ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)» MYOFFICE DOCUMENT APPLICATION PROGRAMMING INTERFACE (API). БИБЛИОТЕКА ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

РУКОВОДСТВО ПРОГРАММИСТА

3.2

Версия 1

На 290 листах

Дата публикации: 17.12.2024

Москва
2024

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.
MyOffice Document API	Программное обеспечение «МойОфис Комплект Средств Разработки (SDK). MyOffice Document API. Библиотека для языка программирования Python».
API	Application Programming Interface (программный интерфейс приложения).
SDK	Software Development Kit (комплект для разработки программного обеспечения).

1 Общие сведения

1.1 Назначение

Библиотека MyOffice Document API для языка программирования Python используется в составе прикладных информационных систем или отдельных приложений под управлением ОС Microsoft Windows или Linux. Библиотека предназначена для решения задач по созданию и наполнению текстовых и табличных документов в пакетном режиме.

1.2 Библиотека MyOffice Document API для языка программирования Python

Библиотека MyOffice Document API для языка программирования Python предоставляет возможность выполнения следующих операций:

1. Создание, открытие, сохранение изменений в электронных текстовых и табличных документах в следующих форматах:
   - текстовые и табличные документы, создаваемые с помощью Microsoft Office в формате OOXML, расширения файлов DOCX и XLSX;
   - текстовые и табличные документы, создаваемые с помощью LibreOffice в формате ODF, расширения файлов ODT и ODS;
   - текстовые и табличные документы, создаваемые с помощью МойОфис в формате ODF, расширения файлов XODT и XODS;
   - экспорт документов в формате PDF.
2. Изменение содержимого документов в пакетном режиме, в том числе:
   - добавление, удаление, изменение текста абзаца;
   - вставка, удаление, форматирование таблиц в текстовом документе;
   - вставка, удаление, переименование отдельных листов в табличном документе;
   - установка значения ячейки электронной таблицы и расчет формул;
   - оформление документа с использованием различных шрифтов и цветового оформления.
3. Поиск и замена фрагмента текста в документе.
4. Управление режимом рецензирования документа, отслеживание изменений в документе.
5. Управление закладками в текстовом документе.
6. Написание и запуск макрокоманд.

Для управления содержимым документа используется объектная модель, представляющая собой совокупность структур данных текстового или табличного документа.

1.3 Уровень подготовки пользователя

Пользователь MyOffice Document API должен иметь:

1. Опыт разработки на языке Python для ОС Microsoft Windows или Linux. Полный список поддерживаемых ОС приведен в документе «МойОфис Комплект Средств Разработки (SDK). MyOffice Document Application Programming Interface (API). Системные требования».
2. Навык работы со стандартными офисными приложениями.

1.4 Системные требования

Полный перечень требований к программному и аппаратному обеспечению приведен в документе «МойОфис Комплект Средств Разработки (SDK). MyOffice Document Application Programming Interface (API). Системные требования».

2 Подготовка к работе

2.1 Список дистрибутивов

Дистрибутив MyOffice Document API поставляется в виде файлов формата **whl** (см. таблицу 2).

Таблица 2 - Список дистрибутивов `MyOffice Document API`

ОС	Дистрибутив
Microsoft Windows	MyOfficeSDKDocumentAPI-3.2-cp38-abi3-win_amd64.whl
Linux	MyOfficeSDKDocumentAPI-3.2-cp38-abi3-linux_x86_64.whl

2.2 Установка в ОС Microsoft Windows

Для установки MyOffice Document API в ОС Microsoft Windows необходимо разместить файл дистрибутива в локальной папке и осуществить следующие действия:

1. Открыть окно командной строки ОС Microsoft Windows.
2. Перейти в локальную папку с файлом дистрибутива.
3. Установить программный пакет MyOffice Document API с помощью системы управления пакетами, например:
   pip install MyOfficeSDKDocumentAPI-3.2-cp38-abi3-win_amd64.whl.

Внимание! Следует убедиться в актуальности установленной версии Python. Для использования MyOffice Document API 3.2 необходима версия Python 3.8.6.

2.3 Установка в ОС Linux

Для установки MyOffice Document API в ОС Linux необходимо разместить файл дистрибутива в локальной папке и осуществить следующие действия:

1. Открыть окно терминала ОС Linux.
2. Перейти в локальную папку с файлом дистрибутива.
3. Установить программный пакет MyOffice Document API с помощью системы управления пакетами, например:
   pip install MyOfficeSDKDocumentAPI-3.2-cp38-abi3-linux_x86_64.whl.

Внимание! Следует убедиться в актуальности установленной версии Python. Для использования MyOffice Document API 3.2 необходима версия Python 3.8.6.

2.4 Проверка работоспособности

Для проверки работоспособности MyOffice Document API необходимо выполнить тестовый пример.

Тестовый пример использует вызовы MyOffice Document API для создания текстового документа в формате DOCX.

```
from MyOfficeSDKDocumentAPI import DocumentAPI as myOfficeSDK

application = myOfficeSDK.Application()
document = application.createDocument(myOfficeSDK.DocumentType_Text)
document.getRange().getBegin().insertText("Hello! This is an example!")
document.saveAs("BasicExample.docx")
```

Сохраните код в файле **basic-app.py** и выполните команду:

```
python basic-app.py
```

В результате работы программы в текущем каталоге создается файл **BasicExample.docx**, содержащий текст «Hello! This is an example!».

MyOffice Document API считается работоспособным, если приложение выполнено успешно.

Здесь и далее вместо `DocumentAPI` использован алиас `myOfficeSDK`.

2.5 Распространение разработанных приложений

Распространение разработанного приложения осуществляется посредством передачи файла, содержащего исходный код приложения.

Для запуска разработанного приложения на компьютере пользователя должны присутствовать:

– интерпретатор Python, версии 3.8.6;

- установленный пакет MyOffice Document API для языка программирования Python.

3 Объектная модель МойОфис SDK

МойОфис SDK предоставляет разработчику возможности для управления содержимым текстового и табличного документа.

Библиотека позволяет работать с пользовательскими документами различных форматов, однако внутренняя модель документа представлена в формате ODF (Open Document Format, открытый формат документов для офисных приложений), который принят в качестве ГОСТ (Р ИСО/МЭК 26300-2010). Описание внутреннего формата ODF размещено на ресурсе сообщества OASIS (*Organization for the Advancement of Structured Information Standards*).

В данном документе описана объектная модель API (классы, коллекции, методы доступа) для доступа к компонентам внутренней модели документа.

Основной модуль `DocumentAPI` содержит класс `Application`, который используется для создания и открытия документа. Помимо этого, `DocumentAPI` содержит классы и функции для представления документа и всех его составляющих, которые поддерживает МойОфис: абзацы, таблицы, ячейки, рисунки, колонтитулы и т.д.

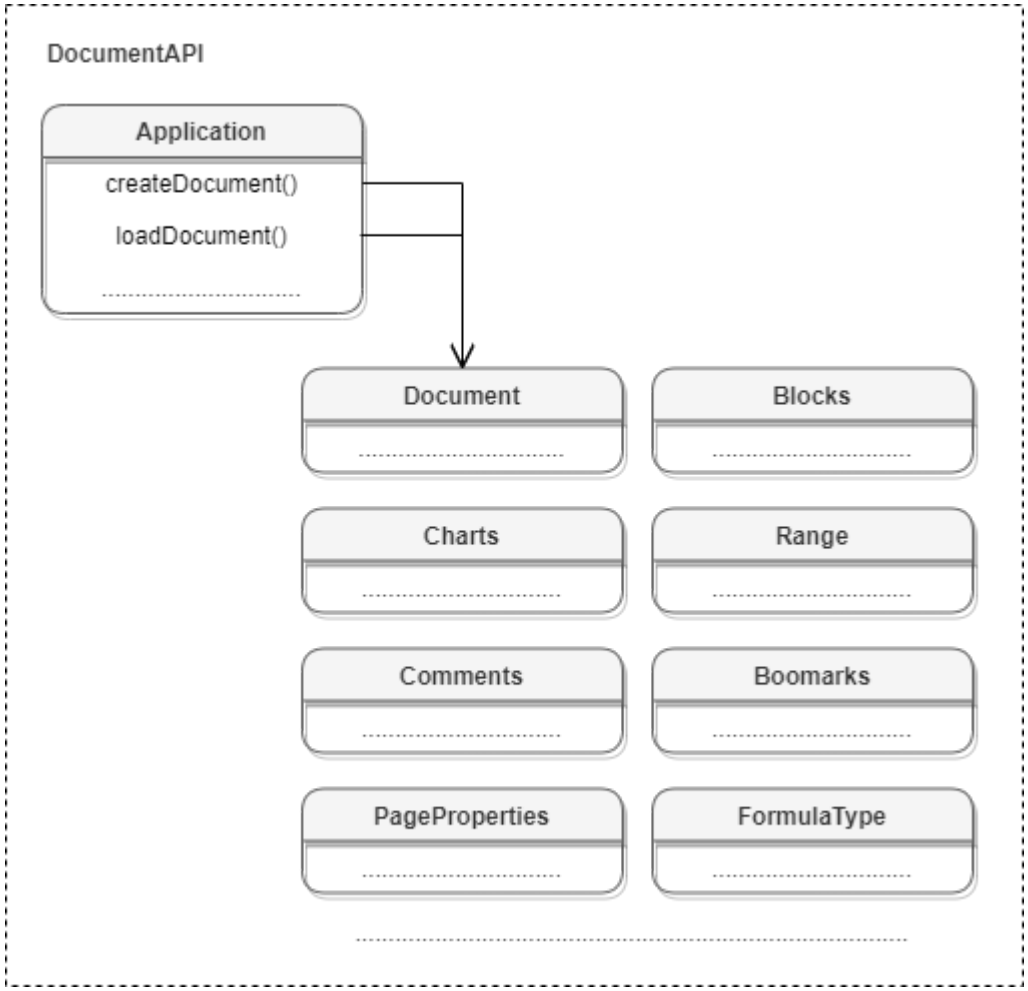


Рисунок 1 – Объектная модель МойОфис SDK.

4 Работа с документами

4.1 Работа с текстовым документом

4.1.1 Создание и открытие текстового документа

Метод Application.createDocument создает документ. В качестве параметра используются DocumentType или DocumentSettings.

Примеры создания текстового документа:

```
document = application.createDocument(myOfficeSDK.DocumentType_Text);
```

```
documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Text;
document = application.createDocument(documentSettings)
```

Метод Application.loadDocument загружает документ. В качестве параметра используется путь к документу. Дополнительно может быть использован параметр LoadDocumentSettings.

Примеры загрузки текстового документа:

```
document = application.loadDocument("test.docx")
```

```
documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Text
loadSettings = myOfficeSDK.LoadDocumentSettings()
loadSettings.commonDocumentSettings = documentSettings
document = application.loadDocument("test.docx", loadSettings)
```

4.1.2 Сохранение и экспорт текстового документа

Метод Document.saveAs сохраняет документ по указанному пути.

Примеры сохранения текстового документа:

```
document.saveAs(filePath)
```

```
saveDocumentSettings.documentFormat = myOfficeSDK.DocumentFormat_OXML
saveDocumentSettings.documentType = myOfficeSDK.DocumentType_Text
saveDocumentSettings.documentPassword = "password"
saveDocumentSettings.isTemplate = False

saveDocumentSettings.dsvSettings = myOfficeSDK.DSVSettings()
saveDocumentSettings.dsvSettings.autofit = True
saveDocumentSettings.dsvSettings.startBlockIndex = 0
```

```
saveDocumentSettings.dsvSettings.lastBlockIndex = 10

document.saveAs(filePath, saveDocumentSettings)
```

Метод `Document.exportAs` экспортирует документ в файл по указанному пути с заданным форматом типа `ExportFormat`.

В настоящее время поддерживается только операция экспорта документа в формат `PDF`.

Примеры экспорта текстового документа:

```
document.exportAs(filePath, myOfficeSDK_ExportFormat.PDFA1)
```

```
textExportSettings = myOfficeSDK.TextExportSettings()
textExportSettings.pageNumbers =
myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1, textExportSettings)
```

4.1.3 Разделы (секции) документа

На рисунке 2 изображена объектная модель классов, относящихся к работе с секциями текстового документа.

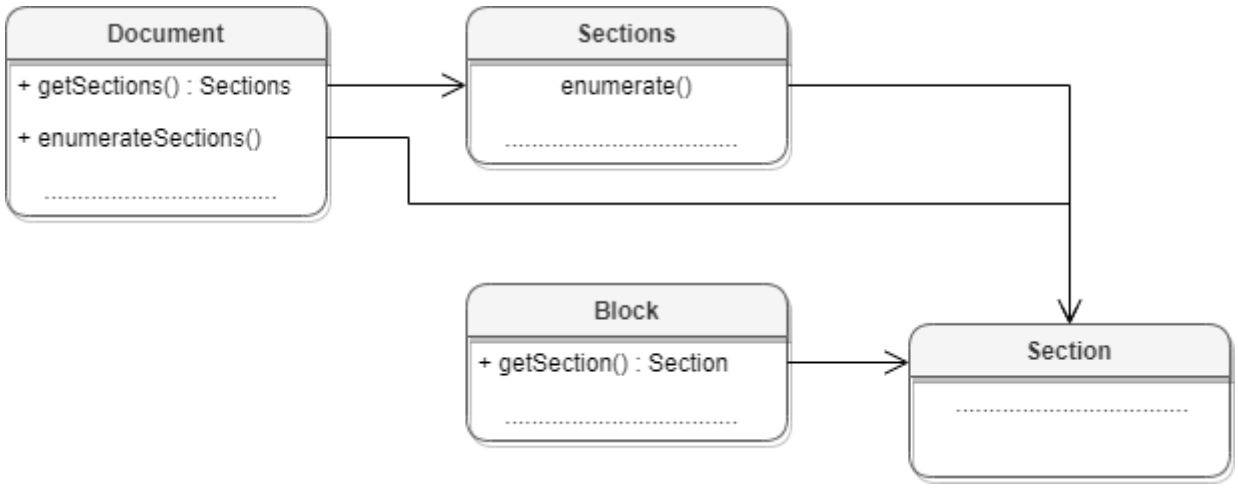


Рисунок 2 – Объектная модель классов для работы с секциями

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 объекта Sections с помощью вызова Document.getSections();
– перечисление всех доступных секций Section с помощью вызова Document.getSectionsEnumerator();
– получение секции Section вызовом метода Block.getSection() для блока, который входит в секцию.

Примеры:

```
sections = document.getSections()
sectionsEnumerator = sections.getEnumerator()
for section in sectionsEnumerator:
    print(section.getPageProperties().width)
```

```
sectionsEnumerator = document.getSectionsEnumerator()
for section in sectionsEnumerator:
    print(section.getPageProperties().width)
```

```
block = document.getBlocks().getBlock(0)
section = block.getSection()
if section != None:
    print(section.getPageProperties().width)
```

4.1.4 Встроенные объекты в текстовом документе

Редакторы текста МойОфис поддерживают несколько типов графических объектов со схожим поведением: изображения (Image) и фигуры (Shape), которые являются разновидностью фигур.

Объектная модель текстового документа в части управления встроенными объектами развивается и дополняется возможностями. На данный момент доступны следующие операции:

– вставка изображений в текстовый документ;
– перечисление графических объектов, находящихся в текстовом документе, определение их типа и геометрических размеров;
– перемещение графических объектов текстового документа, изменение их размеров.

Доступ ко встроенным объектам текстового документа осуществляется посредством использования методов Range.getInlineObjects(), Table.getImages(), Table.getMediaObjects().

4.1.4.1 Вставка изображения

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

Вставка изображения в текстовый документ

```
range = document.getRange()
insertedImage = range.getBegin().insertImage("C://Tmp//123.jpg",
myOfficeSDK.SizeU(100, 100))
```

4.1.4.2 Перечисление встроенных объектов

Перечисление графических объектов в текстовом документе.

```
docRange = document.getRange()
mediaObjects = docRange.getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    print(mediaObject.getFrame().getWrapType())
```

Перечисление изображений в текстовом документе.

```
images = document.getRange().getImages()
imagesEnumerator = images.getEnumerator()
for image in imagesEnumerator:
    print(image.getFrame().getWrapType())
```

Перечисление изображений в таблице текстового документа.

```
blocks = document.getBlocks()
table = table = blocks.getTable(0)
images = table.getImages()
imagesEnumerator = images.getEnumerator()
for image in imagesEnumerator:
    print(image.getFrame().getWrapType())
```

4.1.5 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены на страницах документа.

Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 3). В табличном документе таблицами являются листы документа.

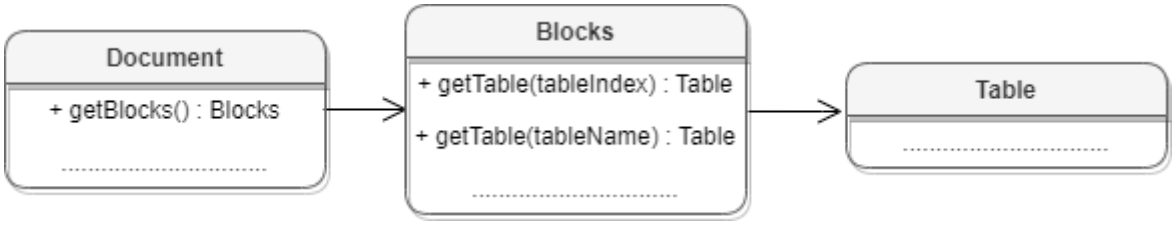


Рисунок 3 – Объектная модель для работы с таблицами

Получение таблицы текстового документа:

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

```
table = blocks.getTable(0)
```

```
table = blocks.getTable("Таблица1")
```

Перечисление таблиц текстового документа:

Для перечисления таблиц текстового документа можно использовать метод `Blocks.getTablesEnumerator()`.

```
blocks = document.getBlocks()
tablesEnumerator = blocks.getTablesEnumerator()
for tableIndex, table in enumerate(tablesEnumerator):
    print(table.getRange().extractText())
```

Вставка таблицы в текстовый документ:

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

```
range = document.getRange()
endPosition = range.getEnd()
table = endPosition.insertTable(3, 3, "Table")
```

Переименование таблицы:

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

```
table = document.getBlocks().getTable("List11")
table.setName("Table1")
print(table.getName()) # Table1
```

Удаление таблицы:

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

```
table = document.getBlocks().getTable(0)
table.remove()
```

4.1.6 Работа с закладками

Основным классом для работы с закладками является `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

```
startDocument = document.getRange().getBegin()
startDocument.insertBookmark("Bookmark")
```

Удаление закладки с заданным именем

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

Поиск закладки по имени

```
bookmarks = document.getBookmarks()
bookmarkRange = bookmarks.getBookmarkRange("Bookmark")
```

Замена текстового содержимого закладки

```
bookmarks = document.getBookmarks()
bookmarkRange = bookmarks.getBookmarkRange("Bookmark")
```

```
if bookmarkRange != None:
    bookmarkRange.replaceText("New bookmark text")
```

Вставка текста в закладку

```
bookmarks = document.getBookmarks()
bookmarkRange = bookmarks.getBookmarkRange("Bookmark")
if bookmarkRange != None:
    bookmarkRange.getBegin().replaceText("New bookmark text")
```

Удаление содержимого закладки

```
bookmarks = document.getBookmarks()
bookmarkRange = bookmarks.getBookmarkRange("Bookmark")
if bookmarkRange != None:
    bookmarkRange.getBegin().removeBackward()
```

Получение текстового содержимого закладки

```
bookmarks = document.getBookmarks()
bookmarkRange = bookmarks.getBookmarkRange("Bookmark")
if bookmarkRange != None:
    print("Bookmark range text:", bookmarkRange.extractText())
```

Вставка таблицы в закладку

```
bookmarks = document.getBookmarks()
bookmarkRange = bookmarks.getBookmarkRange("Bookmark")
if bookmarkRange != None:
    bookmarkRange.getEnd().insertTable(3, 3, "signers_list")
```

4.1.7 Рецензирование документов

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

- помечать изменения, вносимые пользователем в текстовый документ (TrackedChange);
- ассоциировать текстовый комментарий с фрагментом текстового документа (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 классе `Range` (см. Рисунок 4).

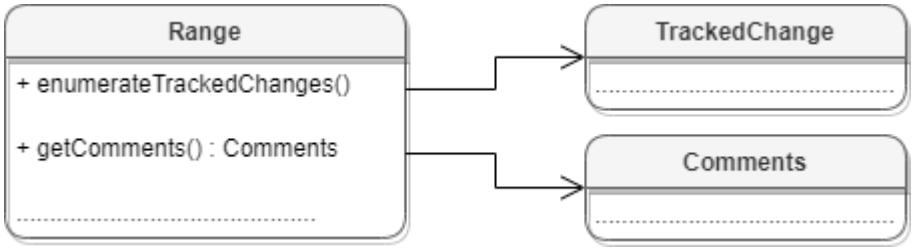


Рисунок 4 – Инструменты рецензирования документа

4.1.8 Работа с элементами управления

К элементам управления относятся следующие объекты: флажок (`CheckBoxControl`), текстовое поле (`InputFieldControl`), поле с календарём (`DatePickerControl`) и поле с выпадающим списком (`DropListControl`). Они могут быть расположены в текстовом документе или его шаблоне.

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

```
controls = document.getContentControls()
```

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

```
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()`:

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

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

```
inputField.setValue(inputField.getValue().replace(' ', '_'))
```

Поле с выпадающим списком дополнительно содержит метод `DropListControl.getChoices()`, который возвращает элементы выпадающего списка.

```
comboBox = controls.findByTitle("select").toDropList()
comboBox.setValue(comboBox.getChoices().index("two"))
```

4.2 Работа с табличным документом

4.2.1 Создание и открытие табличного документа

Метод `Application.createDocument` создает документ. В качестве параметра используется тип `DocumentType`. Для создания табличного документа необходимо выбрать тип `DocumentType.Workbook`.

Пример создания табличного документа:

```
document = application.createDocument(myOfficeSDK.DocumentType_Worksheet);
```

```
documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Worksheet;
document = application.createDocument(documentSettings)
```

Метод `Application.loadDocument` открывает документ, находящийся по указанному пути.

Примеры загрузки табличного документа:

```
document = application.loadDocument("spreadsheet.xlsx")
```

```
documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Worksheet
```

```
loadSettings = myOfficeSDK.LoadDocumentSettings()
loadSettings.commonDocumentSettings = documentSettings
document = application.loadDocument("spreadsheet.xlsx", loadSettings)
```

4.2.2 Сохранение и экспорт табличного документа

Метод `Document.saveAs` сохраняет документ по указанному пути.

Примеры сохранения табличного документа:

```
document.saveAs(filePath)
```

```
saveDocumentSettings.documentFormat = myOfficeSDK.DocumentFormat_OXML
saveDocumentSettings.documentType = myOfficeSDK.DocumentType_Worksheet
saveDocumentSettings.documentPassword = "password"
saveDocumentSettings.isTemplate = False

saveDocumentSettings.dsvSettings = myOfficeSDK.DSVSettings()
saveDocumentSettings.dsvSettings.autofit = True
saveDocumentSettings.dsvSettings.startBlockIndex = 0
saveDocumentSettings.dsvSettings.lastBlockIndex = 10

document.saveAs(filePath, saveDocumentSettings)
```

Метод `Document.exportAs` экспортирует документ в файл по указанному пути с заданным форматом типа `ExportFormat`.

В настоящее время поддерживается только операция экспорта документа в формат `PDF`.

Примеры экспорта табличного документа:

```
document.exportAs(filePath, myOfficeSDK_ExportFormat.PDFA1)
```

```
textExportSettings = myOfficeSDK.WorkbookExportSettings()
textExportSettings.pageNumbers =
myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1, textExportSettings)
```

4.2.3 Диаграммы

Работа с диаграммами реализована только в табличных документах. На рисунке 5 изображена объектная модель классов, относящихся к работе с диаграммами.

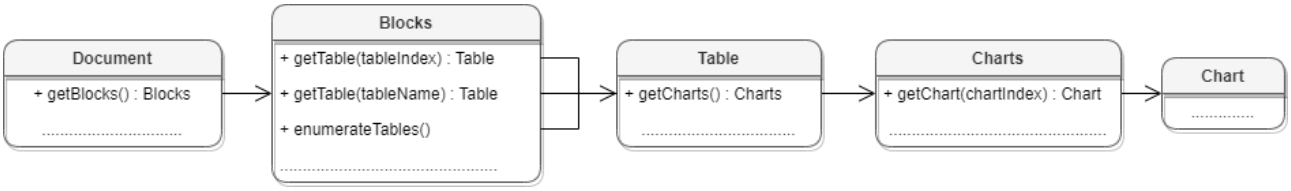


Рисунок 5 – Объектная модель классов для работы с диаграммами

Доступ к списку диаграмм производится через класс `Table`, соответствующий листу табличного документа.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
charts = firstSheet.getCharts()
print(charts.getChartsCount())
```

Для получения диаграммы `Chart` используется метод `Charts.getChart()`.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
charts = firstSheet.getCharts()
chart = charts.getChart(0)
print(chart.getTitle())
```

Создание и удаление диаграмм в текущей версии не поддерживаются.

4.2.4 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 ячейки диапазона `"A1:B2"` в позицию диапазона `"E6:F7"`:

```
table = document.getBlocks().getTable(0)

sourceRange = table.getCellRange("A1:A2")
destRange = table.getCellRange("A2:A3")
sourceRange.moveInto(destRange)
```

```
leftTopCellPositoin = myOfficeSDK.CellPosition(0, 0)
rightBottomCellPositoin = myOfficeSDK.CellPosition(1, 1)
srcCellRangePosition = myOfficeSDK.CellRangePosition(leftTopCellPositoin,
rightBottomCellPositoin)

strTargetRange = "E6:F7"
sourceRange = table.getCellRange(srcCellRangePosition)
destRange = table.getCellRange(strTargetRange)

sourceRange.copyInto(destRange)
```

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

```
sourceRange.moveInto(destRange)
```

Методы `CellRange.copyInto()` и `CellRange.moveInto()` также позволяют копировать и перемещать ячейки между документами и листами документа:

```
document1 = application.loadDocument("sheet1.xods")
document2 = application.loadDocument("sheet2.xods")

sheetList1 = document1.getBlocks().getTable(0)
sheetList2 = document2.getBlocks().getTable(1)

sourceRange = sheetList1.getCellRange("A1:C3")
targetRange = sheetList2.getCellRange("A1:C3")

sourceRange.copyInto(targetRange)
```

4.2.5 Работа с формулами

Метод `Cell.setFormula` позволяет поместить формулу в ячейку таблицы:

```
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` и `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() == myOfficeSDK.CalculationMode_Auto:
    document.setCalculationMode(myOfficeSDK.CalculationMode_Manual)
```

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():
    document.setCalculatedOnSave(False)
```

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()
```

4.2.6 Встроенные объекты в табличном документе

Редакторы таблиц МойОфис поддерживают графические объекты типа `Image`, `Chart`, `Shape`.

Объектная модель табличного документа в части управления изображениями развивается и дополняется возможностями. На данный момент доступны следующие операции:

– перечисление изображений, находящихся в текстовом документе, определение их типа и геометрических размеров\;
– перемещение изображений табличного документа, изменение их размеров и масштаба.

Доступ ко встрроенным объектам табличного документа осуществляется посредством использования методов `Range.getInlineObjects()`, `Table.getImages()`, `Table.getMediaObjects()`.

4.2.6.1 Вставка изображения

Вставка изображений в табличный документ на данный момент не поддерживается.

4.2.6.2 Перечисление встроенных объектов

Список изображений в табличном документе может быть получен с помощью метода `Table.getImages()`, вызванного у объекта листа документа.

Перечисление изображений табличного документа:

```
table = document.getBlocks().getTable(0);

images = table.getImages();
imagesEnumerator = images.getEnumerator();
for image in imagesEnumerator:
    absoluteFrame = image.getFrame();
    if absoluteFrame:
        position = absoluteFrame.getTopLeft();
        print(position);
```

Список всех встроенных объектов в листе табличного документа может быть получен с помощью метода `Table.getMediaObjects()`, вызванного у объекта листа документа

Перечисление встроенных объектов табличного документа:

```
table = document.getBlocks().getTable(0)
mediaObjects = table.getMediaObjects()
for mediaObject in mediaObjects:
    print(mediaObject)
```

4.2.7 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 6). В табличном документе таблицами являются листы документа.

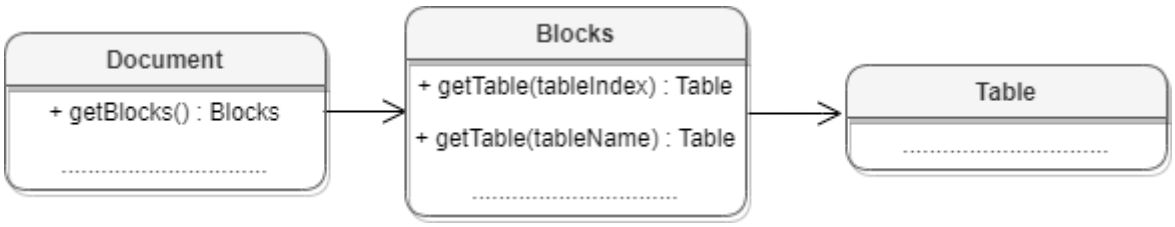


Рисунок 6 – Объектная модель для работы с таблицами

Получение листа табличного документа:

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

```
table = document.getBlocks().getTable(0)
```

```
table = document.getBlocks().getTable("Таблица1")
```

Перечисление страниц табличного документа:

Для перечисления листов табличного документа можно использовать метод `Blocks.getTablesEnumerator()`.

```
tablesEnumerator = document.getBlocks().getTablesEnumerator()
for table in tablesEnumerator:
    print(table.getName())
```

Также доступен вариант перечисления листов документа посредством использования метода `Blocks.getEnumerator()` с дальнейшим преобразованием блока в таблицу.

```
blocksEnumerator = document.getBlocks().getEnumerator()
for block in blocksEnumerator:
    table = block.toTable()
    if table != None:
       print(table.getName())
```

Вставка страницы в табличный документ:

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

```
range = document.getRange()
endPosition = range.getEnd()
table = endPosition.insertTable(3, 3, "Table")
```

Переименование страницы:

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

```
table = document.getBlocks().getTable("List11")
table.setName("Table1")
print(table.getName()) # Table1
```

Скрытие и отображение страниц табличного документа:

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

```
table = document.getBlocks().getTable(0)
table.setVisible(False)
```

Копирование страницы:

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

```
table = document.getBlocks().getTable(0)
table.duplicate()
```

Удаление страницы:

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

```
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 7.

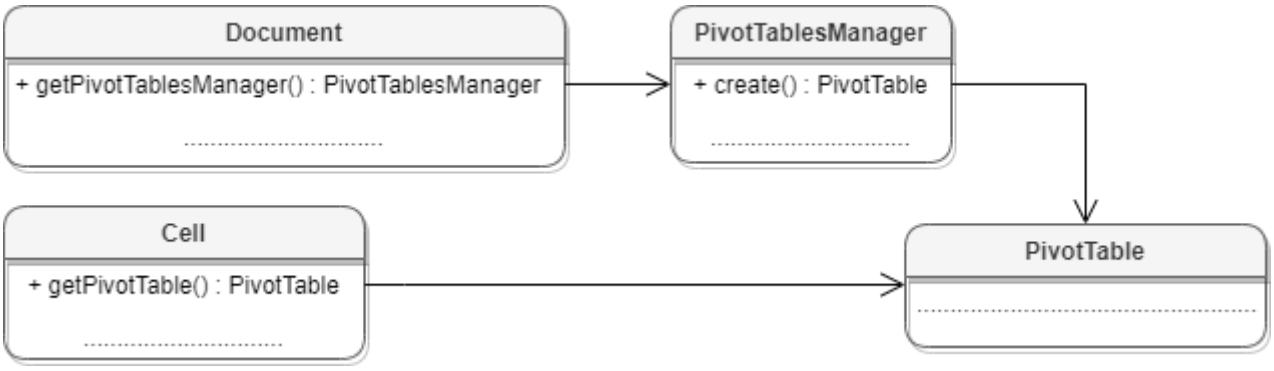


Рисунок 7 – Сводные таблицы

4.2.8.1 Получение сводной таблицы

Для получения диапазона исходных данных сводной таблицы используется метод `Cell.getPivotTable()`.

Пример:

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    print(pivotTable.getSourceRangeAddress())
```

4.2.8.2 Получение диапазона исходных данных сводной таблицы

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

Пример:

```
// Получаем диапазон исходных данных сводной таблицы
sourceRange = pivotTable.getSourceRange()
print(sourceRange.getBeginRow())
print(sourceRange.getBeginColumn())
```

4.2.8.3 Получение диапазона размещения сводной таблицы

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

Пример:

```
pivotTable = pivotTablesManager.create(cellRange)
pivotRange = pivotTable.getPivotRange()
print(pivotRange.getBeginColumn() + ", " + pivotRange.getLastColumn())
```

4.2.8.4 Получение неподдерживаемых свойств сводной таблицы

Для получения неподдерживаемых свойств сводной таблицы используется метод PivotTable.getUnsupportedFeatures().

Пример:

```
unsupportedFeatures = pivotTable.getUnsupportedFeatures()
unsupportedFeaturesEnumerator = unsupportedFeatures.getEnumerator()
for unsupportedFeature in unsupportedFeaturesEnumerator:
    print(unsupportedFeaturesEnumerator)
```

4.2.8.5 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
print(pivotTable.isRowGrandTotalEnabled())
print(pivotTable.isColumnGrandTotalEnabled())
```

4.2.8.6 Получение заголовков сводной таблицы

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

Пример:

```
pivotTableCaptions = pivotTable.getPivotTableCaptions()
print(pivotTableCaptions.errorCaption)
print(pivotTableCaptions.emptyCaption)
print(pivotTableCaptions.grandTotalCaption)
print(pivotTableCaptions.valuesHeaderCaption)
print(pivotTableCaptions.columnHeaderCaption)
print(pivotTableCaptions.rowHeaderCaption)
```

4.2.8.7 Получение и применение фильтра для сводной таблицы

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

Пример:

```
pivotTableFilter = pivotTable.getFilter("Category")
pivotTableFilter.setHidden("Car", True)
pivotTableFilter.setHidden("Technology", True)
pivotTableFilter.setHidden("Furniture", False)
pivotTable.createPivotTableEditor().setFilter(pivotTableFilter).apply()
```

4.2.8.8 Получение полей из области фильтров

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

Пример:

```
pageFields = pivotTable.getPageFields()
pageFieldsEnumerator = pageFields.getEnumerator()
for pivotTableCategory in pageFieldsEnumerator:
    print(pivotTableCategory.fieldProperties.fieldAlias)
    print(pivotTableCategory.fieldProperties.subtotalAlias)
    print(pivotTableCategory.fieldProperties.fieldName)
```

4.2.8.9 Получение полей из области значений

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

Пример:

```
pivotTableValueFields = pivotTable.getValueFields()
pivotTableValueFieldsEnumerator = valueFields.getEnumerator()
for pivotTableValueField in pivotTableValueFieldsEnumerator:
    print(pivotTableValueField.baseFieldName)
    print(pivotTableValueField.cellNumberFormat)
    print(pivotTableValueField.customFormula)
    print(pivotTableValueField.totalFunction)
    print(pivotTableValueField.valueFieldName)
```

4.2.8.10 Получение полей из области строк

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

Пример:

```
rowFields = pivotTable.getRowFields()
rowFieldsEnumerator = rowFields.getEnumerator()
for rowField in rowFieldsEnumerator:
    print(pivotTableCategoryField.fieldProperties.fieldAlias)
    print(pivotTableCategoryField.fieldProperties.subtotalAlias)
    print(pivotTableCategoryField.fieldProperties.fieldName)
    subtotalFunctions = pivotTableCategoryField.subtotalFunctions
    print(subtotalFunctions.Count)
```

4.2.8.11 Получение полей из области колонок

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

Пример:

```
columnFields = pivotTable.getColumnFields()
columnFieldsEnumerator = columnFields.getEnumerator()
for pivotTableCategoryField in columnFieldsEnumerator:
    print(pivotTableCategoryField.fieldProperties.fieldAlias)
    print(pivotTableCategoryField.fieldProperties.subtotalAlias)
    print(pivotTableCategoryField.fieldProperties.fieldName)
    subtotalFunctions = pivotTableCategoryField.subtotalFunctions
    print(subtotalFunctions.Count)
```

4.2.8.12 Получение настроек отображения сводной таблицы

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

Пример:

```
pivotTableLayoutSettings = pivotTable.getPivotTableLayoutSettings()
print(pivotTableLayoutSettings.displayFieldCaptions)
print(pivotTableLayoutSettings.indentForCompactLayout)
print(pivotTableLayoutSettings.isMergeAndCenterLabelsEnabled)
print(pivotTableLayoutSettings.pageFieldOrder)
print(pivotTableLayoutSettings.pageFieldWrapCount)
print(pivotTableLayoutSettings.reportLayout)
print(pivotTableLayoutSettings.useGridDropZones)
print(pivotTableLayoutSettings.valueFieldsOrientation)
```

4.2.8.13 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.
updateResult = pivotTable.update()
if updateResult == myOfficeSDK.PivotTableUpdateResult_FieldAlreadyEnabled:
```

4.2.9 Работа с фильтрами

Работа с фильтрами возможна только в табличном документе. Диаграмма взаимодействия объектов приведена на рисунке 8.

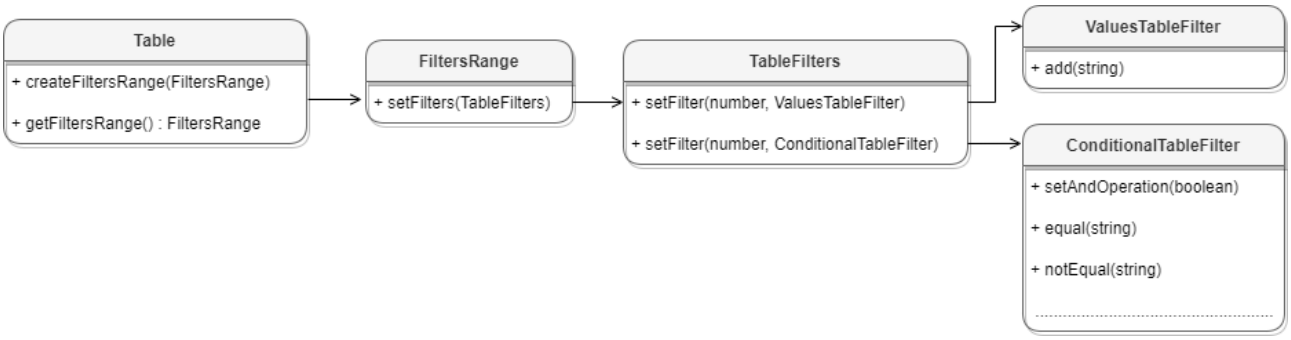


Рисунок 8 – Объектная модель таблиц для работы с фильтрами

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:

```
sheet = document.getBlocks().getTable("Лист1")

cellRange = myOfficeSDK.CellRangePosition(1, 1, 8, 2)
filtersRange = sheet.createFiltersRange(cellRange)
```

```
johnPaulFilter = myOfficeSDK.ValuesTableFilter()
johnPaulFilter.add("John");
johnPaulFilter.add("Paul");

songFilter = myOfficeSDK.ConditionalTableFilter()
songFilter.setAndOperation(True);
songFilter.notEqual("");
songFilter.notBegins("TODO");

tableFilters = myOfficeSDK.TableFilters()
tableFilters.setFilter(0, johnPaulFilter)
tableFilters.setFilter(1, songFilter)

filtersRange.setFilters(tableFilters);
```

4.2.10 Встроенные объекты

4.2.10.1 Определение типа встроенных объектов

Для определения типа графического объекта (`Image`/`Chart`/`Shape`) могут быть использованы методы `MediaObject.toImage()`, `MediaObject.toChart()`. В случае, если объект существует, метод вернет ненулевой объект.

```
for mediaObject in document.getRange().getInlineObjects():
  image = mediaObject.toImage()
  if image != None:
    print("Текущий объект является изображением")
  else:
    chart = mediaObject.toChart()
    if chart != None:
      print("Текущий объект является диаграммой")
    else:
      print("Текущий объект является фигурой")
```

4.2.10.2 Работа со встроенными объектами

Перечисление встроенных объектов описано в разделах Встроенные объекты в текстовом документе и Встроенные объекты в табличном документе.

Остальные методы работы со встроенными объектами общие для текстовых и табличных документов, и зависят от типа `Frame`, в котором находятся:

1. Получение размеров

Размеры встроенного объекта могут быть получены из объектов `InlineFrame` или `AbsoluteFrame`, которые, в свою очередь, могут быть получены посредством использования методов `InlineObject.getFrame()`, `Image.getFrame()`, `Chart.getFrame()` (см раздел `Frame`).

```
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    dimensions = frame.getDimensions()
    print(dimensions.toString())
if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
    dimensions = frame.getDimensions()
    print(dimensions.toString())
```

2. Получение текущей позиции

С помощью методов `InlineFrame.getPosition()`, `AbsoluteFrame.getTopLeft()` можно получить текущую позицию объекта.

```
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    anchoredPosition = frame.getPosition()
    textAnchoredPosition = anchoredPosition.textPosition
    print("horz:", textAnchoredPosition.horizontal, ", vert:",
textAnchoredPosition.vertical)
if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
    topLeftPosition = frame.getTopLeft()
    print("x=", topLeftPosition.x, "y=", topLeftPosition.y)
```

3. Установка размеров

С помощью методов `InlineFrame.setDimensions()`, `AbsoluteFrame.setDimensions()` можно изменить размеры встроенных объектов

```
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    frame.setDimensions(myOfficeSDK.SizeU(50, 50))
if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
    frame.setDimensions(myOfficeSDK.SizeU(50, 50))
```

4. Установка позиции

Для объекта `AbsoluteFrame` используется метод `AbsoluteFrame.moveTo()`

```
newFramePosition = myOfficeSDK.PointU(20, 20)
if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
    frame.moveTo(newFramePosition)
```

Для объекта `InlineFrame` используется метод `InlineFrame.setPosition()`. Примеры использования с различными параметрами приведены в разделе описании метода.

5. Масштабирование размеров

Для объекта `AbsoluteFrame` используется метод `AbsoluteFrame.scale()`

```
newFramePosition = myOfficeSDK.PointU(20, 20)
if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
    frame.scale(0.5, 0.5, myOfficeSDK.ScaleFrom_TopLeft)
```

6. Установка обтекания текстом

Для `InlineFrame` вариант обтекания текстом графического объекта `TextWrapType` может быть задан посредством использованием метода `InlineFrame.setWrapType()`.

```
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    frame.setWrapType(myOfficeSDK.TextWrapType_Inline)
```

4.3 Поиск в документе

Для поиска в текстовом или табличном документе необходимо создать экземпляр класса `Search` посредством вызова `createSearch(document)`, затем использовать метод `Search.findText` (см. Рисунок 9).

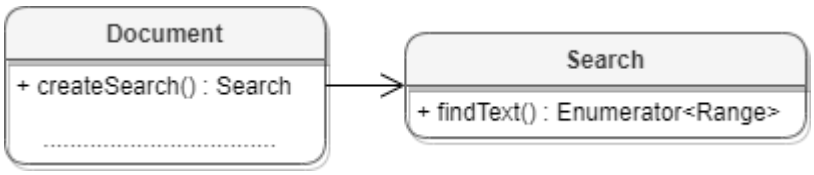


Рисунок 9 – Объектная модель для поиска в документе

Примеры поиска в документе:

```
# Поиск по всему документу
search = myOfficeSDK.createSearch(document)
searchResult = search.findText("Yellow", myOfficeSDK.CaseSensitive_Yes)
```

```
// Поиск в диапазоне блока
```

```
range = document.getBlocks().getBlock(0).getRange();
search = myOfficeSDK.createSearch(document)
searchResult = search.findText("Yellow", range, myOfficeSDK.CaseSensitive_Yes)
```

```
// Поиск в диапазоне ячеек
table = document.getBlocks().getTable(0);
cellRange = table.getCellRange("A1:B2");
search = myOfficeSDK.createSearch(document)
searchResult = search.findText("Yellow", cellRange,
myOfficeSDK.CaseSensitive_Yes)
```

```
// Поиск в таблице
table = document.getBlocks().getTable(0);
search = myOfficeSDK.createSearch(document)
searchResult = search.findText("Yellow", table, myOfficeSDK.CaseSensitive_Yes)
```

```
// Отображение результатов поиска
for searchRange in searchResult:
    print(searchRange.extractText())
```

4.4 Работа с макрокомандами

Класс `Scripts` предоставляет доступ к списку макрокоманд документа. На рисунке 10 изображена объектная модель классов, относящихся к работе с макрокомандами.

Класс `Scripts` предназначен для доступа к списку макрокоманд, доступен через метод `Document.getScripts()`, класс `Scripting` служит для запуска макрокоманд, доступен через `Scripting.createScripting(document)`.

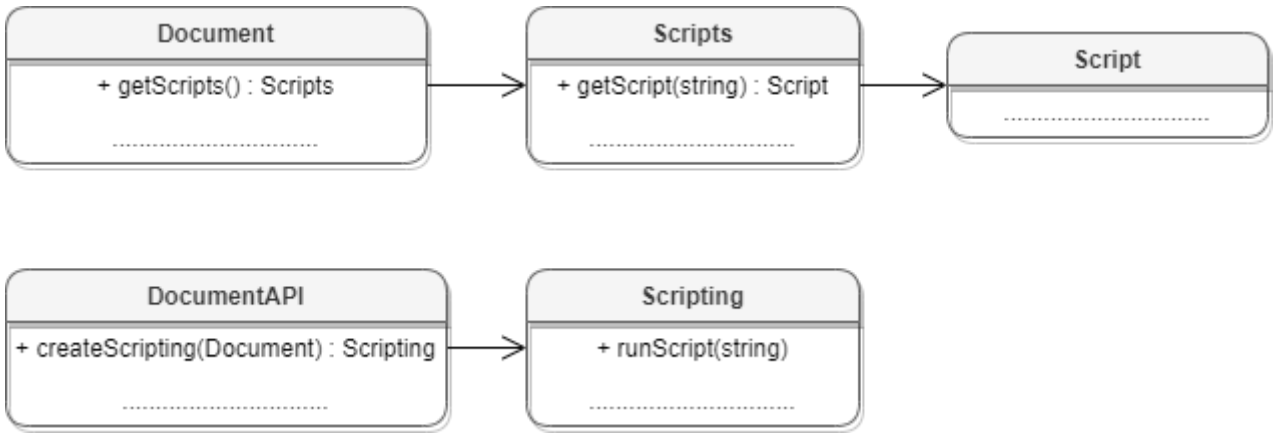


Рисунок 10 – Объектная модель классов для работы с макрокомандами

Доступны следующие операции:

- получение списка макрокоманд;
- добавление макрокоманды;
- получение макрокоманды по имени;
- удаление макрокоманды;
- запуск макрокоманды.

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 11).

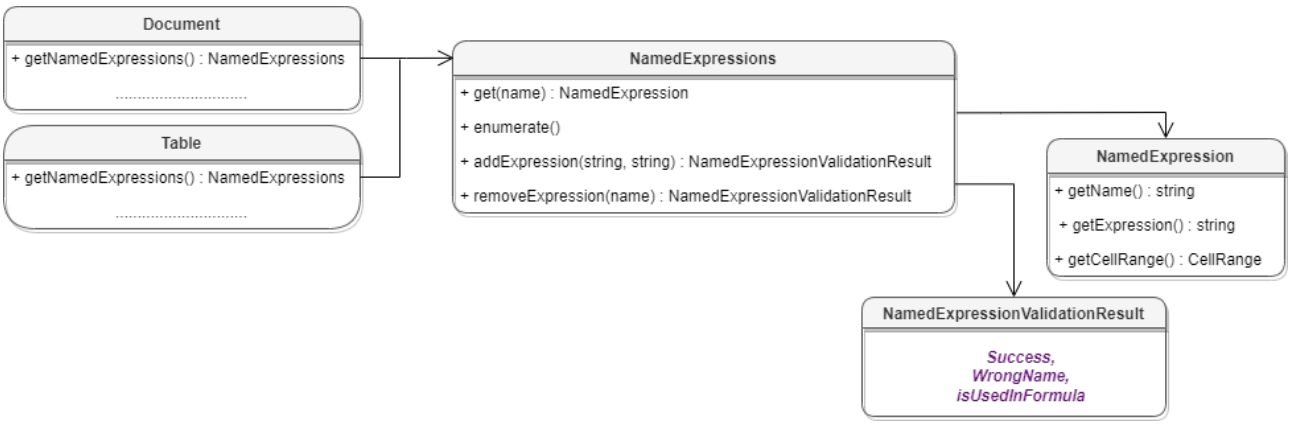


Рисунок 11 – Классы для работы с именованными диапазонами

4.5.1 Доступ к именованным диапазонам

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

Примеры:

```
namedExpressions = document.getNamedExpressions()
```

```
tablesEnumerator = document.getBlocks().getTablesEnumerator()
for table in tablesEnumerator:
    namedExpressions = table.getNamedExpressions()
    namedExpressionsEnumerator = namedExpressions.getEnumerator()
```

```
    for namedExpression in namedExpressionsEnumerator:
        print(namedExpression.getName())
```

4.5.2 Получение коллекции именованных диапазонов

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

Пример:

```
firstSheet = document.getBlocks().getTable(0)
namedExpressionsEnumerator = firstSheet.getNamedExpressions().getEnumerator()
for namedExpressionIndex, namedExpression in
enumerate(namedExpressionsEnumerator):
    print(namedExpression.getName())
    print(namedExpression.getExpression())
```

4.5.3 Получение свойств именованного диапазона

Пример:

```
firstSheet = document.getBlocks().getTable(0)
namedExpressions = firstSheet.getNamedExpressions()
namedExpression = namedExpressions.get("Alice_Age")
name = namedExpression.getName()
formula = namedExpression.getExpression()
range = namedExpression.getCellRange()
```

4.5.4 Добавление именованного диапазона

Для добавления именованного диапазона используется метод NamedExpressions.addExpression(). В качестве результата операции метод возвращает значение типа NamedExpressionsValidationResult.

Пример:

```
expressionName = "Продажи"
expressionValue = "=Формула покупки!$A$6:$A$14"
validationResult = namedExpressions.addExpression(expressionName,
expressionValue)
```

4.5.5 Удаление именованного диапазона

Для удаления именованного диапазона используется метод NamedExpressions.removeExpression(). В качестве результата операции метод возвращает значение типа NamedExpressionsValidationResult.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
namedExpressions = firstSheet.getNamedExpressions()
expressionName = "Продажи"
validationResult = namedExpressions.removeExpression(expressionName)
print(validationResult)
```

4.5.6 Получение параметров именованного диапазона

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

Пример:

```
name = namedExpression.getName()
formula = namedExpression.getExpression()
range = namedExpression.getCellRange()
```

4.6 Работа 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.clearColumnnGroups().

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:

```
table = document.getBlocks().getTable(0)
print(table.isRowVisible(0))
print(table.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:

```
beginRow = 1
lastRow = 3
beginColumn = 2
lastColumn = 3

visibility = False

table.setRowsVisible(beginRow, lastRow - beginRow + 1, visibility)
table.setColumnsVisible(beginColumn, lastColumn - beginColumn + 1, visibility)
```

4.7 Работа с ячейками таблиц

4.7.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 12):

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

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

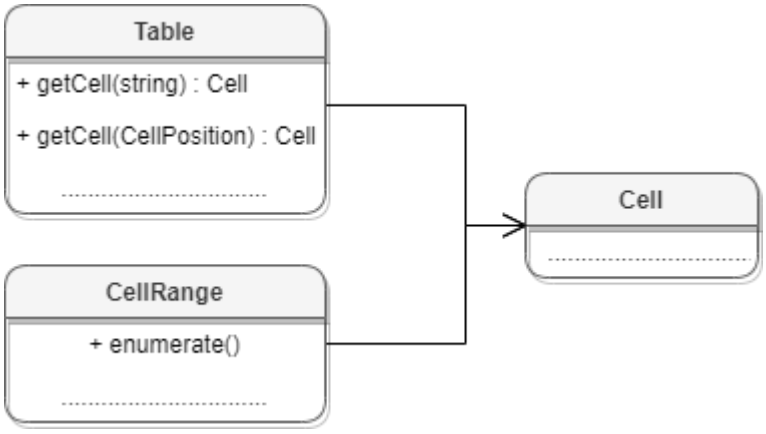


Рисунок 12 – Объектная модель для работы с ячейками таблиц

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

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("B1")
```

Второй вариант доступа к ячейке - перечисление диапазона ячеек с помощью метода `CellRange.getEnumerator()`.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
cellRangesEnumerator = cellRange.getEnumerator()
for cell in cellRangesEnumerator:
    print(cell.getFormattedValue())
```

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

Примеры:

```
table1 = document.getBlocks().getTable(0)
table2 = document.getBlocks().getTable(1)

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

cell1 = table1.getCell("A1")
```

```
cell2 = table1.getCell("C4")
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:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("B1")
cell.setText("Текст")
print(cell.getFormattedValue())

cell.setNumber(10)
print(cell.getFormattedValue())

cell.setFormula("=SUM(B2:B3)")
print(cell.getFormattedValue())

cell.setBool(False)
print(cell.getFormattedValue())

cell.setFormattedValue("12:39")
print(cell.getFormattedValue())
```

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("B1")
cell.setFormattedValue("22.07.2020")
print(cell.getFormattedValue())
```

```
cell.setFormattedValue("12:39")
print(cell.getFormattedValue())
```

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("B1")
cell.setFormat(myOfficeSDK.CellFormat_Accounting)
cell.setNumber(12)
print(cell.getFormattedValue())
```

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("B1")
print(cell.getFormattedValue())
```

4.7.2 Форматирование ячеек

При работе с ячейками таблиц можно использовать следующие варианты форматирования:

- форматирование параметров ячейки `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 Paragraph, и обладает свойствами `ParagraphProperties`. Это дает возможность управлять настройками отображения контента как отдельного абзаца, так и группы абзацев (например, если ячейка содержит несколько предложений текста). Для управления этим настройками используются методы `Cell.getParagraphProperties()` и `Cell.setParagraphProperties()`.

Пример установки и получения свойств параграфа ячейки:

```
firstSheet = document.getBlocks().getTable("Table1")
cell = firstSheet.getCell("A2")

paragraphProperties = cell.getParagraphProperties()
paragraphProperties.alignment = myOfficeSDK.Alignment_Center
cell.setParagraphProperties(paragraphProperties)
```

Управление настройками текста ячейки (шрифт, цвет) производится через соответствующий ему диапазон. Класс Cell позволяет получить диапазон для всего контента с помощью метода `Cell.getRange()`. Далее, метод `Range.getTextProperties()` позволяет получить экземпляр класса `TextProperties`, представляющий свойства текста. После изменения значения свойств их необходимо применить к тексту ячейки с помощью метода `Range.setTextProperties()`.

Пример настроек текста ячейки:

```
firstSheet = document.getBlocks().getTable("Table1")
cell = firstSheet.getCell(myOfficeSDK.CellPosition(0,1))

textProperties = cell.getRange().getTextProperties()
textProperties.bold = True
textProperties.italic = True
textProperties.textColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146, 179,
 200))

cell.getRange().setTextProperties(textProperties)
```

4.7.3 Форматирование границ ячеек

Для оформления границ ячеек используется класс `Borders` (см. Рисунок 13). Он описывает свойства полей, соответствующих границам и диагоналям ячейки: `Left`, `Right`, `Top`, `Bottom`, `DiagonalDown`, `DiagonalUp`, `InnerHorizontal`, `InnerVertical`. Каждая граница ячейки описывается классом `LineProperties`, который, в свою очередь, обладает свойствами `LineStyle`, `LineEndingProperties`, `Color`, `LineWidth`.

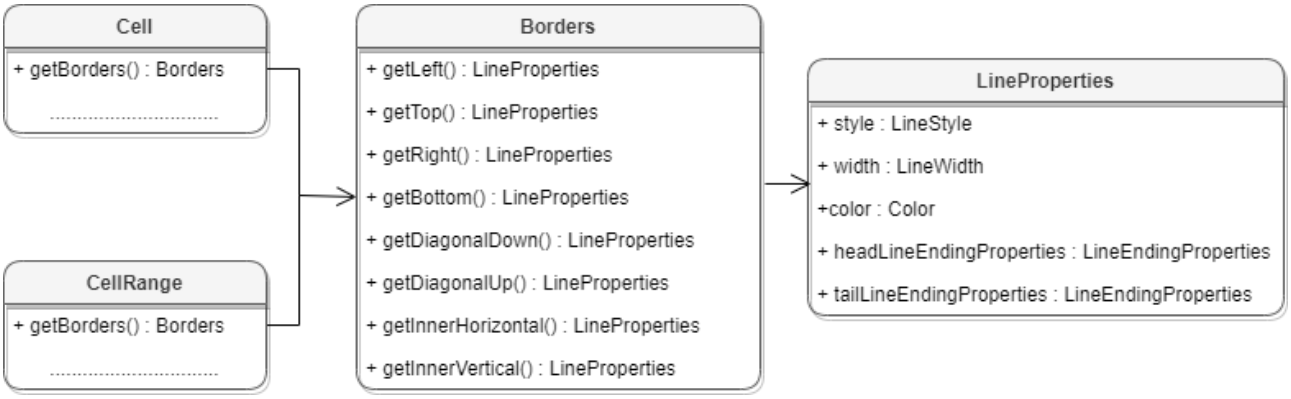


Рисунок 13 – Классы для работы с границами ячеек

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

1. Получить ячейку `Cell` или область ячеек `CellRange`.
2. Настроить параметры для рисования линии границы с помощью экземпляра класса `LineProperties`.
3. Настроить свойства линии: левой границы, верхней границы и т.д. с помощью экземпляра класса `Borders`.
4. Установить границы ячеек с помощью `Cell.setBorders()` или `CellRange.setBorders()`.

Пример настройки границ ячеек:

```
firstSheet = document.getBlocks().getTable("Table1")
cellRange = firstSheet.getCellRange("A3:D5")

lineProperties = myOfficeSDK.LineProperties()
lineProperties.style = myOfficeSDK.LineStyle_Solid
lineProperties.width = 1.5
lineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146, 179,
200))
```

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
firstSheet = document.getBlocks().getTable(0)
firstSheet.getCellRange("A1:A2").merge()
```

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 используется метод `Cell.unmerge()`.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
# Ячейка A1 является результатом объединения диапазона A1:A2
cell = firstSheet.getCell("A1")
cell.unmerge()
```

5 Глобальные методы

5.1 Глобальный метод createSearch

Метод инициализирует механизм поиска для текущего документа. Возвращает объект `Search`, с помощью которого выполняются поисковые запросы.

Пример:

```
search = myOfficeSDK.createSearch(document)
search.findText("API")
```

5.2 Глобальный метод DocumentAPI.createScripting

Вызов `DocumentAPI.createScripting(document)`, возвращает объект класса `Scripting` который используется для запуска макрокоманды.

Пример:

```
scripting = myOfficeSDK.createScripting(document);
```

6 Справочник классов, структур и методов

6.1 Класс AbsoluteFrame

Класс `AbsoluteFrame` описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 14). Предназначен для получения и изменения свойств позиции медиаобъектов. Используется в табличном документе.

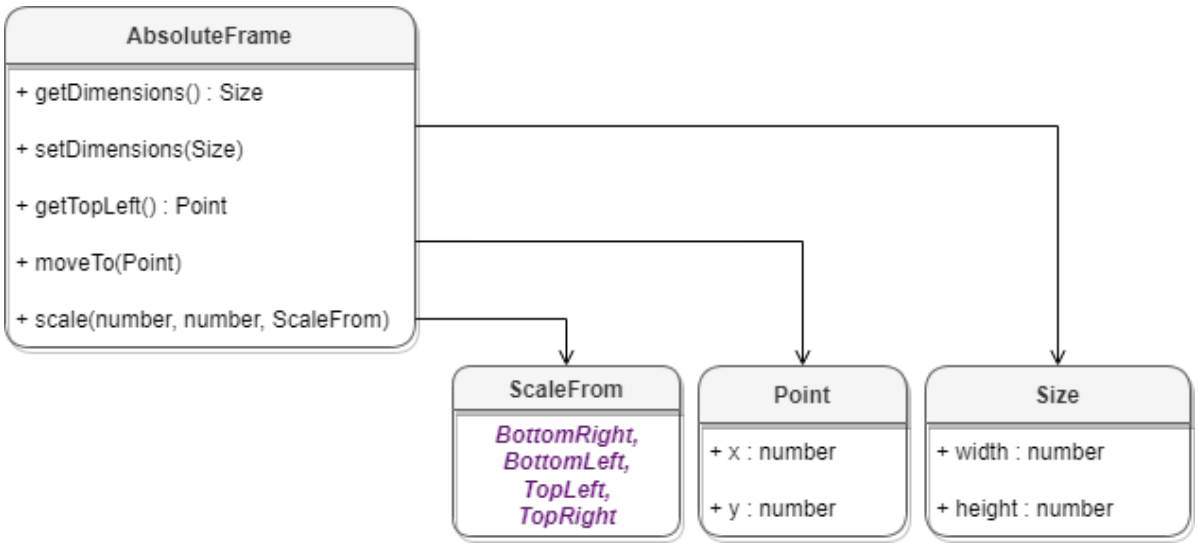


Рисунок 14 – Объектная модель класса `AbsoluteFrame`

Пример для табличного документа:

```
sheet = document.getBlocks().getTable("Лист1")
for mediaObject in sheet.getMediaObjects():
   frame = mediaObject.getFrame()
   if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
      print(frame.getDimensions())
      print(frame.getTopLeft())
```

6.1.1 Метод AbsoluteFrame.getDimensions

Возвращает размеры медиаобъекта, тип - `SizeU`.

6.1.2 Метод AbsoluteFrame.getTopLeft

Метод возвращает позицию верхней левой точки медиаобъекта.

Пример:

```
sheet = document.getBlocks().getTable(0)
for mediaObject in sheet.getMediaObjects().getEnumerator():
```

```
   topLeftPosition = mediaObject.getFrame().getTopLeft()
   print("x=", topLeftPosition.x, "y=", topLeftPosition.y)
```

6.1.3 Метод AbsoluteFrame.moveTo

Метод перемещает объект в заданную позицию.

Пример:

```
sheet = document.getBlocks().getTable(0)
for mediaObject in sheet.getMediaObjects().getEnumerator():
   frame = mediaObject.getFrame()
   newFramePosition = myOfficeSDK.PointU(20, 20)
   if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
      frame.moveTo(newFramePosition)
```

6.1.4 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 - `ScaleFrom`.

Пример:

```
# Уменьшение масштаба всех медиаобъектов на 50%
sheet = document.getBlocks().getTable(0)
for mediaObject in sheet.getMediaObjects().getEnumerator():
   mediaObject.getFrame().scale(0.5, 0.5, myOfficeSDK.ScaleFrom_TopLeft)
```

6.1.5 Метод AbsoluteFrame.setDimensions

Метод задает размеры (изменяет размер) медиаобъекта.

Вызов:

```
setDimensions(size)
```

Параметры:

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

Пример:

```
# Изменение размера всех медиаобъектов
sheet = document.getBlocks().getTable(0)
for mediaObject in sheet.getMediaObjects().getEnumerator():
   mediaObject.getFrame().setDimensions(100, 100)

sheet = document.getBlocks().getTable(0)
for mediaObject in sheet.getMediaObjects().getEnumerator():
   mediaObject.getFrame().setDimensions(myOfficeSDK.SizeU(100, 100))
```

6.2 Класс AccountingCellFormatting

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

Описание полей класса `AccountingCellFormatting` представлено в таблице 3.

Таблица 3 – Описание полей класса `AccountingCellFormatting`

Поле	Описание
`AccountingCellFormatting.decimalPlaces`	Количество десятичных позиций
`AccountingCellFormatting.symbol`	Символ денежной единицы
`AccountingCellFormatting.localeCode`	Идентификатор кода языка (MS-LCID)
`AccountingCellFormatting.fillSymbol`	Символ заполнения
`AccountingCellFormatting.useThousandsSeparator`	Использовать разделитель для тысячных
`AccountingCellFormatting.currencySignPlacement`	Тип размещения знака валюты `CurrencySignPlacement`

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("A2")
```

```
accountingCellFormat = myOfficeSDK.AccountingCellFormatting()
accountingCellFormat.decimalPlaces = 2
accountingCellFormat.symbol = "Руб"

cell.setFormat(accountingCellFormat)
print(cell.getFormattedValue())
```

6.3 Класс Alignment

Тип `Alignment` содержит варианты горизонтального выравнивания текста, в том числе в ячейке таблицы. Варианты выравнивания текста представлены в таблице 4.

Таблица 4 - Типы горизонтального выравнивания текста

Имя константы типа выравнивания текста	Описание
`Alignment_Default`	Выравнивание текста по умолчанию
`Alignment_Left`	Выравнивание текста по левому краю
`Alignment_Center`	Выравнивание текста по центру
`Alignment_Right`	Выравнивание по правому краю
`Alignment_Justify`	Выравнивание по ширине
`Alignment_Distributed`	Распределенное выравнивание, при применении между словами добавляется пробел, чтобы оба края каждой строки были выровнены по обеим сторонам. Последняя строка в абзаце также выравнивается по обеим сторонам, но если строка состоит из одного слова, то выравнивание по правой стороне не осуществляется
`Alignment_Fill`	Распределение текста по горизонтали – заполнение строки текстом

Пример:

```
paragraphProperties = paragraph.getParagraphProperties()
paragraphProperties.alignment = myOfficeSDK.Alignment_Center
paragraph.setParagraphProperties(paragraphProperties)
```

6.4 Класс Application

Класс `Application` управляет параметрами и объектами приложения. Предоставляет интерфейс для создания и загрузки документов. Допустимо использование только одного объекта `Application` для всего сеанса обработки документа.

6.4.1 Метод Application.createDocument

Метод `Application.createDocument` создает новый документ с типом `DocumentType`, либо `DocumentSettings`.

Примеры:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as myOfficeSDK
application = myOfficeSDK.Application()

documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Text
document = application.createDocument(documentSettings)
document.saveAs("NewTextDocument.xodt")
```

```
documentSettings = myOfficeSDK.DocumentSettings()
document = application.createDocument(myOfficeSDK.DocumentType_Workbook)
document.saveAs("NewSheetDocument.xlsx")
```

6.4.2 Метод Application.getMessenger

Метод `Application.getMessenger` возвращает объект `Messenger`, реализующий логирование событий.

Пример:

```
handler = myOfficeSDK.MessageHandler()
messenger = application.getMessenger()
connection = messenger.subscribe(handler)
```

6.4.3 Метод Application.loadDocument

Метод `Application.loadDocument` загружает существующий текстовый или табличный документ из файла, находящегося по указанному пути. Формат и тип документа определяются из расширения файла, если не указаны явно с помощью параметра `LoadDocumentSettings`.

Используется один из следующих вариантов метода:

```
application.loadDocument(path: String)
application.loadDocument(path: String, loadSettings: LoadDocumentSettings)
```

Примеры загрузки текстового документа:

```
document = application.loadDocument("document.docx")
```

```
documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Text
loadSettings = myOfficeSDK.LoadDocumentSettings()
loadSettings.commonDocumentSettings = documentSettings
document = application.loadDocument("document.docx", loadSettings)
```

Примеры загрузки табличного документа:

```
document = application.loadDocument("spreadsheet.xlsx", loadSettings)
```

```
documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Workbook
loadSettings = myOfficeSDK.LoadDocumentSettings()
loadSettings.commonDocumentSettings = documentSettings
document = application.loadDocument("spreadsheet.xlsx", loadSettings)
```

6.5 Класс Block

Класс `Block` является базовой для всех блоков документа. От нее наследуются классы `Paragraph`, `Table`, `Shape`, `Field` (см. Рисунок 15).

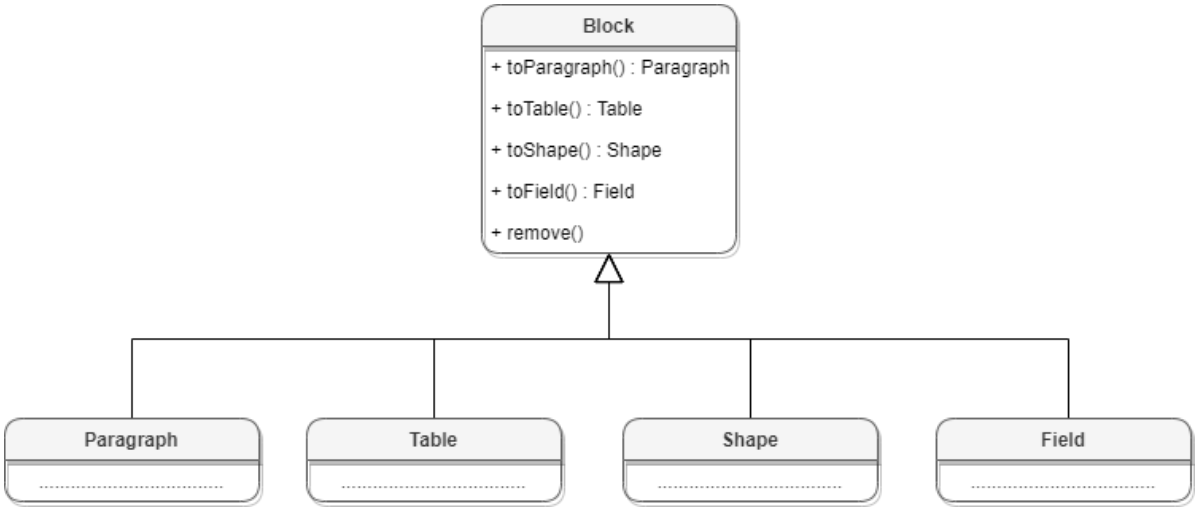


Рисунок 15 – Объектная модель класса `Block`

6.5.1 Метод Block.getRange

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

Пример:

```
blocks = document.getBlocks()
block = blocks.getBlock(0)
```

```
if block != None:
    print(block.getRange().extractText())
```

6.5.2 Метод Block.getSection

Метод возвращает раздел `Section`, содержащий блок.

Пример:

```
blocks = document.getBlocks()
block = blocks.getBlock(0)
if block != None:
    section = block.getSection()
    print(section.getRange().extractText())
```

6.5.3 Метод Block.remove

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

Пример:

```
blocks = document.getBlocks()
block = blocks.getBlock(0)
if block != None:
    block.remove()
```

6.5.4 Методы toParagraph, toTable, toShape, toField

Преобразует объект `Block` в объект соответствующего типа.

Пример:

```
blocks = document.getBlocks()
block = blocks.getBlock(0)
if block != None:
    paragraph = block.toParagraph()
    if paragraph != None:
        print(paragraph.getRange().extractText())
```

6.6 Класс Blocks

Класс `Blocks` обеспечивает доступ к блокам `Block` документа или диапазона документа (см. Рисунок 16). Объект класса `Blocks` может быть получен вызовом метода `Document.getBlocks` или `HeaderFooter.getBlocks`.

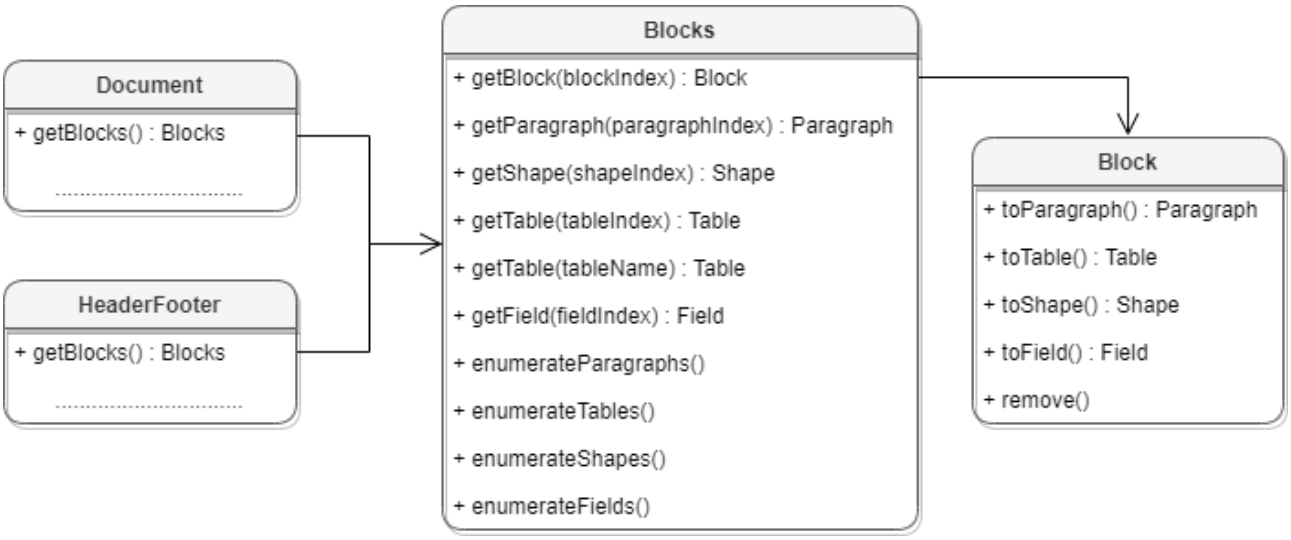


Рисунок 16 – Объектная модель класса `Blocks`

6.6.1 Метод Blocks.getBlock

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

Пример:

```
blocks = document.getBlocks()
firstBlock = blocks.getBlock(0)
if firstBlock != None:
    print(firstBlock.getRange().extractText())
```

6.6.2 Метод Blocks.GetEnumerator

Позволяет реализовать перечисление объектов `Block`.

Пример:

```
blocks = document.getBlocks()
blocksEnumerator = blocks.getEnumerator()
for blockIndex, block in enumerate(blocksEnumerator):
    print(block.getRange().extractText())
```

6.6.3 Метод Blocks.getField

Возвращает объект типа `Field` по заданному индексу.

Пример:

```
blocks = document.getBlocks()
field = blocks.getField(0)
if field != None:
    print(shape.getRange().extractText())
```

6.6.4 Метод Blocks.getFieldsEnumerator

Позволяет перечислить объекты типа `Field`.

Пример:

```
blocks = document.getBlocks()
fieldsEnumerator = blocks.getFieldsEnumerator()
for fieldIndex, field in enumerate(fieldsEnumerator):
    print(field.getRange().extractText())
```

6.6.5 Метод Blocks.getParagraph

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

Пример:

```
blocks = document.getBlocks()
firstParagraph = blocks.getParagraph(0)
if firstParagraph != None:
    print(firstParagraph.getRange().extractText())
```

6.6.6 Метод Blocks.getParagraphsEnumerator

Позволяет реализовать перечисление абзацев `Paragraph`.

Пример:

```
blocks = document.getBlocks()
paragraphsEnumerator = blocks.getParagraphsEnumerator()
for paragraphIndex, paragraph in enumerate(paragraphsEnumerator):
    print(paragraph.getRange().extractText())
```

6.6.7 Метод Blocks.getShape

Возвращает фигуру `Shape` по заданному индексу.

Пример:

```
blocks = document.getBlocks()
shape = blocks.getShape(0)
```

```
if shape != None:
    print(shape.getRange().extractText())
```

6.6.8 Метод Blocks.getShapesEnumerator

Позволяет перечислить объекты типа Shape.

Пример:

```
blocks = document.getBlocks()
shapesEnumerator = blocks.getShapesEnumerator()
for shapeIndex, shape in enumerate(shapesEnumerator):
    print(shape.getRange().extractText())
```

6.6.9 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:

```
blocks = document.getBlocks()
table = blocks.getTable(0)
if table != None:
    print(table.getRange().extractText())
```

В качестве параметра метода также можно указать имя таблицы.

Пример:

```
blocks = document.getBlocks()
table = blocks.getTable("List11")
if table != None:
    print(table.getRange().extractText())
```

6.6.10 Метод Blocks.getTablesEnumerator

Позволяет перечислить объекты типа Table.

Пример:

```
blocks = document.getBlocks()
tablesEnumerator = blocks.getTablesEnumerator()
for tableIndex, table in enumerate(tablesEnumerator):
    print(table.getRange().extractText())
```

6.7 Класс Bookmarks

Предоставляет доступ к операциям с закладками в документе.

6.7.1 Метод Bookmarks.getBookmarkRange

Возвращает экземпляр объекта `Range` для дальнейшей работы с содержимым закладки.

Пример:

```
bookmarks = document.getBookmarks()
bookmarkRange = bookmarks.getBookmarkRange("Bookmark")
if bookmarkRange != None:
    bookmarkRange.replaceText("New bookmark text");
    print("Bookmark range text : ", bookmarkRange.extractText());
```

6.7.2 Метод Bookmarks.removeBookmark

Удаляет закладку по ее названию.

Пример:

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

6.8 Класс Borders

Класс `Borders` предназначен для оформления границ отдельной ячейки таблицы (см. таблицу 5). Параметры линии, такие как тип линии, ее ширина и цвет, задаются с помощью объектов типа `LineProperties`.

Таблица 5 – Описание методов класса `Borders`

Метод	Описание
`Borders.setLeft`	Установка левой границы ячейки, метод возвращает `Borders`
`Borders.setRight`	Установка правой границы ячейки, метод возвращает `Borders`
`Borders.setTop`	Установка верхней границы ячейки, метод возвращает `Borders`
`Borders.setBottom`	Установка нижней границы ячейки, метод возвращает `Borders`
`Borders.setDiagonalDown`	Установка диагональной линии, метод возвращает `Borders` 

Метод	Описание
`Borders.setDiagonalUp`	Установка диагональной линии, метод возвращает `Borders` 
`Borders.setOuter`	Установка внешних границ ячейки, метод возвращает `Borders`
`Borders.setDiagonals`	Установка обеих типов диагональных линий одновременно, метод возвращает `Borders`
`Borders.setInnerHorizontal`	Установка внутренних горизонтальных границ ячейки, метод возвращает `Borders`
`Borders.setInnerVertical`	Установка внутренних вертикальных границ ячейки, метод возвращает `Borders`
`Borders.setInner`	Установка внутренних границ ячейки, метод возвращает `Borders`
`Borders.setAll`	Установка всех границ ячейки, метод возвращает `Borders`
`Borders.getLeft`	Получение левой границы ячейки
`Borders.getRight`	Получение правой границы ячейки
`Borders.getTop`	Получение верхней границы ячейки
`Borders.getBottom`	Получение нижней границы ячейки
`Borders.getDiagonalDown`	Получение диагональной линии
`Borders.getDiagonalUp`	Получение диагональной линии
`Borders.getInnerHorizontal`	Получение внутренних горизонтальных границ ячейки
`Borders.getInnerVertical`	Получение внутренних вертикальных границ ячейки

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("C3")

lineProperties = myOfficeSDK.LineProperties()
lineProperties.style = myOfficeSDK.LineStyle_Solid
lineProperties.width = 1.5
lineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146, 179,
200))

borders = myOfficeSDK.Borders()
borders.setLeft(lineProperties)
borders.setTop(lineProperties)
```

```
borders.setRight(lineProperties)
borders.setBottom(lineProperties)
cell.setBorders(borders)
```

6.9 Класс CalculationMode

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

Таблица 6 – Режимы пересчета формул

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

6.10 Класс CaseSensitive

Параметры настройки учета регистра при поиске (см. метод `Search.FindText()`) представлены в таблице 7.

Таблица 7 – Параметры регистра при поиске

Наименование константы	Описание
`myOfficeSDK.CaseSensitive_Yes`	Поиск с учетом регистра
`myOfficeSDK.CaseSensitive_No`	Поиск без учета регистра

6.11 Класс Cell

Класс `Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 17).

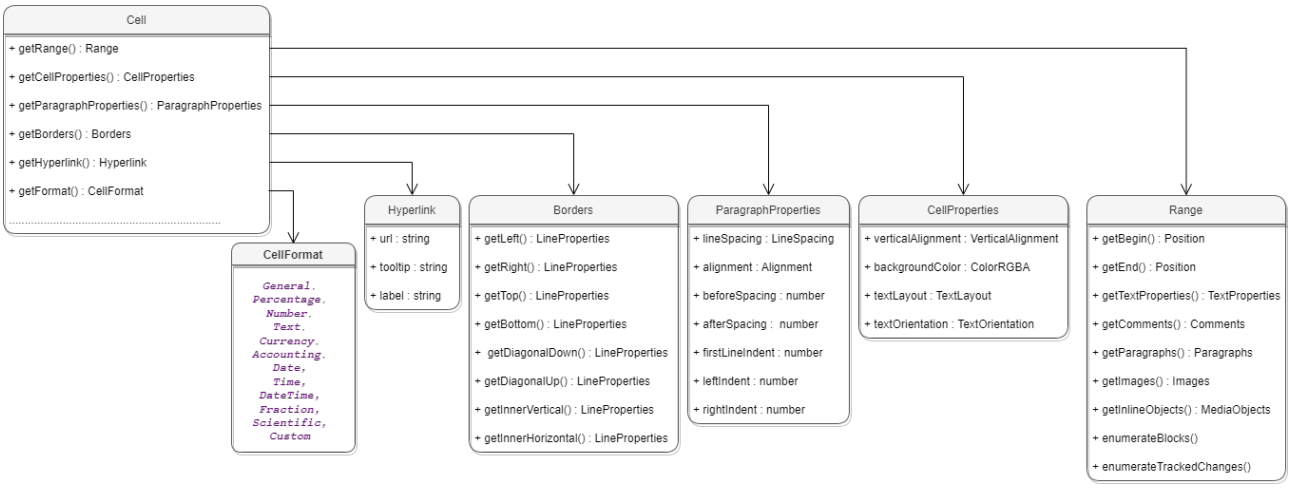


Рисунок 17 – Объектная модель ячейки таблиц

6.11.1 Метод Cell.getBorders

Позволяет получить границы ячейки `Borders`.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
borders = cell.getBorders()
print(borders.getLeft().width)
```

6.11.2 Метод Cell.getCellProperties

Позволяет получить свойства `CellProperties` ячейки.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
cellProperties = cell.getCellProperties()
print(cellProperties.verticalAlignment)
```

6.11.3 Метод Cell.getCustomFormat

Возвращает строку формата ячейки.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
print(cell.getCustomFormat())
```

6.11.4 Метод Cell.getFormat

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cellFormatting = myOfficeSDK.PercentageCellFormatting()
cellFormatting.decimalPlaces = 2
cell.setFormat(cellFormatting)
print(cell.getFormat())
```

6.11.5 Метод Cell.getFormattedValue

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cell.setNumber(2.3)
print(cell.getFormattedValue())
```

6.11.6 Метод Cell.getFormulaAsString

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cell.setFormula("=SUM(A1:A2)")
print(cell.getFormulaAsString())
```

6.11.7 Метод Cell.getHyperlink

Возвращает первый объект в ячейке типа Hyperlink.

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
hyperlink = cell.getHyperlink()
```

6.11.8 Метод Cell.getParagraphProperties

Возвращает свойства абзаца ParagraphProperties, находящегося в ячейке.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
paragraphProperties = cell.getParagraphProperties()
print(paragraphProperties.alignment)
```

6.11.9 Метод Cell.getPivotTable

Возвращает сводную таблицу PivotTable, относящуюся к ячейке.

Пример:

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    print(pivotTable.getSourceRangeAddress())
```

6.11.10 Метод Cell.getProtectionProperties

Метод возвращает параметры защиты ячейки табличного документа.

Вызов:

```
CellProtectionProperties getProtectionProperties()
```

Возвращает:

– CellProtectionProperties: свойства защиты ячейки (None, если ячейка находится в сводной таблице).

Пример:

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

cellProps = cell.getProtectionProperties()
cellProps.lockedForChanges = False
cellProps.formulasNotDisplayed = False

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

Метод setProtectionProperties() позволяет задать параметры защиты ячейки. Вы также можете использовать метод isProtected(), чтобы узнать, защищена ли ячейка от редактирования.

6.11.11 Метод Cell.getRange

Метод возвращает объект Range для управления содержимым ячейки.

6.11.12 Метод Cell.getRawValue

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
print(cell.getRawValue())
```

6.11.13 Метод Cell.isPivotTableRoot

Метод позволяет определить является ли ячейка основанием сводной таблицы.

Пример:

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("A1")
print(cell.isPivotTableRoot())
```

6.11.14 Метод Cell.isProtected

Метод возвращает статус защиты от редактирования ячейки в табличном документе.

Вызов:

```
bool isProtected()
```

Методы setProtectionProperties() и getProtectionProperties() позволяют задать и получить параметры защиты ячейки (CellProtectionProperties).

6.11.15 Метод Cell.setBool

Устанавливает для ячейки значение логического типа.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cell.setBool(True)
print(cell.getFormattedValue())
```

6.11.16 Метод Cell.setBorders

Метод предназначен для установки границ ячейки. Примеры использования приведены в разделе Borders.

6.11.17 Метод Cell.setCellProperties

Позволяет установить свойства ячейки CellProperties.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
cellProperties = cell.getCellProperties()
cellProperties.verticalAlignment = myOfficeSDK.VerticalAlignment_Center
cell.setCellProperties(cellProperties)
```

6.11.18 Метод Cell.setContent

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

Пример:

```
cell = firstSheet.getCell("A1")
cell.setContent("=A2+A3")
```

6.11.19 Метод Cell.setCustomFormat

Устанавливает формат ячейки.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cell.setContent("11")
cell.setCustomFormat("0.00")
print(cell.getFormattedValue())
```

6.11.20 Метод Cell.setFormat

Метод устанавливает формат ячейки. Существуют несколько вариантов использования метода.

Варианты вызова метода:

```
setFormat(cellFormat)
```

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

```
setFormat(accountingCellFormatting)
```

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

```
setFormat(percentageCellFormatting)
```

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

```
setFormat(numberCellFormatting)
```

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

```
setFormat(currencyCellFormatting)
```

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

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

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

```
setFormat(fractionCellFormatting)
```

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

```
setFormat(scientificCellFormatting)
```

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

Примеры использования:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("A1")
cell.setNumber(2.3)
```

```
// Формат: Общий
cell.setFormat(myOfficeSDK.CellFormat_General)
print(cell.getFormat()) # 0
print(cell.getRange().extractText()) # 2,3
```

```
// Формат : Процентный
percentageCellFormatting = myOfficeSDK.PercentageCellFormatting()
percentageCellFormatting.decimalPlaces = 1
cell.setFormat(percentageCellFormatting)
print(cell.getFormat()) # 1
print(cell.getRange().extractText()) # 230,0%
```

```
// Формат : Числовой
numberCellFormatting = myOfficeSDK.NumberCellFormatting()
numberCellFormatting.decimalPlaces = 2
cell.setFormat(numberCellFormatting)
print(cell.getFormat()); # 2
print(cell.getRange().extractText()); # 2,30
```

```
// Формат : Денежный
currencyCellFormatting = myOfficeSDK.CurrencyCellFormatting()
currencyCellFormatting.symbol = "$"
cell.setFormat(currencyCellFormatting)
print(cell.getFormat()) # 4
print(cell.getRange().extractText()) # 2,30$
```

```
// Формат : Финансовый
accountingCellFormatting = myOfficeSDK.AccountingCellFormatting()
accountingCellFormatting.symbol = "₽"
cell.setFormat(accountingCellFormatting)
print(cell.getFormat()) # 5
print(cell.getRange().extractText()) # 2,30₽
```

```
// Формат : Дата / Время
dateTimeCellFormatting = myOfficeSDK.DateTimeCellFormatting()
dateTimeCellFormatting.dateListID = myOfficeSDK.DatePatterns_FullDate
dateTimeCellFormatting.timeListID = myOfficeSDK.TimePatterns_ShortTime
cell.setFormat(dateTimeCellFormatting)
print(cell.getFormat()); # 8
print(cell.getRange().extractText()) # Monday, January 1, 1900 7:12 AM
```

```
// Формат : Дробный
fractionCellFormatting = myOfficeSDK.FractionCellFormatting()
fractionCellFormatting.minNumeratorDigits = 2;
cell.setFormat(fractionCellFormatting)
print(cell.getFormat()); # 9
print(cell.getRange().extractText()); # 2 2 / 7
```

```
// Формат : Научный
cellFormatting = myOfficeSDK.ScientificCellFormatting()
cellFormatting.decimalPlaces = 5
cell.setFormat(cellFormatting)
print(cell.getFormat()) # 10
print(cell.getRange().extractText()) # 2, 30000E+00
```

6.11.21 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 `CellFormat.Text`.

Список поддерживаемых форматов см. в разделе `CellFormat`.

6.11.22 Метод Cell.setFormula

Метод позволяет вставить формулу в ячейку табличного документа.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cell.setFormula("=SUM(A1:A2)")
print(cell.getFormulaAsString())
```

6.11.23 Метод Cell.setNumber

Устанавливает для ячейки значение числового типа.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cell.setCustomFormat("0.0000")
cell.setNumber(0.0112)
print(cell.getFormattedValue())
```

6.11.24 Метод Cell.setParagraphProperties

Устанавливает свойства абзаца `ParagraphProperties`, находящегося в ячейке.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
paragraphProperties = cell.getParagraphProperties()
paragraphProperties.alignment = myOfficeSDK.Alignment_Center
cell.setParagraphProperties(paragraphProperties)
```

6.11.25 Метод Cell.setProtectionProperties

Метод задает параметры защиты ячейки в табличном документе.

Вызов:

```
setProtectionProperties(protectionProps)
```

Параметры:

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

Пример:

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

cellProps = cell.getProtectionProperties()
cellProps.lockedForChanges = False
cellProps.formulasNotDisplayed = False

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

Метод `getProtectionProperties()` позволяет получить текущие параметры защиты ячейки. Вы также можете использовать метод `isProtected()`, чтобы узнать, защищена ли ячейка от редактирования.

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.

6.11.26 Метод Cell.setText

Устанавливает для ячейки значение строкового типа.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")
cell.setText("A1 content")
print(cell.getFormattedValue())
```

6.11.27 Метод Cell.unmerge

Разъединяет несколько ячеек, которые были объединены ранее.

Пример:

```
firstSheet = document.getBlocks().getTable("List11");
cell = firstSheet.getCell("A1")
cell.unmerge()
```

6.12 Класс CellFormat

По умолчанию при создании документа всем ячейкам присваивается формат «Общий». Полный список форматов представлен в таблице 8.

Таблица 8 – Поддерживаемые форматы ячеек таблицы

Наименование константы	Описание
`CellFormat_General`	Формат ячейки «Общий». В этом формате в ячейке отображаются первые 9 символов числа, остальные доступны для просмотра в строке формул. Для дробных чисел в формате «Общий» незначащие нули в дробной части не отображаются. Числа, состоящие более чем из 12 символов, переводятся в экспоненциальную форму после завершения ввода в ячейку.
`CellFormat_Percentage`	Формат ячейки «Процентный». Этот формат используется для представления чисел как процентов. При применении формата «Процентный» введенное число умножается на 100 и обозначается знаком «%».
`CellFormat_Number`	Формат ячейки «Числовой». Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя.
`CellFormat_Text`	Формат ячейки «Текстовый».
`CellFormat_Currency`	Формат ячейки «Денежный». Этот формат используется для представления чисел со знаком или кодом валюты.
`CellFormat_Accounting`	Формат ячейки «Финансовый». Этот формат применяется для чисел, используемых в бухгалтерских документах. В формате «Финансовый»

Наименование константы	Описание
	введенное число автоматически дополняется названием валюты, которая соответствует настройкам системы компьютера. Отрицательные числа в формате «Финансовый» заключаются в круглые скобки в ячейке, а в строке формул остаются в том виде, в котором они были введены.
`CellFormat_Date`	Формат ячейки «Дата». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ.
`CellFormat_Time`	Формат ячейки «Время». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.
`CellFormat_Fraction`	Формат ячейки «Дробный». Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель).
`CellFormat_Scientific`	Формат ячейки «Экспоненциальный». Экспоненциальный (или научный) формат используется для представления больших чисел в короткой форме. Все введенные числа длиной более 12 символов автоматически переводятся в этот формат. В ячейке число в формате «Экспоненциальный» представлено следующим образом: – целая часть, всегда состоящая из одной цифры; – разделитель целой и дробной части; – дробная часть, по умолчанию состоящая из двух цифр; – показатель степени числа 10 в виде **E<знак показателя степени> <показатель степени>**.
`CellFormat_Custom`	Пользовательский формат.

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

Примеры использования:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("B1")
cell.setFormat(myOfficeSDK.CellFormat_General);
```

```
cell = firstSheet.getCell("B2")
cell.setFormat(myOfficeSDK.CellFormat_Percentage)
cell = firstSheet.getCell("B3");
cell.setFormat(myOfficeSDK.CellFormat_Number)
```

Результат:

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

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

6.13 Класс CellPosition

Класс `CellPosition` позволяет задать координаты ячейки электронной таблицы или таблицы в составе текстового документа. Также используется для описания полей `topLeft`, `rightBottom` класса `CellRangePosition`.

Примеры:

```
table = document.getBlocks().getTable(0)
cell = table.getCell(myOfficeSDK.CellPosition(2, 0))
```

```
table = document.getBlocks().getTable("List11")
charts = table.getCharts()
rangeInfo = charts.getChart(0).getRange(0)
cellRangePosition = rangeInfo.tableRangeInfo
tableRange = cellRangePosition.tableRange
topLeftCellPosition = tableRange.topLeft
print("top left row:", topLeftCellPosition.row, ", top left column:",
topLeftCellPosition.column)
```

6.13.1 Поле CellPosition.column

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

Пример:

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

6.13.2 Поле CellPosition.row

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

Пример:

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

6.13.3 Метод CellPosition.toString

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

Пример:

```
cellPosition = myOfficeSDK.CellPosition(0, 0)
print(cellPosition.toString())
```

6.13.4 CellPosition.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `CellPosition`.

Пример:

```
firstCellPosition = myOfficeSDK.CellPosition(10, 20)
secondCellPosition = myOfficeSDK.CellPosition(10, 20)

if firstCellPosition.__eq__(secondCellPosition):
    print("Equals")
```

6.13.5 CellPosition.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `CellPosition`.

Пример:

```
firstCellPosition = myOfficeSDK.CellPosition(10, 20)
secondCellPosition = myOfficeSDK.CellPosition(10, 30)

if firstCellPosition.__ne__(secondCellPosition):
    print("Not equals")
```

6.14 Класс CellProperties

Класс `CellProperties` предназначен для форматирования содержимого в ячейках таблицы. Описание полей представлено в таблице 9.

Для задания свойств ячейки используется метод `Cell.setCellProperties()`. Для получения свойств ячейки используется метод `Cell.getCellProperties()`. Иерархия классов и полей `CellProperties` отображена на рисунке 18.

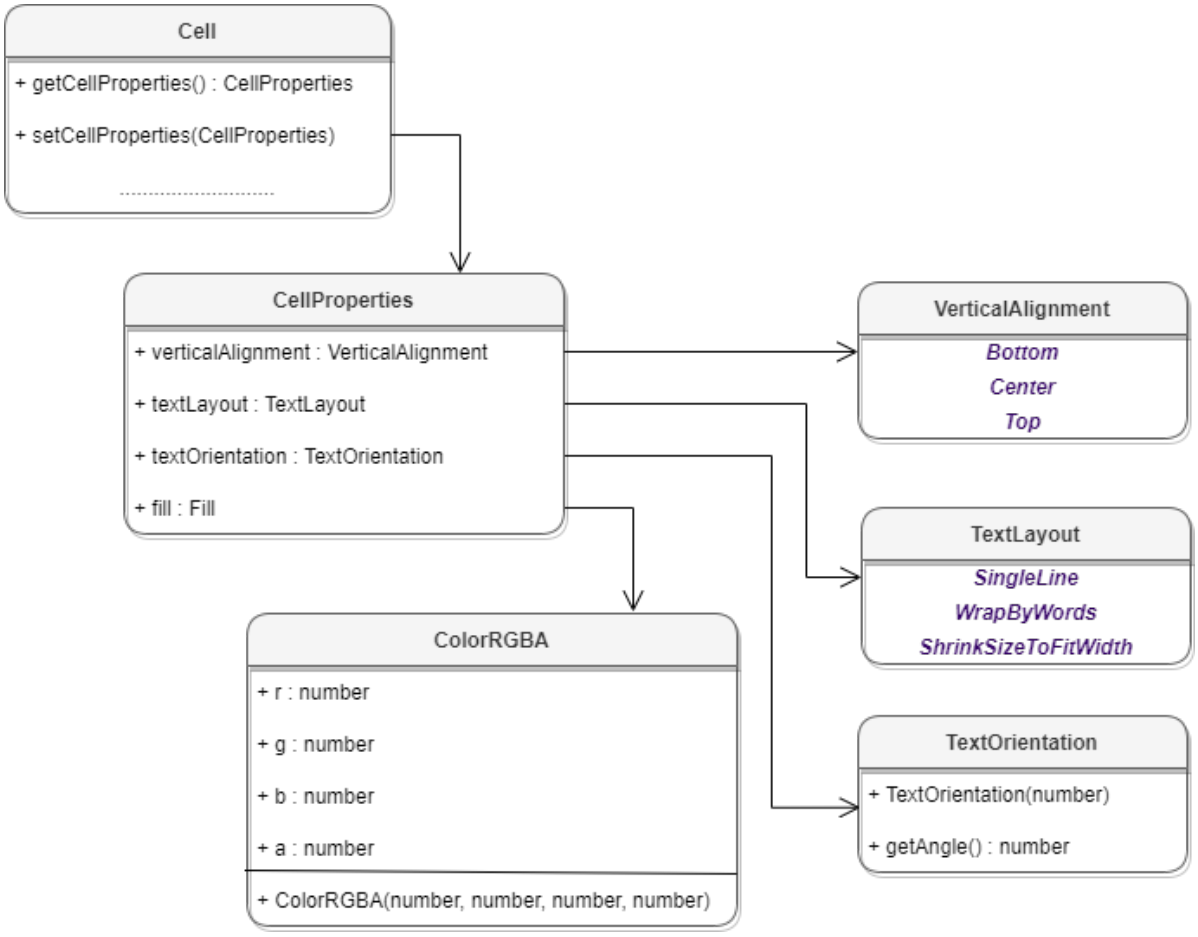


Рисунок 18 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 9 – Описание полей класса `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:

```
cellProps = cell.getCellProperties()
cellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center
cellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth
cellProps.fill = myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255,
255, 0, 255)))
cellProps.textOrientation = myOfficeSDK.TextOrientation(45)
```

6.14.1 CellProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `CellProperties`.

Пример:

```
firstCellProps = myOfficeSDK.CellProperties()
firstCellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center
firstCellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth
firstCellProps.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))
firstCellProps.textOrientation = myOfficeSDK.TextOrientation(45)

secondCellProps = myOfficeSDK.CellProperties()
secondCellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center
secondCellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth
secondCellProps.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))
secondCellProps.textOrientation = myOfficeSDK.TextOrientation(45)

if firstCellProps.__eq__(secondCellProps):
    print("Equals")
```

6.14.2 CellProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `CellProperties`.

Пример:

```
firstCellProps = myOfficeSDK.CellProperties()
firstCellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center
firstCellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth
firstCellProps.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))
```

```
firstCellProps.textOrientation = myOfficeSDK.TextOrientation(45)

secondCellProps = myOfficeSDK.CellProperties()
secondCellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center
secondCellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth
secondCellProps.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))
secondCellProps.textOrientation = myOfficeSDK.TextOrientation(45)

if firstCellProps.__ne__(secondCellProps):
    print("Not equals")
```

6.15 Класс CellProtectionProperties

Класс `CellProtectionProperties` предназначен для настройки параметров защиты ячеек в табличном документе (аналог раздела «Свойства ячеек» в меню «Управление защитой»). Данный класс используется в методах `Cell.setProtectionProperties()`, `Cell.getProtectionProperties()`, `CellRange.setProtectionProperties()` и `CellRange.getProtectionProperties()`.

Таблица 10 – Описание полей класса `CellProtectionProperties`

Поле	Значение по умолчанию	Описание
`CellProtectionProperties.lockedForChanges`	`True`	Запретить редактирование значения ячейки
`CellProtectionProperties.formulasNotDisplayed`	`False`	Отображать в строке формул только результат

Пример:

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

cellProps = cell.getProtectionProperties()
cellProps.lockedForChanges = False
cellProps.formulasNotDisplayed = False

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

6.16 Класс CellRange

Класс `CellRange` описывает диапазон ячеек таблицы.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
```

6.16.1 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.).

Метод вызывает исключение `OutOfRangeError` в случае, если источник или место назначения находятся за пределами таблицы.

Пример:

```
firstSheet = document.getBlocks().getTable("List1")
cellRange = firstSheet.getCellRange("A1:A2")
cellRange.autoFill(myOfficeSDK.CellPosition(2, 0))
```

6.16.2 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:

```
auto cellRange = sheetList.getCellRange("A1:C4");
auto cell = sheetList.getCell("A1");
print(cellRange.containsCell(cell))
```

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

6.16.3 Метод CellRange.copyInto

Метод позволяет копировать (аналог **Ctrl+C**, **Ctrl+V** в редакторе таблиц) ячейки текущего диапазона в заданную позицию, представленную параметром типа `CellRange`. Вы можете копировать ячейки в пределах одного листа, а также между листами и документами.

Данный метод реализован только в табличных документах.

Пример (только для табличного документа):

```
sourceRange = table.getCellRange("A1:A2")
destRange = table.getCellRange("A2:A3")
sourceRange.copyInto(destRange)
```

Пример копирования ячеек между листами:

```
document = application.loadDocument("sheet.xods")

sheetList1 = document.getBlocks().getTable(0)
sheetList2 = document.getBlocks().getTable(1)

sourceRange = sheetList1.getCellRange("A1:C3")
destRange = sheetList2.getCellRange("A1:C3")

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 19).

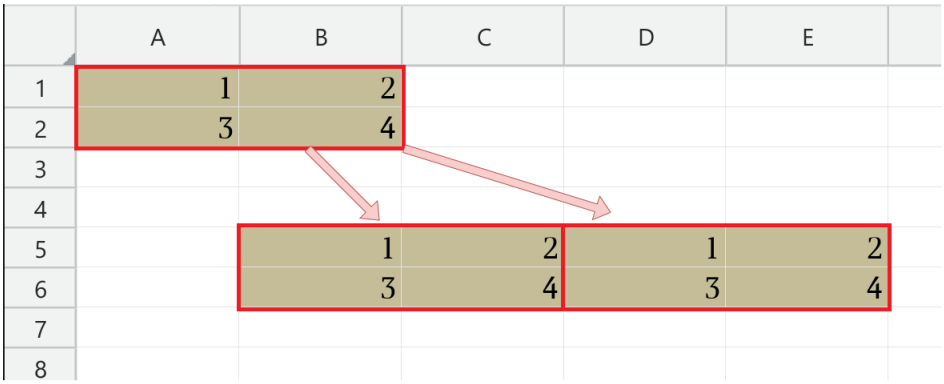


Рисунок 19 – Копирование ячеек табличного документа

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

6.16.4 Метод CellRange.getBeginColumn

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
print(cellRange.getBeginColumn())
```

6.16.5 Метод CellRange.getBeginRow

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
print(cellRange.getBeginRow())
```

6.16.6 Метод CellRange.getCellProperties

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
```

```
cellProperties = cellRange.getCellProperties()
print(cellProperties.fill.getColor().getRGBAColor().r)
```

6.16.7 Метод CellRange.getEnumerator

Метод возвращает коллекцию ячеек в диапазоне.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
cellEnumerator = cellRange.getEnumerator()
for cell in cellEnumerator:
    print(cell.getFormattedValue())
```

6.16.8 Метод CellRange.getLastColumn

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
print(cellRange.getLastColumn())
```

6.16.9 Метод CellRange.getLastRow

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
print(cellRange.getLastRow())
```

6.16.10 Метод CellRange.getProtectionProperties

Метод возвращает параметры защиты диапазона ячеек табличного документа.

Вызов:

```
CellProtectionProperties getProtectionProperties()
```

Возвращает:

– `CellProtectionProperties`: свойства защиты диапазона ячеек (`None`, если диапазон содержит только ячейки сводной таблицы).

Пример:

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

cellProps = cellRange.getProtectionProperties()
cellProps.lockedForChanges = False
cellProps.formulasNotDisplayed = False

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

Свойства возвращаемого объекта `CellProtectionProperties` могут быть `None`, если текущий диапазон содержит ячейки с разными параметрами. Метод `setProtectionProperties()` позволяет задать параметры защиты диапазона ячеек. Вы также можете использовать метод `isProtected()`, чтобы узнать, защищены ли ячейки диапазона от редактирования.

6.16.11 Метод CellRange.getTable

Метод возвращает таблицу (`Table`) для диапазона ячеек.

Пример:

```
cellRange = firstSheet.getCellRange("A1:A2")
rangeTable = cellRange.getTable()
print(rangeTable.getName())
```

6.16.12 Метод CellRange.getTableRange

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

6.16.13 Метод CellRange.insertCurrentDateTime

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

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("A1")
cellRange.insertCurrentDateTime(myOfficeSDK.DateTimeFormat_DateTime)
```

6.16.14 Метод CellRange.isProtected

Метод возвращает статус защиты от редактирования диапазона ячеек в табличном документе.

Вызов:

```
bool isProtected()
```

Метод `isProtected()` возвращает `None`, если часть ячеек диапазона защищена от редактирования, а часть – нет. Методы `setProtectionProperties()` и `getProtectionProperties()` позволяют задать и получить параметры защиты диапазона ячеек (`CellProtectionProperties`).

6.16.15 Метод CellRange.merge

Метод объединяет несколько ячеек таблицы в одну. Группа ячеек (диапазон) формируется с помощью объекта `CellRange`. Содержимое крайней левой ячейки диапазона помещается в объединенной ячейке.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
cellRange.merge()
```

6.16.16 Метод CellRange.moveInto

Метод позволяет переносить (аналог **Ctrl+X**, **Ctrl+V** в редакторе таблиц) ячейки текущего диапазона в заданную позицию, представленную параметром типа `CellRange`. Вы можете переносить ячейки в пределах одного листа, а также между листами и документами.

Данный метод реализован только в табличных документах.

Пример (только для табличного документа):

```
sourceRange = table.getCellRange("A1:A2")
destRange = table.getCellRange("A2:A3")
sourceRange.moveInto(destRange)
```

Пример перемещения ячеек между документами:

```
document1 = application.loadDocument("sheet1.xods")
document2 = application.loadDocument("sheet2.xods")

sheetList1 = document1.getBlocks().getTable(0)
sheetList2 = document2.getBlocks().getTable(0)
```

```
sourceRange = sheetList1.getCellRange("A1:C3")
destRange = sheetList2.getCellRange("A1:C3")

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.

6.16.17 Метод CellRange.setBorders

Метод предназначен для установки границ диапазона ячеек. Отдельные границы устанавливаются с помощью методов класса `Borders`.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A1")

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

newBorders = myOfficeSDK.Borders()
newBorders.setLeft(lineProperties)
newBorders.setRight(lineProperties)
newBorders.setTop(lineProperties)
newBorders.setBottom(lineProperties)

cell.setBorders(newBorders)
```

6.16.18 Метод CellRange.setCellProperties

Метод предназначен для установки свойств `CellProperties` ячеек диапазона.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
```

```
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))
cellRange.setCellProperties(cellProperties)
```

6.16.19 Метод CellRange.setProtectionProperties

Метод задает параметры защиты диапазона ячеек в табличном документе.

Вызов:

```
setProtectionProperties(protectionProps)
```

Параметры:

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

Пример:

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

cellProps = cellRange.getProtectionProperties()
cellProps.lockedForChanges = False
cellProps.formulasNotDisplayed = False

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

Метод `getProtectionProperties()` позволяет получить текущие параметры защиты ячеек. Вы также можете использовать метод `isProtected()`, чтобы узнать, защищены ли ячейки диапазона от редактирования.

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.

6.17 Класс CellRangePosition

Класс `CellRangePosition` представляет положение диапазона ячеек в таблице. Используется в качестве поля `tableRange` класса `TableRangeInfo`, а также в методах `Table.getCellRange()`, `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.

Описание полей класса `CellRangePosition` представлено в таблице 11.

Таблица 11 – Поля класса `CellRangePosition`

Поле	Тип	Описание
`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.
`bottomRight`	`CellPosition`	Содержит позицию правой нижней ячейки таблицы прямоугольного диапазона.

Примеры:

```
table = document.getBlocks().getTable(0)
cellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 1, 1)
cellRange = table.getCellRange(cellRangePosition)
```

```
table = document.getBlocks().getTable(0)
charts = table.getCharts()
rangeInfo = charts.getChart(0).getRange(0)
cellRangePosition = rangeInfo.tableRangeInfo
tableRange = cellRangePosition.tableRange
print("top left row:", tableRange.topLeft.row, ", top left column:",
tableRange.topLeft.column)
```

6.17.1 Метод CellRangePosition.toString

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

Пример:

```
table = document.getBlocks().getTable(0)
charts = table.getCharts()
rangeInfo = charts.getChart(0).getRange(0)
```

```
cellRangePosition = rangeInfo.tableRangeInfo
tableRange = cellRangePosition.tableRange
print(tableRange.toString())
```

6.17.2 CellRangePosition.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `CellRangePosition`.

Пример:

```
firstCellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 1, 1)
secondCellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 1, 1)

if firstCellRangePosition.__eq__(secondCellRangePosition):
    print("Equals")
```

6.17.3 CellRangePosition.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `CellRangePosition`.

Пример:

```
firstCellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 1, 1)
secondCellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 1, 2)

if firstCellRangePosition.__ne__(secondCellRangePosition):
    print("Not equals")
```

6.18 Класс Chart

Класс `Chart` представляет диаграмму в табличном документе и описывает все ее элементы (заголовок, легенда, тип, данные, диапазон и т.д). Объектная модель `Chart` приведена на рисунке 20.

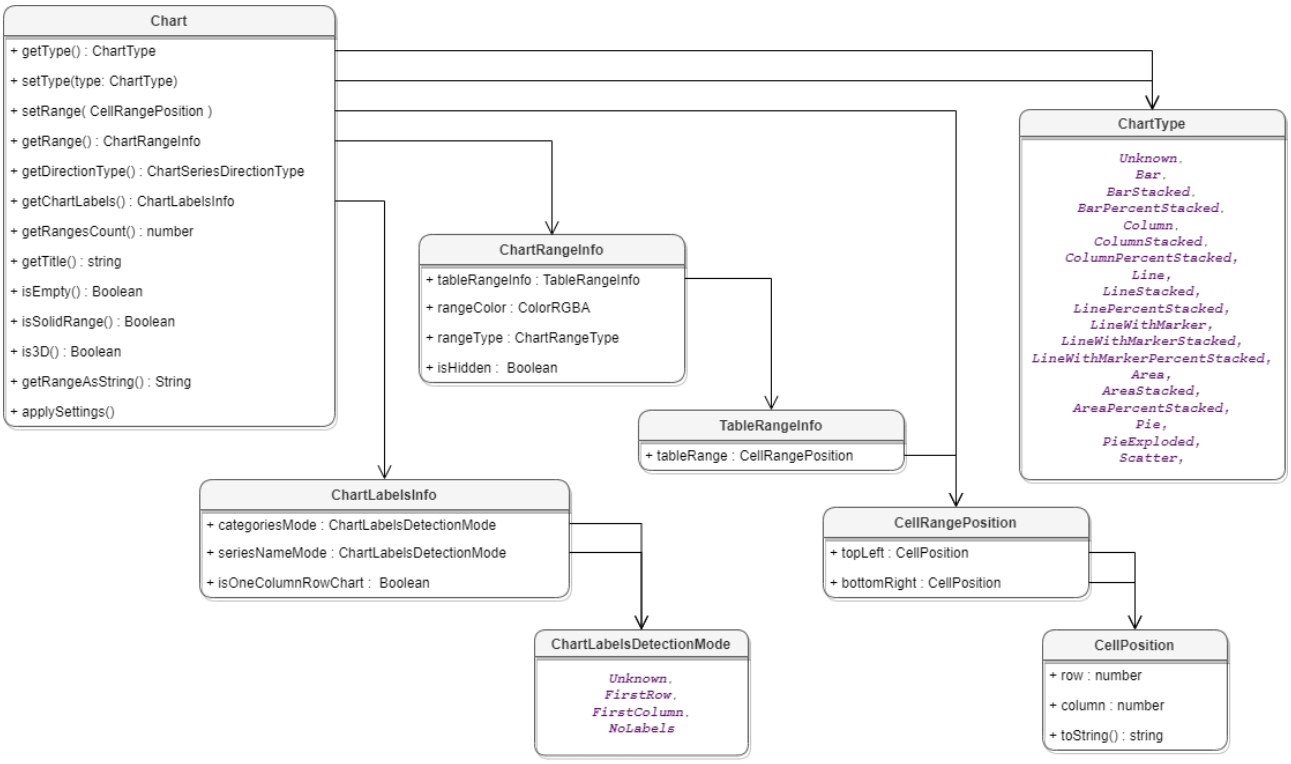


Рисунок 20 – Объектная модель класса `Chart`

6.18.1 Метод Chart.applySettings

Метод позволяет обновить параметры текущей выбранной диаграммы.

Вызов:

```
applySettings(cellRange, directionType, title, labelsInfo)
```

Параметры:

– `cellRange` – обновленный диапазон исходных данных диаграммы `CellRange`;

– `directionType` – направление серий `ChartSeriesDirectionType`;

– `title` – заголовок диаграммы (тип - строка);

– `labelsInfo` – информация о метках диаграммы `ChartLabelsInfo`.

Пример:

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
cellRange = firstSheet.getCellRange("B3:C4")
chartLabelsInfo =
myOfficeSDK.ChartLabelsInfo(myOfficeSDK.ChartLabelsDetectionMode_FirstColumn,
myOfficeSDK.ChartLabelsDetectionMode_FirstRow, False);
chart.applySettings(cellRange, None, "Title", chartLabelsInfo)
```

6.18.2 Метод Chart.getChartLabels

Метод возвращает коллекцию меток диаграммы типа `ChartLabelsInfo`.

Пример:

```
chart = charts.getChart(0)
chartLabelsInfo = chart.getChartLabels();
print(chartLabelsInfo.categoriesMode, chartLabelsInfo.seriesNameMode,
chartLabelsInfo.isOneColumnRowChart)
```

6.18.3 Метод Chart.getDirectionType

Метод возвращает направление `ChartSeriesDirectionType` серий диаграммы.

Пример:

```
chart = charts.getChart(0)
print(chart.getDirectionType())
```

6.18.4 Метод Chart.getFrame

Метод аналогичен методу `MediaObject.getFrame()`, он возвращает свойства позиции диаграммы. Табличные документы работают с абсолютной позицией и используют тип `AbsoluteFrame`.

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
for mediaObject in mediaObjects.getEnumerator():
  chart = mediaObject.toChart()
  if chart != None:
    print(chart.getFrame())
```

6.18.5 Метод Chart.getRange

Метод возвращает диапазон ячеек `ChartRangeInfo` с исходными данными диаграммы. Параметр `rangesIndex` – индекс диапазона.

Пример:

```
chart = charts.getChart(0)
print(chart.getRange(0).rangeType)
```

6.18.6 Метод Chart.getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

Пример:

```
chart = charts.getChart(0)
print(chart.getRangeAsString())
```

6.18.7 Метод Chart.getRangesCount

Метод возвращает количество серий диаграммы.

Пример:

```
chart = charts.getChart(0)
print(chart.getRangesCount())
```

6.18.8 Метод Chart.getTitle

Метод возвращает заголовок диаграммы.

Пример:

```
chart = charts.getChart(0)
print(chart.getTitle())
```

6.18.9 Метод Chart.getType

Метод возвращает тип диаграммы `ChartType`.

Пример:

```
chart = charts.getChart(0)
print(chart.getType())
```

6.18.10 Метод Chart.is3D

Метод возвращает `True`, если диаграмма трехмерная.

Пример:

```
chart = charts.getChart(0)
print(chart.is3D())
```

6.18.11 Метод Chart.isEmpty

Метод возвращает `True`, если диаграмма не содержит значений.

Пример:

```
chart = charts.getChart(0)
print(chart.isEmpty())
```

6.18.12 Метод Chart.isSolidRange

Метод возвращает `True`, если диапазон исходных данных диаграммы может быть выделен одним прямоугольником и не имеет промежутков.

Пример:

```
chart = charts.getChart(0)
print(chart.isSolidRange())
```

6.18.13 Метод Chart.setRange

Метод задает диапазон `CellRangePosition` ячеек с исходными данными для диаграммы.

Пример:

```
chart = charts.getChart(0)
cellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 5, 5)
chart.setRange(cellRangePosition)
print(chart.getRangeAsString())
```

6.18.14 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.

Пример:

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
chart.setRect(myOfficeSDK.RectU(0.0, 0.0, 100.0, 100.0))
```

6.18.15 Метод Chart.setType

Метод устанавливает тип диаграммы `ChartType`. В качестве параметра передается новый тип диаграммы.

Пример:

```
chart = charts.getChart(0)
chart.setType(myOfficeSDK.ChartType_Area)
print(chart.getType())
```

6.19 Класс ChartLabelsDetectionMode

Тип описывает режимы автоматического определения меток диаграмм (см. таблицу 12).

Таблица 12 - Режимы автоматического определения меток диаграмм

Имя константы типа диапазона исходных данных диаграммы	Описание
ChartLabelsDetectionMode_Unknown	Неопределенный тип
ChartLabelsDetectionMode_FirstRow	Метка на первой строке
ChartLabelsDetectionMode_FirstColumn	Метка на первой колонке
ChartLabelsDetectionMode_NoLabels	Не отрисовывать метки

Пример:

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
chartLabels = chart.getChartLabels()
print(chartLabels.categoriesMode, chartLabels.seriesNameMode)
```

6.20 Класс ChartLabelsInfo

Класс ChartLabelsInfo описывает настройки автоматического определения меток диаграммы. Инициализируется конструктором:

```
ChartLabelsInfo(ChartLabelsDetectionMode categoriesMode,
ChartLabelsDetectionMode seriesNameMode, bool oneColumnRow)
```

Параметры конструктора:

- categoriesMode - режим автоматического определения меток для категорий, тип ChartLabelsDetectionMode;
- seriesNameMode - режим автоматического определения меток для серий, тип ChartLabelsDetectionMode;
- oneColumnRow - передается True, если диапазон диаграммы содержит только одну строку или одну колонку.

Описание полей класса ChartLabelsInfo представлено в таблице 13.

Таблица 13 – Описание полей класса `ChartLabelsInfo`

Поле	Описание	Тип
`categoriesMode`	Режим автоматического определения меток для категорий.	`ChartLabelsDetectionMode`
`seriesNameMode`	Режим автоматического определения меток для серий.	`ChartLabelsDetectionMode`
`isOneColumnRowChart`	Поле содержит `True`, если диапазон диаграммы содержит только одну строку или одну колонку.	Boolean

Примеры:

```
chartLabelsInfo =
myOfficeSDK.ChartLabelsInfo(myOfficeSDK.ChartLabelsDetectionMode_FirstRow,
myOfficeSDK.ChartLabelsDetectionMode_NoLabels, False)
```

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
chartLabelsInfo = chart.getChartLabels()
print(chartLabelsInfo.categoriesMode, chartLabelsInfo.seriesNameMode,
chartLabelsInfo.isOneColumnRowChart)
```

6.21 Класс ChartRangeInfo

Класс `ChartRangeInfo` описывает серию диаграммы. Инициализируется конструктором:

```
    ChartRangeInfo(CellRange cellRange, ColorRGBA color, bool hidden,
ChartRangeType type)
```

Параметры конструктора:

– `tableRangeInfo` - диапазон ячеек, тип `TableRangeInfo`;

– `color` – цвет серии диаграммы, тип `ColorRGBA`;

– `hidden` – видимость серии, тип Boolean;

– `rangeType` – тип диапазона исходных данных диаграммы, тип `ChartRangeType`.

Описание полей класса представлено в таблице 14.

Таблица 14 – Описание полей класса `ChartRangeInfo`

Поле	Описание	Тип
`tableRangeInfo`	Исходный диапазон ячеек для серии.	`TableRangeInfo`
`rangeColor`	Цвет для отрисовки серии.	`ColorRGBA`
`isHidden`	Задает видимость серии диаграммы.	`bool`
`rangeType`	Тип диапазона диаграммы .	`ChartRangeType`

Пример:

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
chartRangeInfo = chart.getRange(0);
print(chartRangeInfo.tableRangeInfo.tableRange.toString(),
      chartRangeInfo.rangeColor.a, chartRangeInfo.rangeColor.b,
chartRangeInfo.rangeColor.g,
      chartRangeInfo.isHidden, chartRangeInfo.rangeType);
```

6.22 Класс ChartRangeType

Класс описывает тип диапазона исходных данных диаграммы (см. таблицу 15).

Таблица 15 - Направление серий диаграмм

Имя константы типа диапазона исходных данных диаграммы	Описание
`ChartRangeType_Series`	Серии
`ChartRangeType_SeriesName`	Имена серий
`ChartRangeType_Categories`	Области
`ChartRangeType_DataPoint`	Разметка данных

Пример:

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
```

```
chartRangeInfo = chart.getRange(0)
rangeTypes = [ "Series", "SeriesName", "Categories", "DataPoint" ]
print(rangeTypes[chartRangeInfo.rangeType]);
```

6.23 Класс Charts

Класс `Charts` обеспечивает доступ к списку диаграмм (см. Рисунок 21) табличного документа. Доступ к списку диаграмм осуществляется с помощью метода `Table.getCharts()`.

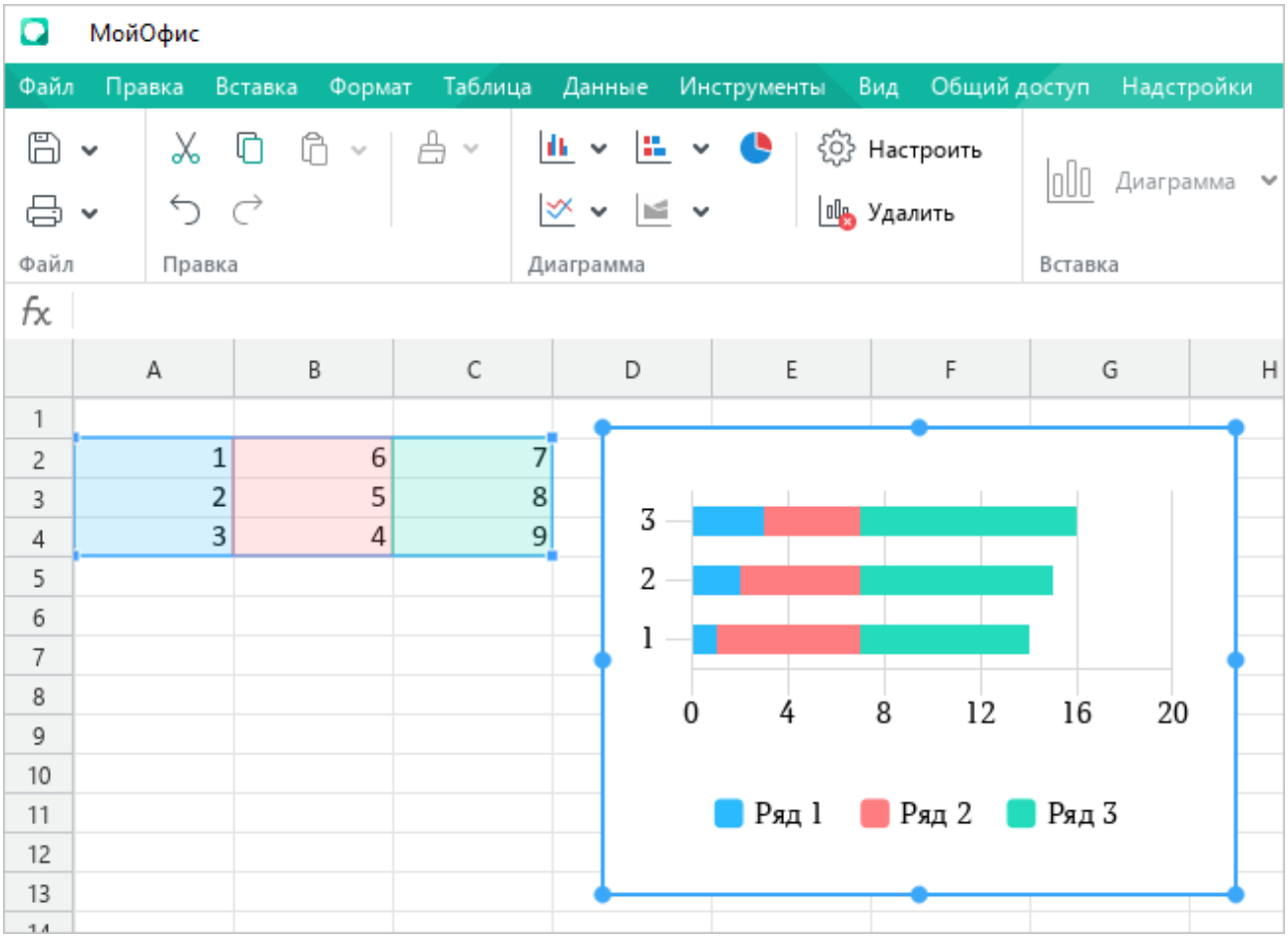


Рисунок 21 – Пример отображения диаграммы в МойОфис Таблица.

6.23.1 Метод Charts.getChart

Метод возвращает диаграмму `Chart` по индексу `chartIndex` в коллекции диаграмм.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
charts = firstSheet.getCharts()
chart = charts.getChart(0)
print(chart.getRangeAsString())
```

6.23.2 Метод Charts.getChartIndexByDrawingIndex

Метод возвращает индекс диаграммы по индексу отрисовки `drawingIndex`.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
charts = firstSheet.getCharts()
chartIndexByDrawingIndex = charts.getChartIndexByDrawingIndex(0)
print(chartIndexByDrawingIndex)
```

6.23.3 Метод Charts.getChartsCount

Метод возвращает общее количество диаграмм в табличном документе.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
charts = firstSheet.getCharts()
print(charts.getChartsCount())
```

6.24 Класс ChartSeriesDirectionType

Тип описывает направление серий диаграмм (см. таблицу 16).

Таблица 16 - Направление серий диаграмм

Имя константы направления серии диаграммы	Описание
`ChartSeriesDirectionType_Unknown`	Неопределенный тип
`ChartSeriesDirectionType_ByRow`	Серии направлены по строкам
`ChartSeriesDirectionType_ByColumn`	Серии направлены по колонкам

Пример:

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
chartSeriesDirectionType = chart.getDirectionType()
directionTypes = [ "Unknown", "ByRow", "ByColumn" ]
print(directionTypes[chartSeriesDirectionType])
```

6.25 Класс ChartType

Перечисление `ChartType` описывает все поддерживаемые типы диаграмм (см. таблицу 17)

Таблица 17 - Типы диаграмм

Имя константы типа диаграммы	Описание
ChartType_Unknown	Неопределенный тип
ChartType_Bar	Линейчатая диаграмма с группировкой
ChartType_BarStacked	Линейчатая диаграмма с накоплением
ChartType_BarPercentStacked	Линейчатая нормированная диаграмма с накоплением
ChartType_Column	Гистограмма с группировкой
ChartType_ColumnStacked	Гистограмма с накоплением
ChartType_ColumnPercentStacked	Нормированная гистограмма с накоплением
ChartType_Line	Стандартный график
ChartType_LineStacked	График с накоплением
ChartType_LinePercentStacked	Нормированный график с накоплением
ChartType_LineWithMarker	Стандартный график с маркерами
ChartType_LineWithMarkerStacked	График с накоплением и маркерами
ChartType_LineWithMarkerPercentStacked	Нормированный график с накоплением и маркерами
ChartType_Area	Стандартная диаграмма с областями
ChartType_AreaStacked	Диаграмма с областями с накоплением
ChartType_AreaPercentStacked	Нормированная диаграмма с областями с накоплением
ChartType_Pie	Круговая диаграмма
ChartType_PieExploded	Круговая диаграмма с отделенными секторами
ChartType_Scatter	Диаграмма рассеяния

Пример:

```
charts = firstSheet.getCharts()
chart = charts.getChart(0)
print(chart.getType())
```

6.26 Класс CheckBoxControl

Представляет собой флаговую кнопку (флажок) в документе. Является наследником класса ContentControl. Методы `CheckBoxControl.getValue()` и `CheckBoxControl.setValue(bool)` позволяют получить и задать значение этого элемента управления.

Пример:

```
controls = document.getContentControls()
checkBox = controls.findByTitle("check1").toCheckBox()
checkBox.setValue(not checkBox.getValue())
```

6.27 Класс Color

Класс `Color` представляет либо цветовой объект RGBA, либо заданные цвета идентификатора темы. В качестве параметров конструктора используются объекты ColorRGBA, ThemeColorID.

Пример:

```
rgbaColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
themeColor = myOfficeSDK.Color(myOfficeSDK.ThemeColorID_Text1)
```

6.27.1 Метод Color.getRGBAColor

Метод возвращает цвет ColorRGBA.

Пример:

```
color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
rgbaColor = color.getRGBAColor()
if rgbaColor != None:
    print(rgbaColor.r)
```

6.27.2 Метод Color.getThemeColorID

Метод возвращает цвет идентификатора темы `ThemeColorID`.

Пример:

```
color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
themeColorId = color.getThemeColorID()
if themeColorId != None:
    print(themeColorId.Value)
```

6.27.3 Метод Color.getTransforms

Метод возвращает текущую трансформацию цвета `ColorTransforms`.

6.27.4 Метод Color.setTransforms

Метод устанавливает трансформацию цвета `ColorTransforms`.

Пример:

```
color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
colorTransforms = myOfficeSDK.ColorTransforms()
colorTransforms.apply(myOfficeSDK.ColorRGBA(55, 146, 179, 200))
color.setTransforms(colorTransforms)
```

6.27.5 Метод Color.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Color`.

Пример:

```
firstColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
secondColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))

if firstColor.__eq__(secondColor):
    print("Equals")
```

6.27.6 Метод Color.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Color`.

Пример:

```
firstColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
secondColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 0))
```

```
if firstColor.__ne__(secondColor):
    print("Not equals")
```

6.28 Класс ColorRGBA

Класс `DocumentAPI.ColorRGBA` предназначен 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`).

Для создания нового объекта используется один из конструкторов:

```
myOfficeSDK.ColorRGBA()
myOfficeSDK.ColorRGBA(r: number, g: number, b: number, a: number)
```

Описание полей таблицы `DocumentAPI.ColorRGBA` представлено в таблице 18.

Таблица 18 – Описание полей класса `ColorRGBA`

Поле	Тип	Описание
r	number	Значение от 0 до 255 для установки интенсивности красного цвета
g	number	Значение от 0 до 255 для установки интенсивности зеленого цвета
b	number	Значение от 0 до 255 для установки интенсивности голубого цвета
a	number	Значение от 0 до 255 для регулировки прозрачности. Значение 255 соответствует полностью непрозрачному цвету

Примеры использования:

```
rgba = myOfficeSDK.ColorRGBA()
rgba.r = 0
rgba.g = 0
rgba.b = 255
rgba.a = 200
# r=0, g=0, b=255, a=200
print("r=", rgba.r, ", g=", rgba.g, ", b=", rgba.b, ", a=", rgba.a)
```

```
rgba = myOfficeSDK.ColorRGBA(55, 146, 179, 200)
# r=55, g=146, b=179, a=200
print("r=", rgba.r, ", g=", rgba.g, ", b=", rgba.b, ", a=", rgba.a)
```

```
line_prop = myOfficeSDK.LineProperties()
line_prop.color = myOfficeSDK.Color(rgba)
```

6.28.1 ColorRGBA.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `ColorRGBA`.

Пример:

```
firstColor = myOfficeSDK.ColorRGBA(100, 100, 100, 0)
secondColor = myOfficeSDK.ColorRGBA(100, 100, 100, 0)

if firstColor.__eq__(secondColor):
    print("Equals")
```

6.28.2 ColorRGBA.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `ColorRGBA`.

Пример:

```
firstColor = myOfficeSDK.ColorRGBA(100, 100, 100, 0)
secondColor = myOfficeSDK.ColorRGBA(100, 100, 100, 255)

if firstColor.__ne__(secondColor):
    print("Not equals")
```

6.29 Класс ColorTransforms

Класс `ColorTransforms` позволяет задать трансформацию цвета для объекта `Color` (см. метод `Color.setTransforms`). Класс обладает пустым конструктором и методом установки цвета трансформации `ColorTransforms.apply`;

Пример:

```
color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
colorTransforms = myOfficeSDK.ColorTransforms()
colorTransforms.apply(myOfficeSDK.ColorRGBA(55, 146, 179, 200))
color.setTransforms(colorTransforms)
```

6.29.1 Метод ColorTransforms.apply

Метод устанавливает цвет трансформации `ColorRGBA`.

Пример:

```
colorTransforms = myOfficeSDK.ColorTransforms()
colorTransforms.apply(myOfficeSDK.ColorRGBA(55, 146, 179, 200))
```

6.30 Класс Comment

Класс `Comment` предоставляет доступ к следующим свойствам комментария:

- диапазон текста `Range`, который описывает комментарий;
- текст комментария;
- информация о комментарии `TrackedChangeInfo`;
- признак того, что комментарий принят;
- список ответов на комментарий `Comments`.

6.30.1 Метод Comment.getAudioUrl

Метод возвращает путь к аудиофайлу, тип `string`.

Пример:

```
comments = document.getRange().getComments()
enumerator = comments.getEnumerator()
for comment in enumerator:
    print(comment.getAudioUrl())
```

6.30.2 Метод Comment.getInfo

Метод предоставляет доступ к информации о комментарии `TrackedChangeInfo` (автор изменения, дата и т. д).

Пример:

```
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    trackedChangeInfo = comment.getInfo()
    print(trackedChangeInfo.author.name)
```

6.30.3 Метод Comment.getRange

Метод возвращает диапазон документа `Range`, которому соответствует комментарий.

Пример:

```
comments = document.getRange().getComments()
enumerator = comments.getEnumerator()
for comment in enumerator:
    commentRange = comment.getRange()
    print(commentRange.extractText())
```

6.30.4 Метод Comment.getReplies

Метод предоставляет доступ к ответам на комментарии. Ответы находятся в такой же таблице `Comments`, как и сами комментарии документа.

Пример:

```
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    replies = comment.getReplies()
    repliesEnumerator = replies.getEnumerator()
    for reply in repliesEnumerator:
        print(reply.extractText())
```

6.30.5 Метод Comment.getText

Метод возвращает текст комментария.

Пример:

```
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    print(comment.getText())
```

6.30.6 Метод Comment.isResolved

Метод возвращает значение `True`, если комментарий принят.

Пример:

```
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    print(comment.isResolved())
```

6.31 Класс Comments

Класс `Comments` содержит коллекцию комментариев диапазона (см. Рисунок 22).

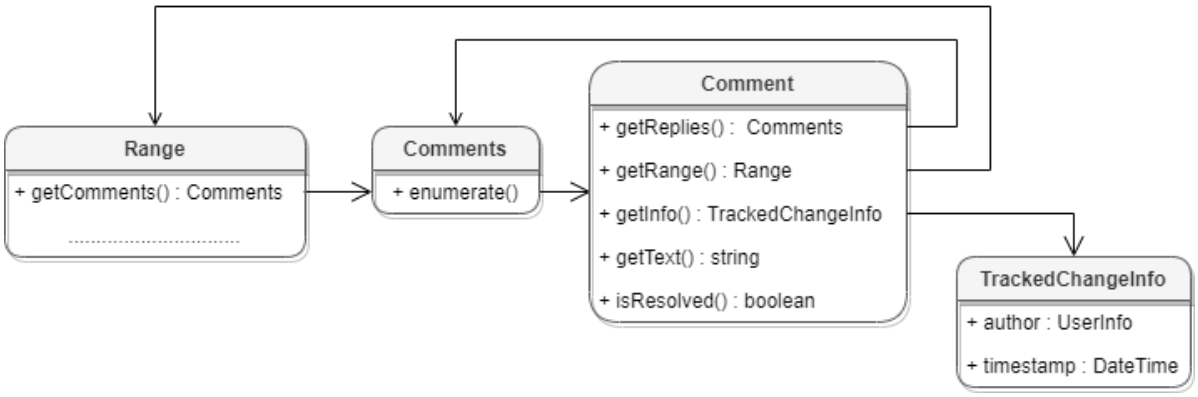


Рисунок 22 – Объектная модель классов для работы с комментариями

Для получения списка комментариев используется метод `Range.getComments()`.

Пример:

```
comments = document.getRange().getComments()
```

6.31.1 Метод Comments.getEnumerator

Метод возвращает коллекцию комментариев всего документа.

Пример:

```
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    trackedChangeInfo = comment.getInfo()
    print(trackedChangeInfo.author.name)
```

6.32 Класс ConditionalTableFilter

Класс `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:

```
ConditionalTableFilter(bool andOperation = False)
```

Параметр:

– `andOperation` – логическая операция фильтра, по умолчанию - OR. В дальнейшем может быть изменена методом `ConditionalTableFilter.setAndOperation`.

Пример использования приведен в разделе Работа с фильтрами.

6.32.1 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:

```
songFilter = myOfficeSDK.ConditionalTableFilter()
songFilter.setAndOperation(True);
```

6.32.2 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.

```
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)
```

Пример:

```
songFilter = myOfficeSDK.ConditionalTableFilter()
songFilter.notEqual("")
songFilter.notBegins("TODO")
```

Пример использования приведен в разделе Работа с фильтрами.

6.33 Класс Connection

Класс `Connection` реализует соединение между `Messenger` и клиентом. Содержит один метод `unsubscribe` для разрыва соединения.

Пример:

```
messageHandler = myOfficeSDK.MessageHandler()
messenger= application.getMessenger()
connection = messenger.subscribe(messageHandler)
.....................................................
connection.unsubscribe()
```

6.34 Класс ContentControl

Базовый класс для элементов управления (см. Работа с элементами управления).

Наследники:

- `CheckBoxControl`
- `DatePickerControl`
- `DropListControl`
- `InputFieldControl`

6.34.1 Метод ContentControl.canEdit

Метод возвращает `True`, если значение элемента управления может быть изменено, и `False` в обратном случае.

6.34.2 Метод ContentControl.getTitle

Метод возвращает название элемента управления.

6.34.3 Методы toCheckBox, toInputField, toDatePicker, toDropList

Методы преобразуют объект `ContentControl` в объект соответствующего типа. Возвращают `None`, если преобразование невозможно.

Пример:

```
controls = document.getContentControls()

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

6.35 Класс 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:

```
controls = document.getContentControls()
textField = controls.findByTitle("input")
```

6.36 Класс CurrencyCellFormatting

Класс содержит параметры для денежного формата ячеек таблицы. Описание полей класса `CurrencyCellFormatting` представлено в таблице 19.

Таблица 19 – Описание полей класса `CurrencyCellFormatting`

Поле	Описание
`CurrencyCellFormatting.decimalPlaces`	Количество десятичных позиций
`CurrencyCellFormatting.symbol`	Символ денежной единицы
`CurrencyCellFormatting.localeCode`	Идентификатор кода языка (MS-LCID)
`CurrencyCellFormatting.useThousandsSeparator`	Использовать разделитель для тысячных

Поле	Описание
`CurrencyCellFormatting.useRedForNegative`	Использовать красный цвет для отрицательных значений
`CurrencyCellFormatting.useBracketsForNegative`	Использовать скобки для отрицательных значений
`CurrencyCellFormatting.hideSign`	Скрывать знак «минус» для отрицательных значений
`CurrencyCellFormatting.currencySignPlacement`	Варианты размещения знака валюты `CurrencySignPlacement`

Экземпляр данного класса используется в качестве аргумента метода `Cell.setFormat()`, см. пример.

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("A2")

cellFormat = myOfficeSDK.CurrencyCellFormatting()
cellFormat.decimalPlaces = 2
cellFormat.useThousandsSeparator = True
cellFormat.useRedForNegative = True
cellFormat.useBracketsForNegative = True
cellFormat.hideSign = False
cellFormat.currencySignPlacement = myOfficeSDK.CurrencySignPlacement_Suffix

cell.setFormat(cellFormat)
print(cell.getFormattedValue())
```

6.37 Класс CurrencySignPlacement

Варианты размещения знака валюты представлены в таблице 20. Данный тип используется в поле `currencyFormat` класса `LocaleInfo`, а также в поле `currencySignPlacement` класса `CurrencyCellFormatting` (см.пример в ее описании).

Таблица 20 – Описание полей класса `CurrencySignPlacement`

Поле	Описание	Пример
`CurrencySignPlacement_Prefix`	Размещение знака валюты до значения	$12.00
`AccountingCellFormatting_Suffix`	Размещение знака валюты после значения	12,00 ₽

6.38 Класс DatePatterns

Форматы даты представлены в таблице 21. Пример использования см. в главе `DateTimeCellFormatting`.

Таблица 21 – Форматы даты

Наименование константы	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'`
`DatePatterns_DayMonthYearLongNonLocalizableSlash`	Нелокализуемый шаблон `'dd/mm/yyyy'`

6.39 Класс DatePickerControl

Представляет собой поле с календарем, используемое для ввода дат в документе. Является наследником класса `ContentControl`. Методы `DatePickerControl.getValue()` и `DatePickerControl.setValue(DateTime)` позволяют получить и задать значение этого элемента управления.

Пример:

```
controls = document.getContentControls()
startDate = controls.findByTitle("startDate").toDatePicker()
endDate = controls.findByTitle("endDate").toDatePicker()
value = startDate.getValue()
value.year += 1
endDate.setValue(value)
```

6.40 Класс DateTime

Класс `DateTime` предоставляет дату и время с точностью до секунды. Используется для поля `TrackedChangeInfo.timeStamp`. Описание полей класса `DateTime` представлено в таблице 22.

Таблица 22 – Описание полей класса `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

6.40.1 DateTime.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `DateTime`.

Пример:

```
firstDateTime = myOfficeSDK.DateTime()
firstDateTime.year = 2020

secondDateTime = myOfficeSDK.DateTime()
secondDateTime.year = 2020

if firstDateTime.__eq__(secondDateTime):
    print("Equals");
```

6.40.2 DateTime.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `DateTime`.

Пример:

```
firstDateTime = myOfficeSDK.DateTime()
firstDateTime.year = 2020

secondDateTime = myOfficeSDK.DateTime()
secondDateTime.year = 2021

if firstDateTime.__ne__(secondDateTime):
    print("Not equals");
```

6.41 Класс DateTimeCellFormatting

Класс содержит параметры для формата ячеек таблицы типа Дата и Время, используется в качестве аргумента метода `Cell.setFormat()`. Описание полей класса `DateTimeCellFormatting` представлено в таблице 23.

Таблица 23 – Описание полей класса `DateTimeCellFormatting`

Поле	Описание
`DateTimeCellFormatting.dateListID`	Формат даты `DatePatterns`
`DateTimeCellFormatting.timeListID`	Формат времени `TimePatterns`

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("A2")

cellFormat = myOfficeSDK.DateTimeCellFormatting()
cellFormat.dateListID = myOfficeSDK.DatePatterns_DayMonthNumberLongYearShort
cellFormat.timeListID = myOfficeSDK.TimePatterns_LongTime

cell.setFormat(cellFormat)
print(cell.getFormattedValue())
```

6.42 Класс DateTimeFormat

В таблице 24 представлены варианты масштабирования при печати табличных документов. Используется в качестве параметра метода `CellRange.insertCurrentDateTime()`.

Таблица 24 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
`DateTimeFormat_DateTime`	Дата/время
`DateTimeFormat_Date`	Дата
`DateTimeFormat_Time`	Время

6.43 Класс Document

Класс `Document` осуществляет доступ к содержимому открытого текстового или табличного документа.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
```

6.43.1 Метод Document.areMirroredMarginsEnabled

Возвращает состояние режима зеркальных полей в документе (разрешены или запрещены).

Пример:

```
print(document.areMirroredMarginsEnabled())
```

6.43.2 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.

6.43.3 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.

6.43.4 Метод Document.exportAs

Метод `Document.exportAs` экспортирует документ в файл по указанному пути с указанным форматом.

Расширенные версии метода позволяют указать дополнительные настройки экспорта документа:

- для текстовых документов – класс `TextExportSettings`;
- для табличных документов – класс `WorkbookExportSettings`;
- для презентационных документов – класс `PresentationExportSettings`.

В настоящее время поддерживается только операция экспорта документа в формат PDF.

Примеры:

```
document.exportAs(filePath, myOfficeSDK_ExportFormat_PDFA1)
```

```
textExportSettings = myOfficeSDK.TextExportSettings()
textExportSettings.pageNumbers =
myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1, textExportSettings)
```

```
workbookSettings = myOfficeSDK.WorkbookExportSettings()
workbookSettings.sheetNames = myOfficeSDK.VectorString()
workbookSettings.sheetNames.push_back("Лист2")
workbookSettings.printingScope =
myOfficeSDK.PrintingScope(myOfficeSDK.PrintingScope.Type_PrintArea)
workbookSettings.pageProperties = myOfficeSDK.PageProperties(100, 200)
workbookSettings.scale = 90
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1, workbookSettings)
```

```
presentationExportSettings = myOfficeSDK.PresentationExportSettings()
presentationExportSettings.skipHiddenSlides = False
```

```
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1,
presentationExportSettings)
```

6.43.5 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:

```
print(document.getAbsoluteFilePath())
```

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.

6.43.6 Метод Document.getBlocks

Метод предоставляет доступ к объекту Blocks и далее к отдельным фрагментам (абзацам, таблицам и т. д.), из которых состоит документ.

Пример:

```
blocks = document.getBlocks()
if blocks != None:
    paragraph = blocks.getParagraph(0)
    if paragraph != None:
        print(paragraph.getListLevel())
```

6.43.7 Метод Document.getBookmarks

Метод предоставляет доступ к списку закладок Bookmarks.

Пример:

```
bookmarks = document.getBookmarks()
if bookmarks != None:
    bookmarksRange = bookmarks.getBookmarkRange("Bookmark")
    bookmarksRange.replaceText("New bookmark text")
    print(bookmarksRange.extractText())
```

6.43.8 Метод Document.getCalculationMode

Метод возвращает текущий режим пересчета формул в документе `CalculationMode`. Чтобы изменить этот режим, используйте метод `Document.setCalculationMode()`.

6.43.9 Метод Document.getComments

Метод обеспечивает доступ к комментариям документа, возвращает объект `Comments`.

Пример:

```
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    print(comment.getText())
```

6.43.10 Метод Document.getContentControls

Метод предоставляет доступ к списку элементов управления `ContentControls`, содержащихся в документе (см. Работа с элементами управления).

Пример:

```
controls = document.getContentControls()
for control in controls:
    print (control.getTitle())
```

6.43.11 Метод Document.getFormulaType

Метод возвращает поддерживаемую адресацию ячеек `FormulaType` документа.

Пример:

```
formulaType = document.getFormulaType()
```

6.43.12 Метод Document.getNamedExpressions

Используется для получения списка именованных диапазонов `NamedExpressions`.

Пример:

```
namedExpressions = document.getNamedExpressions()
```

6.43.13 Метод Document.getPivotTablesManager

Возвращает объект `PivotTablesManager`, который используется для создания сводных таблиц. Метод может быть использован только в табличном редакторе.

Пример:

```
pivotTablesManager = document.getPivotTablesManager()
if pivotTablesManager != None:
    pivotTable = pivotTablesManager.create()
```

6.43.14 Метод Document.getRange

Метод предоставляет доступ ко всему диапазону Range документа.

Пример:

```
docRange = document.getRange()
if docRange != None:
    print(docRange.extractText())
```

6.43.15 Метод Document.getScripts

Метод предоставляет доступ к списку макрокоманд Scripts, содержащихся в документе.

Пример:

```
scripts = document.getScripts()
enumerator = scripts.getEnumerator()
for scriptIndex, script in enumerate(enumerator):
    print(script.getName())
```

6.43.16 Метод Document.getSections

Возвращает объект типа Sections.

Пример:

```
sections = document.getSections()
sectionsEnumerator = sections.getEnumerator()
for section in sectionsEnumerator:
    print(section.getPageOrientation())
```

6.43.17 Метод Document.getSectionsEnumerator

Возвращает список объектов типа Section.

Пример:

```
sectionsEnumerator = document.getSectionsEnumerator()
for section in sectionsEnumerator:
    print(section.getPageOrientation())
```

6.43.18 Метод Document.isCalculatedOnSave

Метод возвращает состояние функции пересчета формул при сохранении документа. Используйте метод `Document.setCalculatedOnSave()`, чтобы включить или отключить эту функцию.

6.43.19 Метод Document.isChangesTrackingEnabled

Метод возвращает текущее состояние отслеживания изменений в документе (`True` - включены).

Пример:

```
print(document.isChangesTrackingEnabled())
```

6.43.20 Метод Document.isStructureProtected

Метод возвращает состояние защиты от изменения структуры табличного документа.

Вызов:

```
bool isStructureProtected()
```

Методы `setStructureProtection()` и `removeStructureProtection()` позволяют установить и снять защиту от изменений структуры.

6.43.21 Метод Document.merge

Метод `Document.merge` сравнивает текущий документ с другим документом, который передается в параметре типа `Document`.

Метод возвращает объект `Document`, содержащий результат сравнения в виде отслеживаемых изменений.

Пример:

```
documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Text
loadSettings = myOfficeSDK.LoadDocumentSettings()
loadSettings.commonDocumentSettings = documentSettings

firstDoc = application.loadDocument("c:\\Tmp\\Sample1.docx", loadSettings)
secondDoc = application.loadDocument("c:\\Tmp\\Sample2.docx", loadSettings)

mergedDoc = firstDoc.merge(secondDoc)
mergedDoc.saveAs("c:\\Tmp\\Sample3.docx")
```

Результат выполнения данного примера (сравнение двух документов, содержащих "1111" и "2222") приведен на рисунке 23.

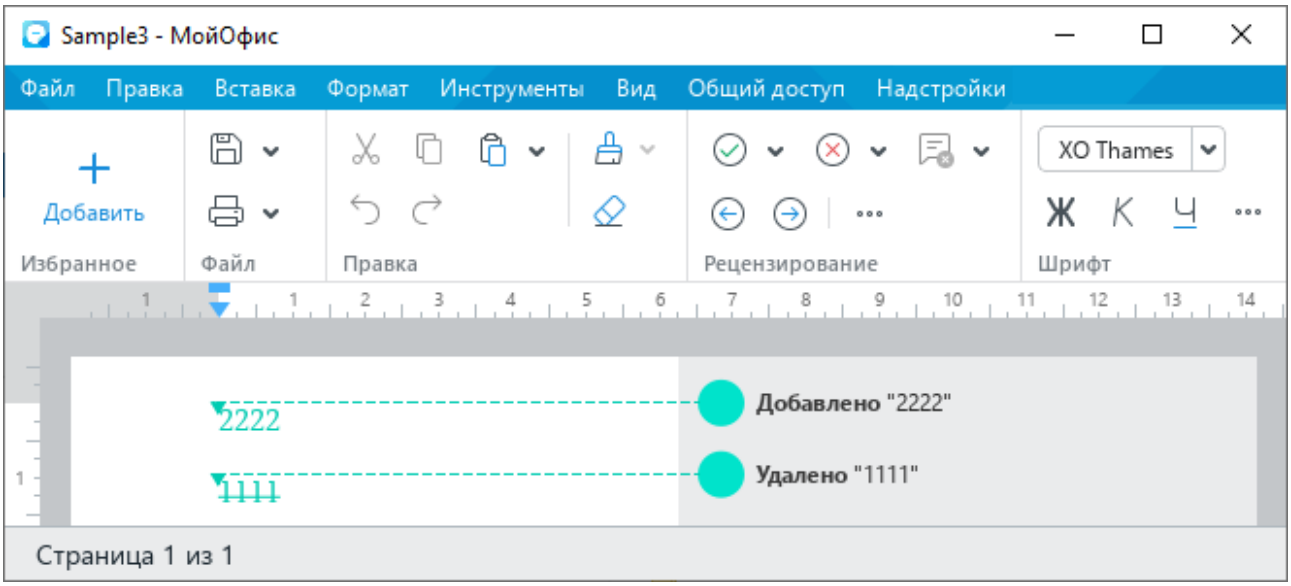


Рисунок 23 – Результат выполнения метода `merge`

6.43.22 Метод Document.removeStructureProtection

Метод снимает защиту от изменений структуры табличного документа.

Вызов:

```
removeStructureProtection(password)
```

Параметры:

– `password`: (необязательный) пароль для снятия защиты, тип `string`.

Пример:

```
document.removeStructureProtection("password")
```

Метод `setStructureProtection()` позволяет установить защиту от изменений структуры. Вы также можете использовать метод `isStructureProtected()`, чтобы узнать текущее состояние защиты. Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

6.43.23 Метод Document.saveAs

Метод `Document.saveAs` сохраняет документ в файл по указанному пути. Формат и тип документа определяются расширением файла, если они не указаны в явном виде.

При необходимости, в качестве второго аргумента можно использовать объект класса `SaveDocumentSettings`, которая содержит формат документа `DocumentFormat`, тип

документа DocumentType и пароль для защиты документа от несанкционированного доступа.

Примеры:

```
document.saveAs(filePath)
```

```
saveDocumentSettings = myOfficeSDK.SaveDocumentSettings()
saveDocumentSettings.documentFormat = myOfficeSDK.DocumentFormat_OXML
saveDocumentSettings.documentType = myOfficeSDK.DocumentType_Workbook
saveDocumentSettings.documentPassword = "password"
saveDocumentSettings.isTemplate = False

saveDocumentSettings.dsvSettings = myOfficeSDK.DSVSettings()
saveDocumentSettings.dsvSettings.autofit = True
saveDocumentSettings.dsvSettings.startBlockIndex = 0
saveDocumentSettings.dsvSettings.lastBlockIndex = 10

document.saveAs(filePath, saveDocumentSettings)
```

6.43.24 Метод Document.setCalculatedOnSave

Метод позволяет включить и отключить функцию пересчета формул при сохранении документа. Используйте метод Document.isCalculatedOnSave(), чтобы узнать текущее состояние этой функции.

6.43.25 Метод Document.setCalculationMode

Метод позволяет задать режим пересчета формул в документе CalculationMode. Используйте метод Document.getCalculationMode(), чтобы получить текущий режим пересчета.

6.43.26 Метод Document.setChangesTrackingEnabled

Метод управляет состоянием отслеживания изменений в документе (включены или выключены).

Пример:

```
document.setChangesTrackingEnabled(True)
```

6.43.27 Метод Document.setFormulaType

Метод устанавливает поддерживаемую адресацию ячеек `FormulaType` документа.

Пример:

```
document.setFormulaType(myOfficeSDK.FormulaType_A1)
```

6.43.28 Метод Document.setMirroredMarginsEnabled

Метод позволяет включать и выключать зеркальные поля в документе.

Пример:

```
document.setMirroredMarginsEnabled(True)
```

6.43.29 Метод Document.setPageOrientation

Метод устанавливает альбомную, либо книжную ориентацию страниц в документе (см. раздел `PageOrientation`).

Пример:

```
document.setPageOrientation(myOfficeSDK.PageOrientation_Landscape)
```

6.43.30 Метод Document.setPageProperties

Метод устанавливает свойство `PageProperties` в документе.

Пример:

```
pageProperties = myOfficeSDK.PageProperties()
pageProperties.width = 100
pageProperties.height = 200
document.setPageProperties(pageProperties)
```

6.43.31 Метод Document.setStructureProtection

Метод устанавливает защиту от изменений структуры табличного документа.

Вызов:

```
setStructureProtection(password)
```

Параметры:

– `password`: (необязательный) пароль для установки защиты, тип `string`.

Пример:

```
document.setStructureProtection("password")
```

Метод `removeStructureProtection()` позволяет снять защиту от изменений структуры. Вы также можете использовать метод `isStructureProtected()`, чтобы узнать текущее состояние защиты. Если метод `setStructureProtection()` применяется к документу с уже защищенной структурой, возникает исключение `SpreadsheetProtectionError`.

6.44 Класс DocumentFormat

В таблице 25 приведены поддерживаемые форматы документов, структура используется в `SaveDocumentSettings`.

Таблица 25 – Форматы документов

Наименование константы	Описание
`DoumentFormat_PlainText`	Используется для работы с файлами TXT.
`DoumentFormat_DSV`	Используется для работы с табличными данными в текстовой форме (CSV, DSV). Строка текста содержит одно или несколько полей данных, разделенных запятыми или иным разделителем.
`DoumentFormat_OXML`	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML.
`DoumentFormat_ODF`	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010).
`DoumentFormat_HTML`	Используется для работы с веб-документами в формате HTML. Работа с веб-документами в формате HTML средствами Document API в настоящий момент не поддерживается.
`DoumentFormat_PDF`	Используется для работы с документами в формате Portable Document Format (PDF) версии 1.4.
`DoumentFormat_PDFA`	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

6.45 Класс DocumentSettings

Класс `DocumentSettings` предоставляет общие настройки документа и используется в `Application.createDocument()`.

Описание полей класса `DocumentSettings` представлено в таблице 26.

Таблица 26 – Описание полей класса `DocumentSettings`

Поле	Тип	Описание
`DocumentSettings_documentType`	`DocumentType`	Тип документа
`DocumentSettings_userInfo`	`UserInfo`	Информация о пользователе
`DocumentSettings_localeInfo`	`LocaleInfo`	Информация о локализации
`DocumentSettings_timeZone`	`TimeZone`	Информация о временной зоне
`DocumentSettings_formulaType`	`FormulaType`	Система адресации ячеек

6.46 Класс DocumentType

В таблице 27 приведены поддерживаемые типы документов, которые используются при создании документа `Application.createDocument()`, `DocumentSettings`.

Таблица 27 - Типы документов

Наименование константы	Описание
`DocumentType_Text`	Используется для работы с текстовыми документами в форматах DOCX, ODT, XODT, TXT.
`DocumentType_Workbook`	Используется для работы с табличными документами в форматах XLSX, ODS, XODS.
`DocumentType_Presentation`	Используется для работы с презентационными документами в форматах PPTX, ODP, XODP.

6.47 Класс DropListControl

Представляет собой поле с выпадающим списком в документе. Является наследником класса `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:

```
controls = document.getContentControls()
comboBox = controls.findByTitle("select").toDropList()
comboBox.setValue(comboBox.getChoices().index("two"))
```

6.48 Класс DSVSettings

Класс `DSVSettings` предоставляет настройки, необходимые для работы с файлами CSV (comma-separated value) и DSV (delimiter-separated value). Используется в `SaveDocumentSettings`, `LoadDocumentSettings`.

Описание полей класса `DSVSettings` представлено в таблице 28.

Таблица 28 – Описание полей класса `DSVSettings`

Поле	Описание
`DSVSettings.autofit`	Признак необходимости автоматического подстраивания ширины столбца под размер данных в ячейке
`DSVSettings.startBlockIndex`	Индекс блока документа сохранения
`DSVSettings.lastBlockIndex`	Индекс блока документа для окончания сохранения

Пример:

```
saveDocumentSettings.dsvSettings = myOfficeSDK.DSVSettings()
saveDocumentSettings.dsvSettings.autofit = True
saveDocumentSettings.dsvSettings.startBlockIndex = 0
saveDocumentSettings.dsvSettings.lastBlockIndex = 10
```

6.48.1 Метод DSVSettings.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `DSVSettings`.

Пример:

```
firstDsvSettings = myOfficeSDK.DSVSettings()
firstDsvSettings.autofit = True
firstDsvSettings.startBlockIndex = 0
firstDsvSettings.lastBlockIndex = 10

secondDsvSettings = myOfficeSDK.DSVSettings()
secondDsvSettings.autofit = True
secondDsvSettings.startBlockIndex = 0
secondDsvSettings.lastBlockIndex = 10

if firstDsvSettings.__eq__(secondDsvSettings):
    print("Equals")
```

6.48.2 Метод DSVSettings.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `DSVSettings`.

Пример:

```
firstDsvSettings = myOfficeSDK.DSVSettings()
firstDsvSettings.autofit = True
firstDsvSettings.startBlockIndex = 0
firstDsvSettings.lastBlockIndex = 10

secondDsvSettings = myOfficeSDK.DSVSettings()
secondDsvSettings.autofit = True
secondDsvSettings.startBlockIndex = 0
secondDsvSettings.lastBlockIndex = 20

if firstDsvSettings.__ne__(secondDsvSettings):
    print("Not equals")
```

6.49 Класс Encoding

В таблице 29 приведены поддерживаемые кодировки документов. Используется в `LoadDocumentSettings`.

Таблица 29 - Кодировки документов

Наименование константы	Кодировка
`Encoding_Unknown`	Невозможно определить кодировку.
`Encoding_UTF8`	UTF8
`Encoding_UTF16BE`	UTF16BE
`Encoding_UTF16LE`	UTF16LE
`Encoding_UTF32BE`	UTF32BE
`Encoding_UTF32LE`	UTF32LE
`Encoding_Windows1250`	Windows1250
`Encoding_Windows1251`	Windows1251
`Encoding_Windows1252`	Windows1252
`Encoding_ISO8859Part5`	ISO8859Part5
`Encoding_KOI8R`	KOI8R
`Encoding_KOI8U`	KOI8U
`Encoding_CP866`	CP866

6.50 Класс ExportFormat

В таблице 30 приведены поддерживаемые форматы экспорта документов, см. `Document.exportAs()`.

Таблица 30 - Форматы экспорта документов

Константа	Описание
`ExportFormat_PDFA1`	Используется для работы с документами в формате Portable Document Format (PDF)

6.51 Класс Field

Класс `Field` предназначен для реализации некоторых полей, например, содержания.

6.52 Класс Fill

Класс описывает свойства заполнения фигуры, используется в `ShapeProperties`, `CellProperties`.

Варианты заполнения:

- без заполнения;
- заполнение цветом;
- фон задается путем к изображению фона.

Примеры:

```
# Без заполнения
cellProps.fill = myOfficeSDK.Fill()
```

```
# Заполнение цветом
cellProps.fill = myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255,
255, 0, 255)))
```

```
# Заполнение шаблоном из url
cellProps.fill = myOfficeSDK.Fill("https://fillpattern.url")
```

6.52.1 Метод Fill.getColor

Метод возвращает цвет заполнения `Color`.

6.52.2 Метод Fill.getUrl

Метод возвращает путь к изображению, которое используется в качестве заполнения, тип - строка.

6.52.3 Метод Fill.isNoFill

Метод возвращает `True`, если заполнения нет.

6.53 Класс FiltersRange

Класс `FiltersRange` реализует диапазон таблицы, позволяющий манипулировать фильтрами столбцов. Пример использования приведен в разделе Работа с фильтрами.

6.53.1 Метод FIltersRange.clear

Метод `FiltersRange.clear()` удаляет автоматически отфильтрованный диапазон и фильтры.

При использовании данного метода может произойти исключение `InvalidObjectError` в случае, если диапазон фильтрации недействителен.

Пример:

```
filtersRange = sheet.createFiltersRange(cellRange)
.....................................
filtersRange.clear()
```

6.53.2 Метод FIltersRange.eraseFilters

Метод `FiltersRange.eraseFilters` удаляет фильтры из диапазона. Диапазон фильтрации остается нетронутым.

При использовании данного метода может произойти исключение `InvalidObjectError` в случае, если диапазон фильтрации недействителен.

Пример:

```
filtersRange = sheet.createFiltersRange(cellRange)
.....................................
filtersRange.eraseFilters()
```

6.53.3 Метод FiltersRange.getCellRange

Метод `FiltersRange.getCellRange` возвращает диапазон ячеек, содержащий текущий объект фильтрации. Возвращаемый диапазон включает строку заголовка.

Пример:

```
cellRange = filtersRange.getCellRange()
print(cellRange.getBeginRow())
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.53.4 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()`.

Вид метода `FiltersRange.setFilters`:

```
void setFilters(TableFilters filters);
```

Метод использует параметр типа `TableFilters`.

При использовании данного метода может произойти исключение `InvalidObjectError` в случае, если диапазон фильтрации недействителен.

Пример:

```
filtersRange = sheet.createFiltersRange(cellRange)
.....................................
tableFilters = myOfficeSDK.TableFilters()
tableFilters.setFilter(0, johnPaulFilter);
tableFilters.setFilter(1, songFilter);
.....................................
filtersRange.setFilters(tableFilters);
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.54 Класс FormulaType

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 31. Используется в `Document.getFormulaType()`, `Document.setFormulaType()`, `DocumentSettings`.

Таблица 31 – Системы адресации ячеек в табличном документе

Константа	Система адресации ячеек	Описание
`FormulaType_A1`	A1	Вариант A1 соответствует наиболее распространенной системе адресации ячеек, при которой столбцы задаются буквами, а строки – числами
`FormulaType_R1C1`	R1C1	Вариант R1C1 соответствует альтернативной системе адресации ячеек, при которой столбцы и строки задаются числами

6.55 Класс FractionCellFormatting

Класс содержит параметры для дробного формата ячеек таблицы. Используется в качестве аргумента метода `Cell.setFormat()`. Описание полей класса `FractionCellFormatting` представлено в таблице 32.

Таблица 32 – Описание полей класса `FractionCellFormatting`

Поле	Описание
`FractionCellFormatting_minNumeratorDigits`	Количество позиций числителя
`FractionCellFormatting_minDenominatorDigits`	Количество позиций знаменателя
`FractionCellFormatting_denominatorValue`	Знаменатель

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("A2")

fractionCellFormat = myOfficeSDK.FractionCellFormatting()
fractionCellFormat.denominatorValue = 22
fractionCellFormat.minDenominatorDigits = 3
fractionCellFormat.minNumeratorDigits = 2

cell.setFormat(fractionCellFormat)
print(cell.getFormattedValue())
```

6.56 Класс Frame

Класс `Frame` описывает прямоугольную область графического объекта документа. Предназначен для получения и изменения свойств графических объектов. Для расположения в позиции текста документа используется `InlineFrame`, в таблице - `AbsoluteFrame` (см. Рисунок 24).

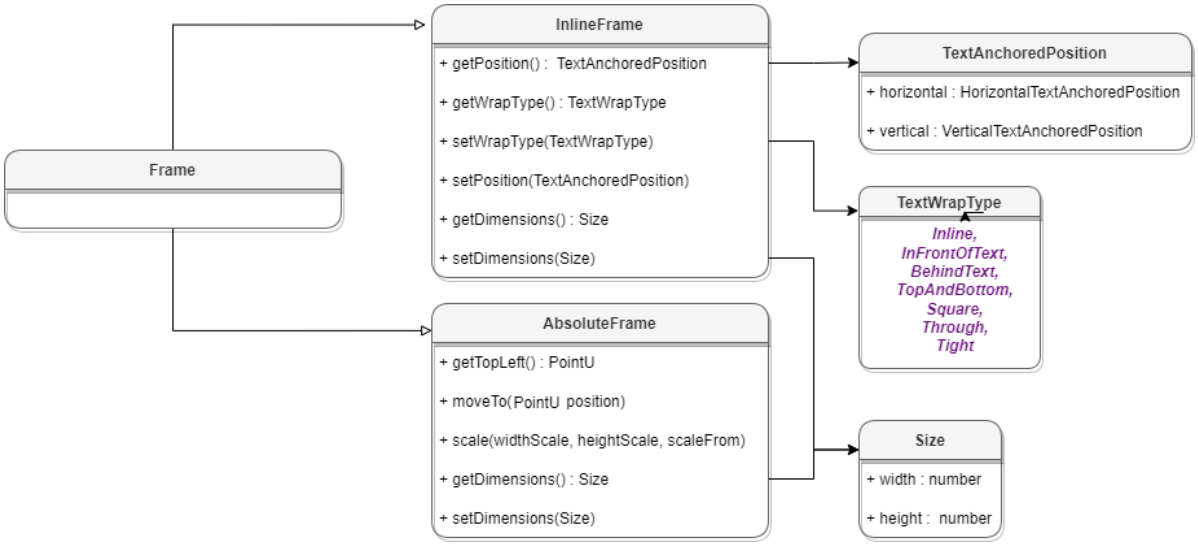


Рисунок 24 – Объектная модель класса `Frame`

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    frame = mediaObject.getFrame()
    if (isinstance(frame, myOfficeSDK.InlineFrame)):
        print("InlineFrame")
    if (isinstance(frame, myOfficeSDK.AbsoluteFrame)):
        print("AbsoluteFrame")
```

6.57 Класс FrozenRangePosition

Класс `FrozenRangePosition` представляет заблокированную область таблицы. Возвращается посредством метода `Table.getFrozenRange()`, устанавливается методом `Table.freeze()`.

6.57.1 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 = myOfficeSDK.FrozenRangePosition();
print(frozenRangePosition.isRowsCols());
```

```
frozenRangePosition = myOfficeSDK.FrozenRangePosition(0, 2, 5, 5);
print(frozenRangePosition.isRowsCols());
```

6.57.2 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 = myOfficeSDK.FrozenRangePosition.create(0, 2, 5, 5)
print(frozenRangePosition.isRowsCols())
```

6.57.3 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 = myOfficeSDK.FrozenRangePosition.createFrozenArea(0, 2)
print(frozenRangePosition.isArea())
```

6.57.4 Метод FrozenRangePosition.createFrozenCols

Создает объект заблокированной области таблицы `FrozenRangePosition`. Область содержит все колонки с `first` по `last`.

Вызов:

```
FrozenRangePosition createFrozenCols(first, last)
```

Пример:

```
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenCols(0, 2)
print(frozenRangePosition.isRowsCols())
```

6.57.5 Метод FrozenRangePosition.createFrozenRows

Создает объект заблокированной области таблицы `FrozenRangePosition`. Область содержит все строки с `first` по `last`.

Вызов:

```
FrozenRangePosition createFrozenRows(first, last)
```

Пример:

```
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenRows(0, 2)
print(frozenRangePosition.isRows())
```

6.57.6 Метод FrozenRangePosition.isArea

Возвращает `True` если диапазон является непрерывной областью.

Пример:

```
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenArea(2, 2)
print(frozenRangePosition.isArea())
```

6.57.7 Метод FrozenRangePosition.isCols

Возвращает `True` если диапазон состоит из колонок.

Пример:

```
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenCols(0, 2)
print(frozenRangePosition.isCols())
```

6.57.8 Метод FrozenRangePosition.isRows

Возвращает `True` если диапазон состоит из строк.

Пример:

```
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenRows(0, 2)
print(frozenRangePosition.isRows())
```

6.57.9 Метод FrozenRangePosition.isRowsCols

Возвращает `True` если диапазон содержит строки и колонки.

Пример:

```
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenArea(2, 2)
print(frozenRangePosition.isRowsCols())
```

6.58 Класс HeaderFooter

Класс `HeaderFooter` определяет колонтитул текстового документа.

6.58.1 Метод HeaderFooter.getBlocks

Метод предоставляет доступ к блокам (`Blocks`), которые содержатся в колонтитуле.

Пример:

```
    headers = section.getHeaders()
    headersEnumerator = headers.getEnumerator()
    for header in headersEnumerator:
        blocks = header.getBlocks()
        blocksEnumerator = blocks.getEnumerator()
        for block in blocksEnumerator:
            print(block.getRange().extractText())
```

6.58.2 Метод HeaderFooter.getRange

Метод предоставляет диапазон (`Range`) с содержанием верхнего или нижнего колонтитулов.

Пример:

```
headers = section.getHeaders()
headersEnumerator = headers.getEnumerator()
for header in headersEnumerator:
    headerRange = header.getRange()
    print(headerRange.extractText())
```

6.58.3 Метод HeaderFooter.getType

Метод предоставляет информацию о типе колонтитула (HeaderFooterType).

Пример:

```
headers = section.getHeaders()
headersEnumerator = headers.getEnumerator()
for header in headersEnumerator:
    headerType = header.getType()
    print(headerType)
```

6.59 Класс HeaderFooterType

Класс HeaderFooterType содержит типы колонтитулов – верхний колонтитул (Header) и нижний колонтитул (Footer).

Поддерживаемые типы колонтитулов страниц документов представлены в таблице 33.

Таблица 33 - Типы колонтитулов страниц документа

Имя константы типа колонтитула	Наименование типа колонтитула
HeaderFooterType_Header	Верхний
HeaderFooterType_Footer	Нижний

Пример:

```
sectionsEnumerator = document.getSectionsEnumerator()
for section in sectionsEnumerator:
    headers = section.getHeaders()
    for header in headers:
        headerFooterType = header.getType()
        print(headerFooterType)
```

6.60 Класс HeadersFooters

Класс HeadersFooters представляет коллекцию верхних и нижних колонтитулов раздела (см. Рисунок 25). Доступ к колонтитулам осуществляется посредством методов Section.getHeaders(), Section.getFooters().

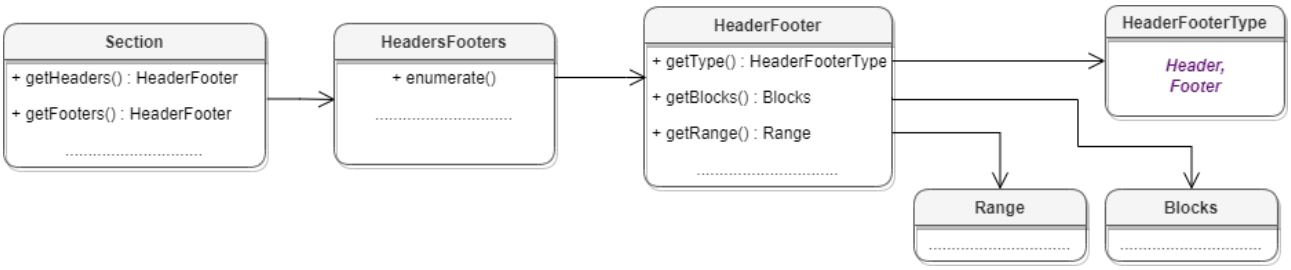


Рисунок 25 – Классы для работы с колонтитулами

6.60.1 Метод HeadersFooters.getEnumerator

Метод возвращает коллекцию колонтитулов.

Примеры:

```
for section in sectionsEnumerator:
    headers = section.getHeaders()
    headersEnumerator = headers.getEnumerator()
    for header in headersEnumerator:
        print (header.getRange().extractText())

    footers = section.getFooters()
    footersEnumerator = footers.getEnumerator()
    for footer in footersEnumerator:
        print(footer.getRange().extractText())
```

6.61 Класс HorizontalAnchorAlignment

В таблице 34 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали.

Таблица 34 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

Наименование константы	Описание
HorizontalAnchorAlignment_Left	По верхнему краю
HorizontalAnchorAlignment_Right	По нижнему краю
HorizontalAnchorAlignment_Center	По центру
HorizontalAnchorAlignment_Inside, HorizontalAnchorAlignment_Outside	По границам

6.62 Класс HorizontalRelativeTo

В таблице 35 представлены типы размещения объекта относительно закрепленной позиции по горизонтали.

Таблица 35 – Типы размещения объекта относительно закрепленной позиции по горизонтали

Наименование константы	Описание
`HorizontalRelativeTo_Character`	Символ
`HorizontalRelativeTo_Column`	Столбец
`HorizontalRelativeTo_ColumnLeftMargin`	Левое поле столбца
`HorizontalRelativeTo_ColumnRightMargin`	Правое поле столбца
`HorizontalRelativeTo_ColumnInsideMargin`	Внутреннее поле столбца
`HorizontalRelativeTo_ColumnOutsideMargin`	Внешнее поле столбца
`HorizontalRelativeTo_Page`	Страница
`HorizontalRelativeTo_PageContent`	Содержимое страницы
`HorizontalRelativeTo_PageLeftMargin`	Левое поле страницы
`HorizontalRelativeTo_PageRightMargin`	Правое поле страницы
`HorizontalRelativeTo_PageInsideMargin`	Внутреннее поле страницы
`HorizontalRelativeTo_PageOutsideMargin`	Внешнее поле страницы

6.63 Класс HorizontalTextAnchoredPosition

Класс `HorizontalTextAnchoredPosition` предназначен для управления относительным положением объекта со смещением или выравниванием по горизонтали.

Описание полей класса `HorizontalTextAnchoredPosition` представлено в таблице 36.

Таблица 36 – Описание полей класса `HorizontalTextAnchoredPosition`

Поле	Описание
`HorizontalTextAnchoredPosition.relativeTo`	Тип размещения объекта относительно закрепленной позиции по горизонтали `HorizontalRelativeTo`

Поле	Описание
`HorizontalTextAnchoredPosition.offset`	Смещение объекта.
`HorizontalTextAnchoredPosition.aligment`	Тип выравнивания объекта относительно закрепленной позиции по горизонтали `HorizontalAnchorAlignment`

6.63.1 HorizontalTextAnchoredPosition.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `HorizontalTextAnchoredPosition`.

Пример:

```
firstHorizontalTextAnchoredPosition =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
firstHorizontalTextAnchoredPosition.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
firstHorizontalTextAnchoredPosition.offset = 10

secondHorizontalTextAnchoredPosition =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
secondHorizontalTextAnchoredPosition.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
secondHorizontalTextAnchoredPosition.offset = 10

if
firstHorizontalTextAnchoredPosition
.__eq__(secondHorizontalTextAnchoredPosition):
    print("Equals")
```

6.63.2 HorizontalTextAnchoredPosition.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `HorizontalTextAnchoredPosition`.

Пример:

```
firstHorizontalTextAnchoredPosition =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
firstHorizontalTextAnchoredPosition.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
firstHorizontalTextAnchoredPosition.offset = 10

secondHorizontalTextAnchoredPosition =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
secondHorizontalTextAnchoredPosition.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
secondHorizontalTextAnchoredPosition.offset = 20

if
firstHorizontalTextAnchoredPosition
.__ne__(secondHorizontalTextAnchoredPosition):
    print("Not equals")
```

6.64 Класс Hyperlink

Класс `Hyperlink` описывает свойства ссылки. Объект `Hyperlink` может быть получен посредством вызова метода `Cell.getHyperlink()`.

Таблица 37 – Описание полей класса `Hyperlink`

Поле	Тип	Описание
`Hyperlink.url`	Строка	Адрес ссылки
`Hyperlink.tooltip`	Строка	Текст подсказки
`Hyperlink.label`	Строка	Текст описания

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A3")
hyperlink = cell.getHyperlink()
print(hyperlink.url)
```

6.64.1 Hyperlink.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа

Hyperlink.

Пример:

```
firstHyperlink = myOfficeSDK.Hyperlink()
firstHyperlink.url = "www.domain.com
firstHyperlink.tooltip = "Press here"
firstHyperlink.label = "Link to feature"

secondHyperlink = myOfficeSDK.Hyperlink()
secondHyperlink.url = "www.domain.com
secondHyperlink.tooltip = "Press here"
secondHyperlink.label = "Link to feature"

if firstHyperlink.__eq__(secondHyperlink):
    print("Equals")
```

6.64.2 Hyperlink.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Hyperlink`.

Пример:

```
firstHyperlink = myOfficeSDK.Hyperlink()
firstHyperlink.url = "www.domain.com
firstHyperlink.tooltip = "Press here"
firstHyperlink.label = "Link to first feature"

secondHyperlink = myOfficeSDK.Hyperlink()
secondHyperlink.url = "www.domain.com
secondHyperlink.tooltip = "Press here"
secondHyperlink.label = "Link to second feature"

if firstHyperlink.__ne__(secondHyperlink):
    print("Not equals")
```

6.65 Класс Image

Класс `Image` представляет собой изображение, находящееся в текстовом или табличном документе.

6.65.1 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 InlineFrame, табличные документы работают с абсолютной позицией и используют тип AbsoluteFrame.

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
for mediaObject in mediaObjects.getEnumerator():
  image = mediaObject.toImage()
  if image != None:
    print(image.getFrame())
```

6.65.2 Метод Image.remove

Метод удаляет изображение из документа.

Пример для текстового документа:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectEnumerator:
    image = mediaObject.toImage()
    if image != None:
        image.remove()
        break
```

6.66 Класс Images

Класс `Images` используется для доступа к коллекции изображений. Объект может быть получен посредством вызова метода `Range.getImages()`.

6.66.1 Метод Images.getEnumerator

Метод позволяет перечислить коллекцию изображений `Image`.

Пример для текстового документа:

```
images = document.getRange().getImages()
for image in images.getEnumerator():
  print(image.getFrame())
```

Пример для табличного документа:

```
sheet = document.getBlocks().getTable(0)
images = sheet.getRange().getImages()
for image in images.getEnumerator():
  print(image.getFrame())
```

6.67 Класс InlineFrame

Класс `InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 26). Предназначен для получения и изменения свойств позиции графических объектов. Используется в текстовом документе.

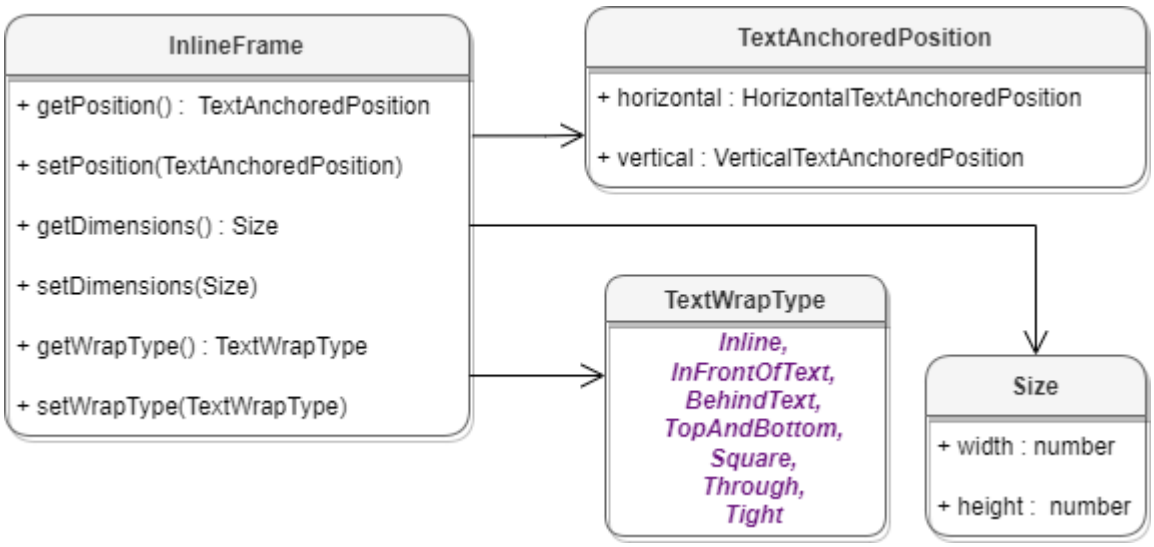


Рисунок 26 – Объектная модель класса `InlineFrame`

Пример для текстового документа:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    frame = mediaObject.getFrame()
    if (isinstance(frame, myOfficeSDK.InlineFrame)):
        print("InlineFrame")
```

6.67.1 Метод InlineFrame.getDimensions

Метод возвращает задает размеры встроенного объекта, тип - `SizeU`.

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
   frame = mediaObject.getFrame()
   if (isinstance(frame, myOfficeSDK.InlineFrame)):
      dimensions = frame.getDimensions()
      print(dimensions.toString())
```

6.67.2 Метод InlineFrame.getPosition

Метод возвращает позицию встроенного объекта на странице типа TextAnchoredPosition.

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.gtEnumerator()
for mediaObject in mediaObjectsEnumerator:
    frame = mediaObject.getFrame()
    anchoredPosition = frame.getPosition()
    textAnchoredPosition = anchoredPosition.textPosition
    print("horz:", textAnchoredPosition.horizontal, ", vert:",
textAnchoredPosition.vertical)
```

6.67.3 Метод InlineFrame.getWrapType

Метод возвращает вариант обтекания текстом встроенного объекта (см. TextWrapType).

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
   frame = mediaObject.getFrame()
   if (isinstance(frame, myOfficeSDK.InlineFrame)):
      wrapType = frame.getWrapType()
      print(wrapType)
```

6.67.4 Метод InlineFrame.setDimensions

Метод задает размер `SizeU` встроенного объекта.

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
medaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
   frame = mediaObject.getFrame()
   if (isinstance(frame, myOfficeSDK.InlineFrame)):
      frame.setDimensions(myOfficeSDK.SizeU(50, 50))
```

6.67.5 Метод InlineFrame.setPosition

Метод задает положение встроенного объекта, тип аргумента TextAnchoredPosition. Новая позиция может быть установлена только для встроенных объектов, тип переноса текста которых не является типом TextWrapType.Inline.

Примеры:

```
// Позиция встроенного объекта не может быть задана,
// если стиль переноса текста - inline.
// Сначала его следует изменить на тип, отличный от inline.
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    wrapType = frame.getWrapType()
    if (wrapType == myOfficeSDK.TextWrapType_Inline):
       frame.setWrapType(myOfficeSDK.TextWrapType_TopAndBottom)
```

Используя классы HorizontalTextAnchoredPosition, VerticalTextAnchoredPosition, можно задать положение встроенных объектов в текстовом документе с учетом относительного смещения.

```
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
   position = myOfficeSDK.TextAnchoredPosition()
   position.horizontal =
myOfficeSDK
.HorizontalTextAnchoredPosition(myOfficeSDK.HorizontalRelativeTo_Page)
   position.horizontal.offset = 10
   position.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageTopMargin)
   position.vertical.offset = 15
   frame.setPosition(position)
```

Используя классы HorizontalTextAnchoredPosition, VerticalTextAnchoredPosition, можно задать положение встроенных объектов в текстовом документе с учетом относительного выравнивания.

```
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
   frame.setWrapType(myOfficeSDK.TextWrapType_TopAndBottom)
   position = myOfficeSDK.TextAnchoredPosition()
   position.horizontal =
```

```
myOfficeSDK
.HorizontalTextAnchoredPosition(myOfficeSDK.HorizontalRelativeTo_Page)
   position.horizontal.alignment = myOfficeSDK.HorizontalAnchorAlignment_Center
   position.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageTopMargin)
   position.vertical.alignment = myOfficeSDK.VerticalAnchorAlignment_Top
   frame.setPosition(position)
```

Используя типы смещения HorizontalRelativeTo.Column и VerticalRelativeTo.Page, можно установить абсолютное положение встроенного объекта в текстовом документе.

```
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
   frame.setWrapType(myOfficeSDK.TextWrapType_TopAndBottom)
   position = myOfficeSDK.TextAnchoredPosition()
   position.horizontal =
myOfficeSDK
.HorizontalTextAnchoredPosition(myOfficeSDK.HorizontalRelativeTo_Column)
   position.horizontal.offset = 125
   position.vertical =
myOfficeSDK.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Page)
   position.vertical.offset = 545
   frame.setPosition(position)
```

6.67.6 Метод InlineFrame.setWrapType

Метод устанавливает вариант обтекания текстом встроенного объекта (см. TextWrapType).

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
   frame = mediaObject.getFrame()
   if (isinstance(frame, myOfficeSDK.InlineFrame)):
      frame.setWrapType(myOfficeSDK.TextWrapType_Inline)
      print(frame.getWrapType())
```

6.68 Класс InputFieldControl

Представляет собой текстовое поле в документе. Является наследником класса `ContentControl`. Методы `InputFieldControl.getValue()` и `InputFieldControl.setValue(string)` позволяют получить и задать значение этого элемента управления.

Пример:

```
controls = document.getContentControls()
inputField = controls.findByTitle("input").toInputField()
inputField.setValue(inputField.getValue().replace(' ', '_'))
```

6.69 Класс Insets

Класс `Insets` предназначен для задания полей, например, страницы. Поля класса `Insets` представлены в таблице 38. Используется в поле `margins` класса `PageProperties`.

Таблица 38 – Описание полей класса `Insets`

Поле	Тип	Описание
`left`	Number	Левая граница поля
`top`	Number	Верхняя граница поля
`right`	Number	Правая граница поля
`bottom`	Number	Нижняя граница поля

Пример:

```
pageInsets = myOfficeSDK.Insets()
pageInsets.left = 0
pageInsets.top = 0
pageInsets.right = 100
pageInsets.bottom = 100

pageProperties = myOfficeSDK.PageProperties()
pageProperties.margins = pageInsets
document.setPageProperties(pageProperties)
```

6.69.1 Insets.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Insets`.

Пример:

```
frameInsets = myOfficeSDK.Insets()
frameInsets.left = 0
frameInsets.top = 0
frameInsets.right = 100
frameInsets.bottom = 100

windowInsets = myOfficeSDK.Insets()
windowInsets.left = 0
windowInsets.top = 0
windowInsets.right = 100
windowInsets.bottom = 101

if frameInsets.__eq__(windowInsets):
    print("Eq")
```

6.69.2 Insets.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Insets`.

Пример:

```
frameInsets = myOfficeSDK.Insets()
frameInsets.left = 0
frameInsets.top = 0
frameInsets.right = 100
frameInsets.bottom = 100

windowInsets = myOfficeSDK.Insets()
windowInsets.left = 0
windowInsets.top = 0
windowInsets.right = 100
windowInsets.bottom = 101

if frameInsets.__ne__(windowInsets):
    print("Ne")
```

6.70 Класс LineEndingProperties

Класс `LineEndingProperties` содержит варианты оформления окончаний линий. Описание полей класса `LineEndingProperties` представлено в таблице 39. Используется в полях `headLineEndingProperties` и `tailLineEndingProperties` класса `LineProperties`.

Таблица 39 – Описание полей класса `LineEndingProperties`

Поле	Тип	Описание
`LineEndingProperties.style`	`LineEndingStyle`	Стиль окончания линии
`LineEndingProperties.relativeExtent`	`Size`	Размер окончания линии относительно ее ширины

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("C3")

lineProperties = myOfficeSDK.LineProperties()
lineProperties.headLineEndingProperties = myOfficeSDK.LineEndingProperties()
lineProperties.headLineEndingProperties.style =
myOfficeSDK.LineEndingStyle_Arrow
lineProperties.headLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
lineProperties.headLineEndingProperties.relativeExtent.width = 2
lineProperties.headLineEndingProperties.relativeExtent.height = 2

lineProperties.tailLineEndingProperties = myOfficeSDK.LineEndingProperties()
lineProperties.tailLineEndingProperties.style =
myOfficeSDK.LineEndingStyle_Arrow
lineProperties.tailLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
lineProperties.tailLineEndingProperties.relativeExtent.width = 2
lineProperties.tailLineEndingProperties.relativeExtent.height = 2

lineProperties.style = myOfficeSDK.LineStyle_Solid
lineProperties.width = 1.5
lineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146, 179,
200))

borders = myOfficeSDK.Borders()
borders.setTop(lineProperties)
cell.setBorders(borders)
```

6.70.1 LineEndingProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `LineSpacing`.

Пример:

```
firstLineEndingProperties = myOfficeSDK.LineEndingProperties()
firstLineEndingProperties.style = myOfficeSDK.LineEndingStyle_Arrow
firstLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
firstLineEndingProperties.relativeExtent.width = 2
firstLineEndingProperties.relativeExtent.height = 2

secondLineEndingProperties = myOfficeSDK.LineEndingProperties()
secondLineEndingProperties.style = myOfficeSDK.LineEndingStyle_Arrow
secondLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
secondLineEndingProperties.relativeExtent.width = 2
secondLineEndingProperties.relativeExtent.height = 2

if firstLineEndingProperties.__eq__(secondLineEndingProperties):
    print("Equals")
```

6.70.2 LineEndingProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `LineSpacing`.

Пример:

```
firstLineEndingProperties = myOfficeSDK.LineEndingProperties()
firstLineEndingProperties.style = myOfficeSDK.LineEndingStyle_Arrow
firstLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
firstLineEndingProperties.relativeExtent.width = 1
firstLineEndingProperties.relativeExtent.height = 1

secondLineEndingProperties = myOfficeSDK.LineEndingProperties()
secondLineEndingProperties.style = myOfficeSDK.LineEndingStyle_Arrow
secondLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
secondLineEndingProperties.relativeExtent.width = 2
secondLineEndingProperties.relativeExtent.height = 2

if firstLineEndingProperties.__ne__(secondLineEndingProperties):
    print("Not equals")
```

6.71 Класс LineEndingStyle

В таблице 40 приведены типы окончания линии. Используется в поле `style` класса `LineEndingProperties`.

Таблица 40 – Типы окончания линии

Наименование константы	Описание
LineEndingStyle_Arrow	
LineEndingStyle_Diamond	
LineEndingStyle_Oval	
LineEndingStyle_Stealth	
LineEndingStyle_Triangle	
LineEndingStyle_None	

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("C3")

lineProperties = myOfficeSDK.LineProperties()
lineProperties.headLineEndingProperties = myOfficeSDK.LineEndingProperties()
lineProperties.headLineEndingProperties.style =
myOfficeSDK.LineEndingStyle_Arrow

borders = myOfficeSDK.Borders()
borders.setTop(lineProperties)
cell.setBorders(borders)
```

6.72 Класс LineProperties

Класс `LineProperties` предназначен для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 27).

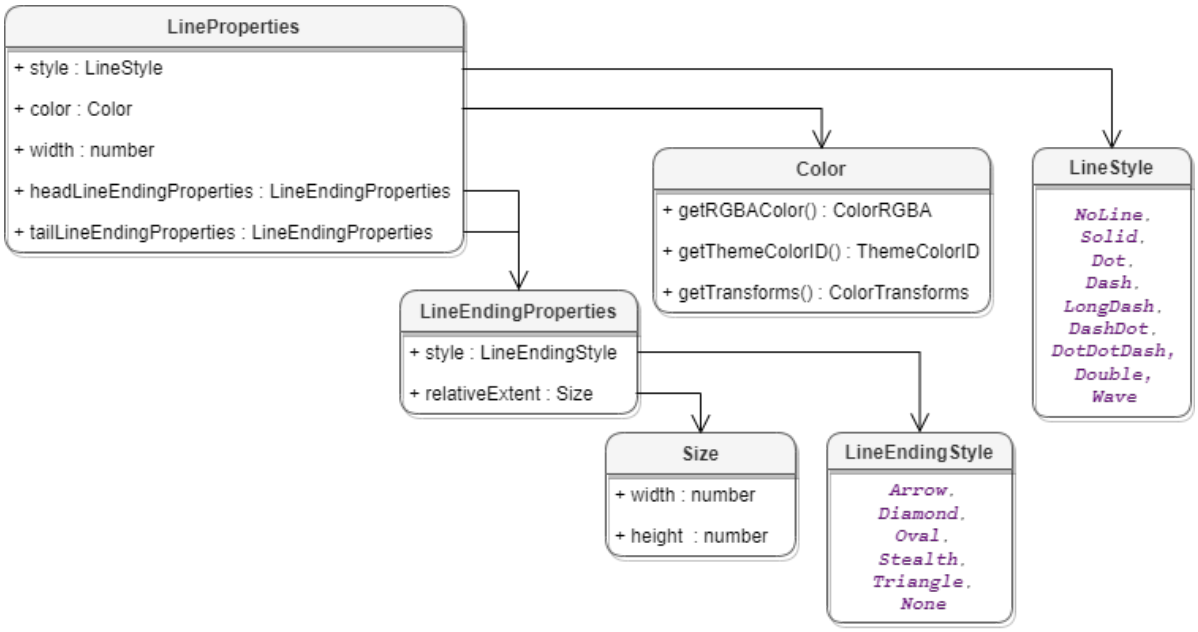


Рисунок 27 – Свойства границ ячеек

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("C3")

lineProperties = myOfficeSDK.LineProperties()
lineProperties.style = myOfficeSDK.LineStyle_Solid
lineProperties.width = 1.5
lineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146, 179,
200))

borders = myOfficeSDK.Borders()
borders.setTop(lineProperties)
cell.setBorders(borders)
```

6.72.1 Поле LineProperties.color

Поле предназначено для установки цвета линии. Тип - Color.

6.72.2 Поле LineProperties.headLineEndingProperties

Поле предназначено для оформления начала линии LineEndingProperties.

6.72.3 Поле LineProperties.style

Поле предназначено для установки типа линии. Допустимые значения представлены в разделе LineStyle.

6.72.4 Поле LineProperties.tailLineEndingProperties

Поле предназначено для оформления конца линии `LineEndingProperties`.

6.72.5 Поле LineProperties.width

Поле предназначено для установки ширины линии. Тип - числовой.

6.72.6 LineProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `LineProperties`.

Пример:

```
firstLineProperties = myOfficeSDK.LineProperties()
firstLineProperties.style = myOfficeSDK.LineStyle_Solid
firstLineProperties.width = 1.5
firstLineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146,
179, 200))

secondLineProperties = myOfficeSDK.LineProperties()
secondLineProperties.style = myOfficeSDK.LineStyle_Solid
secondLineProperties.width = 1.5
secondLineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146,
179, 200))

if firstLineProperties.__eq__(secondLineProperties):
    print("Equals")
```

6.72.7 LineProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `LineProperties`.

Пример:

```
firstLineProperties = myOfficeSDK.LineProperties()
firstLineProperties.style = myOfficeSDK.LineStyle_Solid
firstLineProperties.width = 1.5
firstLineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(55, 146,
179, 200))

secondLineProperties = myOfficeSDK.LineProperties()
secondLineProperties.style = myOfficeSDK.LineStyle_Solid
secondLineProperties.width = 1.5
```

```
secondLineProperties.color = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(57, 146,
179, 200))

if firstLineProperties.__ne__(secondLineProperties):
    print("Not equals")
```

6.73 Класс LineSpacing

Класс `LineSpacing` задает межстрочный интервал абзаца. Поля класса приведены в таблице 41. Для управления значением межстрочного интервала используются значения, представленные в разделе `LineSpacingRule`.

Таблица 41 – Параметры межстрочного интервала

Поле	Описание
`LineSpacing.value`	Значение межстрочного интервала.
`LineSpacing.rule`	Правило формирования межстрочного интервала `LineSpacingRule`.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraphProperties = paragraph.getParagraphProperties()
    paragraphProperties.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple);
```

6.73.1 LineSpacing.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `LineSpacing`.

Пример:

```
lineSpacing = myOfficeSDK.LineSpacing(10, myOfficeSDK.LineSpacingRule_Exact)
lineSpacingEq = myOfficeSDK.LineSpacing(10, myOfficeSDK.LineSpacingRule_Exact)
if lineSpacing.__eq__(lineSpacingEq):
    print("Eq")
```

6.73.2 LineSpacing.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `LineSpacing`.

Пример:

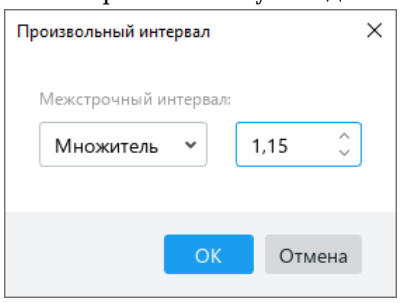
```
lineSpacingExact = myOfficeSDK.LineSpacing(10,
myOfficeSDK.LineSpacingRule_Exact)
lineSpacingMultiple = myOfficeSDK.LineSpacing(10,
myOfficeSDK.LineSpacingRule_Multiple)
if lineSpacingExact.__ne__(lineSpacingMultiple):
    print("Ne")
```

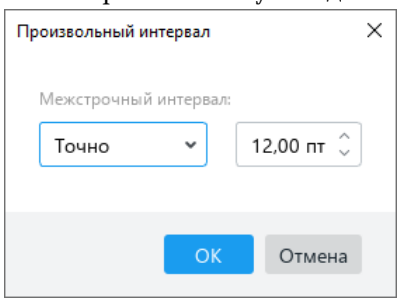
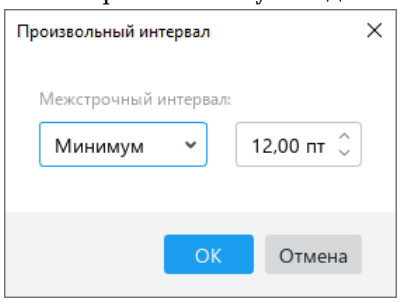
6.74 Класс LineSpacingRule

Класс `LineSpacingRule` содержит типы межстрочного интервалов.

В таблице 42 представлены описания правил формирования межстрочного интервала текстового абзаца.

Таблица 42 – Виды межстрочного интервала

Наименование константы	Описание
`LineSpacingRule_Multiple`	Установка произвольного межстрочного интервала с использованием множителя. При вызове необходимо указать значение множителя, например: `pPr.lineSpacing = myOfficeSDK.LineSpacing(1.15, myOfficeSDK.LineSpacingRule_Multiple)` В данном примере используется значение множителя **1.15**. Действие команды аналогично ручной настройке межстрочного интервала в диалоге **«Произвольный интервал»** (см. документ «МойОфис Текст. Руководство пользователя»). 
`LineSpacingRule_Exact`	Установка произвольного межстрочного интервала с использованием точного значения. При вызове необходимо указать точное значение, например: `pPr.lineSpacing = myOfficeSDK.LineSpacing(12.0, myOfficeSDK.LineSpacingRule_Exact)` В данном примере используется точное значение **12.0**.

Наименование константы	Описание
	Действие команды аналогично ручной настройке межстрочного интервала в диалоге **«Произвольный интервал** (см. документ «МойОфис Текст. Руководство пользователя»). 
`LineSpacingRule_AtLeast`	Установка произвольного межстрочного интервала с использованием минимального значения. При вызове необходимо указать минимальное значение, например: `pPr.lineSpacing = myOfficeSDK.LineSpacing(12.0, myOfficeSDK.LineSpacingRule_AtLeast)` В данном примере используется минимальное значение **12.0**. Действие команды аналогично ручной настройке межстрочного интервала в диалоге **«Произвольный интервал»** (см. документ «МойОфис Текст. Руководство пользователя»). 

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraphProperties = paragraph.getParagraphProperties()
    paragraphProperties.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple);
    paragraph.setParagraphProperties(paragraphProperties)
```

6.75 Класс LineStyle

В таблице 43 приведены типы линий. Используется в поле `style` класса

LineProperties.

Таблица 43 – Типы линий

Наименование константы	Описание
LineStyle_NoLine	Нет линии
LineStyle_Solid	
LineStyle_Dot	
LineStyle_Dash	
LineStyle_LongDash	
LineStyle_DashDot	
LineStyle_DotDotDash	
LineStyle_Double	
LineStyle_Wave	

Пример:

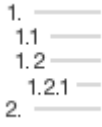
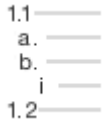
```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("C3")

lineProperties = myOfficeSDK.LineProperties()
lineProperties.style = myOfficeSDK.LineStyle_Solid;
```

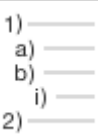
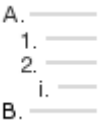
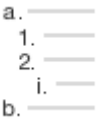
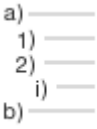
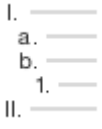
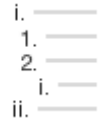
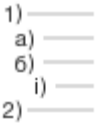
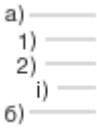
6.76 Класс ListSchema

Класс ListSchema содержит типы схем форматирования списков, которые могут быть применены к абзацам текста. Данные константы используются в методах Paragraph.getListSchema(), Paragraph.setListSchema().

Типы схем абзацев представлены в таблице 44.

Таблица 44 - Типы схем абзацев

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
`ListSchema_Unknown`	Неизвестно.	
`ListSchema_UnknownBullet`	Список без маркера.	Соответствует варианту «нет»
`ListSchema_UnknownNumbering`	Нумерация без номера.	Соответствует варианту «нет»
`ListSchema_BulletCircleSolid`	Список с маркерами в виде круга.	
`ListSchema_BulletCircleContour`	Список с маркерами в виде окружности.	
`ListSchema_BulletSquareSolid`	Список с маркерами в виде квадрата	
`ListSchema_BulletDiamondDots`	Список с маркерами в виде четырех ромбов.	
`ListSchema_BulletHyphen`	Список с маркерами в виде дефиса.	
`ListSchema_BulletConcaveArrowSolid`	Список с маркерами в виде вогнутой стрелки.	
`ListSchema_BulletCheckmark`	Список с маркерами в виде галочки.	
`ListSchema_EnumeratorDecimalDot`	Десятичная нумерация с точкой.	
`ListSchema_EnumeratorDecimalDotMultiLevel`	Многоуровневая десятичная нумерация с точкой.	

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
`ListSchema_Unknown`	Неизвестно.	
`ListSchema_EnumeratorDecimalBracket`	Десятичная нумерация со скобкой.	
`ListSchema_EnumeratorLatinUppercaseDot`	Нумерация латинскими прописными буквами с точкой.	
`ListSchema_EnumeratorLatinLowercaseDot`	Нумерация латинскими строчными буквами с точкой.	
`ListSchema_EnumeratorLatinLowercaseBracket`	Нумерация латинскими строчными буквами со скобкой.	
`ListSchema_EnumeratorRomanUppercaseDot`	Нумерация римскими прописными цифрами с точкой.	
`ListSchema_EnumeratorRomanLowercaseDot`	Нумерация римскими строчными цифрами с точкой.	
`ListSchema_EnumeratorDecimalRussianBracket`	Десятичная нумерация через запятую со скобкой.	
`ListSchema_EnumeratorRussianLowercaseBracket`	Нумерация с русскими строчными буквами со скобкой.	

Пример:

```
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraph.setListSchema(myOfficeSDK.ListSchema_BulletCheckmark)
```

6.77 Класс LoadDocumentSettings

Класс `LoadDocumentSettings` предоставляет дополнительные настройки, необходимые для загрузки документов из файла, см. `Application.loadDocument()`.

Описание полей класса `LoadDocumentSettings` представлено в таблице 45.

Таблица 45 – Описание полей класса LoadDocumentSettings

Поле	Описание
LoadDocumentSettings_commonDocumentSettings	Экземпляр таблицы, общие настройки документа DocumentSettings
LoadDocumentSettings_encoding	Кодировка документа Encoding
LoadDocumentSettings_dsvSettings	Экземпляр класса DSVSettings, настройки, необходимые для работы с файлами CSV и DSV
LoadDocumentSettings_documentPassword	Пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows

6.78 Класс LocaleInfo

Класс LocaleInfo предоставляет информацию о локализации. Используется в поле localeInfo класса DocumentSettings.

Описание полей LocaleInfo представлено в таблице 46.

Таблица 46 – Описание полей класса LocaleInfo

Поле	Описание
LocaleInfo.localeName	Название локализации, представлено в формате **<language> <REGION>**, где языковой код соответствует стандарту ISO-639, а код региона стандарту ISO-3166.
LocaleInfo.decimalSeparator	Десятичный разделитель, отделяет целые и дробные части чисел.
LocaleInfo.thousandSeparator	Символ, разделяющий группы цифр в числовых значениях.
LocaleInfo.listSeparator	Символ, отделяющий элементы в списке.
LocaleInfo.currencySymbol	Символ валюты, используемой в текущей стране или регионе.
LocaleInfo.currencyFormat	Расположение знака валюты в текущем регионе, тип: CurrencySignPlacement.

Поле	Описание
`LocaleInfo.shortDatePattern`	Заданный «короткий» формат отображения даты в текущем регионе (например, 'm/d/yy' для en_US).
`LocaleInfo.longDatePattern`	Заданный «длинный» формат отображения даты в текущем регионе (например, 'dddd, mmmm d, yyyy' для en_US).
`LocaleInfo.timePattern`	Заданный формат отображения времени в текущем регионе (например, 'h:mm AM/PM' для en_US).

6.79 Класс MediaObject

Класс `MediaObject` представляет собой встроенный объект документа.

6.79.1 Метод MediaObject.getFrame

Метод возвращает свойства позиции встроенного объекта `Frame`.

Пример:

```
mediaObjects = document.getRange().getInlineObjects()
for mediaObject in mediaObjects:
  print(mediaObject.getFrame())
```

6.79.2 Метод MediaObject.toChart

Метод возвращает диаграмму `Chart`, связанную со встроенным объектом. Если объект не является диаграммой, метод возвращает `nil`.

Пример:

```
for mediaObject in document.getRange().getInlineObjects():
  chart = mediaObject.toChart()
  if chart != None:
    print("Текущий объект является диаграммой")
  else:
    print("Текущий объект является фигурой")
```

6.79.3 Метод MediaObject.toImage

Метод возвращает изображение `Image`, связанное со встроенным объектом. Если объект не является изображением, метод возвращает `nil`.

Пример:

```
for mediaObject in document.getRange().getInlineObjects():
  image = mediaObject.toImage()
  if image != None:
    print("Текущий объект является изображением")
  else:
    print("Текущий объект является фигурой")
```

6.80 Класс MediaObjects

Класс `MediaObjects` предназначен для доступа к коллекции графических объектов. Объект может быть получен вызовом методов `Range.getInlineObjects()`, `Table.getMediaObjects()` (см. Рисунок 28).

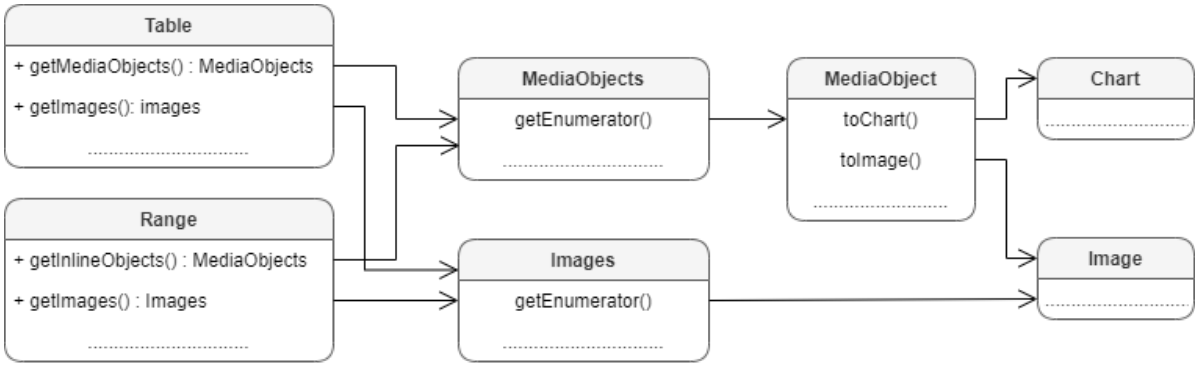


Рисунок 28 – Графические объекты

6.80.1 Метод MediaObjects.getEnumerator

Метод позволяет перечислить коллекцию встроенных объектов.

Примеры для текстового документа:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    print(mediaObject.getFrame().getWrapType())
```

Пример для табличного документа:

```
sheet = document.getBlocks().getTable("Лист1")
mediaObjects = sheet.getMediaObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    image = mediaObject.getImage()
```

```
if image != None:
    print("Текущий объект является изображением")
else:
    print("Текущий объект является фигурой")
```

6.81 Класс Message

Класс `Message` предназначен для формирования событий лога.

6.81.1 Класс Message.Severity

Класс `Message.Severity` (Таблица 47) описывает уровни сообщений лога (информация, предупреждение, ошибка).

Таблица 47 – Описание уровней лога `Message.Severity`

Поле	Описание
`Message.Severity_Info`	Информация
`Message.Severity_Warning`	Предупреждение
`Message.Severity_Error`	Ошибка

6.81.2 Метод Message.getSeverity

Метод возвращает уровень лога `Message.Severity`.

6.81.3 Метод Message.getText

Метод возвращает текст сообщения.

6.81.4 Метод Message.makeError

Метод создает сообщение типа `Message.Severity_Error` с заданным текстом.

6.81.5 Метод Message.makeInfo

Метод создает сообщение типа `Message.Severity_Info` с заданным текстом.

6.81.6 Метод Message.makeWarning

Метод создает сообщение типа `Message.Severity_Warning` с заданным текстом.

6.82 Класс Messenger

6.82.1 Метод Messenger.notify

Метод используется для создания события лога

Пример:

```
messageHandler = myOfficeSDK.MessageHandler()
messenger= application.getMessenger()
messenger.notify(Message.makeWarning("Warning"))
```

6.82.2 Метод Messenger.subscribe

Метод служит для подписки на события лога.

Пример:

```
handler = myOfficeSDK.MessageHandler()
messenger = application.getMessenger()
connection = messenger.subscribe(handler)
```

6.83 Класс NamedExpression

Класс описывает структуру именованного диапазона.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
namedExpressionsEnumerator = firstSheet.getNamedExpressions().getEnumerator()
for namedExpressionIndex, namedExpression in
enumerate(namedExpressionsEnumerator):
    print(namedExpression.getName())
    print(namedExpression.getExpression())
    cellRange = namedExpression.getCellRange()
    print(cellRange.getBeginColumn())
    print(cellRange.getLastColumn())
```

6.83.1 Метод NamedExpression.getCellRange

Возвращает диапазон ячеек `CellRange`. Пример см. в `NamedExpression`.

6.83.2 Метод NamedExpression.getExpression

Возвращает текст выражения (формулы). Пример см. в `NamedExpression`.

6.83.3 Метод NamedExpression.getName

Возвращает имя именованного диапазона. Пример см. в `NamedExpression`.

6.84 Класс NamedExpressions

Класс для представления списка именованных диапазонов. Может быть получена с помощью методов Document.getNamedExpressions(), Table.getNamedExpressions().

6.84.1 Метод NamedExpressions.addExpression

Добавляет новый диапазон в список именованных диапазонов, возвращает результат операции NamedExpressionsValidationResult.

Пример:

```
expressionName = "Продажи"
expressionValue = "=Формула покупки!$A$6:$A$14"
validationResult = namedExpressions.addExpression(expressionName,
expressionValue)
namedExpression = namedExpressions.get(expressionName)
if namedExpression != None:
    print(namedExpression.getName())
```

6.84.2 Метод NamedExpressions.get

Возвращает именованный диапазон NamedExpression по имени, в случае, если оно существует.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
namedExpressions = firstSheet.getNamedExpressions()
namedExpression = namedExpressions.get(expressionName)
if namedExpression != None:
    print(namedExpression.getName())
```

6.84.3 Метод NamedExpressions.getEnumerator

Позволяет получить доступ ко всему списку именованных диапазонов.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
namedExpressionsEnumerator = firstSheet.getNamedExpressions().getEnumerator()
for namedExpressionIndex, namedExpression in
enumerate(namedExpressionsEnumerator):
    print(namedExpression.getName())
    print(namedExpression.getExpression())
```

6.84.4 Метод NamedExpressions.removeExpression

Удаляет именованный диапазон по заданному имени, возвращает результат операции NamedExpressionsValidationResult.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
namedExpressions = firstSheet.getNamedExpressions()
expressionName = "Продажи"
validationResult = namedExpressions.removeExpression(expressionName)
print(validationResult)
```

6.85 Класс NamedExpressionsValidationResult

Класс NamedExpressionsValidationResult описывает результат операций NamedExpressions.addExpression(), NamedExpressions.removeExpression().

Поля класса описаны в таблице 48.

Таблица 48 - Поля класса NamedExpressionsValidationResult

Имя константы	Описание
NamedExpressionsValidationResult_Series	Операция выполнена успешно
NamedExpressionsValidationResult_WrongName	Неправильный формат имени
NamedExpressionsValidationResultIsUsedInFormula	Имя уже используется в формуле

6.86 Класс NumberCellFormatting

Класс содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода Cell.setFormat(). Описание полей класса NumberCellFormatting представлено в таблице 49.

Таблица 49 – Описание полей класса NumberCellFormatting

Поле	Описание
decimalPlaces	Количество десятичных позиций
useThousandsSeparator	Использовать разделитель для тысячных

Поле	Описание
useRedForNegative	Использовать красный цвет для отрицательных значений
useBracketsForNagative	Использовать скобки для отрицательных значений
hideSign	Скрывать знак «минус» для отрицательных значений

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("A2")

numberCellFormat = myOfficeSDK.NumberCellFormatting()
numberCellFormat.decimalPlaces = 2
numberCellFormat.useThousandsSeparator = True
numberCellFormat.useRedForNegative = True;
numberCellFormat.useBracketsForNegative = True
numberCellFormat.hideSign = False

cell.setFormat(numberCellFormat)
print(cell.getFormattedValue())
```

6.87 Класс PageFieldOrder

Класс `PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем класса `PivotTableLayoutSettings`. Описание полей класса представлено в таблице 50.

Таблица 50 – Описание полей класса `PageFieldOrder`

Поле	Описание
PageFieldOrder_DownThenOver	Вниз, затем поперек
PageFieldOrder_OverThenDown	Поперек, затем вниз

6.88 Класс PageNumbers

Класс `PageNumbers` используется в качестве поля `pageNumbers` класса `TextExportSettings` и представляет собой коллекцию страниц для экспорта.

Позволяет установить следующие типы страниц для экспорта:

- нечетные, четные страницы, тип `PageParity`;
- список конкретных номеров страниц, тип `VectorUInt`;
- диапазон страниц с указанием начальной и конечной страницы.

Примеры:

```
# четные страницы
pageNumbers = myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
```

```
# конкретные номера страниц
pages = myOfficeSDK.VectorUInt(3)
pages[0] = 1
pages[1] = 13
pages[2] = 25
pageNumbers = myOfficeSDK.PageNumbers(pages)
```

```
# диапазон страниц
pageNumbers = myOfficeSDK.PageNumbers(1, 20)
```

6.88.1 Метод PageNumbers.contains

Метод служит для проверки вхождения заданного номера страницы в коллекцию номеров страниц `PageNumbers`.

Пример:

```
pageNumbers = myOfficeSDK.PageNumbers(1, 20)
print(pageNumbers.contains(2))
```

6.88.2 Метод PageNumbers.getLast

Метод `PageNumbers.getLast` возвращает последний номер страницы.

Пример:

```
pageNumbers = myOfficeSDK.PageNumbers(1, 20)
print(pageNumbers.getLast())
```

6.88.3 Метод PageNumbers.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `PageNumbers`.

Пример:

```
firstPageNumbers = myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
secondPageNumbers = myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)

if firstPageNumbers.__eq__(secondPageNumbers):
    print("Equals")
```

6.88.4 Метод PageNumbers.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `PageNumbers`.

Пример:

```
firstPageNumbers = myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
secondPageNumbers = myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_All)

if firstPageNumbers.__ne__(secondPageNumbers):
    print("Not equals")
```

6.89 Класс PageOrientation

Тип `PageOrientation` определяет варианты ориентации страницы документа: Альбомная (Landscape) или Книжная (Portrait). Может быть использована для получения / установки ориентации страниц для секции или документа. Поддерживаемые типы ориентации страницы представлены в таблице 51.

Таблица 51 - Типы ориентации страницы

Имя константы типа ориентации страницы	Наименование типа ориентации страницы	Изображение
`PageOrientation_Portrait`	Книжная	
`PageOrientation_Landscape`	Альбомная	

Примеры:

```
block = document.getBlocks().getBlock(0)
section = block.getSection()
section.setPageOrientation(myOfficeSDK.PageOrientation_Portrait)
print(section.getPageOrientation())
```

```
sections = document.getSections()
sectionsEnumerator = sections.getEnumerator()
for sectionIndex, section in enumerate(sectionsEnumerator):
    section.setPageOrientation(myOfficeSDK.PageOrientation_Portrait);
    print(section.getPageOrientation());
```

6.90 Класс PageParity

Варианты выбора страниц для экспорта и печати представлены в таблице 52. Используется в PageNumbers, PrintSettings.

Таблица 52 – Варианты выбора страниц для экспорта и печати

Наименование константы	Описание
PageParity_Odd	Только нечетные страницы
PageParity_Even	Только четные страницы
PageParity_All	Все страницы

6.91 Класс PageProperties

Класс PageProperties предоставляет такие свойства страницы как высота, ширина, размеры полей. Размеры страницы представлены в пунктах (pt). Например, для листа формата A4 значение ширины составляет 595,29 pt, высоты – 841,90 pt (без округления). Описание полей приведено в таблице 53. Используется в Document.setPageProperties(), Section.getPageProperties(), Section.setPageProperties().

Таблица 53 – Описание полей класса PageProperties

Поле	Описание
height	Высота страницы
width	Ширина страницы
margins	Поля страницы, тип - Insets

Примеры:

```
block = document.getBlocks().getBlock(0);
firstSection = block.getSection()
pageProperties = firstSection.getPageProperties()
```

```
pageProperties.height = 100
pageProperties.width = 50
document.setPageProperties(pageProperties)
```

```
pageProperties = myOfficeSDK.PageProperties()
pageProperties.height = 100
pageProperties.width = 50
document.setPageProperties(pageProperties)
```

```
pageProperties = myOfficeSDK.PageProperties(100, 50)
document.setPageProperties(pageProperties)
```

6.91.1 PageProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `PageProperties`.

Пример:

```
firstPageProperties = myOfficeSDK.PageProperties(100, 50)
secondPageProperties = myOfficeSDK.PageProperties(100, 50)

if firstPageProperties.__eq__(secondPageProperties):
    print("Eq")
```

6.91.2 PageProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `PageProperties`.

Пример:

```
firstPageProperties = myOfficeSDK.PageProperties(100, 50)
secondPageProperties = myOfficeSDK.PageProperties(101, 50)

if firstPageProperties.__ne__(secondPageProperties):
    print("Ne")
```

6.92 Класс Paragraph

Класс `Paragraph` предоставляет доступ к свойствам абзаца (см. Рисунок 29).

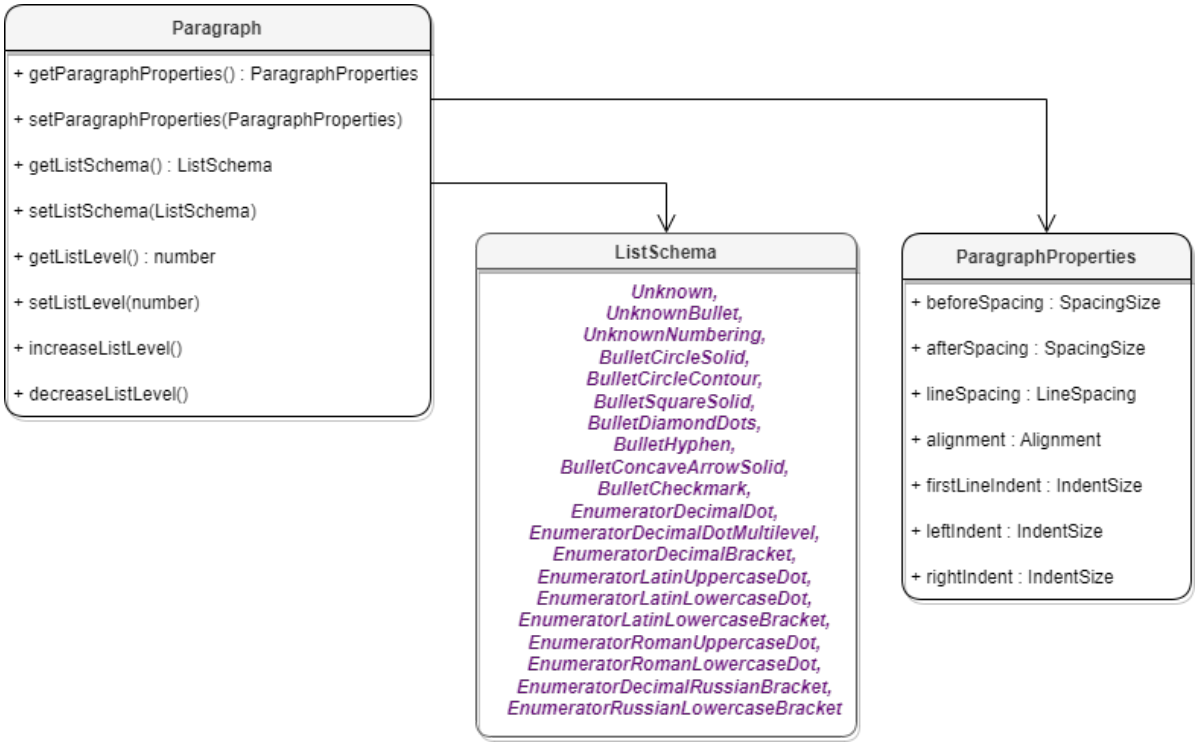


Рисунок 29 – Объектная модель классов для работы со свойствами параграфа

6.92.1 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. Если схема нумерации для абзаца не установлена, будет вызвано исключение `DocumentModificationError`.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    try:
       paragraph.decreaseListLevel()
       print(paragraph.getListLevel())
    except myOfficeSDK.DocumentModificationError as err:
       print(err)
```

6.92.2 Метод Paragraph.getListLevel

Метод позволяет получить глубину вложенности элемента списка. Данный метод используется только в текстовом документе для параграфа, который является частью маркированного или нумерованного списка (для параграфа установлен тип списка ListSchema). В противном случае метод вернет **None**.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    print(paragraph.getListLevel())
```

6.92.3 Метод Paragraph.getListSchema

Метод возвращает схему форматирования абзаца ListSchema, если схема нумерации установлена для абзаца, в противном случае метод вернет **None**. Данный метод используется только в текстовом документе.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    print(paragraph.getListSchema())
```

6.92.4 Метод Paragraph.getParagraphProperties

Метод предоставляет доступ к классу, определяющему такие свойства абзаца ParagraphProperties, как выравнивание текста, межстрочные интервалы, отступы и т. д.

Пример для текстового документа:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
}
```

Пример для табличного документа:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
cellRange = cell.getRange()
```

```
paragraphs = cellRange.getParagraphs()
paragraphsEnumerator = paragraphs.getEnumerator()
for paragraph in paragraphsEnumerator:
    paragraphProperties = paragraph.getParagraphProperties()
    paragraphProperties.alignment = myOfficeSDK.Alignment_Left
    paragraph.setParagraphProperties(paragraphProperties)
```

6.92.5 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. Если схема нумерации для абзаца не установлена, будет вызвано исключение DocumentModificationError.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    try:
       paragraph.increaseListLevel()
       print(paragraph.getListLevel())
    except myOfficeSDK.DocumentModificationError as err:
       print(err)
```

6.92.6 Метод Paragraph.setListLevel

Метод позволяет установить глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Если схема нумерации для абзаца не установлена, будет вызвано исключение DocumentModificationError.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    try:
       paragraph.setListLevel(1)
       print(paragraph.getListSchema())
    except myOfficeSDK.DocumentModificationError as err:
       print(err)
```

6.92.7 Метод Paragraph.setListSchema

Метод позволяет установить тип маркированного или нумерованного списка ListSchema. Данный метод используется только в текстовом документе.

Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraph.setListSchema(myOfficeSDK.ListSchema_BulletCheckmark)
```

6.92.8 Метод Paragraph.setParagraphProperties

Метод предназначен для обновления свойств форматирования абзаца ParagraphProperties.

Пример для текстового документа:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraphProperties = paragraph.getParagraphProperties()
    paragraphProperties.alignment = myOfficeSDK.Alignment_Left
    paragraph.setParagraphProperties(paragraphProperties)
```

Пример для табличного документа:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
cellRange = cell.getRange()
paragraphs = cellRange.getParagraphs()
paragraphsEnumerator = paragraphs.getEnumerator()
for paragraph in paragraphsEnumerator:
    paragraphProperties = paragraph.getParagraphProperties()
    paragraphProperties.alignment = myOfficeSDK.Alignment_Left
    paragraph.setParagraphProperties(paragraphProperties)
```

6.93 Класс ParagraphProperties

Класс `ParagraphProperties` предназначен для управления свойствами форматирования (см. Рисунок 30). Класс `ParagraphProperties` используется в методах `Paragraph.getParagraphProperties` и `Paragraph.setParagraphProperties`.

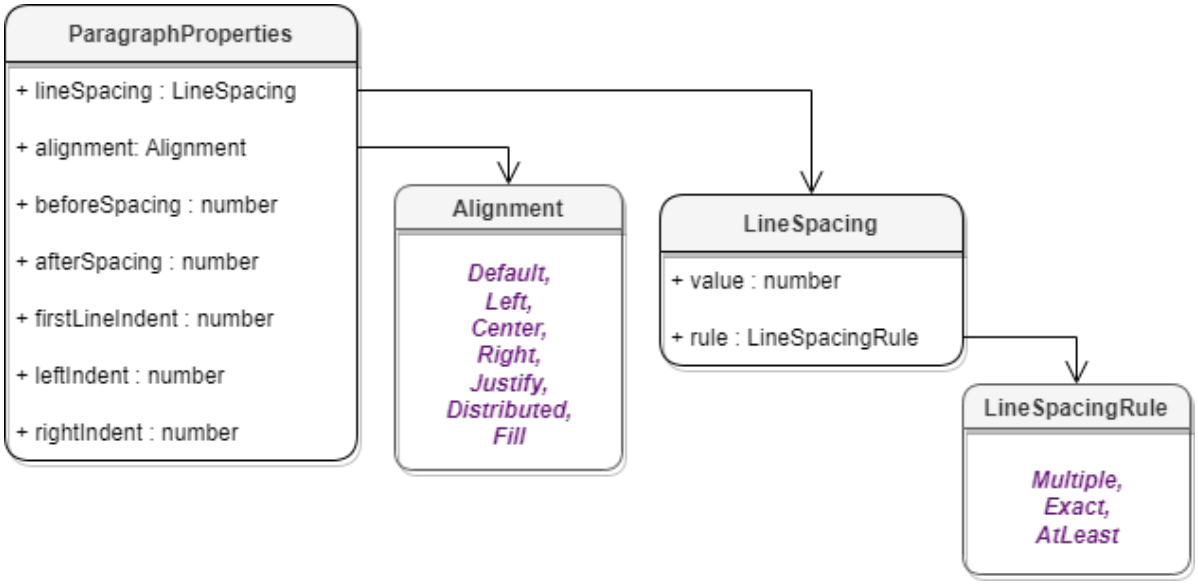
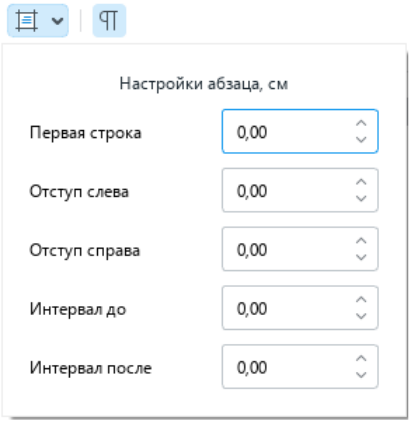


Рисунок 30 – Объектная модель классов для работы со свойствами параграфа

Описание полей класса `ParagraphProperties` представлено в таблице 54.

Таблица 54 – Описание полей класса `ParagraphProperties`

Поле	Описание
`ParagraphProperties_beforeSpacing`	Установка величины расстояния до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца,** (см. рисунок выше), в поле **Интервал до** (подробнее см. в документе «МойОфис Текст. Руководство пользователя»). 
`ParagraphProperties_afterSpacing`	Установка величины расстояния после абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, в поле

Поле	Описание
	Интервал после (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).
ParagraphProperties_lineSpacing	Расстояние между строк одного абзаца (межстрочный интервал), LineSpacing.
ParagraphProperties_alignment	Выравнивание текстового фрагмента по горизонтали. Список допустимых значений находится в разделе Alignment.
ParagraphProperties_firstLineIndent	Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Первая строка** (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).
ParagraphProperties_leftIndent	Расстояние от левого поля документа до абзаца (отступ слева). При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ слева** (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).
ParagraphProperties_rightIndent	Расстояние от правого поля документа до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ справа** (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).

Пример для текстового документа:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraph.setListSchema(myOfficeSDK.ListSchema_BulletCheckmark)
    paraProps = paragraph.getParagraphProperties()
    paraProps.afterSpacing = 28.3      # соответствует 1 см
    paraProps.beforeSpacing = 28.3     # соответствует 1 см
    paraProps.firstLineIndent = 28.3   # соответствует 1 см
    paraProps.leftIndent = 28.3        # соответствует 1 см
```

```
    paraProps.rightIndent = 28.3       # соответствует 1 см
    paraProps.alignment = myOfficeSDK.Alignment_Center
    paraProps.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple)
```

Пример для табличного документа:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
cellRange = cell.getRange()
paragraphs = cellRange.getParagraphs()
paragraphsEnumerator = paragraphs.getEnumerator()
for paragraphIndex, paragraph in enumerate(paragraphsEnumerator):
    paragraphProperties = paragraph.getParagraphProperties()
    print(paragraphProperties.alignment)
```

6.93.1 ParagraphProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `ParagraphProperties`.

Пример:

```
firstParaProps = myOfficeSDK.ParagraphProperties()
firstParaProps.afterSpacing = 28.3
firstParaProps.beforeSpacing = 28.3
firstParaProps.firstLineIndent = 28.3
firstParaProps.leftIndent = 28.3
firstParaProps.rightIndent = 28.3
firstParaProps.alignment = myOfficeSDK.Alignment_Center
firstParaProps.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple)

secondParaProps = myOfficeSDK.ParagraphProperties()
secondParaProps.afterSpacing = 28.3
secondParaProps.beforeSpacing = 28.3
secondParaProps.firstLineIndent = 28.3
secondParaProps.leftIndent = 28.3
secondParaProps.rightIndent = 28.3
secondParaProps.alignment = myOfficeSDK.Alignment_Center
secondParaProps.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple)
```

```
if firstParaProps.__eq__(secondParaProps):
    print("Equals")
```

6.93.2 ParagraphProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `ParagraphProperties`.

Пример:

```
firstParaProps = myOfficeSDK.ParagraphProperties()
firstParaProps.afterSpacing = 28.3
firstParaProps.beforeSpacing = 28.3
firstParaProps.firstLineIndent = 28.3
firstParaProps.leftIndent = 28.3
firstParaProps.rightIndent = 28.3
firstParaProps.alignment = myOfficeSDK.Alignment_Center
firstParaProps.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple)

secondParaProps = myOfficeSDK.ParagraphProperties()
secondParaProps.afterSpacing = 28.3
secondParaProps.beforeSpacing = 28.3
secondParaProps.firstLineIndent = 28.3
secondParaProps.leftIndent = 28.3
secondParaProps.rightIndent = 28.3
secondParaProps.alignment = myOfficeSDK.Alignment_Left
secondParaProps.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple)

if firstParaProps.__ne__(secondParaProps):
    print("Not equals")
```

6.94 Класс Paragraphs

Класс `Paragraphs` предоставляет доступ к коллекции абзацев типа `Paragraph` (см. Рисунок 31). Коллекция абзацев может быть получена из объекта `Range` посредством использования метода `Range.getParagraphs()`.

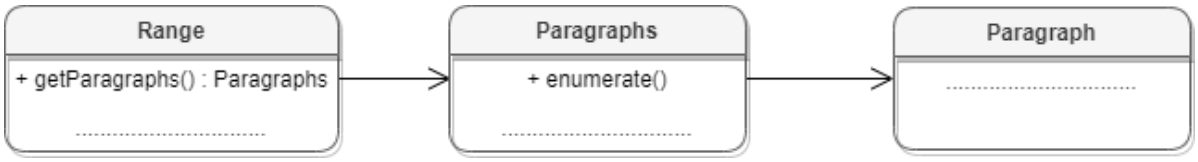


Рисунок 31 – Объектная модель для работы со списком абзацев

Пример для текстового документа:

```
document.getRange()
paragraphs = range.getParagraphs()
```

Пример для табличного документа:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
range = cell.getRange()
paragraphs = range.getParagraphs()
```

6.94.1 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:

```
docRange = document.getRange()
paragraphs = docRange.getParagraphs()
paragraphs.decreaseListLevel()
```

6.94.2 Метод Paragraphs.getEnumerator

Метод позволяет перечислить коллекцию абзацев.

Пример:

```
docRange = document.getRange()
paragraphs = docRange.getParagraphs()
paragraphsEnumerator = paragraphs.getEnumerator()
for paragraph in paragraphsEnumerator:
    print(paragraph.getRange().extractText())
```

6.94.3 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:

```
docRange = document.getRange()
paragraphs = docRange.getParagraphs()
paragraphs.increaseListLevel()
```

6.94.4 Метод Paragraphs.setListLevel

Метод устанавливает глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример:

```
docRange = document.getRange()
paragraphs = docRange.getParagraphs()
paragraphs.setListLevel(1)
```

6.94.5 Метод Paragraphs.setListSchema

Метод устанавливает тип маркированного или нумерованного списка `ListSchema`. Данный метод используется только в текстовом документе.

Пример:

```
docRange = document.getRange()
paragraphs = docRange.getParagraphs()
paragraphs.setListSchema(myOfficeSDK.ListSchema_BulletCheckmark);
```

6.95 Класс PercentageCellFormatting

Содержит параметр для процентного формата ячеек таблицы, используется в качестве аргумента метода `Cell.setFormat()`. Описание полей класса `PercentageCellFormatting` представлено в таблице 55.

Таблица 55 – Описание полей класса `PercentageCellFormatting`

Поле	Описание
`decimalPlaces`	Количество десятичных позиций

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("A2")
```

```
percentageCellFormat = myOfficeSDK.PercentageCellFormatting()
percentageCellFormat.decimalPlaces = 2

cell.setFormat(percentageCellFormat)
print(cell.getFormattedValue())
```

6.96 Класс PivotTable

Класс для представления сводной таблицы. Может быть получен из ячейки `Cell.getPivotTable()`, либо при создании новой сводной таблицы `PivotTablesManager.create()`.

6.96.1 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")
sourceRange = pivotTable.getSourceRange()
print(sourceRange.getBeginColumn() + ", " + sourceRange.getLastColumn())
```

6.96.2 Метод PivotTable.createPivotTableEditor

Метод возвращает объект `PivotTableEditor`, который служит для обновления свойств и редактирования сводной таблицы.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor.addField("Age", myOfficeSDK.PivotTableFieldCategory_Rows)
pivotTableEditor.apply()
```

6.96.3 Метод PivotTable.getColumnFields

Метод возвращает список полей `PivotTableCategoryField` из области колонок.

Пример:

```
columnFields = pivotTable.getColumnFields()
columnFieldsEnumerator = columnFields.getEnumerator()
for pivotTableCategoryField in columnFieldsEnumerator:
    print(pivotTableCategoryField.fieldProperties.fieldAlias)
```

```
    print(pivotTableCategoryField.fieldProperties.subtotalAlias)
    print(pivotTableCategoryField.fieldProperties.fieldName)
    subtotalFunctions = pivotTableCategoryField.subtotalFunctions
    print(subtotalFunctions.Count)
```

6.96.4 Метод PivotTable.getFieldCategories

Метод возвращает список категорий PivotTableFieldCategories, содержащих заданное поле fieldName.

Пример:

```
pivotTableFieldCategories = pivotTable.getFieldCategories("Age")
categoriesEnumerator = categories.getEnumerator()
for pivotTableFieldCategory in categoriesEnumerator:
    print(pivotTableFieldCategory)
```

6.96.5 Метод PivotTable.getFieldItems

Метод возвращает все элементы PivotTableItems сводной таблицы по заданному имени поля fieldName.

Пример:

```
pivotTableItems = pivotTable.getFieldItems("Age")
pivotTableItemsEnumerator = pivotTableItems.getEnumerator()
for pivotTableItem in pivotTableItemsEnumerator:
    print(pivotTableItem.getAlias())
```

6.96.6 Метод PivotTable.getFieldItemsByName

Метод возвращает все элементы PivotTableItems из заданного поля fieldName по имени itemName.

Пример:

```
fieldItemsByName = pivotTable.getFieldItemsByName("Ultimate Question of Life",
"42")
itemsEnumerator = itemsByName.getEnumerator()
for pivotTableItem in itemsEnumerator:
    print(pivotTableItem.getName())
```

6.96.7 Метод PivotTable.getFieldsList

Метод возвращает список PivotTableField всех полей сводной таблицы.

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    pivotTableFields = pivotTable.getFieldsList()
    fieldsEnumerator = pivotTableFields.getEnumerator()
    for pivotTableField in fieldsEnumerator:
        print(pivotTableField.customFormula)
```

6.96.8 Метод PivotTable.getFilter

Метод возвращает фильтр PivotTableFilter по заданному имени поля `fieldName`.

Пример:

```
pivotTableFilter = pivotTable.getFilter("Age")
print(pivotTableFilter.getFieldName())
```

6.96.9 Метод PivotTable.getFilters

Метод возвращает список фильтров PivotTableFilter сводной таблицы.

Пример:

```
pivotTableFilters = pivotTable.getFilters()
pivotTableFiltersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilterin in pivotTableFiltersEnumerator:
    pivotTableFilter.setHidden()
```

6.96.10 Метод PivotTable.getPageFields

Метод возвращает список полей PivotTablePageField из области фильтров.

Пример:

```
pageFields = pivotTable.getPageFields()
pageFieldsEnumerator = pageFields.getEnumerator()
for pivotTableCategory in pageFieldsEnumerator:
    print(pivotTableCategory.fieldProperties.fieldAlias)
    print(pivotTableCategory.fieldProperties.subtotalAlias)
    print(pivotTableCategory.fieldProperties.fieldName)
```

6.96.11 Метод PivotTable.getPivotRange

Метод возвращает диапазон ячеек CellRange, в котором размещена сводная таблица.

Пример:

```
pivotTable = pivotTablesManager.create(cellRange)
pivotRange = pivotTable.getPivotRange()
print(pivotRange.getBeginColumn() + ", " + pivotRange.getLastColumn())
```

6.96.12 Метод PivotTable.getPivotTableCaptions

Метод возвращает информацию PivotTableCaptions о всех заголовках сводной таблицы.

Пример:

```
pivotTableCaptions = pivotTable.getPivotTableCaptions()
print(pivotTableCaptions.errorCaption)
print(pivotTableCaptions.emptyCaption)
print(pivotTableCaptions.grandTotalCaption)
print(pivotTableCaptions.valuesHeaderCaption)
print(pivotTableCaptions.columnHeaderCaption)
print(pivotTableCaptions.rowHeaderCaption)
```

6.96.13 Метод PivotTable.getPivotTableLayoutSettings

Метод возвращает настройки отображения PivotTableLayoutSettings сводной таблицы.

Пример:

```
pivotTableLayoutSettings = pivotTable.getPivotTableLayoutSettings()
print(pivotTableLayoutSettings.displayFieldCaptions)
print(pivotTableLayoutSettings.indentForCompactLayout)
print(pivotTableLayoutSettings.isMergeAndCenterLabelsEnabled)
print(pivotTableLayoutSettings.pageFieldOrder)
print(pivotTableLayoutSettings.pageFieldWrapCount)
print(pivotTableLayoutSettings.reportLayout)
print(pivotTableLayoutSettings.useGridDropZones)
print(pivotTableLayoutSettings.valueFieldsOrientation)
```

6.96.14 Метод PivotTable.getRowFields

Метод возвращает список полей PivotTableCategoryField из области строк.

Пример:

```
rowFields = pivotTable.getRowFields()
rowFieldsEnumerator = rowFields.getEnumerator()
for rowField in rowFieldsEnumerator:
    print(pivotTableCategoryField.fieldProperties.fieldAlias)
    print(pivotTableCategoryField.fieldProperties.subtotalAlias)
    print(pivotTableCategoryField.fieldProperties.fieldName)
    subtotalFunctions = pivotTableCategoryField.subtotalFunctions
    print(subtotalFunctions.Count)
```

6.96.15 Метод PivotTable.getSourceRange

Метод возвращает диапазон CellRange исходных данных сводной таблицы.

Пример:

```
sourceRange = pivotTable.getSourceRange()
print(sourceRange.getBeginColumn())
```

6.96.16 Метод PivotTable.getSourceRangeAddress

Метод возвращает текстовое представление диапазона исходных данных сводной таблицы.

Пример:

```
sheet = document.getBlocks().getTable("Лист1");
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    print(pivotTable.getSourceRangeAddress())
```

6.96.17 Метод PivotTable.getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства PivotTableUnsupportedFeature сводной таблицы.

Пример:

```
unsupportedFeatures = pivotTable.getUnsupportedFeatures()
unsupportedFeaturesEnumerator = unsupportedFeatures.getEnumerator()
```

```
for unsupportedFeature in unsupportedFeaturesEnumerator:
    print(unsupportedFeaturesEnumerator)
```

6.96.18 Метод PivotTable.getValueFields

Метод возвращает список полей `PivotTableValueField` из области значений.

Пример:

```
pivotTableValueFields = pivotTable.getValueFields()
pivotTableValueFieldsEnumerator = valueFields.getEnumerator()
for pivotTableValueField in pivotTableValueFieldsEnumerator:
    print(pivotTableValueField.baseFieldName)
    print(pivotTableValueField.cellNumberFormat)
    print(pivotTableValueField.customFormula)
    print(pivotTableValueField.totalFunction)
    print(pivotTableValueField.valueFieldName)
```

6.96.19 Метод PivotTable.isColumnGrandTotalEnabled

Метод возвращает `True`, если разрешено показывать общие итоги для столбцов.

Пример:

```
print(pivotTable.isColumnGrandTotalEnabled())
```

6.96.20 Метод PivotTable.isRowGrandTotalEnabled

Метод возвращает `True`, если разрешено показывать общие итоги для строк.

Пример:

```
print(pivotTable.isRowGrandTotalEnabled())
```

6.96.21 Метод PivotTable.remove

Метод удаляет сводную таблицу.

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    pivotTable.remove()
```

6.96.22 Метод PivotTable.update

Метод обновляет и полностью пересчитывает сводную таблицу, возвращает PivotTableUpdateResult.

Пример:

```
updateResult = pivotTable.update()
if updateResult == myOfficeSDK.PivotTableUpdateResult_FieldAlreadyEnabled:
```

6.97 Класс PivotTableCaptions

Класс PivotTableCaptions хранит все пользовательские заголовки сводной таблицы. Может быть получен вызовом PivotTable.getPivotTableCaptions(). Описание полей таблицы представлено в таблице 56.

Таблица 56 – Описание полей класса PivotTableCaptions

Поле	Описание
PivotTableCaptions.errorCaption	Алиас для значений, которые возвращают ошибку.
PivotTableCaptions.emptyCaption	Алиас для значений, которые возвращают пустое значение.
PivotTableCaptions.grandTotalCaption	Алиас общих итогов.
PivotTableCaptions.valuesHeaderCaption	Алиас поля из области значений; это поле отображается в отчете в случае, если в сводной таблице наличие более двух полей из области значений, и макет имеют тип ‘outline’ или ‘tabular’.
PivotTableCaptions.rowHeaderCaption	Алиас заголовка строк (виден только при включенном компактном макете, это алиас по умолчанию).
PivotTableCaptions.columnHeaderCaption	Алиас заголовка колонок (виден только при включенном компактном макете, это алиас по умолчанию).

6.98 Класс PivotTableCategoryField

`PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. таблицу 57). Объект может быть получен посредством вызовов `PivotTable.getRowFields()`, `PivotTable.getColumnFields()`.

Таблица 57 – Описание полей `PivotTableCategoryField`

Поле	Описание
`PivotTableCategoryField .fieldProperties`	Свойства поля `PivotTableFieldProperties`
`PivotTableCategoryField .subtotalFunctions`	Список функций `PivotTableFunction` для вычисления подытога

6.99 Класс PivotTableEditor

Предназначен для редактирования сводных таблиц. Возвращается посредством метода `PivotTable.createPivotTableEditor()`.

6.99.1 Метод PivotTableEditor.addField

Метод добавляет новое поле в сводную таблицу, используя параметры: `fieldName` - имя поля, `toCategory` - категория поля (тип - `PivotTableFieldCategory`), `index` - позиция в категории. Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor = pivotTableEditor.addField("CC",
myOfficeSDK.PivotTableFieldCategory_Values)
pivotTableEditor.apply()
```

6.99.2 Метод PivotTableEditor.apply

Метод обновляет сводную таблицу с заданными свойствами и возвращает результат `PivotTableUpdateResult`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
if myOfficeSDK.PivotTableUpdateResult_Success == pivotTableEditor.apply():
    print("Successfully applied")
```

6.99.3 Метод PivotTableEditor.disableField

Метод удаляет поле из всех областей. Параметр `fieldName` - имя поля (тип - строка). Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor = pivotTableEditor.disableField("Age")
pivotTableEditor.apply()
```

6.99.4 Метод PivotTableEditor.enableField

Метод добавляет поле в область, зависящую от типа поля. Параметр `fieldName` - имя поля. Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor = pivotTableEditor.enableField("Age")
pivotTableEditor.apply()
```

6.99.5 Метод PivotTableEditor.moveField

Метод перемещает поле между категориями. Параметры: `fieldName` - имя поля, `toCategory` - область, в которую перемещается поле (тип - `PivotTableFieldCategory`), `index` - позиция в новой категориии. Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor = pivotTableEditor.moveField("CC",
myOfficeSDK.PivotTableFieldCategory_Values, 0)
pivotTableEditor.apply()
```

6.99.6 Метод PivotTableEditor.removeField

Метод удаляет поле из категории. Параметры: `fieldName` - имя поля, `fromCategory` - область, из которой удаляется поле (тип - `PivotTableFieldCategory`). Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor = pivotTableEditor.removeField("BB",
```

```
myOfficeSDK.PivotTableFieldCategory_Values)
pivotTableEditor.apply()
```

6.99.7 Метод PivotTableEditor.reorderField

Метод изменяет позицию поля в пределах категории. Параметры: `fieldName` - имя поля, `category` - область (тип - `PivotTableFieldCategory`), `toIndex` - новая позиция поля. Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor = pivotTableEditor.reorderField("CC",
myOfficeSDK.PivotTableFieldCategory_Values, 0)
pivotTableEditor.apply()
```

6.99.8 Метод PivotTableEditor.setCaptions

Метод задает заголовки сводной таблицы `PivotTableCaptions`, возвращает объект `PivotTableEditor`.

Пример:

```
captions = pivotTable.getPivotTableCaptions()
captions.grandTotalCaption = "Общий итог за год"

pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor.setCaptions(captions)
pivotTableEditor.apply()
```

6.99.9 Метод PivotTableEditor.setFilter

Метод задает фильтр `PivotTableFilter` сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`. Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableFilters = pivotTable.getFilters()
pivotTableFiltersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in pivotTableFiltersEnumerator:
    filterSize = pivotTableFilter.getCount()
    for filterIndex in range(filterSize):
        pivotTableFilter.setHidden(i, True)
```

```
    pivotTableEditor.setFilter(pivotTableFilter)
    pivotTableUpdateResult = pivotTableEditor.apply()
```

6.99.10 Метод PivotTableEditor.setFilters

Метод задает фильтры PivotTableFilters сводной таблицы. Если какой-то из фильтров не может быть применен, он пропускается. Метод возвращает объект PivotTableEditor.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableFilters = pivotTable.getFilters()
pivotTableFiltersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in pivotTableFiltersEnumerator:
    filterSize = pivotTableFilter.getCount()
    for filterIndex in range(filterSize):
        filter.setHidden(filterIndex, True)
    pivotTableEditor.setFilter(filter)
```

6.99.11 Метод PivotTableEditor.setGrandTotalSettings

Метод задает настройки отображения общего итога. Параметры: `isRowGrandTotalEnabled` – показывать общие итоги для строк, `isColGrandTotalEnabled` – показывать общие итоги для столбцов.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
pivotTableEditor = pivotTableEditor.setGrandTotalSettings(True, True)
pivotTableEditor.apply()
```

6.99.12 Метод PivotTableEditor.setLayoutSettings

Метод устанавливает настройки отображения PivotTableLayoutSettings сводной таблицы, возвращает объект PivotTableEditor.

Пример:

```
pivotTableLayoutSettings = pivotTable.getPivotTableLayoutSettings()
pivotTableLayoutSettings.reportLayout =
myOfficeSDK.PivotTableReportLayout_Tabular

pivotTableEditor = pivotTable.createPivotTableEditor()
```

```
pivotTableEditor = pivotTableEditor.setLayoutSettings(pivotTableLayoutSettings)
pivotTableEditor.apply()
```

6.99.13 Метод PivotTableEditor.setSummarizeFunction

Метод задает суммирующую функцию для поля из области значений. Параметр `valueFieldName` - имя поля (тип - строка), `summarizeFunction` – суммирующая функция, тип - `PivotTableFunction`. Метод возвращает объект `PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()
summarizeFunction = myOfficeSDK.PivotTableFunction_Average
pivotTableEditor = pivotTableEditor.setSummarizeFunction("CC",
summarizeFunction)
```

6.100 Класс PivotTableField

Класс `PivotTableField` содержит свойства полей сводной таблицы (см. таблицу 58). Объект может быть получен посредством вызова `PivotTable.getFieldsList()`.

Таблица 58 – Описание полей класса `PivotTableField`

Поле	Описание
`PivotTableField.fieldProperties`	Свойства полей сводной таблицы `PivotTablFieldProperties`
`PivotTableField.fieldCategories`	Категории полей сводной таблицы `PivotTablFieldCategories`
`PivotTableField.customFormula`	Вычисляемая формула (строка)

6.101 Класс PivotTableFieldCategories

Класс обеспечивает доступ к списку категорий поля сводной таблицы. Объект может быть получен посредством использования метода `PivotTable.getFieldCategories()`.

6.101.1 Метод PivotTableFieldCategories.getEnumerator

Метод для перечисления категорий поля `PivotTableFieldCategory`.

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
```

```
pivotTable = cell.getPivotTable()
if pivotTable != None:
    fieldCategories = pivotTable.getFieldCategories("Age")
    fieldCategoriesEnumerator = fieldCategories.getEnumerator()
    for fieldCategory in fieldCategoriesEnumerator:
        if fieldCategory != None:
            print(fieldCategory.getType())
```

6.102 Класс PivotTableFieldCategory

Класс `PivotTableFieldCategory` описывает флаги, которые задают категорию области полей. Описание полей представлено в таблице 59. Пример использования см. в `PivotTable.getFieldCategories()`.

Таблица 59 – Описание полей класса `PivotTableFieldCategory`

Поле	Описание
`PivotTableFieldCategory_Pages`	Область фильтров
`PivotTableFieldCategory_Rows`	Область строк
`PivotTableFieldCategory_Columns`	Область колонок
`PivotTableFieldCategory_Values`	Область значений

6.103 Класс PivotTableFieldProperties

`PivotTableFieldProperties` содержит свойства поля `PivotTableField` сводной таблицы (см. таблицу 60).

Таблица 60 – Описание полей класса `PivotTableFieldProperties`

Поле	Описание
`PivotTableFieldProperties.fieldName`	Имя поля
`PivotTableFieldProperties.fieldAlias`	Псевдоним поля (пользовательское имя)
`PivotTableFieldProperties.subtotalAlias`	Псевдоним подытогов конкретного поля

6.104 Класс PivotTableFilter

Позволяет осуществить доступ к списку фильтров таблицы, каждый из которых обладает свойством видимости. При любом изменении фильтров они должны быть применены к сводной таблице посредством использования методов PivotTableEditor.setFilter(), PivotTableEditor.setFilters().

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    pivotTableFilters = pivotTable.getFilters()
    filtersEnumerator = pivotTableFilters.getEnumerator()
    for pivotTableFilter in filtersEnumerator:
        for filterIndex in range(pivotTableFilter.getCount()):
            pivotTableFilter.setHidden(i, False);
    pivotTableEditor = pivotTable.createPivotTableEditor()
    pivotTableEditor.setFilters(pivotTableFilters)
    pivotTableEditor.apply()
```

6.104.1 Метод PivotTableFilter.getCount

Возвращает количество фильтруемых полей.

Пример:

```
pivotTableFilters = pivotTable.getFilters()
filtersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in filtersEnumerator:
    print(pivotTableFilter.getCount()
```

6.104.2 Метод PivotTableFilter.getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

Пример:

```
pivotTableFilters = pivotTable.getFilters()
filtersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in filtersEnumerator:
    print(pivotTableFilter.getFieldName()
```

6.104.3 Метод PivotTableFilter.getName

Возвращает имя поля для заданного индекса.

Пример:

```
pivotTableFilters = pivotTable.getFilters()
filtersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in filtersEnumerator:
    for filterIndex in range(pivotTableFilter.getCount()):
        print(pivotTableFilter.getName(i)
```

6.104.4 Метод PivotTableFilter.isHidden

Возвращает видимость поля для заданного индекса `itemIndex`. Если `True`, то поле скрыто.

Пример:

```
pivotTableFilters = pivotTable.getFilters()
filtersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in filtersEnumerator:
    for filterIndex in range(pivotTableFilter.getCount()):
        print(pivotTableFilter.istHidden(i)
```

6.104.5 Метод PivotTableFilter.setHidden

Устанавливает видимость поля для заданного индекса. Параметры: `itemName` – индекс поля, `hidden` – видимость (`True` – поле скрыто).

Пример:

```
pivotTableFilters = pivotTable.getFilters()
filtersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in filtersEnumerator:
    for filterIndex in range(pivotTableFilter.getCount()):
        pivotTableFilter.setHidden(i, False)
```

6.105 Класс PivotTableFilters

Класс обеспечивает доступ к списку фильтров. Для получения объекта `PivotTableFilters` используется метод `PivotTable.getFilters()`.

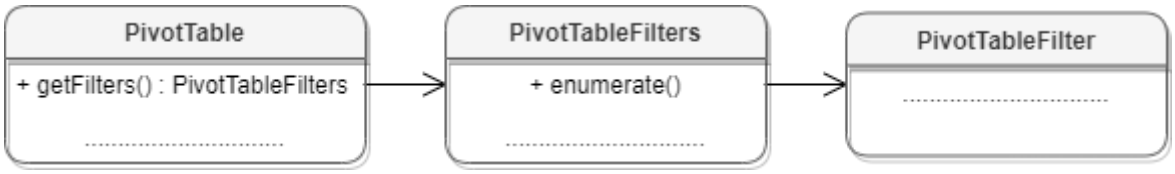


Рисунок 32 – Объектная модель классов для работы с фильтрами

6.105.1 Метод PivotTableFilters.getEnumerator

Метод используется для доступа к коллекции фильтров (см. PivotTableFilter).

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    pivotTableFilters = pivotTable.getFilters()
    filtersEnumerator = pivotTableFilters.getEnumerator()
    for pivotTableFilter in filtersEnumerator:
        print(pivotTableFilter.getFieldName())
```

6.106 Класс PivotTableFunction

Класс PivotTableFunction описывает функции, которые могут быть использованы в сводных таблицах. Описание полей класса представлено в таблице 61. Объект используется в качестве поля subtotalFunctions класса PivotTableCategoryField.

Таблица 61 – Описание полей класса PivotTableFunction

Поле	Описание
PivotTableFunction_Auto	Автозаполнение
PivotTableFunction_Sum	Суммирует все числовые данные
PivotTableFunction_Count	Количество всех ячеек
PivotTableFunction_CountNums	Количество числовых ячеек
PivotTableFunction_Average	Среднее значение
PivotTableFunction_Max	Наибольшее значение
PivotTableFunction_Min	Наименьшее значение
PivotTableFunction_Product	Произведение всех ячеек

Поле	Описание
PivotTableFunction_StdDeviation	Стандартное смещенное отклонение
PivotTableFunction_StdDeviationPopulation	Стандартное несмещенное отклонение
PivotTableFunction_Variance	Смещенная дисперсия
PivotTableFunction_VariancePopulation	Несмещенная дисперсия

6.107 Класс PivotTableItem

PivotTableItem описывает элемент сводной таблицы (см. Рисунок 33). См. пример в главе PivotTableItems.getEnumerator.

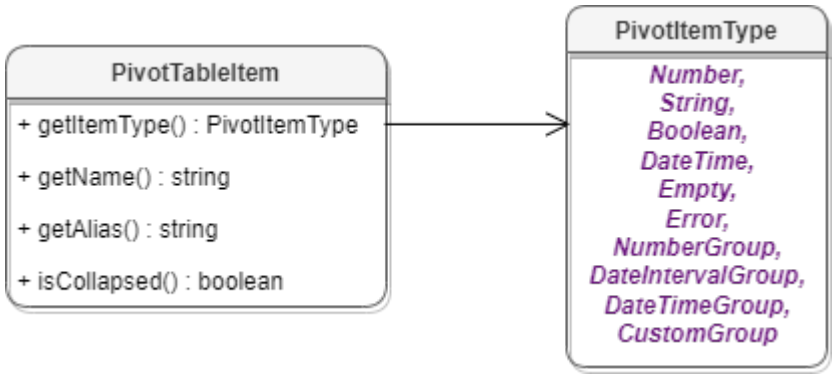


Рисунок 33 – Класс PivotTableItem

6.107.1 Метод PivotTableItem.getAlias

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем), тип - строка. См. пример в главе PivotTableItems.getEnumerator().

6.107.2 Метод PivotTableItem.getItemType

Метод возвращает тип PivotTableItemType элемента сводной таблицы. См. пример в главе PivotTableItems.getEnumerator().

6.107.3 Метод PivotTableItem.getName

Метод возвращает имя элемента сводной таблицы, тип - строка. См. пример в главе PivotTableItems.getEnumerator().

6.107.4 Метод PivotTableItem.isCollapsed

Метод возвращает `True`, если элемент сводной таблицы свернут. См. пример в главе `PivotTableItems.getEnumerator()`.

6.108 Класс PivotTableItems

Класс обеспечивает доступ к списку элементов сводной таблицы (см. Рисунок 34).

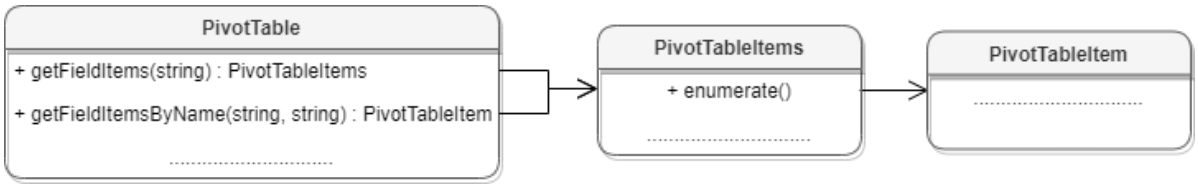


Рисунок 34 – Объектная модель классов для работы с элементами сводных таблиц

6.108.1 Метод PivotTableItems.getEnumerator

Используется для перечисления элементов сводной таблицы.

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    pivotTableItems = pivotTable.getFieldItems("Age")
    pivotTableItemsEnumerator = pivotTableItems.getEnumerator()
    for pivotTableItem in pivotTableItemsEnumerator:
        print(pivotTableItem.GetAlias())
        print(pivotTableItem.GetName())
        print(pivotTableItem.GetItemType())
        print(pivotTableItem.isCollapsed())
```

6.109 Класс PivotTableItemType

Класс `PivotTableItemType` содержит возможные типы элементов сводной таблицы. Описание полей представлено в таблице 62.

Таблица 62 – Описание полей класса `PivotTableItemType`

Поле	Описание
`PivotTableItemType_Number`	Числовой

Поле	Описание
PivotTableItemType_String	Строковый
PivotTableItemType_Boolean	Логический
PivotTableItemType_DateTime	Дата / время
PivotTableItemType_Empty	Пустой тип
PivotTableItemType_Error	Ошибка
PivotTableItemType_NumberGroup	Интервальная группировка
PivotTableItemType_DateIntervalGroup	Интервальная группировка по датам
PivotTableItemType_DateTimeGroup	Группировка по дате / времени
PivotTableItemType_CustomGroup	Пользовательская (произвольная) группировка

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    pivotTableItems = pivotTable.getFieldItems("Age")
    pivotTableItemsEnumerator = pivotTableItems.getEnumerator()
    for pivotTableItem in pivotTableItemsEnumerator:
        print(pivotTableItem.getType())
```

6.110 Класс PivotTableLayoutSettings

Класс `PivotTableLayoutSettings` содержит настройки отображения сводной таблицы. Данный объект может быть получен в результате вызова `PivotTable.getPivotTableLayoutSettings()` и установлен методом `PivotTableEditor.setLayoutSettings()`. Описание полей класса представлено в таблице 63.

Таблица 63 – Описание полей класса `PivotTableLayoutSettings`

Поле	Описание
PivotTableLayoutSettings.reportLayout	Настройка вида макета сводной таблицы (PivotTableReportLayout: компактный, табличный, структурный).
PivotTableLayoutSettings.valueFieldsOrientation	Настраивает положение значений в случае, если в сводной таблице более двух полей значений. Тип - ValueFieldsOrientation.
PivotTableLayoutSettings.pageFieldOrder	Настройка порядка полей фильтров (PageFieldOrder: вниз, затем поперек или сначала поперек, потом вниз).
PivotTableLayoutSettings.indentForCompactLayout	Размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей).
PivotTableLayoutSettings.pageFieldWrapCount	Настройка связана с pageFieldOrder, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д).
PivotTableLayoutSettings.isMergeAndCenterLabelsEnabled	Настройка позволяет объединить ячейки заголовков.
PivotTableLayoutSettings.useGridDropZones	Флаг, отвечающий за отображение классического вида (как в Excel 2003). Влияет только на расположение полей в отчете.
PivotTableLayoutSettings.displayFieldCaptions	Флаг, отвечающий за отображение заголовков полей.

6.111 Класс PivotTablePageField

Содержит свойства поля из области фильтров (см. таблицу 64). Объект может быть получен посредством вызова PivotTable.getPageFields().

Таблица 64 – Описание полей класса PivotTablePageField

Поле	Описание
PivotTablePageField.fieldProperties	Свойства поля PivotTableFieldProperties

6.112 Класс PivotTableReportLayout

Таблица `PivotTableReportLayout` описывает внешний вид отчетов сводной таблицы. Является полем класса `PivotTableLayoutSettings`. Описание полей представлено в таблице 65.

Таблица 65 – Описание полей класса `PivotTableReportLayout`

Поле	Описание
`PivotTableReportLayout_Compact`	Компактный вид
`PivotTableReportLayout_Tabular`	Табличный вид
`PivotTableReportLayout_Outline`	Структурный вид

6.113 Класс PivotTablesManager

Класс `PivotTablesManager` используется для создания сводных таблиц, содержит метод `create()`. Может быть получена вызовом `Document.getPivotTablesManager()`.

Пример:

```
pivotTablesManager = document.getPivotTablesManager()
```

6.113.1 Метод PivotTablesManager.create

Метод создает сводную таблицу `PivotTable` на основе диапазона исходных данных `CellRange`.

Если местоположение не задано, создается новый лист (таблица), и сводная таблица будет расположена по умолчанию.

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cellRange = sheet.getCellRange("B3:C4")
pivotTablesManager = document.getPivotTablesManager()
pivotTable = pivotTablesManager.create(cellRange)
```

6.114 Класс PivotTableUnsupportedFeature

Класс `PivotTableUnsupportedFeature` описывает неподдерживаемую функциональность сводных таблиц. Получение неподдерживаемой функциональности

сводных таблиц описано в PivotTable.getUnsupportedFeatures(). Описание полей таблицы представлено в таблице 66.

Таблица 66 – Описание полей класса PivotTableUnsupportedFeature

Поле	Описание
PivotTableUnsupportedFeature_CalculatedField	Вычисляемые поля
PivotTableUnsupportedFeature_CalculatedItem	Вычисляемые элементы
PivotTableUnsupportedFeature_CollapsedValues	Свернутые поля
PivotTableUnsupportedFeature_ShowDataAs	Вычисления ("Show data" как в MS Excel)

6.115 Класс PivotTableUpdateResult

В таблице 67 приведены константы, которые соответствуют возможным результатам обновления сводной таблицы (см. методы PivotTable.update(), PivotTableEditor.apply()).

Таблица 67 – Результаты обновления сводной таблицы

Наименование константы	Описание
PivotTableUpdateResult_Success	Успешное обновление таблицы
PivotTableUpdateResult_NoPivotTable	Сводная таблица не найдена
PivotTableUpdateResult_NoSuchFieldInCategory	Не найдено поле в категории
PivotTableUpdateResult_NoSuchFieldInPivotTable	Не найдено поле в сводной таблице
PivotTableUpdateResult_InvalidIndex	Ошибка в индексе
PivotTableUpdateResult_FieldAlreadyEnabled	Поле уже существует
PivotTableUpdateResult_MovingFieldToTheSameCategoryForbidden	Попытка перемещения поля в рамках текущей категории

Наименование константы	Описание
PivotTableUpdateResult_InvalidFunction	Неправильная функция
PivotTableUpdateResult_InvalidCategory	Неправильная область
PivotTableUpdateResult_InvalidDataSourceRange	Ошибка диапазона исходных данных
PivotTableUpdateResult_NoDataRowsInDataSource	В исходных данных нет строк с данными
PivotTableUpdateResult_EmptyDataSourceHeaders	Пустые заголовки исходных данных
PivotTableUpdateResult_NoReferenceUnderDefine	Попытка обновить или создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу
PivotTableUpdateResult_NoSuchItem	Элемент не найден
PivotTableUpdateResult_CannotExpandCollapseLeafItem	Не удается раскрыть свернутый элемент
PivotTableUpdateResult_AnotherPivotInsideDataSource	Найдена другая сводная таблица в этом же диапазоне
PivotTableUpdateResult_Canceled	Обновление сводной таблицы отменено

6.116 Класс PivotTableValueField

`PivotTableValueField` содержит свойства поля сводной таблицы, использующегося как значение столбец (см. таблицу 68). Объект класса может быть получен посредством вызова `PivotTable.getValueFields()`.

Таблица 68 – Описание полей класса `PivotTableValueField`

Поле	Описание
PivotTableValueField.baseFieldName	Оригинальное поле на основе которого было создано данное поле, тип - строка.
PivotTableValueField.valueFieldName	Автоматический уникальный псевдоним такой как “Sum of %имя поля%“, тип – строка.
PivotTableValueField.cellNumberFormat	Числовой формат типа `CellFormat` для конкретного поля значений.
PivotTableValueField.totalFunction	Агрегирующая функция `PivotTableFunction` поля значений (SUM, COUNT, MAX и т.д).

Поле	Описание
PivotTableValueField.customFormula	Вычисляемая формула для поля значений, тип - строка.

6.117 Класс PointU

Класс PointU представляет точку двухмерном пространстве. Описание полей класса PointU представлено в таблице 69.

Таблица 69 – Описание полей класса PointU

Поле	Тип	Описание
PointU.x	number	Координата точки по оси x
PointU.y	number	Координата точки по оси y

Пример:

```
point = myOfficeSDK.PointU(50, 50)
print("x=", point.x, ", height=", point.y) # x= 50.0 , height= 50.0
```

6.117.1 PointU.toString

Возвращает информацию о координатах точки в виде строкового значения формата (x: <value>, y: <value>).

Пример:

```
point = myOfficeSDK.PointU(50, 50)
print(point.toString()) # (x: 50.0, y: 50.0)
```

6.118 Класс Position

Класс Position представляет местоположение в текстовом документе. Используется для обозначения начала и конца диапазона Range.

6.118.1 Метод Position.getCell

Метод возвращает ячейку Cell для заданной позиции.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("A3")
rng = cell.getRange()
position = rne.getBegin()
cellAtPosition = position.getCell()
```

6.118.2 Метод Position.insertBookmark

Вставляет закладку с наименованием в текущую позицию.

Пример:

```
document.getRange().getEnd().insertBookmark("Bookmark example")
```

6.118.3 Метод Position.insertHyperlink

Метод `insertHyperlink` вставляет ссылку в текущую позицию. В качестве параметров передаются адрес ссылки и текст ссылки.

Параметры:

– `url` – адрес ссылки;

– `label` – текст ссылки.

Пример:

```
document.getRange().getBegin().insertHyperlink("https://testhyperlink.com",
"Hyperlink")
```

6.118.4 Метод Position.insertImage

Вставляет изображение из файла в текущую позицию. Возвращает объект `Image`, содержащий вставленное изображение.

Параметры:

– `url` – полный путь к файлу, строка;

– `size` – геометрические размеры изображения для вставки, тип `SizeU`.

Пример:

```
insertedImage = document.getRange().getBegin().insertImage("C://Tmp//123.jpg",
myOfficeSDK.SizeU(100, 100))
```

6.118.5 Метод Position.insertLineBreak

Метод предназначен для вставки перевода строки.

Пример:

```
docRange = document.getRange()
endPosition = docRange.getEnd()
endPosition.insertLineBreak()
```

6.118.6 Метод Position.insertPageBreak

Метод предназначен для вставки разрыва страницы в указанную позицию в документе.

Пример:

```
docRange = document.getRange()
endPosition = docRange.getEnd()
endPosition.insertPageBreak()
```

6.118.7 Метод Position.insertSectionBreak

Вставляет разрыв раздела в текущую позицию.

Пример:

```
docRange = document.getRange()
endPosition = docRange.getEnd()
endPosition.insertSectionBreak()
```

6.118.8 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

```
table = position.insertTable(3, 3, "Table")
```

приведет к созданию в текстовом документе таблицы с именем «Table1».

Пример вставки таблицы в текстовый документ:

```
range = document.getRange()
beginPosition = range.getBegin()
table = beginPosition.insertTable(3, 3, "Table")
```

В табличном документе данный метод используется для вставки нового рабочего листа.

Пример вставки нового листа в табличный документ:

```
range = document.getRange()
endPosition = range.getEnd()
table = endPosition.insertTable(3, 3, "Table")
```

6.118.9 Метод Position.insertText

Метод предназначен для вставки текстовой строки в заданное местоположение в документе.

Пример:

```
docRange = document.getRange()
startPosition = docRange.getBegin()
startPosition.insertText("Текст в начале строки")
```

6.118.10 Метод Position.removeBackward

Метод удаляет заданное количество объектов (символов, картинок и т.д.) до текущей позиции.

Пример:

```
docRange = document.getRange()
beginPosition = docRange.getBegin()
beginPosition.removeBackward(3)
```

6.118.11 Метод Position.removeForward

Метод удаляет заданное количество объектов (символов, картинок и т.д.) после текущей позиции.

Пример:

```
docRange = document.getRange()
beginPosition = docRange.getBegin()
beginPosition.removeForward(3)
```

6.118.12 Positon.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Position`.

Пример:

```
firstPosition = document.getRange().getBegin()
lastPosition = document.getRange().getBegin()

if firstPosition.__eq__(lastPosition):
    print("Equals");
```

6.118.13 Position.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Position`.

Пример:

```
firstPosition = document.getRange().getBegin()
lastPosition = document.getRange().getEnd()

if firstPosition.__ne__(lastPosition):
    print("Not equals");
```

6.119 PresentationExportSettings

Класс `PresentationExportSettings` предоставляет настройки, необходимые для экспорта презентационных документов (см. `Document.exportAs`). Поле объекта `PresentationExportSettings.skipHiddenSlides` позволяет включить в созданный документ скрытые слайды.

Пример:

```
presentationExportSettings = myOfficeSDK.PresentationExportSettings()
presentationExportSettings.skipHiddenSlides = False
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1,
presentationExportSettings)
```

6.120 Класс PrintingScope

Класс `PrintingScope` содержит настройки для экспорта табличных документов. Используется в поле `printingScope` класса `WorkbookExportSettings`.

Позволяет создать области печати следующих типов:

- выбранная область печати либо весь документ (см. `PrintingScope.Type`);
- указанный диапазон ячеек (см. `CellRangePosition`).

Примеры:

```
# по умолчанию - PrintingScope.Type.PrintArea
printingScope = myOfficeSDK.PrintingScope()
```

```
# область печати
printingScope =
myOfficeSDK.PrintingScope(myOfficeSDK.PrintingScope.Type_PrintArea)
```

```
# диапазон ячеек
cellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 5, 5)
printingScope = myOfficeSDK.PrintingScope(cellRangePosition)
```

6.120.1 Метод PrintingScope.getCellRange

Метод возвращает диапазон ячеек таблицы (см. CellRangePosition).

6.120.2 Метод PrintingScope.usePrintArea

Метод возвращает True, если область печати должна использоваться во время печати, экспорта и т. д.

6.121 Класс PrintingScope.Type

Варианты выбора диапазона страниц для экспорта и печати представлены в таблице 70. Используется в PrintingScope

Таблица 70 – Диапазон страниц для экспорта и печати

Наименование константы	Описание
PrintingScope.Type_PrintArea	Выбранная область печати (по умолчанию).
PrintingScope.Type_WholeSheet	Печать всего документа.

Пример:

```
printingScope =
myOfficeSDK.PrintingScope(myOfficeSDK.PrintingScope.Type_PrintArea)
```

6.122 Класс Range

Класс Range предоставляет доступ к диапазону документа. На рисунке 35 изображена объектная модель классов, относящихся к работе с диапазонами.

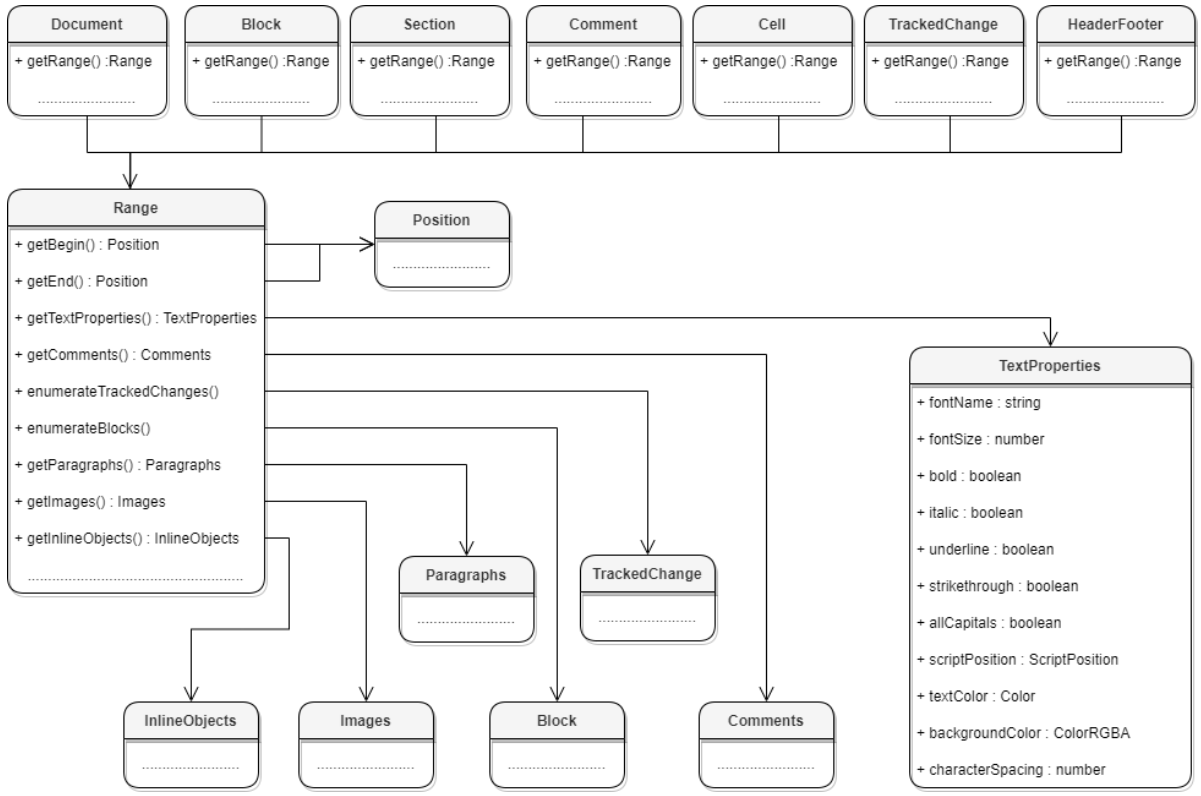


Рисунок 35 – Объектная модель для работы с классом `Range`

Варианты получения диапазона для текстового документа:

```
# диапазон всего документа
docRange = document.getRange()

# диапазон блока
block = document.getBlocks().getBlock(0)
if block != None:
    blockRange = block.getRange()

# диапазон секций
sections = document.getSections()
sectionsEnumerator = sections.getEnumerator()
for section in sectionsEnumerator:
    print(section.getRange().extractText())

# диапазон комментариев
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    print(comment.getRange().extractText())
```

```
# диапазон ячейки
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()

# диапазон верхних колонтитулов
block = document.getBlocks().getBlock(0)
if block != None:
    section = block.getSection()
    headers = section.getHeaders()
    headersEnumerator = headers.getEnumerator()
    for header in headersEnumerator:
        print(header.getRange().extractText())

# диапазон отслеживаемых изменений
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
for trackedChange in trackedChangesEnumerator:
    print(trackedChange.getRange().extractText())
```

6.122.1 Конструктор Range

Для создания объекта Range вы можете использовать следующий конструктор:

```
documentRange = myOfficeSDK.Range(begin, end)
```

Параметры:

– begin: начальная позиция диапазона, тип Position;

– end: конечная позиция диапазона, тип Position.

6.122.2 Метод Range.extractText

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и другие объекты игнорируются.

Пример для текстового документа:

```
docRange = document.getRange()
print(docRange.ExtractText())
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    print(cellRange.ExtractText())
```

6.122.3 Метод Range.getBegin

Метод возвращает позицию в начале диапазона.

Пример для текстового документа:

```
beginDocPosition = document.getRange().getBegin()
beginDocPosition.insertText("API")
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    beginDocPos = cellRange.getBegin()
    beginDocPos.insertText("API")
```

6.122.4 Метод Range.getBlocksEnumerator

Предоставляет возможность итерации по блокам.

Пример для текстового документа:

```
docRange = document.getRange()
blocksEnumerator = docRange.getBlocksEnumerator()
for block in blocksEnumerator:
    print(block.getRange().extractText())
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    blocksEnumerator = cellRange.getBlocksEnumerator()
    for block in blocksEnumerator:
        print(block.getRange().extractText())
```

6.122.5 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:

```
comments = document.getRange().getComments()
commentsEnumerator = comments.getEnumerator()
for comment in commentsEnumerator:
    print(comment.getText())
```

6.122.6 Метод Range.getEnd

Метод возвращает позицию в конце диапазона, не включая последний символ paragraph mark.

Пример для текстового документа:

```
beginDocPosition = document.getRange().getEnd()
beginDocPosition.insertText("API")
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    endDocPos = cellRange.getEnd()
    endDocPos.insertText("API")
```

6.122.7 Метод Range.getImages

Обеспечивает доступ к изображениям (Image) в диапазоне.

Примеры:

```
images = document.getRange().getImages()
imagesEnumerator = images.getEnumerator()
for image in imagesEnumerator:
    print(image.getFrame().getWrapType())
```

6.122.8 Метод Range.getInlineObjects

Обеспечивает доступ к перечислению MediaObjects графических объектов диапазона.

Пример:

```
docRange = document.getRange()
mediaObjects = docRange.getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    print(mediaObject.getFrame().getWrapType())
```

6.122.9 Метод Range.getParagraphs

Обеспечивает доступ к абзацам Paragraphs в диапазоне.

Пример для текстового документа:

```
paragraphs = document.getRange().getParagraphs()
paragraphsEnumerator = paragraphs.getEnumerator()
for paragraph in paragraphsEnumerator:
    print(paragraph.getRange().extractText())
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    paragraphs = cellRange.getParagraphs();
    paragraphsEnumerator = paragraphs.getEnumerator()
    for paragraph in paragraphsEnumerator:
        print(paragraph.getRange().extractText())
```

6.122.10 Метод Range.getTextProperties

Метод возвращает объект с текущими настройками форматирования для фрагмента текстового документа. Описание настроек форматирования осуществляется с помощью объекта TextProperties.

Пример для текстового документа:

```
docRange = document.getRange()
textProperties = docRange.getTextProperties()
print(textProperties.fontName)
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
```

```
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    textProperties = cellRange.getTextProperties()
    print(textProperties.fontName)
```

6.122.11 Метод Range.getTrackedChangesEnumerator

Предоставляет возможность итерации по отслеживаемым изменениям TrackedChange. Метод может быть использован только в текстовых документах.

Пример:

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
for trackedChange in trackedChangesEnumerator:
    print(trackedChange.getRange().extractText())
```

6.122.12 Метод Range.isContentLocked

Метод возвращает значение `True`, если изменения содержимого диапазона запрещены.

Пример для текстового документа:

```
docRange = document.getRange()
print(docRange.isContentLocked())
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    print(cellRange.isContentLocked())
```

6.122.13 Метод Range.lockContent

Метод запрещает изменения содержимого диапазона.

Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа:

```
docRange = document.getRange()
docRange.lockContent()
print(docRange.isContentLocked())
```

Пример для таблицы внутри текстового документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    cellRange.lockContent()
    print(cellRange.isContentLocked())
```

6.122.14 Метод Range.removeContent

Метод полностью удаляет содержимое диапазона.

Пример для текстового документа:

```
docRange = document.getRange()
docRange.removeContent();
print(docRange.ExtractText())
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    cellRange.removeContent()
    print(cellRange.ExtractText())
```

6.122.15 Метод Range.replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример для текстового документа:

```
docRange = document.getRange()
docRange.replaceText("Range text")
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    cellRange.replaceText("New text")
```

6.122.16 Метод Range.setHyperlink

Метод `setHyperlink` вставляет ссылку в содержимое диапазона и заменяет его текст текстом ссылки.

Параметры:

– `url` – адрес ссылки;

– `label` – текст ссылки.

Пример для текстового документа:

```
docRange = document.getRange()
docRange.setHyperlink("https://testhyperlink.com", "Hyperlink")
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    cellRange.setHyperlink("https://testhyperlink.com", "Hyperlink")
```

6.122.17 Метод Range.setTextProperties

Метод применяет настройки форматирования `TextProperties` для диапазона.

Пример для текстового документа:

```
docRange = document.getRange()
textProperties = docRange.getTextProperties()
textProperties.fontName = "Arial"
docRange.setTextProperties(textProperties)
```

Пример для табличного документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    textProperties = cellRange.getTextProperties()
    textProperties.fontName = "Arial"
    cellRange.setTextProperties(textProperties)
```

6.122.18 Метод Range.unlockContent

Метод разрешает изменения содержимого диапазона.

Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа:

```
docRange = document.getRange()
docRange.unlockContent()
print(docRange.isContentLocked())
```

Пример для таблицы внутри текстового документа:

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    cellRange.unlockContent()
    print(cellRange.isContentLocked())
```

6.122.19 Метод Range.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Range`.

Пример:

```
table = document.getBlocks().getTable(0)
if table != None:
    firstRange = table.getCell("A1").getRange()
    secondRange = table.getCell("A1").getRange()
    if firstRange.__eq__(secondRange):
        print("Equals")
```

6.122.20 Метод Range.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Range`.

Пример:

```
table = document.getBlocks().getTable(0)
if table != None:
    firstRange = table.getCell("A1").getRange()
```

```
    secondRange = table.getCell("A2").getRange()
    if firstRange.__ne__(secondRange):
        print("Not equals")
```

6.123 Класс RangeBorders

Класс `RangeBorders` оставлен для совместимости. Вместо него необходимо использовать класс `Borders`.

6.124 Класс RectU

Класс `RectU` описывает прямоугольник в двухмерном пространстве. Описание полей таблицы `RectU` представлено в таблице 71.

Таблица 71 – Описание полей класса `RectU`

Поле	Тип	Описание
`RectU.topLeft`	number	Координата левой верхней вершины прямоугольника
`RectU.bottomRight`	number	Координата правой нижней вершины прямоугольника

Пример:

```
rect = myOfficeSDK.RectU(0, 0, 50, 50)
print("topLeft.x=", rect.topLeft.x, ", topLeft.y=", rect.topLeft.y) # x= 50.0 ,
height= 50.0
```

6.124.1 RectU.toString

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

Пример:

```
rect = myOfficeSDK.RectU(0, 0, 50, 50)
print(rect.toString()) # [topLeft: (x: 0.0, y: 0.0), bottomRight: (x: 50.0, y:
50.0)]
```

6.125 Класс SaveDocumentSettings

Класс `SaveDocumentSettings` предоставляет настройки, используемые для сохранения документа в файл, см. `Document.saveAs()`. Описание полей класса `SaveDocumentSettings` представлено в таблице 72.

Таблица 72 – Описание полей класса SaveDocumentSettings

Поле	Описание
SaveDocumentSettings_documentFormat	Формат документа DocumentFormat
SaveDocumentSettings_documentType	Тип документа DocumentType
SaveDocumentSettings_documentPassword	Пароль для защиты электронного документа от несанкционированного доступа
SaveDocumentSettings_isTemplate	Флаг, обозначающий, что документ должен быть сохранен как шаблон
SaveDocumentSettings_dsvSettings	Структура DSVSettings, необходимая для сохранения в формате DSV

6.126 Класс ScaleFrom

В таблице 73 представлены позиции якоря при масштабировании объекта AbsoluteFrame. Используется в AbsoluteFrame.scale().

Таблица 73 – Неизменные позиции объекта при масштабировании

Наименование константы	Позиция
ScaleFrom_BottomRight	Правый нижний угол
ScaleFrom_BottomLeft	Левый нижний угол
ScaleFrom_TopLeft	Левый верхний угол
ScaleFrom_TopRight	Правый верхний угол

6.127 Класс ScientificCellFormatting

Класс содержит параметры для экспоненциального формата ячеек таблицы. Используется в качестве аргумента метода Cell.setFormat(). Описание полей класса ScientificCellFormatting представлено в таблице 74.

Таблица 74 – Описание полей класса ScientificCellFormatting

Поле	Описание
ScientificCellFormatting.decimalPlaces	Количество десятичных позиций

Поле	Описание
`ScientificCellFormatting.minExponentDigits`	Минимальное количество позиций экспоненты

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1")
cell = firstSheet.getCell("A2")

scientificCellFormat = myOfficeSDK.ScientificCellFormatting()
scientificCellFormat.decimalPlaces = 2;
scientificCellFormat.minExponentDigits = 3;

cell.setFormat(scientificCellFormat)
print(cell.getFormattedValue())
```

6.128 Класс Script

Класс `Script` предназначен для управления отдельной макрокомандой. Содержит поля `Name` и `Body`.

6.128.1 Метод Script.getBody

Метод возвращает текст макрокоманды в виде строки.

Пример:

```
scripts = document.getScripts()
enumerator = scripts.getEnumerator()
for scriptIndex, script in enumerate(enumerator):
    print(script.getBody())
```

6.128.2 Метод Script.getName

Метод возвращает имя макрокоманды.

Пример:

```
scripts = document.getScripts()
enumerator = scripts.getEnumerator()
for scriptIndex, script in enumerate(enumerator):
    print(script.getName())
```

6.128.3 Метод Script.setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

Пример:

```
scripts = document.getScripts()

scriptName = "Enumerate scripts for document"
scriptBody = "local scripts = document:getScripts()\nfor script in
scripts:enumerate() do\nprint(script:getName())\nend"
scripts.setScript(scriptName, scriptBody)

script = scripts.getScript(scriptName)
if script != None:
    newScriptBody = "local scripts = document:getScripts()"
    script.setBody(newScriptBody)
    print("Script body was changed to '" + script.getBody() + "'")
```

6.128.4 Метод Script.setName

Метод устанавливает имя для макрокоманды.

Пример:

```
scripts = document.getScripts()

scriptName = "Enumerate scripts for document"
scriptCode = "local scripts = document:getScripts()"
scripts.setScript(scriptName, scriptCode)

script = scripts.getScript(scriptName)
if script != None:
    newScriptName = "New script name"
    script.setName(newScriptName)
    print("Script was renamed to '" + script.getName() + "'")
```

6.129 Класс Scripting

Объект класса `Scripting` может быть получен путем вызова `DocumentAPI.createScripting(document)`, и содержит метод `runScript`, который используется для запуска макрокоманды.

Пример:

```
scripting = myOfficeSDK.createScripting(document)
```

6.129.1 Метод Scripting.runScript

Запускает макрокоманду с указанным именем. В случае невозможности запуска макрокоманды вызывает исключение ScriptExecutionError.

Пример:

```
try:
    scripting = myOfficeSDK.createScripting(document);
    scripting.runScript("New script")
except myOfficeSDK.ScriptExecutionError as err:
    print("Error execution script: ", err)
```

6.130 Класс ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 75. Используется в качестве поля scriptPosition класса TextProperties.

Таблица 75 – Типы надстрочного и подстрочного форматирования

Наименование константы	Описание
ScriptPosition_SuperScript	Надстрочный знак (верхний индекс)
ScriptPosition_SubScript	Подстрочный знак (нижний индекс)
ScriptPosition_NoramlScript	Без указания индекса

Пример:

```
textProperties = myOfficeSDK.TextProperties()
textProperties.scriptPosition = myOfficeSDK.ScriptPosition_NormalScript
document.getRange().setTextProperties(textProperties)
```

6.131 Класс Scripts

Класс Scripts предоставляет доступ к списку макрокоманд документа. Коллекцию макрокоманд Scripts можно получить из документа посредством вызова метода Document.getScripts().

Пример:

```
scripts = document.getScripts()
enumerator = scripts.getEnumerator()
for scriptIndex, script in enumerate(enumerator):
    print(script.getName())
```

6.131.1 Метод Scripts.getEnumerator

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

Пример:

```
scripts = document.getScripts()
enumerator = scripts.getEnumerator()
for scriptIndex, script in enumerate(enumerator):
    print(script.getName())
```

6.131.2 Метод Scripts.getScript

Метод возвращает объект класса Script, описывающий макрокоманду. В качестве аргумента используется имя макрокоманды.

Пример:

```
scripts = document.getScripts()
script = scripts.getScript("ScriptName")
if script == None:
    print("Script not found")
else:
    print("Script body:" + script.getBody())
```

6.131.3 Метод Scripts.removeScript

Метод удаляет макрокоманду из текущего документа. В качестве аргумента используется имя макрокоманды.

Пример:

```
scripts = document.getScripts()
scriptName = "Enumerate scripts for document"
script = scripts.removeScript(scriptName)
script = scripts.getScript(scriptName)
if script == None:
    print("Script was removed")
```

6.131.4 Метод Scripts.setScript

Метод добавляет макрокоманду в текущий документ. Если макрокоманда с таким именем уже существует, будет обновлено ее содержимое.

Пример:

```
scripts = document.getScripts()


scriptName = "Enumerate scripts for document"
scriptCode = "local scripts = document:getScripts()\nfor script in
```

```
scripts:enumerate() do\nprint(script:getName())\nend"
scripts.setScript(scriptName, scriptCode)

script = scripts.getScript(scriptName)
if script == None:
    print("Script not found")
else:
    print("Script was added")
```

6.132 Класс Search

Класс `Search` предоставляет доступ к механизму поиска и замены фрагментов документа, открытого в редакторе текста или таблиц.

6.132.1 Метод Search.findText

Метод выполняет поиск строки с учетом и без учета регистра:

- во всем документе;
- в выбранном диапазоне;
- в выбранном диапазоне ячеек;
- в таблице.

Результат возвращается в виде диапазона `Range`, содержащего искомый фрагмент.

Если строка не обнаружена, возвращается пустой диапазон.

Возможно использование следующих вариантов метода:

```
RangesIterator findText(String text)
RangesIterator findText(String text, CaseSensitive caseSensitive)
RangesIterator findText(String text, Range range)
RangesIterator findText(String text, Range range, CaseSensitive caseSensitive)
RangesIterator findText(String text, CellRange cellRange)
RangesIterator findText(String text, CellRange cellRange, CaseSensitive
caseSensitive)
RangesIterator findText(String text, Table table)
RangesIterator findText(String text, Table table, CaseSensitive caseSensitive)
```

Параметры:

- `text` – текст для поиска;
- `caseSensitive` – регистр поиска , тип `CaseSensitive`;
- `range` – диапазон поиска, тип `Range`;
- `cellRange` – диапазон ячеек поиска, тип `CellRange`;

– `table` – таблица для поиска, тип `Table`.

Пример:

```
# Поиск по всему документу
search = myOfficeSDK.createSearch(document)
searchResult = search.findText("Yellow", myOfficeSDK.CaseSensitive_Yes)
// Отображение результатов поиска
for searchRange in searchResult:
    print(searchRange.extractText())
```

Дополнительные примеры использования метода `Search.findText` приведены в разделе Поиск в документе.

6.133 Класс Section

Класс `Section` представляет собой раздел в документе.

6.133.1 Метод Section.getFooters

Метод возвращает коллекцию `HeadersFooters` нижних колонтитулов раздела.

Пример:

```
for section in sectionsEnumerator:
    footers = section.getFooters()
    for footer in footers:
        print(footer.getRange().extractText())
```

6.133.2 Метод Section.getHeaders

Метод возвращает коллекцию `HeadersFooters` верхних колонтитулов раздела.

Пример:

```
for section in sectionsEnumerator:
    headers = section.getHeaders()
    for header in headers:
        print(header.getRange().extractText())
```

6.133.3 Метод Section.getPageOrientation

Метод возвращает ориентацию страниц раздела.

Пример:

```
for section in sectionsEnumerator:
    print(section.getPageOrientation())
```

6.133.4 Метод Section.getPageProperties

Метод возвращает параметры страниц раздела `PageProperties`.

Пример:

```
for section in sectionsEnumerator:
    pageProperties = section.getPageProperties()
    print(pageProperties.height)
```

6.133.5 Метод Section.getRange

Метод возвращает диапазон `Range` в документе, соответствующий данному разделу.

Пример:

```
for section in sectionsEnumerator:
    sectionRange = section.getRange()
    print(sectionRange.extractText())
```

6.133.6 Метод Section.setPageOrientation

Метод задает ориентацию страниц раздела.

Пример:

```
for section in sectionsEnumerator:
    section.setPageOrientation(myOfficeSDK.PageOrientation_Portrait)
    print(section.getPageOrientation())
```

6.133.7 Метод Section.setPageProperties

Метод устанавливает параметры `PageProperties` страниц, находящихся в разделе.

Пример:

```
for section in sectionsEnumerator:
    section.setPageProperties(myOfficeSDK.PageProperties(100, 50))
    pageProperties = section.getPageProperties()
    print(pageProperties.height)
```

6.134 Класс Sections

Класс `Sections` используется для доступа к коллекции секций документа. Описание секции см. в разделе `Section`.

6.134.1 Метод Sections.getEnumerator

Метод позволяет перечислить коллекцию секций документа.

Пример:

```
sectionsEnumerator = document.getSectionsEnumerator()
for section in sectionsEnumerator:
    sectionRange = section.getRange()
    print(sectionRange.extractText())
```

6.135 Класс Shape

Класс `Shape` представляет собой фигуру, содержит методы для установки и получения свойств `ShapeProperties`.

6.135.1 Метод Shape.getShapeProperties

Метод возвращает свойства фигуры `ShapeProperties`.

Пример:

```
shape = document.getBlocks().getShape(0)
if shape != None:
    shapeProperties = shape.getShapeProperties()
    print(shapeProperties.verticalAlignment)
```

6.135.2 Метод Shape.setShapeProperties

Метод устанавливает свойства фигуры `ShapeProperties`.

Пример:

```
shape = document.getBlocks().getShape(0)
if shape != None:
    shapeProperties = shape.getShapeProperties()
    shapeProperties.verticalAlignment = myOfficeSDK.VerticalAlignment.Center
    shape.setShapeProperties(shapeProperties)
```

6.136 Класс ShapeProperties

Класс описывает свойства фигуры и содержит следующие поля:

– `verticalAlignment` - вертикальное выравнивание, тип `VerticalAlignment`;

– `borderProperties` - свойства границ фигуры, тип `LineProperties`;

– `fill` - свойства заполнения фигуры, тип `Fill`;

– `shapeTextLayout` - свойства текста внутри фигуры, тип `ShapeTextLayout`.

6.136.1 Поле ShapeProperties.borderProperties

Поле предназначено для установки свойств границ фигуры `LineProperties`.

6.136.2 Поле ShapeProperties.fill

Поле предназначено для установки свойств заполнения фигуры `Fill` .

6.136.3 Поле ShapeProperties.shapeTextLayout

Поле предназначено для установки свойств текста внутри фигуры `ShapeTextLayout`.

6.136.4 Поле ShapeProperties.verticalAlignment

Поле предназначено для установки типа вертикального выравнивания `VerticalAlignment`.

6.137 Класс ShapeTextLayout

Класс `ShapeTextLayout` описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 76. Используется в качестве поля класса `ShapeProperties`.

Таблица 76 – Описание полей класса `ShapeTextLayout`

Поле	Описание
`ShapeTextLayout_DoNotAutoFit`	Размещение текста в фигуре по умолчанию
`ShapeTextLayout_FitShapeExtentToText`	Расширение фигуры под текст
`ShapeTextLayout_FitTextToShape`	Заполнение фигуры текстом

6.138 Класс SizeU

Класс `SizeU` представляет размер объекта в двухмерном пространстве. Описание полей класса `SizeU` представлено в таблице 77.

Таблица 77 – Описание полей классе `SizeU`

Поле	Тип	Описание
`SizeU.width`	number	Ширина
`SizeU.height`	number	Высота

Пример:

```
size = myOfficeSDK.SizeU(2, 3)
print("width=", size.width, ", height=", size.height) # width= 2.0 , height= 3.0
```

6.138.1 Метод SizeU.toString

Возвращает информацию о размерах в виде строкового значения формата `(width: <value>, height: <value>)`.

Пример:

```
size = myOfficeSDK.SizeU(2, 3)
print(size.toString()) # (width: 2.0, height: 3.0)
```

6.139 Класс Table

Класс `Table` предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 36).

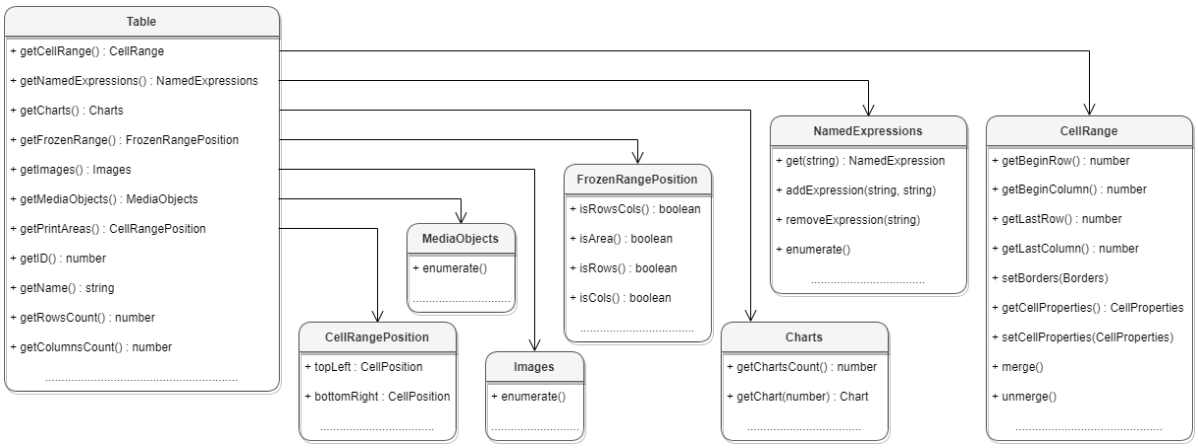


Рисунок 36 – Структура полей класса `Table`

6.139.1 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.

6.139.2 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.

6.139.3 Метод Table.createFiltersRange

Метод `Table.createFiltersRange` задает диапазон, который используется как диапазон фильтрации.

```
FiltersRange createFiltersRange(const CellRangePosition& range);
```

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.

Метод может формировать следующие исключения:

- `NotImplementedError`: метод вызывается для неподдерживаемого документа
- `IncorrectArgumentError`: диапазон слишком мал или имеет недопустимые элементы

– DocumentModificationError: диапазон пересекает объединенные ячейки или содержит сводную таблицу
– OutOfRangeError: диапазон находится за пределами таблицы

Пример:

```
sheet = document.getBlocks().getTable("Лист1")
cellRange = myOfficeSDK.CellRangePosition(1, 1, 8, 2)
filtersRange = sheet.createFiltersRange(cellRange)
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.139.4 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:

```
table = document.getBlocks().getTable(0)
table.duplicate()
```

6.139.5 Метод Table.freeze

Метод `freeze` закрепляет заданную область `FrozenRangePosition` таблицы.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenCols(0, 2)
firstSheet.freeze(frozenRangePosition)
print(firstSheet.getFrozenRange().isCols())
```

6.139.6 Метод Table.getCell

Метод позволяет получить доступ к отдельной ячейке таблицы. В качестве аргумента может выступать текстовое представление адреса ячейки, либо экземпляр класса `CellPosition`.

Примеры:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("B2")
print(cell.getFormattedValue())
```

```
firstSheet = document.getBlocks().getTable(0)
cellPosition = myOfficeSDK.CellPosition(2, 1)
cell = firstSheet.getCell(cellPosition)
print(cell.getFormattedValue())
```

6.139.7 Метод Table.getCellRange

Метод позволяет получить доступ к диапазону ячеек таблицы CellRange. В качестве аргумента может использоваться строка, описывающая диапазон ("B3:C4"), либо объект типа CellRangePosition.

Примеры:

```
firstSheet = document.getBlocks().getTable("Table1")
cellRange = firstSheet.getCellRange("B3:C4")
cellRangesEnumerator = cellRange.getEnumerator()
for cell in cellRangesEnumerator:
    print(cell.getFormattedValue())
```

```
firstSheet = document.getBlocks().getTable("Table1")
cellRangePosition = myOfficeSDK.CellRangePosition(0, 0, 1, 1)
cellRange = firstSheet.getCellRange(cellRangePosition)
```

6.139.8 Метод Table.getCharts

Для получения списка диаграмм (Charts) таблицы используется метод Table.getCharts.

Пример:

```
tablesEnumerator = document.getBlocks().getTablesEnumerator()
for table in tablesEnumerator:
    charts = table.getCharts()
    print(charts.getChartsCount())
```

6.139.9 Метод Table.getColumnsCount

Метод позволяет получить количество столбцов таблицы.

Пример:

```
table = document.getBlocks().getTable("List11")
print(table.getColumnsCount())
```

6.139.10 Метод Table.getFiltersRange

Метод `Table.getFiltersRange` возвращает текущий диапазон фильтрации, принадлежащий таблице. Рабочий лист табличного документа может содержать только один диапазон фильтрации.

```
FiltersRange getFiltersRange()
```

Метод возвращает `FiltersRange`, если диапазон фильтрации существует.

Метод может быть использован только в табличных документах.

При использовании данного метода может произойти исключение `InvalidObjectError` в случае, если диапазон недействителен.

Пример:

```
filtersRange = sheet.getFiltersRange()
cellRange = filtersRange.getCellRange()
print(cellRange.getBeginRow())
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.139.11 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 `FrozenRangePosition`.

Пример:

```
table = document.getBlocks().getTable(0)
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenCols(0, 2)
table.freeze(frozenRangePosition)
print(table.getFrozenRange().isCols())
```

6.139.12 Метод Table.getImages

Для получения списка изображений (`Images`) таблицы используется метод `Table.getImages`.

Пример:

```
tablesEnumerator = document.getBlocks().getTablesEnumerator()
for table in tablesEnumerator:
```

```
    imagesEnumerator = table.getRange().getImages().getEnumerator()
    for image in imagesEnumerator:
        print(image.getFrame().getWrapType())
```

6.139.13 Метод Table.getMediaObjects

Метод используется для получения списка медиаобъектов MediaObjects.

Пример:

```
table = document.getBlocks().getTable(0)
mediaObjects = table.getMediaObjects()
for mediaObject in mediaObjects:
    print(mediaObject)
```

6.139.14 Метод Table.getName

Метод позволяет получить наименование листа табличного документа.

Пример:

```
table = document.getBlocks().getTable("List11")
print(table.getName())
```

6.139.15 Метод Table.getNamedExpressions

Для получения списка именованных диапазонов NamedExpressions используется метод Table.getNamedExpressions().

Пример:

```
tablesEnumerator = document.getBlocks().getTablesEnumerator()
for table in tablesEnumerator:
    namedExpressions = table.getNamedExpressions()
    namedExpressionsEnumerator = namedExpressions.getEnumerator()
    for namedExpression in namedExpressionsEnumerator:
        print(namedExpression.getName())
```

6.139.16 Метод Table.getPrintAreas

Метод Table.getPrintAreas возвращает текущие области печати - вектор элементов CellRangePosition.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
firstSheet.setPrintArea(myOfficeSDK.CellRangePosition(0, 0, 5, 5))
printAreas = firstSheet.getPrintAreas()
print(printAreas[0].toString())
```

6.139.17 Метод Table.getProtectionProperties

Метод возвращает параметры защиты от изменений листа табличного документа.

Вызов:

```
TableProtectionProperties getProtectionProperties()
```

Возвращает:

– `TableProtectionProperties`: свойства защиты листа документа (`None`, если защита листа не установлена).

Используйте методы `setProtection()` и `removeProtection()`, чтобы установить и снять защиту от изменений листа. Вы также можете использовать метод `isProtected()`, чтобы узнать, установлена ли защита на лист.

6.139.18 Метод Table.getRowsCount

Метод позволяет получить количество строк таблицы.

Пример:

```
table = document.getBlocks().getTable("List11")
print(table.getRowsCount())
```

6.139.19 Метод Table.getShowZeroValue

Для проверки режима отображения нулевых значений ячеек используется метод `getShowZeroValue`.

Пример:

```
table = document.getBlocks().getTable(0)
table.setShowZeroValue(False)
print(table.getShowZeroValue())
```

6.139.20 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.

6.139.21 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.

6.139.22 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
table = document.getRange().getEnd().insertTable(2, 2, "NewTable")
# Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
table.insertColumnAfter(0, False, 2)
```

6.139.23 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
table = document.getRange().getEnd().insertTable(2, 2, "NewTable")
# Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
table.insertColumnBefore(1, False, 2)
```

6.139.24 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
table = document.getRange().getEnd().insertTable(2, 2, "NewTable")
# Добавление двух строк в середину таблицы, без наследования настроек
форматирования
table.insertRowAfter(0, False, 2)
```

6.139.25 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
table = document.getRange().getEnd().insertTable(2, 2, "NewTable")
# Добавление двух строк в середину таблицы, без наследования настроек
форматирования
table.insertRowBefore(1, False, 2)
```

6.139.26 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:

```
table = document.getBlocks().getTable(0)
print(table.isColumnVisible(0))
```

Дополнительный пример использования метода `Table.isColumnVisible` приведен в разделе Управление видимостью строк / колонок.

6.139.27 Метод Table.isProtected

Метод возвращает состояние защиты от изменений листа табличного документа.

Вызов:

```
bool isProtected()
```

Используйте методы `setProtection()` и `removeProtection()`, чтобы установить и снять защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод `getProtectionProperties()`.

6.139.28 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:

```
table = document.getBlocks().getTable(0)
print(table.isRowVisible(0))
```

Дополнительный пример использования метода `Table.isRowVisible` приведен в разделе Управление видимостью строк / колонок.

6.139.29 Метод Table.isVisible

Метод возвращает значение `True`, если лист таблицы в табличном документе отображается в редакторе таблиц.

Пример:

```
table = document.getBlocks().getTable(0)
if table.isVisible() == False:
    table.setVisible(True)
```

6.139.30 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:

```
table = document.getBlocks().getTable(0)
table.moveTo(1)
```

6.139.31 Метод Table.remove

Для удаления таблицы в текстовом документе или листа в табличном документе используется метод `remove()`.

Пример:

```
table = document.getBlocks().getTable(0)
table.remove()
```

6.139.32 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.

Пример:

```
# Создать в документе новую таблицу 2x2
table = document.getRange().getEnd().insertTable(2, 2, "NewTable")
# Удаление одной строки начиная с первой
table.removeColumn(1, 1)
```

6.139.33 Метод Table.removeProtection

Метод снимает защиту от изменений с листа табличного документа.

Вызов:

```
removeProtection(password)
```

Параметры:

– `password`: (необязательный) пароль для снятия защиты, тип `string`.

Пример:

```
firstSheet = document.getBlocks().getTable(0)
firstSheet.removeProtection("password")
```

Используйте метод `setProtection()`, чтобы установить защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод `getProtectionProperties()`. Вы также можете использовать метод `isProtected()`, чтобы узнать, установлена ли защита на лист.

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

6.139.34 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.

Пример:

```
# Создать в документе новую таблицу 2x2
table = document.getRange().getEnd().insertTable(2, 2, "NewTable")
```

```
# Удаление одной колонки начиная с первой
table.removeRow(1, 1)
```

6.139.35 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.

6.139.36 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.

6.139.37 Метод Table.setColumnsVisible

Метод `Table.setColumnsVisible` позволяет задавать видимость столбцов, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
setColumnsVisible(first, columnsCount, visible)
```

Параметры:

`first` – начальный индекс;

`columnsCount` – количество столбцов;

`visible` – видимость.

6.139.38 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:

```
table = document.getRange().getBegin().insertTable(2, 2, "NewTable")
# Установить ширину столбца в 400 pt
table.setColumnWidth(1, 400)
```

6.139.39 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:

```
table = document.getBlocks().getTable("List11")
table.setName("Table1")
print(table.getName()) # Table1
```

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:

```
tableName = "Table1"
table = document.getBlocks().getTable(0)
table.setName(tableName)
table = document.getBlocks().getTable(tableName)
```

6.139.40 Метод Table.setPrintArea

Метод служит для установки и сброса области печати CellRangePosition.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
firstSheet.setPrintArea(myOfficeSDK.CellRangePosition(0, 0, 2, 2))
```

6.139.41 Метод Table.setPrintAreas

Метод `Table.setPrintAreas` задает множественные области печати или экспорта `CellRangePositions`, где `CellRangePositions` - вектор из элементов CellRangePosition. Метод может быть использован только в табличных документах.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
ranges = myOfficeSDK.CellRangePositions()
ranges.push_back(myOfficeSDK.CellRangePosition(0, 0, 5, 5))
ranges.push_back(myOfficeSDK.CellRangePosition(1, 2, 5, 5))
firstSheet.setPrintAreas(ranges)

printAreas = firstSheet.getPrintAreas()
print(printAreas[0].toString(), printAreas[1].toString())
```

6.139.42 Метод Table.setProtection

Метод устанавливает защиту от изменений на лист табличного документа.

Вызов:

```
setProtection(tableProtectionProperties, password)
```

Параметры:

– `tableProtectionProperties`: параметры защиты листа, тип TableProtectionProperties;

– `password`: (необязательный) пароль для установки защиты, тип `string`.

Пример:

```
tableProps = myOfficeSDK.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

firstSheet = document.getBlocks().getTable(0)
firstSheet.setProtection(tableProps, "password")
```

Используйте метод `removeProtection()`, чтобы снять защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод `getProtectionProperties()`. Вы также можете использовать метод `isProtected()`, чтобы узнать, установлена ли защита на лист.

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`.

6.139.43 Метод Table.setRowHeight

Метод устанавливает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов:

```
setRowHeight(rowIndex, height)
```

Параметры:

- `rowIndex` – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- `height` – высота строки в пунктах (1/72 дюйма).
- `rowHeightRule` – точность значения (`RowHeightRule.Exact` – точно, `RowHeightRule.AtLeast` – не меньше).

Пример:

```
table = document.getRange().getBegin().insertTable(2, 2, "NewTable")
# Установить высоту строки в 100 pt
table.setRowHeight(1, 100, myOfficeSDK.RowHeightRule_Exact)
```

6.139.44 Метод Table.setRowsVisible

Метод `Table.setRowsVisible` позволяет задавать видимость строк, начиная с заданного индекса. Индексация строк начинается с нуля.

Вызов:

```
setRowsVisible(first, rowsCount, visible)
```

Параметры:

`first` – начальный индекс;

`columnsCount` – количество строк;

`visible` – видимость.

6.139.45 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 табличных документах.

Пример:

```
table = document.getBlocks().getTable(0)
table.setShowZeroValue(False)
```

6.139.46 Метод Table.setVisible

Метод управляет видимостью листа таблицы. Используется только в табличном документе.

Пример:

```
table = document.getBlocks().getTable(0)
table.setVisible(False)
```

6.139.47 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.

6.139.48 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.

6.139.49 Table.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Table`.

Пример:

```
firstTable = document.getBlocks().getTable(0)
secondTable = document.getBlocks().getTable(0)

if firstTable.__eq__(secondTable):
    print("Equals")
```

6.139.50 Table.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Table`.

Пример:

```
firstTable = document.getBlocks().getTable(0)
secondTable = document.getBlocks().getTable(1)
```

```
if firstTable.__ne__(secondTable):
    print("Not equals")
```

6.140 Класс TableFilters

`TableFilters` - это объект, который хранит фильтры столбцов. Фильтры можно применять к диапазону фильтрации `FiltersRange`. При применении фильтров, соответствующие строки рабочего листа будут скрыты.

Конструктор по умолчанию:

```
TableFilters()
```

Конструктор копирования:

```
TableFilters(TableFilters other)
```

Пример использования приведен в разделе Работа с фильтрами.

6.140.1 Метод TableFilters.clear

Метод `TableFilters.clear` удаляет все фильтры столбцов, которые были сохранены ранее.

Пример:

```
tableFilters = myOfficeSDK.TableFilters()
tableFilters.setFilter(0, johnPaulFilter)
tableFilters.setFilter(1, songFilter)
.............................
tableFilters.clear()
```

6.140.2 Метод TableFilters.erase

Метод `TableFilters.erase` удаляет фильтр из заданного столбца. В качестве параметра используется индекс столбца.

Пример:

```
tableFilters = myOfficeSDK.TableFilters()
tableFilters.setFilter(0, johnPaulFilter)
tableFilters.setFilter(1, songFilter)
.............................
tableFilters.erase(1)
```

6.140.3 Метод TableFilters.setFilter

Метод `TableFilters.setFilter` устанавливает фильтр для конкретного столбца.

Нумерация столбцов начинается с нуля, относительно левой позиции диапазона фильтрации.

```
setFilter(int column, ValuesTableFilter filter)
setFilter(int column, ConditionalTableFilter filter)
```

Параметры:

- `column` – индекс столбца;
- `filter` – вариант фильтра: `ValuesTableFilter` или `ConditionalTableFilter`.

Пример:

```
johnPaulFilter = myOfficeSDK.ValuesTableFilter()
johnPaulFilter.add("John")
johnPaulFilter.add("Paul")

songFilter = myOfficeSDK.ConditionalTableFilter()
songFilter.setAndOperation(True)
songFilter.notEqual("")
songFilter.notBegins("TODO")

tableFilters = myOfficeSDK.TableFilters()
tableFilters.setFilter(0, johnPaulFilter)
tableFilters.setFilter(1, songFilter)
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.141 Класс TableProtectionProperties

Класс `TableProtectionProperties` предназначен для настройки параметров защиты листа в табличном документе (аналог раздела «Разрешенные действия» в меню «Управление защитой»). Данный класс используется в методах `Table.setProtection()` и `Table.getProtectionProperties()`.

Таблица 78 – Описание полей класса `TableProtectionProperties`

Поле	Значение по умолчанию	Описание
`TableProtectionProperties.deleteColumns`	`False`	Разрешить удалять колонки
`TableProtectionProperties.deleteRows`	`False`	Разрешить удалять строки
`TableProtectionProperties.filterData`	`False`	Разрешить фильтровать данные

Поле	Значение по умолчанию	Описание
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:

```
tableProps = myOfficeSDK.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

firstSheet = document.getBlocks().getTable(0)
firstSheet.setProtection(tableProps, "password")
```

6.142 Класс TableRangeInfo

Класс `TableRangeInfo` описывает диапазон ячеек таблицы.

Описание полей класса `TableRangeInfo` представлено в таблице 79.

Таблица 79 – Поля класса `TableRangeInfo`

Поле	Тип	Описание
`tableRange`	`CellRangePosition`	Диапазон ячеек

Пример:

```
table = document.getBlocks().getTable(0)
charts = table.getCharts()
rangeInfo = charts.getChart(0).getRange(0)
cellRangePosition = rangeInfo.tableRangeInfo
tableRange = cellRangePosition.tableRange
print("Top left row:", tableRange.topLeft.row, ", top left column:",
tableRange.topLeft.column)
```

6.143 Класс TextAnchoredPosition

Класс `TextAnchoredPosition` представляет позицию объекта на странице текстового документа. Описание полей представлено в таблице 80.

Таблица 80 – Описание полей класса `TextAnchoredPosition`

Поле	Описание
`TextAnchoredPosition.horizontal`	Позиция по горизонтали `HorizontalTexAnchoredPosition`
`TextAnchoredPosition.vertical`	Позиция по вертикали `VerticalTextAnchoredPosition`

6.143.1 TextAnchoredPosition.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `TextAnchoredPosition`.

Пример:

```
firstTextAnchoredPosition = myOfficeSDK.TextAnchoredPosition()
firstTextAnchoredPosition.horizontal =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
firstTextAnchoredPosition.horizontal.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
firstTextAnchoredPosition.horizontal.offset = 10
firstTextAnchoredPosition.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageContent)
firstTextAnchoredPosition.vertical.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
firstTextAnchoredPosition.vertical.offset = 10

secondTextAnchoredPosition = myOfficeSDK.TextAnchoredPosition()
secondTextAnchoredPosition.horizontal =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
secondTextAnchoredPosition.horizontal.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
secondTextAnchoredPosition.horizontal.offset = 10
secondTextAnchoredPosition.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageContent)
secondTextAnchoredPosition.vertical.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondTextAnchoredPosition.vertical.offset = 10

if firstTextAnchoredPosition.__eq__(secondTextAnchoredPosition):
    print("Equals")
```

6.143.2 TextAnchoredPosition.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `TextAnchoredPosition`.

Пример:

```
firstTextAnchoredPosition = myOfficeSDK.TextAnchoredPosition()
firstTextAnchoredPosition.horizontal =
```

```
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
firstTextAnchoredPosition.horizontal.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
firstTextAnchoredPosition.horizontal.offset = 10
firstTextAnchoredPosition.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageContent)
firstTextAnchoredPosition.vertical.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
firstTextAnchoredPosition.vertical.offset = 10

secondTextAnchoredPosition = myOfficeSDK.TextAnchoredPosition()
secondTextAnchoredPosition.horizontal =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
secondTextAnchoredPosition.horizontal.aligment =
myOfficeSDK.HorizontalAnchorAlignment_Center
secondTextAnchoredPosition.horizontal.offset = 20
secondTextAnchoredPosition.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageContent)
secondTextAnchoredPosition.vertical.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondTextAnchoredPosition.vertical.offset = 20

if firstTextAnchoredPosition.__ne__(secondTextAnchoredPosition):
    print("Not equals")
```

6.144 Класс TextExportSettings

Класс `TextExportSettings` предоставляет настройки, необходимые для экспорта текстовых документов, см. `Document.exportAs()`. Поле `TextExportSettings.pageNumbers` является экземпляром класса `PageNumbers`, в котором содержатся настройки страниц для экспорта текстовых документов.

Пример:

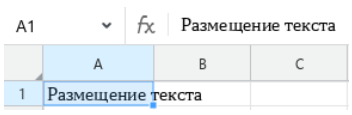
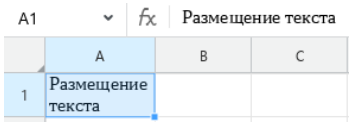
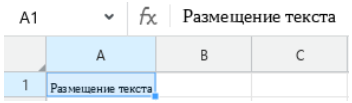
```
textExportSettings = myOfficeSDK.TextExportSettings()
textExportSettings.pageNumbers =
```

```
myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1, textExportSettings)
```

6.145 Класс TextLayout

В таблице 81 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле `textLayout` таблицы `CellProperties`.

Таблица 81 – Варианты размещения текста в ячейках таблицы

Наименование константы	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:

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("A1")
cellProps = cell.getCellProperties()
cellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth
cell.setCellProperties(cellProps)
```

6.146 Класс TextOrientation

Класс `TextOrientation` предоставляет доступ к свойствам ориентации текста в ячейке, фигуре и т. д (см. `CellProperties`).

Пример:

```
firstSheet = document.getBlocks().getTable("Лист1");
cell = firstSheet.getCell("A3")
cellProps = cell.getCellProperties()
```

```
textOrientation = myOfficeSDK.TextOrientation(45)
cell.setCellProperties(cellProps)
```

6.146.1 Метод TextOrientation.getAngle

Возвращает угол направления текста в ячейке. Значение угла указывается в градусах.

Пример:

```
firstSheet = document.getBlocks().getTable("List11")
cell = firstSheet.getCell("A3")
cellProps = cell.getCellProperties()
print(cellProps.textOrientation.isStackedChars())
```

6.146.2 TextOrientation.isStackedChars

Возвращает `True`, если ориентация текста - вертикальный столбец.

```
firstSheet = document.getBlocks().getTable("List1")
cell = firstSheet.getCell("A1")
cellProps = cell.getCellProperties()
print(cellProps.textOrientation.isStackedChars())
```

6.146.3 TextOrientation.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `LineSpacing`.

Пример:

```
firstTextOrientation = myOfficeSDK.TextOrientation(30)
secondTextOrientation = myOfficeSDK.TextOrientation(30)
if firstTextOrientation.__eq__(secondTextOrientation):
    print("Equals")
```

6.146.4 TextOrientation.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `TextOrientation`.

Пример:

```
firstTextOrientation = myOfficeSDK.TextOrientation(30)
secondTextOrientation = myOfficeSDK.TextOrientation(45)
if firstTextOrientation.__ne__(secondTextOrientation):
    print("Not equals")
```

6.147 Класс TextProperties

Класс `DocumentAPI.TextProperties` содержит поля, задающие параметры текста. На рисунке 37 изображена объектная модель класса `DocumentAPI.TextProperties`.

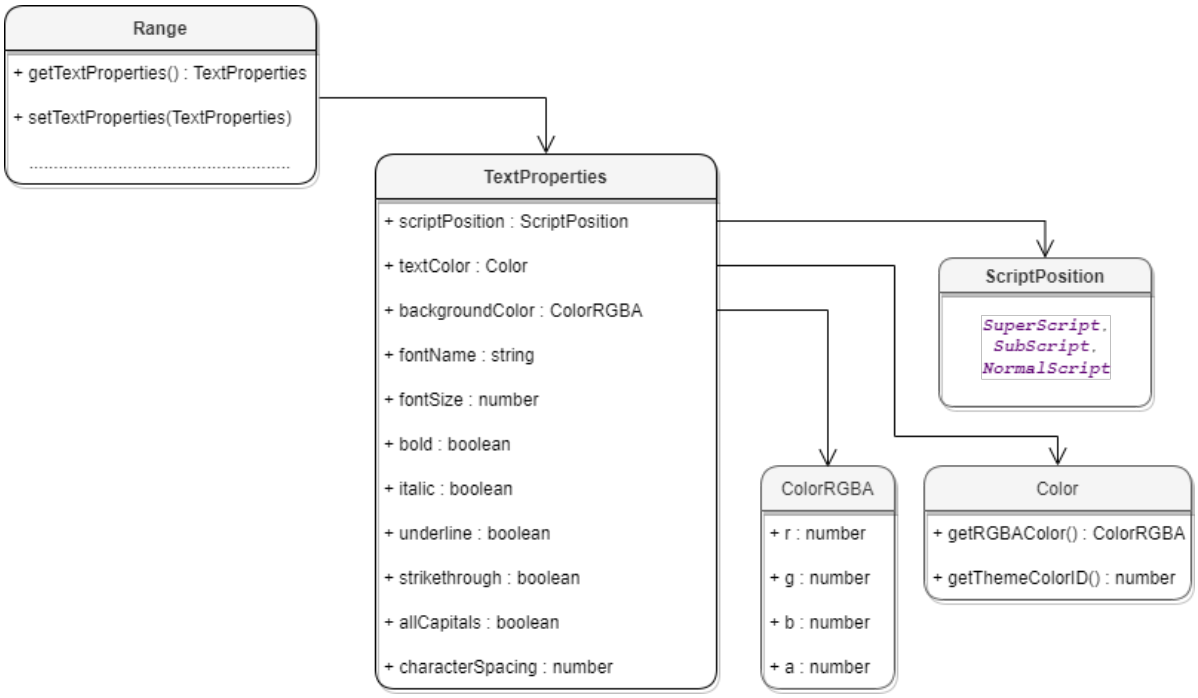


Рисунок 37 – Объектная модель для работы с классом `DocumentAPI.TextProperties`

Описание полей класса `TextProperties` представлено в таблице 82. Свойства `TextProperties` применяются к диапазону текста `Range` (методы `Range.getTextProperties()`, `Range.setTextProperties()`).

Таблица 82 – Описание полей класса `TextProperties`

Поле	Тип	Описание
`TextProperties.fontName`	Строковое	Наименование шрифта, использованного для оформления фрагмента документа.
`TextProperties.fontSize`	Числовое	Размер шрифта, использованного для оформления фрагмента документа.
`TextProperties.bold`	Логическое	Значение `True` устанавливает жирное начертание для указанного фрагмента текста.
`TextProperties.italic`	Логическое	Значение `True` устанавливает начертание курсивом для указанного фрагмента текста.
`TextProperties.underline`	Логическое	Значение `True` устанавливает подчеркивание для указанного фрагмента текста.

Поле	Тип	Описание
TextProperties.strikethrough	Логическое	Значение True устанавливает начертание «зачеркнутый» для указанного фрагмента текста.
TextProperties.allCapitals	Логическое	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	Числовое	Размер межсимвольного интервала.

Пример:

```
textProperties = myOfficeSDK.TextProperties()
textProperties.fontName = "XO Oriel"
textProperties.fontSize = 20
paragraph = document.getBlocks().getParagraph(2)
if paragraph != None:
    range = paragraph.getRange()
    range.setTextProperties(textProperties)
```

6.147.1 TextProperties.__eq__

Метод __eq__ используется для определения эквивалентности двух объектов типа TextProperties.

Пример:

```
firstTextProperties = myOfficeSDK.TextProperties()
firstTextProperties.fontName = "XO Oriel";
firstTextProperties.fontSize = 20;

secondTextProperties = myOfficeSDK.TextProperties()
secondTextProperties.fontName = "XO Oriel";
secondTextProperties.fontSize = 20;
```

```
if firstTextProperties.__eq__(secondTextProperties):
    print("Equals")
```

6.147.2 TextProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `TextProperties`.

Пример:

```
firstTextProperties = myOfficeSDK.TextProperties()
firstTextProperties.fontName = "XO Oriel";
firstTextProperties.fontSize = 20;

secondTextProperties = myOfficeSDK.TextProperties()
secondTextProperties.fontName = "XO Oriel";
secondTextProperties.fontSize = 30;

if firstTextProperties.__ne__(secondTextProperties):
    print("Not equals")
```

6.148 Класс TextWrapType

В таблице 83 представлены варианты обтекания текстом встроенного объекта. Используется в `InlineFrame.setWrapType()`.

Таблица 83 – Варианты обтекания текстом встроенного объекта

Наименование константы	Описание
`TextWrapType_Inline`	Встроенный объект располагается в тексте
`TextWrapType_InFrontOfText`	Встроенный объект располагается перед текстом
`TextWrapType_BehindText`	Встроенный объект располагается за текстом
`TextWrapType_TopAndBottom`	Текст располагается сверху и снизу от встроенного объекта
`TextWrapType_Square`	Текст располагается вокруг прямоугольной рамки встроенного объекта
`TextWrapType_Through`	Текст обтекает встроенный объект по сторонам и внутри
`TextWrapType_Tight`	Текст располагается на одинаковых расстояниях от границ объекта

6.149 Класс ThemeColorID

В таблице 84 представлены типы идентификаторов цветов тем. Используется в `Color`.

Таблица 84 – Типы идентификаторов цветов тем

Наименование константы	Описание
ThemeColorID_Background1	Фон1
ThemeColorID_Text1	Текст1
ThemeColorID_Background2	Фон2
ThemeColorID_Text2	Текст2
ThemeColorID_Dark1	Темная1
ThemeColorID_Dark2	Темная2
ThemeColorID_Light1	Светлая1
ThemeColorID_Light2	Светлая2
ThemeColorID_Accent1	Акцент1
ThemeColorID_Accent2	Акцент2
ThemeColorID_Accent3	Акцент3
ThemeColorID_Accent4	Акцент4
ThemeColorID_Accent5	Акцент5
ThemeColorID_Accent6	Акцент6
ThemeColorID_Hyperlink	Гиперссылка
ThemeColorID_FollowedHyperlink	Следующая гиперссылка

6.150 Класс TimePatterns

Форматы времени представлены в таблице 85. Пример использования см. в главе DateTimeCellFormatting.

Таблица 85 – Форматы времени

Наименование константы	Описание
TimePatterns_ShortTime	'hh:mm AM/PM' для языка en_US
TimePatterns_LongTime	'hh:mm:ss AM/PM' для языка en_US

6.151 Класс TimeZone

Класс `TimeZone` предоставляет настройки, необходимые для экспорта текстовых документов.

Поле класса `TimeZone.offsetInSecondsToUTC` (числовой тип) содержит значение, с помощью которого задается смещение или разность между временем в данном часовом поясе и временем в формате UTC (Всемирное координированное время).

6.152 Класс TrackedChange

Класс `TrackedChange` представляет отслеживаемое изменение в диапазоне документа (см. Рисунок 38).

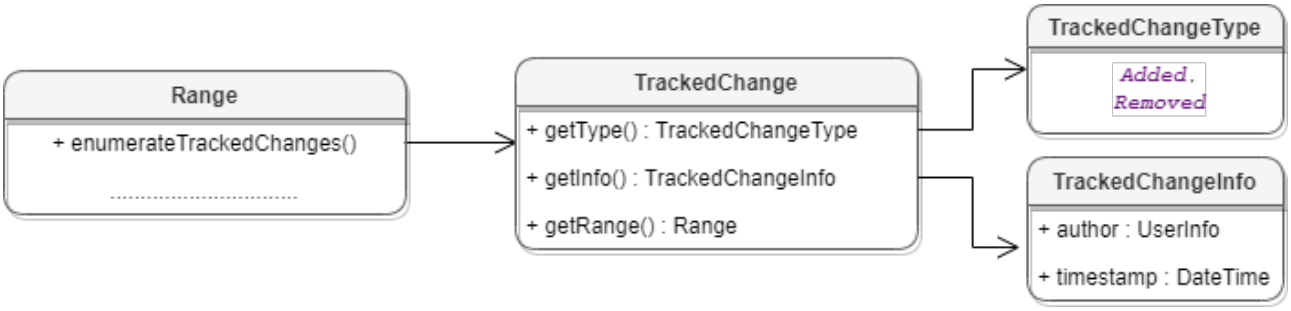


Рисунок 38 – Объектная модель классов для работы с отслеживаемыми изменениями

Для получения списка отслеживаемых изменений используется метод `Range.getTrackedChangesEnumerator()`.

Пример:

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
```

6.152.1 Метод TrackedChange.getInfo

Метод позволяет получить информацию об отслеживаемых изменениях `TrackedChangeInfo`.

Пример:

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
if trackedChangesEnumerator != None:
    for trackedChange in trackedChangesEnumerator:
        trackedChangeInfo = trackedChange.getInfo()
```

6.152.2 Метод TrackedChange.getRange

Метод возвращает объект Range, который соответствует измененному диапазону внутри абзаца.

Пример:

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
if trackedChangesEnumerator != None:
    for trackedChange in trackedChangesEnumerator:
        trackedChangeRange = trackedChange.getRange()
        print(trackedChangeRange.extractText())
```

6.152.3 Метод TrackedChange.getType

Метод позволяет получить информацию о типе отслеживаемого изменения TrackedChangeType.

Пример:

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
if trackedChangesEnumerator != None:
    for trackedChange in trackedChangesEnumerator:
        trackedChangeType = trackedChange.getType()
```

6.153 Класс TrackedChangeInfo

Класс TrackedChangeInfo содержит информацию об отслеживаемых изменениях. Описание полей представлено в таблице 86.

Таблица 86 – Описание полей класса TrackedChangeInfo

Поле	Тип	Описание
TrackedChangeInfo.author	UserInfo	Автор изменений
TrackedChangeInfo.timeStamp	DateTime	Дата и время изменений

Пример:

```
trackedChangeInfo = trackedChange.getInfo()
author = trackedChangeInfo.author
if author != None:
    print(author.name)
timeStamp = trackedChangeInfo.timeStamp
if timeStamp != None:
    print(timeStamp.year, timeStamp.month, timeStamp.day)
```

6.154 Класс TrackedChangeType

Поддерживаемые типы отслеживаемых изменений представлены в таблице 87.

Таблица 87 - Типы отслеживаемых изменений

Имя константы	Описание
TrackedChangeType_Added	Добавленные изменения
TrackedChangeType_Removed	Удаленные изменения

Пример:

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
if trackedChangesEnumerator != None:
    for trackedChange in trackedChangesEnumerator:
        if trackedChange.getType() == myOfficeSDK.TrackedChangeType_Added:
            print("Added")
        else:
            print("Removed")
```

6.155 Класс UserInfo

Класс `UserInfo` предоставляет информацию о пользователе.

Описание полей таблицы `UserInfo` представлено в таблице 88.

Таблица 88 – Описание полей класса `UserInfo`

Поле	Описание	Тип
UserInfo.name	Имя пользователя	Строка
UserInfo.email	Адрес электронной почты пользователя	Строка

6.155.1 Метод UserInfo.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `UserInfo`.

Пример:

```
firstUserInfo = myOfficeSDK.UserInfo()
firstUserInfo.name = "user"
firstUserInfo.email = "user@domain.com

secondUserInfo = myOfficeSDK.UserInfo()
```

```
secondUserInfo.name = "user"
secondUserInfo.email = "user@domain.com

if firstUserInfo.__eq__(secondUserInfo):
    print("Equals");
```

6.155.2 Метод UserInfo.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `UserInfo`.

Пример:

```
firstUserInfo = myOfficeSDK.UserInfo()
firstUserInfo.name = "user"
firstUserInfo.email = "user@domain.com

secondUserInfo = myOfficeSDK.UserInfo()
secondUserInfo.name = "second user"
secondUserInfo.email = "user@domain.com

if firstUserInfo.__ne__(secondUserInfo):
    print("Not equals")
```

6.156 Класс ValueFieldsOrientation

Класс `ValueFueldsOrientation` описывает варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений. Является полем класса `PivotTableLayoutSettings`. Описание полей представлено в таблице 89.

Таблица 89 – Описание полей `ValueFueldsOrientation`

Поле	Описание
`ValueFueldsOrientation_ByRows`	По строкам
`ValueFueldsOrientation_ByColumns`	По столбцам

6.157 Класс ValuesTableFilter

Класс `ValuesTableFilter` реализует фильтр, содержащий значения, которые должны быть показаны в диапазоне фильтрации.

Конструктор по умолчанию:

```
ValuesTableFilter()
```

Конструктор копирования:

```
ValuesTableFilter(ValuesTableFilters other)
```

Пример использования приведен в разделе Работа с фильтрами.

6.157.1 Метод ValuesTableFilter.add

Метод `ValuesTableFilter.add` добавляет значение, которое должно быть отображено в таблице.

Пример:

```
johnPaulFilter = myOfficeSDK.ValuesTableFilter()
johnPaulFilter.add("John")
johnPaulFilter.add("Paul")
```

6.157.2 Метод ValuesTableFilter.clear

Метод `ValuesTableFilter.clear` удаляет все элементы фильтра.

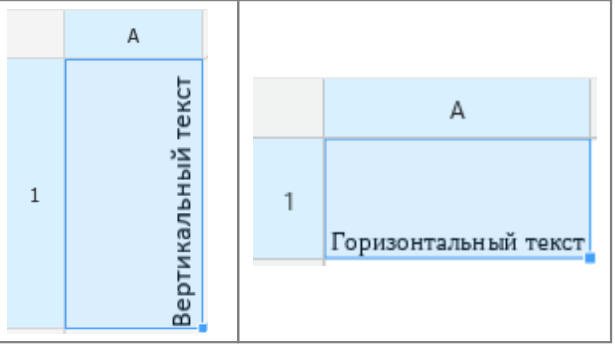
Пример:

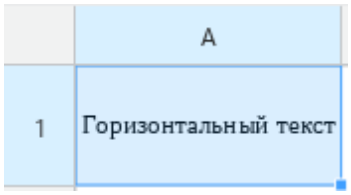
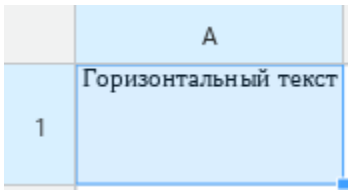
```
johnPaulFilter = myOfficeSDK.ValuesTableFilter()
johnPaulFilter.add("John")
johnPaulFilter.add("Paul")
.........................
johnPaulFilter.clear()
```

6.158 Класс VerticalAlignment

В таблице 90 представлены константы,описывающие варианты выравнивания текста по вертикали. Используется в `CellProperties`, `ShapeProperties`.

Таблица 90 – Виды выравнивания текста по вертикали

Наименование константы	Представление в интерфейсе
`VerticalAlignment_Bottom`	

Наименование константы	Представление в интерфейсе	
VerticalAlignment_Center		
VerticalAlignment_Top		

Пример:

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

cellProps = cell.getCellProperties()
cellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center

cell.setCellProperties(cellProps)
```

6.159 Класс VerticalAnchorAlignment

В Таблице 91 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали.

Таблица 91 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
VerticalAnchorAlignment_Top	По верхнему краю
VerticalAnchorAlignment_Bottom	По нижнему краю

Наименование константы	Описание
VerticalAnchorAlignment_Center	По центру
VerticalAnchorAlignment_Inside, VerticalAnchorAlignment_Outside	По границам

6.160 Класс VerticalRelativeTo

В таблице 92 представлены типы размещения объекта относительно закрепленной позиции по вертикали.

Таблица 92 – Типы размещения объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
VerticalRelativeTo_Character	Символ
VerticalRelativeTo_BaseLine	Базовая линия
VerticalRelativeTo_Paragraph	Абзац
VerticalRelativeTo_Page	Страница
VerticalRelativeTo_PageContent	Содержимое страницы
VerticalRelativeTo_PageTopMargin	Верхнее поле страницы
VerticalRelativeTo_PageBottomMargin	Нижнее поле страницы
VerticalRelativeTo_PageInsideMargin	Внутреннее поле страницы
VerticalRelativeTo_PageOutsideMargin	Внешнее поле страницы

6.161 Класс VerticalTextAnchoredPosition

Класс VerticalTextAnchoredPosition предназначен для управления относительным положением объекта со смещением или выравниванием по вертикали.

Описание полей класса VerticalTextAnchoredPosition представлено в таблице 93.

Таблица 93 – Описание полей класса `VerticalTextAnchoredPosition`

Поле	Описание
`VerticalTextAnchoredPosition.relativeTo`	Тип размещения объекта относительно закрепленной позиции по вертикали `VerticalRelativeTo`
`VerticalTextAnchoredPosition.offset`	Смещение объекта.
`VerticalTextAnchoredPosition.aligment`	Тип выравнивания объекта относительно закрепленной позиции по вертикали `VerticalAnchorAlignment`

6.161.1 VerticalTextAnchoredPosition.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `TextAnchoredPosition`.

Пример:

```
firstVerticalTextAnchoredPosition =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Paragraph)
firstVerticalTextAnchoredPosition.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
firstVerticalTextAnchoredPosition.offset = 10

secondVerticalTextAnchoredPosition =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Paragraph)
secondVerticalTextAnchoredPosition.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondVerticalTextAnchoredPosition.offset = 10

if firstVerticalTextAnchoredPosition.__eq__(secondVerticalTextAnchoredPosition):
    print("Equals")
```

6.161.2 VerticalTextAnchoredPosition.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `TextAnchoredPosition`.

Пример:

```
firstVerticalTextAnchoredPosition =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Paragraph)
firstVerticalTextAnchoredPosition.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
firstVerticalTextAnchoredPosition.offset = 10

secondVerticalTextAnchoredPosition =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Paragraph)
secondVerticalTextAnchoredPosition.aligment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondVerticalTextAnchoredPosition.offset = 20

if firstVerticalTextAnchoredPosition.__ne__(secondVerticalTextAnchoredPosition):
    print("Not equals")
```

6.162 Класс WorkbookExportSettings

Класс `WorkbookExportSettings` предоставляет настройки, необходимые для экспорта табличных документов, см. `Document.exportAs()`.

Описание полей класса `WorkbookExportSettings` представлено в таблице 94.

Таблица 94 – Описание полей класса `WorkbookExportSettings`

Поле	Описание
`sheetNames`	Представляет коллекцию имен листов для экспорта, тип `VectorString`. Если коллекция пуста, экспортируются все листы.
`printingScope`	Представляет область печати (весь документ, область печати, пользовательский диапазон и т. д.) `PrintingScope`.
`pageProperties`	Представляют свойства страницы для выходного документа (высота и ширина страницы в пунктах pt) `PageProperties`.

Поле	Описание
scale	Представляет масштаб экспорта выходного документа в процентах (например, 50,0%, 150,63%, 400,0% и т. д.).

Пример:

```
workbookSettings = myOfficeSDK.WorkbookExportSettings()
workbookSettings.sheetNames = myOfficeSDK.VectorString()
workbookSettings.sheetNames.push_back("List1")
workbookSettings.printingScope =
myOfficeSDK.PrintingScope(myOfficeSDK.PrintingScope.Type_PrintArea)
workbookSettings.pageProperties = myOfficeSDK.PageProperties(100, 200)
workbookSettings.scale = 90
document.exportAs(filePath, ExportFormat.PDFA1, workbookSettings)
```

6.163 Исключения

6.163.1 Класс BaseError

Класс `BaseError` является базовым классом для всех исключений SDK.

```
class BaseError(Exception)
```

6.163.2 Класс ApplicationCreateError

Исключение `ApplicationCreateError` вызывается в случае, когда объект `Application` не может быть создан.

```
class ApplicationCreateError(BaseError)
```

6.163.3 Класс DocumentCreateError

Исключение `DocumentCreateError` вызывается в случае, когда документ не может быть создан.

```
class DocumentCreateError(BaseError)
```

6.163.4 Класс DocumentExportError

Исключение `DocumentExportError` вызывается в случае, когда документ не может быть экспортирован.

```
class DocumentExportError(BaseError)
```

6.163.5 Класс DocumentLoadError

Исключение `DocumentLoadError` вызывается в случае, когда документ не может быть загружен.

```
class DocumentLoadError(BaseError)
```

6.163.6 Класс DocumentModificationError

Исключение `DocumentModificationError` вызывается, когда невозможно выполнить операцию по изменению документа. Например, оно возникает при попытке применить методы `Paragraph.setListLevel()`, `Paragraph.increaseListLevel()`, `Paragraph.decreaseListLevel()` для параграфа, который не является списком.

```
class DocumentModificationError(BaseError)
```

6.163.7 Класс DocumentSaveError

Исключение `DocumentSaveError` вызывается в случае, когда документ не может быть сохранен.

```
class DocumentSaveError(BaseError)
```

6.163.8 Класс ForbiddenActionError

Исключение `ForbiddenActionError` вызывается в случае выполнения запрещенной операции.

```
class ForbiddenActionError(BaseError)
```

6.163.9 Класс IncorrectArgumentError

Исключение `IncorrectArgumentError` вызывается в случае, когда один из аргументов метода или функции имеет недействительное значение.

```
class IncorrectArgumentError(BaseError)
```

6.163.10 Класс InvalidObjectError

Исключение `InvalidObjectError` вызывается в случае, когда объект больше не может быть использован.

```
class InvalidObjectError(BaseError)
```

6.163.11 Класс NoSuchElementError

Исключение `NoSuchElementError` вызывается в случае, когда элемент не существует.

```
class NoSuchElementError(BaseError)
```

6.163.12 Класс NotImplementedError

Исключение `NotImplementedError` вызывается в случае, если обнаружена нереализованная функциональность.

```
class NotImplementedError(BaseError)
```

6.163.13 Класс OutOfRangeError

Исключение `OutOfRangeError` вызывается в случае обнаружения выхода значения за пределы диапазона.

```
class OutOfRangeError(BaseError)
```

6.163.14 Класс ParseError

Исключение `ParseError` вызывается в случае ошибки синтаксического разбора текста.

```
class ParseError(BaseError)
```

6.163.15 Класс PivotTableError

Исключение `PivotTableError` вызывается в случае ошибки при работе со сводными таблицами. Например, использование фильтра, который не может быть применен к сводной таблице.

```
class PivotTableError(BaseError)
```

6.163.16 Класс PositionDocumentMismatchError

Исключение `PositionDocumentsMismatchError` вызывается в случае, когда несколько позиций относятся к различным документам и не могут быть использованы в одной операции. Например, при попытке пользователя создать диапазон `Range`, включающий позиции `Position`, принадлежащие нескольким различным документам, и выполнить операцию для такого диапазона.

```
class PositionDocumentMismatchError(BaseError)
```

6.163.17 Класс ScriptExecutionError

Исключение `ScriptExecutionError` вызывается в случае, когда сценарий не удается выполнить (см. `Scripting.runScript()`).

```
class ScriptExecutionError(BaseError)
```

6.163.18 Класс UnknownError

Исключение `UnknownError` вызывается в случае, когда критическое исключение возникло по неизвестной причине. Приложение должно быть завершено, поскольку возникло неопределенное состояние ядра Document API.

```
class UnknownError(BaseError)
```

7 Версии Document API

7.1 Механизм контроля версий

Константы версии Document API `Major` и `Minor` позволяют проверить совместимость предыдущей и текущей версии Document API.

Если была изменена константа `Major` версии Document API, т. е. в Document API произошли обратно несовместимые изменения, то программный код должен быть пересмотрен и обновлен. Обратно несовместимыми изменениями считаются: переименование, удаление или несовместимое изменение подписи существующих классов или методов, а также добавление новых методов, типов и членов класса.

Если была изменена константа `Minor` версии Document API, то в Document API произошли только обратно совместимые изменения, и нет необходимости менять программный код, чтобы он работал с более новой версией Document API. Но гарантируется совместимость только на уровне исходного кода, поэтому необходимо перекомпилировать программный код приложения с более новой версией библиотеки Document API.

Рекомендуется проверить версию Document API до инициализации, как указано ниже:

```
print(myOfficeSDK.Minor)
print(myOfficeSDK.Major)

if __name__ == '__main__':
    expected_major_api_version = 1
    expected_minor_api_version = 0

    if not myOfficeSDK.isAPIVersionCompatible(expected_major_api_version,
expected_minor_api_version):

        # Вывод сообщения о серьезной ошибке несовместимости версии библиотеки
Document API и выход из программы

        pass

    # Работа с библиотекой Document API (создание объекта Application и т. д.)
```