроенного объекта.

**Пример**

```
inlineFrame:setDimensions(DocumentAPI.SizeU(100, 100))
```

### 5.103.5 Метод InlineFrame:setPosition

Метод задает положение встроенного объекта, тип аргумента

DocumentAPI.TextAnchoredPosition. Новая позиция может быть установлена только для встроенных объектов, тип переноса текста которых не является типом DocumentAPI.TextWrapType_Inline.

**Пример**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
   local inlineFrame = mediaObject:getFrame()
   if (inlineFrame:getWrapType() ~= DocumentAPI.TextWrapType_Inline) then
     local pos = DocumentAPI.TextAnchoredPosition()

     -- Установка смещения по горизонтали относительно края колонки
     pos.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
     pos.horizontal.offset = 10
     -- Установка смещения по вертикали относительно края страницы
     pos.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
     pos.vertical.offset = 10

     -- Установка позиции рамки графического объекта
     inlineFrame:setPosition(pos)
   end
end
```

### 5.103.6 Метод InlineFrame:setWrapType

Метод устанавливает вариант обтекания текстом встроенного объекта (см. DocumentAPI.TextWrapType).

**Пример**

```
local mediaObjects = document:getRange():getImages()
for mediaObject in mediaObjects:enumerate() do
  local inlineFrame = mediaObject:getFrame()
  inlineFrame:setWrapType(DocumentAPI.TextWrapType_InFrontOfText)
end
```

### 5.104 Таблица DocumentAPI.InputFieldControl

Представляет собой элемент управления "Поле ввода" в документе. Является наследником таблицы ContentControl. Методы InputFieldControl:getValue() и InputFieldControl:setValue(string) позволяют получить и задать значение этого элемента управления.

**Пример**

```
local controls = document:getContentControls()
local inputField = controls:findByTitle("input"):toInputField()
local text = inputField:getValue():gsub(' ', '_')
inputField.setValue(text)
```

### 5.105 Таблица DocumentAPI.Insets

Таблица DocumentAPI.Insets предназначена для задания полей, например, страницы. Данный тип используется в поле margins таблицы DocumentAPI.PageProperties.

Таблица 56 – Описание полей таблицы DocumentAPI.Insets

| Поле | Тип | Описание |
|---|---|---|
| Insets.left | number | Левая граница поля |
| Insets.top | number | Верхняя граница поля |
| Insets.right | number | Правая граница поля |
| Insets.bottom | number | Нижняя граница поля |

**Пример**

```
local insets = DocumentAPI.Insets()
insets.left = 10.0
print(insets.left)
```

### 5.106 Таблица DocumentAPI.LineEndingProperties

Таблица DocumentAPI.LineEndingProperties содержит варианты оформления окончаний линий. Используется в полях headLineEndingProperties и tailLineEndingProperties таблицы DocumentAPI.LineProperties.

Таблица 57 – Описание полей таблицы `DocumentAPI.LineEndingProperties`

| Поле | Тип | Описание |
|---|---|---|
| `LineEndingProperties.style` | `LineEndingStyle` | Стиль окончания линии |
| `LineEndingProperties.relativeExtent` | `SizeU` | Размер окончания линии относительно ее ширины |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("C3")

lineProperties = DocumentAPI.LineProperties()
lineProperties.headLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.headLineEndingProperties.style = DocumentAPI.LineEndingStyle_Arrow

lineProperties.headLineEndingProperties.relativeExtent = DocumentAPI.SizeU()
lineProperties.headLineEndingProperties.relativeExtent.width = 2
lineProperties.headLineEndingProperties.relativeExtent.height = 2

lineProperties.tailLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.tailLineEndingProperties.style = DocumentAPI.LineEndingStyle_Arrow
lineProperties.tailLineEndingProperties.relativeExtent = DocumentAPI.SizeU()
lineProperties.tailLineEndingProperties.relativeExtent.width = 2
lineProperties.tailLineEndingProperties.relativeExtent.height = 2

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

### 5.106.1 Метод LineEndingProperties:__eq

Метод используется для определения эквивалентности значений двух объектов `LineEndingProperties`.

**Пример**

```
lineEnding1 = DocumentAPI.LineEndingProperties()
lineEnding1.style = DocumentAPI.LineEndingStyle_Arrow
```

```
lineEnding2 = DocumentAPI.LineEndingProperties()
lineEnding2.style = DocumentAPI.LineEndingStyle_Diamond
print("Eq: " .. tostring(lineEnding1:__eq(lineEnding2)))
```

### 5.107 Таблица DocumentAPI.LineEndingStyle

В таблице 58 приведены типы окончания линии. Используется в поле `style` таблицы DocumentAPI.LineEndingProperties.

Таблица 58 – Типы окончания линии

| Значение | Описание |
|---|---|
| DocumentAPI.LineEndingStyle_Arrow |  |
| DocumentAPI.LineEndingStyle_Diamond |  |
| DocumentAPI.LineEndingStyle_Oval |  |
| DocumentAPI.LineEndingStyle_Stealth |  |
| DocumentAPI.LineEndingStyle_Triangle |  |
| DocumentAPI.LineEndingStyle_None |  |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("C3")

lineProperties = DocumentAPI.LineProperties()
lineProperties.headLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.headLineEndingProperties.style = DocumentAPI.LineEndingStyle_Oval

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

### 5.108 Таблица DocumentAPI.LineProperties

Таблица `DocumentAPI.LineProperties` предназначена для установки таких

параметров линии, как тип, ширина, цвет (см. Рисунок 46).

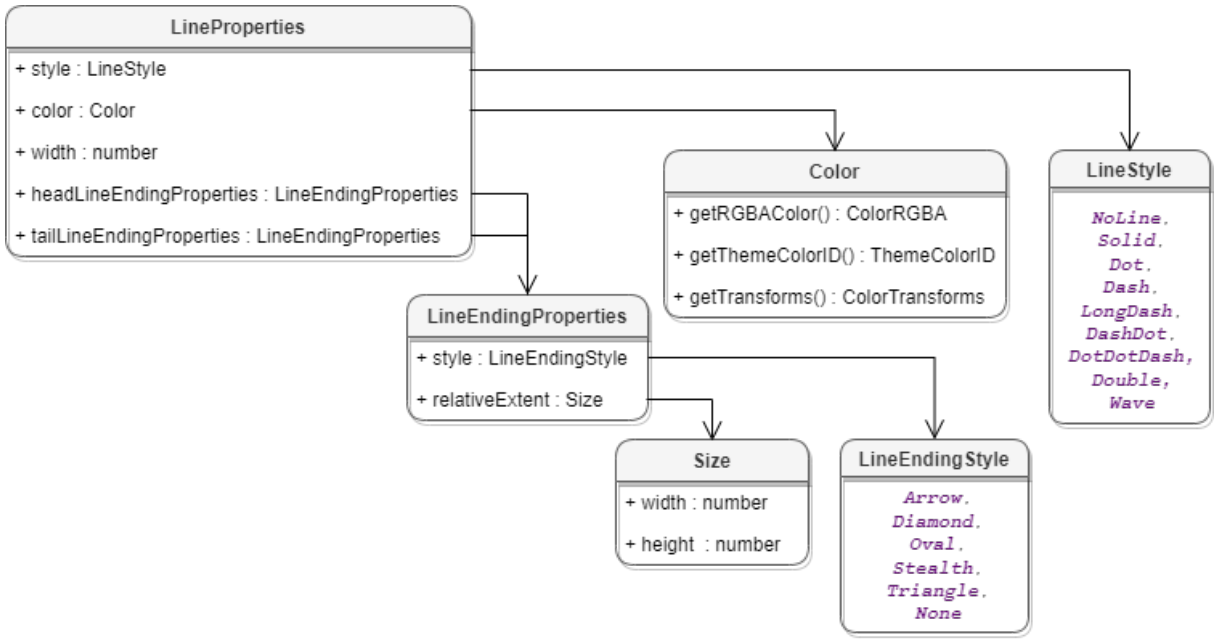



Рисунок 46 – Свойства границ ячеек

Таблица 59 – Описание полей таблицы `LineProperties`

| Поле | Тип | Описание |
|---|---|---|
| `LineProperties.style` | LineStyle | Тип линии |
| `LineProperties.width` | number | Толщина линии |
| `LineProperties.color` | Color | Цвет линии |
| `LineProperties.headLineEndingProperties` | LineEndingProperties | Тип начала линии |
| `LineProperties.tailLineEndingProperties` | LineEndingProperties | Тип окончания линии |

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("C3")

lineProperties = DocumentAPI.LineProperties()
lineProperties.style = DocumentAPI.LineStyle_Solid
lineProperties.width = 1.5
lineProperties.color = DocumentAPI.Color(DocumentAPI.ColorRGBA(55, 146, 179,
200))

borders = DocumentAPI.Borders()
```

```
borders = borders:setTop(lineProperties)
local brds = cell:setBorders(borders)
```

### 5.108.1 Метод LineProperties:__eq

Метод используется для определения эквивалентности значений двух объектов LineProperties.

**Пример**

```
lineProperties1 = DocumentAPI.LineProperties()
lineProperties1.style = DocumentAPI.LineStyle_Solid

lineProperties2 = DocumentAPI.LineProperties()
lineProperties2.style = DocumentAPI.LineStyle_Dot

print("Eq: " .. tostring(lineProperties1:__eq(lineProperties2)))
```

## 5.109 Таблица DocumentAPI.LineSpacing

Таблица DocumentAPI.LineSpacing задает межстрочный интервал абзаца. Для управления значением межстрочного интервала используются значения, представленные в разделе DocumentAPI.LineSpacingRule.

Таблица 60 – Параметры межстрочного интервала

| Поле | Тип | Описание |
|---|---|---|
| LineSpacing.value | number | Значение межстрочного интервала |
| LineSpacing.rule | LineSpacingRule | Правило формирования межстрочного интервала |

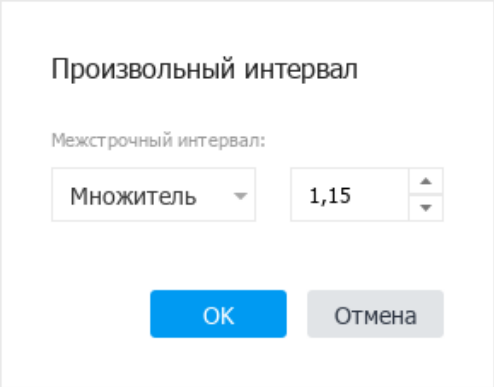
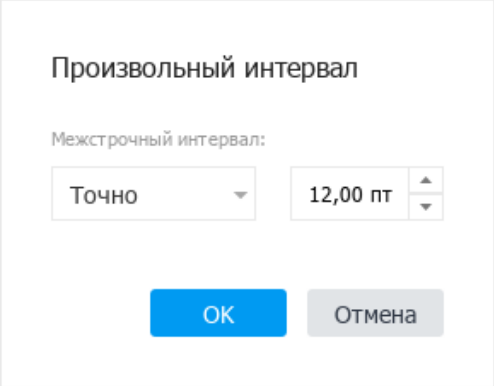
**Пример**

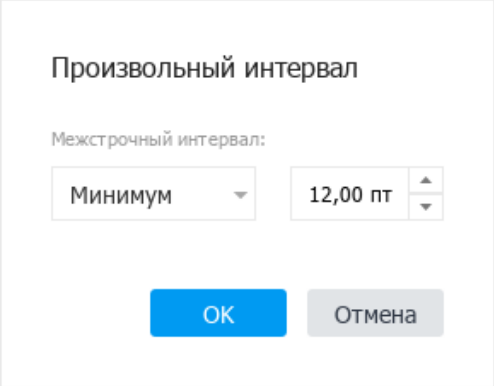
```
-- Конструктор
local lineSpacing = DocumentAPI.LineSpacing(1.5,
DocumentAPI.LineSpacingRule_Multiple)
-- Обращение к полям
lineSpacing.value = 1
lineSpacing.rule = DocumentAPI.LineSpacingRule_Exact
```

### 5.110 Таблица DocumentAPI.LineSpacingRule

В таблице 61 представлены варианты правил формирования межстрочного интервала текстового абзаца.

Таблица 61 – Виды межстрочного интервала

| Значение | Описание |
|---|---|
| `LineSpacingRule_Multiple` | Установка произвольного межстрочного интервала с использованием множителя.<br>При вызове необходимо указать значение множителя, например:<br>`pPr.lineSpacing = DocumentAPI.LineSpacing(1.15, DocumentAPI.LineSpacingRule_Multiple)`<br>В данном примере используется значение множителя **1.15**.<br>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «**Произвольный интервал**».<br>  |
| `LineSpacingRule_Exact` | Установка произвольного межстрочного интервала с использованием точного значения.<br>При вызове необходимо указать точное значение, например:<br>`pPr.lineSpacing = DocumentAPI.LineSpacing(12.0, DocumentAPI.LineSpacingRule_Exact)`<br>В данном примере используется точное значение **12.0**.<br>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «**Произвольный интервал**».<br>  |

| Значение | Описание |
|---|---|
| LineSpacingRule_AtLeast | Установка произвольного межстрочного интервала с использованием минимального значения.<br>При вызове необходимо указать минимальное значение, например:<br>`pPr.lineSpacing = DocumentAPI.LineSpacing(12.0, DocumentAPI.LineSpacingRule_AtLeast)`<br>В данном примере используется точное значение **12.0**.<br>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «**Произвольный интервал**».<br>  |

**Пример**

```
paragraph = document:getBlocks():getParagraph(0)
props = paragraph:getParagraphProperties()
props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
paragraph:setParagraphProperties(props)
```

### 5.111 Таблица DocumentAPI.LineStyle

В таблице 62 приведены типы линий. Используется в поле `style` таблицы DocumentAPI.LineProperties.

Таблица 62 – Типы линий

| Значение | Описание |
|---|---|
| DocumentAPI.LineStyle_NoLine | Нет линии |
| DocumentAPI.LineStyle_Solid |  |
| DocumentAPI.LineStyle_Dot |  |

| Значение | Описание |
|---|---|
| DocumentAPI.LineStyle_Dash |  |
| DocumentAPI.LineStyle_LongDash |  |
| DocumentAPI.LineStyle_DashDot |  |
| DocumentAPI.LineStyle_DotDotDash |  |
| DocumentAPI.LineStyle_Double |  |
| DocumentAPI.LineStyle_Wave |  |

**Пример**

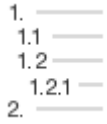
```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("C3")

lineProperties = DocumentAPI.LineProperties()
lineProperties.style = DocumentAPI.LineStyle_Wave

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

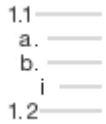
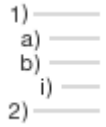
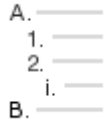
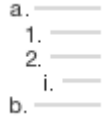
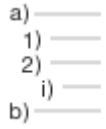
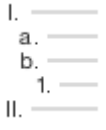
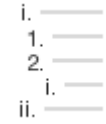
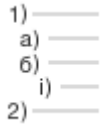
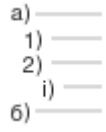
### 5.112 Таблица DocumentAPI.ListSchema

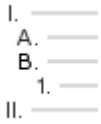
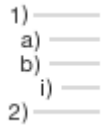
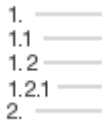
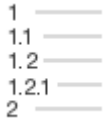
Типы схем форматирования списков, которые могут быть применены к абзацам текста, представлены в таблице 63. Данные константы используются в методах Paragraph:getListSchema(), Paragraph:setListSchema().

Таблица 63 – Типы схем форматирования списков

| Значение | Тип схемы списка | Изображение |
|---|---|---|
| DocumentAPI.ListSchema_ Unknown | Неизвестно | |

| Значение | Тип схемы списка | Изображение |
|---|---|---|
| `DocumentAPI.ListSchema_UnknownBullet` | Список без маркера | Соответствует варианту «нет» |
| `DocumentAPI.ListSchema_UnknownNumbering` | Нумерация без номера | Соответствует варианту «нет» |
| `DocumentAPI.ListSchema_BulletCircleSolid` | Список с маркерами в виде круга |  |
| `DocumentAPI.ListSchema_BulletCircleContour` | Список с маркерами в виде окружности |  |
| `DocumentAPI.ListSchema_BulletSquareSolid` | Список с маркерами в виде квадрата |  |
| `DocumentAPI.ListSchema_BulletDiamondDots` | Список с маркерами в виде четырех ромбов |  |
| `DocumentAPI.ListSchema_BulletHyphen` | Список с маркерами в виде дефиса |  |
| `DocumentAPI.ListSchema_BulletConcaveArrowSolid` | Список с маркерами в виде вогнутой стрелки |  |
| `DocumentAPI.ListSchema_BulletCheckmark` | Список с маркерами в виде галочки |  |
| `DocumentAPI.ListSchema_EnumeratorDecimalDot` | Десятичная нумерация с точкой |  |

| Значение | Тип схемы списка | Изображение |
|---|---|---|
| DocumentAPI.ListSchema_ EnumeratorDecimalDotMultiLevel | Многоуровневая десятичная нумерация с точкой |  |
| DocumentAPI.ListSchema_ EnumeratorDecimalBracket | Десятичная нумерация со скобкой |  |
| DocumentAPI.ListSchema_ EnumeratorLatinUppercaseDot | Нумерация латинскими прописными буквами с точкой |  |
| DocumentAPI.ListSchema_ EnumeratorLatinLowercaseDot | Нумерация латинскими строчными буквами с точкой |  |
| DocumentAPI.ListSchema_ EnumeratorLatinLowercaseBracket | Нумерация латинскими строчными буквами со скобкой |  |
| DocumentAPI.ListSchema_ EnumeratorRomanUppercaseDot | Нумерация римскими прописными цифрами с точкой |  |
| DocumentAPI.ListSchema_ EnumeratorRomanLowercaseDot | Нумерация римскими строчными цифрами с точкой |  |
| DocumentAPI.ListSchema_ EnumeratorDecimalRussianBracket | Десятичная нумерация через запятую со скобкой |  |
| DocumentAPI.ListSchema_ EnumeratorRussianLowercaseBracket | Нумерация с русскими строчными буквами со скобкой |  |

| Значение | Тип схемы списка | Изображение |
|---|---|---|
| `DocumentAPI.ListSchema_EnumeratorOutlineDefault` | Нумерация римскими прописными цифрами с точкой |  |
| `DocumentAPI.ListSchema_EnumeratorNumberValue` | Десятичная нумерация со скобкой, после третьего уровня вложенности – нумерация в скобках |  |
| `DocumentAPI.ListSchema_EnumeratorDecimalDotMultiLevelHeading` | Десятичная нумерация без левого отступа с точкой |  |
| `DocumentAPI.ListSchema_EnumeratorDecimalDotNoTrailingDotMultiLevelHeading` | Десятичная нумерация без левого отступа |  |

**Пример**

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

### 5.113 Таблица DocumentAPI.MediaObject

Таблица `DocumentAPI.MediaObject` представляет собой встроенный объект документа.

#### 5.113.1 Метод MediaObject:getFrame

Метод возвращает свойства позиции встроенного объекта. В зависимости от текущего редактора метод возвращает разные типы таблиц. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют таблицу `DocumentAPI.InlineFrame`, табличные документы работают с абсолютной позицией и используют таблицу `DocumentAPI.AbsoluteFrame`.

**Пример для текстового документа**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  print(mediaObject:getFrame()) -- <userdata of type
'CO::API::Document::InlineFrame'>
end
```

**Пример для табличного документа**

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  print(mediaObject:getFrame()) -- <userdata of type
'CO::API::Document::AbsoluteFrame'>
end
```

### 5.113.2 Метод MediaObject:toChart

Метод возвращает диаграмму `DocumentAPI.Chart`, связанную со встроенным объектом. Если объект не является диаграммой, метод возвращает `nil`.

Диаграммы реализованы только в табличных документах.

**Пример для табличного документа**

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  local chart = mediaObject:toChart()
  if chart ~= nil then
    print("Текущий объект является диаграммой")
  else
    print("Текущий объект является фигурой")
  end
end
```

### 5.113.3 Метод MediaObject:toImage

Метод возвращает изображение `DocumentAPI.Image`, связанное со встроенным объектом. Если объект не является изображением, метод возвращает `nil`.

**Пример для текстового документа**

```
for mediaObject in document:getRange():getInlineObjects():enumerate() do
  local image = mediaObject:toImage()
  if image then
    print("Текущий объект является изображением")
  else
    print("Текущий объект является фигурой")
  end
end
```

**Пример для табличного документа**

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if image ~= nil then
    print("Текущий объект является изображением")
  else
    print("Текущий объект является фигурой")
  end
end
```

### 5.114 Таблица DocumentAPI.MediaObjects

Таблица `DocumentAPI.MediaObjects` предназначен для доступа к коллекции графических объектов. Может быть получена вызовом методов `Table.getMediaObjects()` или `Range.getInlineObjects()` (см. Рисунок 47).

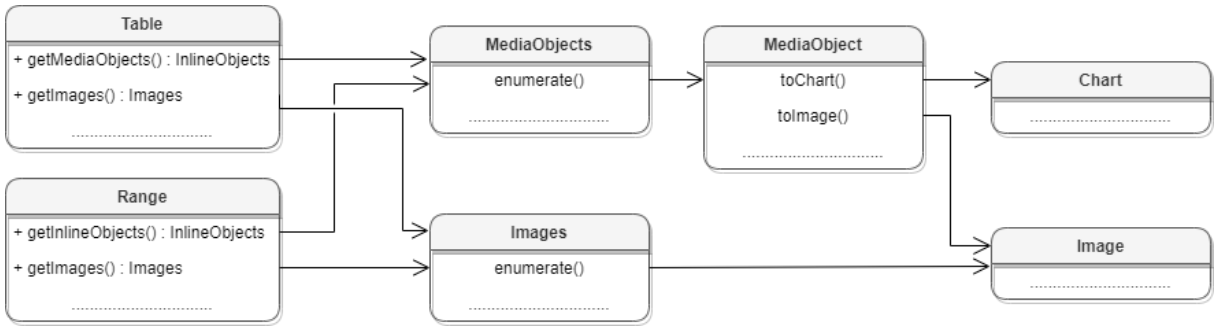



Рисунок 47 – Графические объекты

### 5.114.1 Метод MediaObjects:enumerate

Метод позволяет перечислить коллекцию встроенных объектов.

**Примеры для текстового документа**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  print("Встроенный объект:", mediaObject)
end
```

```
local mediaObjects = EditorAPI.getSelection():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  print("Встроенный объект:", mediaObject)
end
```

**Пример для табличного документа**

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if image ~= nil then
    print("Объект является изображением")
  else
    local chart = mediaObject:toChart()
    if chart ~= nil then
      print("Объект является диаграммой")
    else
      print("Объект является фигурой")
    end
  end
end
```

### 5.115 Таблица DocumentAPI.NamedExpression

Класс описывает структуру именованного диапазона.

**Пример**

```
local namedExpressions = sheet:getNamedExpressions()
for namedExpression in namedExpressions:enumerate() do
   print(namedExpression:getName())
   print(namedExpression:getExpression())
   cellRange = namedExpression:getCellRange()
```

```
   print(cellRange:getBeginRow(), cellRange:getLastRow())
end
```

### 5.115.1 Метод NamedExpression:getCellRange

Возвращает диапазон ячеек DocumentAPI.CellRange именованного диапазона. Пример см. в DocumentAPI.NamedExpression.

### 5.115.2 Метод NamedExpression:getExpression

Возвращает текст диапазона (формулы). Пример см. в DocumentAPI.NamedExpression.

### 5.115.3 Метод NamedExpression:getName

Возвращает имя именованного диапазона. Пример см. в DocumentAPI.NamedExpression.

### 5.115.4 Метод NamedExpression:setName

Метод позволяет переименовать именованный диапазон.

**Вызов**

```
setName(newName)
```

**Параметры**

– `newName`: новое имя диапазона, тип `string`.

**Пример**

```
local expressions = document:getNamedExpressions()
local expression = expressions:get("Prices")
expression:setName("Totals")
```

## 5.116 Таблица DocumentAPI.NamedExpressions

Таблица для представления списка именованных диапазонов. Может быть получена с помощью методов Document:getNamedExpressions(), Table:getNamedExpressions().

### 5.116.1 Метод NamedExpressions:addExpression

Добавляет новый диапазон в список именованных диапазонов.

**Пример**

```
local expressionName = "Покупки"
local expressionValue = "='Формула покупки'!$E$6:$E$14"
local namedExpressions = document:getNamedExpressions()
namedExpressions:addExpression(expressionName, expressionValue)
```

### 5.116.2 Метод NamedExpressions:enumerate

Позволяет получить доступ ко всему списку именованных диапазонов.

**Пример**

```
local namedExpressions = sheet:getNamedExpressions()
for namedExpression in namedExpressions:enumerate() do
   print(namedExpression)
end
```

### 5.116.3 Метод NamedExpressions:get

Возвращает именованный диапазон NamedExpression по имени `name`, если он существует.

**Пример**

```
local namedExpressions = document:getNamedExpressions()
local namedExpression = namedExpressions:get("Продажи")
if (namedExpression) then
    print(namedExpression:getName()) -- Продажи
else
    print("No named expression was found")
end
```

### 5.116.4 Метод NamedExpressions:removeExpression

Удаляет именованный диапазон по заданному имени.

**Пример**

```
local namedExpression = namedExpressions:get(expressionName)
if (namedExpression) then
```

```
    namedExpressions:removeExpression(expressionName)
end
```

### 5.117 Таблица DocumentAPI.NullaryConditionalFormatOperator

Таблица `NullaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил без параметров. Используется при создании правила условного форматирования (`ConditionalFormatRule`).

**Конструктор**

```
    NullaryConditionalFormatOperator(ConditionalFormatNullaryCondition
condition)
```

**Пример**

| Дата заказа |
|---|
| 10 января 2025 г. |
| 15 февраля 2025 г. |
| 1 марта 2025 г. |
| 20 апреля 2025 г. |
| 5 мая 2025 г. |
| 18 марта 2025 г. |
| 2 июня 2025 г. |
| 15 июля 2025 г. |
| 1 августа 2025 г. |
| 20 сентября 2025 г. |
| 1 октября 2025 г. |

Рисунок 48 – Пример создания правила для дат в прошлом месяце

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local nullaryStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
nullaryStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("B2:B12")
local cellRangePosition = cellRange:getTableRange()

local nullaryOperator =
DocumentAPI.createNullaryConditionalFormatOperator(DocumentAPI.ConditionalFormatN
ullaryCondition_LastMonth)
```

```
local nullaryRule = DocumentAPI.ConditionalFormatRule(nullaryOperator,
nullaryStyle, cellRangePosition, false)
rules:addRule(nullaryRule)
```

### 5.117.1 Метод NullaryConditionalFormatOperator:getCondition

Метод возвращает условие применения форматирования.

**Вызов**

```
ConditionalFormatNullaryCondition getCondition()
```

**Возвращает**

– условие применения форматирования, тип ConditionalFormatNullaryCondition.

### 5.117.2 Метод NullaryConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

## 5.118 Таблица DocumentAPI.NumberCellFormatting

Таблица содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода Cell:setFormat().

Таблица 64 – Описание полей таблицы DocumentAPI.NumberCellFormatting

| Поле | Тип | Описание |
|---|---|---|
| NumberCellFormatting.decimalPlaces | number | Количество десятичных позиций |
| NumberCellFormatting.useThousandsSeparator | bool | Использовать разделитель для тысячных |
| NumberCellFormatting.useRedForNegative | bool | Использовать красный цвет для отрицательных |

| Поле | Тип | Описание |
|---|---|---|
| | | значений |
| NumberCellFormatting.useBracketsForNegative | bool | Использовать скобки для отрицательных значений |
| NumberCellFormatting.hideSign | bool | Скрывать знак «минус» для отрицательных значений |

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A2")

local numberCellFormatting = DocumentAPI.NumberCellFormatting()
numberCellFormatting.decimalPlaces = 2
numberCellFormatting.useThousandsSeparator = true
numberCellFormatting.useRedForNegative = true
numberCellFormatting.useBracketsForNegative = true
numberCellFormatting.hideSign = false

cell:setFormat(numberCellFormatting)
print(cell:getFormattedValue())
```

### 5.119 Таблица DocumentAPI.PageFieldOrder

Таблица `DocumentAPI.PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем таблицы `DocumentAPI.PivotTableLayoutSettings`.

Таблица 65 – Виды отображения полей из области фильтров

| Значение | Описание |
|---|---|
| DocumentAPI.PageFieldOrder_DownThenOver | Вниз, затем поперек |
| DocumentAPI.PageFieldOrder_OverThenDown | Поперек, затем вниз |

### 5.120 Таблица DocumentAPI.PageOrientation

Таблица `PageOrientation` содержит варианты ориентации страницы документа. Может быть использована для получения / установки ориентации страниц для секции или документа в методах `Section:setPageOrientation()`, `Section:getPageOrientation()`.

Таблица 66 – Типы ориентации страницы

| Значение | Описание |
|---|---|
| `DocumentAPI.PageOrientation_Landscape` | Альбомная ориентация страницы |
| `DocumentAPI.PageOrientation_Portrait` | Книжная ориентация страницы |

**Примеры**

```
local section = document:getBlocks():getBlock(0):getSection()
section:setPageOrientation(DocumentAPI.PageOrientation_Landscape)
print(section:getPageOrientation())
```

```
local section = document:setPageOrientation(DocumentAPI.PageOrientation_Portrait)
local section = document:getBlocks():getBlock(0):getSection()
print(section:getPageOrientation())
```

### 5.121 Таблица DocumentAPI.PageParity

Варианты выбора страниц для экспорта и печати представлены в таблице 67. Используется в `DocumentAPI.PrintSettings`.

Таблица 67 – Варианты выбора страниц для экспорта и печати

| Значение | Описание |
|---|---|
| `DocumentAPI.PageParity_Odd` | Только нечетные страницы |
| `DocumentAPI.PageParity_Even` | Только четные страницы |
| `DocumentAPI.PageParity_All` | Все страницы |

### 5.122 Таблица DocumentAPI.PageProperties

Таблица `DocumentAPI.PageProperties` предоставляет такие свойства страницы как высота, ширина, размеры полей. Описание полей приведено в таблице 68. Используется в `Document.setPageProperties()`, `Section.getPageProperties()`, `Section.setPageProperties()`. Применяется только в текстовом документе.

Таблица 68 – Описание полей таблицы `DocumentAPI.PageProperties`

| Поле | Тип | Описание |
|---|---|---|
| `PageProperties.height` | `number` | Высота страницы |
| `PageProperties.width` | `number` | Ширина страницы |
| `PageProperties.margins` | `Insets` | Поля страницы |

**Примеры**

```
local section = document:getBlocks():getBlock(0):getSection()
local pageProperties = section:getPageProperties()
pageProperties.width = 100
pageProperties.height = 200
pageProperties.margins.left = 10
section:setPageProperties(pageProperties)
```

```
local pageProperties = DocumentAPI.PageProperties()
pageProperties.width = 100
pageProperties.height = 200
document:setPageProperties(pageProperties)
```

```
local pageProperties = DocumentAPI.PageProperties(100, 200)
document:setPageProperties(pageProperties)
```

### 5.122.1 Метод PageProperties:__eq

Метод позволяет использовать оператор сравнения `__eq` для определения эквивалентности содержимого двух структур `DocumentAPI.PageProperties`.

**Пример**

```
local pageProperties1 = DocumentAPI.PageProperties(100, 200)
document:setPageProperties(pageProperties1)

local pageProperties2 = DocumentAPI.PageProperties(100, 200)
document:setPageProperties(pageProperties2)

print(pageProperties1:__eq(pageProperties2)) -- true
```

### 5.123 Таблица DocumentAPI.Paragraph

Таблица `DocumentAPI.Paragraph` предоставляет доступ к свойствам абзаца (см. Рисунок 49).

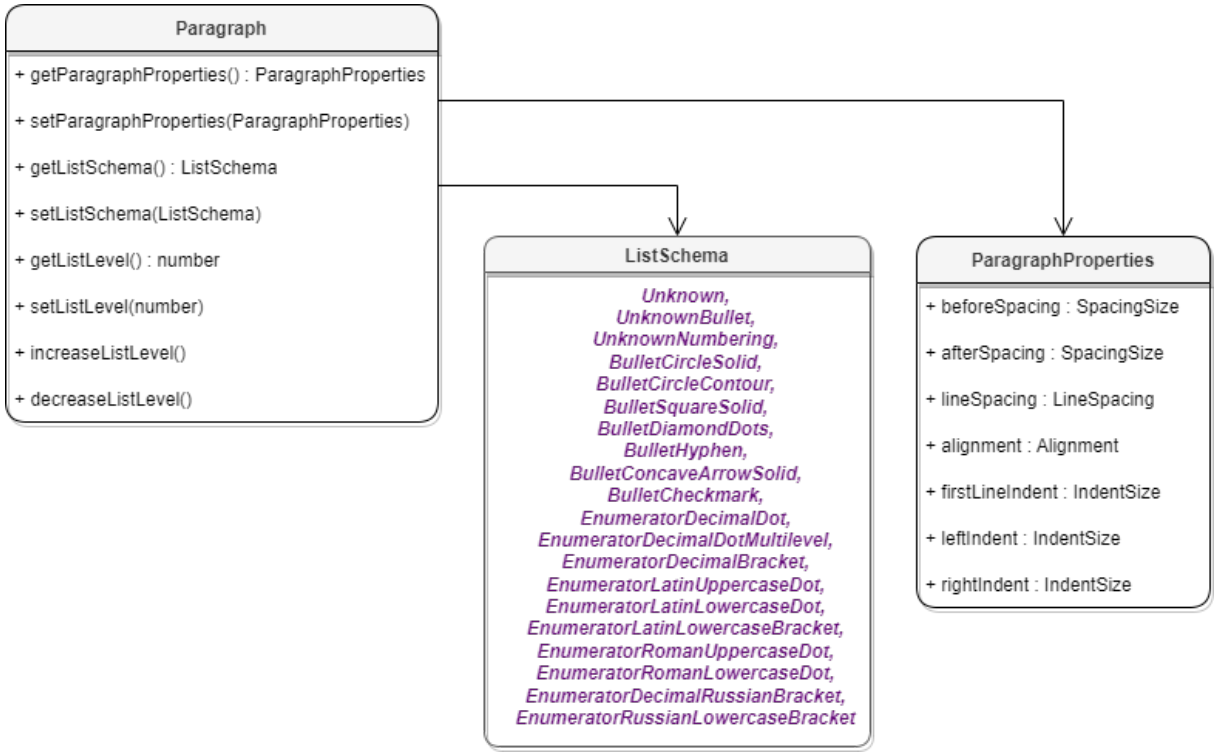



Рисунок 49 – Объектная модель таблиц для работы со свойствами абзаца

#### 5.123.1 Метод Paragraph:decreaseListLevel

Метод позволяет уменьшить на единицу глубину вложенности элемента списка. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:decreaseListLevel()
```

### 5.123.2 Метод Paragraph:getListLevel

Метод позволяет получить глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraph = document:getBlocks():getParagraph(0)
local level = paragraph:getListLevel()
```

### 5.123.3 Метод Paragraph:getListSchema

Метод возвращает схему форматирования абзаца DocumentAPI.ListSchema либо значение `nil`, если схема нумерации не установлена для абзаца. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraph = document:getBlocks():getParagraph(0)
local schema = paragraph:getListSchema()
```

### 5.123.4 Метод Paragraph:getParagraphProperties

Метод предоставляет доступ к таблице свойств форматирования абзаца DocumentAPI.ParagraphProperties, таким как выравнивание текста, межстрочные интервалы, отступы и т. д.

**Пример для текстового документа**

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
print(para_props.afterSpacing)
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
```

```
    print(para_props.afterSpacing)
end
```

### 5.123.5 Метод Paragraph:getResolvedStyle

Метод возвращает комбинированный стиль текущего абзаца. Комбинированным стилем является стиль всего абзаца, общий стиль всех его частей или стиль по умолчанию. Данный стиль отображается в интерфейсе редактора.

**Вызов**

```
TextStyle getResolvedStyle()
```

**Возвращает**

– комбинированный стиль текущего абзаца, тип TextStyle.

**Пример**

```
local paragraph = document:getRange():getBegin():getParagraph()
print(paragraph:getResolvedStyle():getName())
```

### 5.123.6 Метод Paragraph:getStyle

Метод возвращает стиль текущего абзаца.

**Вызов**

```
TextStyle getStyle()
```

**Возвращает**

– стиль текущего абзаца, тип TextStyle.

**Пример**

```
local paragraphs = document:getRange():getParagraphs()
for paragraph in paragraphs:enumerate() do
    print(paragraph:getStyle():getName())
end
```

### 5.123.7 Метод Paragraph:increaseListLevel

Метод позволяет увеличить на единицу глубину вложенности элемента списка. В случае, если максимальный уровень уже установлен, увеличения не происходит. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:increaseListLevel()
```

### 5.123.8 Метод Paragraph:setListLevel

Метод позволяет установить глубину вложенности элемента списка.

Значение может быть равным `nil`, если схема нумерации не установлена для абзаца. В этом случае будет установлено минимальное значение. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraph = document:getBlocks():getParagraph(0)
local level = paragraph:setListLevel(1)
```

### 5.123.9 Метод Paragraph:setListSchema

Метод позволяет установить тип маркированного или нумерованного списка `DocumentAPI.ListSchema`. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

### 5.123.10 Метод Paragraph:setParagraphProperties

Метод предназначен для обновления таблицы свойств форматирования абзаца `DocumentAPI.ParagraphProperties`.

**Пример для текстового документа**

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
para_props.alignment = DocumentAPI.Alignment_Right
para:setParagraphProperties(para_props)
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    para_props.alignment = DocumentAPI.Alignment_Right
    para:setParagraphProperties(para_props)
end
```

### 5.123.11 Метод Paragraph:setStyle

Метод задает стиль абзаца.

**Вызов**

```
setStyle(textStyle)
```

**Параметры**

– `textStyle`: стиль абзаца, тип `TextStyle`.

**Пример**

```
local styles = document:getTextStyles()
local normalStyle = styles:get(DocumentAPI.PredefinedTextStyle_Normal)
local paragraphs = document:getRange():getParagraphs()
for paragraph in paragraphs:enumerate() do
    paragraph:setStyle(normalStyle)
end
```

## 5.124 Таблица DocumentAPI.ParagraphProperties

Таблица `DocumentAPI.ParagraphProperties` предназначена для управления свойствами форматирования абзаца (см. Рисунок 50). Таблица `DocumentAPI.ParagraphProperties` используется в методах `Paragraph:getParagraphProperties()`, `Paragraph:setParagraphProperties()`, `Cell:getParagraphProperties()`, `Cell:setParagraphProperties()`, `CellRange:getParagraphProperties()` и `CellRange:setParagraphProperties()`.

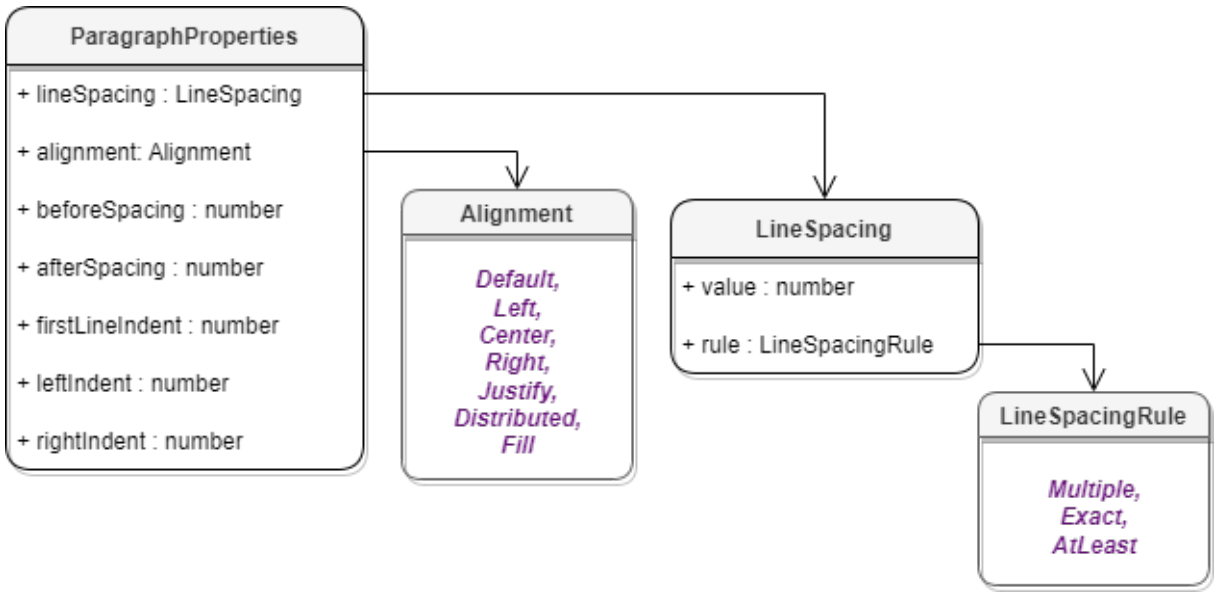
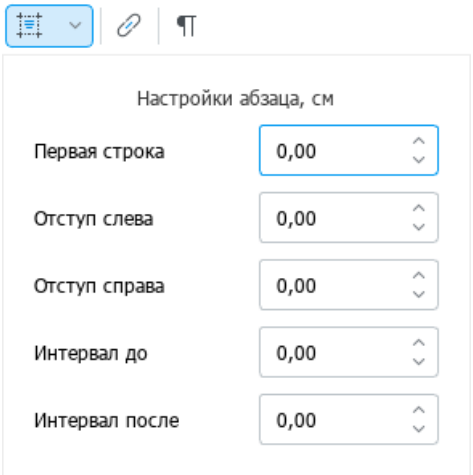



Рисунок 50 – Объектная модель таблиц для работы со свойствами абзаца

Описание полей таблицы `DocumentAPI.ParagraphProperties` представлено в таблице 69.

Таблица 69 – Описание полей таблицы `DocumentAPI.ParagraphProperties`

| Поле | Тип | Описание |
|---|---|---|
| `ParagraphProperties.beforeSpacing` | `number` | Установка величины расстояния до абзаца.<br>При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца,** (см. рисунок ниже), в поле **Интервал до**.<br>  |
| `ParagraphProperties.afterSpacing` | `number` | Установка величины расстояния после абзаца.<br>При работе с пользовательским интерфейсом приложения соответствует значению, указанному в |

| Поле | Тип | Описание |
|---|---|---|
| | | диалоговом окне **Настройки абзаца**, в поле **Интервал после**. |
| ParagraphProperties.lineSpacing | LineSpacing | Расстояние между строк одного абзаца (межстрочный интервал). |
| ParagraphProperties.alignment | Alignment | Выравнивание текстового фрагмента по горизонтали. |
| ParagraphProperties.firstLineIndent | number | Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева.<br>При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Первая строка**. |
| ParagraphProperties.leftIndent | number | Расстояние от левого поля документа до абзаца (отступ слева).<br>При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ слева**. |
| ParagraphProperties.rightIndent | number | Расстояние от правого поля документа до абзаца.<br>При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ справа**. |

**Пример для текстового документа**

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
--
para_props.afterSpacing = 28.3 -- значение соответствует 1 см
para_props.beforeSpacing = 28.3 -- значение соответствует 1 см
para_props.alignment = DocumentAPI.Alignment_Center
para_props.firstLineIndent = 28.3 -- значение соответствует 1 см
para_props.leftIndent = 28.3 -- значение соответствует 1см
para_props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
para_props.rightIndent = 28.3 -- значение соответствует 1см
--
para:setParagraphProperties(para_props)
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
   local para_props = para:getParagraphProperties()
   para_props.afterSpacing = 28.3 -- значение соответствует 1 см
   para_props.beforeSpacing = 28.3 -- значение соответствует 1 см
   para_props.alignment = DocumentAPI.Alignment_Center
   para_props.firstLineIndent = 28.3 -- значение соответствует 1 см
   para_props.leftIndent = 28.3 -- значение соответствует 1см
   para_props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
   para_props.rightIndent = 28.3 -- значение соответствует 1см
   para:setParagraphProperties(para_props)
end
```

### 5.125 Таблица DocumentAPI.Paragraphs

Таблица `DocumentAPI.Paragraphs` предоставляет доступ к коллекции абзацев типа `DocumentAPI.Paragraph` (см. Рисунок 51). Коллекция абзацев может быть получена из таблицы `DocumentAPI.Range` посредством использования вызова `Range:getParagraphs()`.

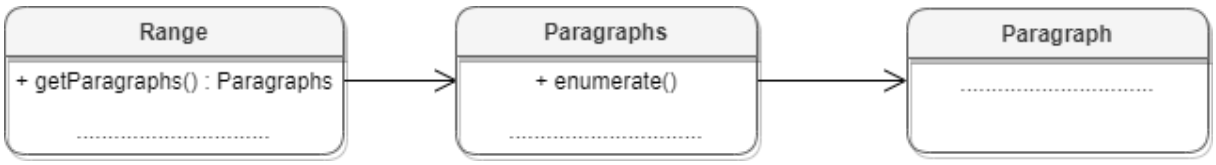



Рисунок 51 – Объектная модель для работы со списком абзацев

**Пример для текстового документа**

```
local paragraphs = document:getRange():getParagraphs()
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local paragraphs = cell:getRange():getParagraphs()
```

### 5.125.1 Метод Paragraphs:decreaseListLevel

Метод уменьшает уровень списка на единицу. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:decreaseListLevel()
```

### 5.125.2 Метод Paragraphs:enumerate

Метод позволяет перечислить коллекцию абзацев.

**Пример для текстового документа**

```
local paragraphs = document:getRange():getParagraphs()
for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    print(para_props.alignment)
end
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    print(para_props.alignment)
end
```

### 5.125.3 Метод Paragraphs:increaseListLevel

Метод увеличивает уровень списка на единицу. В случае, если максимальный уровень уже установлен, увеличения не происходит. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:increaseListLevel()
```

### 5.125.4 Метод Paragraphs:setListLevel

Метод устанавливает глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:setListLevel(1)
```

### 5.125.5 Метод Paragraphs:setListSchema

Метод устанавливает тип маркированного или нумерованного списка `DocumentAPI.ListSchema`. Данный метод используется только в текстовом документе.

**Пример**

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

## 5.126 Таблица DocumentAPI.PercentageCellFormatting

Содержит параметр для процентного формата ячеек таблицы, используется в качестве аргумента метода `Cell:setFormat()`.

Таблица 70 – Описание полей таблицы `DocumentAPI.PercentageCellFormatting`

| Поле | Тип | Описание |
|---|---|---|
| `PercentageCellFormatting.decimalPlaces` | `number` | Количество десятичных позиций |

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")

local percentageCellFormatting = DocumentAPI.PercentageCellFormatting()
percentageCellFormatting.decimalPlaces = 2

cell:setFormat(percentageCellFormatting)
print(cell:getFormattedValue())
```

### 5.127 Таблица DocumentAPI.PivotTable

Таблица для представления сводной таблицы. Может быть получена из ячейки `Cell.getPivotTable()`, или при создании новой сводной таблицы `PivotTablesManager.create()`.

#### 5.127.1 Метод PivotTable:areAllFiltersInDefaultState

Метод позволяет определить, применен ли к сводной таблице хоть один фильтр.

**Вызов**

```
bool areAllFiltersInDefaultState()
```

**Возвращает**

– `true`, если к сводной таблице не применен ни один фильтр, в ином случае – `false`.

#### 5.127.2 Метод PivotTable:changeSourceRange

Метод позволяет задать новый диапазон исходных данных сводной таблицы без обновления самой таблицы. Параметр `sourceRange` – строка, представляющая новый диапазон таблицы.

**Пример**

```
pivotTable:changeSourceRange("I3:K5")
local cellRange = pivotTable:getSourceRange()
print(cellRange:getBeginRow(), cellRange:getLastRow())
```

### 5.127.3 Метод PivotTable:createPivotTableEditor

Метод возвращает объект DocumentAPI.PivotTableEditor, который служит для обновления свойств и редактирования сводной таблицы.

**Пример**

```
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local pivotTableEditor = pivotTable:createPivotTableEditor()
```

### 5.127.4 Метод PivotTable:getColumnFields

Метод возвращает список полей DocumentAPI.PivotTableCategoryField из области колонок.

**Пример**

```
local columnFields = pivotTable:getColumnFields()
for fieldIdx = 0, columnFields:size() - 1 do
   print(columnFields[fieldIdx].fieldProperties.fieldName)
end
```

### 5.127.5 Метод PivotTable:getConditionalLabelFilter

Метод возвращает примененный условный фильтр по подписи.

**Вызов**

```
PivotTableConditionalLabelFilter getConditionalLabelFilter(fieldName)
```

**Параметры**

– `fieldName`: название поля в сводной таблице, тип `string`.

**Возвращает**

– условный фильтр по подписи, тип PivotTableConditionalLabelFilter.

– `nil`, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

**Пример**

```
local labelOperation = DocumentAPI.PivotTableConditionalLabelFilterOperation()
labelOperation.operationType =
DocumentAPI.PivotTableConditionalLabelFilterOperationType_Contains
labelOperation.operand = "to"
```

```
local labelFilter = pivotTable:getConditionalLabelFilter("Product Name")
labelFilter:setOperation(labelOperation)

tableEditor:setFilter(labelFilter):apply()
```

### 5.127.6 Метод PivotTable:getConditionalValueFilter

Метод возвращает примененный условный фильтр по значению.

**Вызов**

```
PivotTableConditionalValueFilter getConditionalValueFilter(fieldName)
```

**Параметры**

– `fieldName`: название поля в сводной таблице, тип `string`.

**Возвращает**

– условный фильтр по значению, тип `PivotTableConditionalValueFilter`.

– `nil`, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

**Пример**

```
local valueOperation =
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.Pivo
tTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

local valueFilter = pivotTable:getConditionalValueFilter("Product Name")
valueFilter:setOperation(valueOperation)

tableEditor:setFilter(valueFilter):apply()
```

### 5.127.7 Метод PivotTable:getFieldCategories

Метод возвращает список категорий `DocumentAPI.PivotTableFieldCategories`, содержащих заданное поле `fieldName`.

**Пример**

```
local fieldCategories = pivotTable:getFieldCategories("Age")
```

### 5.127.8 Метод PivotTable:getFieldItems

Метод возвращает все элементы DocumentAPI.PivotTableItems сводной таблицы по заданному имени поля fieldName.

**Пример**

```
local pivotTableItems = pivotTable:getFieldItems("Age")
print(pivotTableItems)
```

### 5.127.9 Метод PivotTable:getFieldItemsByName

Метод возвращает все элементы DocumentAPI.PivotTableItems из заданного поля fieldName по имени itemName.

**Пример**

```
local pivotTableItemsByName = pivotTable:getFieldItemsByName("Ultimate Question
of Life", "42")
print(pivotTableItemsByName)
```

### 5.127.10 Метод PivotTable:getFieldsList

Метод возвращает список DocumentAPI.PivotTableField всех полей сводной таблицы.

**Пример**

```
local fieldsList = pivotTable:getFieldsList()
print(fieldsList:size())
for fieldIdx = 0, fieldsList:size() - 1 do
   print(fieldsList[fieldIdx].fieldProperties.fieldName)
end
```

### 5.127.11 Метод PivotTable:getFilter

Метод возвращает фильтр DocumentAPI.PivotTableFilter по заданному имени поля fieldName.

**Пример**

```
local filter = pivotTable:getFilter("Age")
print(filter:getFieldName())
```

### 5.127.12 Метод PivotTable:getFilters

Метод возвращает список фильтров DocumentAPI.PivotTableFilter сводной таблицы.

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   -- use filter
end
```

### 5.127.13 Метод PivotTable:getPageFields

Метод возвращает список полей DocumentAPI.PivotTablePageField из области фильтров.

**Пример**

```
local pageFields = pivotTable:getPageFields()
print(pageFields:size())
```

### 5.127.14 Метод PivotTable:getPivotRange

Метод возвращает диапазон ячеек DocumentAPI.CellRange, в котором размещена сводная таблица.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local cellRange = pivotTable:getPivotRange()
print(cellRange:getBeginRow(), cellRange:getLastRow()) -- 7  10
```

### 5.127.15 Метод PivotTable:getPivotTableCaptions

Метод возвращает информацию DocumentAPI.PivotTableCaptions о всех заголовках сводной таблицы.

**Пример**

```
local pivotTableCaptions = pivotTable:getPivotTableCaptions()
print(pivotTableCaptions.grandTotalCaption)
```

```
print(pivotTableCaptions.valuesHeaderCaption)
print(pivotTableCaptions.rowHeaderCaption)
print(pivotTableCaptions.columnHeaderCaption)
print(pivotTableCaptions.errorCaption)
print(pivotTableCaptions.emptyCaption)
```

### 5.127.16 Метод PivotTable:getPivotTableLayoutSettings

Метод возвращает настройки отображения DocumentAPI.PivotTableLayoutSettings сводной таблицы.

**Пример**

```
local settings = pivotTable:getPivotTableLayoutSettings()
print(settings.reportLayout)
print(settings.valueFieldsOrientation)
print(settings.pageFieldOrder)
print(settings.indentForCompactLayout)
print(settings.pageFieldWrapCount)
```

### 5.127.17 Метод PivotTable:getRowFields

Метод возвращает список полей DocumentAPI.PivotTableCategoryField из области строк.

**Пример**

```
local rowFields = pivotTable:getRowFields()
for fieldIdx = 0, rowFields:size() - 1 do
   print(rowFields[fieldIdx].fieldProperties.fieldName)
end
```

### 5.127.18 Метод PivotTable:getSortingParams

Метод возвращает параметры сортировки, примененные к заданному полю сводной таблицы.

**Вызов**

```
PivotTableSortingParams getSortingParams(fieldName)
```

**Параметры**

– `fieldName`: название поля таблицы, тип `string`.

**Возвращает**

– параметры сортировки, тип PivotTableSortingParams.

– nil, если параметры сортировки не удалось получить.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("L1")
local pivotTable = cell:getPivotTable()
local sorting = pivotTable:getSortingParams("Product")
print(sorting.sortingType)
print(sorting.sortingOrder)
```

### 5.127.19 Метод PivotTable:getSourceRange

Метод возвращает диапазон DocumentAPI.CellRange исходных данных сводной таблицы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local cellRange = pivotTable:getSourceRange()
print(cellRange:getBeginRow(), cellRange:getLastRow()) -- 2  6
```

### 5.127.20 Метод PivotTable:getSourceRangeAddress

Метод возвращает текстовое представление диапазона исходных данных сводной таблицы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
if (pivotTable) then
  print(pivotTable:getSourceRangeAddress()) -- 'Sheet1'!I3:K7
end
```

### 5.127.21 Метод PivotTable:getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства, которые представлены в сводной таблице.

**Вызов**

```
PivotTableUnsupportedFeatures getUnsupportedFeatures()
```

**Возвращает**

– коллекция значений PivotTableUnsupportedFeature.

**Пример**

```
local unsupportedFeatures = pivotTable:getUnsupportedFeatures()
for featureIndex = 0, unsupportedFeatures:size() - 1 do
   print(unsupportedFeatures[featureIndex])
end
```

### 5.127.22 Метод PivotTable:getValueFields

Метод возвращает список полей DocumentAPI.PivotTableValueField из области значений.

**Пример**

```
local valueFields = pivotTable:getValueFields()
for fieldIdx = 0, valueFields:size() - 1 do
   print(valueFields[fieldIdx].baseFieldName)
   print(valueFields[fieldIdx].valueFieldName)
   print(valueFields[fieldIdx].cellNumberFormat)
   print(valueFields[fieldIdx].totalFunction)
end
```

### 5.127.23 Метод PivotTable:getViewDetailsEnabled

Метод позволяет определить включена ли возможность просмотра детализации данных для значений текущей сводной таблицы.

**Вызов**

```
bool getViewDetailsEnabled()
```

**Возвращает**

– `true`, если включена возможность просмотра детализации данных, в ином случае – `false`.

### 5.127.24 Метод PivotTable:isColumnGrandTotalEnabled

Метод возвращает `true`, если разрешено показывать общие итоги для столбцов.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A3")
local pivotTable = cell:getPivotTable()
print(pivotTable:isColumnGrandTotalEnabled())
```

### 5.127.25 Метод PivotTable:isRowGrandTotalEnabled

Метод возвращает `true`, если разрешено показывать общие итоги для строк.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A3")
local pivotTable = cell:getPivotTable()
print(pivotTable:isRowGrandTotalEnabled())
```

### 5.127.26 Метод PivotTable:remove

Метод удаляет сводную таблицу.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
if (pivotTable) then
   pivotTable:remove()
end
```

### 5.127.27 Метод PivotTable:update

Метод обновляет и полностью пересчитывает сводную таблицу, возвращает `DocumentAPI.PivotTableUpdateResult`.

**Пример**

```
local updateResult = pivotTable:update()
if (updateResult ~= DocumentAPI.PivotTableUpdateResult_Success) then
```

```
   print(updateResult)
end
```

### 5.128 Таблица DocumentAPI.PivotTableBaseItemPosition

Таблица `PivotTableBaseItemPosition` содержит типы сравнения значений для дополнительных вычислений. Используется в `PivotTableCalculationDataBaseItem`.

Таблица 71 – Описание типов сравнения значений

| Значение | Описание |
|---|---|
| `PivotTableBaseItemPosition_Previous` | Каждое значение сравнивается с предыдущим |
| `PivotTableBaseItemPosition_Next` | Каждое значение сравнивается со следующим |

### 5.129 Таблица DocumentAPI.PivotTableCalculationData

Таблица `PivotTableCalculationData` представляет собой настройки дополнительных вычислений для поля значений сводной таблицы. Используется в методе `PivotTableEditor:setAdditionalCalculations()` и поле `PivotTableValueField.calculationData`.

**Конструкторы**

```
PivotTableCalculationData()
```

```
PivotTableCalculationData(PivotTableDataCalculationType calculationType)
```

```
PivotTableCalculationData(PivotTableDataCalculationType calculationType, PivotTableCalculationDataBaseEntity base)
```

Таблица 72 – Описание полей таблицы `PivotTableCalculationData`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableCalculationData.calculationType` | `PivotTableDataCalculationType` | Тип вычисления |
| `PivotTableCalculationData.baseEntity` | `PivotTableCalculationDataBaseEntity` | Данные для вычисления |

**Пример дополнительных вычислений типа "% от общего итога"**

```
local sheet = document:getBlocks():getTable(0)
local pivotTable = sheet:getCell("J8"):getPivotTable()
local tableEditor = pivotTable:createPivotTableEditor()

local data = DocumentAPI.PivotTableCalculationData()
```

```
data.calculationType = DocumentAPI.PivotTableDataCalculationType_PercentOfTotal

tableEditor:setAdditionalCalculations("Sum of Total", data)
tableEditor:apply()
```

**Пример дополнительных вычислений типа "% от родительского итога" поля Product Name**

```
local data = DocumentAPI.PivotTableCalculationData()
data.calculationType = DocumentAPI.PivotTableDataCalculationType_PercentOfParent
data.baseEntity = DocumentAPI.PivotTableCalculationDataBaseEntity("Product Name")
```

**Пример дополнительных вычислений типа "Отличие" от предыдущего значения в поле Country**

```
local data = DocumentAPI.PivotTableCalculationData()
data.calculationType = DocumentAPI.PivotTableDataCalculationType_Difference
local baseItem =
DocumentAPI.PivotTableCalculationDataBaseItem(DocumentAPI.PivotTableBaseItemPosit
ion_Previous)
data.baseEntity = DocumentAPI.PivotTableCalculationDataBaseEntity("Country",
baseItem)
```

### 5.130 Таблица DocumentAPI.PivotTableCalculationDataBaseEntity

Таблица `PivotTableCalculationDataBaseEntity` представляет собой настройки данных для задания дополнительных вычислений. Используется в поле `PivotTableCalculationData.baseEntity`.

**Конструкторы**

```
PivotTableCalculationDataBaseEntity()
```

```
PivotTableCalculationDataBaseEntity(string field)
```

```
PivotTableCalculationDataBaseEntity(string field,
PivotTableCalculationDataBaseItem item)
```

Таблица 73 – Описание полей таблицы `PivotTableCalculationDataBaseEntity`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableCalculationDataBaseEntity.baseField` | `string` | Базовое поле |

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableCalculationDataBaseEntity.baseItem` | `PivotTableCalculationDataBaseItem` | Базовое значение |

### 5.131 Таблица DocumentAPI.PivotTableCalculationDataBaseItem

Таблица `PivotTableCalculationDataBaseItem` позволяет задать значение поля для сравнения при дополнительных вычислениях (определенное значение или предыдущее/следующее значение). Используется в поле `PivotTableCalculationDataBaseEntity.baseItem`.

**Конструкторы**

```
PivotTableCalculationDataBaseItem(PivotTableItem baseItem)
```

```
PivotTableCalculationDataBaseItem(PivotTableBaseItemPosition baseItemPosition)
```

#### 5.131.1 Метод PivotTableCalculationDataBaseItem:getItem

Метод возвращает текущее значение для сравнения при дополнительных вычислениях.

**Вызов**

```
PivotTableItem getItem()
```

**Возвращает**

– элемент сводной таблицы, тип `PivotTableItem`.

– `nil`, если вместо значения задано сравнение.

#### 5.131.2 Метод PivotTableCalculationDataBaseItem:getPosition

Метод возвращает текущий тип сравнения значений для дополнительных вычислений.

**Вызов**

```
PivotTableBaseItemPosition getPosition()
```

**Возвращает**

– тип сравнения значений, тип PivotTableBaseItemPosition.

– nil, если вместо сравнения задано конкретное значение.

### 5.132 Таблица DocumentAPI.PivotTableCaptions

Таблица DocumentAPI.PivotTableCaptions хранит все пользовательские заголовки сводной таблицы (см. PivotTable:getPivotTableCaptions() и PivotTableEditor:setCaptions()).

Таблица 74 – Описание полей таблицы DocumentAPI.PivotTableCaptions

| Поле | Тип | Описание |
|---|---|---|
| PivotTableCaptions.errorCaption | string | Подпись для значений, которые возвращают ошибку |
| PivotTableCaptions.emptyCaption | string | Подпись для значений, которые возвращают пустое значение |
| PivotTableCaptions.grandTotalCaption | string | Подпись общих итогов |
| PivotTableCaptions.valuesHeaderCaption | string | Подпись поля из области значений; это поле отображается в отчете в случае, если в сводной таблице находится более двух полей из области значений, и макет имеет тип ‘табличный’ или ‘структурный’ |
| PivotTableCaptions.rowHeaderCaption | string | Подпись заголовка строк (видна только при включенном компактном макете, это макет по умолчанию) |
| PivotTableCaptions.columnHeaderCaption | string | Подпись заголовка колонок (видна только при включенном компактном макете, это макет по умолчанию) |

### 5.133 Таблица DocumentAPI.PivotTableCategoryField

`DocumentAPI.PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. таблицу 75). Таблица может быть получена посредством вызовов `PivotTable:getRowFields()`, `PivotTable:getColumnFields()`.

Таблица 75 – Описание полей таблицы `DocumentAPI.PivotTableCategoryField`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableCategoryField.fieldProperties` | `PivotTableFieldProperties` | Свойства поля |
| `PivotTableCategoryField.subtotalFunctions` | Коллекция объектов `PivotTableFunction` | Список функций для вычисления подытога |

### 5.134 Таблица DocumentAPI.PivotTableConditionalLabelFilter

Таблица `PivotTableConditionalLabelFilter` представляет собой условный фильтр по подписи. Фильтрация производится по строковым значениям, которые содержит поле сводной таблицы. Используется в методах `PivotTable:getConditionalLabelFilter()` и `PivotTableEditor:setFilter()`.

#### 5.134.1 Метод PivotTableConditionalLabelFilter:getFieldName

Метод возвращает имя поля, с которым ассоциирован фильтр.

**Вызов**

```
string getFieldName()
```

**Возвращает**

– имя поля, тип `string`.

#### 5.134.2 Метод PivotTableConditionalLabelFilter:getOperation

Метод возвращает текущие настройки фильтра.

**Вызов**

```
PivotTableConditionalLabelFilterOperation getOperation()
```

**Возвращает**

– настройки фильтра, тип PivotTableConditionalLabelFilterOperation.

### 5.134.3 Метод PivotTableConditionalLabelFilter:isInDefaultState

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

**Вызов**

```
bool isInDefaultState()
```

**Возвращает**

– `true`, если фильтр не содержит настроек, в ином случае – `false`.

### 5.134.4 Метод PivotTableConditionalLabelFilter:reset

Метод сбрасывает настройки текущего фильтра.

**Вызов**

```
reset()
```

### 5.134.5 Метод PivotTableConditionalLabelFilter:setOperation

Метод устанавливает настройки фильтра.

**Вызов**

```
setOperation(operation)
```

**Параметры**

– `operation`: настройки фильтра, тип PivotTableConditionalLabelFilterOperation.

**Пример**

```
local labelOperation = DocumentAPI.PivotTableConditionalLabelFilterOperation()
labelOperation.operationType =
DocumentAPI.PivotTableConditionalLabelFilterOperationType_Contains
labelOperation.operand = "to"

local labelFilter = pivotTable:getConditionalLabelFilter("Product Name")
labelFilter:setOperation(labelOperation)

tableEditor:setFilter(labelFilter):apply()
```

### 5.135 Таблица DocumentAPI.PivotTableConditionalLabelFilterOperation

Таблица `PivotTableConditionalLabelFilterOperation` представляет собой настройки условного фильтра по подписи. Используется в методах `PivotTableConditionalLabelFilter:getOperation()` и `PivotTableConditionalLabelFilter:setOperation()`.

Таблица 76 – Описание полей таблицы `PivotTableConditionalLabelFilterOperation`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableConditionalLabelFilterOperation.operationType` | `PivotTableConditionalLabelFilterOperationType` | Тип операции сравнения |
| `PivotTableConditionalLabelFilterOperation.operand` | `string` | Значение для сравнения |

**Пример**

```
local labelOperation = DocumentAPI.PivotTableConditionalLabelFilterOperation()
labelOperation.operationType =
DocumentAPI.PivotTableConditionalLabelFilterOperationType_Contains
labelOperation.operand = "to"

local labelFilter = pivotTable:getConditionalLabelFilter("Product Name")
labelFilter:setOperation(labelOperation)

tableEditor:setFilter(labelFilter):apply()
```

### 5.136 Таблица DocumentAPI.PivotTableConditionalLabelFilterOperationType

Таблица `PivotTableConditionalLabelFilterOperationType` определяет тип операции сравнения, применяемый к фильтру по подписи. Используется в поле `PivotTableConditionalLabelFilterOperation.operationType`.

Таблица 77 – Описание типов операций сравнения

| Значение | Описание |
|---|---|
| `PivotTableConditionalLabelFilterOperationType_Equal` | Равные |
| `PivotTableConditionalLabelFilterOperationType_NotEqual` | Не равные |

| Значение | Описание |
|---|---|
| `PivotTableConditionalLabelFilterOperationType_BeginsWith` | Начинаются с |
| `PivotTableConditionalLabelFilterOperationType_NotBeginsWith` | Не начинается с |
| `PivotTableConditionalLabelFilterOperationType_EndsWith` | Заканчиваются на |
| `PivotTableConditionalLabelFilterOperationType_NotEndsWith` | Не заканчиваются на |
| `PivotTableConditionalLabelFilterOperationType_Contains` | Содержат |
| `PivotTableConditionalLabelFilterOperationType_NotContains` | Не содержат |

### 5.137 Таблица DocumentAPI.PivotTableConditionalValueFilter

Таблица `PivotTableConditionalValueFilter` представляет собой условный фильтр по значению. Фильтрация производится по значениям, которые соответствуют полю в строке или столбце сводной таблицы. Используется в методах `PivotTable:getConditionalValueFilter()` и `PivotTableEditor:setFilter()`.

#### 5.137.1 Метод PivotTableConditionalValueFilter:getDefaultOperation

Метод возвращает настройки условного фильтра соответствующие заданной операции сравнения.

**Вызов**

```
static PivotTableConditionalValueFilterOperation getDefaultOperation(operationType)
```

**Параметры**

– `operationType`: тип операции сравнения, тип `PivotTableConditionalValueFilterOperationType`.

**Возвращает**

– настройки условного фильтра по значению, тип `PivotTableConditionalValueFilterOperation`.

**Пример**

```
local valueOperation =
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.Pivo
tTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

local valueFilter = pivotTable:getConditionalValueFilter("Product Name")
valueFilter:setOperation(valueOperation)

tableEditor:setFilter(valueFilter):apply()
```

### 5.137.2 Метод PivotTableConditionalValueFilter:getExpectedOperandType

Метод возвращает тип данных, который должен использоваться с заданной операцией сравнения.

**Вызов**

```
static PivotTableConditionalValueFilterOperandType
getExpectedOperandType(operationType)
```

**Параметры**

– `operationType`: тип операции сравнения, тип `PivotTableConditionalValueFilterOperationType`.

**Возвращает**

– тип данных, тип `PivotTableConditionalValueFilterOperandType`.

### 5.137.3 Метод PivotTableConditionalValueFilter:getFieldName

Метод возвращает имя поля, с которым ассоциирован фильтр.

**Вызов**

```
string getFieldName()
```

**Возвращает**

– имя поля, тип `string`.

### 5.137.4 Метод PivotTableConditionalValueFilter:getOperation

Метод возвращает текущие настройки фильтра.

**Вызов**

```
PivotTableConditionalValueFilterOperation getOperation()
```

**Возвращает**

– настройки фильтра, тип PivotTableConditionalValueFilterOperation.

### 5.137.5 Метод PivotTableConditionalValueFilter:isInDefaultState

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

**Вызов**

```
bool isInDefaultState()
```

**Возвращает**

– `true`, если фильтр не содержит настроек, в ином случае – `false`.

### 5.137.6 Метод PivotTableConditionalValueFilter:reset

Метод сбрасывает настройки текущего фильтра.

**Вызов**

```
reset()
```

### 5.137.7 Метод PivotTableConditionalValueFilter:setOperation

Метод устанавливает настройки фильтра.

**Вызов**

```
setOperation(operation)
```

**Параметры**

– `operation`: настройки фильтра, тип PivotTableConditionalValueFilterOperation.

**Пример**

```
local valueOperation =
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.Pivo
tTableConditionalValueFilterOperationType_Between)
```

```
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

local valueFilter = pivotTable:getConditionalValueFilter("Product Name")
valueFilter:setOperation(valueOperation)

tableEditor:setFilter(valueFilter):apply()
```

### 5.138 Таблица DocumentAPI.PivotTableConditionalValueFilterOperandType

Таблица `PivotTableConditionalValueFilterOperandType` определяет тип данных, с которыми могут работать различные операции сравнения. Используется в методе `PivotTableConditionalValueFilter:getExpectedOperandType()`.

Таблица 78 – Описание типов данных

| Значение | Описание |
|---|---|
| `PivotTableConditionalValueFilterOperandType_PositiveInt` | Положительное целое число |
| `PivotTableConditionalValueFilterOperandType_Percent` | Число с плавающей запятой в диапазоне от 0 до 100 |
| `PivotTableConditionalValueFilterOperandType_Double` | Дробное число |
| `PivotTableConditionalValueFilterOperandType_NonNegativeDouble` | Положительное дробное число |

### 5.139 Таблица DocumentAPI.PivotTableConditionalValueFilterOperation

Таблица `PivotTableConditionalValueFilterOperation` представляет собой настройки условного фильтра по значению. Используется в методах `PivotTableConditionalValueFilter:getDefaultOperation()`, `PivotTableConditionalValueFilter:getOperation()` и `PivotTableConditionalValueFilter:setOperation()`.

Таблица 79 – Описание полей таблицы `PivotTableConditionalValueFilterOperation`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableConditionalValueFilterOperation.operationType` | `PivotTableConditionalValueFilterOperationType` | Тип операции сравнения |
| `PivotTableConditionalValueFilterOperation.valueFieldIndex` | `number` | Индекс поля, к которому |

| Поле | Тип | Описание |
|---|---|---|
| | | применяется фильтр |
| PivotTableConditionalValueFilterOperation.operand1 | string | Первое значение для сравнения |
| PivotTableConditionalValueFilterOperation.operand2 | string | Второе значение для операции сравнения Between |

**Пример**

```
local valueOperation =
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.Pivo
tTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

local valueFilter = pivotTable:getConditionalValueFilter("Product Name")
valueFilter:setOperation(valueOperation)

tableEditor:setFilter(valueFilter):apply()
```

### 5.140 Таблица DocumentAPI.PivotTableConditionalValueFilterOperationType

Таблица PivotTableConditionalValueFilterOperationType определяет тип операции сравнения, применяемый к фильтру по значению. Используется в поле PivotTableConditionalValueFilterOperation.operationType.

Таблица 80 – Описание типов операций сравнения

| Значение | Описание |
|---|---|
| PivotTableConditionalValueFilterOperationType_Equal | Равные |
| PivotTableConditionalValueFilterOperationType_NotEqual | Не равные |
| PivotTableConditionalValueFilterOperationType_Greater | Больше |
| PivotTableConditionalValueFilterOperationType_GreaterOrEqual | Больше или равные |

| Значение | Описание |
|---|---|
| PivotTableConditionalValueFilterOperationType_Less | Меньше |
| PivotTableConditionalValueFilterOperationType_LessOrEqual | Меньше или равные |
| PivotTableConditionalValueFilterOperationType_Between | Между |
| PivotTableConditionalValueFilterOperationType_TopPercent | Не поддерживается |
| PivotTableConditionalValueFilterOperationType_BottomPercent | Не поддерживается |
| PivotTableConditionalValueFilterOperationType_TopSum | Не поддерживается |
| PivotTableConditionalValueFilterOperationType_BottomSum | Не поддерживается |
| PivotTableConditionalValueFilterOperationType_TopValues | Не поддерживается |
| PivotTableConditionalValueFilterOperationType_BottomValues | Не поддерживается |

### 5.141 Таблица DocumentAPI.PivotTableDataCalculationType

Таблица PivotTableDataCalculationType содержит типы дополнительных вычислений для поля значений сводной таблицы. Для некоторых типов вычислений необходимо дополнительно указать данные для сравнения. Используется в поле PivotTableCalculationData.calculationType.

Таблица 81 – Описание типов дополнительных вычислений

| Значение | Описание |
|---|---|
| PivotTableDataCalculationType_Normal | Без вычислений |
| PivotTableDataCalculationType_Difference | Отличие (от значения заданного поля) |
| PivotTableDataCalculationType_Percent | % от (значения заданного поля) |
| PivotTableDataCalculationType_PercentDiff | Отличие в % (от значения заданного поля) |
| PivotTableDataCalculationType_PercentOfCol | % от итога по столбцу |
| PivotTableDataCalculationType_PercentOfRow | % от итога по строке |
| PivotTableDataCalculationType_PercentOfTotal | % от общего итога |
| PivotTableDataCalculationType_RunTotal | Не поддерживается |

| Значение | Описание |
|---|---|
| PivotTableDataCalculationType_Index | Не поддерживается |
| PivotTableDataCalculationType_PercentOfParent | % от родительского итога (по заданному полю) |
| PivotTableDataCalculationType_PercentOfParentCol | % от итога по родительскому столбцу |
| PivotTableDataCalculationType_PercentOfParentRow | % от итога по родительской строке |
| PivotTableDataCalculationType_PercentOfRunningTotal | Не поддерживается |
| PivotTableDataCalculationType_RankAscending | Не поддерживается |
| PivotTableDataCalculationType_RankDescending | Не поддерживается |

### 5.142 Таблица DocumentAPI.PivotTableEditor

Предназначена для редактирования сводных таблиц. Возвращается посредством метода PivotTable:createPivotTableEditor().

#### 5.142.1 Метод PivotTableEditor:addField

Метод добавляет новое поле в сводную таблицу, используя параметры:

– fieldName - имя поля;

– toCategory - категория поля (тип - DocumentAPI.PivotTableFieldCategory);

– index - позиция в категории.

Метод возвращает объект DocumentAPI.PivotTableEditor.

**Пример**

```
pivotTableEditor = pivotTableEditor:addField("CC",
DocumentAPI.PivotTableFieldCategory_Values)
pivotTableEditor:apply()
```

#### 5.142.2 Метод PivotTableEditor:apply

Метод обновляет сводную таблицу с заданными свойствами и возвращает результат DocumentAPI.PivotTableUpdateResult.

**Пример**

```
local pivotTab8leEditor = pivotTable:createPivotTableEditor()
if DocumentAPI.PivotTableUpdateResult_Success == pivotTableEditor:apply() then
    print("Successfully applied");
end
```

### 5.142.3 Метод PivotTableEditor:disableField

Метод удаляет поле из всех областей. Параметр `fieldName` - имя поля (тип - строка). Метод возвращает объект `DocumentAPI.PivotTableEditor`.

**Пример**

```
local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor:disableField("Age")
pivotTableEditor:apply()
```

### 5.142.4 Метод PivotTableEditor:enableField

Метод добавляет поле в область, зависящую от типа поля. Параметр `fieldName` - имя поля. Метод возвращает объект `DocumentAPI.PivotTableEditor`.

**Пример**

```
local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor:enableField("Age")
pivotTableEditor:apply()
```

### 5.142.5 Метод PivotTableEditor:moveField

Метод перемещает поле между категориями.

**Параметры:**

- `fieldName` - имя поля;
- `toCategory` - область, в которую перемещается поле (тип - `DocumentAPI.PivotTableFieldCategory`);
- `index` - позиция в новой категории.

Метод возвращает объект `DocumentAPI.PivotTableEditor`.

**Пример**

```
pivotTableEditor = pivotTableEditor:moveField("BB",
DocumentAPI.PivotTableFieldCategory_Values, 0)
pivotTableEditor:apply()
```

### 5.142.6 Метод PivotTableEditor:removeField

Метод удаляет поле из категории.

**Параметры:**

– `fieldName` - имя поля,

– `fromCategory` - область, из которой удаляется поле (тип - `DocumentAPI.PivotTableFieldCategory`).

Метод возвращает объект `DocumentAPI.PivotTableEditor`.

**Пример**

```
pivotTableEditor = pivotTableEditor:removeField("Age",
DocumentAPI.PivotTableFieldCategory_Values)
pivotTableEditor:apply()
```

### 5.142.7 Метод PivotTableEditor:reorderField

Метод изменяет позицию поля в пределах категории.

**Параметры:**

– `fieldName` - имя поля;

– `category` - область (тип - `DocumentAPI.PivotTableFieldCategory`);

– `toIndex` - новая позиция поля.

Метод возвращает объект `DocumentAPI.PivotTableEditor`.

**Пример**

```
pivotTableEditor = pivotTableEditor:reorderField("Age",
DocumentAPI.PivotTableFieldCategory_Values, 0)
pivotTableEditor:apply()
```

### 5.142.8 Метод PivotTableEditor:resetAllFilters

Метод сбрасывает все фильтры, примененные к сводной таблице, и обновляет её.

**Вызов**

```
PivotTableEditor resetAllFilters()
```

**Возвращает**

– текущий редактор сводных таблиц, тип PivotTableEditor.

### 5.142.9 Метод PivotTableEditor:setAdditionalCalculations

Метод задает дополнительные вычисления для поля значений сводной таблицы.

**Вызов**

```
PivotTableEditor setAdditionalCalculations(valueFieldName, calculationData)
```

**Параметры**

– `valueFieldName`: название поля значений, тип `string`.

– `calculationData`: настройки дополнительных вычислений, тип PivotTableCalculationData.

**Возвращает**

– текущий редактор сводных таблиц, тип PivotTableEditor.

**Пример дополнительных вычислений типа "Отличие" от предыдущего значения в поле Country**

```
local sheet = document:getBlocks():getTable(0)
local pivotTable = sheet:getCell("J8"):getPivotTable()
local tableEditor = pivotTable:createPivotTableEditor()

local data = DocumentAPI.PivotTableCalculationData()
data.calculationType = DocumentAPI.PivotTableDataCalculationType_Difference
local baseItem =
DocumentAPI.PivotTableCalculationDataBaseItem(DocumentAPI.PivotTableBaseItemPosit
ion_Previous)
data.baseEntity = DocumentAPI.PivotTableCalculationDataBaseEntity("Country",
baseItem)

tableEditor:setAdditionalCalculations("Sum of Total", data)
tableEditor:apply()
```

### 5.142.10 Метод PivotTableEditor:setCaptions

Метод задает заголовки сводной таблицы DocumentAPI.PivotTableCaptions, возвращает объект DocumentAPI.PivotTableEditor.

**Пример**

```
local pivotTableCaptions = pivotTable:getPivotTableCaptions()
pivotTableCaptions.grandTotalCaption = "Общий итог за год"

local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor = pivotTableEditor:setCaptions(pivotTableCaptions)
pivotTableEditor:apply()
```

### 5.142.11 Метод PivotTableEditor:setColumnFieldsCollapsed

Метод сворачивает или разворачивает все поля в области столбцов сводной таблицы.

**Вызов**

```
PivotTableEditor setColumnFieldsCollapsed(collapse)
```

**Параметры**

– `collapse`: `true`, чтобы свернуть поля в области столбцов, в ином случае – `false`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

### 5.142.12 Метод PivotTableEditor:setFieldAlias

Метод задает альтернативное название для указанного поля.

**Вызов**

```
PivotTableEditor setFieldAlias(fieldName, newAlias)
```

**Параметры**

– `fieldName`: название поля, тип `string`.

– `newAlias`: альтернативное название поля, тип `string`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local pivotTable = sheet:getCell("J8"):getPivotTable()
local tableEditor = pivotTable:createPivotTableEditor()

tableEditor:setFieldAlias("Country", "Location"):apply()
```

### 5.142.13 Метод PivotTableEditor:setFieldCollapsed

Метод сворачивает или разворачивает заданное поле сводной таблицы.

**Вызов**

```
PivotTableEditor setFieldCollapsed(fieldName, collapse)
```

**Параметры**

– `fieldName`: название поля, тип `string`.

– `collapse`: `true`, чтобы свернуть заданное поле, в ином случае – `false`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

### 5.142.14 Метод PivotTableEditor:setFilter

Метод задает фильтр `DocumentAPI.PivotTableFilter`, `DocumentAPI.PivotTableConditionalLabelFilter` или `DocumentAPI.PivotTableConditionalValueFilter` сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`. Метод возвращает объект `DocumentAPI.PivotTableEditor`.

**Перегрузки**

```
PivotTableEditor setFilter(PivotTableFilter filter)
```

```
PivotTableEditor setFilter(PivotTableConditionalLabelFilter filter)
```

```
PivotTableEditor setFilter(PivotTableConditionalValueFilter filter)
```

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      filter:setHidden(filterIdx, false)
      pivotTableEditor:setFilter(filter)
```

```
   end
end
pivotTableEditor:apply()
```

### 5.142.15 Метод PivotTableEditor:setFilters

Метод задает фильтры DocumentAPI.PivotTableFilters сводной таблицы. Если какой-то из фильтров не может быть применен, он пропускается. Метод возвращает объект DocumentAPI.PivotTableEditor.

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      filter:setHidden(filterIdx, false)
   end
end
pivotTableEditor:setFilters(filters)
pivotTableEditor:apply()
```

### 5.142.16 Метод PivotTableEditor:setGrandTotalSettings

Метод задает настройки отображения общего итога.

**Параметры:**

– `isRowGrandTotalEnabled` – показывать общие итоги для строк;

– `isColGrandTotalEnabled` – показывать общие итоги для столбцов.

**Пример**

```
local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor:setGrandTotalSettings(true, true)
```

### 5.142.17 Метод PivotTableEditor:setItemCollapsed

Метод сворачивает или разворачивает заданный элемент сводной таблицы.

**Вызов**

```
PivotTableEditor setItemCollapsed(item, collapse)
```

**Параметры**

– `item`: элемент сводной таблицы, тип PivotTableItem.

– `collapse`: `true`, чтобы свернуть заданный элемент, в ином случае – `false`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

### 5.142.18 Метод PivotTableEditor:setLayoutSettings

Метод устанавливает настройки отображения `DocumentAPI.PivotTableLayoutSettings` сводной таблицы, возвращает объект `DocumentAPI.PivotTableEditor`.

**Пример**

```
local layoutSettings = pivotTable:getPivotTableLayoutSettings()
layoutSettings.reportLayout = DocumentAPI.PivotTableReportLayout_Tabular

local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor = pivotTableEditor:setLayoutSettings(layoutSettings)
pivotTableEditor:apply()
```

### 5.142.19 Метод PivotTableEditor:setRowFieldsCollapsed

Метод сворачивает или разворачивает все поля в области строк сводной таблицы.

**Вызов**

```
PivotTableEditor setRowFieldsCollapsed(collapse)
```

**Параметры**

– `collapse`: `true`, чтобы свернуть поля в области строк, в ином случае – `false`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

### 5.142.20 Метод PivotTableEditor:setSortingByLabel

Метод позволяет отсортировать данные в сводной таблице по названию строк и столбцов.

**Вызов**

```
PivotTableEditor setSortingByLabel(fieldName, order)
```

**Параметры**

– `fieldName`: название поля для сортировки, тип `string`.

– `order`: порядок сортировки, тип `PivotTableSortingOrder`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("L1")
local pivotTable = cell:getPivotTable()
local editor = pivotTable:createPivotTableEditor()
editor:setSortingByLabel("Product", DocumentAPI.PivotTableSortingOrder_Ascending)
editor:apply()
```

### 5.142.21 Метод PivotTableEditor:setSortingByValue

Метод позволяет отсортировать данные в сводной таблице по значениям и итогам.

**Вызов**

```
PivotTableEditor setSortingByValue(fieldName, order, valueFieldName,
sortingByValueSlice, sortingFieldSubtotal, valueSliceSubtotal)
```

**Параметры**

– `fieldName`: название поля для сортировки, тип `string`.

– `order`: порядок сортировки, тип `PivotTableSortingOrder`.

– `valueFieldName`: название поля значений, которое содержит данные для сортировки, тип `string`.

– `sortingByValueSlice`: (необязательный) срез данных для сортировки по значениям, содержит название строки или столбца с данными или путь до него во вложенной структуре, тип `VectorString`.

– `sortingFieldSubtotal`: (необязательный) функция для сортировки по подытогам, применяется к заданному в параметре `fieldName` полю, тип `PivotTableFunction`.

– `valueSliceSubtotal`: (необязательный) функция для сортировки по подытогам, применяется к заданному в параметре `sortingByValueSlice` полю, тип `PivotTableFunction`.

**Возвращает**

– текущий редактор сводных таблиц, тип PivotTableEditor.

**Пример сортировки по общим итогам**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("L1")
local pivotTable = cell:getPivotTable()
local editor = pivotTable:createPivotTableEditor()
editor:setSortingByValue("Manager", DocumentAPI.PivotTableSortingOrder_Ascending,
"Сумма по полю Total")
editor:apply()
```

**Пример сортировки по значениям во вложенном столбце "Russia > Furniture"**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("L1")
local pivotTable = cell:getPivotTable()
local editor = pivotTable:createPivotTableEditor()
local columnSlice = DocumentAPI.VectorString()
columnSlice:push_back("Russia")
columnSlice:push_back("Furniture")
editor:setSortingByValue("Manager",
DocumentAPI.PivotTableSortingOrder_Descending, "Сумма по полю Total",
columnSlice)
editor:apply()
```

**Пример сортировки по промежуточным итогам в столбце "Russia"**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("L1")
local pivotTable = cell:getPivotTable()
local editor = pivotTable:createPivotTableEditor()
local columnSlice = DocumentAPI.VectorString()
columnSlice:push_back("Russia")
editor:setSortingByValue("Manager",
DocumentAPI.PivotTableSortingOrder_Descending, "Сумма по полю Total",
columnSlice, nil, DocumentAPI.PivotTableFunction_Auto)
editor:apply()
```

### 5.142.22 Метод PivotTableEditor:setSubtotalFunctions

Метод задает функции вычисления промежуточных итогов для указанного поля.

**Вызов**

```
PivotTableEditor setSubtotalFunctions(fieldName, subtotalFunctions)
```

**Параметры**

– `fieldName`: название поля, тип `string`.

– `subtotalFunctions`: функции для вычисления промежуточных итогов, тип коллекция объектов `PivotTableFunction`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local pivotTable = sheet:getCell("J8"):getPivotTable()
local tableEditor = pivotTable:createPivotTableEditor()

local functions = DocumentAPI.PivotTableFunctions()
functions:push_back(DocumentAPI.PivotTableFunction_Count)
functions:push_back(DocumentAPI.PivotTableFunction_Average)

tableEditor:setSubtotalFunctions("Product Name", functions):apply()
```

### 5.142.23 Метод PivotTableEditor:setSubtotalOnTop

Метод задает расположение промежуточных итогов для указанного поля.

**Вызов**

```
PivotTableEditor setSubtotalOnTop(fieldName, isSubtotalOnTop)
```

**Параметры**

– `fieldName`: название поля, тип `string`.

– `isSubtotalOnTop`: `true`, чтобы отображать промежуточные итоги в заголовке группы, `false` – отображать под группой.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local pivotTable = sheet:getCell("J8"):getPivotTable()
local tableEditor = pivotTable:createPivotTableEditor()

tableEditor:setSubtotalOnTop("Product Name", true):apply()
```

### 5.142.24 Метод PivotTableEditor:setSummarizeFunction

Метод задает суммирующую функцию для поля из области значений.

**Параметры**

– `valueFieldName` - имя поля (тип - строка);

– `summarizeFunction` – суммирующая функция, тип - `DocumentAPI.PivotTableFunction`.

Метод возвращает объект `DocumentAPI.PivotTableEditor`.

**Пример**

```
pivotTableEditor = pivotTableEditor:setSummarizeFunction("Age",
DocumentAPI.PivotTableFunction_Sum)
pivotTableEditor:apply()
```

### 5.142.25 Метод PivotTableEditor:setViewDetailsEnabled

Метод позволяет задать возможность просмотра детализации данных для значений сводной таблицы.

**Вызов**

```
PivotTableEditor setViewDetailsEnabled(enabled)
```

**Параметры**

– `enabled`: `true`, чтобы включить возможность просмотра детализации данных, в ином случае – `false`.

**Возвращает**

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local pivotTable = sheet:getCell("J8"):getPivotTable()
local tableEditor = pivotTable:createPivotTableEditor()

tableEditor:setViewDetailsEnabled(true):apply()
```

### 5.143 Таблица DocumentAPI.PivotTableField

Таблица `DocumentAPI.PivotTableField` содержит свойства полей сводной таблицы (см. таблицу 82). Таблица может быть получена посредством вызова `PivotTable:getFieldsList()`.

Таблица 82 – Описание полей таблицы `DocumentAPI.PivotTableField`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableField.fieldProperties` | `PivotTableFieldProperties` | Свойства полей сводной таблицы |
| `PivotTableField.fieldCategories` | `PivotTableFieldCategories` | Категории полей сводной таблицы |
| `PivotTableField.customFormula` | `string` | Вычисляемая формула |

### 5.144 Таблица DocumentAPI.PivotTableFieldCategories

Класс обеспечивает доступ к списку категорий поля сводной таблицы. Может быть получена посредством использования метода `PivotTable:getFieldCategories()`.

#### 5.144.1 Метод PivotTableFieldCategories:enumerate

Метод для перечисления категорий поля `DocumentAPI.PivotTableFieldCategory`.

**Пример**

```
local fieldCategories = pivotTable:getFieldCategories("Age")
for fieldCategory in fieldCategories:enumerate() do
   print(fieldCategory)
end
```

### 5.145 Таблица DocumentAPI.PivotTableFieldCategory

Таблица `DocumentAPI.PivotTableFieldCategory` описывает флаги, которые задают категорию области полей. Пример использования см. в `PivotTable:getFieldCategories()`.

Таблица 83 – Категории области полей

| Поле | Описание |
|---|---|
| `DocumentAPI.PivotTableFieldCategory_Pages` | Область фильтров |
| `DocumentAPI.PivotTableFieldCategory_Rows` | Область строк |
| `DocumentAPI.PivotTableFieldCategory_Columns` | Область столбцов |
| `DocumentAPI.PivotTableFieldCategory_Values` | Область значений |

### 5.146 Таблица DocumentAPI.PivotTableFieldProperties

`DocumentAPI.PivotTableFieldProperties` содержит свойства поля `DocumentAPI.PivotTableField` сводной таблицы (см. таблицу 84).

Таблица 84 – Описание полей таблицы `DocumentAPI.PivotTableFieldProperties`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableFieldProperties.fieldName` | `string` | Имя поля |
| `PivotTableFieldProperties.fieldAlias` | `string` | Псевдоним поля (пользовательское имя) |
| `PivotTableFieldProperties.subtotalAlias` | `string` | Псевдоним подытогов конкретного поля |

### 5.147 Таблица DocumentAPI.PivotTableFilter

Позволяет осуществить доступ к списку фильтров таблицы, каждый из которых обладает свойством видимости (Рис. 52).

. При любом изменении фильтров они должны быть применены к сводной таблице посредством использования методов PivotTableEditor:setFilter(), PivotTableEditor:setFilters().

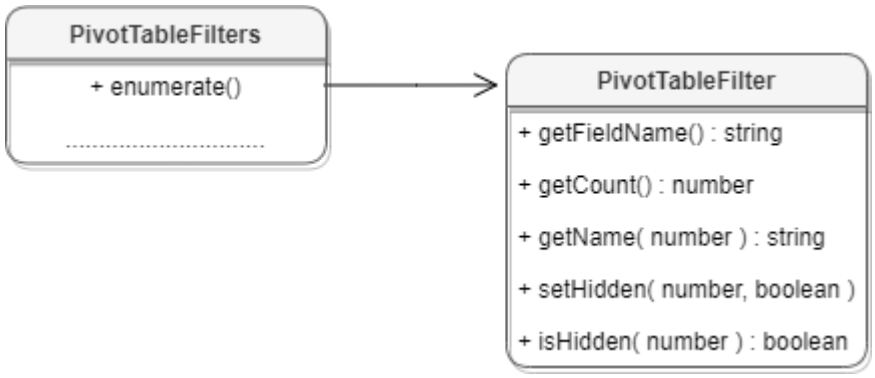



Рисунок 52 – Таблица DocumentAPI.PivotTableFilter

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      filter:setHidden(filterIdx, false)
   end
end
pivotTableEditor:setFilters(filters)
pivotTableEditor:apply()
```

### 5.147.1 Метод PivotTableFilter:getCount

Возвращает количество фильтруемых полей.

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   print(filter:getCount())
end
```

### 5.147.2 Метод PivotTableFilter:getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      print(filter:getFieldName())
   end
end
```

### 5.147.3 Метод PivotTableFilter:getName

Возвращает имя поля для заданного индекса.

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      print(filter:getName(filterIdx))
   end
end
```

### 5.147.4 Метод PivotTableFilter:isHidden

Возвращает видимость поля для заданного индекса `itemIndex`. Если `true`, то поле скрыто.

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      print(filter:isHidden(filterIdx))
   end
end
```

### 5.147.5 Метод PivotTableFilter:isInDefaultState

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

**Вызов**

```
bool isInDefaultState()
```

**Возвращает**

– `true`, если фильтр не содержит настроек, в ином случае – `false`.

### 5.147.6 Метод PivotTableFilter:reset

Метод сбрасывает настройки текущего фильтра.

**Вызов**

```
reset()
```

### 5.147.7 Метод PivotTableFilter:setHidden

Устанавливает видимость поля для заданного индекса. Параметры: `itemName` – индекс поля, `hidden` – видимость (`true` – поле скрыто).

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      print(filter:setName(filterIdx, false))
   end
end
```

### 5.148 Таблица DocumentAPI.PivotTableFilters

Таблица обеспечивает доступ к списку фильтров. Для получения `DocumentAPI.PivotTableFilters` используется метод `PivotTable:getFilters()`.

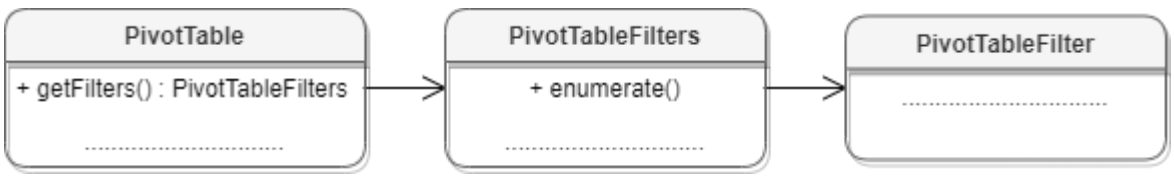



Рисунок 53 – Объектная модель таблиц для работы с фильтрами

### 5.148.1 Метод PivotTableFilters:enumerate

Метод используется для доступа к коллекции фильтров (см. DocumentAPI.PivotTableFilter).

**Пример**

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   print(filter:getName(0))
   print(filter:getFieldName())
end
```

## 5.149 Таблица DocumentAPI.PivotTableFunction

Таблица DocumentAPI.PivotTableFunction описывает функции, которые могут быть использованы в сводных таблицах. Описание полей таблицы представлено в таблице 85. Таблица используется в качестве поля subtotalFunctions таблицы DocumentAPI.PivotTableCategoryField.

Таблица 85 – Описание полей таблицы DocumentAPI.PivotTableFunction

| Значение | Описание |
|---|---|
| DocumentAPI.PivotTableFunction_Auto | Автозаполнение |
| DocumentAPI.PivotTableFunction_Sum | Суммирует все числовые данные |
| DocumentAPI.PivotTableFunction_Count | Количество всех ячеек |
| DocumentAPI.PivotTableFunction_CountNums | Количество числовых ячеек |
| DocumentAPI.PivotTableFunction_Average | Среднее значение |
| DocumentAPI.PivotTableFunction_Max | Наибольшее значение |
| DocumentAPI.PivotTableFunction_Min | Наименьшее значение |
| DocumentAPI.PivotTableFunction_Product | Произведение всех ячеек |

| Значение | Описание |
|---|---|
| `DocumentAPI.PivotTableFunction_StdDeviation` | Стандартное смещенное отклонение |
| `DocumentAPI.PivotTableFunction_StdDeviationPopulation` | Стандартное несмещенное отклонение |
| `DocumentAPI.PivotTableFunction_Variance` | Смещенная дисперсия |
| `DocumentAPI.PivotTableFunction_VariancePopulation` | Несмещенная дисперсия |

### 5.150 Таблица DocumentAPI.PivotTableItem

`DocumentAPI.PivotTableItem` описывает элемент сводной таблицы (см. Рисунок 54). См. пример в главе `PivotTableItems:enumerate`.

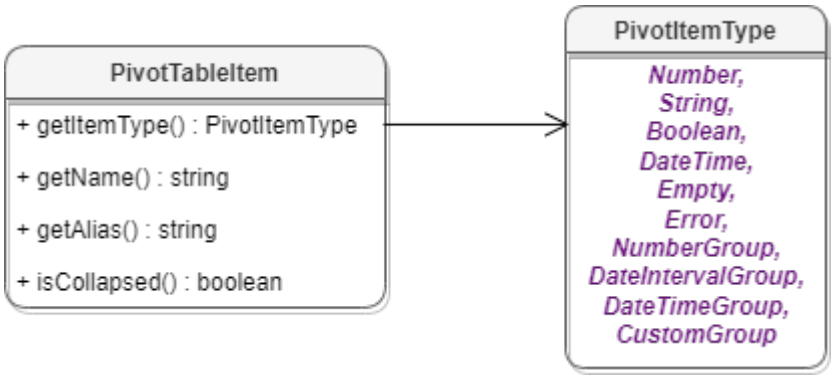



Рисунок 54 – Таблица `DocumentAPI.PivotTableItem`

#### 5.150.1 Метод PivotTableItem:getAlias

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем), тип - строка. См. пример в главе `PivotTableItems:enumerate`.

#### 5.150.2 Метод PivotTableItem:getItemType

Метод возвращает тип `DocumentAPI.PivotTableItemType` элемента сводной таблицы. См. пример в главе `PivotTableItems:enumerate`.

### 5.150.3 Метод PivotTableItem:getName

Метод возвращает имя элемента сводной таблицы, тип - строка. См. пример в главе PivotTableItems:enumerate.

### 5.150.4 Метод PivotTableItem:isCollapsed

Метод возвращает `true`, если элемент сводной таблицы свернут. См. пример в главе PivotTableItems:enumerate.

## 5.151 Таблица DocumentAPI.PivotTableItemForCategory

Таблица `PivotTableItemForCategory` представляет собой элемент коллекции PivotTableSlicePath.

Таблица 86 – Описание полей таблицы `PivotTableItemForCategory`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableItemForCategory.fieldName` | `string` | Название поля сводной таблицы |
| `PivotTableItemForCategory.item` | PivotTableItem | Элемент сводной таблицы |

## 5.152 Таблица DocumentAPI.PivotTableItems

Таблица обеспечивает доступ к списку элементов сводной таблицы (см. Рисунок 55).

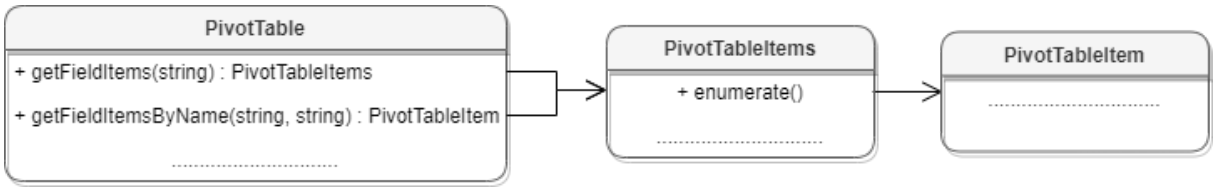



Рисунок 55 – Объектная модель таблиц для работы с элементами сводных таблиц

### 5.152.1 Метод PivotTableItems:enumerate

Используется для перечисления элементов сводной таблицы.

**Пример**

```
local fieldItems = pivotTable:getFieldItems("Age")
for fieldItem in fieldItems:enumerate() do
```

```
   print(fieldItem:getName())
   print(fieldItem:getAlias())
   print(fieldItem:getItemType())
   print(fieldItem:isCollapsed())
end
```

### 5.153 Таблица DocumentAPI.PivotTableItemType

Таблица `DocumentAPI.PivotTableItemType` содержит возможные типы элементов сводной таблицы. Описание полей таблицы представлено в таблице 87.

Таблица 87 – Типы элементов сводной таблицы

| Поле | Описание |
|---|---|
| `DocumentAPI.PivotTableItemType_Number` | Числовой |
| `DocumentAPI.PivotTableItemType_String` | Строковый |
| `DocumentAPI.PivotTableItemType_Boolean` | Логический |
| `DocumentAPI.PivotTableItemType_DateTime` | Дата / время |
| `DocumentAPI.PivotTableItemType_Empty` | Пустой тип |
| `DocumentAPI.PivotTableItemType_Error` | Ошибка |
| `DocumentAPI.PivotTableItemType_NumberGroup` | Интервальная группировка |
| `DocumentAPI.PivotTableItemType_DateIntervalGroup` | Интервальная группировка по датам |
| `DocumentAPI.PivotTableItemType_DateTimeGroup` | Группировка по дате / времени |
| `DocumentAPI.PivotTableItemType_CustomGroup` | Пользовательская (произвольная) группировка |

**Пример**

```
local fieldItems = pivotTable:getFieldItems("Age")
for fieldItem in fieldItems:enumerate() do
   if (fieldItem:getItemType() == DocumentAPI.PivotTableItemType_Number) then
```

```
      print("Numeric type")
   end
end
```

### 5.154 Таблица DocumentAPI.PivotTableLayoutSettings

Таблица `DocumentAPI.PivotTableLayoutSettings` содержит настройки отображения сводной таблицы. Данная таблица может быть получена в результате вызова `PivotTable:getPivotTableLayoutSettings()` и установлена методом `PivotTableEditor:setLayoutSettings()`.

Таблица 88 – Описание полей таблицы `DocumentAPI.PivotTableLayoutSettings`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableLayoutSettings.reportLayout` | `PivotTableReportLayout` | Настройка вида макета сводной таблицы (компактный, табличный, структурный). |
| `PivotTableLayoutSettings.valueFieldsOrientation` | `ValueFieldsOrientation` | Настраивает положение значений в случае, если в сводной таблице более двух полей значений. |
| `PivotTableLayoutSettings.pageFieldOrder` | `PageFieldOrder` | Настройка порядка полей фильтров (вниз, затем поперек или сначала поперек, потом вниз) |
| `PivotTableLayoutSettings.indentForCompactLayout` | `number` | Размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей) |
| `PivotTableLayoutSettings.pageFieldWrapCount` | `number` | Настройка связана с `pageFieldOrder`, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д) |

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableLayoutSettings.isMergeAndCenterLabelsEnabled` | `bool` | Настройка позволяет объединить ячейки заголовков |
| `PivotTableLayoutSettings.useGridDropZones` | `bool` | Флаг, отвечающий за отображение классического вида (как в Excel 2003). Влияет только на расположение полей в отчете |
| `PivotTableLayoutSettings.displayFieldCaptions` | `bool` | Флаг, отвечающий за отображение заголовков полей |

### 5.155 Таблица DocumentAPI.PivotTablePageField

Содержит свойства поля из области фильтров (см. таблицу 89). Таблица может быть получена посредством вызова `PivotTable:getPageFields()`.

**Конструкторы**

```
PivotTablePageField()
```

```
PivotTablePageField(fieldName, fieldAlias, subtotalAlias)
```

**Параметры**

– `fieldName`: название поля, тип `string`.

– `fieldAlias`: псевдоним поля, тип `string`.

– `subtotalAlias`: псевдоним промежуточных итогов поля, тип `string`.

Таблица 89 – Описание полей таблицы `DocumentAPI.PivotTablePageField`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTablePageField.fieldProperties` | `PivotTableFieldProperties` | Свойства поля |

### 5.156 Таблица DocumentAPI.PivotTableReportLayout

Таблица `DocumentAPI.PivotTableReportLayout` описывает внешний вид отчетов сводной таблицы. Является полем таблицы `DocumentAPI.PivotTableLayoutSettings`.

Таблица 90 – Варианты внешнего вида отчетов

| Значение | Описание |
|---|---|
| `DocumentAPI.PivotTableReportLayout_Compact` | Компактный вид |
| `DocumentAPI.PivotTableReportLayout_Tabular` | Табличный вид |
| `DocumentAPI.PivotTableReportLayout_Outline` | Структурный вид |

### 5.157 Таблица DocumentAPI.PivotTableSlicePath

Таблица `PivotTableSlicePath` представляет собой путь до столбца или строки в сводной таблице. Данная таблица является коллекцией объектов `PivotTableItemForCategory`. Используется в поле `PivotTableSortingParams.sortByValueSlice`.

### 5.158 Таблица DocumentAPI.PivotTablesManager

Таблица `PivotTablesManager` используется для создания сводных таблиц, содержит метод `create()`. Может быть получена вызовом `Document:getPivotTablesManager()`.

**Пример**

```
local pivotTablesManager = document:getPivotTablesManager()
```

#### 5.158.1 Метод PivotTablesManager:create

Метод создает сводную таблицу на основе диапазона исходных данных.

**Вызов**

```
PivotTable create(cellRange, destination)
```

```
PivotTable create(formula, destination)
```

**Параметры**

– `cellRange`: диапазон данных для сводной таблицы, тип `CellRange`.

– `formula`: формула, результатом которой является диапазон данных для сводной таблицы, тип `string`.

– `destination`: (необязательный) местоположение левой верхней ячейки сводной таблицы, тип `Cell`. Если местоположение не задано, создается новый лист, и сводная таблица будет расположена в его верхнем левом углу.

**Возвращает**

– созданная сводная таблица, тип `PivotTable`.

**Пример**

```
local pivotTablesManager = document:getPivotTablesManager()
local tbl = document:getBlocks():getTable(0)
local cellRange = tbl:getCellRange("I3:K7")
local pivotTable = pivotTablesManager:create(cellRange, tbl:getCell("L8"))
```

### 5.159 Таблица DocumentAPI.PivotTableSortingOrder

Таблица `PivotTableSortingOrder` определяет порядок сортировки данных в сводной таблице. Используется в методах `PivotTableEditor:setSortingByLabel()` и `PivotTableEditor:setSortingByValue()` и в поле `PivotTableSortingParams.sortingOrder`.

Таблица 91 – Варианты порядка сортировки данных в сводной таблице

| Значение | Описание |
|---|---|
| `PivotTableSortingOrder_Ascending` | Сортировка по возрастанию |
| `PivotTableSortingOrder_Descending` | Сортировка по убыванию |

### 5.160 Таблица DocumentAPI.PivotTableSortingParams

Таблица `PivotTableSortingParams` содержит настройки сортировки данных в сводной таблице. Данная таблица используется в методе `PivotTable:getSortingParams()`.

Таблица 92 – Описание полей таблицы PivotTableSortingParams

| Поле | Тип | Описание |
|---|---|---|
| PivotTableSortingParams.sortByValueSlice | PivotTableSlicePath | Срез данных для сортировки по значениям, который содержит путь до столбца или строки с данными |
| PivotTableSortingParams.sortFieldSubtotal | PivotTableFunction | Функция для сортировки по подытогам, которая применена к текущему полю |
| PivotTableSortingParams.sortingOrder | PivotTableSortingOrder | Порядок сортировки |
| PivotTableSortingParams.sortingType | PivotTableSortingType | Тип сортировки |
| PivotTableSortingParams.valueFieldNameForSorting | string | Название поля значений, которое содержит данные для сортировки |
| PivotTableSortingParams.valueSliceSubtotal | PivotTableFunction | Функция для сортировки по подытогам, которая применена к sortByValueSlice полю |

### 5.161 Таблица DocumentAPI.PivotTableSortingType

Таблица PivotTableSortingType определяет тип сортировки данных в сводной таблице. Используется в поле PivotTableSortingParams.sortingType.

Таблица 93 – Типы сортировки данных в сводной таблице

| Значение | Описание |
|---|---|
| PivotTableSortingType_Manual | Ручная сортировка |
| PivotTableSortingType_Label | Сортировка по названию строк или столбцов |
| PivotTableSortingType_Value | Сортировка по значениям и итогам |

### 5.162 Таблица DocumentAPI.PivotTableUnsupportedFeature

Таблица DocumentAPI.PivotTableUnsupportedFeature описывает неподдерживаемую функциональность сводных таблиц. Их получение описано в PivotTable:getUnsupportedFeatures().

Таблица 94 – Описание значений `DocumentAPI.PivotTableUnsupportedFeature`

| Значение | Описание |
|---|---|
| `PivotTableUnsupportedFeature_CalculatedItem` | Вычисляемые элементы |
| `PivotTableUnsupportedFeature_ShowDataAs` | Вычисления ("Show data" как в MS Excel) |
| `PivotTableUnsupportedFeature_MultipleFilters` | Множественная фильтрация |
| `PivotTableUnsupportedFeature_GroupFieldFilter` | Условная фильтрация для сгруппированных полей |
| `PivotTableUnsupportedFeature_UnsupportedFilter` | Неподдерживаемый фильтр |

### 5.163 Таблица DocumentAPI.PivotTableUpdateResult

В таблице 95 приведены константы, которые соответствуют возможным результатам обновления сводной таблицы (см. методы `PivotTable:update()`, `PivotTableEditor:apply()`).

Таблица 95 – Результаты обновления сводной таблицы

| Значение | Описание |
|---|---|
| `DocumentAPI.PivotTableUpdateResult_Success` | Успешное обновление таблицы |
| `DocumentAPI.PivotTableUpdateResult_NoPivotTable` | Сводная таблица не найдена |
| `DocumentAPI.PivotTableUpdateResult_NoSuchFieldInCategory` | Не найдено поле в категории |
| `DocumentAPI.PivotTableUpdateResult_NoSuchFieldInPivotTable` | Не найдено поле в сводной таблице |
| `DocumentAPI.PivotTableUpdateResult_InvalidIndex` | Ошибка в индексе |
| `DocumentAPI.PivotTableUpdateResult_FieldAlreadyEnabled` | Поле уже существует |
| `DocumentAPI.PivotTableUpdateResult_MovingFieldToTheSameCategoryForbidden` | Попытка перемещения поля в рамках текущей категории |
| `DocumentAPI.PivotTableUpdateResult_InvalidFunction` | Неправильная функция |

| Значение | Описание |
|---|---|
| `DocumentAPI.PivotTableUpdateResult_InvalidCategory` | Неправильная область |
| `DocumentAPI.PivotTableUpdateResult_InvalidDataSourceRange` | Ошибка диапазона исходных данных |
| `DocumentAPI.PivotTableUpdateResult_NoDataRowsInDataSource` | В исходных данных нет строк с данными |
| `DocumentAPI.PivotTableUpdateResult_EmptyDataSourceHeaders` | Пустые заголовки исходных данных |
| `DocumentAPI.PivotTableUpdateResult_NoReferenceUnderDefine` | Попытка обновить или создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу |
| `DocumentAPI.PivotTableUpdateResult_NoSuchItem` | Элемент не найден |
| `DocumentAPI.PivotTableUpdateResult_CannotExpandCollapseLeafItem` | Не удается раскрыть свернутый элемент |
| `DocumentAPI.PivotTableUpdateResult_AnotherPivotInsideDataSource` | Найдена другая сводная таблица в этом же диапазоне |
| `DocumentAPI.PivotTableUpdateResult_AnotherFieldHasTheSameName` | Заданное альтернативное название совпадает с названием существующего поля |
| `DocumentAPI.PivotTableUpdateResult_InvalidCalculationData` | Неверно заданы настройки дополнительных вычислений |
| `DocumentAPI.PivotTableUpdateResult_InvalidConditionalFilterParams` | Неверно заданы настройки условного фильтра |
| `DocumentAPI.PivotTableUpdateResult_Canceled` | Обновление сводной таблицы отменено |
| `DocumentAPI.PivotTableUpdateResult_NoSuchSheetInExternalDocument` | Не найден лист во внешнем документе |
| `DocumentAPI.PivotTableUpdateResult_InvalidSortingParams` | Неправильно заданы настройки сортировки |

### 5.164 Таблица DocumentAPI.PivotTableValueField

`DocumentAPI.PivotTableValueField` содержит свойства поля сводной таблицы, использующегося как значение столбец (см. таблицу 96). Таблица может быть получена посредством вызова `PivotTable:getValueFields()`.

Таблица 96 – Описание полей таблицы `DocumentAPI.PivotTableValueField`

| Поле | Тип | Описание |
|---|---|---|
| `PivotTableValueField.baseFieldName` | `string` | Оригинальное поле на основе которого было создано данное поле. |
| `PivotTableValueField.valueFieldName` | `string` | Автоматический уникальный псевдоним такой как “Sum of %имя поля%“. |
| `PivotTableValueField.cellNumberFormat` | `CellFormat` | Числовой формат для конкретного поля значений. |
| `PivotTableValueField.totalFunction` | `PivotTableFunction` | Агрегирующая функция поля значений (SUM, COUNT, MAX и т.д). |
| `PivotTableValueField.customFormula` | `string` | Вычисляемая формула для поля значений. |
| `PivotTableValueField.calculationData` | `PivotTableCalculationData` | Настройки дополнительных вычислений для поля значений. |

### 5.165 Таблица DocumentAPI.PointU

Таблица `DocumentAPI.PointU` представляет позицию объекта (x, y).

Таблица 97 – Описание полей таблицы `DocumentAPI.PointU`

| Поле | Тип | Описание |
|---|---|---|
| `PointU.x` | `number` | Координата точки по оси x |
| `PointU.y` | `number` | Координата точки по оси y |

**Пример**

```
local point = DocumentAPI.PointU(2, 3)
print("x=", point.x, ", y=", point.y)  --(x = 2.0, y = 3.0)
```

#### 5.165.1 Метод PointU:toString

Возвращает информацию о позиции в виде строкового значения формата `(width: <value>, height: <value>)`.

**Пример**

```
local point = DocumentAPI.PointU(2, 3)
print(point:toString())  --(x: 2.0, y: 3.0)
```

### 5.166 Таблица DocumentAPI.Position

Таблица `DocumentAPI.Position` представляет местоположение в текстовом документе. Используется для обозначения начала и конца диапазона `DocumentAPI.Range`.

#### 5.166.1 Метод Position:compare

Метод сравнивает заданную позицию с текущей.

**Вызов**

```
number compare(other)
```

**Параметры**

– `other`: позиция для сравнения с текущей, тип `Position`.

**Возвращает**

– положительное расстояние, если текущая позиция больше заданной.

– отрицательное расстояние, если текущая позиция меньше заданной.

– 0, если позиции равны.

– `nil`, если позиции нельзя сравнить (позиции в разных документах, в тексте и в колонтитуле и т.д.).

**Примеры для текстового документа**

```
table1 = document:getRange():getBegin():insertTable(2, 2, "table1")
positionDoc = document:getRange():getBegin()
positionTable = table1:getRange():getBegin()
positionCell = table1:getCell(DocumentAPI.CellPosition(0,
0)):getRange():getBegin()

if (positionDoc:compare(positionTable) + positionTable:compare(positionCell) ==
0) then
    -- true
end
```

```
document:getRange():getBegin():insertText("Some text")
positionBegin = document:getRange():getBegin()
```

```
positionEnd = document:getRange():getEnd()

print(positionBegin:compare(positionEnd)) -- -10
print(positionEnd:compare(positionBegin)) -- 10
```

**Пример для табличного документа**

```
sheet = document:getBlocks():getTable(0)

firstCellEnd = sheet:getCell("A1"):getRange():getEnd()
secondCellBegin = sheet:getCell("B1"):getRange():getBegin()

print(firstCellEnd:compare(secondCellBegin)) -- 0
```

### 5.166.2 Метод Position:getBefore

Метод возвращает позицию в текстовом документе, которая находится перед заданной таблицей.

**Вызов**

```
static Position getBefore(table)
```

**Параметры**

– `table`: таблица в текстовом документе, тип `Table`.

**Возвращает**

– позиция перед таблицей, тип `Position`.

**Пример**

```
local table = document:getRange():getBegin():insertTable(3, 3, "table1")
document:getRange():getBegin():insertText("Text in the first cell")
local position = DocumentAPI.Position.getBefore(table)
position:insertSectionBreak(DocumentAPI.SectionBreakType_Continuous)
document:getRange():getBegin():insertText("Text before the table")
```

### 5.166.3 Метод Position:getCell

Метод возвращает ячейку, в которой находится позиция, либо `nil` если позиция не находится в ячейке.

**Пример для табличного документа**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")
```

```
local range = cell:getRange()
local pos = range:getBegin()
print(pos:getCell())
```

**Пример для текстового документа**

```
local range = document:getRange()
local pos = rng:getBegin()
print(pos:getCell())
```

### 5.166.4 Метод Position:getCurrentRange

Метод возвращает диапазон, в котором находится текущая позиция. Размер диапазона зависит от выбранной единицы текста.

**Вызов**

```
Range getCurrentRange(textUnit)
```

**Параметры**

– `textUnit`: единица текста, определяющая размер диапазона, тип `TextUnit`.

**Возвращает**

– диапазон документа, содержащий текущую позицию, тип `Range`.

**Пример для текстового документа**

```
local docBegin = document:getRange():getBegin()
local sentence = docBegin:getCurrentRange(DocumentAPI.TextUnit_Sentence)
local word = docBegin:getCurrentRange(DocumentAPI.TextUnit_Word)
local character = docBegin:getCurrentRange(DocumentAPI.TextUnit_Character)

print(sentence:extractText())
-- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
print(word:extractText()) -- Lorem
print(character:extractText()) -- L
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("B2")
local cellBegin = cell:getRange():getBegin()
local sentence = cellBegin:getCurrentRange(DocumentAPI.TextUnit_Sentence)

print(sentence:extractText()) -- First sentence.
```

### 5.166.5 Метод Position:getNextPosition

Метод возвращает позицию, которая находится на заданном расстоянии после текущей. Если результат выходит за пределы текущего абзаца, метод возвращает конец абзаца.

**Вызов**

```
Position getNextPosition(count)
```

**Параметры**

– `count`: (необязательный, по умолчанию `1`) расстояние до следующей позиции.

**Возвращает**

– следующая позиция, тип `Position`.

**Пример для текстового документа**

```
document:getRange():getBegin():insertText("New text")
document:getRange():getBegin():getNextPosition(2):insertText("!")
print(document:getRange():extractText()) -- Ne!w text
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("B2")
cell:setText("New text")
cell:getRange():getBegin():getNextPosition(2):insertText("!")

print(cell:getFormattedValue()) -- Ne!w text
```

### 5.166.6 Метод Position:getNextRange

Метод возвращает диапазон, который расположен после текущего диапазона. Размеры текущего и следующего диапазонов зависят от выбранной единицы текста. Если выбранная единица текста отлична от `Paragraph`, то работа метода `Position:getNextRange` ограничена текущим абзацем.

**Вызов**

```
Range getNextRange(textUnit)
```

**Параметры**

– `textUnit`: единица текста, определяющая размеры диапазонов, тип `TextUnit`.

**Возвращает**

– следующий диапазон документа, тип `Range`.

**Пример для текстового документа**

```
local docBegin = document:getRange():getBegin()
local nextSentence = docBegin:getNextRange(DocumentAPI.TextUnit_Sentence)
local nextWord =
docBegin:getNextRange(DocumentAPI.TextUnit_Word):getBegin():getNextRange(Document
API.TextUnit_Word)
local nextCharacter = docBegin:getNextRange(DocumentAPI.TextUnit_Character)

print(nextSentence:extractText())
-- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
print(nextWord:extractText()) -- ipsum
print(nextCharacter:extractText()) -- o
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("B2")
local cellBegin = cell:getRange():getBegin()
local sentence = cellBegin:getNextRange(DocumentAPI.TextUnit_Sentence)

print(sentence:extractText()) -- Second sentence.
```

### 5.166.7 Метод Position:getParagraph

Метод возвращает абзац (Paragraph) для текущей позиции.

**Пример для текстового документа**

```
local position = document:getRange():getBegin()
local paragraphAtPosition = position:getParagraph()
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("B2")
local position = cell:getRange():getBegin()
local paragraphAtPosition = position:getParagraph()
```

### 5.166.8 Метод Position:getPreviousPosition

Метод возвращает позицию, которая находится на заданном расстоянии перед текущей. Если результат выходит за пределы текущего абзаца, метод возвращает начало абзаца.

**Вызов**

```
Position getPreviousPosition(count)
```

**Параметры**

– `count`: (необязательный, по умолчанию `1`) расстояние до предыдущей позиции.

**Возвращает**

– предыдущая позиция, тип `Position`.

**Пример для текстового документа**

```
document:getRange():getBegin():insertText("New text")
document:getRange():getContentEnd():getPreviousPosition(2):insertText("!")
print(document:getRange():extractText()) -- New te!xt
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("B2")
cell:setText("New text")
cell:getRange():getContentEnd():getPreviousPosition(2):insertText("!")

print(cell:getFormattedValue()) -- New te!xt
```

### 5.166.9 Метод Position:getPreviousRange

Метод возвращает диапазон, который расположен перед текущим диапазоном. Размеры текущего и предыдущего диапазонов зависят от выбранной единицы текста. Если выбранная единица текста отлична от `Paragraph`, то работа метода `Position:getPreviousRange` ограничена текущим абзацем.

**Вызов**

```
Range getPreviousRange(textUnit)
```

**Параметры**

– `textUnit`: единица текста, определяющая размеры диапазонов, тип `TextUnit`.

**Возвращает**

– предыдущий диапазон документа, тип `Range`.

**Пример для текстового документа**

```
local sentenceBegin =
document:getRange():getBegin():getNextRange(DocumentAPI.TextUnit_Sentence):getBeg
in()
```

```
local previousSentence =
sentenceBegin:getPreviousRange(DocumentAPI.TextUnit_Sentence)
local previousWord = sentenceBegin:getPreviousRange(DocumentAPI.TextUnit_Word)
while previousWord:getContentEnd():compare(previousWord:getBegin()) == 1 do
    previousWord =
previousWord:getBegin():getPreviousRange(DocumentAPI.TextUnit_Word)
end
local previousCharacter =
sentenceBegin:getPreviousRange(DocumentAPI.TextUnit_Character)
previousCharacter =
previousCharacter:getBegin():getPreviousRange(DocumentAPI.TextUnit_Character)
previousCharacter =
previousCharacter:getBegin():getPreviousRange(DocumentAPI.TextUnit_Character)

print(previousSentence:extractText())
-- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
print(previousWord:extractText()) -- aliqua
print(previousCharacter:extractText()) -- a
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell("B2")
local cellEnd = cell:getRange():getContentEnd()
local sentence = cellEnd:getPreviousRange(DocumentAPI.TextUnit_Sentence)

print(sentence:extractText()) -- Second sentence.
```

### 5.166.10 Метод Position:getStyle

Метод возвращает стиль фрагмента текста, в котором расположена текущая позиция. Данный метод в основном специализируется на получении стилей из документов, созданных в редакторе Microsoft Word. В большинстве случаев рекомендуется использовать метод `Paragraph:getStyle()` или `Paragraph:getResolvedStyle()`.

**Вызов**

```
TextStyle getStyle()
```

**Возвращает**

– стиль фрагмента, в котором расположена текущая позиция, тип `TextStyle`.

– `nil`, если стиль не задан.

**Пример**

```
local position = document:getRange():getBegin():getNextPosition(12)
print(position:getStyle():getName())
```

### 5.166.11 Метод Position:insertBookmark

Вставляет закладку с наименованием в заданную позицию текстового документа.

Внимание ! Метод может быть использован только в текстовом редакторе.

**Пример**

```
document:getRange():getBegin():insertBookmark("Bookmark example")
```

### 5.166.12 Метод Position:insertEndnote

Метод вставляет концевую сноску в текущую позицию.

**Вызов**

```
Position insertEndnote()
```

**Возвращает**

– позиция содержимого концевой сноски, тип Position.

**Пример**

```
local range = document:getRange()
local sentence = range:getBegin():getCurrentRange(DocumentAPI.TextUnit_Sentence)
local pos = sentence:getContentEnd():insertEndnote()
pos:insertText("Endnote")
```

### 5.166.13 Метод Position:insertFootnote

Метод вставляет сноску в текущую позицию.

**Вызов**

```
Position insertFootnote()
```

**Возвращает**

– позиция содержимого сноски, тип Position.

**Пример**

```
local range = document:getRange()
local sentence = range:getBegin():getCurrentRange(DocumentAPI.TextUnit_Sentence)
local pos = sentence:getContentEnd():insertFootnote()
pos:insertText("Footnote")
```

### 5.166.14 Метод Position:insertHyperlink

Метод `insertHyperlink` вставляет ссылку в текущую позицию. В качестве параметров передаются адрес ссылки и текст ссылки.

**Вызов**

```
insertHyperlink( url, size )
```

**Параметры**

– `url` – адрес ссылки;

– `label` – текст ссылки.

**Пример**

```
document:getRange():getBegin():insertHyperlink("https://testhyperlink.com",
"Hyperlink")
```

### 5.166.15 Метод Position:insertImage

Вставляет изображение в позицию текстового документа.

Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

**Вызов**

```
Image insertImage(url, size)
```

**Параметры**

– `url` – полный путь к локальному файлу, либо ссылка на сетевой ресурс;

– `size` – геометрические размеры изображения для вставки.

**Возвращает**

– `Image`: вставленное изображение.

**Примеры**

```
local image = document:getRange():getBegin():insertImage("C://Tmp//123.jpg",
DocumentAPI.SizeU(100, 100))
```

```
local image =
document:getRange():getBegin():insertImage
("https://www.images.ru/images/fish.jpg", DocumentAPI.SizeU(50, 50))
```

### 5.166.16 Метод Position:insertLineBreak

Метод предназначен для вставки перевода строки в указанную позицию текстового документа.

Внимание ! Метод может быть использован только в текстовом редакторе.

**Пример**

```
local rng = document:getRange()
local end_pos = rng:getEnd()
end_pos:insertLineBreak()
```

### 5.166.17 Метод Position:insertPageBreak

Метод предназначен для вставки разрыва страницы в указанную позицию текстового документа.

Внимание ! Метод может быть использован только в текстовом редакторе.

**Пример**

```
local rng = document:getRange()
local end_pos = rng:getEnd()
end_pos:insertPageBreak()
```

### 5.166.18 Метод Position:insertSectionBreak

Вставляет разрыв раздела в текущую позицию. В качестве параметра выступает тип SectionBreakType. При вызове без параметра используется значение

SectionBreakType_NextPage.

Внимание ! Данный метод может быть использован только в текстовом документе.

**Примеры**

```
local range = document:getRange()
local endPosition = range:getEnd()
endPosition:insertSectionBreak()
```

```
endPosition:insertSectionBreak(DocumentAPI.SectionBreakType_EvenPage)
```

### 5.166.19 Метод Position:insertTable

Метод предназначен для вставки таблицы с заданным числом строк и столбцов в заданное местоположение в документе. Возвращает объект таблицы.

Следует учитывать, что при вставке таблицы к ее имени автоматически добавляется порядковый номер, начинающийся с единицы. Таким образом, вызов

```
t = position:insertTable(3, 3, "Table")
```

приведет к созданию в текстовом документе таблицы с именем «Table1».

**Пример вставки таблицы в начало текстового документа**

```
local rng = document:getRange()
local begin_pos = rng:getBegin()
t = begin_pos:insertTable(3, 3, "Table")
```

**Пример вставки таблицы в конец текстового документа**

```
local rng = document:getRange()
local begin_pos = rng:getEnd()
t = begin_pos:insertTable(3, 3, "Table")
```

В табличном документе данный метод используется для вставки нового рабочего листа.

**Пример вставки нового листа в табличный документ**

```
local rng = document:getRange()
local end_pos = rng:getEnd()
t = end_pos:insertTable(3, 3, "Table")
```

### 5.166.20 Метод Position:insertText

Метод предназначен для вставки текстовой строки в заданное местоположение в документе.

**Пример**

```
local rng = document:getRange()
local begin_pos = rng:getBegin()
begin_pos:insertText("Текст в начале строки")
```

### 5.166.21 Метод Position:removeBackward

Метод удаляет `count` объектов (символов, картинок и т.д.) до текущей позиции.

**Пример**

```
document:getRange():getEnd():removeBackward(3)
```

### 5.166.22 Метод Position:removeForward

Метод удаляет `count` объектов (символов, картинок и т.д.) после текущей позиции.

**Пример**

```
document:getRange():getBegin():removeForward(3)
```

### 5.166.23 Метод Position:__eq

Метод используется для определения эквивалентности значений двух местоположений в документе.

**Пример**

```
print(document:getRange():getBegin():__eq(document:getRange():getEnd()))
```

### 5.167 Таблица DocumentAPI.PredefinedTextStyle

Таблица `PredefinedTextStyle` определяет стандартный стиль абзаца. Используется в методах `TextStyles:create()` и `TextStyles:get()`.

Таблица 98 – Описание стандартных стилей абзацев

| Значение | Описание |
|---|---|
| PredefinedTextStyle_Normal | Обычный |
| PredefinedTextStyle_Heading1 | Заголовок 1 |
| PredefinedTextStyle_Heading2 | Заголовок 2 |
| PredefinedTextStyle_Heading3 | Заголовок 3 |
| PredefinedTextStyle_Heading4 | Заголовок 4 |
| PredefinedTextStyle_Heading5 | Заголовок 5 |
| PredefinedTextStyle_Heading6 | Заголовок 6 |
| PredefinedTextStyle_Heading7 | Заголовок 7 |
| PredefinedTextStyle_Heading8 | Заголовок 8 |
| PredefinedTextStyle_Heading9 | Заголовок 9 |
| PredefinedTextStyle_Title | Название документа |
| PredefinedTextStyle_Subtitle | Подзаголовок |
| PredefinedTextStyle_HeaderFooter | Колонтитул |
| PredefinedTextStyle_Footnote | Сноска |
| PredefinedTextStyle_Endnote | Концевая сноска |
| PredefinedTextStyle_Contents1 | Содержание 1 |
| PredefinedTextStyle_Contents2 | Содержание 2 |
| PredefinedTextStyle_Contents3 | Содержание 3 |
| PredefinedTextStyle_Contents4 | Содержание 4 |
| PredefinedTextStyle_Contents5 | Содержание 5 |
| PredefinedTextStyle_Contents6 | Содержание 6 |
| PredefinedTextStyle_Contents7 | Содержание 7 |
| PredefinedTextStyle_Contents8 | Содержание 8 |
| PredefinedTextStyle_Contents9 | Содержание 9 |
| PredefinedTextStyle_Hyperlink | Ссылка |

### 5.168 Таблица DocumentAPI.PrintDocumentResult

В таблице 99 представлены коды, возвращаемые после печати (см. EditorAPI.showPrintDialog()).

Таблица 99 – Коды, возвращаемые после печати

| Значение | Описание |
|---|---|
| DocumentAPI.PrintDocumentResult_Success | Печать прошла успешно |
| DocumentAPI.PrintDocumentResult_OneCopyPrinted | Напечатана только одна копия из заданных |
| DocumentAPI.PrintDocumentResult_CancelPrinting | Печать была отменена |
| DocumentAPI.PrintDocumentResult_NoPrinter | Принтер не найден |
| DocumentAPI.PrintDocumentResult_BlankDocument | На печать отправлен пустой документ |

### 5.169 Таблица DocumentAPI.PrintSettings

Таблица DocumentAPI.PrintSettings представляет установки, используемые при печати документов. Описание полей таблицы DocumentAPI.PrintSettings представлено в таблице 100. Используется в EditorAPI.printDocument.

Таблица 100 – Описание полей таблицы DocumentAPI.PrintSettings

| Поле | Тип | Описание |
|---|---|---|
| PrintSettings.printerName | string | Имя используемого принтера. Если не указано, то используется принтер по умолчанию. Если принтер с указанным именем недоступен, то возникает ошибка |
| PrintSettings.landscapeOrientation | bool | Если значение равно true, то размер страницы поворачивается на 90 градусов. В настоящее время используется только для рабочих таблиц |
| PrintSettings.leftMargin | number | Ширина левого поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц |

| Поле | Тип | Описание |
|---|---|---|
| PrintSettings.topMargin | number | Ширина верхнего поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц |
| PrintSettings.rightMargin | number | Ширина правого поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц |
| PrintSettings.bottomMargin | number | Ширина нижнего поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц |
| PrintSettings.parity | PageParity | Выбор страниц для печати |
| PrintSettings.firstPage | number | Номер первой страницы для печати |
| PrintSettings.lastPage | number | Номер последней страницы для печати |
| PrintSettings.printSelection | bool | Область печати. Значение по умолчанию false |
| PrintSettings.worksheetPrinterFitType | WorksheetPrinterFitType | Вариант масштабирования при печати табличных документов |
| PrintSettings.copies | number | Количество копий при печати |
| PrintSettings.collateCopies | bool | Если параметр имеет значение false, то печать каждой отдельной страницы будет повторена заданное количество копий раз до начала печати следующей страницы.<br>Если параметр имеет значение true, то все страницы печатаются до запуска печати следующей копии этих страниц. Значение по умолчанию false |

### 5.170 Таблица DocumentAPI.PrintTitles

Таблица PrintTitles представляет собой сквозной заголовок для печати документов. Сквозной заголовок – это диапазон ячеек, который будет отображаться на каждой странице распечатанного листа. Данная таблица используется в методах Table:getPrintTitles() и Table:setPrintTitles().

**Конструкторы**

```
PrintTitles()
```

```
PrintTitles(number top, number left, number bottom, number right)
```

Таблица 101 – Описание полей таблицы `PrintTitles`

| Поле | Тип | Описание |
|---|---|---|
| `PrintTitles.bottomRow` | `number` | Индекс нижней строки диапазона |
| `PrintTitles.leftColumn` | `number` | Индекс левой колонки диапазона |
| `PrintTitles.rightColumn` | `number` | Индекс правой колонки диапазона |
| `PrintTitles.topRow` | `number` | Индекс верхней строки диапазона |

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local titles = DocumentAPI.PrintTitles(4, 4, 7, 7)
sheet:setPrintTitles(titles)

EditorAPI.printDocument()
```

### 5.170.1 Метод PrintTitles:create

Метод создает новый сквозной заголовок с заданными координатами.

**Вызов**

```
static PrintTitles create(top, left, bottom, right)
```

**Параметры**

– `top`: индекс верхней строки диапазона.

– `left`: индекс левой колонки диапазона.

– `bottom`: индекс нижней строки диапазона.

– `right`: индекс правой колонки диапазона.

**Возвращает**

– сквозной заголовок, тип `PrintTitles`.

**Пример**

```
local titles = DocumentAPI.PrintTitles.create(4, 4, 7, 7)
print(titles:isRowsCols()) -- true
```

### 5.170.2 Метод PrintTitles:createPrintTitlesCols

Метод создает новый сквозной заголовок состоящий из целых колонок.

**Вызов**

```
static PrintTitles createPrintTitlesCols(first, last)
```

**Параметры**

– `first`: индекс первой колонки диапазона.

– `last`: индекс последней колонки диапазона.

**Возвращает**

– сквозной заголовок, тип `PrintTitles`.

**Пример**

```
local titles = DocumentAPI.PrintTitles.createPrintTitlesCols(0, 2)
print(titles:isCols()) -- true
```

### 5.170.3 Метод PrintTitles:createPrintTitlesRows

Метод создает новый сквозной заголовок состоящий из целых строк.

**Вызов**

```
static PrintTitles createPrintTitlesRows(first, last)
```

**Параметры**

– `first`: индекс первой строки диапазона.

– `last`: индекс последней строки диапазона.

**Возвращает**

– сквозной заголовок, тип `PrintTitles`.

**Пример**

```
local titles = DocumentAPI.PrintTitles.createPrintTitlesRows(0, 0)
print(titles:isRows()) -- true
```

### 5.170.4 Метод PrintTitles:isCols

Метод позволяет определить состоит ли сквозной заголовок только из целых колонок.

**Вызов**

```
bool isCols()
```

**Возвращает**

– `true`, если сквозной заголовок состоит из целых колонок, в ином случае – `false`.

### 5.170.5 Метод PrintTitles:isRows

Метод позволяет определить состоит ли сквозной заголовок только из целых строк.

**Вызов**

```
bool isRows()
```

**Возвращает**

– `true`, если сквозной заголовок состоит из целых строк, в ином случае – `false`.

### 5.170.6 Метод PrintTitles:isRowsCols

Метод позволяет определить состоит ли сквозной заголовок только из целых строк или столбцов.

**Вызов**

```
bool isRowsCols()
```

**Возвращает**

– `true`, если сквозной заголовок не состоит из целых строк или колонок, в ином случае – `false`.

## 5.171 Таблица DocumentAPI.Range

Таблица `DocumentAPI.Range` предоставляет доступ к диапазону документа. На рисунке 56 изображена объектная модель таблиц, относящихся к работе с диапазонами.

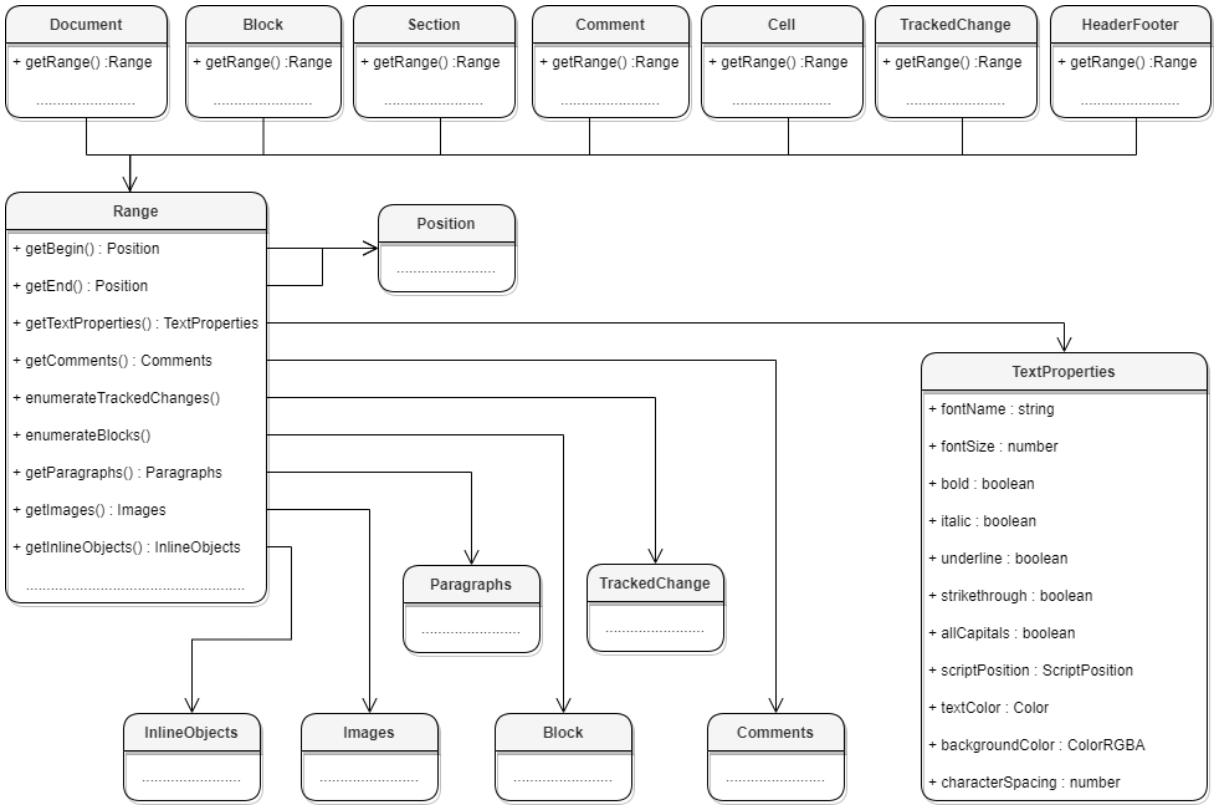



Рисунок 56 – Объектная модель для работы с таблицей `DocumentAPI.Range`

**Варианты получения диапазона для текстового документа**

```
-- диапазон всего документа
documentRange = document:getRange()
-- диапазон блока
block = document:getBlocks():getBlock(0)
blockRange = block:getRange()
-- диапазон секций
sections = document:getSections()
for section in sections:enumerate() do
    sectionRange = section:getRange()
end
-- диапазон комментариев
commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    commentRange = comment:getRange()
end
-- диапазон ячейки
table = document:getBlocks():getTable(0)
cell = table:getCell("B2")
cellRange = cell:getRange()
```

```
-- диапазон верхних колонтитулов
section = document:getBlocks():getBlock(0):getSection()
headers = section.getHeaders()
for header in headers:enumerate() do
    headerRange = header:getRange()
end
-- диапазон отслеживаемых изменений
local trackedChangesList = document:getRange():enumerateTrackedChanges()
for trackedChange in trackedChangesList do
    trackedChangeRange = trackedChange:getRange()
end
```

### 5.171.1 Конструктор DocumentAPI.Range

Для создания объекта DocumentAPI.Range вы можете использовать следующий конструктор:

```
documentRange = DocumentAPI.Range(begin, end)
```

**Параметры**

– begin: начальная позиция диапазона, тип DocumentAPI.Position;

– end: конечная позиция диапазона, тип DocumentAPI.Position.

### 5.171.2 Метод Range:copyInto

Метод копирует текущий диапазон в заданную позицию.

**Вызов**

```
 copyInto(destination)
```

**Параметры**

– destination: конечная позиция, тип Position.

**Пример**

Этот пример копирует первый абзац в конец документа:

```
local startPos = document:getRange():getBegin()
local paragraph = startPos:getCurrentRange(DocumentAPI.TextUnit_Paragraph)
local pos = document:getRange():getEnd()
paragraph:copyInto(pos)
```

### 5.171.3 Метод Range:enumerateBlocks

Предоставляет возможность итерации по блокам.

**Пример для текстового документа**

```
local range = document:getRange()
for block in range:enumerateBlocks() do
    print(block:getRange():extractText())
end
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
for block in range:enumerateBlocks() do
    print(block:getRange():extractText())
end
```

### 5.171.4 Метод Range:enumerateTrackedChanges

Предоставляет возможность итерации по отслеживаемым изменениям DocumentAPI.TrackedChange. Метод может быть использован только в текстовых документах.

**Пример**

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
  print(change:getRange():extractText())
end
```

### 5.171.5 Метод Range:extractText

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и другие объекты игнорируются.

**Пример для текстового документа**

```
local range = document:getRange() -- содержимое всего документа
local text = range:extractText()
print (text)
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки B2
print (range:extractText())
```

### 5.171.6 Метод Range:getBegin

Метод возвращает позицию в начале диапазона.

**Пример для текстового документа**

```
local range = document:getRange() -- содержимое всего документа
local pos = range:getBegin() -- в начало документа
pos:insertText("Привет")
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки B2
local pos = range:getBegin() -- в начало ячейки
pos:insertText("Привет")
```

### 5.171.7 Метод Range:getComments

Обеспечивает доступ к комментариям в диапазоне.

Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам. Если дат нет, то порядок комментариев не определен.

**Пример**

```
local comments = document:getRange():getComments()
for comment in comments:enumerate() do
  print(comment:getRange())
  print(comment:getText())
  print(comment:getInfo().author)
  print(comment:getInfo().timeStamp)
  print(comment:isResolved())
  print(comment:getReplies())
end
```

### 5.171.8 Метод Range:getContentEnd

Метод возвращает позицию в конце содержимого текущего диапазона.

**Вызов**

```
Position getContentEnd()
```

**Возвращает**

– позиция конца диапазона, тип Position.

Метод Range:getContentEnd может использоваться для добавления информации в конец содержимого абзаца/ячейки. Чтобы получить позицию конца диапазона с переходом к следующему абзацу/ячейке, вызовите метод Range:getEnd.

**Пример для текстового документа**

```
document:getRange():getBegin():insertText("Text")
document:getRange():getContentEnd():insertText("#")
print(document:getRange():extractText()) -- Text#\n
```

**Пример для табличного документа**

```
local cell = document:getBlocks():getTable(0):getCell("B2")
cell:setText("Text")
local cellEnd = cell:getRange():getContentEnd()
cellEnd:insertText("#")
print(cell:getRawValue()) -- Text#
```

### 5.171.9 Метод Range:getEnd

Метод возвращает позицию в конце диапазона, включая символ перехода на новую строку/ячейку.

Метод Range:getEnd может использоваться для перехода к следующему абзацу/ячейке. Чтобы получить позицию конца диапазона для добавления в него информации, вызовите метод Range:getContentEnd.

**Пример для текстового документа**

```
document:getRange():getBegin():insertText("First Paragraph")
local endDocPosition = document:getRange():getEnd()
endDocPosition:insertText("Second Paragraph")
print(document:getRange():extractText()) -- First Paragraph\nSecond Paragraph\n
```

**Пример для табличного документа**

```
local sheet = document:getBlocks():getTable(0)
sheet:getCell("B2"):setText("Text")
local cellEnd = sheet:getCell("B2"):getRange():getEnd()
cellEnd:insertText("#")
print(sheet:getCell("B2"):getRawValue()) -- Text
print(sheet:getCell("C2"):getRawValue()) -- #
```

### 5.171.10 Метод Range:getFootnotesEndnotes

Метод возвращает коллекцию сносок и концевых сносок в текущем диапазоне.

**Вызов**

```
FootnotesEndnotes getFootnotesEndnotes()
```

**Возвращает**

– коллекция сносок и концевых сносок, тип FootnotesEndnotes.

**Пример**

```
local range = document:getRange()
local notes = range:getFootnotesEndnotes()
for note in notes:enumerate() do
    print(note:getRange():extractText())
end
```

### 5.171.11 Метод Range:getImages

Обеспечивает доступ к изображениям (DocumentAPI.Image) в диапазоне.

Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

**Примеры**

```
local images = document:getRange():getImages()
for image in images:enumerate() do
  print(image:getFrame():getWrapType())
end
```

```
for image in EditorAPI.getSelection():getImages():enumerate() do
  print(image:getFrame():getWrapType())
end
```

### 5.171.12 Метод Range:getInlineObjects

Обеспечивает доступ к перечислению DocumentAPI.MediaObjects графических объектов диапазона.

Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

**Пример**

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
   print(mediaObject)
end
```

### 5.171.13 Метод Range:getParagraphs

Обеспечивает доступ к абзацам DocumentAPI.Paragraphs в диапазоне.

**Пример для текстового документа**

```
local paragraphs = document:getRange():getParagraphs()
for para in paragraphs:enumerate() do
    print(para:getRange():extractText())
end
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
    print(para:getRange():extractText())
end
```

### 5.171.14 Метод Range:getStyle

Метод возвращает стиль фрагментов текста в текущем диапазоне. Данный метод в основном специализируется на получении стилей из документов, созданных в редакторе

Microsoft Word. В большинстве случаев рекомендуется использовать метод Paragraph:getStyle() или Paragraph:getResolvedStyle().

**Вызов**

```
TextStyle getStyle()
```

**Возвращает**

– стиль фрагментов текста в текущем диапазоне, тип TextStyle.

– nil, если стиль не задан, или фрагменты диапазона содержат разные стили.

**Пример**

```
local range =
document:getRange():getBegin():getNextRange(DocumentAPI.TextUnit_Sentence)
print(range:getStyle():getName())
```

### 5.171.15 Метод Range:getTextProperties

Метод возвращает таблицу с текущими настройками форматирования для фрагмента текстового документа. Описание настроек форматирования осуществляется с помощью таблицы DocumentAPI.TextProperties.

**Пример для текстового документа**

```
local range = document:getRange()
local props = range:getTextProperties()
print(props.italic)
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local props = range:getTextProperties()
print(props.italic)
```

### 5.171.16 Метод Range:isContentLocked

Метод возвращает значение true, если изменения содержимого диапазона запрещены (см. Защита диапазона текстового документа).

**Пример для текстового документа**

```
local range = document:getRange() -- содержимое всего документа
if range:isContentLocked() then
  print("Документ содержит заблокированное содержимое")
end
```

**Пример для таблицы внутри текстового документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
if range:isContentLocked() then
  print("Ячейка содержит заблокированное содержимое")
end
```

### 5.171.17 Метод Range:lockContent

Метод запрещает изменения содержимого диапазона (см. Защита диапазона текстового документа).

Внимание ! Метод может быть использован только в текстовых документах.

**Пример для текстового документа**

```
local range = document:getRange() -- содержимое всего документа
range:lockContent()
```

**Пример для таблицы внутри текстового документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:lockContent()
```

### 5.171.18 Метод Range:moveInto

Метод перемещает текущий диапазон в заданную позицию.

**Вызов**

```
moveInto(destination)
```

**Параметры**

– `destination`: конечная позиция, тип `Position`.

**Пример**

Этот пример перемещает первое предложение в начало следующего абзаца:

```
local startPos = document:getRange():getBegin()
local sentence = startPos:getCurrentRange(DocumentAPI.TextUnit_Sentence)
local pos = startPos:getCurrentRange(DocumentAPI.TextUnit_Paragraph):getEnd()
sentence:moveInto(pos)
```

### 5.171.19 Метод Range:removeContent

Метод полностью удаляет содержимое диапазона.

**Пример для текстового документа**

```
local range = document:getRange() -- содержимое всего документа
range:removeContent()
print (range:extractText())
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:removeContent()
print (range:extractText())
```

### 5.171.20 Метод Range:replaceText

Метод заменяет содержимое фрагмента на указанный текст.

**Пример для текстового документа**

```
local range = document:getRange() -- содержимое всего документа
range:replaceText("Новый текст")
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки таблицы
range:replaceText("Новый текст")
```

### 5.171.21 Метод Range:setHyperlink

Метод `setHyperlink` вставляет ссылку в содержимое диапазона и заменяет его текст текстом ссылки.

**Вызов**

```
setHyperlink( url, label )
```

**Параметры**

– `url` – адрес ссылки;

– `label` – текст ссылки.

**Пример для текстового документа**

```
local range = document:getRange()
range:setHyperlink("https://testhyperlink.com", "Hyperlink")
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
range:setHyperlink("https://testhyperlink.com", "Hyperlink")
print(cell:getFormattedValue())
```

### 5.171.22 Метод Range:setTextProperties

Метод применяет настройки форматирования `DocumentAPI.TextProperties` для диапазона.

**Пример для текстового документа**

```
local range = document:getRange()
local props = range:getTextProperties()
props.italic = true
range:setTextProperties(props) -- текстовый фрагмент оформлен курсивом
```

**Пример для табличного документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local props = range:getTextProperties()
props.italic = true
range:setTextProperties(props)
```

### 5.171.23 Метод Range:unlockContent

Метод разрешает изменения содержимого диапазона (см. Защита диапазона текстового документа).

Внимание ! Метод может быть использован только в текстовых документах.

**Пример для текстового документа**

```
local range = document:getRange() -- содержимое всего документа
range:unlockContent()
```

**Пример для таблицы внутри текстового документа**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:unlockContent()
```

### 5.172 Таблица DocumentAPI.RangeBorders

Таблица `DocumentAPI.RangeBorders` оставлена для совместимости. Вместо нее необходимо использовать таблицу `DocumentAPI.Borders`.

### 5.173 Таблица DocumentAPI.RectU

Таблица `DocumentAPI.RectU` представляет описание прямоугольной области. Описание полей таблицы `DocumentAPI.RectU` представлено в таблице 102.

Таблица 102 – Описание полей таблицы `DocumentAPI.RectU`

| Поле | Тип | Описание |
|---|---|---|
| `RectU.topLeft` | `PointU` | Координата левой верхней вершины прямоугольника |
| `RectU.bottomRight` | `PointU` | Координата правой нижней вершины прямоугольника |

**Пример**

```
local rect = DocumentAPI.RectU(2, 3, 4, 5)
print("tlx=", rect.topLeft.x, ", tly=", rect.topLeft.y, ", brx=",
rect.bottomRight.x, ", bry=", rect.bottomRight.y)  --(tlx = 2.0, tly = 3.0, brx =
4.0, bry = 5.0)
```

### 5.173.1 Метод RectU:toString

Возвращает информацию о прямоугольной области в виде строкового описания координат `[topLeft: (x: <value>, y: <value>), bottomRight: (x: <value>, y: <value>)]`.

**Пример**

```
local point = DocumentAPI.RectU(2, 3, 4, 5)
print(point:toString())  --[topLeft: (x: 2.0, y: 3.0), bottomRight: (x: 4.0, y: 5.0)]
```

### 5.174 Таблица DocumentAPI.ScaleFrom

В таблице 103 представлены позиции объекта, остающиеся неизменными при масштабировании объекта. Используется в `AbsoluteFrame:scale()`.

Таблица 103 – Неизменные позиции объекта при масштабировании

| Значение | Позиция |
|---|---|
| `DocumentAPI.ScaleFrom_BottomRight` | Правый нижний угол |
| `DocumentAPI.ScaleFrom_BottomLeft` | Левый нижний угол |
| `DocumentAPI.ScaleFrom_TopLeft` | Левый верхний угол |
| `DocumentAPI.ScaleFrom_TopRight` | Правый верхний угол |

### 5.175 Таблица DocumentAPI.ScientificCellFormatting

Таблица содержит параметры для экспоненциального формата ячеек таблицы. Данная таблица используется в качестве аргумента метода `Cell:setFormat()`.

Таблица 104 – Описание полей таблицы `DocumentAPI.ScientificCellFormatting`

| Поле | Тип | Описание |
|---|---|---|
| `ScientificCellFormatting.decimalPlaces` | `number` | Количество десятичных позиций |
| `ScientificCellFormatting.minExponentDigits` | `number` | Минимальное количество позиций экспоненты |

**Пример**

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("B2")

local scientificCellFormatting = DocumentAPI.ScientificCellFormatting()
scientificCellFormatting.decimalPlaces = 2
scientificCellFormatting.minExponentDigits = 3

cell:setFormat(scientificCellFormatting)
print(cell:getFormattedValue())
```

### 5.176 Таблица DocumentAPI.Script

Таблица `DocumentAPI.Script` предназначена для управления отдельной макрокомандой. Таблица содержит поля `Name` и `Body`.

#### 5.176.1 Метод Script:getBody

Метод возвращает текст макрокоманды в виде строки.

**Пример**

```
local scripts = document:getScripts()
local script = scripts:getScript("Enumerate scripts for document")
local scriptBody = script:getBody()
print(scriptBody)
```

#### 5.176.2 Метод Script:getName

Метод возвращает имя макрокоманды.

**Пример**

```
local scripts = document:getScripts()
local sc = scripts:getScript("Enumerate scripts for document")
print(sc:getName())
```

### 5.176.3 Метод Script:setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

**Пример**

```
local scripts = document:getScripts()
local script = scripts:getScript("Enumerate scripts for document")
script:setBody("local scripts = document:getScripts()\nfor script in
scripts:enumerate() do\nprint(script:getName())\nend")
```

### 5.176.4 Метод Script:setName

Метод устанавливает имя для макрокоманды.

**Пример**

```
local scripts = document:getScripts()
local sc = scripts:getScript("Enumerate scripts for document")
sc:setName("Enumerate scripts for current document")
```

## 5.177 Таблица DocumentAPI.Scripting

Таблица `DocumentAPI.Scripting` может быть получена путем вызова `DocumentAPI.createScripting()` и содержит метод `runScript`, который используется для запуска макрокоманды.

### 5.177.1 Метод Scripting:runScript

Метод предназначен для запуска макрокоманды, хранящейся в документе. В качестве аргумента передается имя макрокоманды.

**Пример**

```
scripting = DocumentAPI.createScripting(document)
scripting:runScript("Enumerate scripts for document")
```

## 5.178 Таблица DocumentAPI.ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 105. Используется в качестве поля `scriptPosition` таблицы `DocumentAPI.TextProperties`.

Таблица 105 – Типы надстрочного и подстрочного форматирования

| Значение | Описание |
|---|---|
| DocumentAPI.ScriptPosition_SuperScript | Надстрочный знак (верхний индекс) |
| DocumentAPI.ScriptPosition_SubScript | Подстрочный знак (нижний индекс) |
| DocumentAPI.ScriptPosition_NormalScript | Без указания индекса |

**Пример**

```
local props = DocumentAPI.TextProperties()
props.scriptPosition = DocumentAPI.ScriptPosition_SuperScript
range:setTextProperties(props)
```

### 5.179 Таблица DocumentAPI.Scripts

Таблица DocumentAPI.Scripts предоставляет доступ к списку макрокоманд документа. Коллекцию макрокоманд DocumentAPI.Scripts можно получить из документа посредством вызова метода document:getScripts().

**Пример**

```
local scripts = document:getScripts()
for script in scripts:enumerate() do
   print(script:getName())
   print(script:getBody())
end
```

#### 5.179.1 Метод Scripts:enumerate

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

**Пример**

```
for script in document:getScripts():enumerate() do
    print(script:getName())
end
```

#### 5.179.2 Метод Scripts:getScript

Метод возвращает таблицу DocumentAPI.Script, описывающую макрокоманду. В качестве аргумента используется имя макрокоманды.

**Пример**

```
local scripts = document:getScripts()
local script = scripts:getScript("Enumerate scripts for document")
print(script:getName())
```

### 5.179.3 Метод Scripts:removeScript

Метод удаляет макрокоманду из текущего документа. В качестве аргумента используется имя макрокоманды.

**Пример**

```
local scripts = document:getScripts()
scripts:removeScript("Enumerate scripts for document")
```

### 5.179.4 Метод Scripts:setScript

Метод добавляет макрокоманду в текущий документ. Если макрокоманда с таким именем уже существует, будет обновлено ее содержимое.

**Пример**

```
local scripts = document:getScripts()
local script_name = "Enumerate scripts for document"
local script_code = "local scripts = document:getScripts()\nfor script in
scripts:enumerate() do\nprint(script:getName())\nend"
scripts:setScript(script_name, script_code)
```

## 5.180 Таблица DocumentAPI.Search

Таблица `DocumentAPI.Search` предоставляет доступ к механизму поиска фрагментов документа, открытого в редакторе текста или таблиц.

### 5.180.1 Метод Search:findText

Метод выполняет поиск строки без учета регистра во всем документе или выбранном диапазоне документа. Результат возвращается в виде диапазона `DocumentAPI.Range`, содержащего искомый фрагмент.

Если строка не обнаружена, возвращается пустая таблица.

Возможно использование следующих вариантов метода:

```
Range findText(String text)
```

```
Range findText(String text, CaseSensitive caseSensitive)
Range findText(String text, Range range)
Range findText(String text, Range range, CaseSensitive caseSensitive)
Range findText(String text, CellRange cellRange)
Range findText(String text, CellRange cellRange, CaseSensitive caseSensitive)
Range findText(String text, Table tbl)
Range findText(String text, Table tbl, CaseSensitive caseSensitive)
```

**Параметры**

– `text` – строка для поиска;

– `caseSensitive` – поиск с учетом или без учета регистра, тип `DocumentAPI.CaseSensitive`;

– `range` – диапазон, в котором будет производиться поиск, тип `DocumentAPI.Range`;

– `cellRange` – диапазон ячеек, в котором будет производиться поиск, тип `DocumentAPI.CellRange`;

– `table` – таблица, в которой будет производиться поиск, тип `DocumentAPI.Table`.

**Пример**

```
search = DocumentAPI.createSearch(document)
-- Поиск по всему документу
ranges = search:findText("English")
for occurrence in ranges do
    print(occurrence:extractText())
end
```

Дополнительные примеры использования метода `Search:findText` приведены в разделе Поиск в документе.

### 5.181 Таблица DocumentAPI.Section

Таблица `DocumentAPI.Section` представляет собой раздел в документе.

#### 5.181.1 Метод Section:getFooters

Метод возвращает коллекцию `DocumentAPI.HeadersFooters` нижних колонтитулов данного раздела.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local footers = section:getFooters()
for footer in footers:enumerate() do
    if (footer:getType() == DocumentAPI.HeaderFooterType_Header) then
       print("Header") else print("Footer")
    end
end
```

### 5.181.2 Метод Section:getHeaders

Метод возвращает коллекцию DocumentAPI.HeadersFooters верхних колонтитулов данного раздела.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    if (header:getType() == DocumentAPI.HeaderFooterType_Header) then
       print("Header") else print("Footer")
    end
end
```

### 5.181.3 Метод Section:getPageOrientation

Метод возвращает ориентацию страниц раздела.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local orientation = section:getPageOrientation()
print(orientation)
```

### 5.181.4 Метод Section:getPageProperties

Метод возвращает параметры страниц раздела DocumentAPI.PageProperties.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local properties = section:getPageProperties()
print(properties.width)
```

```
print(properties.height)
print(properties.margins.left)
print(properties.margins.top)
```

### 5.181.5 Метод Section:getRange

Метод возвращает диапазон DocumentAPI.Range в документе, соответствующий данному разделу.

**Пример**

```
local sections = document:enumerateSections()
for section in sections do
  print(section:getRange():extractText())
end
```

### 5.181.6 Метод Section:setPageOrientation

Метод задает ориентацию страниц раздела.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
section:setPageOrientation(DocumentAPI.PageOrientation_Landscape)
local orientation = section:getPageOrientation()
print(orientation)
```

### 5.181.7 Метод Section:setPageProperties

Метод устанавливает параметры DocumentAPI.PageProperties страниц, находящихся в разделе.

**Пример**

```
local section = document:getBlocks():getBlock(0):getSection()
local properties = section:getPageProperties()
properties.width = 100
properties.height = 200
properties.margins.left = 10
section:setPageProperties(properties)
```

### 5.182 Таблица DocumentAPI.SectionBreakType

Перечисление `SectionBreakType` содержит варианты разрыва страниц, используется в `Position:insertSectionBreak()`.

Таблица 106 – Варианты разрыва страниц

| Значение | Описание |
|---|---|
| `SectionBreakType_NextPage` | Следующий раздел начинается с новой страницы |
| `SectionBreakType_Continuous` | Следующий раздел продолжается на текущей странице без вставки разрыва страницы |
| `SectionBreakType_EvenPage` | Следующий раздел начинается на ближайшей четной странице |
| `SectionBreakType_OddPage` | Следующий раздел начинается на ближайшей нечетной странице |

### 5.183 Таблица DocumentAPI.Sections

Таблица `DocumentAPI.Sections` представляет интерфейс для доступа к коллекции секций документа. Может быть получена посредством вызова метода `document:getSections()`. Описание секции см. в разделе `DocumentAPI.Section`.

#### 5.183.1 Метод Sections:enumerate

Метод возвращает коллекцию секций документа.

**Пример**

```
local sections = document:getSections()
for section in sections:enumerate() do
  local properties = section:getPageProperties()
  print(properties.width)
  print(properties.height)
end
```

### 5.184 Таблица DocumentAPI.Shape

Таблица `Shape` представляет собой фигуру, содержит методы для установки и получения ее свойств `DocumentAPI.ShapeProperties`.

### 5.184.1 Метод Shape:getShapeProperties

Метод возвращает свойства фигуры DocumentAPI.ShapeProperties.

**Пример**

```
local shape = document:getBlocks():getShape(0)
local shape_properties = shape:getShapeProperties()
```

### 5.184.2 Метод Shape:setShapeProperties

Метод устанавливает свойства фигуры DocumentAPI.ShapeProperties.

**Пример**

```
local shape = document:getBlocks():getShape(0)
local shape_properties = shape:getShapeProperties()
shape_properties.verticalAlignment = DocumentAPI.VerticalAlignment_Center
shape:setShapeProperties(shape_properties)
```

### 5.185 Таблица DocumentAPI.ShapeProperties

Таблица описывает свойства фигуры. Используется в методах Shape:getShapeProperties() и Shape:setShapeProperties().

Таблица 107 – Описание полей таблицы ShapeProperties

| Поле | Тип | Описание |
|---|---|---|
| ShapeProperties.verticalAlignment | VerticalAlignment | Вертикальное выравнивание |
| ShapeProperties.borderProperties | LineProperties | Свойства границ фигуры |
| ShapeProperties.fill | Fill | Свойства заполнения фигуры |
| ShapeProperties.shapeTextLayout | ShapeTextLayout | Свойства текста внутри фигуры |

### 5.186 Таблица DocumentAPI.ShapeTextLayout

Таблица `DocumentAPI.ShapeTextLayout` описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 108. Используется в таблице `DocumentAPI.ShapeProperties`.

Таблица 108 – Свойства текста внутри фигуры

| Значение | Описание |
|---|---|
| `ShapeTextLayout_DoNotAutoFit` | Размещение текста в фигуре по умолчанию |
| `ShapeTextLayout_FitShapeExtentToText` | Расширение фигуры под текст |
| `ShapeTextLayout_FitTextToShape` | Заполнение фигуры текстом |

### 5.187 Таблица DocumentAPI.SizeU

Таблица `DocumentAPI.SizeU` представляет размер объекта в двухмерном пространстве.

Таблица 109 – Описание полей таблицы `DocumentAPI.SizeU`

| Поле | Тип | Описание |
|---|---|---|
| `SizeU.width` | number | Ширина объекта |
| `SizeU.height` | number | Высота объекта |

**Пример**

```
local size = DocumentAPI.SizeU(2, 3)
print("width =", size.width, ", height =", size.height)  --(width = 2,0, height = 3,0)
```

#### 5.187.1 Метод SizeU:toString

Возвращает информацию о размерах в виде строкового значения формата `(width: <value>, height: <value>)`.

**Пример**

```
local size = DocumentAPI.SizeU(2, 3)
print(size:toString())  --(width: 2.0, height: 3.0)
```

### 5.188 Таблица DocumentAPI.SortingConditions

Представляет собой коллекцию условий для сортировки строк. Таблица `SortingConditions` используется в методе `CellRange:sort()`.

Конструктор по умолчанию:

```
DocumentAPI.SortingConditions()
```

Конструктор копирования:

```
DocumentAPI.SortingConditions(sortingConditions)
```

Порядок условий в коллекции определяет порядок применения сортировки. Например, можно задать дополнительное условие для другого столбца, чтобы отсортировать строки с одинаковыми значениями в первом столбце.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local range = sheet:getCellRange("A1:C11")

local conditions = DocumentAPI.SortingConditions()
conditions:add(0, DocumentAPI.SortingDirection_Descending)
conditions:add(1, DocumentAPI.SortingDirection_Ascending)

range:sort(conditions)
```

#### 5.188.1 Метод SortingConditions:add

Метод добавляет заданное условие сортировки в коллекцию.

**Вызов**

```
add(index, sortingDirection)
```

**Параметры**

– `index`: индекс столбца диапазона для применения сортировки.

– `sortingDirection`: порядок сортировки, тип `SortingDirection`.

**Пример**

```
local conditions = DocumentAPI.SortingConditions()
conditions:add(0, DocumentAPI.SortingDirection_Descending)
conditions:add(1, DocumentAPI.SortingDirection_Ascending)
```

### 5.188.2 Метод SortingConditions:clear

Метод очищает коллекцию условий.

## 5.189 Таблица DocumentAPI.SortingDirection

В таблице 110 представлены варианты порядка сортировки строк. Используется в методе SortingConditions:add().

Таблица 110 – Варианты порядка сортировки строк

| Значение | Описание |
|---|---|
| SortingDirection_Ascending | Сортировка по возрастанию |
| SortingDirection_Descending | Сортировка по убыванию |

**Пример**

```
local conditions = DocumentAPI.SortingConditions()
conditions:add(0, DocumentAPI.SortingDirection_Descending)
conditions:add(1, DocumentAPI.SortingDirection_Ascending)
```

## 5.190 Таблица DocumentAPI.StyledTableRange

Таблица StyledTableRange представляет собой умную таблицу в табличном документе. Используется в методе Table:getStyledTableRange().

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local smartTable = sheet:getStyledTableRange("SmartTable1")
local tableCells = smartTable:getCellRange()
local tableFiltersRange = smartTable:getFiltersRange()
```

### 5.190.1 Метод StyledTableRange:getCellRange

Метод возвращает диапазон ячеек, который принадлежит умной таблице. Данный диапазон также включает строки заголовков и итогов.

**Вызов**

```
CellRange getCellRange()
```

**Возвращает**

– диапазон ячеек, тип CellRange.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local smartTable = sheet:getStyledTableRange("SmartTable1")
local tableCells = smartTable:getCellRange()
print(tableCells:getTableRange():toString())
```

### 5.190.2 Метод StyledTableRange:getFiltersRange

Метод возвращает диапазон фильтрации умной таблицы.

**Вызов**

```
FiltersRange getFiltersRange()
```

**Возвращает**

– диапазон фильтрации, тип FiltersRange.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local smartTable = sheet:getStyledTableRange("SmartTable1")
local tableFiltersRange = smartTable:getFiltersRange()

local filters = DocumentAPI.TableFilters()
local filter1 = DocumentAPI.ConditionalTableFilter()
filter1:begins("T")
local filter2 = DocumentAPI.ConditionalTableFilter()
filter2:begins("M")
filters:setFilter(0, filter1)
filters:setFilter(1, filter2)

tableFiltersRange:setFilters(filters)
```

## 5.191 Таблица DocumentAPI.Table

Таблица `DocumentAPI.Table` предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 57).

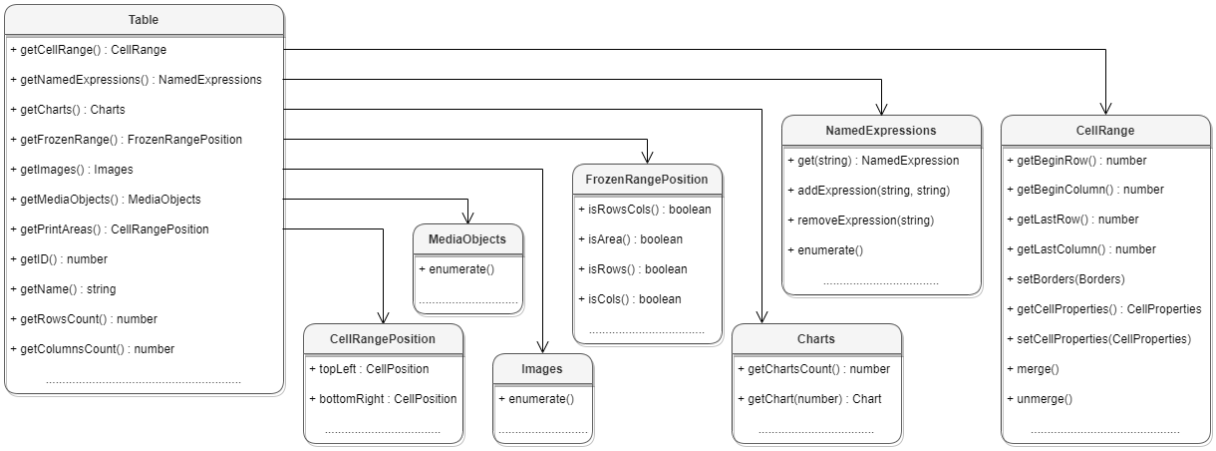



Рисунок 57 – Структура полей таблицы `DocumentAPI.Table`

### 5.191.1 Метод Table:clearColumnGroups

Метод предназначен для очистки группированных столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

**Вызов**

```
clearColumnGroups(columnIndex, columnsCount)
```

**Параметры**

- `columnIndex` – индекс столбца, начиная с которого будет начата очистка групп;
- `columnsCount` – количество столбцов для очистки групп.

### 5.191.2 Метод Table:clearRowGroups

Метод предназначен для очистки группированных строк таблицы, начиная с заданного индекса. Индексация строк начинается с нуля.

**Вызов**

```
clearRowGroups(rowIndex, rowCount)
```

**Параметры**

- `rowIndex` – индекс строки, начиная с которой будет начата очистка групп;
- `rowCount` – количество строк для очистки групп.

### 5.191.3 Метод Table:createFiltersRange

Метод `Table:createFiltersRange` задает диапазон, который используется как диапазон фильтрации.

В качестве параметра используется диапазон ячеек типа `CellRangePosition`. Метод возвращает `FiltersRange`.

Разрешен только один диапазон фильтрации на таблицу. Это означает, что данный метод удаляет ранее определенный диапазон фильтрации. При этом есть исключение: если новый диапазон начинается с той же позиции, предыдущие фильтры будут сохранены.

Диапазон фильтрации должен включать дополнительную строку, которая используется как заголовок таблицы. Эта строка никогда не фильтруется.

Метод может быть использован только в табличных документах.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local cellRange = DocumentAPI.CellRangePosition(1, 1, 8, 2)
local filtersRange = sheet:createFiltersRange(cellRange)
```

Более подробный пример приведен в разделе Работа с фильтрами.

### 5.191.4 Метод Table:duplicate

Для создания копии листа в табличном документе используется метод `duplicate`. Созданная копия листа размещается после копируемого листа. Метод может быть использован только в табличном документе.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:duplicate()
```

### 5.191.5 Метод Table:find

Метод выполняет поиск ячеек, соответствующих заданному запросу.

**Вызов**

```
function find(string, settings)
```

**Параметры**

– `string`: поисковый запрос, тип `string`.

– `settings`: (необязательный) параметры поиска, тип `TableSearchSettings`.

**Возвращает**

– список ячеек, соответствующих поисковому запросу.

**Пример**

```
local sheet = document:getBlocks():getTable(0)

local searchProps = DocumentAPI.TableSearchSettings()
searchProps.caseSensitive = DocumentAPI.CaseSensitive_No
searchProps.matchBehaviour = DocumentAPI.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = DocumentAPI.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = true

local results = sheet:find("*eye", searchProps)
for cell in results do
    print(cell:getFormattedValue()) -- Steeleye Stout
end
```

### 5.191.6 Метод Table:freeze

Метод `freeze` закрепляет заданную область `DocumentAPI.FrozenRangePosition` таблицы. Может быть использован только в табличном документе.

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local tbl = document:getBlocks():getTable(0)
tbl:freeze(frozenRangePosition)
print(tbl:getFrozenRange():isCols())
```

### 5.191.7 Метод Table:getCell

Метод позволяет получить доступ к отдельной ячейке таблицы. В качестве аргумента может выступать текстовое представление адреса ячейки, либо экземпляр таблицы `DocumentApi.CellPosition`.

**Примеры**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
print(cell:getFormattedValue())
```

```
local cellPosition = DocumentAPI.CellPosition(2, 1)
local cell = tbl:getCell(cellPosition)
print(cell:getFormattedValue())
```

### 5.191.8 Метод Table:getCellRange

Метод позволяет получить доступ к диапазону ячеек таблицы `DocumentAPI.CellRange`. В качестве аргумента может использоваться строка, описывающая диапазон (`"A1:C4"`), либо объект типа `DocumentAPI.CellRangePosition`.

**Примеры**

```
local table = document:getBlocks():getTable(0)
local range = table:getCellRange("A1:C4")
for cell in range:enumerate() do
   print(cell:getFormattedValue())
end
```

```
local table = document:getBlocks():getTable(0)
local range = table:getCellRange(DocumentAPI.CellRangePosition(0, 0, 2, 2))
for cell in range:enumerate() do
   print(cell:getFormattedValue())
end
```

### 5.191.9 Метод Table:getCharts

Для получения списка диаграмм (`DocumentAPI.Charts`) таблицы используется метод `Table:getCharts`.

**Пример**

```
for tbl in document:getBlocks():enumerateTables() do
  print(tbl:getCharts():getChartsCount())
end
```

### 5.191.10 Метод Table:getColumnsCount

Метод позволяет получить количество столбцов таблицы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getColumnsCount())
```

### 5.191.11 Метод Table:getColumnWidth

Метод возвращает ширину столбца таблицы в пунктах (1/72 дюйма).

**Вызов**

```
number getColumnWidth(columnIndex)
```

**Параметры**

– `columnIndex` – индекс столбца в таблице, для которого возвращается значение ширины. Индексация столбцов начинается с нуля.

**Возвращает**

– ширина столбца в пунктах (1/72 дюйма).

**Пример**

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
width = tbl:getColumnWidth(1)
```

Задать ширину столбца таблицы позволяет метод `Table:setColumnWidth`.

### 5.191.12 Метод Table:getConditionalFormatRules

Метод позволяет получить коллекцию правил условного форматирования для текущего листа.

**Вызов**

```
ConditionalFormatTableRules getConditionalFormatRules()
```

**Возвращает**

– коллекция правил условного форматирования для листа, тип `ConditionalFormatTableRules`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
```

```
local topBottomStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
topBottomStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("F2:F12")
local cellRangePosition = cellRange:getTableRange()

local topBottomOperator =
DocumentAPI.createTopBottomConditionalFormatOperator(DocumentAPI.ConditionalForma
tTopBottomCondition_Top, 20, true)

local topBottomRule = DocumentAPI.ConditionalFormatRule(topBottomOperator,
topBottomStyle, cellRangePosition, false)
rules:addRule(topBottomRule)
```

### 5.191.13 Метод Table:getFiltersRange

Метод `Table:getFiltersRange` возвращает текущий диапазон фильтрации, принадлежащий таблице. Рабочий лист табличного документа может содержать только один диапазон фильтрации. Чтобы получить диапазон фильтрации умной таблицы, используйте метод `StyledTableRange:getFiltersRange`.

Метод возвращает `FiltersRange`, если диапазон фильтрации существует.

Метод может быть использован только в табличных документах.

**Пример**

```
local filtersRange = sheet:getFiltersRange()
local cellRange = filtersRange:getCellRange()
print(cellRange:getBeginRow() .. ", " .. cellRange:getLastRow())
```

Более подробный пример приведен в разделе Работа с фильтрами.

### 5.191.14 Метод Table:getFrozenRange

Существует возможность закрепления диапазона строк и столбцов. Такие диапазоны всегда остаются видимыми на экране в случае, когда пользователь осуществляет навигацию по таблице.

Метод `getFrozenRange` возвращает закрепленный диапазон `DocumentAPI.FrozenRangePosition`.

**Пример**

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local tbl = document:getBlocks():getTable(0)
tbl:freeze(frozenRangePosition)
print(tbl:getFrozenRange():isCols())
```

### 5.191.15 Метод Table:getImages

Для получения списка изображений (DocumentAPI.Images) таблицы используется метод `Table:getImages`.

**Пример**

```
tbl = document:getBlocks():getTable(0)
images = tbl:getImages()

for image in images:enumerate() do
  print(image)
end
```

### 5.191.16 Метод Table:getLabelColor

Метод возвращает цвет ярлыка листа. Ярлыки используются для визуального разделения листов табличного документа.

**Вызов**

```
Color getLabelColor()
```

**Возвращает**

– цвет ярлыка листа, тип Color.

– `nil`, если цвет не задан.

**Пример**

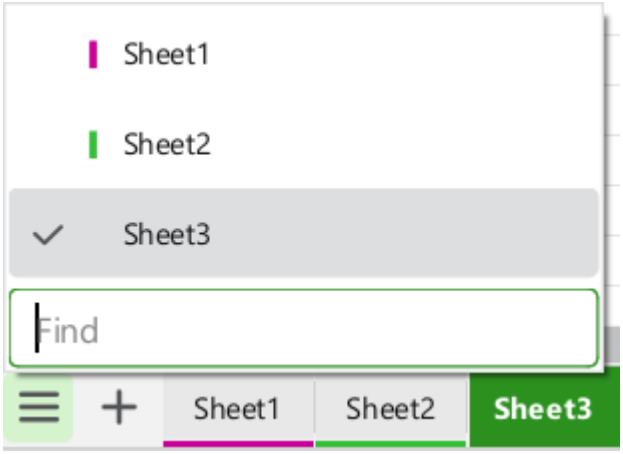



Рисунок 58 – Ярлыки листов табличного документа

```
local sheet = document:getBlocks():getTable(1)
local rgba = sheet:getLabelColor():getRGBAColor()
print("r:", rgba.r, ", g:", rgba.g, ", b:", rgba.b, ", a:", rgba.a)
```

### 5.191.17 Метод Table:getLastNonEmptyCellInColumn

Возвращает индекс последней заполненной ячейки в столбце. Последней заполненной считается ячейка, после которой все ячейки пустые.

**Вызов**

```
number getLastNonEmptyCellInColumn(columnIndex)
```

**Параметры**

– `columnIndex`: индекс столбца.

**Возвращает**

– индекс строки, которая содержит последнюю заполненную ячейку.

– `nil`, если все ячейки в столбце пустые.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
print(sheet:getLastNonEmptyCellInColumn(0)) -- 10
print(sheet:getLastNonEmptyCellInRow(0)) -- 2
```

### 5.191.18 Метод Table:getLastNonEmptyCellInRow

Возвращает индекс последней заполненной ячейки в строке. Последней заполненной

считается ячейка, после которой все ячейки пустые.

**Вызов**

```
number getLastNonEmptyCellInRow(rowIndex)
```

**Параметры**

– `rowIndex`: индекс строки.

**Возвращает**

– индекс столбца, который содержит последнюю заполненную ячейку.

– `nil`, если все ячейки в строке пустые.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
print(sheet:getLastNonEmptyCellInColumn(0)) -- 10
print(sheet:getLastNonEmptyCellInRow(0)) -- 2
```

### 5.191.19 Метод Table:getMediaObjects

Для получения списка медиаобъектов (DocumentAPI.MediaObjects) таблицы используется метод `Table:getMediaObjects`.

**Пример**

```
tbl = document:getBlocks():getTable(0)
mediaObjects = tbl:getMediaObjects()

for mediaObject in mediaObjects:enumerate() do
  print(mediaObject)
end
```

### 5.191.20 Метод Table:getName

Метод позволяет получить наименование листа табличного документа.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getName())
```

### 5.191.21 Метод Table:getNamedExpressions

Метод используется для получения списка именованных диапазонов DocumentAPI:NamedExpressions.

### 5.191.22 Метод Table:getPrintAreas

Метод `Table:getPrintAreas` возвращает текущие области печати - вектор элементов DocumentAPI.CellRangePosition.

**Пример**

```
tbl = document:getBlocks():getTable(0)
tbl:setPrintArea(DocumentAPI.CellRangePosition(0, 0, 5, 5))

printAreas = tbl:getPrintAreas()
print(printAreas[0]:toString())
```

### 5.191.23 Метод Table:getPrintTitles

Метод возвращает сквозной заголовок для текущего листа.

**Вызов**

```
PrintTitles getPrintTitles()
```

**Возвращает**

– сквозной заголовок, тип PrintTitles.

– `nil`, если сквозной заголовок не установлен.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local printTitles = sheet:getPrintTitles()
print(printTitles:isCols())
print(printTitles:isRows())
print(printTitles:isRowsCols())
```

### 5.191.24 Метод Table:getProtectionProperties

Метод возвращает параметры защиты от изменений листа табличного документа (см. Защита листа табличного документа).

**Вызов**

```
TableProtectionProperties getProtectionProperties()
```

**Возвращает**

– TableProtectionProperties: свойства защиты листа документа (nil, если защита листа не установлена).

### 5.191.25 Метод Table:getRowHeight

Метод возвращает высоту строки таблицы в пунктах (1/72 дюйма).

**Вызов**

```
number getRowHeight(rowIndex)
```

**Параметры**

– rowIndex – индекс строки в таблице, для которой возвращается значение высоты. Индексация строк начинается с нуля.

**Возвращает**

– высота строки в пунктах (1/72 дюйма).

**Пример**

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
height = tbl:getRowHeight(1)
```

Задать высоту строки таблицы позволяет метод Table:setRowHeight.

### 5.191.26 Метод Table:getRowsCount

Метод позволяет получить количество строк таблицы.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getRowsCount())
```

### 5.191.27 Метод Table:getShowZeroValue

Для проверки режима отображения нулевых значений ячеек используется метод getShowZeroValue.

**Пример**

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(false)
print(tbl:getShowZeroValue())
```

### 5.191.28 Метод Table:getStyledTableRange

Метод позволяет получить умную таблицу из листа табличного документа по ее названию.

**Вызов**

```
StyledTableRange getStyledTableRange(name)
```

**Параметры**

– `name`: название умной таблицы, тип `string`.

**Возвращает**

– умная таблица, тип `StyledTableRange`.

– `nil`, если если умной таблицы с таким названием не существует.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local smartTable = sheet:getStyledTableRange("SmartTable1")
local tableCells = smartTable:getCellRange()
local tableFiltersRange = smartTable:getFiltersRange()
```

### 5.191.29 Метод Table:getUsedRange

Метод возвращает диапазон ячеек, используемый в текущем листе табличного документа. К используемым относятся ячейки, которые содержат:

- данные;
- заметки;
- форматирование;
- объединенные ячейки;
- фильтры;
- сводные таблицы;
- умные таблицы.

**Вызов**

```
CellRange getUsedRange()
```

**Возвращает**

– используемый диапазон ячеек, тип `CellRange`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local usedRange = sheet:getUsedRange()
```

### 5.191.30 Метод Table:groupColumns

Метод предназначен для группировки столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

**Вызов**

```
groupColumns(columnIndex, columnsCount)
```

**Параметры**

- `columnIndex` – индекс столбца, начиная с которого будет начата группировка столбцов;
- `columnsCount` – количество столбцов для группировки.

### 5.191.31 Метод Table:groupRows

Метод предназначен для группировки строк таблицы, начиная с заданного индекса. Индексация строк начинается с нуля.

**Вызов**

```
groupRows(rowIndex, rowsCount)
```

**Параметры**

- `rowIndex` – индекс строки, начиная с которого будет начата группировка строк;
- `rowsCount` – количество строк для группировки.

### 5.191.32 Метод Table:insertColumnAfter

Метод предназначен для вставки нового столбца после указанной позиции в таблице.

**Вызов**

```
insertColumnAfter( columnIndex, copyColumnStyle, columnsCount )
```

**Параметры**

- `columnIndex` – индекс столбца в таблице, после которого производится вставка. Индексация столбцов начинается с нуля.

- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

**Пример**

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
-- Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
tbl:insertColumnAfter(0, false, 2)
```

### 5.191.33 Метод Table:insertColumnBefore

Метод предназначен для вставки нового столбца до указанной позиции в таблице.

**Вызов**

```
insertColumnBefore( columnIndex, copyColumnStyle, columnsCount )
```

**Параметры**

- `columnIndex` – индекс столбца в таблице, перед которым производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

**Пример**

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух столбцов в середину таблицы, без наследования настроек форматирования
tbl:insertColumnBefore(1, false, 2)
```

### 5.191.34 Метод Table:insertImage

Метод вставляет изображение из файла в заданные координаты на листе табличного документа.

**Вызов**

```
Image insertImage(url, rect)
```

**Параметры**

– `url`: путь к файлу, тип `string`.

– `rect`: координаты для вставки изображения, тип `RectU`.

**Возвращает**

– вставленное изображение, тип `Image`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rect = DocumentAPI.RectU()
rect.topLeft = DocumentAPI.PointU(100, 100)
rect.bottomRight = DocumentAPI.PointU(200, 150)
local insertedImage = sheet:insertImage("image.png", rect)
```

### 5.191.35 Метод Table:insertRowAfter

Метод предназначен для вставки новой строки после указанной позиции в таблице.

**Вызов**

```
insertRowAfter( rowIndex, copyRowStyle, rowsCount )
```

**Параметры**

– `rowIndex` – индекс строки в таблице, после которой производится вставка. Индексация строк начинается с нуля.

– `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.

– `rowsCount` – количество вставляемых строк. Значение по умолчанию 1.

**Пример**

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl:insertRowAfter(0, false, 2)
```

### 5.191.36 Метод Table:insertRowBefore

Метод предназначен для вставки новой строки до указанной позиции в таблице.

**Вызов**

```
insertRowBefore( rowIndex, copyRowStyle, rowsCount )
```

**Параметры**

- `rowIndex` – индекс строки в таблице, перед которой производится вставка. Индексация строк начинается с нуля.
- `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `rowsCount` – количество вставляемых строк. Значение по умолчанию 1.

**Пример**

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl:insertRowBefore(1, false, 2)
```

### 5.191.37 Метод Table:isColumnVisible

Метод `Table::isColumnVisible` позволяет определять видимость столбца по заданному индексу. Индексация столбцов начинается с нуля. Метод возвращает `true` если столбец отображается.

Для задания видимости столбцов таблицы применяется метод `Table::setColumnsVisible`.

**Вызов**

```
isColumnVisible(columnIndex)
```

**Параметр**

`columnIndex` – индекс столбца.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
print(tbl:isColumnVisible(0))
```

Дополнительный пример использования метода `Table::isColumnVisible` приведен в разделе Управление видимостью строк / колонок.

### 5.191.38 Метод Table:isProtected

Метод возвращает состояние защиты от изменений листа табличного документа (см. Защита листа табличного документа).

**Вызов**

```
bool isProtected()
```

### 5.191.39 Метод Table:isRowVisible

Метод `Table::isRowVisible` позволяет определять видимость строки по заданному индексу. Индексация строк начинается с нуля. Метод возвращает `true` если строка отображается.

Для задания видимости строк таблицы применяется метод `Table::setRowsVisible`.

**Вызов**

```
isRowVisible(rowIndex)
```

**Параметр**

`rowIndex` – индекс строки.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
print(tbl:isRowVisible(0))
```

Дополнительный пример использования метода `Table::isRowVisible` приведен в разделе Управление видимостью строк / колонок.

### 5.191.40 Метод Table:isVisible

Метод возвращает значение true, если лист таблицы в табличном документе отображается в редакторе таблиц.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
if not tbl:isVisible() then
  tbl:setVisible(true)
end
```

### 5.191.41 Метод Table:moveTo

Для перемещения листа таблицы по указанному индексу в табличном документе используется метод `moveTo`. Указанный индекс должен быть меньше или равен количеству листов в документе. Индексация листов начинается с нуля. Метод может быть использован только в табличном документе.

**Пример**

```
-- В табличном документе два листа с индексами 0 и 1.
-- Поменяем их местами.
local tbl = document:getBlocks():getTable(0)
tbl:moveTo(1)
```

### 5.191.42 Метод Table:remove

Для удаления таблицы в текстовом документе или листа в табличном документе используется метод `remove()`.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:remove()
```

### 5.191.43 Метод Table:removeColumn

Метод предназначен для удаления столбца таблицы, начиная с заданного индекса.

**Вызов**

```
removeColumn(columnIndex, columnsCount)
```

**Параметры**

- `columnIndex` – индекс столбца, начиная с которого будет удалено заданное количество столбцов. Индексация столбцов начинается с нуля.
- `columnsCount` – количество столбцов для удаления. Значение по умолчанию 1.

### 5.191.44 Метод Table:removeProtection

Метод снимает защиту от изменений с листа табличного документа (см. Защита листа табличного документа).

**Вызов**

```
removeProtection(password)
```

**Параметры**

– `password`: (необязательный) пароль для снятия защиты, тип `string`.

**Пример**

```
local firstSheet = document:getBlocks():getTable(0)
firstSheet:removeProtection("password")
```

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

### 5.191.45 Метод Table:removeRow

Метод предназначен для удаления строки таблицы, начиная с заданного индекса.

**Вызов**

```
removeRow(rowIndex, rowsCount)
```

**Параметры**

- `rowIndex` – индекс строки, начиная с которого будет удалено `rowsCount` строк. Индексация строк начинается с нуля.
- `rowsCount` – количество строк для удаления. Значение по умолчанию 1.

### 5.191.46 Метод Table:removeVisibleColumns

Метод удаляет видимые столбцы таблицы, находящиеся между заданными индексами (включительно).

**Вызов**

```
removeVisibleColumns(firstIndex, lastIndex)
```

**Параметры**

- `firstIndex`: индекс первого столбца. Индексация столбцов начинается с нуля.
- `lastIndex`: индекс последнего столбца.

### 5.191.47 Метод Table:removeVisibleRows

Метод удаляет видимые строки таблицы, находящиеся между заданными индексами (включительно).

**Вызов**

```
removeVisibleRows(firstIndex, lastIndex)
```

**Параметры**

- `firstIndex`: индекс первой строки. Индексация строк начинается с нуля.
- `lastIndex`: индекс последней строки.

### 5.191.48 Метод Table:setColumnsVisible

Метод `Table::setColumnsVisible` позволяет задавать видимость столбцов, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Метод предназначен для использования только в табличном редакторе.

**Вызов**

```
setColumnsVisible(first, columnsCount, visible)
```

**Параметры**

`first` – начальный индекс;

`columnsCount` – количество столбцов;

`visible` – видимость.

**Пример использования в табличном редакторе**

```
local tbl = document:getBlocks():getTable(0)
tbl:setColumnsVisible(0, 2, false)
```

Дополнительный пример использования метода `Table::setColumnsVisible` приведен в разделе Управление видимостью строк / колонок.

### 5.191.49 Метод Table:setColumnWidth

Метод устанавливает ширину столбца таблицы в пунктах (1/72 дюйма).

**Вызов**

```
setColumnWidth(columnIndex, width)
```

**Параметры**

- `columnIndex` – индекс столбца в таблице, для которого устанавливается значение ширины. Индексация столбцов начинается с нуля.
- `width` – ширина столбца в пунктах (1/72 дюйма).

**Пример**

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Установить ширину столбца в 400 pt
tbl:setColumnWidth(1,400)
```

Метод `Table:getColumnWidth` позволяет получить ширину столбца таблицы.

### 5.191.50 Метод Table:setLabelColor

Метод позволяет задать цвет ярлыка листа. Ярлыки используются для визуального разделения листов табличного документа.

**Вызов**

```
setLabelColor(labelColor)
```

**Параметры**

– `labelColor`: цвет ярлыка листа, тип `Color`. `nil`, чтобы сбросить цвет ярлыка.

**Пример**

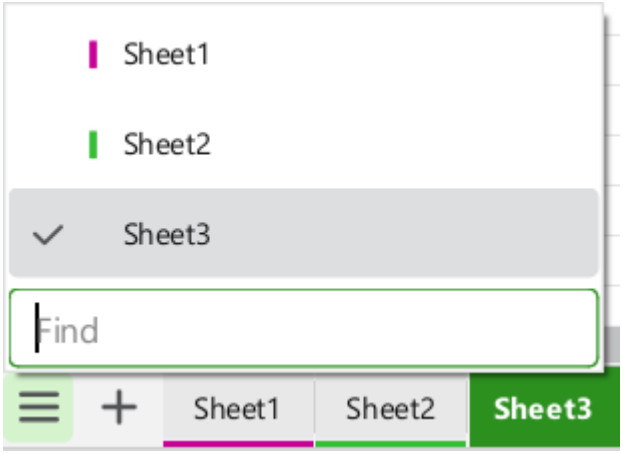



Рисунок 59 – Ярлыки листов табличного документа

```
local sheet = document:getBlocks():getTable(0)
sheet:setLabelColor(DocumentAPI.Color(DocumentAPI.ColorRGBA(200, 0, 150, 255)))
```

### 5.191.51 Метод Table:setName

Метод задает имя таблицы. В случае с табличным документом это имя будет являться заголовком листа документа. Данное значение должно быть уникальным, т.к. может использоваться для ссылки на таблицу, например, из формул.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:setName("Первый")
```

Для текстовых документов использование данного метода также допустимо, наименование таблицы нигде не отображается, но в дальнейшем его можно использовать для доступа к таблице по имени.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:setName("Первый")
tbl = document:getBlocks():getTable("Первый")
```

### 5.191.52 Метод Table:setPrintArea

Метод служит для установки и сброса области печати DocumentAPI.CellRangePosition.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:setPrintArea(DocumentAPI.CellRangePosition(0, 0, 5, 5)) -- уставить область печати размером в пять строк и пять колонок, начиная с левого верхнего угла таблицы
```

### 5.191.53 Метод Table:setPrintAreas

Метод `Table:setPrintAreas` задает множественные области печати или экспорта `CellRangePositions`, где `CellRangePositions` - вектор из элементов `CellRangePosition`.

**Пример**

```
tbl = document:getBlocks():getTable(0)
ranges = DocumentAPI.CellRangePositions()
ranges:push_back(DocumentAPI.CellRangePosition(0, 0, 5, 5))
ranges:push_back(DocumentAPI.CellRangePosition(1, 2, 5, 5))
tbl:setPrintAreas(ranges)

printAreas = tbl:getPrintAreas()
print(printAreas[0]:toString(), printAreas[1]:toString())
```

### 5.191.54 Метод Table:setPrintTitles

Метод задает сквозной заголовок (диапазон ячеек, который будет отображаться на каждой странице распечатанного документа) для текущего листа.

**Вызов**

```
setPrintTitles(range)
```

**Параметры**

– `range`: сквозной заголовок, тип `PrintTitles`.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local titles = DocumentAPI.PrintTitles(4, 4, 7, 7)
sheet:setPrintTitles(titles)

EditorAPI.printDocument()
```

### 5.191.55 Метод Table:setProtection

Метод устанавливает защиту от изменений на лист табличного документа (см. Защита листа табличного документа).

**Вызов**

```
setProtection(tableProtectionProperties, password)
```

**Параметры**

– `tableProtectionProperties`: параметры защиты листа, тип `TableProtectionProperties`;

– `password`: (необязательный) пароль для установки защиты, тип `string`.

**Пример**

```
local tableProps = DocumentAPI.TableProtectionProperties()
tableProps.deleteColumns = false
tableProps.deleteRows = false
tableProps.filterData = true
tableProps.formatCells = true
tableProps.formatColumns = true
tableProps.formatRows = true
tableProps.insertAndEditObjects = false
tableProps.insertAndEditPivotTables = false
tableProps.insertColumns = false
tableProps.insertLinks = true
tableProps.insertRows = false
tableProps.selectProtectedCells = true
tableProps.sortData = true

local firstSheet = document:getBlocks():getTable(0)
firstSheet:setProtection(tableProps, "password")
```

Метод `setProtectionProperties()` объектов `Cell` и `CellRange` позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой защиты).

Если метод `setProtection()` применяется к уже защищенному листу, возникает исключение `SpreadsheetProtectionError`.

### 5.191.56 Метод Table:setRowHeight

Метод устанавливает высоту строки таблицы в пунктах (1/72 дюйма).

**Вызов**

```
setRowHeight(rowIndex, height, heightRule)
```

**Параметры**

- `rowIndex` – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- `height` – высота строки в пунктах (1/72 дюйма).
- `heightRule` – точность значения (`DocumentAPI.RowHeightRule_Exact` – точно, `DocumentAPI.RowHeightRule_AtLeast` – не меньше).

**Пример**

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Установить высоту строки в 100 pt
tbl:setRowHeight(1,100, DocumentAPI.RowHeightRule_Exact)
```

Метод `Table:getRowHeight` позволяет получить высоту строки таблицы.

### 5.191.57 Метод Table:setRowsVisible

Метод `Table::setRowsVisible` позволяет задавать видимость строк, начиная с заданного индекса. Индексация строк начинается с нуля.

Метод предназначен для использования только в табличном редакторе.

**Вызов**

```
setRowsVisible(first, rowsCount, visible)
```

**Параметры**

`first` – начальный индекс;

`columnsCount` – количество строк;

`visible` – видимость.

**Пример использования в табличном редакторе**

```
local tbl = document:getBlocks():getTable(0)
tbl:setRowsVisible(0, 2, false)
```

Дополнительный пример использования метода `Table::setRowsVisible` приведен в разделе Управление видимостью строк / колонок.

### 5.191.58 Метод Table:setShowZeroValue

Для упрощения чтения таблицы нулевые значения ячеек могут быть скрыты. Для управления скрытием/показом ячеек используется метод `setShowZeroValue`. Метод может быть использован только в табличном документе.

**Пример**

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(true)
```

### 5.191.59 Метод Table:setVisible

Метод управляет видимостью листа таблицы. Используется только в табличном документе.

**Вызов**

```
setVisible( visible )
```

**Параметр**

`visible` – параметр, задающий видимость листа. Если значение параметра `visible` равно `true`, то лист таблицы отображается в редакторе таблиц.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
tbl:setVisible(false)
```

### 5.191.60 Метод Table:ungroupColumns

Метод предназначен для разгруппировки столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

**Вызов**

```
ungroupColumns(columnIndex, columnsCount)
```

**Параметры**

- `columnIndex` – индекс столбца, начиная с которого будет начата разгруппировка столбцов;

– `columnsCount` – количество столбцов для разгруппировки.

### 5.191.61 Метод Table:ungroupRows

Метод предназначен для разгруппировки строк таблицы, начиная с заданного индекса. Индексация строк начинается с нуля.

**Вызов**

```
ungroupRows(rowIndex, rowsCount)
```

**Параметры**

– `rowIndex` – индекс строки, начиная с которого будет начата разгруппировка строк;

– `rowsCount` – количество строк для разгруппировки.

### 5.191.62 Метод Table:__eq

Метод используется для определения эквивалентности двух таблиц.

**Пример**

```
local tbl1 = document:getBlocks():getTable(0)
local tbl2 = document:getBlocks():getTable("Table Name")
if tbl1:__eq(tbl2) then
    print("tbl1 и tbl2 ссылаются на общую таблицу в документе")
end
```

### 5.192 Таблица DocumentAPI.TableFilters

`TableFilters` - это таблица, которая хранит фильтры столбцов. Фильтры можно применять к диапазону фильтрации `FiltersRange`. При применении фильтров, соответствующие строки рабочего листа будут скрыты.

Конструктор по умолчанию:

```
local firstTableFilters = DocumentAPI.TableFilters()
```

Конструктор копирования:

```
local secondTableFilters = DocumentAPI.TableFilters(firstTableFilters)
```

Пример использования приведен в разделе Работа с фильтрами.

### 5.192.1 Метод TableFilters:clear

Метод `TableFilters:clear` удаляет все фильтры столбцов, которые были сохранены ранее.

**Пример**

```
local tableFilters = DocumentAPI.TableFilters()
tableFilters:setFilter(0, johnPaulFilter)
tableFilters:setFilter(1, songFilter)
.....................
tableFilters:clear()
```

### 5.192.2 Метод TableFilters:erase

Метод `TableFilters:erase` удаляет фильтр из заданного столбца. В качестве параметра используется индекс столбца.

**Пример**

```
local tableFilters = DocumentAPI.TableFilters()
tableFilters:setFilter(0, johnPaulFilter)
tableFilters:setFilter(1, songFilter)
.....................
tableFilters:erase(1)
```

### 5.192.3 Метод TableFilters:getAsConditionalFilter

Метод возвращает фильтр по условию, установленный в заданном столбце.

**Вызов**

```
ConditionalTableFilter getAsConditionalFilter(column)
```

**Параметры**

– `column`: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `number`.

**Возвращает**

– фильтр по условию, тип `ConditionalTableFilter`.

– `nil`, если фильтр отсутствует или он другого типа.

**Пример**

```
if filters:getFilterType(0) == DocumentAPI.ContainingTableFilter_Conditional then
    conditionalFilter = filters:getAsConditionalFilter(0)
end
```

### 5.192.4 Метод TableFilters:getAsValueFilter

Метод возвращает фильтр по значению, установленный в заданном столбце.

**Вызов**

```
ValuesTableFilter getAsValueFilter(column)
```

**Параметры**

– `column`: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `number`.

**Возвращает**

– фильтр по значению, тип `ValuesTableFilter`.

– `nil`, если фильтр отсутствует или он другого типа.

**Пример**

```
if filters:getFilterType(0) == DocumentAPI.ContainingTableFilter_Values then
    valuesFilter = filters:getAsValueFilter(0)
end
```

### 5.192.5 Метод TableFilters:getFilterType

Метод возвращает тип фильтра, который установлен в заданном столбце.

**Вызов**

```
ContainingTableFilter getFilterType(column)
```

**Параметры**

– `column`: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `number`.

**Возвращает**

– тип фильтра столбцов таблицы, тип `ContainingTableFilter`.

**Пример**

```
if filters:getFilterType(0) == DocumentAPI.ContainingTableFilter_Values then
    valuesFilter = filters:getAsValueFilter(0)
end
```

### 5.192.6 Метод TableFilters:setFilter

Метод `TableFilters:setFilter` устанавливает фильтр для конкретного столбца. В качестве параметров используются индекс столбца, а также используемый фильтр: `ValuesTableFilter` или `ConditionalTableFilter`.

**Пример**

```
local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")
johnPaulFilter:clear()

local songFilter = DocumentAPI.ConditionalTableFilter()
songFilter:setAndOperation(true)
songFilter:notEqual("")
songFilter:notBegins("TODO")

local tableFilters = DocumentAPI.TableFilters()
tableFilters:setFilter(0, johnPaulFilter)
tableFilters:setFilter(1, songFilter)
```

Более подробный пример приведен в разделе Работа с фильтрами.

### 5.193 Таблица DocumentAPI.TableProtectionProperties

Таблица `DocumentAPI.TableProtectionProperties` предназначена для настройки параметров защиты листа в табличном документе (аналог раздела «Разрешенные действия» в меню «Управление защитой»). Данная таблица используется в методах `Table:setProtection()` и `Table:getProtectionProperties()`.

Таблица 111 – Описание полей таблицы `DocumentAPI.TableProtectionProperties`

| Поле | Значение по умолчанию | Описание |
|---|---|---|
| `TableProtectionProperties.deleteColumns` | `false` | Разрешить удалять колонки |

| Поле | Значение по умолчанию | Описание |
|---|---|---|
| `TableProtectionProperties.deleteRows` | `false` | Разрешить удалять строки |
| `TableProtectionProperties.filterData` | `false` | Разрешить фильтровать данные |
| `TableProtectionProperties.formatCells` | `false` | Разрешить форматировать ячейки |
| `TableProtectionProperties.formatColumns` | `false` | Разрешить форматировать столбцы |
| `TableProtectionProperties.formatRows` | `false` | Разрешить форматировать строки |
| `TableProtectionProperties.insertAndEditObjects` | `false` | Разрешить вставлять и редактировать объекты: изображения, диаграммы, фигуры и текстовые поля |
| `TableProtectionProperties.insertAndEditPivotTables` | `false` | Разрешить вставлять и редактировать сводные таблицы |
| `TableProtectionProperties.insertColumns` | `false` | Разрешить вставлять столбцы |
| `TableProtectionProperties.insertLinks` | `false` | Разрешить вставлять и редактировать ссылки в ячейках |
| `TableProtectionProperties.insertRows` | `false` | Разрешить вставлять строки |
| `TableProtectionProperties.selectProtectedCells` | `true` | Разрешить выделение защищенных ячеек |
| `TableProtectionProperties.sortData` | `false` | Разрешить сортировать данные |

**Пример**

```
local tableProps = DocumentAPI.TableProtectionProperties()
tableProps.deleteColumns = false
tableProps.deleteRows = false
tableProps.filterData = true
tableProps.formatCells = true
tableProps.formatColumns = true
tableProps.formatRows = true
tableProps.insertAndEditObjects = false
tableProps.insertAndEditPivotTables = false
tableProps.insertColumns = false
tableProps.insertLinks = true
tableProps.insertRows = false
```

```
tableProps.selectProtectedCells = true
tableProps.sortData = true

local firstSheet = document:getBlocks():getTable(0)
firstSheet:setProtection(tableProps, "password")
```

### 5.194 Таблица DocumentAPI.TableSearchSettings

Таблица `TableSearchSettings` предназначена для настройки параметров поиска ячеек. Данная таблица используется в методах `Table:find()` и `CellRange:find()`.

Таблица 112 – Описание полей таблицы `TableSearchSettings`

| Поле | Тип | Описание |
|---|---|---|
| `TableSearchSettings.caseSensitive` | `CaseSensitive` | Позволяет учитывать регистр при поиске |
| `TableSearchSettings.matchBehaviour` | `TableSearchSettings.MatchBehaviour` | Задает алгоритм сравнения запроса и значения |
| `TableSearchSettings.searchIn` | `TableSearchSettings.SearchProperty` | Позволяет выбрать значения, по которым производится поиск |
| `TableSearchSettings.wholeWords` | `bool` | Разбивает текст в ячейке на слова, которые считаются отдельными значениями для сравнения |

**Пример**

```
local sheet = document:getBlocks():getTable(0)

local searchProps = DocumentAPI.TableSearchSettings()
searchProps.caseSensitive = DocumentAPI.CaseSensitive_No
searchProps.matchBehaviour = DocumentAPI.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = DocumentAPI.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = true

local results = sheet:find("*eye", searchProps)
for cell in results do
    print(cell:getFormattedValue()) -- Steeleye Stout
end
```

#### 5.194.1 Таблица TableSearchSettings.MatchBehaviour

Таблица `MatchBehaviour` определяет алгоритм сравнения запроса и значения.

Используется в поле TableSearchSettings.matchBehaviour.

Таблица 113 – Описание алгоритмов сравнения

| Значение | Описание |
|---|---|
| TableSearchSettings.MatchBehaviour_Partial | Значение частично содержит запрос |
| TableSearchSettings.MatchBehaviour_Total | Значение полностью соответствует запросу |
| TableSearchSettings.MatchBehaviour_Glob | Позволяет использовать шаблоны, содержащие символы ***** и **?**, для создания запроса |

### 5.194.2 Таблица TableSearchSettings.SearchProperty

Таблица SearchProperty определяет значения, по которым производится поиск в таблице. Используется в поле TableSearchSettings.searchIn.

Таблица 114 – Описание вариантов поиска в таблице

| Значение | Описание |
|---|---|
| TableSearchSettings.SearchProperty_Value | Поиск по значениям ячеек |
| TableSearchSettings.SearchProperty_Formula | Поиск по тексту формул в ячейках |
| TableSearchSettings.SearchProperty_Notes | Поиск по тексту заметок |

### 5.195 Таблица DocumentAPI.TextAnchoredPosition

Таблица DocumentAPI.TextAnchoredPosition (см. Рисунок 60) представляет позицию объекта на странице текстового документа. Пример использования см. в разделе InlineFrame:setPosition().

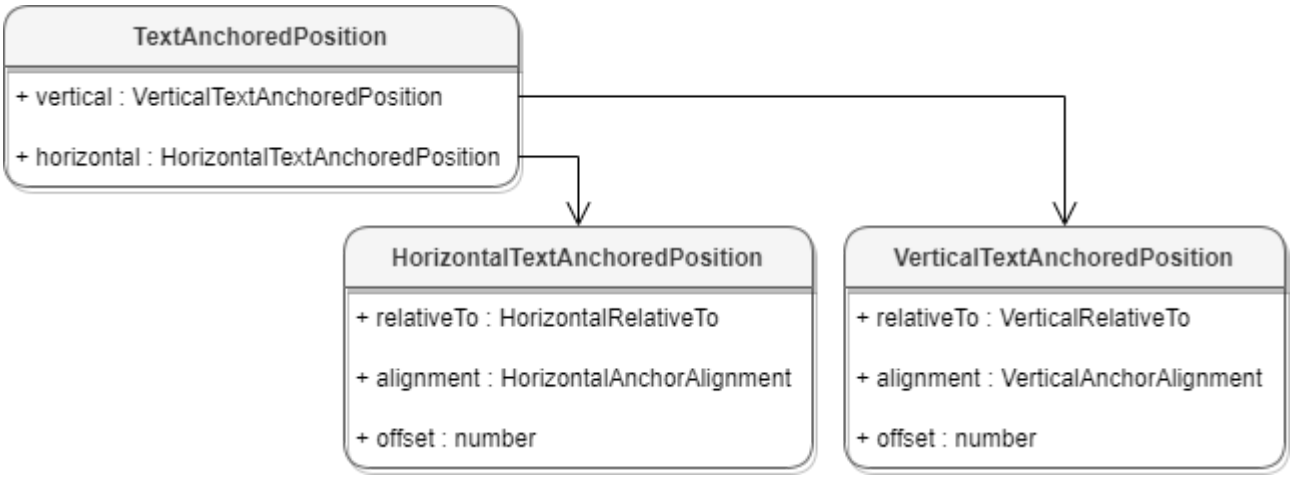



Рисунок 60 – Поля таблицы `DocumentAPI.TextAnchoredPosition`

Описание полей таблицы представлено в таблице 115.

Таблица 115 – Описание полей таблицы `DocumentAPI.TextAnchoredPosition`

| Поле | Тип | Описание |
|---|---|---|
| `TextAnchoredPosition.horizontal` | `HorizontalTexAnchoredPosition` | Позиция по горизонтали |
| `TextAnchoredPosition.vertical` | `VerticalTextAnchoredPosition` | Позиция по вертикали |

### 5.195.1 Метод TextAnchoredPosition:__eq

Метод используется для определения эквивалентности значений двух позиций объектов.

**Пример**

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos1.horizontal.offset = 1

local pos2 = DocumentAPI.TextAnchoredPosition()
pos2.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
```

```
pos2.horizontal.offset = 1

print(pos1:__eq(pos2))
```

### 5.196 Таблица DocumentAPI.TextConditionalFormatOperator

Таблица `TextConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Текст". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

**Вызов**

```
TextConditionalFormatOperator(ConditionalFormatTextCondition condition, string argument)
```

**Пример**

| Товар |
|---|
| Ноутбук |
| Смартфон |
| Наушники |
| Телевизор |
| Планшет |
| Игровая приставка |
| Монитор |
| Клавиатура |
| Мышь |
| Фотоаппарат |
| Часы |

Рисунок 61 – Пример создания правила для текстовых значений

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local textStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
textStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("A2:A12")
local cellRangePosition = cellRange:getTableRange()

local textOperator =
DocumentAPI.createTextConditionalFormatOperator(DocumentAPI.ConditionalFormatText
```

```
Condition_BeginsWith, "M")

local textRule = DocumentAPI.ConditionalFormatRule(textOperator, textStyle,
cellRangePosition, false)
rules:addRule(textRule)
```

### 5.196.1 Метод TextConditionalFormatOperator:getArgument

Метод возвращает аргумент условия.

**Вызов**

```
string getArgument()
```

**Возвращает**

– аргумент условия, тип `string`.

### 5.196.2 Метод TextConditionalFormatOperator:getCondition

Метод возвращает условие применения форматирования.

**Вызов**

```
ConditionalFormatTextCondition getCondition()
```

**Возвращает**

– условие применения форматирования, тип `ConditionalFormatTextCondition`.

### 5.196.3 Метод TextConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

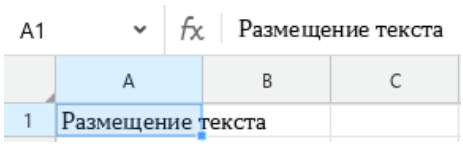
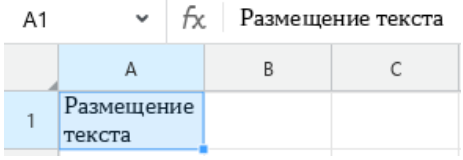
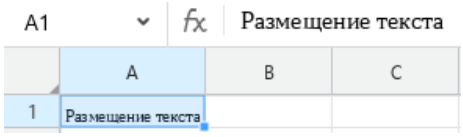
**Возвращает**

– тип оператора условного форматирования, тип `ConditionalFormatOperatorType`.

## 5.197 Таблица DocumentAPI.TextLayout

В таблице 116 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле `textLayout` таблицы `CellProperties`.

Таблица 116 – Варианты размещения текста в ячейках таблицы

| Значение | Описание | Отображение |
|---|---|---|
| `TextLayout_SingleLine` | Текст располагается в одну строку с наложением на соседние ячейки. |  |
| `TextLayout_WrapByWords` | Текст внутри ячейки переносится по словам. Высота ряда увеличивается чтобы разместить текст полностью. |  |
| `TextLayout_ ShrinkSizeToFitWidth` | Текст располагается в одну линию, отображение масштабируется таким образом, чтобы полностью разместиться в ячейке без изменения ее размера. Размер шрифта не изменяется, данная настройка влияет только на отображение содержимого ячейки таблицы. |  |



**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell_A1 = tbl:getCell("A1")
local props = cell_A1:getCellProperties()
props.textLayout = DocumentAPI.TextLayout_ShrinkSizeToFitWidth
cell_A1:setCellProperties(props)
```

### 5.198 Таблица DocumentAPI.TextOrientation

Таблица `DocumentAPI.TextOrientation` предоставляет доступ к свойствам ориентации текста в ячейке, фигуре и т. д. (см. `DocumentAPI.CellProperties`).

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("D2") --(DocumentAPI.CellPosition(3,1))
local props = cell:getCellProperties()
props.textOrientation = DocumentAPI.TextOrientation(45)
```

```
cell:setCellProperties(props)
print(props.textOrientation:getAngle())
```

### 5.198.1 Метод TextOrientation:getAngle

Возвращает угол ориентации текста в ячейке. Значение угла указывается в градусах.

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("D2") --(DocumentAPI.CellPosition(3,1))
local cellProperties = cell:getCellProperties()
print(cellProperties.textOrientation:getAngle())
```

### 5.198.2 Метод TextOrientation:isStackedChars

Возвращает `True` в случае, если ориентация текста представляет собой вертикальный столбец.

**Пример**

```
local cellProperties = cell:getCellProperties()
print(cellProperties.textOrientation:isStackedChars())
```

### 5.198.3 Метод TextOrientation:__eq

Метод используется для определения эквивалентности значений двух объектов `TextOrientation`.

**Пример**

```
print(DocumentAPI.TextOrientation(45):__eq(DocumentAPI.TextOrientation(45)))
```

### 5.199 Таблица DocumentAPI.TextProperties

Таблица `DocumentAPI.TextProperties` содержит поля, задающие параметры текста. На рисунке 62 изображена объектная модель таблицы `DocumentAPI.TextProperties`.

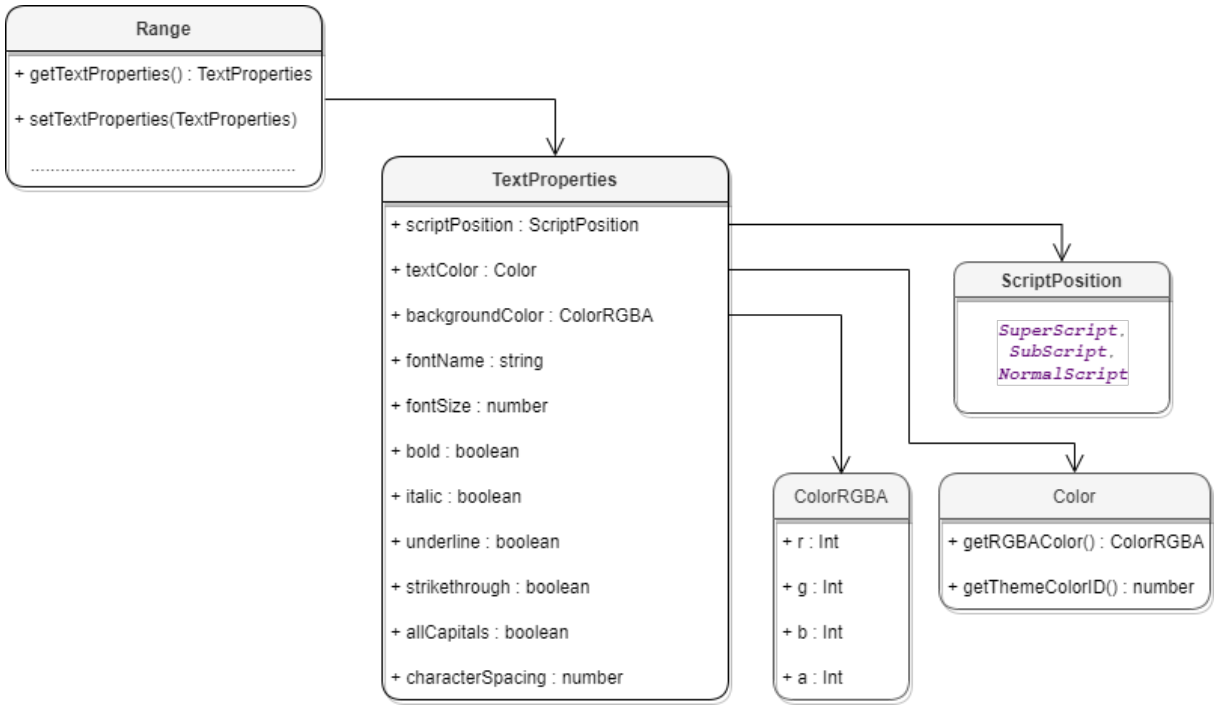



Рисунок 62 – Объектная модель для работы с таблицей `DocumentAPI.TextProperties`

Описание полей таблицы `DocumentAPI.TextProperties` представлено в таблице 117. Свойства `DocumentAPI.TextProperties` применяются к диапазону текста `DocumentAPI.Range` (методы `Range:getTextProperties()`, `Range:setTextProperties()`), ячейке `DocumentAPI.Cell` (методы `Cell:getTextProperties()`, `Cell:setTextProperties()`) и диапазону ячеек `DocumentAPI.CellRange` (методы `CellRange:getTextProperties()`, `CellRange:setTextProperties()`).

Таблица 117 – Описание полей таблицы `DocumentAPI.TextProperties`

| Поле | Тип | Описание |
|---|---|---|
| `TextProperties.fontName` | `string` | Наименование шрифта, использованного для оформления фрагмента документа |
| `TextProperties.fontSize` | `number` | Размер шрифта, использованного для оформления фрагмента документа |
| `TextProperties.bold` | `bool` | Значение `true` устанавливает жирное начертание для |

| Поле | Тип | Описание |
|---|---|---|
| | | указанного фрагмента текста |
| `TextProperties.italic` | `bool` | Значение `true` устанавливает начертание курсивом для указанного фрагмента текста |
| `TextProperties.underline` | `bool` | Значение `true` устанавливает подчеркивание для указанного фрагмента текста |
| `TextProperties.strikethrough` | `bool` | Значение `true` устанавливает начертание «зачеркнутый» для указанного фрагмента текста |
| `TextProperties.allCapitals` | `bool` | Значение `true` устанавливает все буквы указанного фрагмента текста как прописные. Значение `false` устанавливает все буквы указанного фрагмента текста как строчные |
| `TextProperties.scriptPosition` | `ScriptPosition` | Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме |
| `TextProperties.textColor` | `Color` | Цвет указанного фрагмента документа |
| `TextProperties.backgroundColor` | `ColorRGBA` | Цвет фона указанного фрагмента документа |
| `TextProperties.characterSpacing` | `number` | Размер межсимвольного интервала |

Поля класса `TextProperties` могут быть `nil`, если фрагмент документа содержит текст с разными параметрами.

**Пример**

```
local props = DocumentAPI.TextProperties()
props.fontName = "XO Oriel"
props.fontSize = 20

-- текст третьего абзаца
local range = document:getBlocks():getParagraph(2):getRange()

-- установить свойства фрагмента текста
range:setTextProperties(props)
```

### 5.200 Таблица DocumentAPI.TextStyle

Таблтца `TextStyle` представляет собой стиль абзаца. Используется в методах `Paragraph:getStyle()`, `Paragraph:getResolvedStyle()`, `Paragraph:setStyle()`, `TextStyles:create()`, `TextStyles:get()`, `TextStyle:getParent()`, `TextStyle:getNextParagraphStyle()` и `TextStyle:setNextParagraphStyle()`.

#### 5.200.1 Метод TextStyle:createChild

Метод создает новый стиль на основе текущего стиля абзаца.

**Вызов**

```
createChild(name)
```

**Параметры**

– `name`: название нового стиля, тип `string`.

**Пример**

```
local styles = document:getTextStyles()
local normalStyle = styles:get(DocumentAPI.PredefinedTextStyle_Normal)
normalStyle:createChild("myStyle")
local myStyle = styles:get("myStyle")
local parentStyle = myStyle:getParent()
print(myStyle:getName()) -- myStyle
print(parentStyle:getName()) -- Normal
```

#### 5.200.2 Метод TextStyle:getName

Метод возвращает название текущего стиля.

**Вызов**

```
string getName()
```

**Возвращает**

– название стиля, тип `string`.

#### 5.200.3 Метод TextStyle:getNextParagraphStyle

Метод возвращает стиль следующего абзаца.

**Вызов**

```
TextStyle getNextParagraphStyle()
```

**Возвращает**

– стиль следующего абзаца, тип TextStyle.

**Пример**

```
local styles = document:getTextStyles()
local titleStyle = styles:get(DocumentAPI.PredefinedTextStyle_Title)
print(titleStyle:getNextParagraphStyle():getName()) -- Normal
```

### 5.200.4 Метод TextStyle:getParagraphProperties

Метод возвращает настройки форматирования абзаца.

**Вызов**

```
ParagraphProperties getParagraphProperties()
```

**Возвращает**

– настройки форматирования абзаца, тип ParagraphProperties.

### 5.200.5 Метод TextStyle:getParent

Метод возвращает стиль, на котором основан текущий стиль абзаца.

**Вызов**

```
TextStyle getParent()
```

**Возвращает**

– родительский стиль абзаца, тип TextStyle.

– `nil`, если текущий стиль не основан ни на каком стиле.

**Пример**

```
local styles = document:getTextStyles()
local normalStyle = styles:get(DocumentAPI.PredefinedTextStyle_Normal)
normalStyle:createChild("myStyle")
local myStyle = styles:get("myStyle")
local parentStyle = myStyle:getParent()
print(myStyle:getName()) -- myStyle
print(parentStyle:getName()) -- Normal
```

### 5.200.6 Метод TextStyle:getTextProperties

Метод возвращает настройки форматирования текста.

**Вызов**

```
TextProperties getTextProperties()
```

**Возвращает**

– настройки форматирования текста, тип TextProperties.

### 5.200.7 Метод TextStyle:setName

Метод задает название стиля.

**Вызов**

```
setName(newName)
```

**Параметры**

– newName: название стиля, тип string.

**Пример**

```
local styles = document:getTextStyles()
local normalStyle = styles:get(DocumentAPI.PredefinedTextStyle_Normal)
normalStyle:setName("myStyle")
```

### 5.200.8 Метод TextStyle:setNextParagraphStyle

Метод задает стиль следующего абзаца.

**Вызов**

```
setNextParagraphStyle(paragraphStyle)
```

**Параметры**

– paragraphStyle: стиль следующего абзаца, тип TextStyle.

**Пример**

```
local styles = document:getTextStyles()
local titleStyle = styles:get(DocumentAPI.PredefinedTextStyle_Title)
titleStyle:setNextParagraphStyle(styles:get(DocumentAPI.PredefinedTextStyle_Subti
tle))
```

### 5.200.9 Метод TextStyle:setParagraphProperties

Метод задает настройки форматирования абзаца.

**Вызов**

```
setParagraphProperties(properties)
```

**Параметры**

– `properties`: настройки форматирования абзаца, тип `ParagraphProperties`.

**Пример**

```
local styles = document:getTextStyles()
local normalStyle = styles:get(DocumentAPI.PredefinedTextStyle_Normal)
local props = normalStyle:getParagraphProperties()
props.beforeSpacing = 2.1
normalStyle:setParagraphProperties(props)
```

### 5.200.10 Метод TextStyle:setTextProperties

Метод задает настройки форматирования текста.

**Вызов**

```
setTextProperties(textProperties)
```

**Параметры**

– `textProperties`: настройки форматирования текста, тип `TextProperties`.

**Пример**

```
local styles = document:getTextStyles()
local normalStyle = styles:get(DocumentAPI.PredefinedTextStyle_Normal)
local props = normalStyle:getTextProperties()
props.fontSize = 20
normalStyle:setTextProperties(props)
```

### 5.201 Таблица DocumentAPI.TextStyles

Таблица `TextStyles` предоставляет доступ к стилям абзацев в документе. Используется в методе `Document:getTextStyles()`.

### 5.201.1 Метод TextStyles:create

Метод создает новый стиль абзаца с заданным названием или идентификатором. Если такой стиль уже существует, возникает `IncorrectArgumentError`.

**Вызов**

```
TextStyle create(newStyleName)
```

```
TextStyle create(predefinedTextStyle)
```

**Параметры**

– `newStyleName`: название нового стиля, тип `string`.

– `predefinedTextStyle`: идентификатор нового стиля, тип `PredefinedTextStyle`.

**Возвращает**

– созданный стиль абзаца, тип `TextStyle`.

**Пример**

```
local styles = document:getTextStyles()
local myStyle = styles:create("myStyle")
local myHeadingStyle = styles:create(DocumentAPI.PredefinedTextStyle_Heading6)
```

### 5.201.2 Метод TextStyles:enumerate

Метод возвращает перечисление стилей абзацев.

**Вызов**

```
TextStylesEnumerator enumerate()
```

**Возвращает**

– перечисление стилей абзацев.

**Пример**

```
local styles = document:getTextStyles()
for style in styles:enumerate() do
    print(style:getName())
end
```

### 5.201.3 Метод TextStyles:get

Метод возвращает стиль абзаца по его названию или идентификатору.

**Вызов**

```
TextStyle get(styleName)
```

```
TextStyle get(predefinedTextStyle)
```

**Параметры**

– `styleName`: название стиля, тип `string`.

– `predefinedTextStyle`: идентификатор стиля, тип `PredefinedTextStyle`.

**Возвращает**

– стиль абзаца, тип `TextStyle`.

– `nil`, если стиля с заданным названием или идентификатором не существует.

**Пример**

```
local styles = document:getTextStyles()
local normalStyle = styles:get("Normal")
local titleStyle = styles:get(DocumentAPI.PredefinedTextStyle_Title)
print(normalStyle:getName())
print(titleStyle:getName())
```

### 5.202 Таблица TextToColumnsSettings

Таблица `TextToColumnsSettings` предназначен для настройки параметров разделения текста по ячейкам. Данная таблица используется в методе `CellRange:textToColumns()`.

Таблица 118 – Описание полей таблицы `TextToColumnsSettings`

| Поле | Тип | Описание |
|---|---|---|
| `TextToColumnsSettings.customDelimiter` | `string` | Позволяет задать пользовательский разделитель |
| `TextToColumnsSettings.textQualifier` | `TextQualifier` | Задает символ-ограничитель. Текст между двумя ограничителями записывается в одну ячейку, даже если содержит разделители |
| `TextToColumnsSettings.treatMultipleDelimitersAsOne` | `bool` | Объединять разделители, расположенные подряд |
| `TextToColumnsSettings.useCommaDelimiter` | `bool` | Использовать запятую (,) в качестве разделителя |
| `TextToColumnsSettings.usePercentDelimiter` | `bool` | Использовать процент (%) в качестве разделителя |

| Поле | Тип | Описание |
|---|---|---|
| TextToColumnsSettings.useSemicolonDelimiter | bool | Использовать точку с запятой (;) в качестве разделителя |
| TextToColumnsSettings.useSlashDelimiter | bool | Использовать косую черту (/) в качестве разделителя |
| TextToColumnsSettings.useSpaceDelimiter | bool | Использовать пробел в качестве разделителя |
| TextToColumnsSettings.useTabDelimiter | bool | Использовать знак табуляции в качестве разделителя |

**Пример**

```
local settings = DocumentAPI.TextToColumnsSettings()
settings.useCommaDelimiter = true
settings.useSpaceDelimiter = true
settings.treatMultipleDelimitersAsOne = true
settings.textQualifier =
DocumentAPI.TextToColumnsSettings.TextQualifier_SingleQuotes
cellRange:textToColumns(settings)
```

### 5.202.1 Таблица TextToColumnsSettings.TextQualifier

Таблица TextQualifier содержит доступные символы-ограничители для горизонтального разделения текста по ячейкам. Текст между двумя ограничителями записывается в одну ячейку, даже если содержит разделители. Используется в поле TextToColumnsSettings.textQualifier.

Таблица 119 – Варианты символов-ограничителей

| Значение | Описание |
|---|---|
| TextToColumnsSettings.TextQualifier_None | Без ограничителей |
| TextToColumnsSettings.TextQualifier_SingleQuotes | Одинарные кавычки (') |
| TextToColumnsSettings.TextQualifier_DoubleQuotes | Двойные кавычки (") |

**Пример**

```
local settings = DocumentAPI.TextToColumnsSettings()
settings.useCommaDelimiter = true
settings.useSpaceDelimiter = true
settings.treatMultipleDelimitersAsOne = true
settings.textQualifier =
```

```
DocumentAPI.TextToColumnsSettings.TextQualifier_SingleQuotes
cellRange:textToColumns(settings)
```

### 5.203 Таблица DocumentAPI.TextUnit

В таблице 120 представлены варианты разделения текста на диапазоны. Используется в методах Position:getCurrentRange(), Position:getNextRange() и Position:getPreviousRange().

Таблица 120 – Варианты разделения текста на диапазоны

| Значение | Описание |
|---|---|
| DocumentAPI.TextUnit_Character | Разделение по символам |
| DocumentAPI.TextUnit_Word | Разделение по словам (символы-разделители считаются отдельными словами) |
| DocumentAPI.TextUnit_Sentence | Разделение по предложениям (разделители после предложения считаются его частью) |
| DocumentAPI.TextUnit_Paragraph | Разделение по абзацам |

**Пример**

```
local secondSentence =
document:getRange():getBegin():getNextRange(DocumentAPI.TextUnit_Sentence)
local firstSentenceLastWord =
secondSentence:getBegin():getPreviousRange(DocumentAPI.TextUnit_Word)
while
firstSentenceLastWord:getContentEnd():compare(firstSentenceLastWord:getBegin())
== 1 do
    firstSentenceLastWord =
firstSentenceLastWord:getBegin():getPreviousRange(DocumentAPI.TextUnit_Word)
end
local thirdSentenceFirstLetter =
secondSentence:getContentEnd():getCurrentRange(DocumentAPI.TextUnit_Character)

print(secondSentence:extractText())
-- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
print(firstSentenceLastWord:extractText())
-- aliqua
```

```
print(thirdSentenceFirstLetter:extractText())
-- D
```

### 5.204 Таблица DocumentAPI.TextWrapType

В таблице 121 представлены варианты обтекания текстом встроенного объекта. Используется в `InlineFrame:setWrapType()`, `InlineFrame:getWrapType()`.

Таблица 121 – Варианты обтекания текстом встроенного объекта

| Значение | Описание |
|---|---|
| `DocumentAPI.TextWrapType_Inline` | Встроенный объект располагается в тексте |
| `DocumentAPI.TextWrapType_InFrontOfText` | Встроенный объект располагается перед текстом |
| `DocumentAPI.TextWrapType_BehindText` | Встроенный объект располагается за текстом |
| `DocumentAPI.TextWrapType_TopAndBottom` | Текст располагается сверху и снизу от встроенного объекта |
| `DocumentAPI.TextWrapType_Square` | Текст располагается вокруг прямоугольной рамки встроенного объекта |
| `DocumentAPI.TextWrapType_Through` | Текст обтекает встроенный объект по сторонам и внутри |
| `DocumentAPI.TextWrapType_Tight` | Текст располагается на одинаковых расстояниях от границ объекта |

### 5.205 Таблица DocumentAPI.ThemeColorID

В таблице 122 представлены типы идентификаторов цветов тем. Используется в `DocumentAPI.Color`.

Таблица 122 – Типы идентификаторов цветов тем

| Значение | Описание |
|---|---|
| `DocumentAPI.ThemeColorID_Background1` | Фон1 |
| `DocumentAPI.ThemeColorID_Text1` | Текст1 |
| `DocumentAPI.ThemeColorID_Background2` | Фон2 |
| `DocumentAPI.ThemeColorID_Text2` | Текст2 |
| `DocumentAPI.ThemeColorID_Dark1` | Темная1 |
| `DocumentAPI.ThemeColorID_Dark2` | Темная2 |

| Значение | Описание |
|---|---|
| DocumentAPI.ThemeColorID_Light1 | Светлая1 |
| DocumentAPI.ThemeColorID_Light2 | Светлая2 |
| DocumentAPI.ThemeColorID_Accent1 | Акцент1 |
| DocumentAPI.ThemeColorID_Accent2 | Акцент2 |
| DocumentAPI.ThemeColorID_Accent3 | Акцент3 |
| DocumentAPI.ThemeColorID_Accent4 | Акцент4 |
| DocumentAPI.ThemeColorID_Accent5 | Акцент5 |
| DocumentAPI.ThemeColorID_Accent6 | Акцент6 |
| DocumentAPI.ThemeColorID_Hyperlink | Гиперссылка |
| DocumentAPI.ThemeColorID_FollowedHyperlink | Следующая гиперссылка |

### 5.206 Таблица DocumentAPI.TimePatterns

Форматы времени представлены в таблице 123. Пример использования см. в главе DocumentAPI.DateTimeCellFormatting.

Таблица 123 – Форматы времени

| Значение | Описание |
|---|---|
| DocumentAPI.TimePatterns_ShortTime | 'hh:mm AM/PM' для языка en_US |
| DocumentAPI.TimePatterns_LongTime | 'hh:mm:ss AM/PM' для языка en_US |

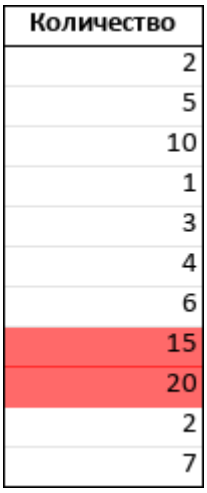
### 5.207 Таблица DocumentAPI.TopBottomConditionalFormatOperator

Таблица TopBottomConditionalFormatOperator представляет собой оператор, который содержит настройки применения форматирования для правила "Наибольшие и наименьшие значения". Используется при создании правила условного форматирования (ConditionalFormatRule).

**Конструктор**

```
TopBottomConditionalFormatOperator(ConditionalFormatTopBottomCondition condition, number value, bool usePercent)
```

**Пример**



| Количество |
|---|
| 2 |
| 5 |
| 10 |
| 1 |
| 3 |
| 4 |
| 6 |
| 15 |
| 20 |
| 2 |
| 7 |

Рисунок 63 – Пример создания правила для наибольших 20% значений

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local topBottomStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
topBottomStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("F2:F12")
local cellRangePosition = cellRange:getTableRange()

local topBottomOperator =
DocumentAPI.createTopBottomConditionalFormatOperator(DocumentAPI.ConditionalForma
tTopBottomCondition_Top, 20, true)

local topBottomRule = DocumentAPI.ConditionalFormatRule(topBottomOperator,
topBottomStyle, cellRangePosition, false)
rules:addRule(topBottomRule)
```

### 5.207.1 Метод TopBottomConditionalFormatOperator:getCondition

Метод возвращает условие применения форматирования.

**Вызов**

```
ConditionalFormatTopBottomCondition getCondition()
```

**Возвращает**

– условие применения форматирования, тип ConditionalFormatTopBottomCondition.

### 5.207.2 Метод TopBottomConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

### 5.207.3 Метод TopBottomConditionalFormatOperator:getUsePercent

Метод позволяет определить, выделяется ли определенный процент значений или определенное количество значений.

**Вызов**

```
bool getUsePercent()
```

**Возвращает**

– `true`, если условное форматирование выделяет определенный процент значений; если выделяется определенное количество значений – `false`.

### 5.207.4 Метод TopBottomConditionalFormatOperator:getValue

Метод возвращает количество/процент значений для выделения.

**Вызов**

```
number getValue()
```

**Возвращает**

– количество/процент значений для выделения, тип `number`.

## 5.208 Таблица DocumentAPI.TrackedChange

Таблица `DocumentAPI.TrackedChange` представляет отслеживаемое изменение в диапазоне текстового документа (см. Рисунок 64).

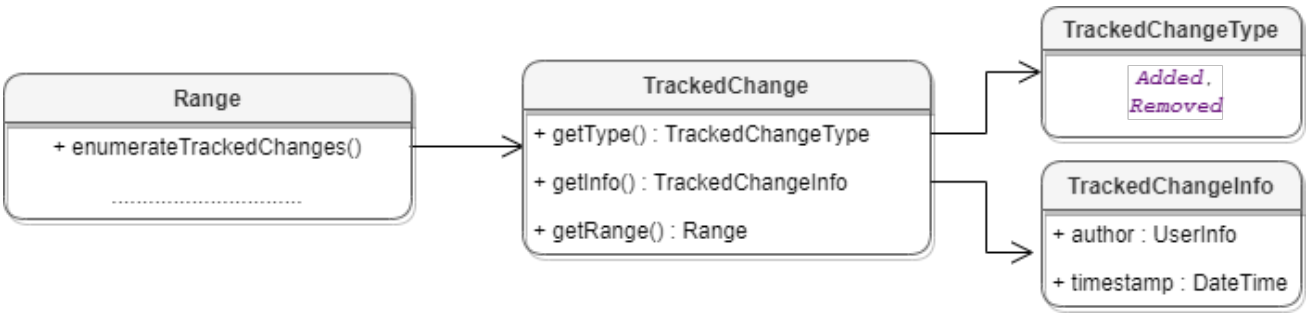



Рисунок 64 – Объектная модель таблиц для работы с отслеживаемыми изменениями

Для получения списка отслеживаемых изменений используется метод `Range.enumerateTrackedChanges()`.

**Пример**

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
  print(change:getRange():extractText())
end
```

### 5.208.1 Метод TrackedChange:getInfo

Метод позволяет получить информацию об отслеживаемых изменениях (`DocumentAPI.TrackedChangeInfo`).

**Пример**

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
  print(tracked_change:getInfo().author.name)
end
```

### 5.208.2 Метод TrackedChange:getRange

Метод возвращает объект `DocumentAPI.Range`, который соответствует измененному диапазону внутри абзаца.

**Пример**

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
```

```
  print(tracked_change:getRange():extractText())
end
```

### 5.208.3 Метод TrackedChange:getType

Метод позволяет получить информацию о типе отслеживаемого изменения DocumentAPI.TrackedChangeType.

**Пример**

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
  print(tracked_change:getType())
end
```

## 5.209 Таблица DocumentAPI.TrackedChangeInfo

Таблица DocumentAPI.TrackedChangeInfo содержит информацию об отслеживаемых изменениях. Используется в TrackedChange:getInfo, Comment:getInfo.

Таблица 124 – Описание полей таблицы DocumentAPI.TrackedChangeInfo

| Поле | Тип | Описание |
|---|---|---|
| TrackedChangeInfo.author | UserInfo | Автор изменений |
| TrackedChangeInfo.timeStamp | DateTime | Дата и время изменений |

**Пример**

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
  local trackedChangeInfo = change:getInfo()
  local author = trackedChangeInfo.author
  local ts = trackedChangeInfo.timeStamp
  local ts_msg = string.format("%d/%d/%d - %d:%d:%d", ts.day, ts.month, ts.year,
ts.hour, ts.minute, ts.second)
  print(author.name, ts_msg)
end
```

### 5.209.1 Метод TrackedChangeInfo:__eq

Метод используется для определения эквивалентности двух отслеживаемых изменений.

## 5.210 Таблица DocumentAPI.TrackedChangeType

Таблица `TrackedChangeType` содержит типы отслеживаемых изменений, возвращается методом `TrackedChange:getType()`.

Таблица 125 – Типы отслеживаемых изменений

| Значение | Описание |
|---|---|
| `DocumentAPI.TrackedChangeType_Added` | Добавленные изменения |
| `DocumentAPI.TrackedChangeType_Removed` | Удаленные изменения |

**Пример**

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
  if DocumentAPI.TrackedChangeType_Added == change:getType() then action =
"Добавлено: " else action = "Удалено: " end
  print(action)
end
```

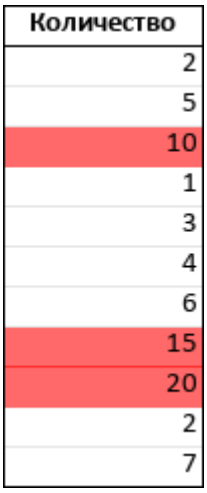
## 5.211 Таблица DocumentAPI.UnaryConditionalFormatOperator

Таблица `UnaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

**Конструктор**

```
      UnaryConditionalFormatOperator(ConditionalFormatUnaryCondition condition,
string argument)
```

**Пример**



| Количество |
|---|
| 2 |
| 5 |
| 10 |
| 1 |
| 3 |
| 4 |
| 6 |
| 15 |
| 20 |
| 2 |
| 7 |

Рисунок 65 – Пример создания правила для значений >= 10

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local unaryStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
unaryStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("F2:F12")
local cellRangePosition = cellRange:getTableRange()

local unaryOperator =
DocumentAPI.createUnaryConditionalFormatOperator(DocumentAPI.ConditionalFormatUna
ryCondition_GreaterOrEqual, "10")

local unaryRule = DocumentAPI.ConditionalFormatRule(unaryOperator, unaryStyle,
cellRangePosition, false)
rules:addRule(unaryRule)
```

### 5.211.1 Метод UnaryConditionalFormatOperator:getArgument

Метод возвращает аргумент условия.

**Вызов**

```
string getArgument()
```

**Возвращает**

– аргумент условия, тип `string`.

### 5.211.2 Метод UnaryConditionalFormatOperator:getCondition

Метод возвращает условие применения форматирования.

**Вызов**

```
ConditionalFormatUnaryCondition getCondition()
```

**Возвращает**

– условие применения форматирования, тип ConditionalFormatUnaryCondition.

### 5.211.3 Метод UnaryConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

## 5.212 Таблица DocumentAPI.UniquenessConditionalFormatOperator

Таблица `UniquenessConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Уникальные и повторяющиеся значения". Используется при создании правила условного форматирования (ConditionalFormatRule).

**Конструктор**

```
UniquenessConditionalFormatOperator(ConditionalFormatUniquenessCondition
condition)
```

**Пример**

| Город |
| --- |
| Москва |
| Санкт-Петербург |
| Алматы |
| Минск |
| Екатеринбург |
| Казань |
| Новосибирск |
| Самара |
| Астана |
| Екатеринбург |
| Минск |

Рисунок 66 – Пример создания правила для повторяющихся значений

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()

local uniqueStyle = DocumentAPI.ConditionalFormatCellStyle()
local cellProperties = DocumentAPI.CellProperties()
cellProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 150)))
uniqueStyle.cellProperties = cellProperties

local cellRange = sheet:getCellRange("D2:D12")
local cellRangePosition = cellRange:getTableRange()

local uniqueOperator =
DocumentAPI.createUniquenessConditionalFormatOperator(DocumentAPI.ConditionalForm
atUniquenessCondition_Duplicate)

local uniqueRule = DocumentAPI.ConditionalFormatRule(uniqueOperator, uniqueStyle,
cellRangePosition, false)
rules:addRule(uniqueRule)
```

### 5.212.1 Метод UniquenessConditionalFormatOperator:getCondition

Метод возвращает условие применения форматирования.

**Вызов**

```
ConditionalFormatUniquenessCondition getCondition()
```

**Возвращает**

– условие применения форматирования, тип ConditionalFormatUniquenessCondition.

### 5.212.2 Метод UniquenessConditionalFormatOperator:getType

Метод возвращает тип оператора условного форматирования.

**Вызов**

```
ConditionalFormatOperatorType getType()
```

**Возвращает**

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

## 5.213 Таблица DocumentAPI.UserInfo

Таблица DocumentAPI.UserInfo предоставляет информацию о пользователе. Используется в поле author таблицы DocumentAPI.TrackedChangeInfo.

Таблица 126 – Описание полей таблицы DocumentAPI.UserInfo

| Поле | Тип | Описание |
|---|---|---|
| UserInfo.name | string | Имя пользователя |
| UserInfo.email | string | Адрес электронной почты пользователя |

## 5.214 Таблица DocumentAPI.ValueFieldsOrientation

Таблица DocumentAPI.ValueFieldsOrientation описывает варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений. Является полем таблицы DocumentAPI.PivotTableLayoutSettings.

Таблица 127 – Варианты ориентации значений

| Значение | Описание |
|---|---|
| DocumentAPI.ValueFieldsOrientation_ByRows | По строкам |
| DocumentAPI.ValueFieldsOrientation_ByColumns | По столбцам |

### 5.215 Таблица DocumentAPI.ValuesTableFilter

Таблица `ValuesTableFilter` реализует фильтр, содержащий значения, которые должны быть показаны в диапазоне фильтрации.

Конструктор по умолчанию:

```
local firstFilter = DocumentAPI.ValuesTableFilter()
```

Конструктор копирования:

```
local secondFilter = DocumentAPI.ValuesTableFilter(firstFilter)
```

Пример использования приведен в разделе Работа с фильтрами.

#### 5.215.1 Метод ValuesTableFilter:add

Метод `ValuesTableFilter::add` добавляет значение, которое должно быть отображено в таблице.

**Пример**

```
local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")
```

#### 5.215.2 Метод ValuesTableFilter:clear

Метод `ValuesTableFilter::clear` удаляет все элементы фильтра.

**Пример**

```
local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")
.........................
johnPaulFilter:clear()
```

#### 5.215.3 Метод ValuesTableFilter:remove

Метод удаляет заданное значение из фильтра.

**Вызов**

```
remove(value)
```

**Параметры**

– `value`: значение фильтра, тип `string`.

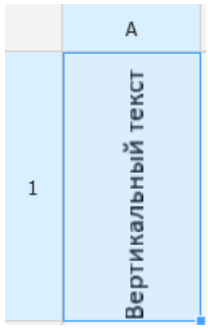
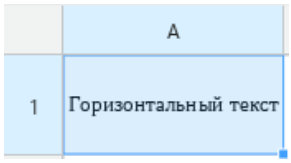
**Пример**

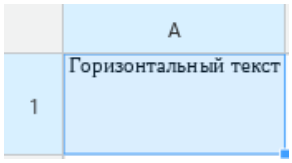
```
johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")
-- ...
johnPaulFilter:remove("Sam")
```

### 5.216 Таблица DocumentAPI.VerticalAlignment

В таблице 128 представлены константы видов выравнивания текста по вертикали. Используется в `DocumentAPI.CellProperties`, `DocumentAPI.ShapeProperties`.

Таблица 128 – Виды выравнивания текста по вертикали

| Значение | Представление в интерфейсе | |
|---|---|---|
| `DocumentAPI.VerticalAlignment_Bottom` |  |  |
| `DocumentAPI.VerticalAlignment_Center` |  |  |

| Значение | Представление в интерфейсе | |
|---|---|---|
| DocumentAPI.VerticalAlignment_Top |  |  |

**Пример**

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A1")
local props = cell:getCellProperties()
props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
cell:setCellProperties(props)
```

### 5.217 Таблица DocumentAPI.VerticalAnchorAlignment

В таблице 129 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали. Используется в DocumentAPI.VerticalTextAnchoredPosition.

Таблица 129 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

| Значение | Описание |
|---|---|
| DocumentAPI.VerticalAnchorAlignment_Top | По верхнему краю |
| DocumentAPI.VerticalAnchorAlignment_Bottom | По нижнему краю |
| DocumentAPI.VerticalAnchorAlignment_Center | По центру |
| DocumentAPI.VerticalAnchorAlignment_Inside<br>DocumentAPI.VerticalAnchorAlignment_Outside | По границам |

### 5.218 Таблица DocumentAPI.VerticalRelativeTo

В таблице 130 представлены типы размещения объекта относительно закрепленной позиции по вертикали. Используется в DocumentAPI.VerticalTextAnchoredPosition.

Таблица 130 – Типы размещения объекта относительно закрепленной позиции по вертикали

| Значение | Описание |
|---|---|
| `DocumentAPI.VerticalRelativeTo_Character` | Символ |
| `DocumentAPI.VerticalRelativeTo_BaseLine` | Базовая линия |
| `DocumentAPI.VerticalRelativeTo_Paragraph` | Абзац |
| `DocumentAPI.VerticalRelativeTo_Page` | Страница |
| `DocumentAPI.VerticalRelativeTo_PageContent` | Содержимое страницы |
| `DocumentAPI.VerticalRelativeTo_PageTopMargin` | Верхнее поле страницы |
| `DocumentAPI.VerticalRelativeTo_PageBottomMargin` | Нижнее поле страницы |
| `DocumentAPI.VerticalRelativeTo_PageInsideMargin` | Внутреннее поле страницы |
| `DocumentAPI.VerticalRelativeTo_PageOutsideMargin` | Внешнее поле страницы |

### 5.219 Таблица DocumentAPI.VerticalTextAnchoredPosition

Таблица `DocumentAPI.VerticalTextAnchoredPosition` предназначена для управления относительным положением объекта со смещением или выравниванием по вертикали. Пример использования см. в `InlineFrame:setPosition()`.

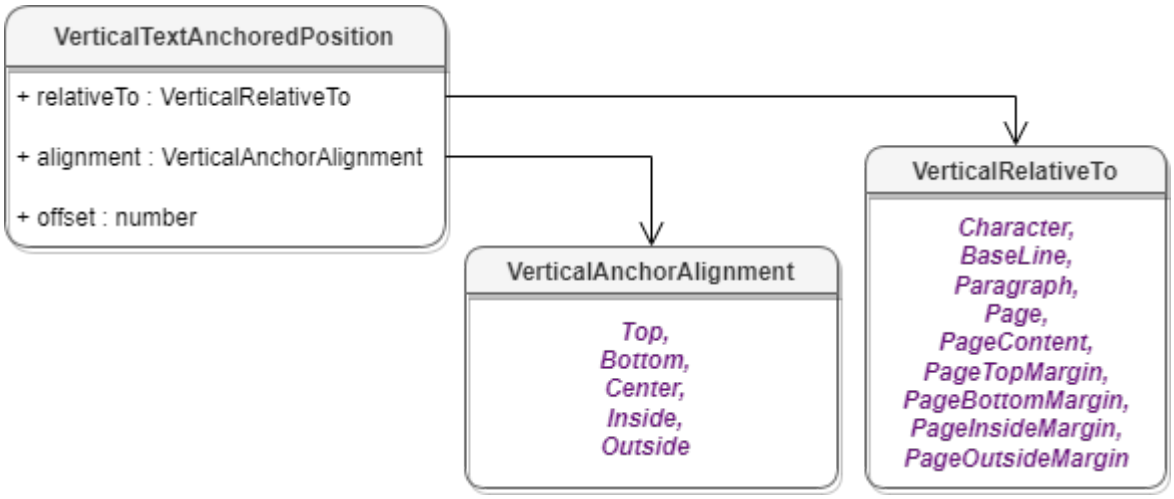



Рисунок 67 – Поля таблицы `DocumentAPI.VerticalTextAnchoredPosition`

Описание полей таблицы `DocumentAPI.VerticalTextAnchoredPosition` представлено в таблице 131.

Таблица 131 – Описание полей таблицы `DocumentAPI.VerticalTextAnchoredPosition`

| Поле | Тип | Описание |
|---|---|---|
| `VerticalTextAnchoredPosition.relativeTo` | `VerticalRelativeTo` | Тип размещения объекта относительно закрепленной позиции по вертикали |
| `VerticalTextAnchoredPosition.offset` | `number` | Смещение объекта |
| `VerticalTextAnchoredPosition.alignment` | `VerticalAnchorAlignment` | Тип выравнивания объекта относительно закрепленной позиции по вертикали |

### 5.219.1 Метод VerticalTextAnchoredPosition:__eq

Метод используется для определения эквивалентности двух положений объекта по вертикали.

**Пример**

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
pos1.vertical.offset = 1

local pos2 = DocumentAPI.TextAnchoredPosition()
pos2.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
pos2.vertical.offset = 1

print(pos1.vertical:__eq(pos2.vertical))
```

### 5.220 Таблица DocumentAPI.WorksheetPrinterFitType

В таблице 132 представлены варианты масштабирования при печати табличных документов. Используется в качестве поля `worksheetPrinterFitType` таблицы `DocumentAPI.PrintSettings`.

Таблица 132 – Варианты масштабирования при печати табличных документов

| Значение | Описание |
|---|---|
| `DocumentAPI.WorksheetPrinterFitType_ActualSize` | Фактический размер |
| `DocumentAPI.WorksheetPrinterFitType_ByPageScale` | По масштабу страницы |
| `DocumentAPI.WorksheetPrinterFitType_ByPageBreaksOnly` | По разрыву страниц |
| `DocumentAPI.WorksheetPrinterFitType_FitToPage` | Вписать в страницу |
| `DocumentAPI.WorksheetPrinterFitType_FitToWidth` | Вписать по ширине |
| `DocumentAPI.WorksheetPrinterFitType_FitToHeight` | Вписать по высоте |

# 6 СПРАВОЧНИК ФУНКЦИЙ DOCUMENTAPI

В данном разделе приведено описание функций пространства имен `DocumentAPI`.

## 6.1 Функция DocumentAPI.createSearch

Функция инициализирует механизм поиска для текущего документа. Возвращает ссылку на таблицу `DocumentAPI.Search`, с помощью методов которой выполняются поисковые запросы.

**Пример**

```
search = DocumentAPI.createSearch(document)
ranges = search:findText("English")
```

## 6.2 Функция DocumentAPI.createScripting

Функция `DocumentAPI.createScripting` возвращает таблицу `DocumentAPI.Scripting`. В качестве параметра используется текущий документ.

**Пример**

```
scripting = DocumentAPI.createScripting(document)
```

## 6.3 Функции условного форматирования

Данный раздел содержит функции, которые могут использоваться для создания и приведения операторов условного форматирования.

### 6.3.1 Функция DocumentAPI.castToAboveAverageConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к `AboveAverageConditionalFormatOperator`.

**Вызов**

```
    static                                  AboveAverageConditionalFormatOperator
castToAboveAverageConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

**Возвращает**

– оператор для правил "Выше среднего" и "Ниже среднего", тип AboveAverageConditionalFormatOperator.

– nil, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_AboveAverage
then
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.2 Функция DocumentAPI.castToBinaryConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к BinaryConditionalFormatOperator.

**Вызов**

```
      static                                          BinaryConditionalFormatOperator
castToBinaryConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

**Возвращает**

– оператор для правил "Между" и "Не между", тип BinaryConditionalFormatOperator.

– nil, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_Binary then
```

```
        aboveOperator =
DocumentAPI.castToBinaryConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.3 Функция DocumentAPI.castToColorScaleConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к ColorScaleConditionalFormatOperator.

**Вызов**

```
    static                                  ColorScaleConditionalFormatOperator
castToColorScaleConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип ConditionalFormatOperator.

**Возвращает**

– оператор для правила "Цветовая шкала", тип ColorScaleConditionalFormatOperator.

– `nil`, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_ColorScale
then
        aboveOperator =
DocumentAPI.castToColorScaleConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.4 Функция DocumentAPI.castToDataBarConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к DataBarConditionalFormatOperator.

**Вызов**

```
static DataBarConditionalFormatOperator
castToDataBarConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

**Возвращает**

– оператор для правила "Гистограмма", тип `DataBarConditionalFormatOperator`.

– `nil`, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_DataBar then
        aboveOperator =
DocumentAPI.castToDataBarConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.5 Функция DocumentAPI.castToIconSetConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к `IconSetConditionalFormatOperator`.

**Вызов**

```
static IconSetConditionalFormatOperator
castToIconSetConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

**Возвращает**

– оператор для правила "Значки", тип `IconSetConditionalFormatOperator`.

– `nil`, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
```

```
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_IconSet then
        aboveOperator =
DocumentAPI.castToIconSetConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.6 Функция DocumentAPI.castToNullaryConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к NullaryConditionalFormatOperator.

**Вызов**

```
    static                                          NullaryConditionalFormatOperator
castToNullaryConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип ConditionalFormatOperator.

**Возвращает**

– оператор для правил без параметров, тип NullaryConditionalFormatOperator.

– `nil`, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_Nullary then
        aboveOperator =
DocumentAPI.castToNullaryConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.7 Функция DocumentAPI.castToTextConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к TextConditionalFormatOperator.

**Вызов**

```
static                                          TextConditionalFormatOperator
castToTextConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

**Возвращает**

– оператор для правила "Текст", тип `TextConditionalFormatOperator`.

– `nil`, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_Text then
        aboveOperator =
DocumentAPI.castToTextConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.8 Функция DocumentAPI.castToTopBottomConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к `TopBottomConditionalFormatOperator`.

**Вызов**

```
static                                      TopBottomConditionalFormatOperator
castToTopBottomConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

**Возвращает**

– оператор для правила "Наибольшие и наименьшие значения", тип `TopBottomConditionalFormatOperator`.

– `nil`, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_TopBottom then
        aboveOperator =
DocumentAPI.castToTopBottomConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.9 Функция DocumentAPI.castToUnaryConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к UnaryConditionalFormatOperator.

**Вызов**

```
      static                                    UnaryConditionalFormatOperator
castToUnaryConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– `conditionalFormatOperator`: оператор условного форматирования, тип ConditionalFormatOperator.

**Возвращает**

– оператор для правил "Больше", "Меньше", "Равно" и "Не равно", тип UnaryConditionalFormatOperator.

– `nil`, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_Unary then
        aboveOperator =
DocumentAPI.castToUnaryConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.10 Функция DocumentAPI.castToUniquenessConditionalFormat

Функция позволяет привести базовый оператор условного форматирования к UniquenessConditionalFormatOperator.

**Вызов**

```
static UniquenessConditionalFormatOperator
castToUniquenessConditionalFormat(conditionalFormatOperator)
```

**Параметры**

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

**Возвращает**

– оператор для правила "Уникальные и повторяющиеся значения", тип UniquenessConditionalFormatOperator.

– nil, если приведение невозможно.

**Пример**

```
local sheet = document:getBlocks():getTable(0)
local rules = sheet:getConditionalFormatRules()
local aboveOperator
for rule in rules:enumerate() do
    if rule:getType() == DocumentAPI.ConditionalFormatOperatorType_Uniqueness
then
        aboveOperator =
DocumentAPI.castToUniquenessConditionalFormat(rule:getOperator())
    end
end
```

### 6.3.11 Функция DocumentAPI.createAboveAverageConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правил "Выше среднего" и "Ниже среднего". Пример использования смотри в разделе AboveAverageConditionalFormatOperator.

**Вызов**

```
static ConditionalFormatOperator
createAboveAverageConditionalFormatOperator(condition, stdDev)
```

**Параметры**

– `condition`: условие применения форматирования, тип `ConditionalFormatAboveAverageCondition`.

– `stdDev`: (необязательный) количество стандартных отклонений, тип `number`.

**Возвращает**

– оператор условного форматирования, тип `ConditionalFormatOperator`.

### 6.3.12 Функция DocumentAPI.createBinaryConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правил "Между" и "Не между". Пример использования смотри в разделе `BinaryConditionalFormatOperator`.

**Вызов**

```
static ConditionalFormatOperator createBinaryConditionalFormatOperator(condition, firstArgument, secondArgument)
```

**Параметры**

– `condition`: условие применения форматирования, тип `ConditionalFormatBinaryCondition`.

– `firstArgument`: первый аргумент, тип `string`.

– `secondArgument`: второй аргумент, тип `string`.

**Возвращает**

– оператор условного форматирования, тип `ConditionalFormatOperator`.

### 6.3.13 Функция DocumentAPI.createColorScaleConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правила "Цветовая шкала". Пример использования смотри в разделе `ColorScaleConditionalFormatOperator`.

**Вызов**

```
static ConditionalFormatOperator createColorScaleConditionalFormatOperator(entries)
```

**Параметры**

– `entries`: набор правил применения цветов, тип `ConditionalFormatColorScaleEntries`.

**Возвращает**

– оператор условного форматирования, тип `ConditionalFormatOperator`.

### 6.3.14 Функция DocumentAPI.createDataBarConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правила "Гистограмма". Пример использования смотри в разделе `DataBarConditionalFormatOperator`.

**Вызов**

```
static ConditionalFormatOperator createDataBarConditionalFormatOperator(params)
```

**Параметры**

– `params`: настройки гистограммы, тип `ConditionalFormatDataBarParams`.

**Возвращает**

– оператор условного форматирования, тип `ConditionalFormatOperator`.

### 6.3.15 Функция DocumentAPI.createIconSetConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правила "Значки". Пример использования смотри в разделе `IconSetConditionalFormatOperator`.

**Вызов**

```
static ConditionalFormatOperator createIconSetConditionalFormatOperator(entries, isValueShown)
```

**Параметры**

– `entries`: набор правил отображения значков, тип `ConditionalFormatIconSetEntries`.

– `isValueShown`: `true`, чтобы показывать значение ячейки при примененном форматировании, в ином случае – `false`.

**Возвращает**

– оператор условного форматирования, тип ConditionalFormatOperator.

### 6.3.16 Функция DocumentAPI.createNullaryConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правил без параметров. Пример использования смотри в разделе NullaryConditionalFormatOperator.

**Вызов**

```
static ConditionalFormatOperator createNullaryConditionalFormatOperator(condition)
```

**Параметры**

– condition: условие применения форматирования, тип ConditionalFormatNullaryCondition.

**Возвращает**

– оператор условного форматирования, тип ConditionalFormatOperator.

### 6.3.17 Функция DocumentAPI.createTextConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правила "Текст". Пример использования смотри в разделе TextConditionalFormatOperator.

**Вызов**

```
static ConditionalFormatOperator createTextConditionalFormatOperator(condition, argument)
```

**Параметры**

– condition: условие применения форматирования, тип ConditionalFormatTextCondition.

– argument: аргумент условия, тип string.

**Возвращает**

– оператор условного форматирования, тип ConditionalFormatOperator.

### 6.3.18 Функция DocumentAPI.createTopBottomConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правила "Наибольшие и наименьшие значения". Пример использования смотри в разделе TopBottomConditionalFormatOperator.

**Вызов**

```
static ConditionalFormatOperator createTopBottomConditionalFormatOperator(condition, value, usePercent)
```

**Параметры**

– `condition`: условие применения форматирования, тип ConditionalFormatTopBottomCondition.

– `value`: количество/процент значений для выделения, тип `number`.

– `usePercent`: `true`, чтобы выделять определенный процент значений; чтобы выделять определенное количество значений – `false`.

**Возвращает**

– оператор условного форматирования, тип ConditionalFormatOperator.

### 6.3.19 Функция DocumentAPI.createUnaryConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Пример использования смотри в разделе UnaryConditionalFormatOperator.

**Вызов**

```
static ConditionalFormatOperator createUnaryConditionalFormatOperator(condition, argument)
```

**Параметры**

– `condition`: условие применения форматирования, тип ConditionalFormatUnaryCondition.

– `argument`: аргумент условия, тип `string`.

**Возвращает**

– оператор условного форматирования, тип ConditionalFormatOperator.

### 6.3.20 Функция DocumentAPI.createUniquenessConditionalFormatOperator

Функция создает оператор, который содержит настройки применения форматирования для правила "Уникальные и повторяющиеся значения". Пример использования смотри в разделе UniquenessConditionalFormatOperator.

**Вызов**

```
static ConditionalFormatOperator createUniquenessConditionalFormatOperator(condition)
```

**Параметры**

– `condition`: условие применения форматирования, тип ConditionalFormatUniquenessCondition.

**Возвращает**

– оператор условного форматирования, тип ConditionalFormatOperator.

# 7 СПРАВОЧНИК ТАБЛИЦ EDITORAPI

В данном разделе приведено описание таблиц пространства имен `EditorAPI`.

## 7.1 Таблица EditorAPI.SelectionDirection

Таблица содержит параметры для управления направлением выделения. Используется в качестве аргумента метода `EditorAPI.changeSelection()`. Описание полей таблицы `EditorAPI.SelectionDirection` представлено в таблице 133.

Таблица 133 – Описание полей таблицы `DocumentAPI.SelectionDirection`

| Поле | Описание |
|---|---|
| `EditorAPI.SelectionDirection.Up` | Изменить выделение вверх |
| `EditorAPI.SelectionDirection.Down` | Изменить выделение вниз |
| `EditorAPI.SelectionDirection.Right` | Изменить выделение направо |
| `EditorAPI.SelectionDirection.Left` | Изменить выделение налево |
| `EditorAPI.SelectionDirection.DownRight` | Изменить выделение вниз и направо |
| `EditorAPI.SelectionDirection.UpLeft` | Изменить выделение вверх и налево |

**Пример**

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)
```

## 7.2 Таблица EditorAPI.SelectionMode

Таблица содержит варианты изменения текущего выделения. Используется в качестве аргумента метода `EditorAPI.changeSelection()`. Описание полей таблицы `EditorAPI.SelectionMode` представлено в таблице 134.

Таблица 134 – Описание полей таблицы `EditorAPI.SelectionMode`

| Поле | Описание |
|---|---|
| `EditorAPI.SelectionMode.Move` | Переместить выделение |
| `EditorAPI.SelectionMode.Resize` | Изменить размер текущего выделения |

**Пример**

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.UpLeft, EditorAPI.TextSelectionUnit.Word)
```

### 7.3 Таблица EditorAPI.TableSelectionUnit

Таблица содержит параметры для управления шагом выделения в таблице документа. Используется в качестве аргумента метода `EditorAPI.changeSelection()`. Описание полей таблицы `EditorAPI.TableSelectionUnit` представлено в таблице 135.

Таблица 135 – Описание полей таблицы `EditorAPI.TableSelectionUnit`

| Поле | Описание |
|---|---|
| `EditorAPI.TableSelectionUnit.ToEdge` | Направление смены выделения - угол таблицы |
| `EditorAPI.TableSelectionUnit.ToClosestCell` | Направление смены выделения - ближайшая ячейка |

**Пример**

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.UpLeft, EditorAPI.TableSelectionUnit.ToClosestCell)
```

### 7.4 Таблица EditorAPI.TextSelectionUnit

Таблица содержит параметры для управления шагом выделения в тексте документа. Используется в качестве аргумента метода `EditorAPI.changeSelection()`. Описание полей таблицы `EditorAPI.TextSelectionUnit` представлено в таблице 136.

Таблица 136 – Описание полей таблицы `EditorAPI.TextSelectionUnit`

| Поле | Описание |
|---|---|
| `EditorAPI.TextSelectionUnit.Character` | Изменять выделение с шагом в один символ |
| `EditorAPI.TextSelectionUnit.Word` | Изменять выделение с шагом в одно слово |
| `EditorAPI.TextSelectionUnit.Paragraph` | Изменять выделение с шагом в один абзац |
| `EditorAPI.TextSelectionUnit.Line` | Изменять выделение с шагом в одну строку |

**Пример**

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)
```

# 8 СПРАВОЧНИК ФУНКЦИЙ EDITORAPI

Глобальная таблица `EditorAPI` содержит функции доступа к внешней функциональности редактора.

## 8.1 Функция EditorAPI.changeSelection

Функция `EditorAPI.changeSelection` позволяет изменить текущее выделение в текстовом или табличном документе.

### Вызов функции

```
EditorAPI.changeSelection(mode, direction, unit, count = 1)
```

Где:

- `mode` – режим выделения, тип `DocumentAPI.SelectionMode`;
- `direction` – направление выделения, тип `DocumentAPI.SelectionDirection`;
- `unit` – шаг изменения выделения, тип `DocumentAPI.TableSelectionUnit` для таблиц, или `DocumentAPI.TextSelectionUnit` для текста;
- `count` – количество шагов, необязательный параметр, по умолчанию используется значение 1.

Функция возвращает `true` в случае, если выделение было изменено.

### Пример для текстового документа

```
if (EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)) then
   EditorAPI.messageBox("Selection changed")
end
```

### Пример для табличного документа

```
if (EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Right, EditorAPI.TableSelectionUnit.ToClosestCell,
2)) then
   EditorAPI.messageBox("Selection changed")
end
```

### 8.2 Функция EditorAPI.getActiveWorksheet

Функция `EditorAPI.getActiveWorksheet()` возвращает активный лист в табличном документе (класс `Table`). Метод не предназначен для текстовых документов.

**Пример**

```
activeWorksheet = EditorAPI.getActiveWorksheet()
print(activeWorksheet:getName()) -- Лист1
```

### 8.3 Функция EditorAPI.getSelection

Функция `EditorAPI.getSelection()` предоставляет доступ к выделенному фрагменту документа. Выделенный фрагмент может представлять собой диапазон текстового документа (`Range`), встроенный объект (`MediaObject`), диапазон ячеек (`CellRange`) или несколько диапазонов ячеек (`CellRanges`).

В открытом документе может быть выделен только один фрагмент.

**Пример**

```
local selection = EditorAPI.getSelection()
if selection:isInstanceOf(DocumentAPI.Range) then
    print(selection:extractText())
end
if selection:isInstanceOf(DocumentAPI.MediaObject) then
    print(selection:getFrame())
end
if selection:isInstanceOf(DocumentAPI.CellRange) then
    print(selection:getAddress(DocumentAPI.CellRangeAddressSettings()))
end
if selection:isInstanceOf(DocumentAPI.CellRanges) then
    for cellRange in selection:enumerate() do
        print(cellRange:getAddress(DocumentAPI.CellRangeAddressSettings()))
    end
end
```

Если в текстовом документе отсутствует выделенный фрагмент, то функция `EditorAPI.getSelection()` возвращает пустой диапазон (`Range`), который соответствует текущей позиции курсора.

**Вставка текста в позицию курсора**

```
local range = EditorAPI.getSelection()
range:getBegin():insertText("MyText")
```

### 8.4 Функция EditorAPI.isPrinterAvailable

Функция `EditorAPI.isPrinterAvailable` позволяет проверить доступность последнего использованного принтера. Возвращает `false`, если принтер недоступен.

**Пример**

```
if EditorAPI.isPrinterAvailable() then
    EditorAPI.messageBox("Printer is available")
else
    EditorAPI.messageBox("Printer is not available")
end
```

### 8.5 Функция EditorAPI.messageBox

Функция `EditorAPI.messageBox()` выводит на экран сообщение с заданным текстом и отображением кнопки **ОК**, при этом исполнение макрокоманды приостанавливается до нажатия кнопки **ОК**.

**Вызов**

```
messageBox(prompt : string)
messageBox(prompt : string)
messageBox(prompt : string, title : string)
```

**Параметры**

- `prompt` – текст сообщения;
- `title` – заголовок окна сообщения.

**Пример**

```
EditorAPI.messageBox(cell:getFormattedValue())
```

### 8.6 Функция EditorAPI.printDocument

Функция `EditorAPI.printDocument()` предоставляет возможность печати документа с заданными параметрами печати. Описание параметров печати представлено в разделе `DocumentAPI.PrintSettings`.

Значения, возвращаемые функцией `EditorAPI.printDocument()`, перечислены в разделе `DocumentAPI.PrintDocumentResult`.

**Пример**

```
local printSettings = {}
printSettings.printSelection = true
EditorAPI.printDocument(printSettings)
```

### 8.7 Функция EditorAPI.setActiveWorksheet

Функция `EditorAPI.setActiveWorksheet()` устанавливает активный лист в табличном документе. В качестве параметра используется имя листа. Возвращает `true`, если лист найден по имени и активирован. Метод не предназначен для текстовых документов.

**Пример**

```
print(EditorAPI.setActiveWorksheet("Лист1")) -- true or false
```

### 8.8 Функция EditorAPI.setSelection

Функция `EditorAPI.setSelection()` позволяет выделить фрагмент документа. Данная функция может выделить диапазон текстового документа (`Range`), встроенный объект (`MediaObject`), диапазон ячеек (`CellRange`) или несколько диапазонов ячеек (`CellRanges`). Если выделение успешно установлено, функция возвращает `true`.

В открытом документе может быть выделен только один фрагмент.

**Пример выделения диапазона текстового документа**

```
local range = document:getBlocks():getParagraph(0):getRange()
EditorAPI.setSelection(range)
```

**Пример выделения встроенного объекта**

```
local pos = document:getRange():getBegin()
local image = pos:insertImage("image.jpg", DocumentAPI.SizeU(100, 100))
EditorAPI.setSelection(image)
```

**Пример выделения диапазона ячеек**

```
cellRange = document:getBlocks():getTable(0):getCellRange("A1:E5")
EditorAPI.setSelection(cellRange)
```

**Пример выделения нескольких диапазонов ячеек**

```
local cellRanges = DocumentAPI.CellRanges(document)
local sheet = document:getBlocks():getTable(0)
cellRanges:addRange(sheet:getCellRange("A1:B2"))
cellRanges:addRange(sheet:getCellRange("C3:D4"))
EditorAPI.setSelection(cellRanges)
```

### 8.9 Функция EditorAPI.showPrintDialog

Функция `EditorAPI.showPrintDialog()` показывает стандартное окно печати редактора и распечатывает документ, если пользователь подтверждает необходимость печати. Значения, возвращаемые функцией `EditorAPI.showPrintDialog()` перечислены в разделе `DocumentAPI.PrintDocumentResult`.

**Пример**

```
printDocumentResult = EditorAPI.showPrintDialog()
print(printDocumentResult)
```

# 9 СПРАВОЧНИК МЕТОДОВ EVENTSAPI

Глобальная таблица `EventsAPI` содержит методы, которые позволяют подписаться на события редактора табличных документов (см. Обработка событий табличного документа).

## 9.1 Метод EventsAPI.subscribeWorkbookBeforeClose

Метод `EventsAPI.subscribeWorkbookBeforeClose` позволяет задать функцию, которая выполняется перед закрытием документа.

**Пример**

```
EventsAPI.subscribeWorkbookBeforeClose(function(toCancel)
    EditorAPI.messageBox(string.format('Macro "BeforeClose" is working!
(toCancel=%s)', tostring(toCancel)))
    return not toCancel
end)
```

**Параметры**

– `toCancel`: `false`, если документ будет закрыт после завершения события.

Если передаваемая функция возвращает `true`, то стандартный обработчик закрытия документа не будет вызван.

При одновременном закрытии нескольких окон с документами, выполнится только одно событие `subscribeWorkbookBeforeClose`, закрытие остальных окон будет отменено.

## 9.2 Метод EventsAPI.subscribeWorkbookBeforeSave

Метод `EventsAPI.subscribeWorkbookBeforeSave` позволяет задать функцию, которая выполняется перед сохранением документа.

**Пример**

```
EventsAPI.subscribeWorkbookBeforeSave(function(isSaveAs, toCancel)
    EditorAPI.messageBox(string.format('Macro "BeforeSave" is working! (isSaveAs=
%s; toCancel=%s)', tostring(isSaveAs), tostring(toCancel)))
    return not toCancel
end)
```

**Параметры**

– `isSaveAs`: `true`, если событие было вызвано функцией "Сохранить как";

– `toCancel`: `false`, если документ будет сохранен после завершения события.

Если передаваемая функция возвращает `true`, то стандартный обработчик сохранения документа не будет вызван.

Событие `subscribeWorkbookBeforeSave` не выполняется при совместном редактировании в облаке.

### 9.3 Метод EventsAPI.subscribeWorkbookOpen

Метод `EventsAPI.subscribeWorkbookOpen` позволяет задать функцию, которая выполняется при открытии существующего документа.

**Пример**

```
EventsAPI.subscribeWorkbookOpen(function()
    EditorAPI.messageBox("Macro 'Open' is working")
end)
```

### 9.4 Метод EventsAPI.subscribeWorksheetActivate

Метод `EventsAPI.subscribeWorksheetActivate` позволяет задать функцию, которая выполняется при переключении между листами документа.

**Пример**

```
EventsAPI.subscribeWorksheetActivate(function(sheet)
    EditorAPI.messageBox(string.format('Macro "Activate" is working! (%s)',
sheet:getName()))
end)
```

**Параметры**

– `sheet` (необязательный): лист, на который переключились, тип `DocumentAPI.Table`.

### 9.5 Метод EventsAPI.subscribeWorksheetChange

Метод `EventsAPI.subscribeWorksheetChange` позволяет задать функцию, которая выполняется при изменении содержимого ячейки или нескольких ячеек.

**Пример**

```
EventsAPI.subscribeWorksheetChange(function(target)
    EditorAPI.messageBox(
        string.format('Macro "Change" is working! (BeginCell (%i, %i), LastCell
```

```
(%i, %i)',
            target:getBeginRow()+1,
            target:getBeginColumn()+1,
            target:getLastRow()+1,
            target:getLastColumn()+1))
end)
```

**Параметры**

– `target`: измененный диапазон, тип `DocumentAPI.CellRange`.

Событие `subscribeWorksheetChange` не выполняется при:

– изменении стилей;

– изменении формата;

– изменении выравнивания;

– добавлении / удалении / изменении гиперссылок;

– добавлении / удалении комментариев;

– фильтрации / сортировке;

– скрытии строки / столбца;

– пересчете формул.

### 9.6 Метод EventsAPI.subscribeWorksheetDeactivate

Метод `EventsAPI.subscribeWorksheetDeactivate` позволяет задать функцию, которая выполняется при переключении между листами документа.

**Пример**

```
EventsAPI.subscribeWorksheetDeactivate(function(sheet)
    EditorAPI.messageBox(string.format('Macro "Deactivate" is working! (%s)',
sheet and sheet:getName() or 'nil'))
end)
```

**Параметры**

– `sheet` (необязательный): лист, с которого переключились, тип `DocumentAPI.Table`. Возвращает `nil`, если переключение вызвано удалением листа.

### 9.7 Метод EventsAPI.subscribeWorksheetSelectionChange

Метод `EventsAPI.subscribeWorksheetSelectionChange` позволяет задать

функцию, которая выполняется при изменении выделения в документе.

### Пример

```
EventsAPI.subscribeWorksheetSelectionChange(function(target)
    EditorAPI.messageBox(
        string.format('Macro "SelectionChange" is working! (BeginCell (%i, %i), LastCell (%i, %i)',
            target:getBeginRow()+1,
            target:getBeginColumn()+1,
            target:getLastRow()+1,
            target:getLastColumn()+1))
end)
```

### Параметры

– `target`: новая область выделения, тип DocumentAPI.CellRange.

## 10 ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ В ФОРМАТЕ ЮНИКОД (UTF-8)

Для работы со строками, содержащими русские символы, можно использовать методы таблицы `utf8`. Предполагается, что аргументы методов являются допустимыми строками UTF-8.

### 10.1 Функция utf8.char

Функция `utf8.char` возвращает строку в формате UTF-8, соответствующую коду символа.

**Вызов**

```
utf8.char(code)
```

**Параметры**

– `code`: код символа UTF-8, тип `number`.

**Возвращает**

– `string`: символ UTF-8, полученный по коду.

**Пример**

```
print(utf8.char(244)) --  ô
```

### 10.2 Функция utf8.charpattern

Функция `utf8.charpattern` возвращает шаблон `"[\0-\x7F\xC2-\xF4][\x80-\xBF]*"` для определения последовательности символов формата UTF-8.

**Пример**

```
function len(s)
   local n = 0
   for match in s:gmatch(utf8.charpattern) do
      n = n + 1
   end
   return n
end

str = "МойОфис"
print(len(str))
```

### 10.3 Функция utf8.codepoint

Функция `utf8.codepoint` возвращает код заданного символа.

**Вызов**

```
utf8.codepoint(char)
```

**Параметры**

– `char`: символ UTF-8, тип `utf-char`.

**Возвращает**

– `number`: код символа UTF-8.

**Примеры**

```
print(utf8.codepoint("")) --  29790
print(utf8.codepoint("A")) --  1040
```

### 10.4 Функция utf8.codes

Функция `utf8.codes` возвращает последовательность кодов символов, из которых состоит строка UTF-8.

**Вызов**

```
utf8.codes(str)
```

**Параметры**

– `str`: строка в формате UTF-8

**Возвращает**

– итератор, с помощью которого можно получить коды символов исходной строки.

**Пример**

```
str = "МойОфис"
for p, c in utf8.codes(str) do
   print(c)
end

-- 1052, 1086, 1081, 1054, 1092, 1080, 1089
```

### 10.5 Функция utf8.compare

Функция `utf8.compare` возвращает результат сравнения двух строк согласно алгоритму сортировки по Юникоду.

**Вызов**

```
utf8.compare(str1, str2, opt)
```

**Параметры**

– `str1` – первая строка (`string`) в формате UTF-8;

– `str2` – вторая строка (`string`) в формате UTF-8;

– `opt` – параметр (`number`) учета регистра при сравнении:

- `0` – без учета регистра;
- `1` – с учетом регистра.

**Возвращает**

– `number`: результат сравнения аргументов:

- -1 – если str1 < str2;
- 0 – если str1 = str2;
- 1 – если str1 > str2.

**Пример**

```
print(utf8.compare("A", "a",  0)) --  0, arg1 = arg2 с учетом регистра
print(utf8.compare("A", "a",  1)) -- -1, arg1 < arg2 без учета регистра
```

### 10.6 Функция utf8.isalpha

Функция `utf8.isalpha` проверяет, является ли буквенным символом переданный символ или число.

```
utf8.isalpha(char)
```

**Параметр**

– `char`: символ в кодировке UTF-8.

**Возвращает**

– `boolean`: `true`, если передан код буквенного символа.

**Пример**

```
print(utf8.isalpha('A')) -- true
print(utf8.isalpha('1')) -- false
```

### 10.7 Функция utf8.isdigit

Функция `utf8.isdigit` проверяет, является ли цифровым символом переданный символ или число.

```
utf8.isdigit(char)
```

**Параметр**

– `char: UTF-8-character` – символ в кодировке UTF-8.

**Возвращает**:

– `boolean:` значение `true`, если передан код цифрового символа.

**Пример**

```
print(utf8.isdigit("a")) -- false
print(utf8.isdigit("1")) -- true
```

### 10.8 Функция utf8.islower

Функция `utf8.islower` проверяет, находится ли в нижнем регистре переданный символ или строка.

```
utf8.islower(str)
```

**Параметр**

– `str:` строка, символ или число, представляющее код UTF-8.

**Возвращает**:

– `boolean: true`, если передан код символа в нижнем регистре.

**Пример**

```
print(utf8.islower("a")) -- false
print(utf8.islower("A")) -- true
```

### 10.9 Функция utf8.isupper

Функция `utf8.isupper` проверяет, находится ли в верхнем регистре переданный символ или строка.

```
utf8.isupper(str)
```

**Параметр**

– `str:` строка, символ или число, представляющее код UTF-8.

**Возвращает**:

– `boolean:` `true`, если передан код символа в верхнем регистре.

**Пример**

```
print(utf8.islower("a")) -- false
print(utf8.islower("A")) -- true
```

### 10.10 Функция utf8.len

Функция `utf8.len` позволяет определить длину заданной строки в символах.

```
utf8.len(str)
```

**Параметр**

– `str: string` – строка в формате UTF-8.

**Возвращает**:

– `number:` длина заданной строки в символах.

**Пример**

```
print(utf8.len("МойОфис")) -- 7
```

### 10.11 Функция utf8.lower

Функция `utf8.lower` возвращает строку в формате UTF-8, полученную из исходной строки путем преобразования в нижний регистр.

**Вызов**

```
utf8.lower(str)
```

**Параметры**

– `str:` строка в формате UTF-8

**Возвращает**

– `string:` строка в нижнем регистре.

**Пример**

```
print(utf8.lower("A")) -- a
print(utf8.lower("Abc")) -- abc
```

### 10.12 Функция utf8.next

Функция `utf8.next` позволяет получить байтовое смещение символа, следующего за указанным.

**Вызов**

```
utf8.next(str, offset)
```

**Параметры**

– `str` – строка (`string`) в формате UTF-8;

– `offset` – байтовое смещение внутри UTF-8 строки (по умолчанию равно 1).

**Возвращает**

– `number` – байтовое смещение следующего символа.

**Пример**

```
next_idx = utf8.next("АБВГДЕЖЗ", 5)
print(next_idx)
```

### 10.13 Функция utf8.offset

Функция `utf8.offset` возвращает позицию (в байтах), с которой начинается кодирование символа с заданной позицией.

```
utf8.offset(str, charPos)
```

**Параметр**

– `str: string` – строка в формате UTF-8;

– `charPos: number` – позиция символа.

**Возвращает**:

– `number:` позиция (в байтах), с которой начинается кодирование символа с заданной позицией.

**Пример**

```
print(utf8.offset("АБВГДЕЖЗ", 5)) -- 9
```

### 10.14 Функция utf8.substr

Функция `utf8.substr` возвращает подстроку в формате UTF-8, начиная с индекса `first` и заканчивая индексом `last`.

```
utf8.substr(str, first[, last])
```

**Параметры**

– `str:` `string` – исходная строка в формате UTF-8;

– `first:` `number` – позиция первого символа подстроки;

– `last:` `number` – позиция последнего символа подстроки (по умолчанию равна позиции последнего символа в строке).

**Возвращает**:

– `string:` подстрока в формате UTF-8

- если позиция первого или последнего символа находится вне строки, то диапазон усекается до корректного;
- если диапазон задан некорректно, то возвращается пустая строка.

**Пример**

```
print(utf8.substr("регистр",  5))    --  стр
print(utf8.substr("регистр",  0, 2)) --  ре
print(utf8.substr("регистр",  2, 1)) --
print(utf8.substr("регистр",  2, 100)) -- егистр
```

### 10.15 Функция utf8.upper

Функция `utf8.upper` возвращает строку в формате UTF-8, полученную из исходной строки путем преобразования в верхний регистр.

**Вызов**

```
utf8.upper(str)
```

**Параметры**

– `str:` строка в формате UTF-8

**Возвращает**

– `string:` строка в верхнем регистре.

**Пример**

```
print(utf8.upper("a")) -- A
print(utf8.upper("Abc")) -- ABC
```

# 11 ФУНКЦИИ ДЛЯ РАБОТЫ С РЕГУЛЯРНЫМИ ВЫРАЖЕНИЯМИ

Для работы с объектами `Regex` можно использовать методы таблицы `Re`, представленные в данном разделе.

## 11.1 Функция Re.create

Функция `Re.create` компилирует регулярное выражение и возвращает его в виде объекта. По умолчанию используется Perl - совместимый формат регулярных выражений.

**Вызов**

```
Re.create(pattern)
```

**Параметры**

- `pattern` (`string`) - строка шаблона.

**Возвращает**

- `regex` (`object`) - объект Regex, который содержит скомпилированное регулярное выражение для дальнейшего использования;
- `err` (`string`) - сообщение об ошибке или `nil`.

## 11.2 Функция Re.match

Сопоставляет скомпилированное регулярное выражение с заданной исходной строкой. Возвращает найденные подстроки.

**Вызов**

```
Re.match(subject, matchFlags, pattern)
```

**Параметры**

- `subject` (`string`) – исходная строка;
- `matchFlags` (`int`) – флаги, задающие правила применения регулярного выражения;
- `pattern` (`string`, `Regex`) – строка шаблона или скомпилированный шаблон.

**Возвращает**

- `matches` (`object`) – подстроки, найденные в соответствии с шаблоном;
- `err` (`string`) - сообщение об ошибке или nil.

### 11.2.1 Флаги, используемые в Re.match

Эти флаги определены в пространстве имен `Re.Match`. Они используются во всех алгоритмах. Когда регулярное выражение применяется к последовательности символов, применяются правила, описанные в таблице 137.

Таблица 137 – Описание флагов `Re.Match`

| Флаг | Описание |
|---|---|
| `Default` | Указывает, что работа с регулярными выражениями происходит в соответствии с обычными правилами: ECMA-262, спецификация языка ECMAScript, глава 15, часть 10, Регулярные Выражения. |
| `NotBOB` | Указывает, что выражения ”\A” и "\`" не должны совпадать с подмножеством *[first, first)*. |
| `NotEOB` | Указывает, что выражения "\`", "\ z" и "\Z" не должны совпадать с подмножеством *[last, last)*. |
| `NotBOL` | Указывает, что выражение "^" не должны совпадать с подмножеством *[first, first)*. |
| `NotEOL` | Указывает, что выражение "$" не должно совпадать с подмножеством *[last, last)*. |
| `NotBOW` | Указывает, что выражения "\<" и "\b" не должны совпадать с подмножеством *[first, first)*. |
| `NotEOW` | Указывает, что выражения "\>" и "\b" не должны совпадать с подмножеством *[last, last)*. |
| `Any` | Указывает, что если существует более одного совпадения, то любое из совпадений является приемлемым результатом. Будет применено самое первое встретившееся совпадение, при этом, не всегда самое точное. Используйте этот флаг, если для вас важна скорость обработки, но не особо важно качество результата. |
| `NotNull` | Указывает, что выражение не может быть использовано для пустой последовательности. |
| `Continuous` | Указывает, что выражение должно применяться к подмножеству, которое начинается с начала. |
| `Partial` | Указывает, что если не найдено ни одного совпадения, то допустимо вернуть совпадение *[from, last)*, при этом *from! = last*, если может существовать более длинная последовательность символов *[from, to)*, из которых *[from, last)* - это префикс, который приведет к полному совпадению. Этот флаг используется при сопоставлении неполных или очень длинных текстов; дополнительную информацию см. документацию по частичным сравнениям. |
| `Extra` | Дает указание механизму поиска сохранять всю доступную информацию о наличии совпадений; если совпадение повторяется снова, то информация о каждом из них будет доступна через методы `match_results::captures()` или `sub_match_captures()`. |

| Флаг | Описание |
|---|---|
| `SingleLine` | Эквивалентно обратному модификатору "m/" языка Perl; предотвращает поиск ^ после встроенного символа новой строки (для того, чтобы он совпадал только в начале исходного текста) и $ от поиска перед встроенным символом новой строки (чтобы он совпадал только в конце исходного текста). |
| `PrevAvail` | Указывает, что `--first` является допустимой позицией итератора, когда этот флаг установлен, тогда флаги `match_not_bol` и `match_not_bow` игнорируются алгоритмами регулярных выражений (RE.7) и итераторов (RE.8). |
| `DotNewLine` | Указывает, что выражение "." не распознается как символ новой строки. Это инверсия модификатора "s/" языка Perl. |
| `NotDotNull` | Указывает, что выражение "." не распознается как символ null ('\0'). |
| `Posix` | Указывает, что выражение должно быть обработано в соответствии с правилом POSIX *"leftmost-longest"*, независимо от того, какое выражение было скомпилировано. Эти правила плохо работают со многими специфичными для Perl моментами, например, такими как ленивые (*"non-greedy"*) повторы. |
| `NoSubs` | Заставляет выражение вести себя так, как будто оно не имеет найденных подмножеств, независимо от того, сколько их присутствует на самом деле. Класс `Matches` будет содержать только информацию об общем совпадении, а не о совпадении в подмножествах. |

### 11.3 Функция Re.replace

Находит в заданной строке все фрагменты, удовлетворяющие регулярному выражению. Каждый найденный фрагмент форматируется в соответствии с форматтером и заменяет собой исходный текст.

**Вызов**

```
Re.replace(subject, formatter, matchFlags, pattern)
```

**Параметры**

- `subject` (`string`) – исходная строка для поиска;
- `formatter` (`string`) – строка, задающая форматирование найденных фрагментов;
- `matchFlags` (`int`) – флаги, задающие правила применения регулярного выражения, а также флаги, специфичные для замены;
- `pattern` (`string`, `Regex`) – строка шаблона или скомпилированный шаблон.

**Возвращает**

- `newString` (`string`) – новая строка с замененными подстроками;
- `err` (`string`) - сообщение об ошибке или nil.

### 11.4 Функция Re.search

Ищет скомпилированное регулярное выражение по заданной строке. Метод возвращает найденные подстроки.

**Вызов**

```
Re.search(subject, matchFlags, pattern)
```

**Параметры**

- `subject` (`string`) – исходная строка;
- `matchFlags` (`int`) – флаги, задающие правила применения регулярного выражения;
- `pattern` (`string`, `Regex`) – строка шаблона или скомпилированный шаблон.

**Возвращает**

- `matches` (`object`) – подстроки, найденные в соответствии с шаблоном;
- `err` (`string`) - сообщение об ошибке или `nil`.

### 11.5 Флаги, используемые для замены

Эти флаги определены в пространстве имен `Re.Replace`. Они используются в алгоритме, используемом методом `Re.replace()` и находятся в таблице 138

Таблица 138 – Описание флагов для функции `Re.replace()`

| Флаг | Описание |
|---|---|
| `FormatDefault` | Когда к исходному тексту применяется регулярное выражение, и находится очередной фрагмент для замены, новая строка формируется с использованием функции замены *ECMAScript*, ECMA-262, Спецификация языка ECMAScript, глава 15, часть 5.4.11 String.prototype.replace.<br>Эта функциональность идентична описанной в руководстве Perl Format String Syntax.<br>После того, как все неперекрывающиеся вхождения регулярного выражения найдены и заменены, фрагменты исходного текста, не соответствующие регулярному выражению, копируются в результирующую строку без изменений. |
| `FormatSed` | В случае, когда регулярное выражение применяется для замены в строке, новая строка формируется с использованием правил, описанных в стандарте IEEE Std 1003.1-2001, Portable Operating SystemInterface (POSIX ), Shells and Utilities.<br>См. также Sed Format String Syntax. |
| `FormatPerl` | В случае, когда регулярное выражение применяется для замены в строке, новая строка формируется по правилам Perl 5. |

| Флаг | Описание |
|---|---|
| `FormatLiteral` | В случае, когда регулярное выражение применяется для замены в строке, новая строка будет являться строковой копией заменяемого текста. |
| `FormatNoCopy` | В случае, когда регулярное выражение применяется для замены в строке, фрагменты, не соответствующие регулярному выражению, не будут копироваться в результирующую строку. |
| `FormatFirstOnly` | Когда данный флаг установлен при операции поиска или замены, то заменяется только первое вхождение регулярного выражения. |
| `FormatAll` | Все синтаксические расширения включены, включая условные замены (*?ddexpression1:expression2*). Для дополнительных деталей см. Руководство по форматированию строк Boost. |

## 12 ФУНКЦИИ ДЛЯ РАБОТЫ С ДАТОЙ И ВРЕМЕНЕМ

В данном разделе описаны функции, предназначенные для работы с датой и временем: os.clock(), os,date(), os.difftime(), os.time().

Примеры задач, которые позволяют решать данные функции:

### Преобразование DATETIME в POSIX

```
datetime = {year = 2013, month = 09, day = 13, hour = 21, min = 40, sec = 15}
seconds_since_epoch = os.time(datetime)
print(tostring(seconds_since_epoch))
```

### Преобразование POSIX в DATETIME

```
seconds_since_epoch  = 1379094015
datetime = os.date("!*t",seconds_since_epoch)
print( "year = "           .. tostring(datetime.year) .. " " ..
       "month = "          .. tostring(datetime.month) .. " " ..
       "day = "            .. tostring(datetime.day) .. " " ..
       "hour = "           .. tostring(datetime.hour) .. " " ..
       "min = "            .. tostring(datetime.min) .. " " ..
       "sec = "            .. tostring(datetime.sec) .. " " ..
       "weekday = "        .. tostring(datetime.wday) .. " " ..
       "day of year = "    .. tostring(datetime.yday) .. " " ..
       "iddst = "          .. tostring(datetime.isdst))
```

### Текущее время в формате POSIX

```
print(os.time())
```

### Текущее время в формате DATETIME

```
datetime = os.date("!*t",os.time())
print(tostring(datetime.year) .. " " ..
      tostring(datetime.month) .. " " ..
      tostring(datetime.day) .. " " ..
      tostring(datetime.hour) .. " " ..
      tostring(datetime.min) .. " " ..
      tostring(datetime.sec) .. " " ..
      tostring(datetime.wday) .. " " ..
      tostring(datetime.yday) .. " " ..
      tostring(datetime.isdst))
```

**Текущий месяц**

```
print(os.date("%B", os.time()))
```

**День недели по дате**

```
print(os.date("%w",os.time({ year=1997, month = 11, day = 10})))
```

**Порядковый номер недели в году по дате**

```
print(os.date("%W",os.time({ year=1997, month = 11, day = 10})))
```

**Вывод даты / времени в различных форматах**

```
print(os.date("%d.%m.%Y"))
print(os.date("%X",os.time()))
print(os.date("Сейчас %H часов %M минут %S секунд",os.time()))
print(os.date())
```

### 12.1 Функция os.clock

Функция `os.clock` возвращает время от начала запуска приложения / процесса. Время возвращается в секундах с точностью до миллисекунд.

Типичное применение - измерение времени выполнения фрагмента кода.

**Пример измерения времени выполнения фрагмента кода**

```
local x = os.clock()
local s = 0
for i=1,100000 do s = s + i end
print(string.format("elapsed time: %.2f\n", os.clock() - x))
```

### 12.2 Функция os.date

Функция `os.date(format, time)` возвращает форматированную дату / время. В качестве первого аргумента выступает формат, вторым аргументом является время в секундах. Оба аргумента не обязательны. При отсутствии второго аргумента будет использован заданный формат и текущая дата / время. Вызов без аргументов вернет текущую дату / время в формате 07.05.2024 13:33:10.

В строке формата могут быть использованы следующие опции:

%a - день недели, сокр. (англ.) (пример, Wed)

%A - день недели, полностью (англ.) (пример, Wednesday)

%b - месяц, сокр. (англ.) (пример, Sep)

%B - месяц, полностью (англ.) (пример, September)

%c - дата и время (по-умолчанию) (пример, 03/22/15 22:28:11)

%d - день месяца (пример, 22) [диапазон, 01-31]

%H - час, в 24-х часовом формате (пример, 23) [диапазон, 00-23]

%I - час, в 12-и часовом формате (пример, 11) [диапазон, 01-12]

%M - минута (пример, 48) [диапазон, 00-59]

%m - месяц (пример, 09) [диапазон, 01-12]

%p - время суток "am", или "pm"

%S - секунда (пример, 10) [диапазон, 00-59]

%w - день недели (пример, 3) [диапазон, 0-6, соответствует Sunday-Saturday]

%x - дата (пример, 09/16/98)

%X - время (пример, 23:48:10)

%Y - год, 4 цифры (пример, 2015)

%y - год, 2 цифры (пример, 15) [00-99]

%% - символ "%"

*t - вернет таблицу

!*t - вернет таблицу (по Гринвичу)

Если параметр `format` начинается с '!', то время форматируется в соответствии с универсальным глобальным временем (по Гринвичу). После этого опционального символа, если format равен "*t", то date возвращает таблицу со следующими полями: `year` (год, четыре цифры), `month` (месяц, 1 – 12), day (день, 1 – 31), `hour` (час, 0 – 23), `min` (минуты, 0 – 59), `sec` (секунды, 0 – 61), `wday` (день недели, воскресенью соответствует 1), `yday` (день года), и `isdst` (флаг дневного времени суток, тип boolean).

**Примеры**

```
print (os.date("%x")) --> 07.05.2024
print (os.date("%c")) --> 25/04/07 10:10:05
```

### 12.3 Функция os.difftime

Функция `os.diftime(t1, t2)` возвращает число секунд, прошедших от времени t1 до времени t2.

**Пример сравнения двух дат в днях**

```
reference = os.time{day=15, year=2024, month=2}
daysfrom = os.difftime(os.time(), reference) / (24 * 60 * 60) -- seconds in a day
wholedays = math.floor(daysfrom)
print(wholedays)
```

### 12.4 Функция os.time

Функция `os.time()` возвращает время в формате posix (количество секунд, прошедших с 00:00:00 1 января 1970 года).

При вызове без аргументов возвращает текущее время.

Аргументом может являться таблица с обязательными ключами `year`, `month`, `day`, и необязательными `hour`, `min`, `sec`, `isdst`.

**Пример**

```
print (os.time()) -- текущее время в формате posix

local datetime = {year = 2017, month = 03, day = 1, hour = 14, min = 23, sec = 8}
print (os.time(datetime))  -- время, переданное в параметре
```

## 13 КЛАСС MATCHES

Класс `Matches` содержит результат функций `Re.match()` и `Re.search()`.

### 13.1 Метод getFirst

**Вызов**

```
position, err = matches:getFirst(group)
```

**Параметры**

- `group (int, string)` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

**Возвращает**:

- `position (int)` – первая позиция (в байтах) исходной строки;
- `err (string)` - сообщение об ошибке или `nil`.

### 13.2 Метод getLength

**Вызов**

```
position, err = matches:getLength(group)
```

**Параметры**

- `group (int, string)` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

**Возвращает**:

- `length (int)` – длина исходной строки в байтах;
- `err (string)` - сообщение об ошибке или `nil`.

### 13.3 Метод getSize

**Вызов**

```
size, err = matches:getSize()
```

**Возвращает**:

- `size (int)` – количество найденных групп;
- `err (string)` - сообщение об ошибке или `nil`.

### 13.4 Метод getString

**Вызов**

```
substr, err = matches:getString(group, subject)
```

**Параметры**

- `group` (`int`, `string`) – позиция (или имя группы) найденных результатов, начинающаяся с 1;
- `subject` (`string`) – исходная строка. **Внимание**: объект `Matches` сохраняет только смещения и не хранит исходную строку. Таким образом, необходимо передать ту же строку, которая использовалась для поиска.

**Возвращает**:

- `substr` (`string`) – найденная подстрока;
- `err` (`string`) - сообщение об ошибке или `nil`.

### 13.5 Метод _tostring

Стандартная метафункция.

```
string = matches:__tostring()
```

**Пример**

```
local str = "-Номер:1234"
local regex = Re.create("-(\\w+):(\\d{4})")

local matches, err = Re.match(str, Re.Match.Default, regex)
print(tostring(regex), tostring(matches))

local number = matches:getString(3, str)
print(number)
```