



# МойОфис Комплект Средств Разработки (SDK)

## Руководство программиста

MYOFFICE DOCUMENT API (Python)

**ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»**

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**

**«МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)»**

**MYOFFICE DOCUMENT APPLICATION PROGRAMMING INTERFACE (API).**

**БИБЛИОТЕКА ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON**

**РУКОВОДСТВО ПРОГРАММИСТА**

**3.1**

**Версия 1**

**На 268 листах**

**Дата публикации: 29.07.2024**

**Москва  
2024**

# МойОфис

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

1	Общие сведения .....	24
1.1	Назначение .....	24
1.2	Библиотека MyOffice Document API для языка программирования Python .....	24
1.3	Уровень подготовки пользователя .....	25
1.4	Системные требования .....	25
2	Подготовка к работе .....	26
2.1	Список дистрибутивов .....	26
2.2	Установка в ОС Microsoft Windows .....	26
2.3	Установка в ОС Linux .....	26
2.4	Проверка работоспособности .....	27
2.5	Распространение разработанных приложений .....	27
3	Объектная модель МойОфис SDK .....	29
4	Работа с документами .....	31
4.1	Работа с текстовым документом .....	31
4.1.1	Создание и открытие текстового документа .....	31
4.1.2	Сохранение и экспорт текстового документа .....	31
4.1.3	Разделы (секции) документа .....	32
4.1.4	Встроенные объекты в текстовом документе .....	33
4.1.4.1	Вставка изображения .....	34
4.1.4.2	Перечисление встроенных объектов .....	34
4.1.5	Работа с таблицами текстового документа .....	34
4.1.6	Работа с закладками .....	36
4.1.7	Рецензирование документов .....	37
4.2	Работа с табличным документом .....	38
4.2.1	Создание и открытие табличного документа .....	38
4.2.2	Сохранение и экспорт табличного документа .....	39
4.2.3	Диаграммы .....	39
4.2.4	Копирование ячеек в табличном документе .....	40
4.2.5	Встроенные объекты в табличном документе .....	41
4.2.5.1	Вставка изображения .....	41
4.2.5.2	Перечисление встроенных объектов .....	41
4.2.6	Работа с листами табличного документа .....	42

# МойОфис

4.2.7	Работа со сводными таблицами .....	44
4.2.7.1	Получение сводной таблицы .....	44
4.2.7.2	Получение диапазона исходных данных сводной таблицы .....	45
4.2.7.3	Получение диапазона размещения сводной таблицы .....	45
4.2.7.4	Получение неподдерживаемых свойств сводной таблицы .....	45
4.2.7.5	Получение флагов отображения общих итогов для строк и колонок .....	45
4.2.7.6	Получение заголовков сводной таблицы .....	46
4.2.7.7	Получение и применение фильтра для сводной таблицы .....	46
4.2.7.8	Получение полей из области фильтров .....	46
4.2.7.9	Получение полей из области значений .....	46
4.2.7.10	Получение полей из области строк .....	47
4.2.7.11	Получение полей из области колонок .....	47
4.2.7.12	Получение настроек отображения сводной таблицы .....	48
4.2.7.13	Обновление сводной таблицы .....	48
4.2.8	Работа с фильтрами .....	48
4.2.9	Встроенные объекты .....	49
4.2.9.1	Определение типа встроенных объектов .....	49
4.2.9.2	Работа со встроенными объектами .....	50
4.3	Поиск в документе .....	51
4.4	Работа с макрокомандами .....	52
4.5	Работа с именованными диапазонами .....	53
4.5.1	Доступ к именованным диапазонам .....	54
4.5.2	Получение коллекции именованных диапазонов .....	54
4.5.3	Получение свойств именованного диапазона .....	55
4.5.4	Добавление именованного диапазона .....	55
4.5.5	Удаление именованного диапазона .....	55
4.5.6	Получение параметров именованного диапазона .....	55
4.6	Работа со строками и столбцами таблиц .....	56
4.6.1	Группировка строк и колонок таблицы .....	56
4.6.2	Управление видимостью строк / колонок .....	56
4.7	Работа с ячейками таблиц .....	57
4.7.1	Доступ к ячейкам .....	57
4.7.2	Форматирование ячеек .....	59
4.7.3	Форматирование границ ячеек .....	61

# МойОфис

4.7.4	Объединение и разделение ячеек таблицы .....	62
5	Глобальные методы .....	63
5.1	Глобальный метод createSearch .....	63
6	Справочник классов, структур и методов .....	64
6.1	Класс AbsoluteFrame .....	64
6.1.1	Метод AbsoluteFrame.moveTo .....	64
6.1.2	Метод AbsoluteFrame.getTopLeft .....	65
6.1.3	Метод AbsoluteFrame.scale .....	65
6.1.4	Метод AbsoluteFrame.setDimensions .....	65
6.1.5	Метод AbsoluteFrame.getDimensions .....	66
6.2	Класс AccountingCellFormatting .....	66
6.3	Класс Alignment .....	67
6.4	Класс Application .....	67
6.4.1	Метод Application.createDocument .....	67
6.4.2	Метод Application.loadDocument .....	68
6.4.3	Метод Application.getMessenger .....	69
6.5	Класс Block .....	69
6.5.1	Методы toParagraph, toTable, toShape, toField .....	69
6.5.2	Метод Block.getRange .....	70
6.5.3	Метод Block.remove .....	70
6.5.4	Метод Block.getSection .....	70
6.6	Класс Blocks .....	70
6.6.1	Метод Blocks.getBlock .....	71
6.6.2	Метод Blocks.getParagraph .....	71
6.6.3	Метод Blocks.getTable .....	71
6.6.4	Метод Blocks.getShape .....	72
6.6.5	Метод Blocks.getField .....	72
6.6.6	Метод Blocks.GetEnumerator .....	72
6.6.7	Метод Blocks.getParagraphsEnumerator .....	73
6.6.8	Метод Blocks.getTablesEnumerator .....	73
6.6.9	Метод Blocks.getShapesEnumerator .....	73
6.6.10	Метод Blocks.getFieldsEnumerator .....	73
6.7	Класс Bookmarks .....	74
6.7.1	Метод Bookmarks.getBookmarkRange .....	74

# МойОфис

6.7.2	Метод Bookmarks.removeBookmark .....	74
6.8	Класс Borders .....	74
6.9	Класс CaseSensitive .....	76
6.10	Класс Cell .....	76
6.10.1	Метод Cell.getRange .....	76
6.10.2	Метод Cell.setBorders .....	76
6.10.3	Метод Cell.getBorders .....	77
6.10.4	Метод Cell.setFormula .....	77
6.10.5	Метод Cell.setFormat .....	77
6.10.6	Метод Cell.getFormat .....	79
6.10.7	Метод Cell.getFormattedValue .....	80
6.10.8	Метод Cell.setFormattedValue .....	80
6.10.9	Метод Cell.unmerge .....	80
6.10.10	Метод Cell.getHyperlink .....	80
6.10.11	Метод Cell.setContent .....	80
6.10.12	Метод Cell.getRawValue .....	81
6.10.13	Метод Cell.getCustomFormat .....	81
6.10.14	Метод Cell.setCustomFormat .....	81
6.10.15	Метод Cell.setBool .....	81
6.10.16	Метод Cell.setNumber .....	81
6.10.17	Метод Cell.setText .....	82
6.10.18	Метод Cell.getFormulaAsString .....	82
6.10.19	Метод Cell.getCellProperties .....	82
6.10.20	Метод Cell.setCellProperties .....	82
6.10.21	Метод Cell.getParagraphProperties .....	83
6.10.22	Метод Cell.setParagraphProperties .....	83
6.10.23	Метод Cell.getPivotTable .....	83
6.10.24	Метод Cell.isPivotTableRoot .....	83
6.11	Класс CellFormat .....	83
6.12	Класс CellProperties .....	86
6.12.1	CellProperties.__eq__ .....	87
6.12.2	CellProperties.__ne__ .....	87
6.13	Класс CellPosition .....	88
6.13.1	Поле CellPosition.row .....	88

# МойОфис

6.13.2	Поле <code>CellPosition.column</code> .....	89
6.13.3	Метод <code>CellPosition.toString</code> .....	89
6.13.4	<code>CellPosition.__eq__</code> .....	89
6.13.5	<code>CellPosition.__ne__</code> .....	89
6.14	Класс <code>CellRange</code> .....	90
6.14.1	Метод <code>CellRange.autoFill</code> .....	90
6.14.2	Метод <code>CellRange.containsCell</code> .....	90
6.14.3	Метод <code>CellRange.copyInto</code> .....	91
6.14.4	Метод <code>CellRange.moveInto</code> .....	92
6.14.5	Метод <code>CellRange.getEnumerator</code> .....	92
6.14.6	Метод <code>CellRange.getBeginRow</code> .....	92
6.14.7	Метод <code>CellRange.getBeginColumn</code> .....	92
6.14.8	Метод <code>CellRange.getLastRow</code> .....	93
6.14.9	Метод <code>CellRange.getLastColumn</code> .....	93
6.14.10	Метод <code>CellRange.setBorders</code> .....	93
6.14.11	Метод <code>CellRange.insertCurrentDateTime</code> .....	94
6.14.12	Метод <code>CellRange.getCellProperties</code> .....	94
6.14.13	Метод <code>CellRange.setCellProperties</code> .....	94
6.14.14	Метод <code>CellRange.merge</code> .....	95
6.14.15	Метод <code>CellRange.getTable</code> .....	95
6.15	Класс <code>CellRangePosition</code> .....	95
6.15.1	Метод <code>CellRangePosition.toString</code> .....	96
6.15.2	<code>CellRangePosition.__eq__</code> .....	96
6.15.3	<code>CellRangePosition.__ne__</code> .....	96
6.16	Класс <code>Chart</code> .....	97
6.16.1	Метод <code>Chart.getFrame</code> .....	97
6.16.2	Метод <code>Chart.getType</code> .....	98
6.16.3	Метод <code>Chart.setType</code> .....	98
6.16.4	Метод <code>Chart.getRangesCount</code> .....	98
6.16.5	Метод <code>Chart.getRange</code> .....	98
6.16.6	Метод <code>Chart.getTitle</code> .....	98
6.16.7	Метод <code>Chart.setRange</code> .....	99
6.16.8	Метод <code>Chart.setRect</code> .....	99
6.16.9	Метод <code>Chart.isEmpty</code> .....	99



# МойОфис

6.16.10	Метод Chart.isSolidRange .....	99
6.16.11	Метод Chart.is3D .....	100
6.16.12	Метод Chart.getDirectionType .....	100
6.16.13	Метод Chart.getChartLabels .....	100
6.16.14	Метод Chart.getRangeAsString .....	100
6.16.15	Метод Chart.applySettings .....	100
6.17	Класс ChartLabelsDetectionMode .....	101
6.18	Класс ChartLabelsInfo .....	101
6.19	Класс ChartRangeInfo .....	102
6.20	Класс ChartRangeType .....	103
6.21	Класс Charts .....	104
6.21.1	Метод Charts.getChartsCount .....	104
6.21.2	Метод Charts.getChart .....	105
6.21.3	Метод Charts.getChartIndexByDrawingIndex .....	105
6.22	Класс ChartSeriesDirectionType .....	105
6.23	Класс ChartType .....	105
6.24	Класс Color .....	107
6.24.1	Метод Color.getRGBAColor .....	107
6.24.2	Метод Color.getThemeColorID .....	107
6.24.3	Метод Color.getTransforms .....	107
6.24.4	Метод Color.setTransforms .....	108
6.24.5	Метод Color.__eq__ .....	108
6.24.6	Метод Color.__ne__ .....	108
6.25	Класс ColorTransforms .....	108
6.25.1	Метод ColorTransforms::apply .....	109
6.26	Класс ColorRGBA .....	109
6.26.1	ColorRGBA.__eq__ .....	110
6.26.2	ColorRGBA.__ne__ .....	110
6.27	Класс Comment .....	110
6.27.1	Метод Comment.getAudioUrl .....	111
6.27.2	Метод Comment.getRange .....	111
6.27.3	Метод Comment.getText .....	111
6.27.4	Метод Comment.getInfo .....	111
6.27.5	Метод Comment.isResolved .....	112

# МойОфис

6.27.6	Метод Comment.getReplies .....	112
6.28	Класс Comments .....	112
6.28.1	Метод Comments.getEnumerator .....	113
6.29	Класс ConditionalTableFilter .....	113
6.29.1	Метод ConditionalTableFilter.setAndOperation .....	113
6.29.2	Методы добавления условий .....	114
6.30	Класс Connection .....	114
6.31	Класс CurrencyCellFormatting .....	115
6.32	Класс CurrencySignPlacement .....	116
6.33	Класс DatePatterns .....	116
6.34	Класс DateTime .....	117
6.34.1	DateTime.__eq__ .....	117
6.34.2	DateTime.__ne__ .....	117
6.35	Класс DateTimeCellFormatting .....	118
6.36	Класс DateTimeFormat .....	118
6.37	Класс Document .....	119
6.37.1	Метод Document.getAbsoluteFilePath .....	119
6.37.2	Метод Document.saveAs .....	119
6.37.3	Метод Document.exportAs .....	120
6.37.4	Метод Document.merge .....	121
6.37.5	Метод Document.getBlocks .....	122
6.37.6	Метод Document.getBookmarks .....	122
6.37.7	Метод Document.getScripts .....	122
6.37.8	Метод Document.getRange .....	122
6.37.9	Метод Document.isChangesTrackingEnabled .....	123
6.37.10	Метод Document.setChangesTrackingEnabled .....	123
6.37.11	Метод Document.getComments .....	123
6.37.12	Метод Document.setPageProperties .....	123
6.37.13	Метод Document.setFormulaType .....	123
6.37.14	Метод Document.getFormulaType .....	124
6.37.15	Метод Document.setPageOrientation .....	124
6.37.16	Метод Document.getSectionsEnumerator .....	124
6.37.17	Метод Document.getSections .....	124
6.37.18	Метод Document.setMirroredMarginsEnabled .....	124

# МойОфис

6.37.19	Метод Document.areMirroredMarginsEnabled .....	125
6.37.20	Метод Document.getPivotTablesManager .....	125
6.37.21	Метод Document.getNamedExpressions .....	125
6.38	Класс DocumentFormat .....	125
6.39	Класс DocumentSettings .....	126
6.40	Класс DocumentType .....	126
6.41	Класс DSVSettings .....	127
6.41.1	Метод DSVSettings.__eq__ .....	127
6.41.2	Метод DSVSettings.__ne__ .....	128
6.42	Класс Encoding .....	128
6.43	Класс ExportFormat .....	129
6.44	Класс Field .....	129
6.45	Класс Fill .....	129
6.45.1	Метод Fill.getColor .....	129
6.45.2	Метод Fill.getUrl .....	130
6.45.3	Метод Fill.isNoFill .....	130
6.46	Класс FiltersRange .....	130
6.46.1	Метод FiltersRange.clear .....	130
6.46.2	Метод FiltersRange.eraseFilters .....	130
6.46.3	Метод FiltersRange.getCellRange .....	130
6.46.4	Метод FiltersRange.setFilters .....	131
6.47	Класс FormulaType .....	131
6.48	Класс FractionCellFormatting .....	132
6.49	Класс Frame .....	133
6.50	Класс FrozenRangePosition .....	133
6.50.1	Конструкторы .....	134
6.50.2	Метод FrozenRangePosition.create .....	134
6.50.3	Метод FrozenRangePosition.createFrozenArea .....	134
6.50.4	Метод FrozenRangePosition.createFrozenRows .....	135
6.50.5	Метод FrozenRangePosition.createFrozenCols .....	135
6.50.6	Метод FrozenRangePosition.isRowsCols .....	135
6.50.7	Метод FrozenRangePosition.isArea .....	135
6.50.8	Метод FrozenRangePosition.isRows .....	136
6.50.9	Метод FrozenRangePosition.isCols .....	136

# МойОфис

6.51	Класс HeaderComponent .....	136
6.51.1	Метод HeaderComponent.getType .....	136
6.51.2	Метод HeaderComponent.getBlocks .....	136
6.51.3	Метод HeaderComponent.getRange .....	137
6.52	Класс HeaderComponentType .....	137
6.53	Класс HeadersFooters .....	137
6.53.1	Метод HeadersFooters.getEnumerator .....	138
6.54	Класс HorizontalAnchorAlignment .....	138
6.55	Класс HorizontalRelativeTo .....	139
6.56	Класс HorizontalTextAnchoredPosition .....	139
6.56.1	HorizontalTextAnchoredPosition.__eq__ .....	140
6.56.2	HorizontalTextAnchoredPosition.__ne__ .....	140
6.57	Класс Hyperlink .....	141
6.57.1	Hyperlink.__eq__ .....	141
6.57.2	Hyperlink.__ne__ .....	142
6.58	Класс Image .....	142
6.58.1	Метод Image.getFrame .....	142
6.58.2	Метод Image.remove .....	143
6.59	Класс Images .....	143
6.59.1	Метод Images.getEnumerator .....	143
6.60	Класс InlineFrame .....	144
6.60.1	Метод InlineFrame.setPosition .....	144
6.60.2	Метод InlineFrame.getPosition .....	146
6.60.3	Метод InlineFrame.setDimensions .....	146
6.60.4	Метод InlineFrame.getDimensions .....	147
6.60.5	Метод InlineFrame.setWrapType .....	147
6.60.6	Метод InlineFrame.getWrapType .....	147
6.61	Класс Insets .....	148
6.61.1	Insets.__eq__ .....	148
6.61.2	Insets.__ne__ .....	149
6.62	Класс ListSchema .....	149
6.63	Класс LineEndingProperties .....	151
6.63.1	LineEndingProperties.__eq__ .....	152
6.63.2	LineEndingProperties.__ne__ .....	153

# МойОфис

6.64	Класс LineEndingStyle .....	153
6.65	Класс LineProperties .....	154
6.65.1	Поле LineProperties.style .....	155
6.65.2	Поле LineProperties.width .....	155
6.65.3	Поле LineProperties.color .....	155
6.65.4	Поле LineProperties.headLineEndingProperties .....	155
6.65.5	Поле LineProperties.tailLineEndingProperties .....	155
6.65.6	LineProperties.__eq__ .....	155
6.65.7	LineProperties.__ne__ .....	156
6.66	Класс LineSpacing .....	156
6.66.1	LineSpacing.__eq__ .....	157
6.66.2	LineSpacing.__ne__ .....	157
6.67	Класс LineSpacingRule .....	158
6.68	Класс LineStyle .....	159
6.69	Класс LoadDocumentSettings .....	160
6.70	Класс LocaleInfo .....	161
6.71	Класс MediaObject .....	162
6.71.1	Метод MediaObject.toImage .....	162
6.71.2	Метод MediaObject.toChart .....	162
6.71.3	Метод MediaObject.getFrame .....	162
6.72	Класс MediaObjects .....	163
6.72.1	Метод MediaObjects.getEnumerator .....	163
6.73	Класс Message .....	163
6.73.1	Класс Message.Severity .....	164
6.73.2	Метод Message.getSeverity .....	164
6.73.3	Метод Message.getText .....	164
6.73.4	Метод Message.makeInfo .....	164
6.73.5	Метод Message.makeWarning .....	164
6.73.6	Метод Message.makeError .....	164
6.74	Класс Messenger .....	164
6.74.1	Метод Messenger.subscribe .....	164
6.74.2	Метод Messenger.notify .....	165
6.75	Класс NamedExpression .....	165
6.75.1	Метод NamedExpression.getName .....	165

# МойОфис

6.75.2	Метод NamedExpression.getExpression .....	165
6.75.3	Метод NamedExpression.getCellRange .....	165
6.76	Класс NamedExpressions .....	165
6.76.1	Метод NamedExpressions.get .....	166
6.76.2	Метод NamedExpressions.getEnumerator .....	166
6.76.3	Метод NamedExpression.addExpression .....	166
6.76.4	Метод NamedExpressions.removeExpression .....	166
6.77	Класс NamedExpressionsValidationResult .....	167
6.78	Класс NumberCellFormatting .....	167
6.79	Класс PageFieldOrder .....	168
6.80	Класс PageNumbers .....	168
6.80.1	Метод PageNumbers.contains .....	169
6.80.2	Метод PageNumbers.getLast .....	169
6.80.3	Метод PageNumbers.__eq__ .....	169
6.80.4	Метод PageNumbers.__ne__ .....	169
6.81	Класс PageOrientation .....	170
6.82	Класс PageParity .....	170
6.83	Класс PageProperties .....	171
6.83.1	PageProperties.__eq__ .....	171
6.83.2	PageProperties.__ne__ .....	172
6.84	Класс Paragraphs .....	172
6.84.1	Метод Paragraphs.setListSchema .....	172
6.84.2	Метод Paragraphs.setListLevel .....	173
6.84.3	Метод Paragraphs.increaseListLevel .....	173
6.84.4	Метод Paragraphs.decreaseListLevel .....	173
6.84.5	Метод Paragraphs.getEnumerator .....	174
6.85	Класс Paragraph .....	174
6.85.1	Метод Paragraph.getParagraphProperties .....	174
6.85.2	Метод Paragraph.setParagraphProperties .....	175
6.85.3	Метод Paragraph.getListSchema .....	176
6.85.4	Метод Paragraph.setListSchema .....	176
6.85.5	Метод Paragraph.getListLevel .....	176
6.85.6	Метод Paragraph.setListLevel .....	176
6.85.7	Метод Paragraph.increaseListLevel .....	177

# МойОфис

6.85.8	Метод Paragraph.decreaseListLevel .....	177
6.86	Класс ParagraphProperties .....	178
6.86.1	ParagraphProperties.__eq__ .....	180
6.86.2	ParagraphProperties.__ne__ .....	181
6.87	Класс PercentageCellFormatting .....	182
6.88	Класс PivotTablesManager .....	182
6.88.1	Метод PivotTablesManager.create .....	182
6.89	Класс PivotTable .....	183
6.89.1	Метод PivotTable.remove .....	183
6.89.2	Метод PivotTable.getSourceRangeAddress .....	183
6.89.3	Метод PivotTable.getSourceRange .....	183
6.89.4	Метод PivotTable.getPivotRange .....	184
6.89.5	Метод PivotTable.changeSourceRange .....	184
6.89.6	Метод PivotTable.isRowGrandTotalEnabled .....	184
6.89.7	Метод PivotTable.isColumnGrandTotalEnabled .....	184
6.89.8	Метод PivotTable.getPivotTableCaptions .....	184
6.89.9	Метод PivotTable.getPivotTableLayoutSettings .....	185
6.89.10	Метод PivotTable.getUnsupportedFeatures .....	185
6.89.11	Метод PivotTable.getFieldsList .....	185
6.89.12	Метод PivotTable.getRowFields .....	186
6.89.13	Метод PivotTable.getColumnFields .....	186
6.89.14	Метод PivotTable.getValueFields .....	186
6.89.15	Метод PivotTable.getPageFields .....	187
6.89.16	Метод PivotTable.getFieldCategories .....	187
6.89.17	Метод PivotTable.getFieldItems .....	187
6.89.18	Метод PivotTable.getFieldItemsByName .....	187
6.89.19	Метод PivotTable.getFilter .....	188
6.89.20	Метод PivotTable.getFilters .....	188
6.89.21	Метод PivotTable.update .....	188
6.89.22	Метод PivotTable.createPivotTableEditor .....	188
6.90	Класс PivotTableCaptions .....	189
6.91	Класс PivotTableCategoryField .....	189
6.92	Класс PivotTableEditor .....	190
6.92.1	Метод PivotTableEditor.addField .....	190

# МойОфис

6.92.2	Метод PivotTableEditor.moveField .....	190
6.92.3	Метод PivotTableEditor.removeField .....	190
6.92.4	Метод PivotTableEditor.reorderField .....	191
6.92.5	Метод PivotTableEditor.enableField .....	191
6.92.6	Метод PivotTableEditor.disableField .....	191
6.92.7	Метод PivotTableEditor.setSummarizeFunction .....	191
6.92.8	Метод PivotTableEditor.setFilter .....	192
6.92.9	Метод PivotTableEditor.setFilters .....	192
6.92.10	Метод PivotTableEditor.setCaptions .....	192
6.92.11	Метод PivotTableEditor.setLayoutSettings .....	193
6.92.12	Метод PivotTableEditor.setGrandTotalSettings .....	193
6.92.13	Метод PivotTableEditor.apply .....	193
6.93	Класс PivotTableFieldCategories .....	194
6.93.1	Метод PivotTableFieldCategories.getEnumerator .....	194
6.94	Класс PivotTableFilters .....	194
6.94.1	Метод PivotTableFilters.getEnumerator .....	194
6.95	Класс PivotTableFunction .....	195
6.96	Класс PivotTableFilter .....	195
6.96.1	Метод PivotTableFilter.getFieldName .....	196
6.96.2	Метод PivotTableFilter.getCount .....	196
6.96.3	Метод PivotTableFilter.getName .....	196
6.96.4	Метод PivotTableFilter.isHidden .....	196
6.96.5	Метод PivotTableFilter.setHidden .....	197
6.97	Класс PivotTableField .....	197
6.98	Класс PivotTableFieldCategory .....	197
6.99	Класс PivotTableFieldProperties .....	198
6.100	Класс PivotTableItem .....	198
6.100.1	Метод PivotTableItem.getName .....	198
6.100.2	Метод PivotTableItem.getAlias .....	199
6.100.3	Метод PivotTableItem.getItemType .....	199
6.100.4	Метод PivotTableItem.isCollapsed .....	199
6.101	Класс PivotTableItems .....	199
6.101.1	Метод PivotTableItems.getEnumerator .....	199
6.102	Класс PivotTableItemType .....	200



# МойОфис

6.103 Класс PivotTableLayoutSettings .....	200
6.104 Класс PivotTablePageField .....	201
6.105 Класс PivotTableReportLayout .....	201
6.106 Класс PivotTableValueField .....	202
6.107 Класс PivotTableUnsupportedFeature .....	202
6.108 Класс PivotTableUpdateResult .....	203
6.109 Класс PointU .....	204
6.109.1 PointU.toString .....	204
6.110 Класс Position .....	204
6.110.1 Метод Position.getCell .....	205
6.110.2 Метод Position.insertText .....	205
6.110.3 Метод Position.insertTable .....	205
6.110.4 Метод Position.insertPageBreak .....	206
6.110.5 Метод Position.insertLineBreak .....	206
6.110.6 Метод Position.insertBookmark .....	206
6.110.7 Метод Position.insertSectionBreak .....	206
6.110.8 Метод Position.insertHyperlink .....	206
6.110.9 Метод Position.insertImage .....	207
6.110.10 Метод Position.removeBackward .....	207
6.110.11 Метод Position.removeForward .....	207
6.110.12 Position.__eq__ .....	208
6.110.13 Position.__ne__ .....	208
6.111 Класс PrintingScope .....	208
6.111.1 Метод PrintingScope.getCellRange .....	209
6.111.2 Метод PrintingScope.usePrintArea .....	209
6.112 Класс PrintingScope.Type .....	209
6.113 Класс Range .....	209
6.113.1 Метод Range.getBegin .....	211
6.113.2 Метод Range.getEnd .....	211
6.113.3 Метод Range.extractText .....	212
6.113.4 Метод Range.removeContent .....	212
6.113.5 Метод Range.lockContent .....	213
6.113.6 Метод Range.unlockContent .....	213
6.113.7 Метод Range.isContentLocked .....	214

# МойОфис

6.113.8	Метод Range.replaceText .....	214
6.113.9	Метод Range.setHyperlink .....	214
6.113.10	Метод Range.getTextProperties .....	215
6.113.11	Метод Range.setTextProperties .....	215
6.113.12	Метод Range.getBlocksEnumerator .....	216
6.113.13	Метод Range.getTrackedChangesEnumerator .....	216
6.113.14	Метод Range.getComments .....	217
6.113.15	Метод Range.getParagraphs .....	217
6.113.16	Метод Range.getImages .....	217
6.113.17	Метод Range.getInlineObjects .....	218
6.113.18	Метод Range.__eq__ .....	218
6.113.19	Метод Range.__ne__ .....	218
6.114	Класс RangeBorders .....	218
6.115	Класс RectU .....	219
6.115.1	RectU.toString .....	219
6.116	Класс SaveDocumentSettings .....	219
6.117	Класс ScaleFrom .....	220
6.118	Класс Script .....	220
6.118.1	Метод Script.getName .....	220
6.118.2	Метод Script.setName .....	220
6.118.3	Метод Script.getBody .....	221
6.118.4	Метод Script.setBody .....	221
6.119	Класс ScriptPosition .....	221
6.120	Класс ScientificCellFormatting .....	222
6.121	Класс Scripts .....	222
6.121.1	Метод Scripts.getScript .....	223
6.121.2	Метод Scripts.setScript .....	223
6.121.3	Метод Scripts.removeScript .....	223
6.121.4	Метод Scripts.getEnumerator .....	224
6.122	Класс Scripting .....	224
6.122.1	Метод Scripting.runScript .....	224
6.123	Класс Search .....	224
6.123.1	Метод Search.findText .....	225
6.124	Класс Section .....	226

# МойОфис

6.124.1	Метод Section.setPageProperties .....	226
6.124.2	Метод Section.getPageProperties .....	226
6.124.3	Метод Section.setPageOrientation .....	226
6.124.4	Метод Section.getPageOrientation .....	226
6.124.5	Метод Section.getRange .....	226
6.124.6	Метод Section.getHeaders .....	227
6.124.7	Метод Section.getFooters .....	227
6.125	Класс Sections .....	227
6.125.1	Метод Sections.getEnumerator .....	227
6.126	Класс Shape .....	227
6.126.1	Метод Shape.getShapeProperties .....	228
6.126.2	Метод Shape.setShapeProperties .....	228
6.127	Класс ShapeProperties .....	228
6.127.1	Поле ShapeProperties.borderProperties .....	228
6.127.2	Поле ShapeProperties.verticalAlignment .....	228
6.127.3	Поле ShapeProperties.fill .....	228
6.127.4	Поле ShapeProperties.shapeTextLayout .....	229
6.128	Класс ShapeTextLayout .....	229
6.129	Класс SizeU .....	229
6.129.1	Метод SizeU.toString .....	229
6.130	Класс Table .....	230
6.130.1	Метод Table.clearColumnGroups .....	230
6.130.2	Метод Table.clearRowGroups .....	230
6.130.3	Метод Table.createFiltersRange .....	231
6.130.4	Метод Table.duplicate .....	231
6.130.5	Метод Table.freeze .....	232
6.130.6	Метод Table.getCell .....	232
6.130.7	Метод Table.getCellRange .....	232
6.130.8	Метод Table.getCharts .....	233
6.130.9	Метод Table.getColumnsCount .....	233
6.130.10	Метод Table.getFiltersRange .....	233
6.130.11	Метод Table.getFrozenRange .....	233
6.130.12	Метод Table.getImages .....	234
6.130.13	Метод Table.getMediaObjects .....	234

# МойОфис

6.130.14	Метод Table.getName .....	234
6.130.15	Метод Table.getNamedExpressions .....	234
6.130.16	Метод Table.getPrintAreas .....	235
6.130.17	Метод Table.getRowsCount .....	235
6.130.18	Метод Table.getShowZeroValue .....	235
6.130.19	Метод Table.groupColumns .....	235
6.130.20	Метод Table.groupRows .....	236
6.130.21	Метод Table.insertColumnAfter .....	236
6.130.22	Метод Table.insertColumnBefore .....	237
6.130.23	Метод Table.insertRowAfter .....	237
6.130.24	Метод Table.insertRowBefore .....	238
6.130.25	Метод Table.isVisible .....	238
6.130.26	Метод Table.isColumnVisible .....	239
6.130.27	Метод Table.isRowVisible .....	239
6.130.28	Метод Table.remove .....	240
6.130.29	Метод Table.moveTo .....	240
6.130.30	Метод Table.removeColumn .....	240
6.130.31	Метод Table.removeRow .....	240
6.130.32	Метод Table.setColumnWidth .....	241
6.130.33	Метод Table.setColumnsVisible .....	241
6.130.34	Метод Table.setPrintArea .....	242
6.130.35	Метод Table.setPrintAreas .....	242
6.130.36	Метод Table.setShowZeroValue .....	242
6.130.37	Метод Table.setRowHeight .....	242
6.130.38	Метод Table.setRowsVisible .....	243
6.130.39	Метод Table.setName .....	243
6.130.40	Метод Table.setVisible .....	244
6.130.41	Метод Table.ungroupRows .....	244
6.130.42	Метод Table.ungroupColumns .....	244
6.130.43	Table.__eq__ .....	244
6.130.44	Table.__ne__ .....	245
6.131	Класс TableFilters .....	245
6.131.1	Метод TableFilters.clear .....	245
6.131.2	Метод TableFilters.erase .....	246

# МойОфис

6.131.3	Метод TableFilters.setFilter .....	246
6.132	Класс TableRangeInfo .....	247
6.133	Класс TextAnchoredPosition .....	247
6.133.1	TextAnchoredPosition.__eq__ .....	247
6.133.2	TextAnchoredPosition.__ne__ .....	248
6.134	Класс TextExportSettings .....	249
6.135	Класс TextLayout .....	249
6.136	Класс TextOrientation .....	250
6.136.1	Метод TextOrientation.getAngle .....	251
6.136.2	TextOrientation.isStackedChars .....	251
6.136.3	TextOrientation.__eq__ .....	251
6.136.4	TextOrientation.__ne__ .....	251
6.137	Класс TextProperties .....	251
6.137.1	TextProperties.__eq__ .....	253
6.137.2	TextProperties.__ne__ .....	254
6.138	Класс TextWrapType .....	254
6.139	Класс ThemeColorID .....	254
6.140	Класс TimePatterns .....	255
6.141	Класс TimeZone .....	255
6.142	Класс TrackedChange .....	256
6.142.1	Метод TrackedChange.getRange .....	256
6.142.2	Метод TrackedChange.getType .....	256
6.142.3	Метод TrackedChange.getInfo .....	257
6.143	Класс TrackedChangeInfo .....	257
6.144	Класс TrackedChangeType .....	258
6.145	Класс UserInfo .....	258
6.145.1	Метод UserInfo.__eq__ .....	258
6.145.2	Метод UserInfo.__ne__ .....	259
6.146	Класс ValueFieldsOrientation .....	259
6.147	Класс ValuesTableFilter .....	259
6.147.1	Метод ValuesTableFilter.add .....	260
6.147.2	Метод ValuesTableFilter.clear .....	260
6.148	Класс VerticalAlignment .....	260
6.149	Класс VerticalAnchorAlignment .....	261

# МойОфис

6.150	Класс VerticalRelativeTo .....	262
6.151	Класс VerticalTextAnchoredPosition .....	262
6.151.1	VerticalTextAnchoredPosition.__eq__ .....	263
6.151.2	VerticalTextAnchoredPosition.__ne__ .....	263
6.152	Класс WorkbookExportSettings .....	264
6.153	Исключения .....	265
6.153.1	Класс BaseError .....	265
6.153.2	Класс ApplicationCreateError .....	265
6.153.3	Класс IncorrectArgumentError .....	265
6.153.4	Класс InvalidObjectError .....	265
6.153.5	Класс DocumentCreateError .....	265
6.153.6	Класс DocumentLoadError .....	265
6.153.7	Класс DocumentSaveError .....	265
6.153.8	Класс DocumentExportError .....	266
6.153.9	Класс NoSuchElementError .....	266
6.153.10	Класс NotImplementedError .....	266
6.153.11	Класс OutOfRangeError .....	266
6.153.12	Класс ParseError .....	266
6.153.13	Класс UnknownError .....	266
6.153.14	Класс ForbiddenActionError .....	266
6.153.15	Класс DocumentModificationError .....	267
6.153.16	Класс PivotTableError .....	267
6.153.17	Класс PositionDocumentMismatchError .....	267
6.153.18	Класс ScriptExecutionError .....	267
7	Версии Document API .....	268
7.1	Механизм контроля версий .....	268

## 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-3.1-cp38-abi3-win_amd64.whl
Linux	MyOfficeSDKDocumentAPI-3.1-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-3.1-cp38-abi3-win\_amd64.whl.**



Внимание! Следует убедиться в актуальности установленной версии Python. Для использования MyOffice Document API 3.1 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-3.1-cp38-abi3-linux\_x86\_64.whl.**



Внимание! Следует убедиться в актуальности установленной версии Python. Для использования MyOffice Document API 3.1 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;

# МойОфис

- 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.

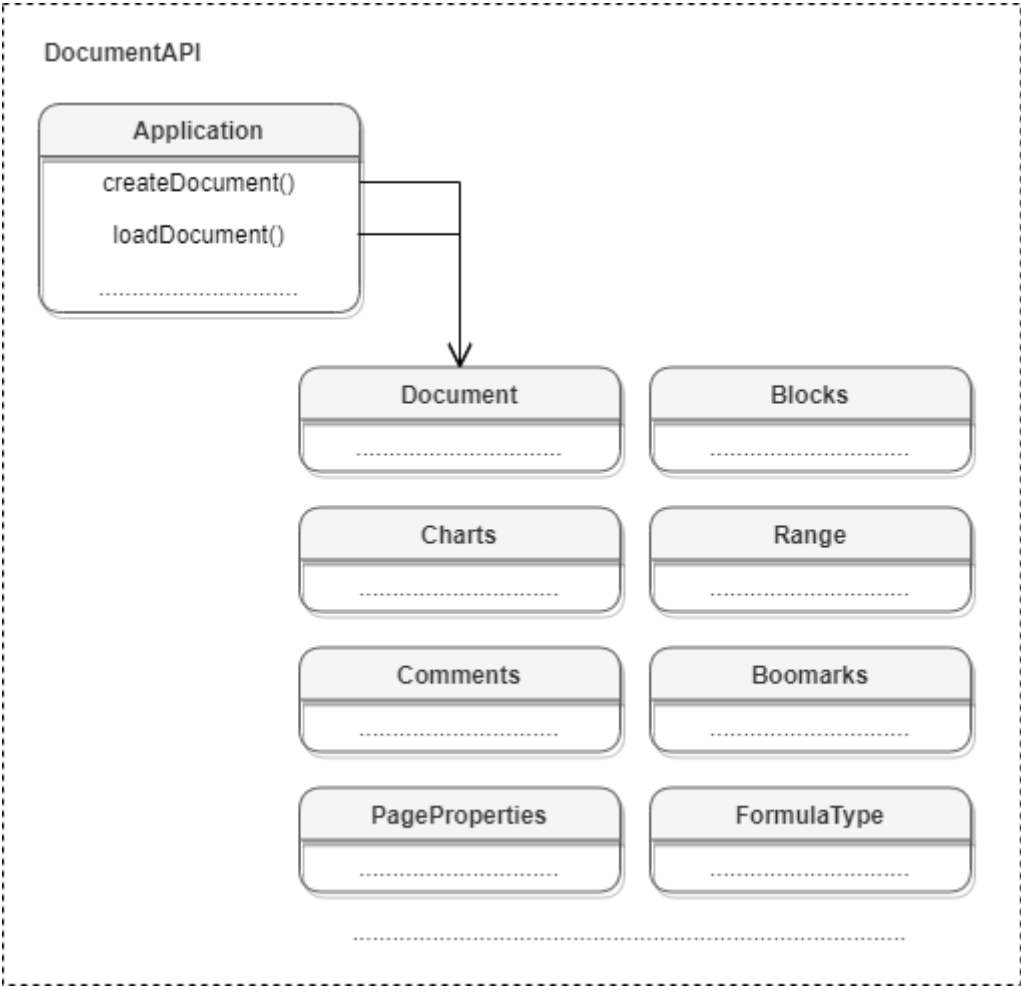


Рисунок 1 – Объектная модель МойОфис SDK.

## 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.

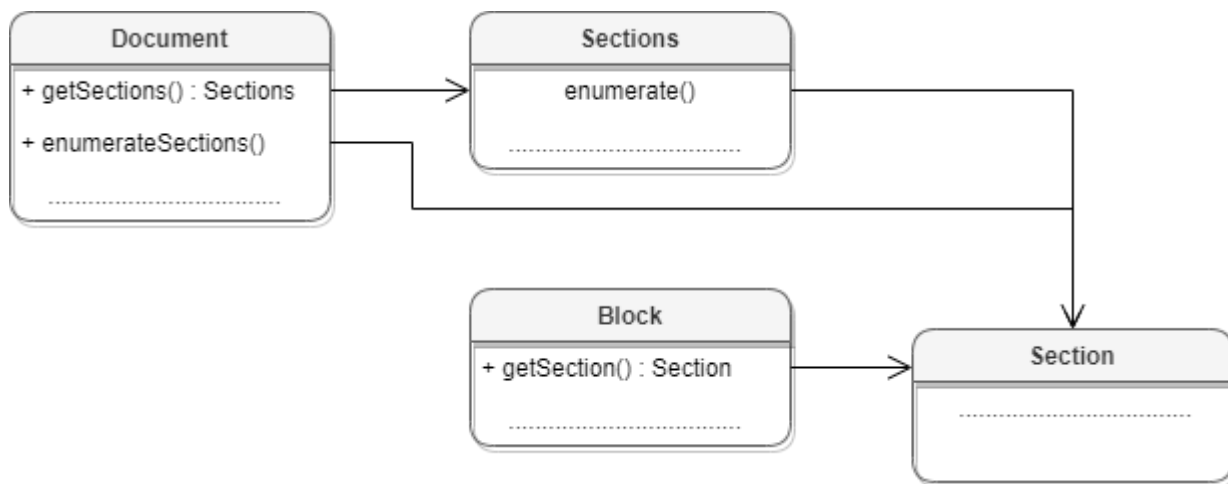


Рисунок 2 – Объектная модель классов для работы с секциями

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()  
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 табличном документе таблицами являются страницы документа.

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

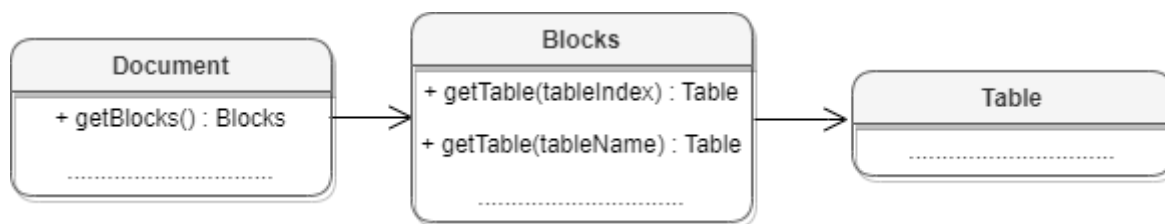


Рисунок 3 – Объектная модель для работы с таблицами

## 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).

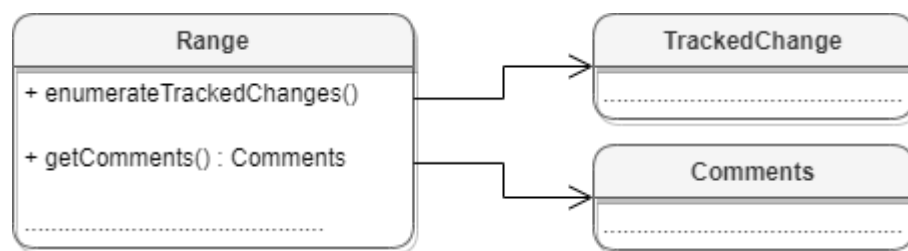


Рисунок 4 – Инструменты рецензирования документа

## 4.2 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.

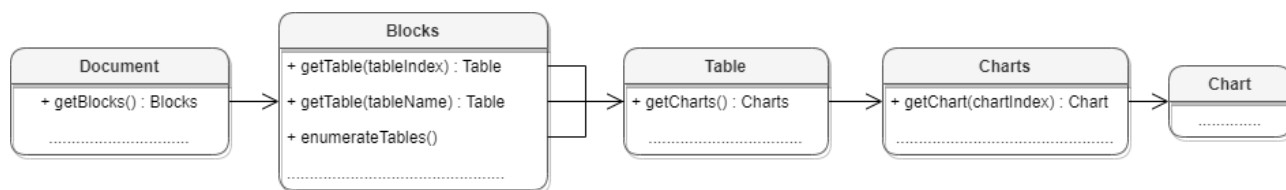


Рисунок 5 – Объектная модель классов для работы с диаграммами

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.moveTo\(\)](#):

```
sourceRange.moveTo(destRange)
```

## 4.2.5 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 встроенным объектам табличного документа осуществляется посредством использования методов [Range::getInlineObjects\(\)](#), [Table.getImages\(\)](#), [Table.getMediaObjects\(\)](#).

### 4.2.5.1 Вставка изображения

Вставка изображений в табличный документ на данный момент не поддерживается.

### 4.2.5.2 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)
```

### 4.2.6 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.

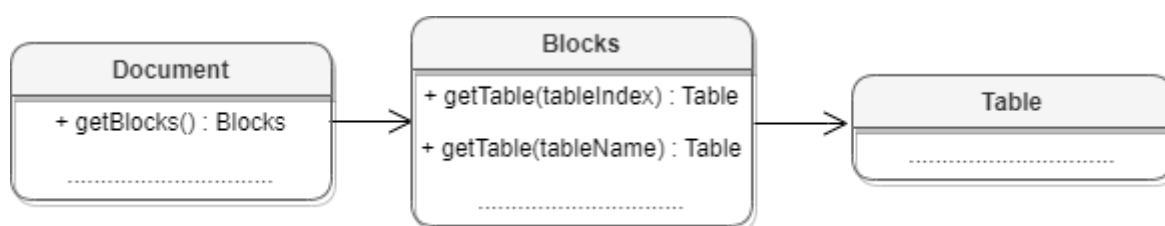


Рисунок 6 – Объектная модель для работы с таблицами

## 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()
```

### 4.2.7 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.

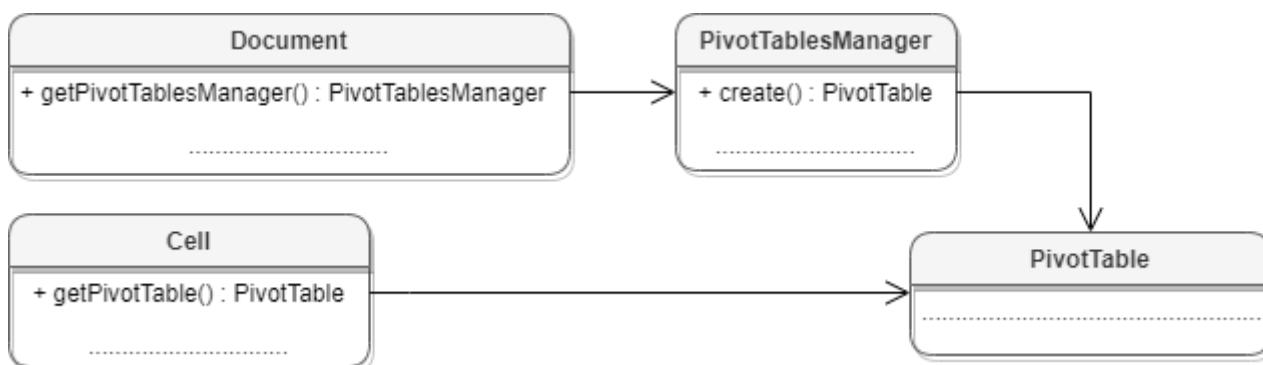


Рисунок 7 – Сводные таблицы

#### 4.2.7.1 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())
```

## 4.2.7.2 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())
```

## 4.2.7.3 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())
```

## 4.2.7.4 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)
```

## 4.2.7.5 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())
```

## 4.2.7.6 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)
```

## 4.2.7.7 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()
```

## 4.2.7.8 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)
```

## 4.2.7.9 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)
```

### 4.2.7.10 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)
```

### 4.2.7.11 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)
```

## 4.2.7.12 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)
```

## 4.2.7.13 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:
```

## 4.2.8 Работа с фильтрами

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

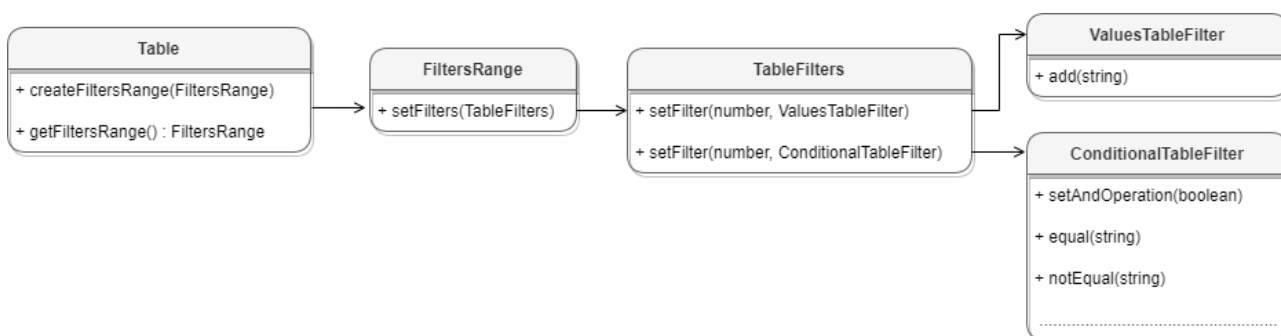


Рисунок 8 – Объектная модель таблиц для работы с фильтрами

Диапазон ячеек для фильтров [FiltersRange](#) формируется посредством метода [Table.createFiltersRange\(\)](#).



# МойОфис

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);
```

### 4.2.9 Встроенные объекты

#### 4.2.9.1 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")
```

## 4.2.9.2 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).

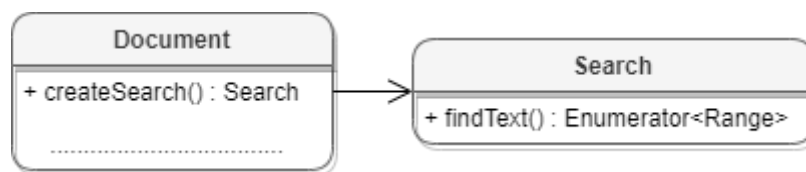


Рисунок 9 – Объектная модель для поиска в документе

## 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 макросами

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

Класс [Scripts](#) предназначен для доступа к списку макросов, доступен через метод [Document.getScripts\(\)](#), таблица [Scripting](#) служит для запуска макросов, доступна через [Scripting.createScripting\(document\)](#).

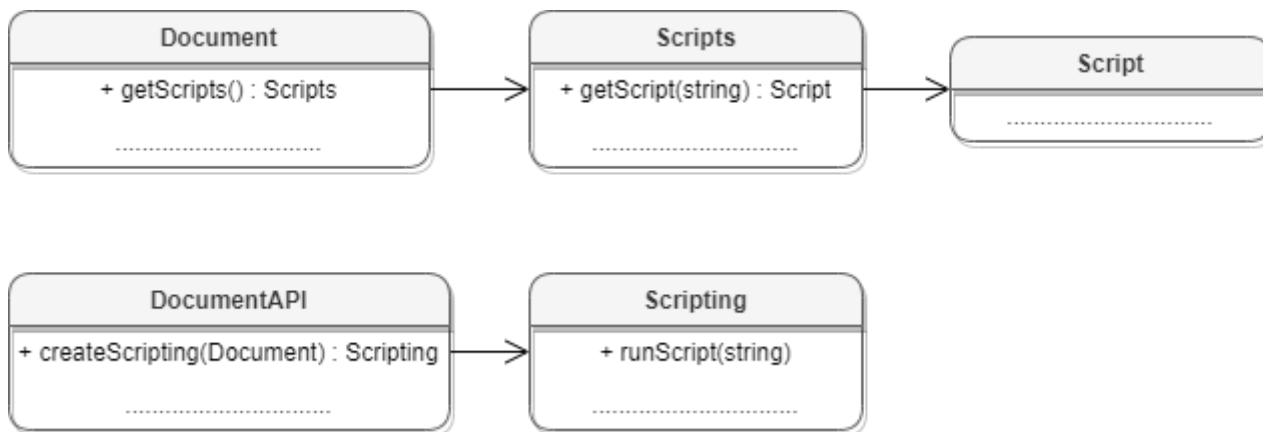


Рисунок 10 – Объектная модель классов для работы с макрокомандами

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).

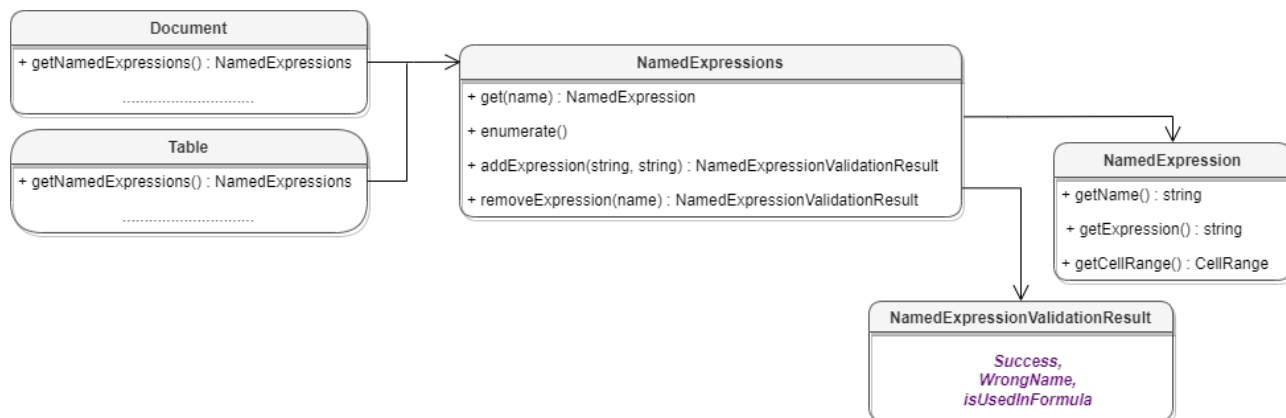


Рисунок 11 – Классы для работы с именованными диапазонами

## 4.5.1 Доступ к именованным диапазонам

Доступ к именованному диапазону 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.clearColumnGroups\(\)](#).

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

Методы могут вызвать исключения [DocumentAPI.OutOfRangeException](#) 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\(\)](#).

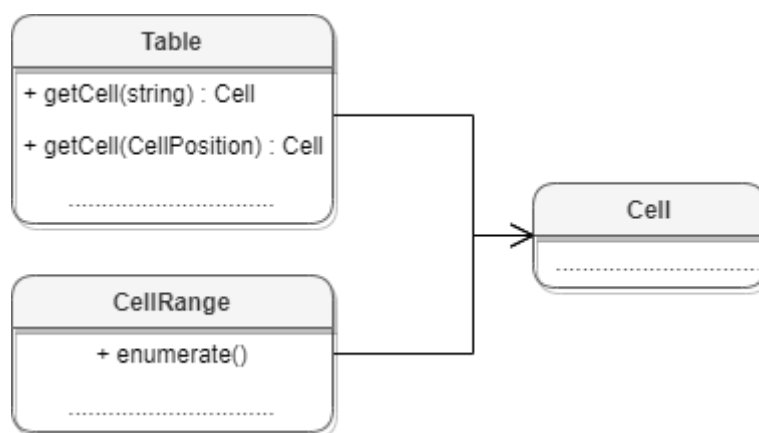


Рисунок 12 – Объектная модель для работы с ячейками таблиц

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")

cell11 = table1.getCell("A1")
cell12 = table1.getCell("C4")
cell13 = table1.getCell("E4")

print(cellRange1.containsCell(cell11))
print(cellRange1.containsCell(cell12))
print(cellRange1.containsCell(cell13))

print(cellRange2.containsCell(cell11))
print(cellRange2.containsCell(cell12))
print(cellRange2.containsCell(cell13))
```

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

# МойОфис

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 этими 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`.

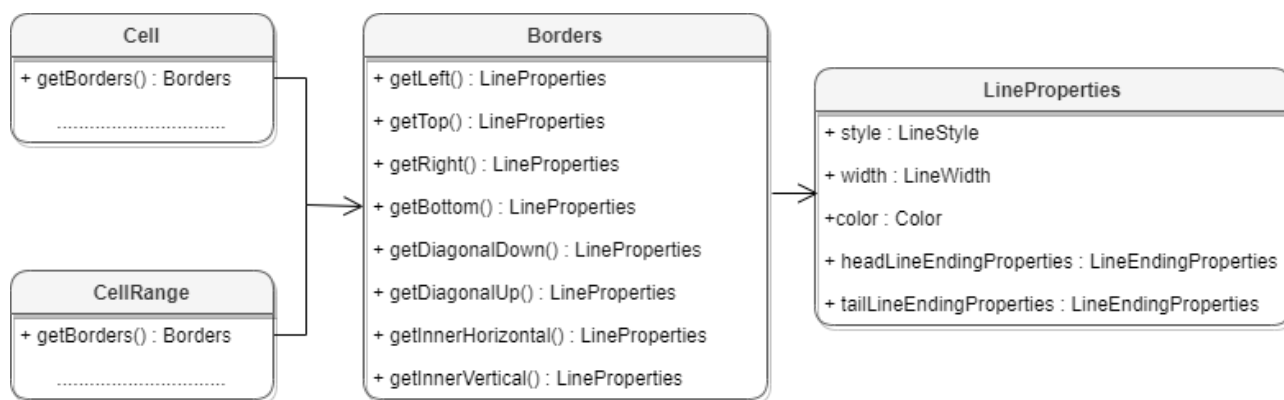


Рисунок 13 – Классы для работы с границами ячеек

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

- получить ячейку [Cell](#) или область ячеек [CellRange](#);
- настроить параметры для рисования линии границы с помощью экземпляра класса [LineProperties](#);
- настроить свойства линии: левой границы, верхней границы и т.д. с помощью экземпляра класса [Borders](#);
- 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, содержащиеся 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")
```

## 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.

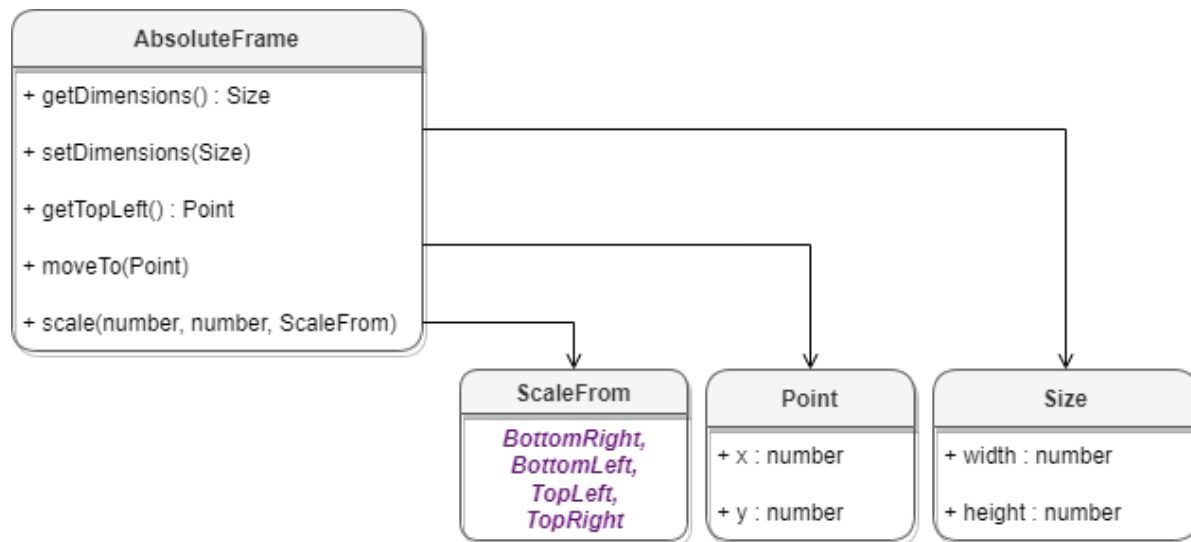


Рисунок 14 – Объектная модель класса `AbsoluteFrame`

#### 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.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)
```



## 6.1.2 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.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)
```

## 6.1.4 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))
```

## 6.1.5 Метод `AbsoluteFrame.getDimensions`

Возвращает размеры медиаобъекта, тип - [SizeU](#).

## 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
<code>AccountingCellFormatting.decimalPlaces</code>	Количество десятичных позиций
<code>AccountingCellFormatting.symbol</code>	Символ денежной единицы
<code>AccountingCellFormatting.localeCode</code>	Идентификатор кода языка (MS-LCID)
<code>AccountingCellFormatting.fillSymbol</code>	Символ заполнения
<code>AccountingCellFormatting.useThousandsSeparator</code>	Использовать разделитель для тысячных
<code>AccountingCellFormatting.currencySignPlacement</code>	Тип размещения знака валюты <a href="#">CurrencySignPlacement</a>

### Пример:

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

accountingCellFormat = myOfficeSDK.AccountingCellFormatting()
accountingCellFormat.decimalPlaces = 2
accountingCellFormat.symbol = "Pyб"

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
<code>Alignment_Default</code>	Выравнивание текста по умолчанию
<code>Alignment_Left</code>	Выравнивание текста по левому краю
<code>Alignment_Center</code>	Выравнивание текста по центру
<code>Alignment_Right</code>	Выравнивание по правому краю
<code>Alignment_Justify</code>	Выравнивание по ширине
<code>Alignment_Distributed</code>	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
<code>Alignment_Fill</code>	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.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)
```

## 6.4.3 Метод Application.getMessenger

Метод `Application.getMessenger` возвращает объект [Messenger](#), реализующий логирование событий.

### Пример:

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

## 6.5 Класс Block

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

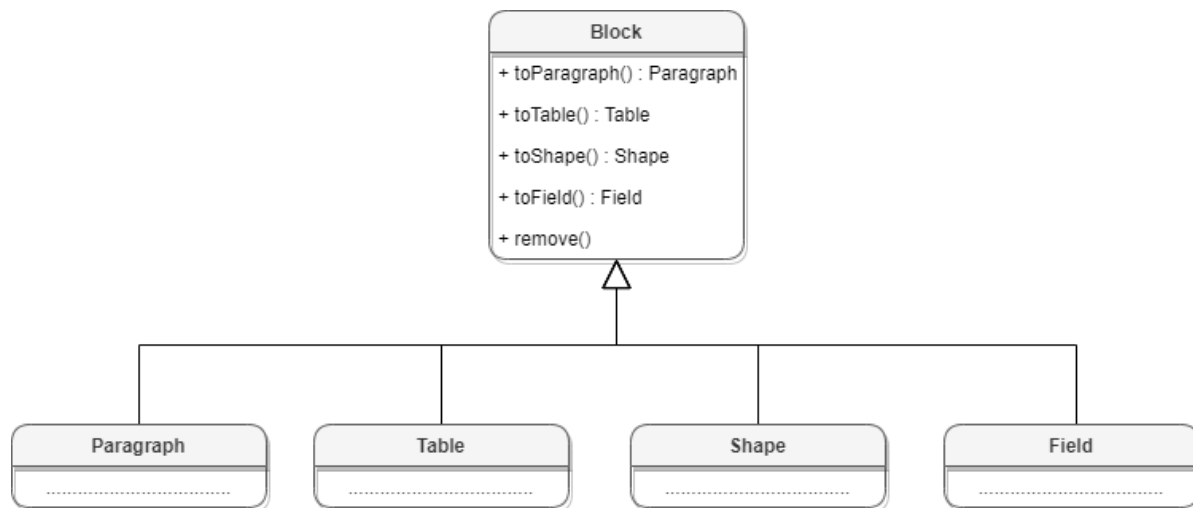


Рисунок 15 – Объектная модель класса `Block`

### 6.5.1 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())
```

## 6.5.2 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())
```

## 6.5.3 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 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())
```

## 6.6 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](#).

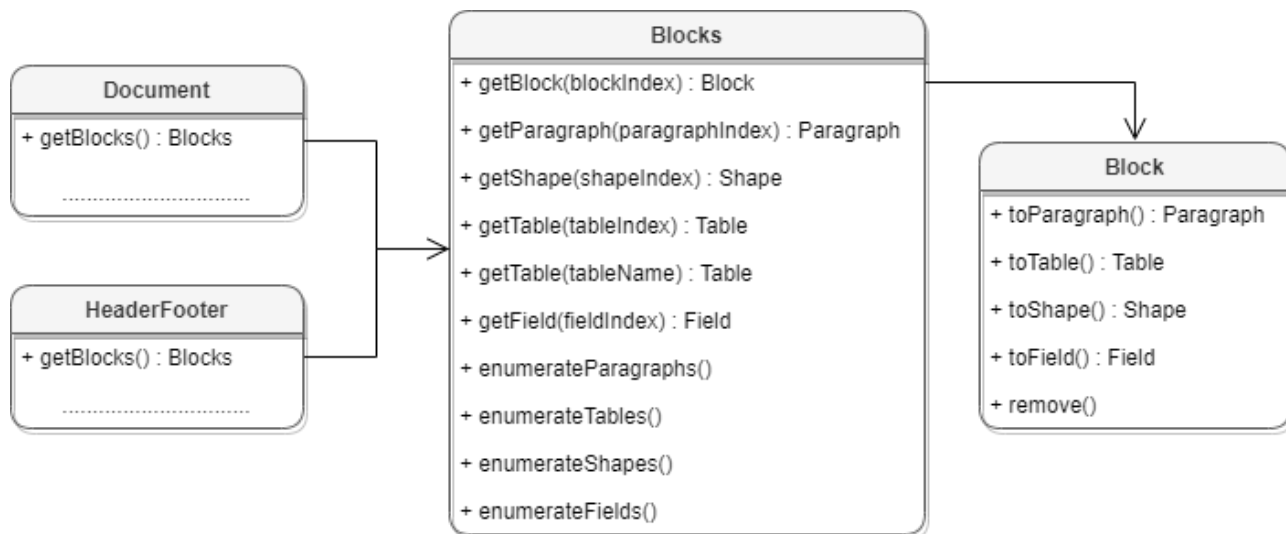


Рисунок 16 – Объектная модель класса `Blocks`

## 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.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())
```

## 6.6.3 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())
```

### 6.6.4 Метод `Blocks.getShape`

Возвращает фигуру [Shape](#) по заданному индексу.

## Пример:

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

### 6.6.5 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())
```

### 6.6.6 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())
```

## 6.6.7 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())
```

## 6.6.8 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())
```

## 6.6.9 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())
```

## 6.6.10 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())
```

## 6.7 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
<code>Borders.setDiagonalUp</code>	Установка диагональной линии, метод возвращает <code>Borders</code> 
<code>Borders.setOuter</code>	Установка внешних границ ячейки, метод возвращает <code>Borders</code>
<code>Borders.setDiagonals</code>	Установка обоих типов диагональных линий одновременно, метод возвращает <code>Borders</code>
<code>Borders.setInnerHorizontal</code>	Установка внутренних горизонтальных границ ячейки, метод возвращает <code>Borders</code>
<code>Borders.setInnerVertical</code>	Установка внутренних вертикальных границ ячейки, метод возвращает <code>Borders</code>
<code>Borders.setInner</code>	Установка внутренних границ ячейки, метод возвращает <code>Borders</code>
<code>Borders.setAll</code>	Установка всех границ ячейки, метод возвращает <code>Borders</code>
<code>Borders.getLeft</code>	Получение левой границы ячейки
<code>Borders.getRight</code>	Получение правой границы ячейки
<code>Borders.getTop</code>	Получение верхней границы ячейки
<code>Borders.getBottom</code>	Получение нижней границы ячейки
<code>Borders.getDiagonalDown</code>	Получение диагональной линии
<code>Borders.getDiagonalUp</code>	Получение диагональной линии
<code>Borders.getInnerHorizontal</code>	Получение внутренних горизонтальных границ ячейки
<code>Borders.getInnerVertical</code>	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 CaseSensitive

Параметры настройки учета регистра при поиске (см. метод [Search.FindText\(\)](#)) представлены в таблице 6.

Таблица 6 – Параметры регистра при поиске

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

## 6.10 Класс Cell

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

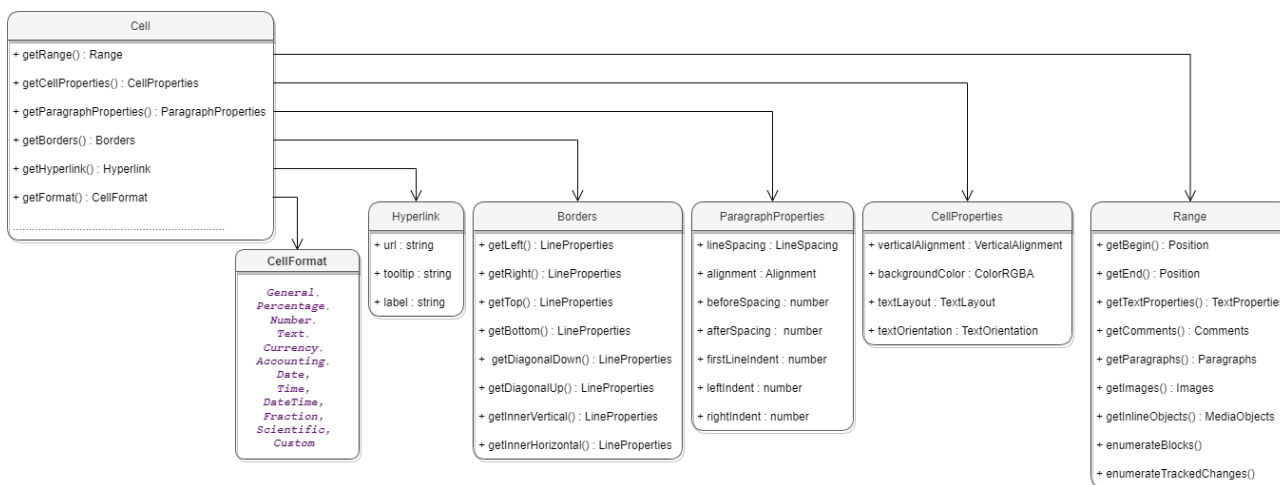


Рисунок 17 – Объектная модель ячейки таблиц

### 6.10.1 Метод Cell.getRange

Метод возвращает объект [Range](#) для управления содержимым ячейки.

### 6.10.2 Метод Cell.setBorders

Метод предназначен для установки границ ячейки. Примеры использования приведены в разделе [Borders](#).

## 6.10.3 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)
```

## 6.10.4 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())
```

## 6.10.5 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 : Экспоненциальный
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
```

## 6.10.6 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())
```

## 6.10.7 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())
```

## 6.10.8 Метод `Cell.setFormattedValue`

Анализирует переданное значение и автоматически устанавливает формат ячейки и ее значение. В случае, если распознать тип переданного значения не удастся, то для ячейки устанавливается формат `CellFormat.Text`.

Список поддерживаемых форматов см. в разделе [CellFormat](#).

## 6.10.9 Метод `Cell.unmerge`

Разъединяет несколько ячеек, которые были объединены ранее.

### Пример:

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

## 6.10.10 Метод `Cell.getHyperlink`

Возвращает первый объект в ячейке типа [Hyperlink](#).

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

## 6.10.11 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")
```



## 6.10.12 Метод `Cell.getRawValue`

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

**Пример:**

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

## 6.10.13 Метод `Cell.getCustomFormat`

Возвращает строку формата ячейки.

**Пример:**

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

## 6.10.14 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())
```

## 6.10.15 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())
```

## 6.10.16 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())
```

## 6.10.17 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())
```

## 6.10.18 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())
```

## 6.10.19 Метод Cell.getCellProperties

Позволяет получить свойства [CellProperties](#) ячейки.

**Пример:**

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

## 6.10.20 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)
```

## 6.10.21 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)
```

## 6.10.22 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)
```

## 6.10.23 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())
```

## 6.10.24 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 Класс `CellFormat`

По умолчанию при создании документа всем ячейкам присваивается формат

«Общий». Полный список форматов представлен в таблице 7.

Таблица 7 – Поддерживаемые форматы ячеек таблицы

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

Наименование константы	Описание
	Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.
CellFormat_Fraction	<p>Формат ячейки «Дробный».</p> <p>Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель).</p>
CellFormat_Scientific	<p>Формат ячейки «Экспоненциальный».</p> <p>Экспоненциальный (или научный) формат используется для представления больших чисел в короткой форме. Все введенные числа длиной более 12 символов автоматически переводятся в этот формат.</p> <p>В ячейке число в формате «Экспоненциальный» представлено следующим образом:</p> <ul style="list-style-type: none"> <li>– целая часть, всегда состоящая из одной цифры;</li> <li>– разделитель целой и дробной части;</li> <li>– дробная часть, по умолчанию состоящая из двух цифр;</li> <li>– показатель степени числа 10 в виде <b>Е&lt;знак показателя степени&gt; &lt;показатель степени&gt;</b>.</li> </ul>
CellFormat_Custom	Пользовательский формат.

Использование данных констант позволяет установить выбранный формат. При этом будут 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	<u>CellFormat.General</u>	1
2	<u>CellFormat.Percentage</u>	100,00%
3	<u>CellFormat.Number</u>	1,00

Пример форматирования ячейки также приведен в [разделе](#), описывающем установку значений ячеек.

## 6.12 Класс CellProperties

Класс CellProperties предназначен для форматирования содержимого в ячейках таблицы. Описание полей представлено в таблице 8.

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.

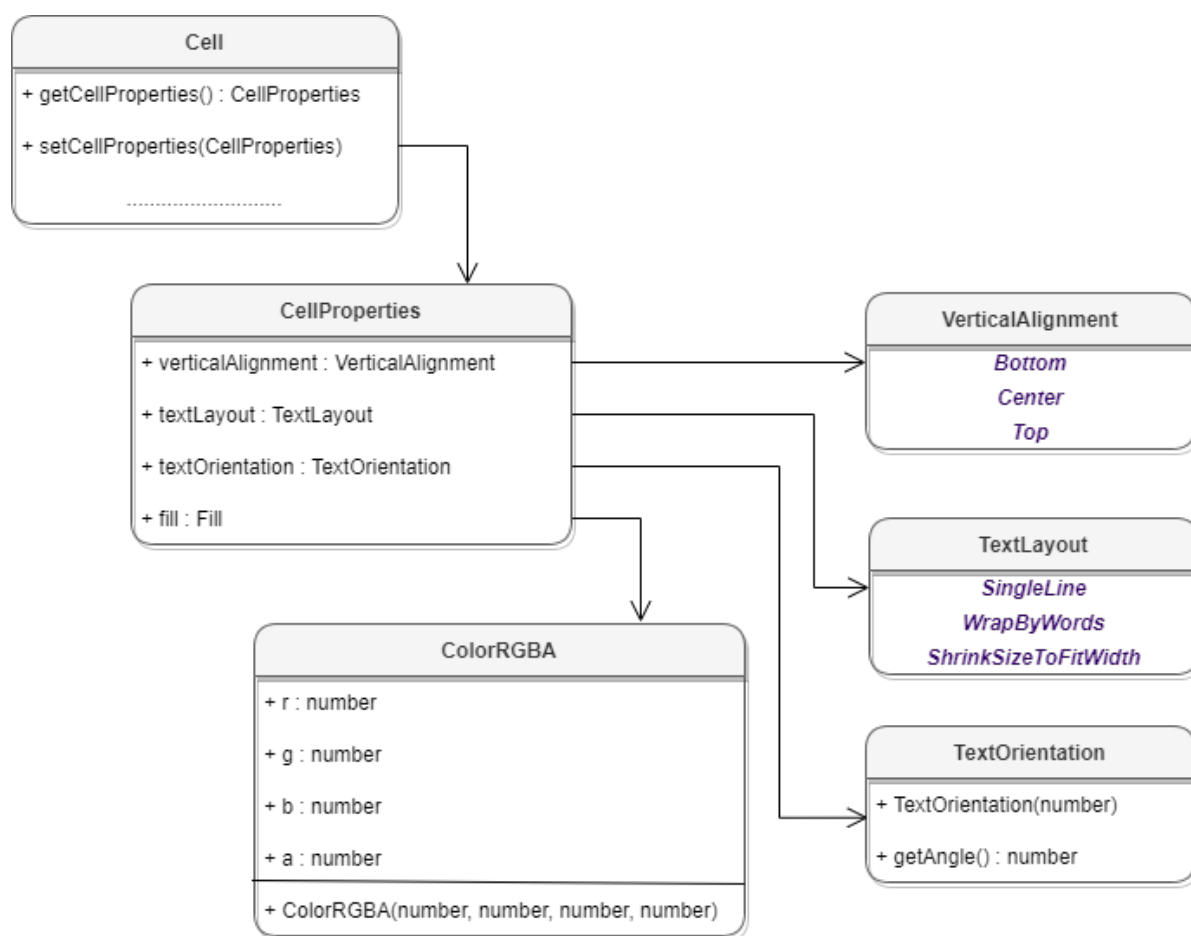


Рисунок 18 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 8 – Описание полей класса CellProperties

Поле	Тип	Значение
CellProperties.verticalAlignment	<a href="#">VerticalAlignment</a>	Вертикальное выравнивание в ячейке

Поле	Тип	Значение
CellProperties.textLayout	<a href="#">TextLayout</a>	Способ отображения значения ячейки
CellProperties.fill	<a href="#">Fill</a>	Заполнение фона ячейки
CellProperties.textOrientation	<a href="#">TextOrientation</a>	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)
```

### 6.12.1 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.backgroundColor = 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.backgroundColor = myOfficeSDK.ColorRGBA(255, 255, 0, 255)
secondCellProps.textOrientation = myOfficeSDK.TextOrientation(45)

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

### 6.12.2 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.backgroundColor = 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.backgroundColor = myOfficeSDK.ColorRGBA(255, 255, 0, 0)
secondCellProps.textOrientation = myOfficeSDK.TextOrientation(45)

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

## 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.row

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

### Пример:

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



## 6.13.2 Поле `CellPosition.column`

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

**Пример:**

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

Класс CellRange описывает диапазон ячеек таблицы.

### Пример:

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

### 6.14.1 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 [OutOfRangeException](#), 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))
```

### 6.14.2 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](#).

### 6.14.3 Метод `CellRange.copyInto`

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

Данный метод реализован только в табличных документах и позволяет работать только в рамках одного листа документа.

#### Пример (только для табличного документа):

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

	A	B	C	D	E	
1	1	2				
2	3	4				
3						
4						
5		1	2	1	2	
6		3	4	3	4	
7						
8						

Рисунок 19 – Копирование ячеек табличного документа

Дополнительный пример использования приведен в разделе [Копирование ячеек в табличном документе](#).

## 6.14.4 Метод `CellRange.moveTo`

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

Данный метод реализован только в табличных документах и позволяет работать только в рамках одного листа документа.

### Пример (только для табличного документа):

```
sourceRange = table.getCellRange("A1:A2");
destRange = table.getCellRange("A2:A3");
sourceRange.moveTo(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](#).

## 6.14.5 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())
```

## 6.14.6 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())
```

## 6.14.7 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())
```

### 6.14.8 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())
```

### 6.14.9 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())
```

### 6.14.10 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)
```

## 6.14.11 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)
```

## 6.14.12 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.backgroundColor.r)
```

## 6.14.13 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.backgroundColor = myOfficeSDK.ColorRGBA(55, 146, 179, 200)
cellRange.setCellProperties(cellProperties)
```

## 6.14.14 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()
```

## 6.14.15 Метод `CellRange.getTable`

Метод возвращает таблицу ([Table](#)) для диапазона ячеек.

**Пример:**

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

## 6.15 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 9.

Таблица 9 – Поля класса `CellRangePosition`

Поле	Тип	Описание
<code>topLeft</code>	<a href="#">CellPosition</a>	Позиция левой верхней ячейки таблицы прямоугольного диапазона. Значение 0,0 соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.
<code>bottomRight</code>	<a href="#">CellPosition</a>	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)
```

### 6.15.1 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())
```

### 6.15.2 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")
```

### 6.15.3 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")

```

## 6.16 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.

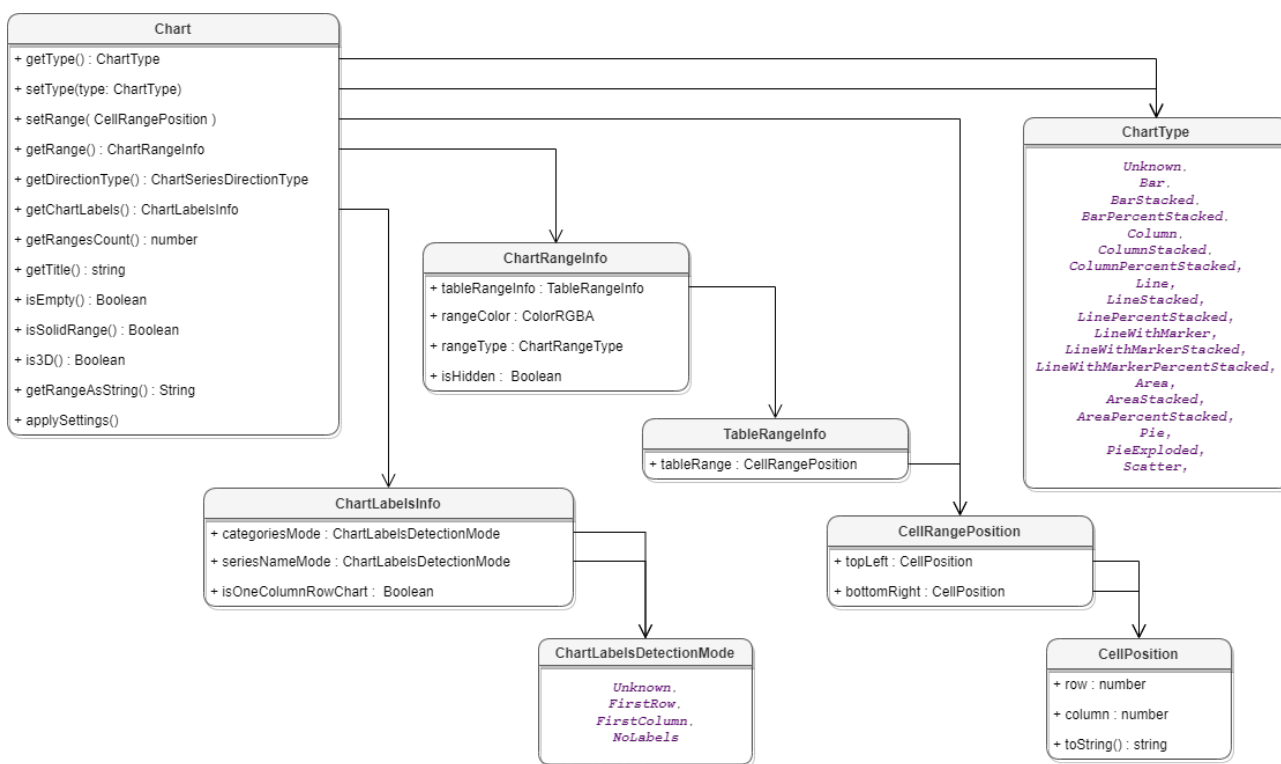


Рисунок 20 – Объектная модель класса Chart

### 6.16.1 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())
```

## 6.16.2 Метод Chart.getType

Метод возвращает тип диаграммы [ChartType](#).

**Пример:**

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

## 6.16.3 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())
```

## 6.16.4 Метод Chart.getRangesCount

Метод возвращает количество серий диаграммы.

**Пример:**

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

## 6.16.5 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)
```

## 6.16.6 Метод Chart.getTitle

Метод возвращает заголовок диаграммы.

**Пример:**

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

## 6.16.7 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())
```

## 6.16.8 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))
```

## 6.16.9 Метод Chart.isEmpty

Метод возвращает True, если диаграмма не содержит значений.

### Пример:

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

## 6.16.10 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())
```

## 6.16.11 Метод `Chart.is3D`

Метод возвращает `True`, если диаграмма трехмерная.

**Пример:**

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

## 6.16.12 Метод `Chart.getDirectionType`

Метод возвращает направление [ChartSeriesDirectionType](#) серий диаграммы.

**Пример:**

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

## 6.16.13 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)
```

## 6.16.14 Метод `Chart.getRangeAsString`

Метод возвращает диапазон ячеек диаграммы в формате строки.

**Пример:**

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

## 6.16.15 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)
```

## 6.17 Класс ChartLabelsDetectionMode

Тип описывает режимы автоматического определения меток диаграмм (см. таблицу 10).

Таблица 10 - 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)
```

## 6.18 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 11.

Таблица 11 – Описание полей класса `ChartLabelsInfo`

Поле	Описание	Тип
<code>categoriesMode</code>	Режим автоматического определения меток для категорий.	<a href="#">ChartLabelsDetectionMode</a>
<code>seriesNameMode</code>	Режим автоматического определения меток для серий.	<a href="#">ChartLabelsDetectionMode</a>
<code>isOneColumnRowChart</code>	Поле содержит <code>True</code> , если диапазон диаграммы содержит только одну строку или одну колонку.	<code>Boolean</code>

## 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)
```

## 6.19 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 12.

Таблица 12 – Описание полей класса ChartRangeInfo

Поле	Описание	Тип
tableRangeInfo	Исходный диапазон ячеек для серии.	<a href="#">TableRangeInfo</a>
rangeColor	Цвет для отрисовки серии.	<a href="#">ColorRGBA</a>
isHidden	Задаёт видимость серии диаграммы.	bool
rangeType	Тип диапазона диаграммы.	<a href="#">ChartRangeType</a>

### 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);
```

## 6.20 Класс ChartRangeType

Класс описывает тип диапазона исходных данных диаграммы (см. таблицу 13).

Таблица 13 - 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]);
```

## 6.21 Класс Charts

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

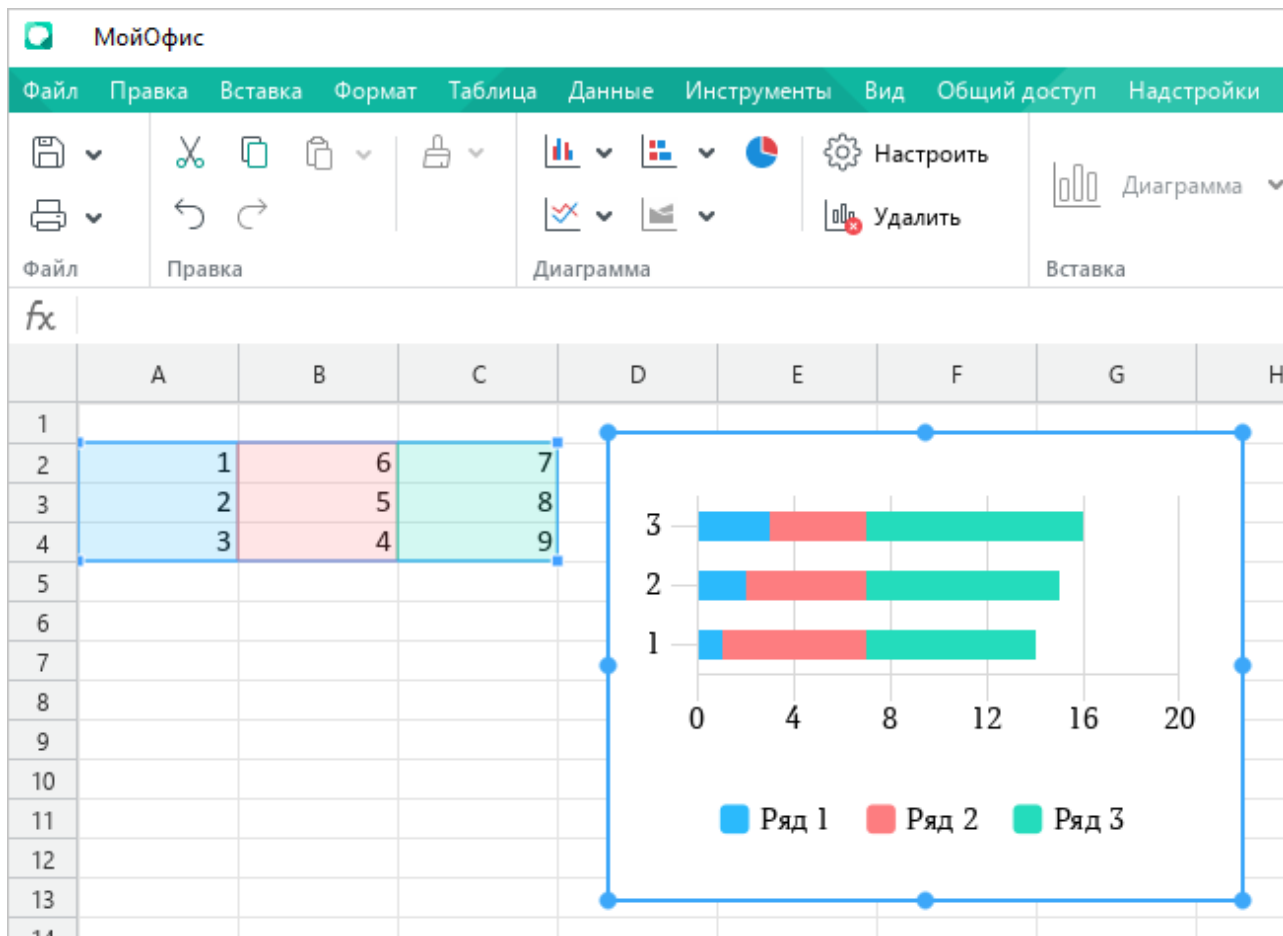


Рисунок 21 – Пример отображения диаграммы в МойОфис Таблица.

### 6.21.1 Метод Charts.getChartsCount

Метод возвращает общее количество диаграмм в табличном документе.

**Пример:**

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



## 6.21.2 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())
```

## 6.21.3 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)
```

## 6.22 Класс ChartSeriesDirectionType

Тип описывает направление серий диаграмм (см. таблицу 14).

Таблица 14 - 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])
```

## 6.23 Класс ChartType

Перечисление ChartType описывает все поддерживаемые типы диаграмм (см. таблицу 15)

Таблица 15 - 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())
```

## 6.24 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)
```

### 6.24.1 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)
```

### 6.24.2 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)
```

### 6.24.3 Метод Color.getTransforms

Метод возвращает текущую трансформацию цвета [ColorTransforms](#).

## 6.24.4 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)
```

## 6.24.5 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")
```

## 6.24.6 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")
```

## 6.25 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)
```

## 6.25.1 Метод ColorTransforms::apply

Метод устанавливает цвет трансформации [ColorRGBA](#).

**Пример:**

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

## 6.26 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 16.

Таблица 16 – 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)
```

## 6.26.1 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")
```

## 6.26.2 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")
```

## 6.27 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](#).

## 6.27.1 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())
```

## 6.27.2 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())
```

## 6.27.3 Метод `Comment.getText`

Метод возвращает текст комментария.

**Пример:**

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

## 6.27.4 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)
```

## 6.27.5 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())
```

## 6.27.6 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())
```

## 6.28 Класс `Comments`

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

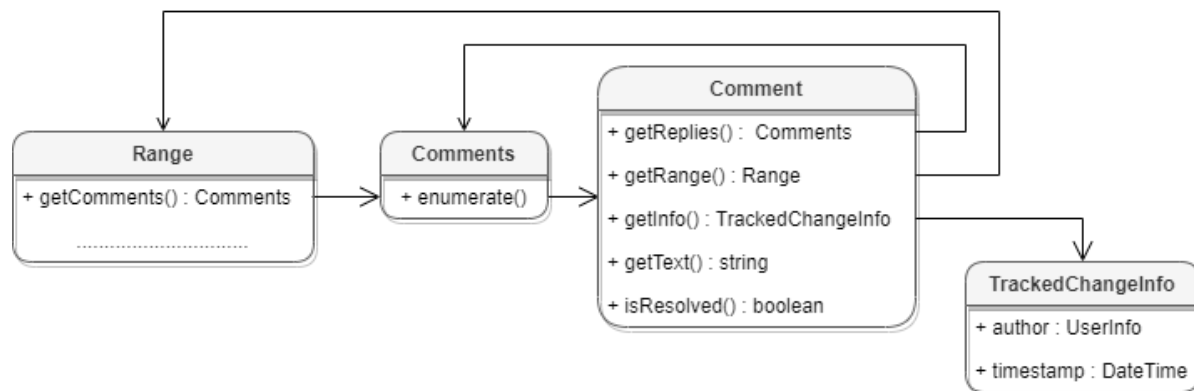


Рисунок 22 – Объектная модель классов для работы с комментариями

Для получения списка комментариев используется метод [Range.getComments\(\)](#).

**Пример:**

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



## 6.28.1 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)
```

## 6.29 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](#).

### 6.29.1 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);
```

### 6.29.2 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](#).

### 6.30 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()
```

## 6.31 Класс CurrencyCellFormatting

Класс содержит параметры для денежного формата ячеек таблицы. Описание полей класса CurrencyCellFormatting представлено в таблице 17.

Таблица 17 – 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
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 <a href="#">CurrencySignPlacement</a>

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())
```

## 6.32 Класс CurrencySignPlacement

Варианты размещения знака валюты представлены в таблице 18. Данный тип используется в поле `currencyFormat` класса [LocaleInfo](#), а также в поле `currencySignPlacement` класса [CurrencyCellFormatting](#) (см. пример в ее описании).

Таблица 18 – Описание полей класса CurrencySignPlacement

Поле	Описание	Пример
<code>CurrencySignPlacement_Prefix</code>	Размещение знака валюты до значения	\$12.00
<code>AccountingCellFormatting_Suffix</code>	Размещение знака валюты после значения	12,00 Р

## 6.33 Класс DatePatterns

Форматы даты представлены в таблице 19. Пример использования см. в главе [DateTimeCellFormatting](#).

Таблица 19 – Форматы даты

Наименование константы	Описание
<code>DatePatterns_DayMonthTextLongYearLong</code>	'mmm dd, yyyy' для языка en_US
<code>DatePatterns_FullDate</code>	'день недели, mmm dd, yyyy' для языка en_US
<code>DatePatterns_DayMonthNumberLongYearLong</code>	'mm/dd/yyyy' для языка en_US
<code>DatePatterns_DayMonthNumberLongYearShort</code>	'mm/dd/yy' для языка en_US
<code>DatePatterns_DayMonthNumberShortYearShort</code>	'm/dd/yy' для языка en_US
<code>DatePatterns_DayMonthTextShort</code>	'dd-mmm' для языка en_US
<code>DatePatterns_MonthTextShortYearShort</code>	'mmm-yy' для языка en_US
<code>DatePatterns_DayMonthTextShortYearShort</code>	'mmm dd, yy' для языка en_US

Наименование константы	Описание
DatePatterns_DayMonthYearLongNonLocalizableHyphen	Нелокализуемый шаблон 'dd-mm-yyuu'
DatePatterns_DayMonthYearLongNonLocalizableSlash	Нелокализуемый шаблон 'dd/mm/yyuu'

## 6.34 Класс DateTime

Класс DateTime предоставляет дату и время с точностью до секунды. Используется для поля TrackedChangeInfo.timeStamp. Описание полей класса DateTime представлено в таблице 20.

Таблица 20 – 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

### 6.34.1 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");
```

### 6.34.2 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");
```

## 6.35 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 21.

Таблица 21 – Описание полей класса DateTimeCellFormatting

Поле	Описание
DateTimeCellFormatting.dateListID	Формат даты <a href="#">DatePatterns</a>
DateTimeCellFormatting.timeListID	Формат времени <a href="#">TimePatterns</a>

### 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())
```

## 6.36 Класс DateTimeFormat

В таблице 22 представлены варианты масштабирования при печати табличных документов. Используется в качестве параметра метода [CellRange.insertCurrentDateTime\(\)](#).

Таблица 22 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
DateTimeFormat_DateTime	Дата/время

Наименование константы	Описание
DateTimeFormat_Date	Дата
DateTimeFormat_Time	Время

## 6.37 Класс Document

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

### Пример:

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

### 6.37.1 Метод Document.getAbsolutePath

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.getAbsolutePath())
```

#### 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.

### 6.37.2 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)
```

### 6.37.3 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](#).

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)
```

## 6.37.4 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.

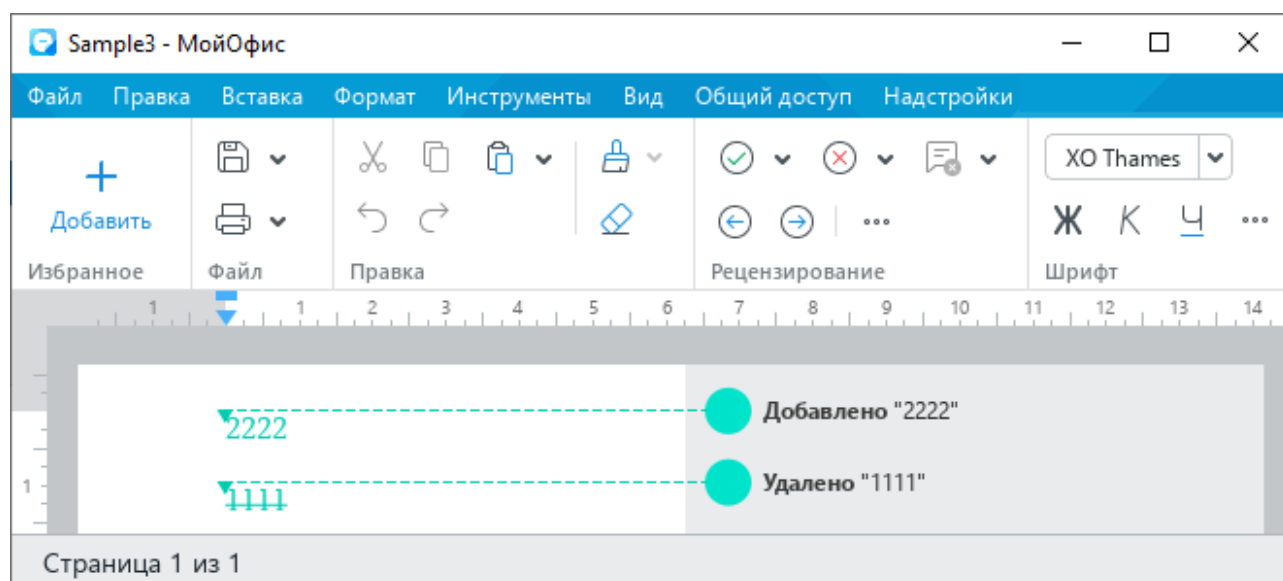


Рисунок 23 – Результат выполнения метода merge

## 6.37.5 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())
```

## 6.37.6 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())
```

## 6.37.7 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())
```

## 6.37.8 Метод Document.getRange

Метод предоставляет доступ ко всему диапазону [Range](#) документа.

**Пример:**

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

## 6.37.9 Метод `Document.isChangesTrackingEnabled`

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

### Пример:

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

## 6.37.10 Метод `Document.setChangesTrackingEnabled`

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

### Пример:

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

## 6.37.11 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())
```

## 6.37.12 Метод `Document.setPageProperties`

Метод устанавливает свойство [PageProperties](#) в документе.

### Пример:

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

## 6.37.13 Метод `Document.setFormulaType`

Метод устанавливает поддерживаемую адресацию ячеек [FormulaType](#) документа.

### Пример:

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

## 6.37.14 Метод `Document.getFormulaType`

Метод возвращает поддерживаемую адресацию ячеек [FormulaType](#) документа.

**Пример:**

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

## 6.37.15 Метод `Document.setPageOrientation`

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

**Пример:**

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

## 6.37.16 Метод `Document.getSectionsEnumerator`

Возвращает список объектов типа [Section](#).

**Пример:**

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

## 6.37.17 Метод `Document.getSections`

Возвращает объект типа [Sections](#).

**Пример:**

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

## 6.37.18 Метод `Document.setMirroredMarginsEnabled`

Метод позволяет включать и выключать зеркальные поля в документе.

**Пример:**

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

## 6.37.19 Метод `Document.areMirroredMarginsEnabled`

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

### Пример:

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

## 6.37.20 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()
```

## 6.37.21 Метод `Document.getNamedExpressions`

Используется для получения списка именованных диапазонов [NamedExpressions](#).

### Пример:

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

## 6.38 Класс `DocumentFormat`

В таблице 23 приведены поддерживаемые форматы документов, структура используется в [DocumentSettings](#).

Таблица 23 – Форматы документов

Наименование константы	Описание
<code>DocumentFormat_PlainText</code>	Используется для работы с файлами TXT.
<code>DocumentFormat_DSV</code>	Используется для работы с табличными данными в текстовой форме (CSV, DSV). Строка текста содержит одно или несколько полей данных, разделенных запятыми или иным разделителем.
<code>DocumentFormat_OXML</code>	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML.
<code>DocumentFormat_ODF</code>	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010).

Наименование константы	Описание
DocumentFormat_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.
DocumentFormat_PDF	Используется для работы с документами в формате Portable Document Format (PDF) версии 1.4.
DocumentFormat_PDFA	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

## 6.39 Класс DocumentSettings

Класс DocumentSettings предоставляет общие настройки документа и используется в [Document.createDocument\(\)](#).

Описание полей класса DocumentSettings представлено в таблице 24.

Таблица 24 – Описание полей класса DocumentSettings

Поле	Тип	Описание
DocumentSettings_documentType	<a href="#">DocumentType</a>	Тип документа
DocumentSettings_userInfo	<a href="#">UserInfo</a>	Информация о пользователе
DocumentSettings_localeInfo	<a href="#">LocaleInfo</a>	Информация о локализации
DocumentSettings_timeZone	<a href="#">TimeZone</a>	Информация о временной зоне
DocumentSettings_formulaType	<a href="#">FormulaType</a>	Система адресации ячеек

## 6.40 Класс DocumentType

В таблице 25 приведены поддерживаемые типы документов, используются при создании документа [Application.createDocument\(\)](#), [DocumentSettings](#).

Таблица 25 - 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. Работа с презентационными документами средствами Document API в настоящий момент не поддерживается.

## 6.41 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 26.

Таблица 26 – Описание полей класса `DSVSettings`

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

### Пример:

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

### 6.41.1 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")
```

## 6.41.2 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")
```

## 6.42 Класс Encoding

В таблице 27 приведены поддерживаемые кодировки документов. Используется в [LoadDocumentSettings](#).

Таблица 27 - 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



## 6.43 Класс ExportFormat

В таблице 28 приведены поддерживаемые форматы экспорта документов, см. [Document.exportAs\(\)](#).

Таблица 28 - Форматы экспорта документов

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

## 6.44 Класс Field

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

## 6.45 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")
```

### 6.45.1 Метод Fill.getColor

Метод возвращает цвет заполнения [Color](#).

## 6.45.2 Метод `Fill.getUrl`

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

## 6.45.3 Метод `Fill.isNoFill`

Метод возвращает `True`, если заполнения нет.

## 6.46 Класс `FiltersRange`

Класс `FiltersRange` реализует диапазон таблицы, позволяющий манипулировать фильтрами столбцов. Пример использования приведен в разделе [Работа с фильтрами](#).

### 6.46.1 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()
```

### 6.46.2 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()
```

### 6.46.3 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](#).

### 6.46.4 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](#).

### 6.47 Класс `FormulaType`

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 29. Используется в [Document.getFormulaType\(\)](#), [Document.setFormulaType\(\)](#), [DocumentSettings](#).

Таблица 29 – 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

## 6.48 Класс FractionCellFormatting

Класс содержит параметры для дробного формата ячеек таблицы. Используется в качестве аргумента метода [Cell.setFormat\(\)](#). Описание полей класса FractionCellFormatting представлено в таблице 30.

Таблица 30 – 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())

```

## 6.49 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).

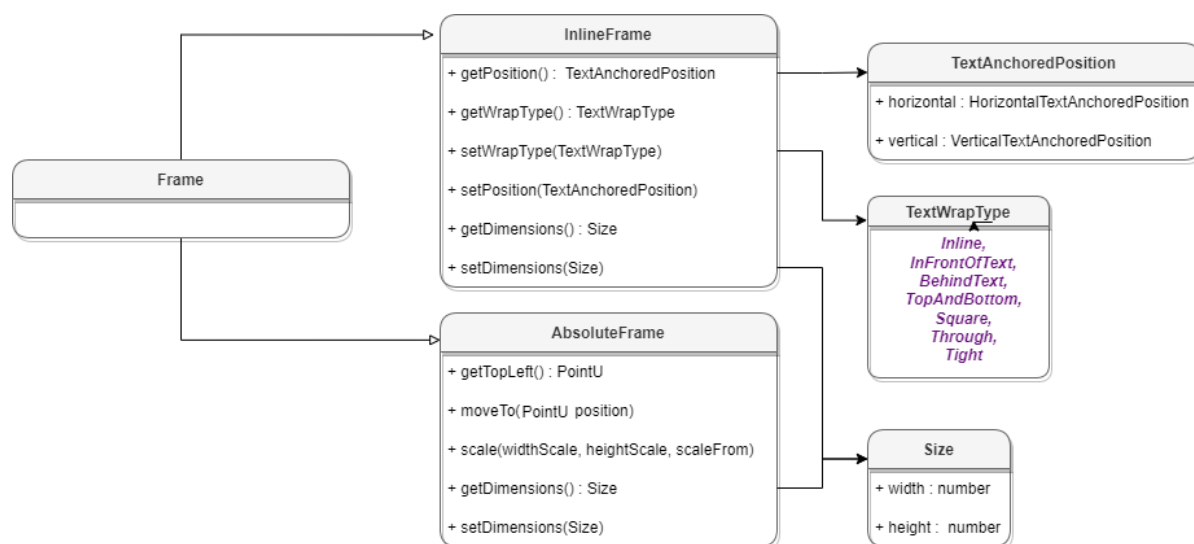


Рисунок 24 – Объектная модель класса `Frame`

### 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")
```

## 6.50 Класс `FrozenRangePosition`

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

## 6.50.1 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());
```

## 6.50.2 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())
```

## 6.50.3 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())
```

## 6.50.4 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())
```

## 6.50.5 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())
```

## 6.50.6 Метод `FrozenRangePosition.isRowsCols`

Возвращает `True` если диапазон содержит строки и колонки.

### Пример:

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

## 6.50.7 Метод `FrozenRangePosition.isArea`

Возвращает `True` если диапазон является непрерывной областью.

### Пример:

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

## 6.50.8 Метод `FrozenRangePosition.isRows`

Возвращает `True` если диапазон состоит из строк.

**Пример:**

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

## 6.50.9 Метод `FrozenRangePosition.isCols`

Возвращает `True` если диапазон состоит из колонок.

**Пример:**

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

## 6.51 Класс `HeaderFooter`

Класс `HeaderFooter` определяет колонтитул текстового документа.

### 6.51.1 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)
```

### 6.51.2 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())
```



## 6.51.3 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())
```

## 6.52 Класс `HeaderFooterType`

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

Поддерживаемые типы колонтитулов страниц документов представлены в таблице 31.

Таблица 31 - Типы колонтитулов страниц документа

Имя константы типа колонтитула	Наименование типа колонтитула
<code>HeaderFooterType_Header</code>	Верхний
<code>HeaderFooterType_Footer</code>	Нижний

### Пример:

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

## 6.53 Класс `HeadersFooters`

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

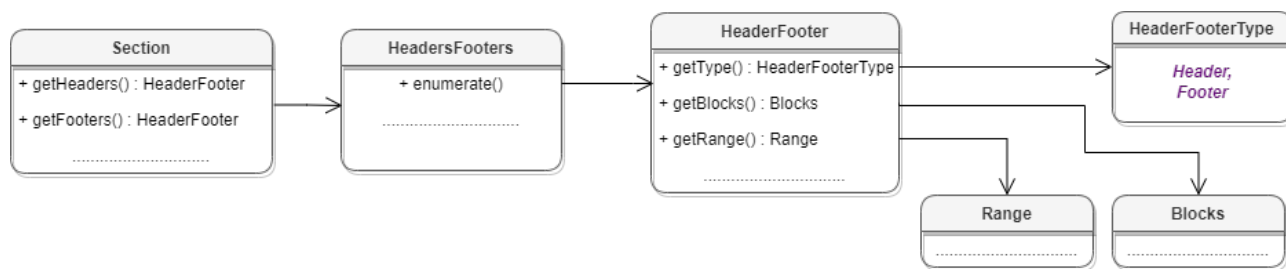


Рисунок 25 – Классы для работы с колонтитулами

### 6.53.1 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())
    
```

### 6.54 Класс HorizontalAnchorAlignment

В таблице 32 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали.

Таблица 32 – 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

## 6.55 Класс `HorizontalRelativeTo`

В таблице 33 представлены типы размещения объекта относительно закрепленной позиции по горизонтали.

Таблица 33 – Типы размещения объекта относительно закрепленной позиции по горизонтали

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

## 6.56 Класс `HorizontalTextAnchoredPosition`

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

Описание полей класса `HorizontalTextAnchoredPosition` представлено в таблице 34.

Таблица 34 – Описание полей класса `HorizontalTextAnchoredPosition`

Поле	Описание
<code>HorizontalTextAnchoredPosition.relativeTo</code>	Тип размещения объекта относительно закрепленной позиции по горизонтали <a href="#">HorizontalRelativeTo</a>
<code>HorizontalTextAnchoredPosition.offset</code>	Смещение объекта.

Поле	Описание
HorizontalTextAnchoredPosition.alignment	Тип выравнивания объекта относительно закрепленной позиции по горизонтали <a href="#">HorizontalAnchorAlignment</a>

## 6.56.1 HorizontalTextAnchoredPosition.\_\_eq\_\_

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

### Пример:

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

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

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

## 6.56.2 HorizontalTextAnchoredPosition.\_\_ne\_\_

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

### Пример:

```
firstHorizontalTextAnchoredPosition =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
firstHorizontalTextAnchoredPosition.alignment =
```

```
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")
```

## 6.57 Класс Hyperlink

Класс Hyperlink описывает свойства ссылки. Объект Hyperlink может быть получен посредством вызова метода [Cell.getHyperlink\(\)](#).

Таблица 35 – 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)
```

### 6.57.1 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")
```

## 6.57.2 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")
```

## 6.58 Класс Image

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

### 6.58.1 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())
```

### 6.58.2 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
```

## 6.59 Класс Images

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

### 6.59.1 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())
```

## 6.60 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.

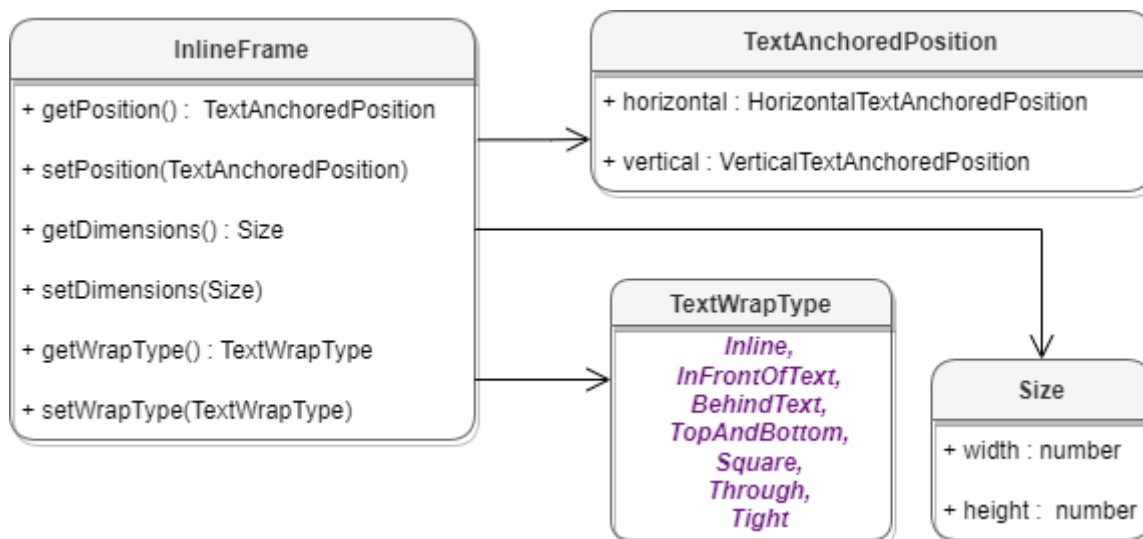


Рисунок 26 – Объектная модель класса `InlineFrame`

### 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")
```

#### 6.60.1 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)
```

## 6.60.2 Метод `InlineFrame.getPosition`

Метод возвращает позицию встроенного объекта на странице типа [TextAnchoredPosition](#).

### Пример:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    frame = mediaObject.getFrame()
    anchoredPosition = frame.getPosition()
    textAnchoredPosition = anchoredPosition.textPosition
    print("horz:", textAnchoredPosition.horizontal, ", vert:",
textAnchoredPosition.vertical)
```

## 6.60.3 Метод `InlineFrame.setDimensions`

Метод задает размер `SizeU` встроенного объекта.

### Пример:

```
mediaObjects = document.getRange().getInlineObjects()
mediaObjectsEnumerator = mediaObjects.getEnumerator()
for mediaObject in mediaObjectsEnumerator:
    frame = mediaObject.getFrame()
    if (isinstance(frame, myOfficeSDK.InlineFrame)):
        frame.setDimensions(myOfficeSDK.SizeU(50, 50))
```

## 6.60.4 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())
```

## 6.60.5 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())
```

## 6.60.6 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)
```

## 6.61 Класс Insets

Класс `Insets` предназначен для задания полей, например, страницы. Поля класса `Insets` представлены в таблице 36. Используется в поле `margins` класса [PageProperties](#).

Таблица 36 – Описание полей класса `Insets`

Поле	Тип	Описание
<code>left</code>	<code>Number</code>	Левая граница поля
<code>top</code>	<code>Number</code>	Верхняя граница поля
<code>right</code>	<code>Number</code>	Правая граница поля
<code>bottom</code>	<code>Number</code>	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)
```

### 6.61.1 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")
```

## 6.61.2 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")
```







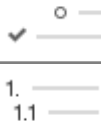
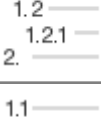
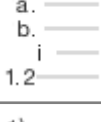
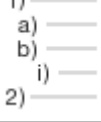
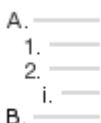
## 6.62 Класс ListSchema

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 37.

Таблица 37 - Типы схем абзацев

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
<code>ListSchema_Unknown</code>	Неизвестно.	
<code>ListSchema_UnknownBullet</code>	Список без маркера.	Соответствует варианту «нет»
<code>ListSchema_UnknownNumbering</code>	Нумерация без номера.	Соответствует варианту «нет»
<code>ListSchema_BulletCircleSolid</code>	Список с маркерами в виде круга.	

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
ListSchema_Unknown	Неизвестно.	
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.	
ListSchema_EnumeratorDecimalBracket	Десятичная нумерация со скобкой.	
ListSchema_EnumeratorLatinUpperCaseDot	Нумерация латинскими прописными буквами с точкой.	
ListSchema_EnumeratorLatinLowerCaseDot	Нумерация латинскими строчными буквами с точкой.	

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
ListSchema_Unknown	Неизвестно.	
ListSchema_EnumeratorLatinLowercaseBracket	Нумерация латинскими строчными буквами со скобкой.	a) _____ 1) _____ 2) _____ i) _____ b) _____
ListSchema_EnumeratorRomanUppercaseDot	Нумерация римскими прописными цифрами с точкой.	I. _____ a. _____ b. _____ 1. _____ II. _____
ListSchema_EnumeratorRomanLowercaseDot	Нумерация римскими строчными цифрами с точкой.	i. _____ 1. _____ 2. _____ i. _____ ii. _____
ListSchema_EnumeratorDecimalRussianBracket	Десятичная нумерация через запятую со скобкой.	1) _____ a) _____ б) _____ i) _____ 2) _____
ListSchema_EnumeratorRussianLowercaseBracket	Нумерация с русскими строчными буквами со скобкой.	a) _____ 1) _____ 2) _____ i) _____ б) _____

## Пример:

```
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraph.setListSchema(myOfficeSDK.ListSchema_BulletCheckmark)
```

## 6.63 Класс LineEndingProperties

Класс LineEndingProperties содержит варианты оформления окончаний линий. Описание полей класса LineEndingProperties представлено в таблице 38. Используется в полях headLineEndingProperties и tailLineEndingProperties класса [LineProperties](#).

Таблица 38 – Описание полей класса LineEndingProperties

Поле	Тип	Описание
LineEndingProperties.style	<a href="#">LineEndingStyle</a>	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)
```

### 6.63.1 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")
```

## 6.63.2 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")
```

## 6.64 Класс LineEndingStyle

В таблице 39 приведены типы окончания линии. Используется в поле `style` класса [LineEndingProperties](#).

Таблица 39 – Типы окончания линии

Наименование константы	Описание
<code>LineEndingStyle_Arrow</code>	
<code>LineEndingStyle_Diamond</code>	
<code>LineEndingStyle_Oval</code>	
<code>LineEndingStyle_Stealth</code>	

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)
    
```

## 6.65 Класс LineProperties

Класс `LineProperties` предназначен для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 27).

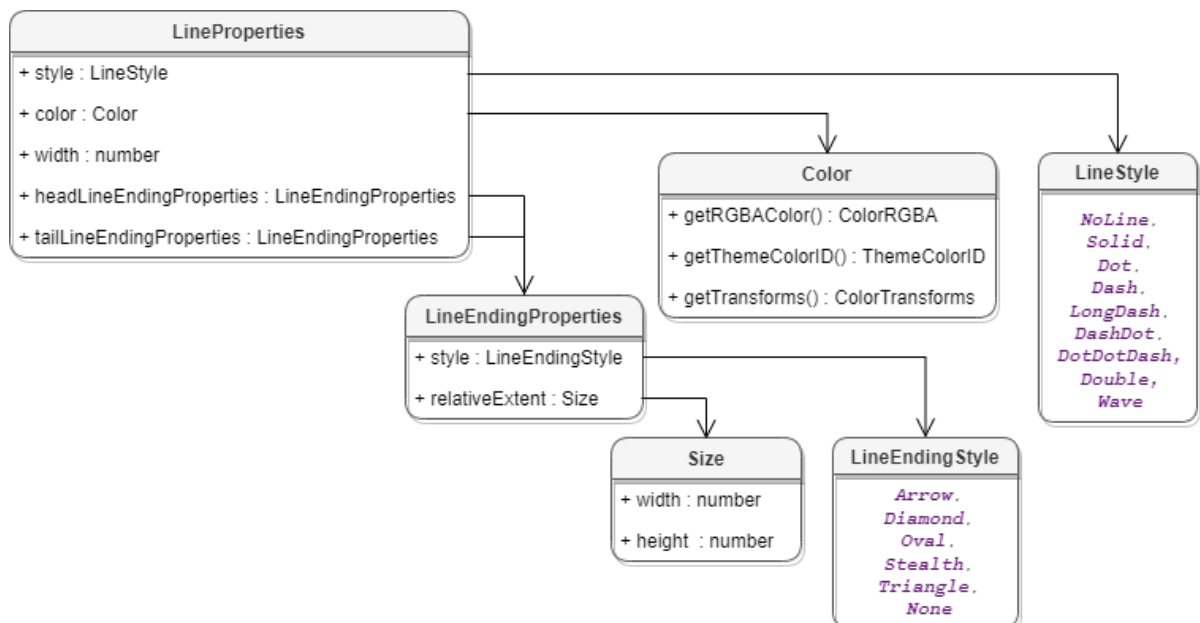


Рисунок 27 – Свойства границ ячеек

## 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)
```

### 6.65.1 Поле `LineProperties.style`

Поле предназначено для установки типа линии. Допустимые значения представлены в разделе [LineStyle](#).

### 6.65.2 Поле `LineProperties.width`

Поле предназначено для установки ширины линии. Тип - числовой.

### 6.65.3 Поле `LineProperties.color`

Поле предназначено для установки цвета линии. Тип - [Color](#).

### 6.65.4 Поле `LineProperties.headLineEndingProperties`

Поле предназначено для оформления начала линии [LineEndingProperties](#).

### 6.65.5 Поле `LineProperties.tailLineEndingProperties`

Поле предназначено для оформления конца линии [LineEndingProperties](#).

### 6.65.6 `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")
```

## 6.65.7 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")
```

## 6.66 Класс LineSpacing

Класс `LineSpacing` задает межстрочный интервал абзаца. Поля класса приведены в таблице 40. Для управления значением межстрочного интервала используются значения, представленные в разделе [LineSpacingRule](#).

Таблица 40 – Параметры межстрочного интервала

Поле	Описание
<code>LineSpacing.value</code>	Значение межстрочного интервала.

Поле	Описание
LineSpacing.rule	Правило формирования межстрочного интервала <a href="#">LineSpacingRule</a> .

### Пример:

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    paragraphProperties = paragraph.getParagraphProperties()
    paragraphProperties.lineSpacing = myOfficeSDK.LineSpacing(5.0,
myOfficeSDK.LineSpacingRule_Multiple);
```

#### 6.66.1 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")
```

#### 6.66.2 LineSpacing.\_\_ne\_\_

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

### Пример:

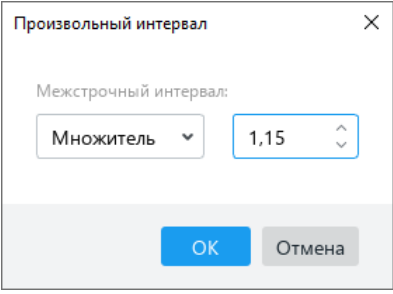
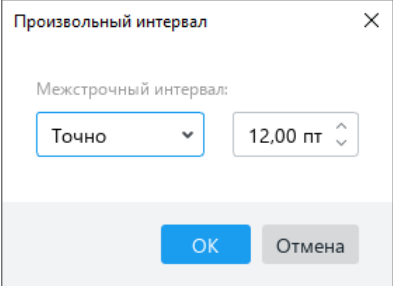
```
lineSpacingExact = myOfficeSDK.LineSpacing(10,
myOfficeSDK.LineSpacingRule_Exact)
lineSpacingMultiple = myOfficeSDK.LineSpacing(10,
myOfficeSDK.LineSpacingRule_Multiple)
if lineSpacingExact.__ne__(lineSpacingMultiple):
    print("Ne")
```

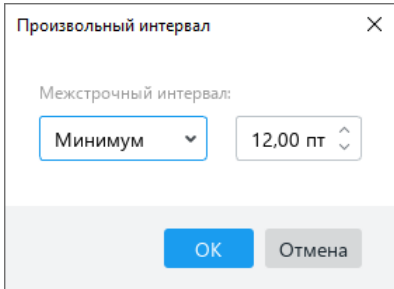
## 6.67 Класс LineSpacingRule

Класс LineSpacingRule содержит типы межстрочного интервала.

В таблице 41 представлены описания правил формирования межстрочного интервала текстового абзаца.

Таблица 41 – Виды межстрочного интервала

Наименование константы	Описание
LineSpacingRule_Multiple	<p>Установка произвольного межстрочного интервала с использованием множителя.</p> <p>При вызове необходимо указать значение множителя, например:</p> <pre>pPr.lineSpacing = myOfficeSDK.LineSpacing(1.15, myOfficeSDK.LineSpacingRule_Multiple)</pre> <p>В данном примере используется значение множителя <b>1.15</b>.</p> <p>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «<b>Произвольный интервал</b>» (см. документ «МойОфис Текст. Руководство пользователя»).</p> 
LineSpacingRule_Exact	<p>Установка произвольного межстрочного интервала с использованием точного значения.</p> <p>При вызове необходимо указать точное значение, например:</p> <pre>pPr.lineSpacing = myOfficeSDK.LineSpacing(12.0, myOfficeSDK.LineSpacingRule_Exact)</pre> <p>В данном примере используется точное значение <b>12.0</b>.</p> <p>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «<b>Произвольный интервал</b>» (см. документ «МойОфис Текст. Руководство пользователя»).</p> 

Наименование константы	Описание
LineSpacingRule_AtLeast	<p>Установка произвольного межстрочного интервала с использованием минимального значения.</p> <p>При вызове необходимо указать минимальное значение, например:</p> <pre>pPr.lineSpacing = myOfficeSDK.LineSpacing(12.0, myOfficeSDK.LineSpacingRule_AtLeast)</pre> <p>В данном примере используется минимальное значение <b>12.0</b>.</p> <p>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «<b>Произвольный интервал</b>» (см. документ «МойОфис Текст. Руководство пользователя»).</p> 


### 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)
```

### 6.68 Класс LineStyle

В таблице 42 приведены типы линий. Используется в поле `style` класса [LineProperties](#).

Таблица 42 – Типы линий

Наименование константы	Описание
LineStyle_NoLine	Нет линии
LineStyle_Solid	

Наименование константы	Описание
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;

```

## 6.69 Класс LoadDocumentSettings

Класс `LoadDocumentSettings` предоставляет дополнительные настройки, необходимые для загрузки документов из файла, см. [Application.loadDocument\(\)](#).

Описание полей класса `LoadDocumentSettings` представлено в таблице 43.

Таблица 43 – Описание полей класса `LoadDocumentSettings`

Поле	Описание
<code>LoadDocumentSettings_commonDocumentSettings</code>	Экземпляр таблицы, общие настройки документа <a href="#">DocumentSettings</a>
<code>LoadDocumentSettings_encoding</code>	Кодировка документа <a href="#">Encoding</a>



Поле	Описание
LoadDocumentSettings_dsvSettings	Экземпляр класса <a href="#">DSVSettings</a> , 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

## 6.70 Класс LocaleInfo

Класс `LocaleInfo` предоставляет информацию о локализации. Используется в поле `localeInfo` класса [DocumentSettings](#).

Описание полей `LocaleInfo` представлено в таблице 44.

Таблица 44 – Описание полей класса `LocaleInfo`

Поле	Описание
<code>LocaleInfo.localeName</code>	Название локализации, представлено в формате <b>&lt;language&gt; &lt;REGION&gt;</b> , где языковой код соответствует стандарту ISO-639, а код региона стандарту ISO-3166.
<code>LocaleInfo.decimalSeparator</code>	Десятичный разделитель, отделяет целые и дробные части чисел.
<code>LocaleInfo.thousandSeparator</code>	Символ, разделяющий группы цифр в числовых значениях.
<code>LocaleInfo.listSeparator</code>	Символ, отделяющий элементы в списке.
<code>LocaleInfo.currencySymbol</code>	Символ валюты, используемой в текущей стране или регионе.
<code>LocaleInfo.currencyFormat</code>	Расположение знака валюты в текущем регионе, тип: <a href="#">CurrencySignPlacement</a> .
<code>LocaleInfo.shortDatePattern</code>	Заданный «короткий» формат отображения даты в текущем регионе (например, 'm/d/yy' для en_US).
<code>LocaleInfo.longDatePattern</code>	Заданный «длинный» формат отображения даты в текущем регионе (например, 'dddd, mmmm d, yyyy' for en_US).
<code>LocaleInfo.timePattern</code>	Заданный формат отображения времени в текущем регионе (например, 'h:mm AM/PM' для en_US).

## 6.71 Класс `MediaObject`

Класс `MediaObject` представляет собой встроенный объект документа.

### 6.71.1 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")
```

### 6.71.2 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")
```

### 6.71.3 Метод `MediaObject.getFrame`

Метод возвращает свойства позиции встроенного объекта [Frame](#).

**Пример:**

```
mediaObjects = document.getRange().getInlineObjects()
for mediaObject in mediaObjects:
    print(mediaObject.getFrame())
```

## 6.72 Класс MediaObjects

Класс `MediaObjects` предназначен для доступа к коллекции графических объектов. Объект может быть получен вызовом методов [Range.getInlineObjects\(\)](#), [Table.getMediaObjects\(\)](#) (см. Рисунок 28).

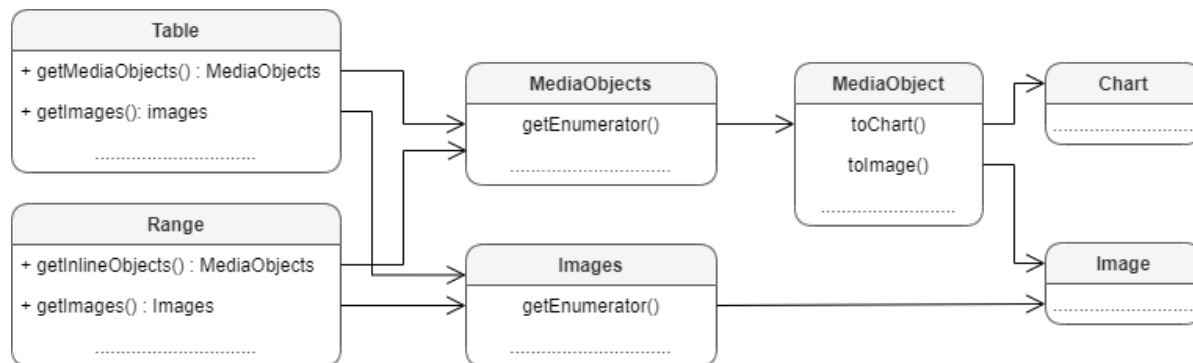


Рисунок 28 – Графические объекты

### 6.72.1 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")
```

## 6.73 Класс Message

Класс `Message` предназначен для формирования событий лога.

## 6.73.1 Класс `Message.Severity`

Класс `Message.Severity` (Таблица 45) описывает уровни сообщений лога (информация, предупреждение, ошибка).

Таблица 45 – Описание уровней лога `Message.Severity`

Поле	Описание
<code>Message.Severity_Info</code>	Информация
<code>Message.Severity_Warning</code>	Предупреждение
<code>Message.Severity_Error</code>	Ошибка

## 6.73.2 Метод `Message.getSeverity`

Метод возвращает уровень лога [Message.Severity](#).

## 6.73.3 Метод `Message.getText`

Метод возвращает текст сообщения.

## 6.73.4 Метод `Message.makeInfo`

Метод создает сообщение типа [Message.Severity\\_Info](#) с заданным текстом.

## 6.73.5 Метод `Message.makeWarning`

Метод создает сообщение типа [Message.Severity\\_Warning](#) с заданным текстом.

## 6.73.6 Метод `Message.makeError`

Метод создает сообщение типа [Message.Severity\\_Error](#) с заданным текстом.

## 6.74 Класс `Messenger`

### 6.74.1 Метод `Messenger.subscribe`

Метод служит для подписки на события лога.

#### Пример:

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

## 6.74.2 Метод `Messenger.notify`

Метод используется для создания события лога

### Пример:

```
messageHandler = myOfficeSDK.MessageHandler()  
messenger= application.getMessenger()  
messenger.notify(Message.makeWarning("Warning"))
```

## 6.75 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())
```

### 6.75.1 Метод `NamedExpression.getName`

Возвращает имя именованного диапазона. Пример см. в [NamedExpression](#).

### 6.75.2 Метод `NamedExpression.getExpression`

Возвращает текст выражения (формулы). Пример см. в [NamedExpression](#).

### 6.75.3 Метод `NamedExpression.getCellRange`

Возвращает диапазон ячеек [CellRange](#). Пример см. в [NamedExpression](#).

## 6.76 Класс `NamedExpressions`

Класс для представления списка именованных диапазонов. Может быть получена с помощью методов [Document.getNamedExpressions\(\)](#), [Table.getNamedExpressions\(\)](#).

## 6.76.1 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())
```

## 6.76.2 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())
```

## 6.76.3 Метод `NamedExpression.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())
```

## 6.76.4 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)
```

## 6.77 Класс NamedExpressionsValidationResult

Класс `NamedExpressionsValidationResult` описывает результат операций [NamedExpressions.addExpression\(\)](#), [NamedExpressions.removeExpression\(\)](#).

Поля класса описаны в таблице 46.

Таблица 46 - Поля класса `NamedExpressionsValidationResult`

Имя константы	Описание
<code>NamedExpressionsValidationResult_Series</code>	Операция выполнена успешно
<code>NamedExpressionsValidationResult_WrongName</code>	Неправильный формат имени
<code>NamedExpressionsValidationResultIsUsedInFormula</code>	Имя уже используется в формуле

## 6.78 Класс NumberCellFormatting

Класс содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода [Cell.setFormat\(\)](#). Описание полей класса `NumberCellFormatting` представлено в таблице 47.

Таблица 47 – Описание полей класса `NumberCellFormatting`

Поле	Описание
<code>decimalPlaces</code>	Количество десятичных позиций
<code>useThousandsSeparator</code>	Использовать разделитель для тысячных
<code>useRedForNegative</code>	Использовать красный цвет для отрицательных значений
<code>useBracketsForNagative</code>	Использовать скобки для отрицательных значений
<code>hideSign</code>	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())
```

## 6.79 Класс PageFieldOrder

Класс `PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем класса [PivotTableLayoutSettings](#). Описание полей класса представлено в таблице 48.

Таблица 48 – Описание полей класса `PageFieldOrder`

Поле	Описание
<code>PageFieldOrder_DownThenOver</code>	Вниз, затем поперек
<code>PageFieldOrder_OverThenDown</code>	Поперек, затем вниз

## 6.80 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)
```

## 6.80.1 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))
```

## 6.80.2 Метод PageNumbers.getLast

Метод `PageNumbers.getLast` возвращает последний номер страницы.

### Пример:

```
pageNumbers = myOfficeSDK.PageNumbers(1, 20)
print(pageNumbers.getLast())
```

## 6.80.3 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")
```

## 6.80.4 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")
```

## 6.81 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 49.

Таблица 49 - 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());
```

## 6.82 Класс PageParity

Варианты выбора страниц для экспорта и печати представлены в таблице 50. Используется в [PageNumbers](#), PrintSettings.

Таблица 50 – Варианты выбора страниц для экспорта и печати

Наименование константы	Описание
PageParity_Odd	Только нечетные страницы
PageParity_Even	Только четные страницы
PageParity_All	Все страницы

## 6.83 Класс PageProperties

Класс PageProperties предоставляет такие свойства страницы как высота, ширина, размеры полей. Размеры страницы представлены в пунктах (pt). Например, для листа формата А4 значение ширины составляет 595,29 pt, высоты – 841,90 pt (без округления). Описание полей приведено в таблице 51. Используется в [Document.setPageProperties\(\)](#), [Section.getPageProperties\(\)](#), [Section.setPageProperties\(\)](#).

Таблица 51 – Описание полей класса PageProperties

Поле	Описание
height	Высота страницы
width	Ширина страницы
margins	Поля страницы, тип - <a href="#">Insets</a>

### 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)
```

### 6.83.1 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")
```

## 6.83.2 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")
```

## 6.84 Класс Paragraphs

Класс `Paragraphs` предоставляет доступ к коллекции абзацев типа [Paragraph](#) (см. Рисунок 29). Коллекция абзацев может быть получена из объекта [Range](#) посредством использования метода [Range.getParagraphs\(\)](#).

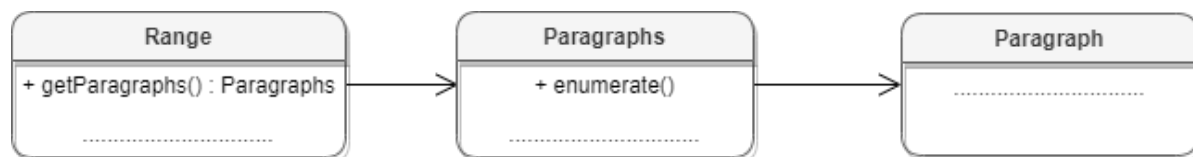


Рисунок 29 – Объектная модель для работы со списком абзацев

### 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()
```

## 6.84.1 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);
```

### 6.84.2 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)
```

### 6.84.3 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()
```

### 6.84.4 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()
```

## 6.84.5 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())
```

## 6.85 Класс Paragraph

Класс Paragraph предоставляет доступ к свойствам абзаца (см. Рисунок 30).

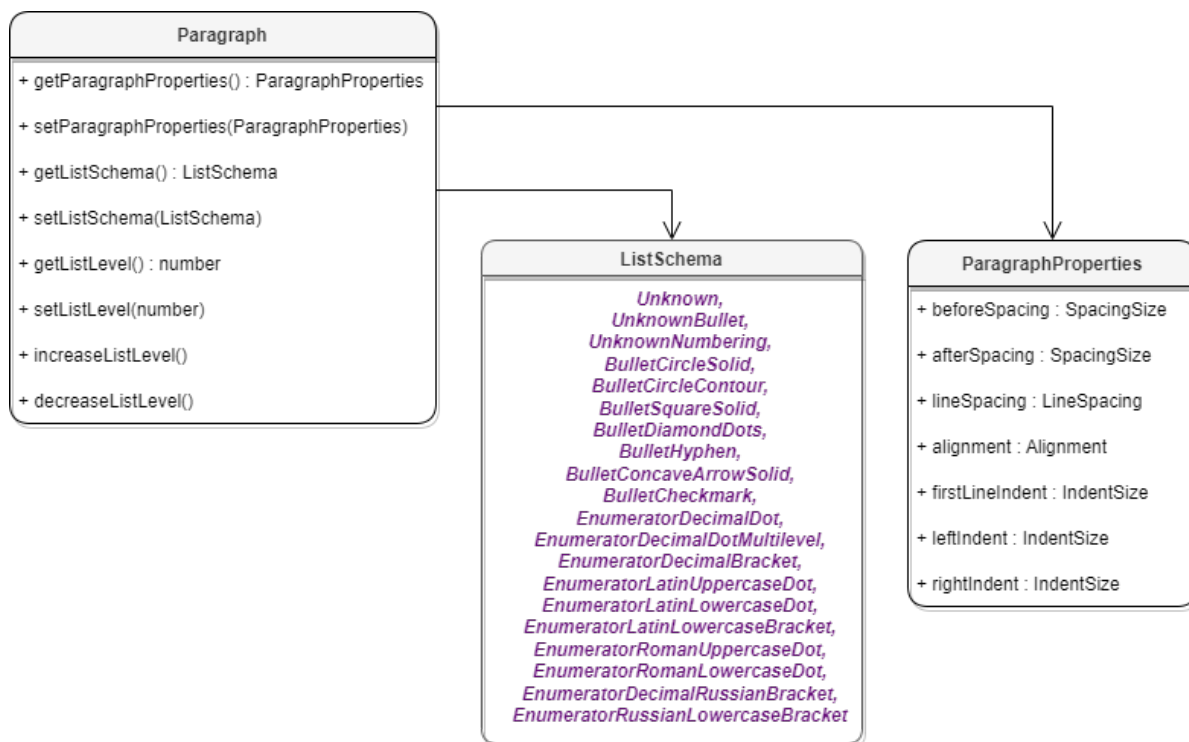


Рисунок 30 – Объектная модель классов для работы со свойствами параграфа

### 6.85.1 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)
```

## 6.85.2 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)
```

## 6.85.3 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())
```

## 6.85.4 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)
```

## 6.85.5 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())
```

## 6.85.6 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)
```

### 6.85.7 Метод Paragraph.increaseListLevel

Метод позволяет увеличить на единицу глубину вложенности элемента списка. В случае, если максимальный уровень уже установлен, увеличения не происходит. Данный метод используется только в текстовом документе. 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)
```

### 6.85.8 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)
```

## 6.86 Класс ParagraphProperties

Класс `ParagraphProperties` предназначен для управления свойствами форматирования (см. Рисунок 31). Класс `ParagraphProperties` используется в методах [Paragraph.getParagraphProperties](#) и [Paragraph.setParagraphProperties](#).

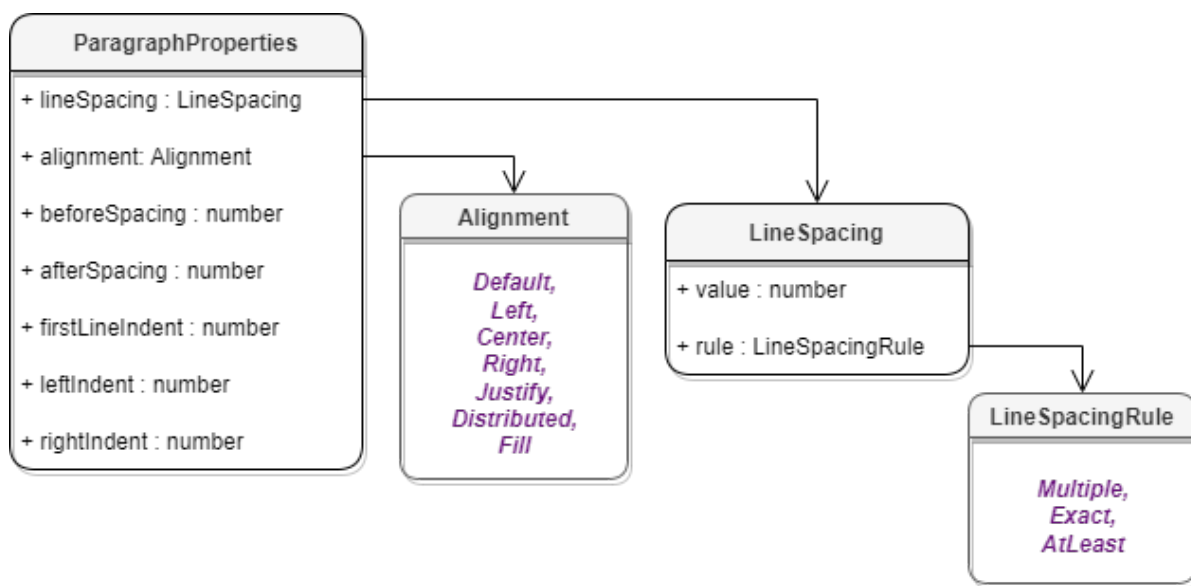
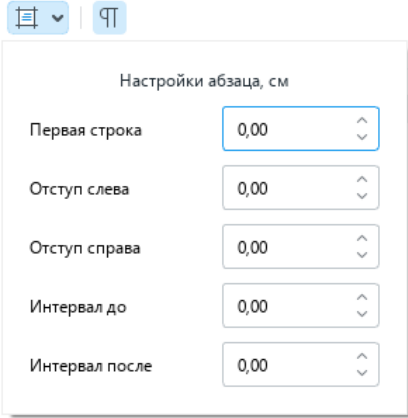


Рисунок 31 – Объектная модель классов для работы со свойствами параграфа

Описание полей класса [ParagraphProperties](#) представлено в таблице 52.

Таблица 52 – Описание полей класса `ParagraphProperties`

Поле	Описание
<code>ParagraphProperties_beforeSpacing</code>	Установка величины расстояния до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне <b>Настройки абзаца</b> , (см. рисунок выше), в поле <b>Интервал до</b> (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).

Поле	Описание
	
ParagraphProperties_afterSpacing	<p>Установка величины расстояния после абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне <b>Настройки абзаца</b>, в поле <b>Интервал после</b> (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).</p>
ParagraphProperties_lineSpacing	<p>Расстояние между строк одного абзаца (межстрочный интервал), <a href="#">LineSpacing</a>.</p>
ParagraphProperties_alignment	<p>Выравнивание текстового фрагмента по горизонтали. Список допустимых значений находится в разделе <a href="#">Alignment</a>.</p>
ParagraphProperties_firstLineIndent	<p>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 <b>Настройки абзаца</b>, (см. рисунок выше), в поле <b>Первая строка</b> (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).</p>
ParagraphProperties_leftIndent	<p>Расстояние от левого поля документа до абзаца (отступ слева). При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне <b>Настройки абзаца</b>, (см. рисунок выше), в поле <b>Отступ слева</b> (подробнее см. в документе «МойОфис Текст. Руководство пользователя»).</p>
ParagraphProperties_rightIndent	<p>Расстояние от правого поля документа до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне <b>Настройки абзаца</b>, (см. рисунок выше), в поле <b>Отступ справа</b> (подробнее см. в</p>

Поле	Описание
	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)
```

### 6.86.1 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")
```

## 6.86.2 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")
```

## 6.87 Класс PercentageCellFormatting

Содержит параметр для процентного формата ячеек таблицы, используется в качестве аргумента метода [Cell.setFormat\(\)](#). Описание полей класса PercentageCellFormatting представлено в таблице 53.

Таблица 53 – 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())
```

## 6.88 Класс PivotTablesManager

Класс [PivotTablesManager](#) используется для создания сводных таблиц, содержит метод `create()`. Может быть получена вызовом [Document.getPivotTablesManager\(\)](#).

### Пример:

```
pivotTablesManager = document.getPivotTablesManager()
```

### 6.88.1 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)
```

## 6.89 Класс PivotTable

Класс для представления сводной таблицы. Может быть получен из ячейки [Cell.getPivotTable\(\)](#), либо при создании новой сводной таблицы [PivotTablesManager.create\(\)](#).

### 6.89.1 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()
```

### 6.89.2 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())
```

### 6.89.3 Метод PivotTable.getSourceRange

Метод возвращает диапазон [CellRange](#) исходных данных сводной таблицы.

**Пример:**

```
sourceRange = pivotTable.getSourceRange()  
print(sourceRange.getBeginColumn())
```

## 6.89.4 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())
```

## 6.89.5 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())
```

## 6.89.6 Метод `PivotTable.isRowGrandTotalEnabled`

Метод возвращает `True`, если разрешено показывать общие итоги для строк.

**Пример:**

```
print(pivotTable.isRowGrandTotalEnabled())
```

## 6.89.7 Метод `PivotTable.isColumnGrandTotalEnabled`

Метод возвращает `True`, если разрешено показывать общие итоги для столбцов.

**Пример:**

```
print(pivotTable.isColumnGrandTotalEnabled())
```

## 6.89.8 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)
```

## 6.89.9 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)
```

## 6.89.10 Метод `PivotTable.getUnsupportedFeatures`

Метод возвращает неподдерживаемые свойства [PivotTableUnsupportedFeature](#) сводной таблицы.

### Пример:

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

## 6.89.11 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)
```

## 6.89.12 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)
```

## 6.89.13 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)
```

## 6.89.14 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)
```

## 6.89.15 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)
```

## 6.89.16 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)
```

## 6.89.17 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())
```

## 6.89.18 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())
```

### 6.89.19 Метод PivotTable.getFilter

Метод возвращает фильтр [PivotTableFilter](#) по заданному имени поля fieldName.

## Пример:

```
pivotTableFilter = pivotTable.getFilter("Age")
print(pivotTableFilter.getFieldName())
```

### 6.89.20 Метод PivotTable.getFilters

Метод возвращает список фильтров [PivotTableFilter](#) сводной таблицы.

## Пример:

```
pivotTableFilters = pivotTable.getFilters()
pivotTableFiltersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in pivotTableFiltersEnumerator:
    pivotTableFilter.setHidden()
```

### 6.89.21 Метод PivotTable.update

Метод обновляет и полностью пересчитывает сводную таблицу, возвращает [PivotTableUpdateResult](#).

## Пример:

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

### 6.89.22 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()
```

## 6.90 Класс PivotTableCaptions

Класс `PivotTableCaptions` хранит все пользовательские заголовки сводной таблицы. Может быть получен вызовом [PivotTable.getPivotTableCaptions\(\)](#). Описание полей таблицы представлено в таблице 54.

Таблица 54 – Описание полей класса `PivotTableCaptions`

Поле	Описание
<code>PivotTableCaptions.errorCaption</code>	Алиас для значений, которые возвращают ошибку.
<code>PivotTableCaptions.emptyCaption</code>	Алиас для значений, которые возвращают пустое значение.
<code>PivotTableCaptions.grandTotalCaption</code>	Алиас общих итогов.
<code>PivotTableCaptions.valuesHeaderCaption</code>	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'.
<code>PivotTableCaptions.rowHeaderCaption</code>	Алиас заголовка строк (виден только при включенном компактном макете, это алиас по умолчанию).
<code>PivotTableCaptions.columnHeaderCaption</code>	Алиас заголовка колонок (виден только при включенном компактном макете, это алиас по умолчанию).

## 6.91 Класс PivotTableCategoryField

`PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. таблицу 55). Объект может быть получен посредством вызовов [PivotTable.getRowFields\(\)](#), [PivotTable.getColumnFields\(\)](#).

Таблица 55 – Описание полей `PivotTableCategoryField`

Поле	Описание
<code>PivotTableCategoryField.fieldProperties</code>	Свойства поля <a href="#">PivotTableFieldProperties</a>
<code>PivotTableCategoryField.subtotalFunctions</code>	Список функций <a href="#">PivotTableFunction</a> для вычисления подытога

## 6.92 Класс PivotTableEditor

Предназначен для редактирования сводных таблиц. Возвращается посредством метода [PivotTable.createPivotTableEditor\(\)](#).

### 6.92.1 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()
```

### 6.92.2 Метод PivotTableEditor.moveField

Метод перемещает поле между категориями. Параметры: `fieldName` - имя поля, `toCategory` - область, в которую перемещается поле (тип - [PivotTableFieldCategory](#)), `index` - позиция в новой категории. Метод возвращает объект [PivotTableEditor](#).

#### Пример:

```
pivotTableEditor = pivotTable.createPivotTableEditor()  
pivotTableEditor = pivotTableEditor.moveField("CC",  
myOfficeSDK.PivotTableFieldCategory_Values, 0)  
pivotTableEditor.apply()
```

### 6.92.3 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()
```

## 6.92.4 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()
```

## 6.92.5 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()
```

## 6.92.6 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()
```

## 6.92.7 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)
```

## 6.92.8 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()
```

## 6.92.9 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)
```

## 6.92.10 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()
```

### 6.92.11 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.setLayoutSettings(pivotTableLayoutSettings)
pivotTableEditor.apply()
```

### 6.92.12 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.setGrandTotalSettings(True, True)
pivotTableEditor.apply()
```

### 6.92.13 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")
```

## 6.93 Класс PivotTableFieldCategories

Класс обеспечивает доступ к списку категорий поля сводной таблицы. Объект может быть получен посредством использования метода [PivotTable.getFieldCategories\(\)](#).

### 6.93.1 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())
```

## 6.94 Класс PivotTableFilters

Класс обеспечивает доступ к списку фильтров. Для получения объекта PivotTableFilters используется метод [PivotTable.getFilters\(\)](#).

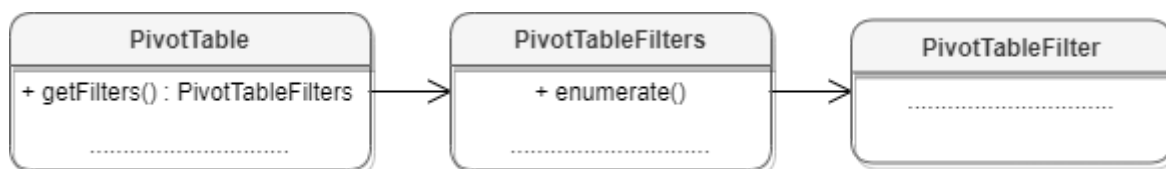


Рисунок 32 – Объектная модель классов для работы с фильтрами

### 6.94.1 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())
```

## 6.95 Класс PivotTableFunction

Класс `PivotTableFunction` описывает функции, которые могут быть использованы в сводных таблицах. Описание полей класса представлено в таблице 56. Объект используется в качестве поля `subtotalFunctions` класса [PivotTableCategoryField](#).

Таблица 56 – Описание полей класса `PivotTableFunction`

Поле	Описание
<code>PivotTableFunction_Auto</code>	Автозаполнение
<code>PivotTableFunction_Sum</code>	Суммирует все числовые данные
<code>PivotTableFunction_Count</code>	Количество всех ячеек
<code>PivotTableFunction_CountNums</code>	Количество числовых ячеек
<code>PivotTableFunction_Average</code>	Среднее значение
<code>PivotTableFunction_Max</code>	Наибольшее значение
<code>PivotTableFunction_Min</code>	Наименьшее значение
<code>PivotTableFunction_Product</code>	Произведение всех ячеек
<code>PivotTableFunction_StdDeviation</code>	Стандартное смещенное отклонение
<code>PivotTableFunction_StdDeviationPopulation</code>	Стандартное несмещенное отклонение
<code>PivotTableFunction_Variance</code>	Смещенная дисперсия
<code>PivotTableFunction_VariancePopulation</code>	Несмещенная дисперсия

## 6.96 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()
```

## 6.96.1 Метод PivotTableFilter.getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

**Пример:**

```
pivotTableFilters = pivotTable.getFilters()
filtersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in filtersEnumerator:
    print(pivotTableFilter.getFieldName())
```

## 6.96.2 Метод PivotTableFilter.getCount

Возвращает количество фильтруемых полей.

**Пример:**

```
pivotTableFilters = pivotTable.getFilters()
filtersEnumerator = pivotTableFilters.getEnumerator()
for pivotTableFilter in filtersEnumerator:
    print(pivotTableFilter.getCount())
```

## 6.96.3 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))
```

## 6.96.4 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.isHidden(i))
```

### 6.96.5 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)
```

### 6.97 Класс PivotTableField

Класс `PivotTableField` содержит свойства полей сводной таблицы (см. таблицу 57). Объект может быть получен посредством вызова [PivotTable.getFieldsList\(\)](#).

Таблица 57 – Описание полей класса `PivotTableField`

Поле	Описание
<code>PivotTableField.fieldProperties</code>	Свойства полей сводной таблицы <a href="#">PivotTableFieldProperties</a>
<code>PivotTableField.fieldCategories</code>	Категории полей сводной таблицы <a href="#">PivotTableFieldCategories</a>
<code>PivotTableField.customFormula</code>	Вычисляемая формула (строка)

### 6.98 Класс PivotTableFieldCategory

Класс `PivotTableFieldCategory` описывает флаги, которые задают категорию области полей. Описание полей представлено в таблице 58. Пример использования см. в [PivotTable.getFieldCategories\(\)](#).

Таблица 58 – Описание полей класса `PivotTableFieldCategory`

Поле	Описание
<code>PivotTableFieldCategory_Pages</code>	Область фильтров

Поле	Описание
PivotTableFieldCategory_Rows	Область строк
PivotTableFieldCategory_Columns	Область колонок
PivotTableFieldCategory_Values	Область значений

## 6.99 Класс PivotTableFieldProperties

PivotTableFieldProperties содержит свойства поля [PivotTableField](#) сводной таблицы (см. таблицу 59).

Таблица 59 – Описание полей класса PivotTableFieldProperties

Поле	Описание
PivotTableFieldProperties.fieldName	Имя поля
PivotTableFieldProperties.fieldAlias	Псевдоним поля (пользовательское имя)
PivotTableFieldProperties.subtotalAlias	Псевдоним подытогов конкретного поля

## 6.100 Класс PivotTableItem

PivotTableItem описывает элемент сводной таблицы (см. Рисунок 33). См. пример в главе [PivotTableItems.getEnumerator\(\)](#).

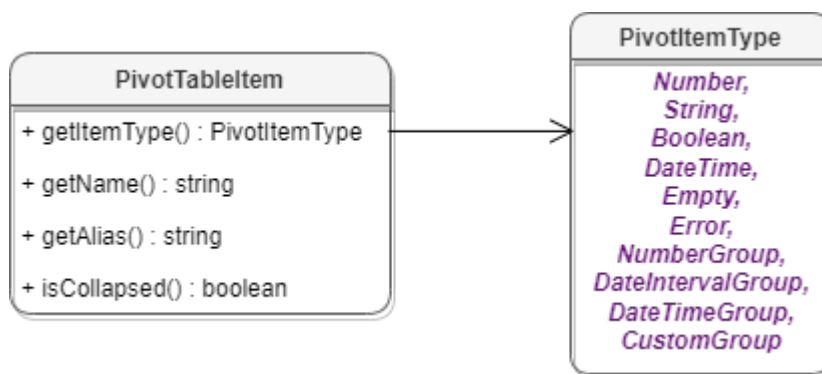


Рисунок 33 – Класс PivotTableItem

### 6.100.1 Метод PivotTableItem.getName

Метод возвращает имя элемента сводной таблицы, тип - строка. См. пример в главе [PivotTableItems.getEnumerator\(\)](#).

## 6.100.2 Метод `PivotTableItem.getAlias`

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем), тип - строка. См. пример в главе [PivotTableItems.getEnumerator\(\)](#).

## 6.100.3 Метод `PivotTableItem.getItemType`

Метод возвращает тип [PivotTableItemType](#) элемента сводной таблицы. См. пример в главе [PivotTableItems.getEnumerator\(\)](#).

## 6.100.4 Метод `PivotTableItem.isCollapsed`

Метод возвращает `True`, если элемент сводной таблицы свернут. См. пример в главе [PivotTableItems.getEnumerator\(\)](#).

## 6.101 Класс `PivotTableItems`

Класс обеспечивает доступ к списку элементов сводной таблицы (см. Рисунок 34).

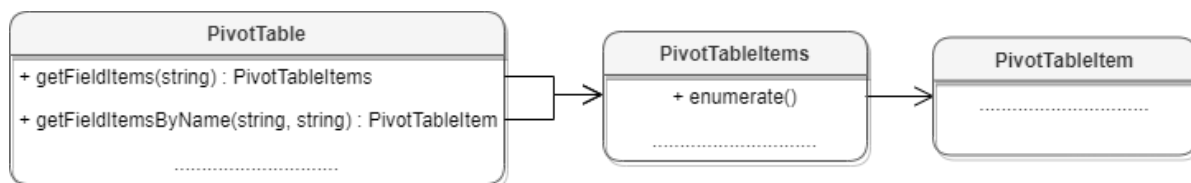


Рисунок 34 – Объектная модель классов для работы с элементами сводных таблиц

### 6.101.1 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.GetItemTypeId())
        print(pivotTableItem.isCollapsed())
```

## 6.102 Класс PivotTableItemType

Класс `PivotTableItemType` содержит возможные типы элементов сводной таблицы. Описание полей представлено в таблице 60.

Таблица 60 – Описание полей класса `PivotTableItemType`

Поле	Описание
<code>PivotTableItemType_Number</code>	Числовой
<code>PivotTableItemType_String</code>	Строковый
<code>PivotTableItemType_Boolean</code>	Логический
<code>PivotTableItemType_DateTime</code>	Дата / время
<code>PivotTableItemType_Empty</code>	Пустой тип
<code>PivotTableItemType_Error</code>	Ошибка
<code>PivotTableItemType_NumberGroup</code>	Интервальная группировка
<code>PivotTableItemType_DateIntervalGroup</code>	Интервальная группировка по датам
<code>PivotTableItemType_DateTimeGroup</code>	Группировка по дате / времени
<code>PivotTableItemType_CustomGroup</code>	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())
```

## 6.103 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 61.

Таблица 61 – Описание полей класса `PivotTableLayoutSettings`



Поле	Описание
<code>PivotTableLayoutSettings.reportLayout</code>	Настройка вида макета сводной таблицы ( <a href="#">PivotTableReportLayout</a> : компактный, табличный, структурный).
<code>PivotTableLayoutSettings.valueFieldsOrientation</code>	Настраивает положение значений в случае, если в сводной таблице более двух полей значений. Тип - <a href="#">ValueFieldsOrientation</a> .
<code>PivotTableLayoutSettings.pageFieldOrder</code>	Настройка порядка полей фильтров ( <a href="#">PageFieldOrder</a> : вниз, затем поперек или сначала поперек, потом вниз).
<code>PivotTableLayoutSettings.indentForCompactLayout</code>	Размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей).
<code>PivotTableLayoutSettings.pageFieldWrapCount</code>	Настройка связана с <code>pageFieldOrder</code> , она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д.).
<code>PivotTableLayoutSettings.isMergeAndCenterLabelsEnabled</code>	Настройка позволяет объединить ячейки заголовков.
<code>PivotTableLayoutSettings.useGridDropZones</code>	Флаг, отвечающий за отображение классического вида (как в Excel 2003). Влияет только на расположение полей в отчете.
<code>PivotTableLayoutSettings.displayFieldCaptions</code>	Флаг, отвечающий за отображение заголовков полей.

## 6.104 Класс PivotTablePageField

Содержит свойства поля из области фильтров (см. таблицу 62). Объект может быть получен посредством вызова [PivotTable.getPageFields\(\)](#).

Таблица 62 – Описание полей класса PivotTablePageField

Поле	Описание
<code>PivotTablePageField.fieldProperties</code>	Свойства поля <a href="#">PivotTableFieldProperties</a>

## 6.105 Класс PivotTableReportLayout

Таблица `PivotTableReportLayout` описывает внешний вид отчетов сводной таблицы. Является полем класса [PivotTableLayoutSettings](#). Описание полей представлено в таблице 63.

Таблица 63 – Описание полей класса PivotTableReportLayout

Поле	Описание
PivotTableReportLayout_Compact	Компактный вид
PivotTableReportLayout_Tabular	Табличный вид
PivotTableReportLayout_Outline	Структурный вид

## 6.106 Класс PivotTableValueField

PivotTableValueField содержит свойства поля сводной таблицы, используемого как значение столбец (см. таблицу 64). Объект класса может быть получен посредством вызова [PivotTable.GetValueFields\(\)](#).

Таблица 64 – 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 <a href="#">CellFormat</a> для конкретного поля значений.
PivotTableValueField.totalFunction	Агрегирующая функция <a href="#">PivotTableFunction</a> поля значений (SUM, COUNT, MAX и т.д).
PivotTableValueField.customFormula	Вычисляемая формула для поля значений, тип - строка.

## 6.107 Класс PivotTableUnsupportedFeature

Класс PivotTableUnsupportedFeature описывает неподдерживаемую функциональность сводных таблиц. Получение неподдерживаемой функциональности сводных таблиц описано в [PivotTable.getUnsupportedFeatures\(\)](#). Описание полей таблицы представлено в таблице 65.

Таблица 65 – Описание полей класса PivotTableUnsupportedFeature

Поле	Описание
PivotTableUnsupportedFeature_CalculatedField	Вычисляемые поля
PivotTableUnsupportedFeature_CalculatedItem	Вычисляемые элементы

Поле	Описание
PivotTableUnsupportedFeature_CollapsedValues	Свернутые поля
PivotTableUnsupportedFeature_ShowDataAs	Вычисления ("Show data" как в MS Excel)

## 6.108 Класс PivotTableUpdateResult

В таблице 66 приведены константы, которые соответствуют возможным результатам обновления сводной таблицы (см. методы [PivotTable.update\(\)](#), [PivotTableEditor.apply\(\)](#)).

Таблица 66 – 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
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

Наименование константы	Описание
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

## 6.109 Класс PointU

Класс `PointU` представляет точку двумерном пространстве. Описание полей класса `PointU` представлено в таблице 67.

Таблица 67 – Описание полей класса `PointU`

Поле	Тип	Описание
<code>PointU.x</code>	number	Координата точки по оси x
<code>PointU.y</code>	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
```

### 6.109.1 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)
```

## 6.110 Класс Position

Класс `Position` представляет местоположение в текстовом документе. Используется для обозначения начала и конца диапазона [Range](#).

## 6.110.1 Метод `Position.getCell`

Метод возвращает ячейку [Cell](#) для заданной позиции.

**Пример:**

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("A3")
rng = cell.getRange()
position = rng.getBegin()
cellAtPosition = position.getCell()
```

## 6.110.2 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")
```

## 6.110.3 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")
```

### 6.110.4 Метод `Position.insertPageBreak`

Метод предназначен для вставки разрыва страницы в указанную позицию в документе.

#### Пример:

```
docRange = document.getRange()  
endPosition = docRange.getEnd()  
endPosition.insertPageBreak()
```

### 6.110.5 Метод `Position.insertLineBreak`

Метод предназначен для вставки перевода строки.

#### Пример:

```
docRange = document.getRange()  
endPosition = docRange.getEnd()  
endPosition.insertLineBreak()
```

### 6.110.6 Метод `Position.insertBookmark`

Вставляет закладку с наименованием в текущую позицию.

#### Пример:

```
document.getRange().getEnd().insertBookmark("Bookmark example")
```

### 6.110.7 Метод `Position.insertSectionBreak`

Вставляет разрыв раздела в текущую позицию.

#### Пример:

```
docRange = document.getRange()  
endPosition = docRange.getEnd()  
endPosition.insertSectionBreak()
```

### 6.110.8 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")
```

### 6.110.9 Метод `Position.insertImage`

Вставляет изображение из файла в текущую позицию.

## Параметры:

- url – полный путь к файлу, строка;
- size – геометрические размеры изображения для вставки, тип [SizeU](#).

## Пример:

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

### 6.110.10 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)
```

### 6.110.11 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)
```

## 6.110.12 Position.\_\_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");
```

## 6.110.13 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");
```

## 6.111 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)
```

## 6.111.1 Метод `PrintingScope.getCellRange`

Метод возвращает диапазон ячеек таблицы (см. [CellRangePosition](#)).

## 6.111.2 Метод `PrintingScope.usePrintArea`

Метод возвращает `True`, если область печати должна использоваться во время печати, экспорта и т. д.

## 6.112 Класс `PrintingScope.Type`

Варианты выбора диапазона страниц для экспорта и печати представлены в таблице 68. Используется в [PrintingScope](#)

Таблица 68 – Диапазон страниц для экспорта и печати

Наименование константы	Описание
<code>PrintingScope.Type_PrintArea</code>	Выбранная область печати (по умолчанию).
<code>PrintingScope.Type_WholeSheet</code>	Печать всего документа.

### Пример:

```
printingScope =
myOfficeSDK.PrintingScope(myOfficeSDK.PrintingScope.Type_PrintArea)
```

## 6.113 Класс `Range`

Класс `Range` предоставляет доступ к диапазону документа. На рисунке 35 изображена объектная модель классов, относящихся к работе с диапазонами.

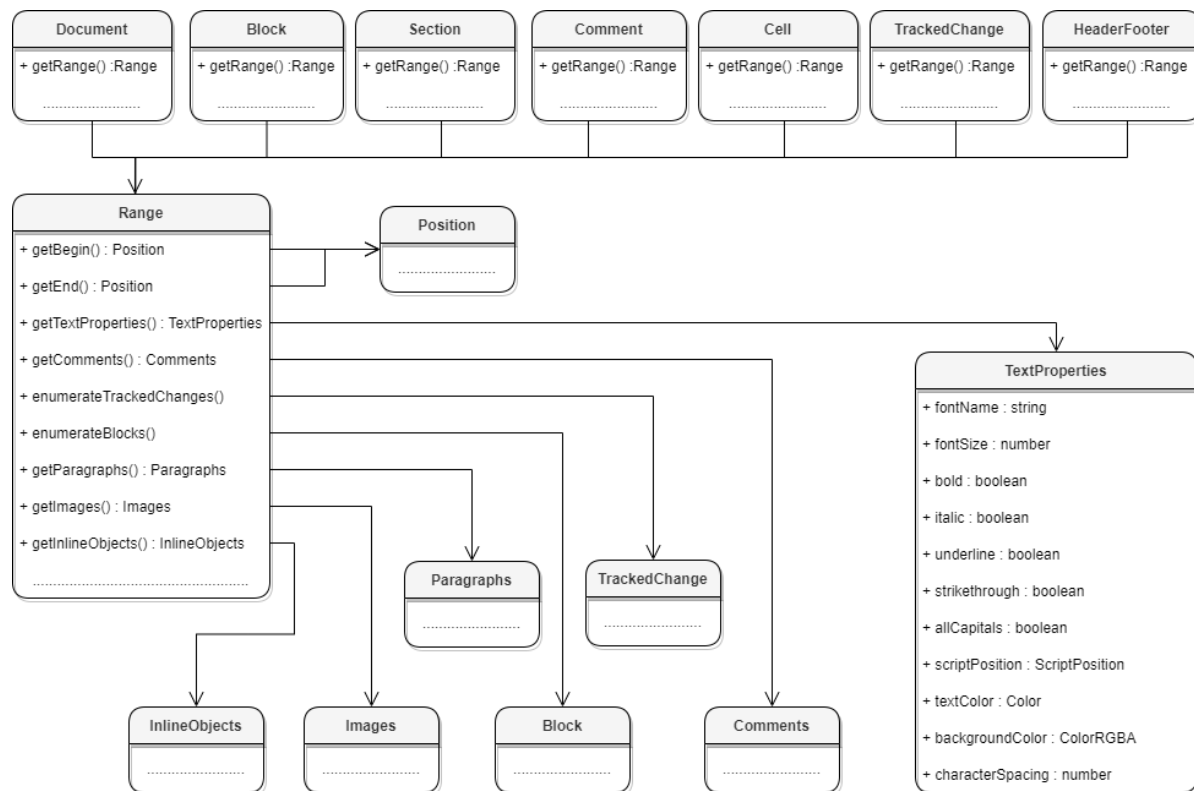


Рисунок 35 – Объектная модель для работы с классом Range

## 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())
```

## 6.113.1 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")
```

## 6.113.2 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")
```

### 6.113.3 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())
```

### 6.113.4 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())
```

## 6.113.5 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())
```

## 6.113.6 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())
```

## 6.113.7 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())
```

## 6.113.8 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")
```

## 6.113.9 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")
```

### 6.113.10 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)
```

### 6.113.11 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)
```

### 6.113.12 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())
```

### 6.113.13 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())
```



## 6.113.14 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())
```

## 6.113.15 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())
```

## 6.113.16 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())
```

## 6.113.17 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())
```

## 6.113.18 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")
```

## 6.113.19 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")
```

## 6.114 Класс `RangeBorders`

Класс `RangeBorders` оставлен для совместимости. Вместо него необходимо использовать класс [Borders](#).

## 6.115 Класс RectU

Класс RectU описывает прямоугольник в двумерном пространстве. Описание полей таблицы RectU представлено в таблице 69.

Таблица 69 – 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
```

### 6.115.1 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)]
```

## 6.116 Класс SaveDocumentSettings

Класс SaveDocumentSettings предоставляет настройки, используемые для сохранения документа в файл, см. [Document.saveAs\(\)](#). Описание полей класса SaveDocumentSettings представлено в таблице 70.

Таблица 70 – Описание полей класса SaveDocumentSettings

Поле	Описание
SaveDocumentSettings_documentFormat	Формат документа <a href="#">DocumentFormat</a>
SaveDocumentSettings_documentType	Тип документа <a href="#">DocumentType</a>
SaveDocumentSettings_documentPassword	Пароль для защиты электронного документа от несанкционированного

Поле	Описание
	доступа
SaveDocumentSettings_isTemplate	Флаг, обозначающий, что документ должен быть сохранен как шаблон
SaveDocumentSettings_dsvSettings	Структура <a href="#">DSVSettings</a> , необходимая для сохранения в формате DSV

## 6.117 Класс ScaleFrom

В таблице 71 представлены позиции якоря при масштабировании объекта `AbsoluteFrame`. Используется в [AbsoluteFrame.scale\(\)](#).

Таблица 71 – 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

## 6.118 Класс Script

Класс `Script` предназначен для управления отдельной макрокомандой. Содержит поля `Name` и `Body`.

### 6.118.1 Метод Script.getName

Метод возвращает имя макрокоманды.

**Пример:**

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

### 6.118.2 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() + "'")
```

### 6.118.3 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())
```

### 6.118.4 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() + "'")
```

### 6.119 Класс ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 72. Используется в качестве поля scriptPosition класса [TextProperties](#).

Таблица 72 – Типы надстрочного и подстрочного форматирования

Наименование константы	Описание
ScriptPosition_SuperScript	Надстрочный знак (верхний индекс)
ScriptPosition_SubScript	Подстрочный знак (нижний индекс)
ScriptPosition_NormalScript	Без указания индекса

**Пример:**

```
textProperties = myOfficeSDK.TextProperties()
textProperties.scriptPosition = myOfficeSDK.ScriptPosition_NormalScript
document.getRange().setTextProperties(textProperties)
```

### 6.120 Класс ScientificCellFormatting

Класс содержит параметры для экспоненциального формата ячеек таблицы. Используется в качестве аргумента метода [Cell.setFormat\(\)](#). Описание полей класса ScientificCellFormatting представлено в таблице 73.

Таблица 73 – Описание полей класса ScientificCellFormatting

Поле	Описание
ScientificCellFormatting.decimalPlaces	Количество десятичных позиций
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())
```

### 6.121 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())
```

### 6.121.1 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())
```

### 6.121.2 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\nscripts: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")
```

### 6.121.3 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")
```

### 6.121.4 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())
```

### 6.122 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);
```

#### 6.122.1 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)
```

### 6.123 Класс `Search`

Класс `Search` предоставляет доступ к механизму поиска и замены фрагментов документа, открытого в редакторе текста или таблиц.



## 6.123.1 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](#).

## 6.124 Класс Section

Класс `Section` представляет собой раздел в документе.

### 6.124.1 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)
```

### 6.124.2 Метод `Section.getPageProperties`

Метод возвращает параметры страниц раздела [PageProperties](#).

**Пример:**

```
for section in sectionsEnumerator:  
    pageProperties = section.getPageProperties()  
    print(pageProperties.height)
```

### 6.124.3 Метод `Section.setPageOrientation`

Метод задает ориентацию страниц раздела.

**Пример:**

```
for section in sectionsEnumerator:  
    section.setPageOrientation(myOfficeSDK.PageOrientation_Portrait)  
    print(section.getPageOrientation())
```

### 6.124.4 Метод `Section.getPageOrientation`

Метод возвращает ориентацию страниц раздела.

**Пример:**

```
for section in sectionsEnumerator:  
    print(section.getPageOrientation())
```

### 6.124.5 Метод `Section.getRange`

Метод возвращает диапазон [Range](#) в документе, соответствующий данному разделу.

**Пример:**

```
for section in sectionsEnumerator:  
    sectionRange = section.getRange()  
    print(sectionRange.extractText())
```

## 6.124.6 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())
```

## 6.124.7 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())
```

## 6.125 Класс Sections

Класс `Sections` используется для доступа к коллекции секций документа. Описание секции см. в разделе [Section](#).

### 6.125.1 Метод Sections.getEnumerator

Метод позволяет перечислить коллекцию секций документа.

**Пример:**

```
sectionsEnumerator = document.getSectionsEnumerator()  
for section in sectionsEnumerator:  
    sectionRange = section.getRange()  
    print(sectionRange.extractText())
```

## 6.126 Класс Shape

Класс `Shape` представляет собой фигуру, содержит методы для установки и получения свойств [ShapeProperties](#).

## 6.126.1 Метод `Shape.getShapeProperties`

Метод возвращает свойства фигуры [ShapeProperties](#).

**Пример:**

```
shape = document.getBlocks().getShape(0)
if shape != None:
    shapeProperties = shape.getShapeProperties()
    print(shapeProperties.verticalAlignment)
```

## 6.126.2 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)
```

## 6.127 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](#).

### 6.127.1 Поле `ShapeProperties.borderProperties`

Поле предназначено для установки свойств границ фигуры [LineProperties](#).

### 6.127.2 Поле `ShapeProperties.verticalAlignment`

Поле предназначено для установки типа вертикального выравнивания [VerticalAlignment](#).

### 6.127.3 Поле `ShapeProperties.fill`

Поле предназначено для установки свойств заполнения фигуры [Fill](#).

## 6.127.4 Поле `ShapeProperties.shapeTextLayout`

Поле предназначено для установки свойств текста внутри фигуры [ShapeTextLayout](#).

## 6.128 Класс `ShapeTextLayout`

Класс `ShapeTextLayout` описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 74. Используется в качестве поля класса [ShapeProperties](#).

Таблица 74 – Описание полей класса `ShapeTextLayout`

Поле	Описание
<code>ShapeTextLayout_DoNotAutoFit</code>	Размещение текста в фигуре по умолчанию
<code>ShapeTextLayout_FitShapeExtentToText</code>	Расширение фигуры под текст
<code>ShapeTextLayout_FitTextToShape</code>	Заполнение фигуры текстом

## 6.129 Класс `SizeU`

Класс `SizeU` представляет размер объекта в двухмерном пространстве. Описание полей класса `SizeU` представлено в таблице 75.

Таблица 75 – Описание полей классе `SizeU`

Поле	Тип	Описание
<code>SizeU.width</code>	number	Ширина
<code>SizeU.height</code>	number	Высота

### Пример:

```
size = myOfficeSDK.SizeU(2, 3)
print("width=", size.width, ", height=", size.height) # width= 2.0 , height= 3.0
```

### 6.129.1 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)
```

## 6.130 Класс Table

Класс Table предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 36).

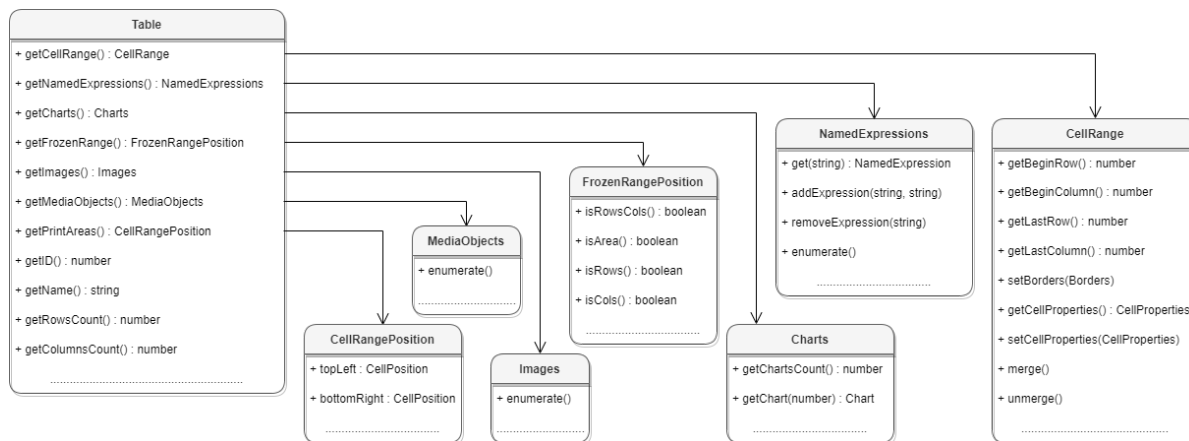


Рисунок 36 – Структура полей класса Table

### 6.130.1 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.

### 6.130.2 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.

## 6.130.3 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
- [OutOfRangeException](#): 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](#).

## 6.130.4 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()
```

## 6.130.5 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())
```

## 6.130.6 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())
```

## 6.130.7 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)
```



## 6.130.8 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())
```

## 6.130.9 Метод `Table.getColumnsCount`

Метод позволяет получить количество столбцов таблицы.

### Пример:

```
table = document.getBlocks().getTable("List11")  
print(table.getColumnsCount())
```

## 6.130.10 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](#).

## 6.130.11 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())
```

### 6.130.12 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())
```

### 6.130.13 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)
```

### 6.130.14 Метод `Table.getName`

Метод позволяет получить наименование листа табличного документа.

### Пример:

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

### 6.130.15 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())
```

### 6.130.16 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())
```

### 6.130.17 Метод Table.getRowsCount

Метод позволяет получить количество строк таблицы.

## Пример:

```
table = document.getBlocks().getTable("List11")
print(table.getRowsCount())
```

### 6.130.18 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())
```

### 6.130.19 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.

### 6.130.20 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.

### 6.130.21 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)
```

## 6.130.22 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)
```

## 6.130.23 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)
```

### 6.130.24 Метод Table.insertRowBefore

Метод предназначен для вставки новой строки до указанной позиции в таблице.

#### Вызов:

```
insertRowBefore(rowIndex, copyRowStyle, rowCount)
```

#### 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`.
- `rowCount` – 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)
```

### 6.130.25 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)
```

## 6.130.26 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](#).

## 6.130.27 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](#).

## 6.130.28 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()
```

## 6.130.29 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)
```

## 6.130.30 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)
```

## 6.130.31 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 `rowCount` строк. Индексация строк начинается с нуля.
- `rowCount` – 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)
```

### 6.130.32 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)
```

### 6.130.33 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.

## 6.130.34 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))
```

## 6.130.35 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())
```

## 6.130.36 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)
```

## 6.130.37 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)
```

### 6.130.38 Метод `Table.setRowsVisible`

Метод `Table.setRowsVisible` позволяет задавать видимость строк, начиная с заданного индекса. Индексация строк начинается с нуля.

## Вызов:

```
setRowsVisible(first, rowCount, visible)
```

## Параметры:

`first` – начальный индекс;  
`rowCount` – количество строк;  
`visible` – видимость.

### 6.130.39 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)
```

## 6.130.40 Метод `Table.setVisible`

Метод управляет видимостью листа таблицы. Используется только в табличном документе.

### Пример:

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

## 6.130.41 Метод `Table.ungroupRows`

Метод предназначен для разгруппировки строк таблицы, начиная с заданного индекса. Индексация строк начинается с нуля.

### Вызов:

```
ungroupRows(rowIndex, rowCount)
```

### Параметры:

- `rowIndex` – индекс строки, начиная с которого будет начата разгруппировка строк;
- `rowCount` – количество строк для разгруппировки.

## 6.130.42 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.

## 6.130.43 `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")
```

## 6.130.44 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")
```

## 6.131 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](#).

### 6.131.1 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()
```

## 6.131.2 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)
```

## 6.131.3 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](#).

## 6.132 Класс `TableRangeInfo`

Класс `TableRangeInfo` описывает диапазон ячеек таблицы.

Описание полей класса `TableRangeInfo` представлено в таблице 76.

Таблица 76 – Поля класса `TableRangeInfo`

Поле	Тип	Описание
<code>tableRange</code>	<a href="#">CellRangePosition</a>	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)
```

## 6.133 Класс `TextAnchoredPosition`

Класс `TextAnchoredPosition` представляет позицию объекта на странице текстового документа. Описание полей представлено в таблице 77.

Таблица 77 – Описание полей класса `TextAnchoredPosition`

Поле	Описание
<code>TextAnchoredPosition.horizontal</code>	Позиция по горизонтали <a href="#">HorizontalTextAnchoredPosition</a>
<code>TextAnchoredPosition.vertical</code>	Позиция по вертикали <a href="#">VerticalTextAnchoredPosition</a>

### 6.133.1 `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")
```

## 6.133.2 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.alignment =
myOfficeSDK.VerticalAnchorAlignment_Center
firstTextAnchoredPosition.vertical.offset = 10

secondTextAnchoredPosition = myOfficeSDK.TextAnchoredPosition()
secondTextAnchoredPosition.horizontal =
myOfficeSDK
.HorizontalTextAnchoredPosition
(myOfficeSDK.HorizontalRelativeTo_ColumnLeftMargin)
secondTextAnchoredPosition.horizontal.alignment =
myOfficeSDK.HorizontalAnchorAlignment_Center
secondTextAnchoredPosition.horizontal.offset = 20
secondTextAnchoredPosition.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageContent)
secondTextAnchoredPosition.vertical.alignment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondTextAnchoredPosition.vertical.offset = 20

if firstTextAnchoredPosition.__ne__(secondTextAnchoredPosition):
    print("Not equals")
```

## 6.134 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)
```

## 6.135 Класс TextLayout

В таблице 78 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле `textLayout` таблицы [CellProperties](#).

Таблица 78 – 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)
```

### 6.136 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)
```

## 6.136.1 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())
```

## 6.136.2 `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())
```

## 6.136.3 `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")
```

## 6.136.4 `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")
```

## 6.137 Класс `TextProperties`

Класс `DocumentAPI.TextProperties` содержит поля, задающие параметры текста. На рисунке 37 изображена объектная модель класса `DocumentAPI.TextProperties`.

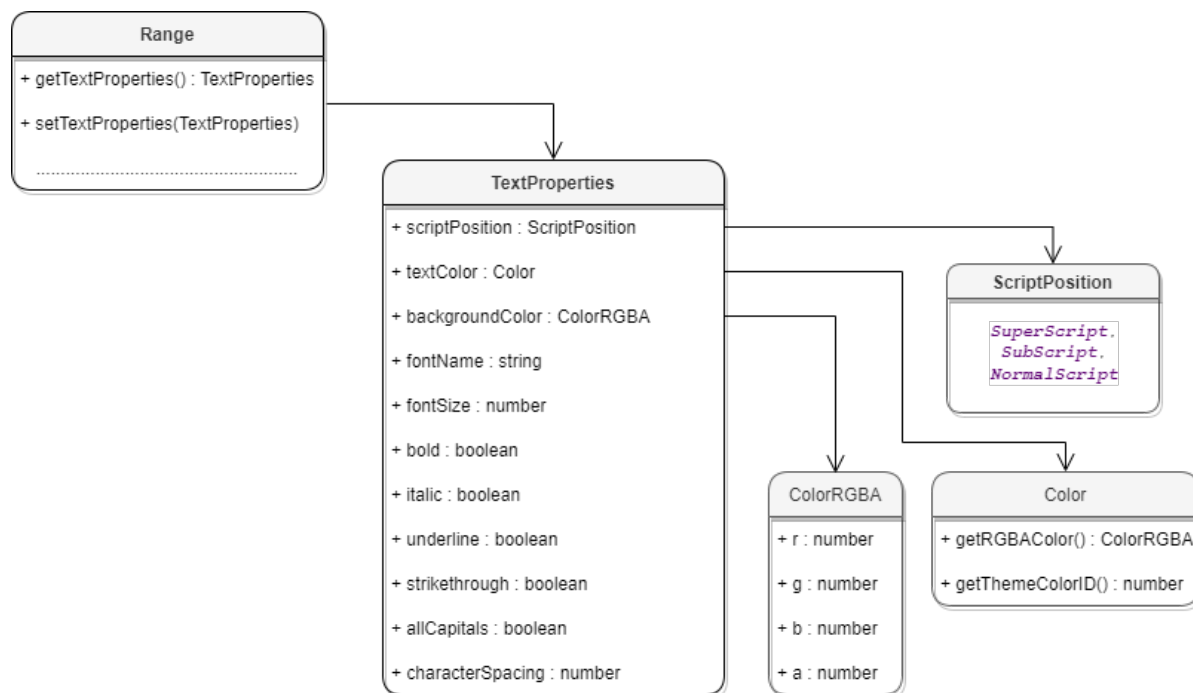


Рисунок 37 – Объектная модель для работы с классом

DocumentAPI.TextProperties

Описание полей класса TextProperties представлено в таблице 79. Свойства TextProperties применяются к диапазону текста Range (методы [Range.getTextProperties\(\)](#), [Range.setTextProperties\(\)](#)).

Таблица 79 – 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.
TextProperties.strikethrough	Логическое	Значение True устанавливает начертание «зачеркнутый» для указанного фрагмента текста.

Поле	Тип	Описание
<code>TextProperties.allCapitals</code>	Логическое	Значение <code>True</code> устанавливает все буквы указанного фрагмента текста как прописные. Значение <code>False</code> устанавливает все буквы указанного фрагмента текста как строчные.
<code>TextProperties.scriptPosition</code>	<a href="#">ScriptPosition</a>	Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме.
<code>TextProperties.textColor</code>	<a href="#">Color</a>	Цвет указанного фрагмента документа.
<code>TextProperties.backgroundcolor</code>	<a href="#">ColorRGBA</a>	Цвет фона указанного фрагмента документа.
<code>TextProperties.characterSpacing</code>	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)
```

### 6.137.1 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")
```

## 6.137.2 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")
```

## 6.138 Класс TextWrapType

В таблице 80 представлены варианты обтекания текстом встроенного объекта. Используется в [InlineFrame.setWrapType\(\)](#).

Таблица 80 – Варианты обтекания текстом встроенного объекта

Наименование константы	Описание
<code>TextWrapType_Inline</code>	Встроенный объект располагается в тексте
<code>TextWrapType_InFrontOfText</code>	Встроенный объект располагается перед текстом
<code>TextWrapType_BehindText</code>	Встроенный объект располагается за текстом
<code>TextWrapType_TopAndBottom</code>	Текст располагается сверху и снизу от встроенного объекта
<code>TextWrapType_Square</code>	Текст располагается вокруг прямоугольной рамки встроенного объекта
<code>TextWrapType_Through</code>	Текст обтекает встроенный объект по сторонам и внутри
<code>TextWrapType_Tight</code>	Текст располагается на одинаковых расстояниях от границ объекта

## 6.139 Класс ThemeColorID

В таблице 81 представлены типы идентификаторов цветов тем. Используется в [Color](#).

Таблица 81 – Типы идентификаторов цветов тем

Наименование константы	Описание
<code>ThemeColorID_Background1</code>	Фон1

Наименование константы	Описание
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

## 6.140 Класс TimePatterns

Форматы времени представлены в таблице 82. Пример использования см. в главе [DateTimeCellFormatting](#).

Таблица 82 – 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

## 6.141 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).

## 6.142 Класс `TrackedChange`

Класс `TrackedChange` представляет отслеживаемое изменение в диапазоне документа (см. Рисунок 38).

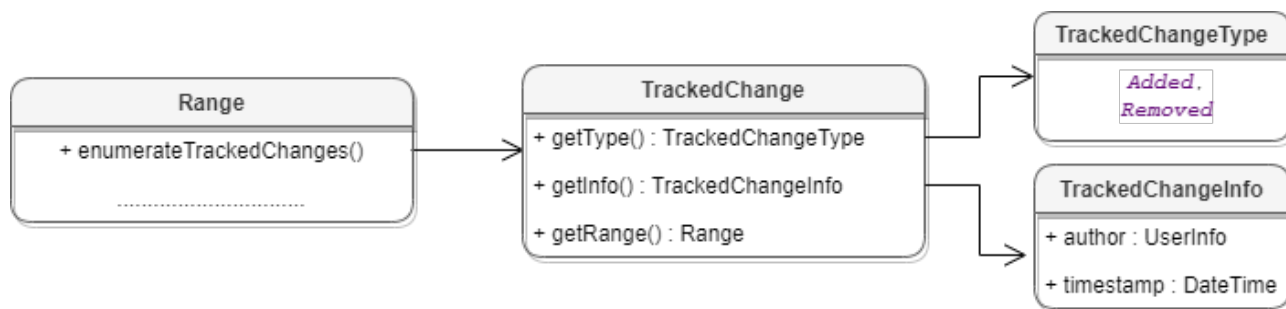


Рисунок 38 – Объектная модель классов для работы с отслеживаемыми изменениями

Для получения списка отслеживаемых изменений используется метод [`Range.getTrackedChangesEnumerator\(\)`](#).

### Пример:

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
```

### 6.142.1 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())
```

### 6.142.2 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()
```

### 6.142.3 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()
```

### 6.143 Класс TrackedChangeInfo

Класс TrackedChangeInfo содержит информацию об отслеживаемых изменениях. Описание полей представлено в таблице 83.

Таблица 83 – Описание полей класса TrackedChangeInfo

Поле	Тип	Описание
TrackedChangeInfo.author	<a href="#">UserInfo</a>	Автор изменений
TrackedChangeInfo.timeStamp	<a href="#">DateTime</a>	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)
```

## 6.144 Класс TrackedChangeType

Поддерживаемые типы отслеживаемых изменений представлены в таблице 84.

Таблица 84 - 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")
```

## 6.145 Класс UserInfo

Класс `UserInfo` предоставляет информацию о пользователе.

Описание полей таблицы `UserInfo` представлено в таблице 85.

Таблица 85 – Описание полей класса `UserInfo`

Поле	Описание	Тип
<code>UserInfo.name</code>	Имя пользователя	Строка
<code>UserInfo.email</code>	Адрес электронной почты пользователя	Строка

### 6.145.1 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");
```

## 6.145.2 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")
```

## 6.146 Класс ValueFieldsOrientation

Класс `ValueFieldsOrientation` 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 86.

Таблица 86 – Описание полей `ValueFieldsOrientation`

Поле	Описание
<code>ValueFieldsOrientation_ByRows</code>	По строкам
<code>ValueFieldsOrientation_ByColumns</code>	По столбцам

## 6.147 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](#).

## 6.147.1 Метод `ValuesTableFilter.add`

Метод `ValuesTableFilter.add` добавляет значение, которое должно быть отображено в таблице.

**Пример:**

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

## 6.147.2 Метод `ValuesTableFilter.clear`

Метод `ValuesTableFilter.clear` удаляет все элементы фильтра.


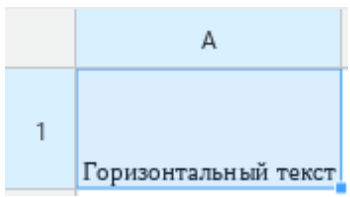
**Пример:**


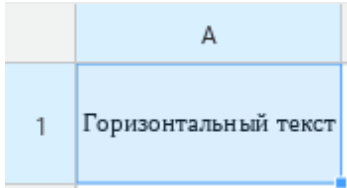

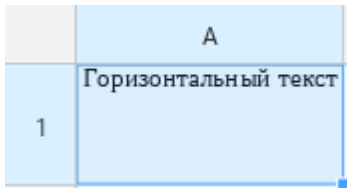
```
johnPaulFilter = myOfficeSDK.ValuesTableFilter()  
johnPaulFilter.add("John")  
johnPaulFilter.add("Paul")  
.....  
johnPaulFilter.clear()
```

## 6.148 Класс `VerticalAlignment`

В таблице 87 представлены константы, описывающие варианты выравнивания текста по вертикали. Используется в [CellProperties](#), [ShapeProperties](#).

Таблица 87 – Виды выравнивания текста по вертикали

Наименование константы	Представление в интерфейсе
<code>VerticalAlignment_Bottom</code>	 

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)
    
```

### 6.149 Класс VerticalAnchorAlignment

В Таблице 88 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали.

Таблица 88 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
VerticalAnchorAlignment_Top	По верхнему краю
VerticalAnchorAlignment_Bottom	По нижнему краю
VerticalAnchorAlignment_Center	По центру

Наименование константы	Описание
VerticalAnchorAlignment_Inside, VerticalAnchorAlignment_Outside	По границам

## 6.150 Класс VerticalRelativeTo

В таблице 89 представлены типы размещения объекта относительно закрепленной позиции по вертикали.

Таблица 89 – 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

## 6.151 Класс VerticalTextAnchoredPosition

Класс VerticalTextAnchoredPosition предназначен для управления относительным положением объекта со смещением или выравниванием по вертикали.

Описание полей класса VerticalTextAnchoredPosition представлено в таблице 90.

Таблица 90 – Описание полей класса VerticalTextAnchoredPosition

Поле	Описание
VerticalTextAnchoredPosition.relativeTo	Тип размещения объекта относительно закрепленной позиции по <a href="#">VerticalRelativeTo</a> вертикали

Поле	Описание
<code>VerticalTextAnchoredPosition.offset</code>	Смещение объекта.
<code>VerticalTextAnchoredPosition.aligment</code>	Тип выравнивания объекта относительно закрепленной позиции по вертикали <a href="#">VerticalAnchorAlignment</a>

## 6.151.1 `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")
```

## 6.151.2 `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.alignment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondVerticalTextAnchoredPosition.offset = 20

if firstVerticalTextAnchoredPosition.__ne__(secondVerticalTextAnchoredPosition):
    print("Not equals")
```

## 6.152 Класс WorkbookExportSettings

Класс `WorkbookExportSettings` предоставляет настройки, необходимые для экспорта табличных документов, см. [Document.exportAs\(\)](#).

Описание полей класса `WorkbookExportSettings` представлено в таблице 91.

Таблица 91 – Описание полей класса `WorkbookExportSettings`

Поле	Описание
<code>sheetNames</code>	Представляет коллекцию имен листов для экспорта, тип <code>VectorString</code> . Если коллекция пуста, экспортируются все листы.
<code>printingScope</code>	Представляет область печати (весь документ, область печати, пользовательский диапазон и т. д.) <a href="#">PrintingScope</a> .
<code>pageProperties</code>	Представляют свойства страницы для выходного документа (высота и ширина страницы в пунктах pt) <a href="#">PageProperties</a> .
<code>scale</code>	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)
```



## 6.153 Исключения

### 6.153.1 Класс `BaseError`

Класс `BaseError` является базовым классом для всех исключений SDK.

```
class BaseError(Exception)
```

### 6.153.2 Класс `ApplicationCreateError`

Исключение `ApplicationCreateError` вызывается в случае, когда объект [Application](#) не может быть создан.

```
class ApplicationCreateError(BaseError)
```

### 6.153.3 Класс `IncorrectArgumentError`

Исключение `IncorrectArgumentError` вызывается в случае, когда один из аргументов метода или функции имеет недействительное значение.

```
class IncorrectArgumentError(BaseError)
```

### 6.153.4 Класс `InvalidObjectError`

Исключение `InvalidObjectError` вызывается в случае, когда объект больше не может быть использован.

```
class InvalidObjectError(BaseError)
```

### 6.153.5 Класс `DocumentCreateError`

Исключение `DocumentCreateError` вызывается в случае, когда документ не может быть создан.

```
class DocumentCreateError(BaseError)
```

### 6.153.6 Класс `DocumentLoadError`

Исключение `DocumentLoadError` вызывается в случае, когда документ не может быть загружен.

```
class DocumentLoadError(BaseError)
```

### 6.153.7 Класс `DocumentSaveError`

Исключение `DocumentSaveError` вызывается в случае, когда документ не может быть сохранен.

```
class DocumentSaveError(BaseError)
```

## 6.153.8 Класс `DocumentExportError`

Исключение `DocumentExportError` вызывается в случае, когда документ не может быть экспортирован.

```
class DocumentExportError(BaseError)
```

## 6.153.9 Класс `NoSuchElementError`

Исключение `NoSuchElementError` вызывается в случае, когда элемент не существует.

```
class NoSuchElementError(BaseError)
```

## 6.153.10 Класс `NotImplementedError`

Исключение `NotImplementedError` вызывается в случае, если обнаружена нереализованная функциональность.

```
class NotImplementedError(BaseError)
```

## 6.153.11 Класс `OutOfRangeException`

Исключение `OutOfRangeException` вызывается в случае обнаружения выхода значения за пределы диапазона.

```
class OutOfRangeError(BaseError)
```

## 6.153.12 Класс `ParseError`

Исключение `ParseError` вызывается в случае ошибки синтаксического разбора текста.

```
class ParseError(BaseError)
```

## 6.153.13 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)
```

## 6.153.14 Класс `ForbiddenActionError`

Исключение `ForbiddenActionError` вызывается в случае выполнения запрещенной операции.

```
class ForbiddenActionError(BaseError)
```

## 6.153.15 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)
```

## 6.153.16 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)
```

## 6.153.17 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)
```

## 6.153.18 Класс `ScriptExecutionError`

Исключение `ScriptExecutionError` вызывается в случае, когда сценарий не удастся выполнить (см. [Scripting.runScript\(\)](#)).

```
class ScriptExecutionError(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.)
```