

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

**Программное обеспечение
МойОфис Комплект Средств Разработки (SDK)**

MyOffice Document Application Programming Interface (API)

Библиотека для языка программирования Python

3.6

Руководство программиста

На 482 листах

Версия документа: 1

Дата публикации: 25.11.2025

**Москва
2025**

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	38
1.1 Назначение	38
1.2 Библиотека MyOffice Document API для языка программирования Python	38
1.3 Уровень подготовки пользователя	39
1.4 Системные требования	39
2 Подготовка к работе	40
2.1 Список дистрибутивов	40
2.2 Установка в ОС Microsoft Windows	40
2.3 Установка в ОС Linux	40
2.4 Проверка работоспособности	41
2.5 Распространение разработанных приложений	42
3 Объектная модель МойОфис SDK	43
4 Работа с документами	45
4.1 Работа с текстовым документом	45
4.1.1 Создание и открытие текстового документа	45
4.1.2 Сохранение и экспорт текстового документа	46
4.1.3 Разделы (секции) документа	46
4.1.3.1 Работа с колонтитулами раздела	48
4.1.3.2 Управление ориентацией и свойствами страниц раздела	48
4.1.4 Встроенные объекты в текстовом документе	49
4.1.4.1 Вставка изображения	49
4.1.4.2 Перечисление встроенных объектов	50
4.1.5 Работа с таблицами текстового документа	50
4.1.6 Работа с закладками	52
4.1.7 Рецензирование документов	53
4.1.8 Работа с элементами управления	54
4.2 Работа с табличным документом	55
4.2.1 Создание и открытие табличного документа	55

4.2.2	Сохранение и экспорт табличного документа	56
4.2.3	Диаграммы	57
4.2.4	Копирование ячеек в табличном документе	57
4.2.5	Работа с формулами	58
4.2.6	Проверка данных	59
4.2.7	Встроенные объекты в табличном документе	63
4.2.7.1	Вставка изображения	64
4.2.7.2	Перечисление встроенных объектов	64
4.2.8	Работа с листами табличного документа	65
4.2.9	Работа со сводными таблицами	66
4.2.9.1	Создание сводной таблицы	67
4.2.9.2	Получение сводной таблицы	67
4.2.9.3	Получение диапазона исходных данных сводной таблицы	68
4.2.9.4	Получение диапазона размещения сводной таблицы	68
4.2.9.5	Получение флагов отображения общих итогов для строк и колонок	68
4.2.9.6	Получение заголовков сводной таблицы	69
4.2.9.7	Получение и применение фильтра для сводной таблицы	69
4.2.9.8	Получение полей из области фильтров	69
4.2.9.9	Получение полей из области значений	70
4.2.9.10	Получение полей из области строк	70
4.2.9.11	Получение полей из области колонок	70
4.2.9.12	Получение настроек отображения сводной таблицы	71
4.2.9.13	Обновление сводной таблицы	71
4.2.10	Работа с фильтрами	71
4.3	Встроенные объекты	73
4.3.1	Определение типа встроенных объектов	73
4.3.2	Работа со встроенными объектами	73
4.4	Поиск в документе	75
4.5	Работа с макрокомандами	76
4.6	Работа с именованными диапазонами	77
4.6.1	Доступ к именованным диапазонам	78

4.6.2	Получение коллекции именованных диапазонов	78
4.6.3	Добавление именованного диапазона	79
4.6.4	Получение параметров именованного диапазона	79
4.6.5	Переименование именованного диапазона	79
4.6.6	Удаление именованного диапазона	79
4.7	Работа со строками и столбцами таблиц	80
4.7.1	Группировка строк и колонок таблицы	80
4.7.2	Управление видимостью строк / колонок	80
4.8	Работа с ячейками таблиц	81
4.8.1	Доступ к ячейкам	81
4.8.2	Форматирование ячеек	84
4.8.3	Форматирование границ ячеек	85
4.8.4	Объединение и разделение ячеек таблицы	86
4.9	Защита документов	87
4.9.1	Защита диапазона текстового документа	87
4.9.2	Защита листа табличного документа	88
4.9.3	Защита структуры табличного документа	89
4.10	Локализация документов	90
5	Глобальные методы	92
5.1	Глобальный метод createSearch	92
5.2	Глобальный метод createScripting	92
5.3	Глобальный метод exportWorksheetToHtml	92
5.4	Методы условного форматирования	93
5.4.1	Глобальный метод castToAboveAverageConditionalFormat	93
5.4.2	Глобальный метод castToBinaryConditionalFormat	94
5.4.3	Глобальный метод castToColorScaleConditionalFormat	94
5.4.4	Глобальный метод castToDataBarConditionalFormat	95
5.4.5	Глобальный метод castToIconSetConditionalFormat	96
5.4.6	Глобальный метод castToNullaryConditionalFormat	96
5.4.7	Глобальный метод castToTextConditionalFormat	97
5.4.8	Глобальный метод castToTopBottomConditionalFormat	98

5.4.9	Глобальный метод castToUnaryConditionalFormat	98
5.4.10	Глобальный метод castToUniquenessConditionalFormat	99
5.4.11	Глобальный метод createAboveAverageConditionalFormatOperator	100
5.4.12	Глобальный метод createBinaryConditionalFormatOperator	100
5.4.13	Глобальный метод createColorScaleConditionalFormatOperator	101
5.4.14	Глобальный метод createDataBarConditionalFormatOperator	101
5.4.15	Глобальный метод createIconSetConditionalFormatOperator	101
5.4.16	Глобальный метод createNullaryConditionalFormatOperator	102
5.4.17	Глобальный метод createTextConditionalFormatOperator	102
5.4.18	Глобальный метод createTopBottomConditionalFormatOperator	103
5.4.19	Глобальный метод createUnaryConditionalFormatOperator	103
5.4.20	Глобальный метод createUniquenessConditionalFormatOperator	104
6	Справочник классов, структур и методов	105
6.1	Класс AboveAverageConditionalFormatOperator	105
6.1.1	Метод AboveAverageConditionalFormatOperator.getCondition	106
6.1.2	Метод AboveAverageConditionalFormatOperator.getStdDev	106
6.1.3	Метод AboveAverageConditionalFormatOperator.getType	106
6.2	Класс AbsoluteFrame	107
6.2.1	Метод AbsoluteFrame.getDimensions	107
6.2.2	Метод AbsoluteFrame.getTopLeft	108
6.2.3	Метод AbsoluteFrame.moveTo	108
6.2.4	Метод AbsoluteFrame.scale	108
6.2.5	Метод AbsoluteFrame.setDimensions	109
6.3	Класс AccountingCellFormatting	109
6.4	Перечисление Alignment	110
6.5	Класс Application	111
6.5.1	Метод Application.createDocument	111
6.5.2	Метод Application.getMessenger	111
6.5.3	Метод Application.loadDocument	112
6.6	Класс BinaryConditionalFormatOperator	112
6.6.1	Метод BinaryConditionalFormatOperator.getCondition	114

6.6.2	Метод BinaryConditionalFormatOperator.getFirstArgument	114
6.6.3	Метод BinaryConditionalFormatOperator.getSecondArgument	114
6.6.4	Метод BinaryConditionalFormatOperator.getType	114
6.7	Класс Block	115
6.7.1	Метод Block.getRange	115
6.7.2	Метод Block.getSection	115
6.7.3	Метод Block.remove	116
6.7.4	Методы toParagraph, toTable, toShape, toField	116
6.8	Класс Blocks	116
6.8.1	Метод Blocks.getBlock	117
6.8.2	Метод Blocks.GetEnumerator	117
6.8.3	Метод Blocks.getField	118
6.8.4	Метод Blocks.getFieldsEnumerator	118
6.8.5	Метод Blocks.getParagraph	118
6.8.6	Метод Blocks.getParagraphsEnumerator	118
6.8.7	Метод Blocks.getShape	119
6.8.8	Метод Blocks.getShapesEnumerator	119
6.8.9	Метод Blocks.getTable	119
6.8.10	Метод Blocks.getTablesEnumerator	120
6.9	Класс Bookmarks	120
6.9.1	Метод Bookmarks.getBookmarkRange	120
6.9.2	Метод Bookmarks.removeBookmark	120
6.10	Класс Borders	120
6.11	Перечисление CalculationMode	122
6.12	Перечисление CaseSensitive	122
6.13	Класс Cell	123
6.13.1	Метод Cell.checkDataValidation	123
6.13.2	Метод Cell.getBorders	124
6.13.3	Метод Cell.getCellProperties	124
6.13.4	Метод Cell.getColumnIndex	124
6.13.5	Метод Cell.getCurrentRegion	124

6.13.6	Метод Cell.getCustomFormat	125
6.13.7	Метод Cell.getDataValidation	125
6.13.8	Метод Cell.getFormat	125
6.13.9	Метод Cell.getFormattedValue	126
6.13.10	Метод Cell.getFormulaAsString	126
6.13.11	Метод Cell.getHyperlink	126
6.13.12	Метод Cell.getMergedRange	126
6.13.13	Метод Cell.getNote	127
6.13.14	Метод Cell.getParagraphProperties	127
6.13.15	Метод Cell.getPivotTable	128
6.13.16	Метод Cell.getProtectionProperties	128
6.13.17	Метод Cell.getRange	128
6.13.18	Метод Cell.getRawValue	129
6.13.19	Метод Cell.getResolvedBorders	129
6.13.20	Метод Cell.getRowIndex	129
6.13.21	Метод Cell.getTable	130
6.13.22	Метод Cell.getTextProperties	130
6.13.23	Метод Cell.isEmpty	131
6.13.24	Метод Cell.isContentEmpty	131
6.13.25	Метод Cell.isInMergedRange	132
6.13.26	Метод Cell.isPivotTableRoot	133
6.13.27	Метод Cell.isProtected	133
6.13.28	Метод Cell.removeNote	133
6.13.29	Метод Cell.setBool	133
6.13.30	Метод Cell.setBorders	134
6.13.31	Метод Cell.setCellProperties	134
6.13.32	Метод Cell.setContent	134
6.13.33	Метод Cell.setCustomFormat	134
6.13.34	Метод Cell.setFormat	135
6.13.35	Метод Cell.setFormattedValue	137
6.13.36	Метод Cell.setFormula	137

6.13.37	Метод Cell.setNote	137
6.13.38	Метод Cell.setNumber	138
6.13.39	Метод Cell.setParagraphProperties	138
6.13.40	Метод Cell.setProtectionProperties	138
6.13.41	Метод Cell.setText	139
6.13.42	Метод Cell.setTextProperties	139
6.13.43	Метод Cell.unmerge	140
6.14	Перечисление CellFormat	140
6.15	Класс CellPosition	142
6.15.1	Поле CellPosition.column	143
6.15.2	Поле CellPosition.row	143
6.15.3	Метод CellPosition.toString	143
6.15.4	CellPosition.__eq__	144
6.15.5	CellPosition.__ne__	144
6.16	Класс CellProperties	144
6.16.1	CellProperties.__eq__	146
6.16.2	CellProperties.__ne__	146
6.17	Класс CellProtectionProperties	147
6.18	Класс CellRange	148
6.18.1	Метод CellRange.autoFill	148
6.18.2	Метод CellRange.clearDataValidations	148
6.18.3	Метод CellRange.clearFormat	149
6.18.4	Метод CellRange.clearStyle	149
6.18.5	Метод CellRange.containsCell	149
6.18.6	Метод CellRange.copyInto	150
6.18.7	Метод CellRange.find	151
6.18.8	Метод CellRange.getAddress	151
6.18.9	Метод CellRange.getBeginColumn	152
6.18.10	Метод CellRange.getBeginRow	152
6.18.11	Метод CellRange.getCellProperties	153
6.18.12	Метод CellRange.getEnumerator	153

6.18.13	Метод <code>CellRange.getLastColumn</code>	153
6.18.14	Метод <code>CellRange.getLastRow</code>	153
6.18.15	Метод <code>CellRange.getParagraphProperties</code>	154
6.18.16	Метод <code>CellRange.getProtectionProperties</code>	154
6.18.17	Метод <code>CellRange.getTable</code>	155
6.18.18	Метод <code>CellRange.getTableRange</code>	155
6.18.19	Метод <code>CellRange.getTextProperties</code>	155
6.18.20	Метод <code>CellRange.insert</code>	156
6.18.21	Метод <code>CellRange.insertCurrentDateTime</code>	157
6.18.22	Метод <code>CellRange.intersect</code>	157
6.18.23	Метод <code>CellRange.isEmpty</code>	157
6.18.24	Метод <code>CellRange.isProtected</code>	158
6.18.25	Метод <code>CellRange.merge</code>	159
6.18.26	Метод <code>CellRange.moveInto</code>	159
6.18.27	Метод <code>CellRange.remove</code>	160
6.18.28	Метод <code>CellRange.removeContent</code>	161
6.18.29	Метод <code>CellRange.setArrayFormula</code>	161
6.18.30	Метод <code>CellRange.setBorders</code>	161
6.18.31	Метод <code>CellRange.setCellProperties</code>	162
6.18.32	Метод <code>CellRange.setDataValidation</code>	162
6.18.33	Метод <code>CellRange.setParagraphProperties</code>	163
6.18.34	Метод <code>CellRange.setProtectionProperties</code>	163
6.18.35	Метод <code>CellRange.setTextProperties</code>	164
6.18.36	Метод <code>CellRange.sort</code>	165
6.18.37	Метод <code>CellRange.textToColumns</code>	165
6.19	Перечисление <code>CellRangeAddressFormat</code>	166
6.20	Класс <code>CellRangeAddressSettings</code>	167
6.21	Класс <code>CellRangePosition</code>	168
6.21.1	Метод <code>CellRangePosition.toString</code>	169
6.21.2	<code>CellRangePosition.__eq__</code>	169

6.21.3	CellRangePosition.__ne__	169
6.22	Перечисление CellShiftAxis	170
6.23	Класс Chart	170
6.23.1	Метод Chart.applySettings	171
6.23.2	Метод Chart.getChartLabels	172
6.23.3	Метод Chart.getDirectionType	172
6.23.4	Метод Chart.getFrame	172
6.23.5	Метод Chart.getRange	172
6.23.6	Метод Chart.getRangeAsString	173
6.23.7	Метод Chart.getRangesCount	173
6.23.8	Метод Chart.getTitle	173
6.23.9	Метод Chart.getType	173
6.23.10	Метод Chart.is3D	174
6.23.11	Метод Chart.isEmpty	174
6.23.12	Метод Chart.isSolidRange	174
6.23.13	Метод Chart.setRange	174
6.23.14	Метод Chart.setRect	175
6.23.15	Метод Chart.setType	175
6.24	Перечисление ChartLabelsDetectionMode	175
6.25	Класс ChartLabelsInfo	176
6.26	Класс ChartRangeInfo	177
6.27	Перечисление ChartRangeType	177
6.28	Класс Charts	178
6.28.1	Метод Charts.getChart	179
6.28.2	Метод Charts.getChartIndexByDrawingIndex	179
6.28.3	Метод Charts.getChartsCount	180
6.29	Перечисление ChartSeriesDirectionType	180
6.30	Перечисление ChartType	180
6.31	Класс CheckBoxControl	181
6.32	Класс Color	182
6.32.1	Метод Color.getRGBAColor	182

6.32.2	Метод Color.getThemeColorID	182
6.32.3	Метод Color.getTransforms	183
6.32.4	Метод Color.setTransforms	183
6.32.5	Метод Color.__eq__	183
6.32.6	Метод Color.__ne__	183
6.33	Класс ColorRGBA	184
6.33.1	ColorRGBA.__eq__	184
6.33.2	ColorRGBA.__ne__	185
6.34	Класс ColorScaleConditionalFormatOperator	185
6.34.1	Метод ColorScaleConditionalFormatOperator.getEntries	187
6.34.2	Метод ColorScaleConditionalFormatOperator.getType	187
6.34.3	Метод ColorScaleConditionalFormatOperator.setEntries	187
6.35	Класс ColorTransforms	188
6.35.1	Метод ColorTransforms.apply	188
6.36	Класс Comment	188
6.36.1	Метод Comment.getAudioUrl	188
6.36.2	Метод Comment.getInfo	189
6.36.3	Метод Comment.getRange	189
6.36.4	Метод Comment.getReplies	189
6.36.5	Метод Comment.getText	190
6.36.6	Метод Comment.isResolved	190
6.37	Класс Comments	190
6.37.1	Метод Comments.getEnumerator	191
6.38	Перечисление ConditionalFormatAboveAverageCondition	191
6.39	Перечисление ConditionalFormatBinaryCondition	191
6.40	Класс ConditionalFormatCellStyle	192
6.41	Класс ConditionalFormatColorScaleEntries	192
6.41.1	Метод ConditionalFormatColorScaleEntries.addEntry	193
6.41.2	Метод ConditionalFormatColorScaleEntries.getEntriesCount	193
6.41.3	Метод ConditionalFormatColorScaleEntries.getEntry	193
6.41.4	Метод ConditionalFormatColorScaleEntries.setEntry	193

6.42	Класс ConditionalFormatColorScaleEntry	194
6.42.1	Метод ConditionalFormatColorScaleEntry.getColor	194
6.42.2	Метод ConditionalFormatColorScaleEntry.getValueObject	195
6.42.3	Метод ConditionalFormatColorScaleEntry.setColor	195
6.42.4	Метод ConditionalFormatColorScaleEntry.setValueObject	195
6.43	Перечисление ConditionalFormatDataBarAxisPosition	195
6.44	Перечисление ConditionalFormatDataBarDirection	196
6.45	Перечисление ConditionalFormatDataBarFillType	196
6.46	Класс ConditionalFormatDataBarParams	197
6.47	Класс ConditionalFormatDocumentRules	198
6.47.1	Метод ConditionalFormatDocumentRules.removeAllRules	199
6.48	Перечисление ConditionalFormatIconSet	199
6.49	Класс ConditionalFormatIconSetEntries	200
6.49.1	Метод ConditionalFormatIconSetEntries.addEntry	201
6.49.2	Метод ConditionalFormatIconSetEntries.getEntriesCount	201
6.49.3	Метод ConditionalFormatIconSetEntries.getEntry	201
6.49.4	Метод ConditionalFormatIconSetEntries.setEntry	201
6.50	Класс ConditionalFormatIconSetEntry	202
6.50.1	Метод ConditionalFormatIconSetEntry.getIconIndex	202
6.50.2	Метод ConditionalFormatIconSetEntry.getIconSet	203
6.50.3	Метод ConditionalFormatIconSetEntry.getValueObject	203
6.50.4	Метод ConditionalFormatIconSetEntry.setIconIndex	203
6.50.5	Метод ConditionalFormatIconSetEntry.setIconSet	203
6.50.6	Метод ConditionalFormatIconSetEntry.setValueObject	204
6.51	Перечисление ConditionalFormatNullaryCondition	204
6.52	Класс ConditionalFormatOperator	205
6.52.1	Метод ConditionalFormatOperator.getType	205
6.53	Перечисление ConditionalFormatOperatorType	205
6.54	Класс ConditionalFormatRule	206
6.54.1	Метод ConditionalFormatRule.getOperator	207
6.54.2	Метод ConditionalFormatRule.getRanges	207

6.54.3	Метод ConditionalFormatRule.getStopCalculations	207
6.54.4	Метод ConditionalFormatRule.getStyle	208
6.54.5	Метод ConditionalFormatRule.getUUID	208
6.54.6	Метод ConditionalFormatRule.setOperator	208
6.54.7	Метод ConditionalFormatRule.setRange	209
6.54.8	Метод ConditionalFormatRule.setRanges	209
6.54.9	Метод ConditionalFormatRule.setStopCalculations	209
6.54.10	Метод ConditionalFormatRule.setStyle	210
6.54.11	Метод ConditionalFormatRule.setUUID	210
6.55	Класс ConditionalFormatRuleProxy	210
6.55.1	Метод ConditionalFormatRuleProxy.getData	211
6.55.2	Метод ConditionalFormatRuleProxy.getIndex	211
6.55.3	Метод ConditionalFormatRuleProxy.getOperator	211
6.55.4	Метод ConditionalFormatRuleProxy.getRanges	211
6.55.5	Метод ConditionalFormatRuleProxy.getStopCalculations	212
6.55.6	Метод ConditionalFormatRuleProxy.getStyle	212
6.55.7	Метод ConditionalFormatRuleProxy.getTableName	212
6.55.8	Метод ConditionalFormatRuleProxy.getType	212
6.55.9	Метод ConditionalFormatRuleProxy.remove	213
6.55.10	Метод ConditionalFormatRuleProxy.setOperator	213
6.55.11	Метод ConditionalFormatRuleProxy.setRange	213
6.55.12	Метод ConditionalFormatRuleProxy.setRanges	213
6.55.13	Метод ConditionalFormatRuleProxy.setStopCalculations	214
6.55.14	Метод ConditionalFormatRuleProxy.setStyle	214
6.56	Класс ConditionalFormatTableRules	214
6.56.1	Метод ConditionalFormatTableRules.addRule	215
6.56.2	Метод ConditionalFormatTableRules.getRule	215
6.56.3	Метод ConditionalFormatTableRules.getRuleCount	215
6.56.4	Метод ConditionalFormatTableRules.removeAllRules	216
6.56.5	Метод ConditionalFormatTableRules.removeRulesFromRange	216
6.57	Перечисление ConditionalFormatTextCondition	216

6.58	Перечисление ConditionalFormatTopBottomCondition	217
6.59	Перечисление ConditionalFormatUnaryCondition	217
6.60	Перечисление ConditionalFormatUniquenessCondition	218
6.61	Класс ConditionalFormatValueObject	219
6.61.1	Метод ConditionalFormatValueObject.getType	219
6.61.2	Метод ConditionalFormatValueObject.getValue	219
6.61.3	Метод ConditionalFormatValueObject.isStrictCompare	220
6.61.4	Метод ConditionalFormatValueObject.setStrictCompare	220
6.61.5	Метод ConditionalFormatValueObject.setType	220
6.61.6	Метод ConditionalFormatValueObject.setValue	220
6.62	Перечисление ConditionalFormatValueObjectType	221
6.63	Класс ConditionalTableFilter	222
6.63.1	Метод ConditionalTableFilter.setAndOperation	222
6.63.2	Методы добавления условий	223
6.64	Класс Connection	224
6.65	Перечисление ContainingTableFilter	224
6.66	Класс ContentControl	225
6.66.1	Метод ContentControl.canEdit	225
6.66.2	Метод ContentControl.getTitle	225
6.66.3	Методы toCheckBox, toInputField, toDatePicker, toDropList	225
6.67	Класс ContentControls	225
6.68	Класс CurrencyCellFormatting	226
6.69	Перечисление CurrencySignPlacement	227
6.70	Класс DataBarConditionalFormatOperator	227
6.70.1	Метод DataBarConditionalFormatOperator.getAxisColor	229
6.70.2	Метод DataBarConditionalFormatOperator.getAxisPosition	229
6.70.3	Метод DataBarConditionalFormatOperator.getBarFill	229
6.70.4	Метод DataBarConditionalFormatOperator.getBorderColor	230
6.70.5	Метод DataBarConditionalFormatOperator.getDirection	230
6.70.6	Метод DataBarConditionalFormatOperator.getFillType	230
6.70.7	Метод DataBarConditionalFormatOperator.getLowerThreshold	230

6.70.8	Метод <code>DataBarConditionalFormatOperator.getMaxLength</code>	231
6.70.9	Метод <code>DataBarConditionalFormatOperator.getMinLength</code>	231
6.70.10	Метод <code>DataBarConditionalFormatOperator.getNegativeBarFill</code>	231
6.70.11	Метод <code>DataBarConditionalFormatOperator.getNegativeBorderColor</code>	231
6.70.12	Метод <code>DataBarConditionalFormatOperator.getType</code>	232
6.70.13	Метод <code>DataBarConditionalFormatOperator.getUpperThreshold</code>	232
6.70.14	Метод <code>DataBarConditionalFormatOperator.getValueVisibility</code>	232
6.70.15	Метод <code>DataBarConditionalFormatOperator.setAxisColor</code>	232
6.70.16	Метод <code>DataBarConditionalFormatOperator.setAxisPosition</code>	233
6.70.17	Метод <code>DataBarConditionalFormatOperator.setBarFill</code>	233
6.70.18	Метод <code>DataBarConditionalFormatOperator.setBorderColor</code>	233
6.70.19	Метод <code>DataBarConditionalFormatOperator.setDirection</code>	233
6.70.20	Метод <code>DataBarConditionalFormatOperator.setFillType</code>	234
6.70.21	Метод <code>DataBarConditionalFormatOperator.setLowerThreshold</code>	234
6.70.22	Метод <code>DataBarConditionalFormatOperator.setMaxLength</code>	234
6.70.23	Метод <code>DataBarConditionalFormatOperator.setMinLength</code>	234
6.70.24	Метод <code>DataBarConditionalFormatOperator.setNegativeBarFill</code>	235
6.70.25	Метод <code>DataBarConditionalFormatOperator.setNegativeBorderColor</code>	235
6.70.26	Метод <code>DataBarConditionalFormatOperator.setUpperThreshold</code>	235
6.70.27	Метод <code>DataBarConditionalFormatOperator.setValueVisibility</code>	235
6.71	Класс <code>DataValidation</code>	236
6.71.1	Метод <code>DataValidation.clear</code>	236
6.71.2	Метод <code>DataValidation.getAllowBlank</code>	236
6.71.3	Метод <code>DataValidation.getErrorMessage</code>	236
6.71.4	Метод <code>DataValidation.getErrorStyle</code>	237
6.71.5	Метод <code>DataValidation.getErrorTitle</code>	237
6.71.6	Метод <code>DataValidation.getFormula1</code>	237
6.71.7	Метод <code>DataValidation.getFormula2</code>	238
6.71.8	Метод <code>DataValidation.getOperator</code>	238
6.71.9	Метод <code>DataValidation.getPrompt</code>	238
6.71.10	Метод <code>DataValidation.getPromptTitle</code>	239

6.71.11	Метод <code>DataValidation.getShowDropDown</code>	239
6.71.12	Метод <code>DataValidation.getShowErrorMessage</code>	239
6.71.13	Метод <code>DataValidation.getShowInputMessage</code>	239
6.71.14	Метод <code>DataValidation.getType</code>	240
6.71.15	Метод <code>DataValidation.isEmpty</code>	240
6.71.16	Метод <code>DataValidation.setAllowBlank</code>	240
6.71.17	Метод <code>DataValidation.setErrorMessage</code>	241
6.71.18	Метод <code>DataValidation.setErrorStyle</code>	241
6.71.19	Метод <code>DataValidation.setErrorTitle</code>	242
6.71.20	Метод <code>DataValidation.setFormula1</code>	242
6.71.21	Метод <code>DataValidation.setFormula2</code>	243
6.71.22	Метод <code>DataValidation.setOperator</code>	243
6.71.23	Метод <code>DataValidation.setPrompt</code>	244
6.71.24	Метод <code>DataValidation.setPromptTitle</code>	244
6.71.25	Метод <code>DataValidation.setShowDropDown</code>	245
6.71.26	Метод <code>DataValidation.setShowErrorMessage</code>	245
6.71.27	Метод <code>DataValidation.setShowInputMessage</code>	246
6.71.28	Метод <code>DataValidation.setType</code>	246
6.72	Перечисление <code>DataValidationErrorStyle</code>	247
6.73	Перечисление <code>DataValidationOperator</code>	247
6.74	Класс <code>DataValidationResult</code>	248
6.74.1	Метод <code>DataValidationResult.getDataValidation</code>	248
6.74.2	Метод <code>DataValidationResult.isValid</code>	249
6.75	Перечисление <code>DataValidationType</code>	249
6.76	Перечисление <code>DatePatterns</code>	250
6.77	Класс <code>DatePickerControl</code>	250
6.78	Класс <code>DateTime</code>	251
6.78.1	<code>DateTime.__eq__</code>	251
6.78.2	<code>DateTime.__ne__</code>	251
6.79	Класс <code>DateTimeCellFormatting</code>	252
6.80	Перечисление <code>DateTimeFormat</code>	252

6.81 Класс Document	253
6.81.1 Метод Document.areMirroredMarginsEnabled	253
6.81.2 Метод Document.calculateAllFormulas	253
6.81.3 Метод Document.calculateOutdatedFormulas	253
6.81.4 Метод Document.exportAs	254
6.81.5 Метод Document.getAbsoluteFilePath	254
6.81.6 Метод Document.getBlocks	255
6.81.7 Метод Document.getBookmarks	255
6.81.8 Метод Document.getCalculationMode	255
6.81.9 Метод Document.getComments	256
6.81.10 Метод Document.getConditionalFormatRules	256
6.81.11 Метод Document.getContentControls	256
6.81.12 Метод Document.getFormulaType	257
6.81.13 Метод Document.getNamedExpressions	257
6.81.14 Метод Document.getPivotTablesManager	257
6.81.15 Метод Document.getRange	257
6.81.16 Метод Document.getScripts	257
6.81.17 Метод Document.getSections	258
6.81.18 Метод Document.getSectionsEnumerator	258
6.81.19 Метод Document.getTextStyles	258
6.81.20 Метод Document.isCalculatedOnSave	259
6.81.21 Метод Document.isChangesTrackingEnabled	259
6.81.22 Метод Document.isStructureProtected	259
6.81.23 Метод Document.merge	259
6.81.24 Метод Document.removeStructureProtection	260
6.81.25 Метод Document.saveAs	260
6.81.26 Метод Document.setCalculatedOnSave	261
6.81.27 Метод Document.setCalculationMode	261
6.81.28 Метод Document.setChangesTrackingEnabled	261
6.81.29 Метод Document.setFormulaType	262
6.81.30 Метод Document.setMirroredMarginsEnabled	262

6.81.31	Метод Document.setPageOrientation	262
6.81.32	Метод Document.setPageProperties	262
6.81.33	Метод Document.setStructureProtection	262
6.82	Перечисление DocumentFormat	263
6.83	Класс DocumentSettings	263
6.84	Перечисление DocumentType	264
6.85	Класс DropDownListControl	264
6.86	Класс DSVSettings	265
6.86.1	Метод DSVSettings.__eq__	265
6.86.2	Метод DSVSettings.__ne__	266
6.87	Перечисление Encoding	266
6.88	Перечисление ExportFormat	267
6.89	Класс Field	267
6.90	Класс Fill	267
6.90.1	Метод Fill.getColor	268
6.90.2	Метод Fill.getUrl	268
6.90.3	Метод Fill.isNoFill	268
6.91	Класс FiltersRange	268
6.91.1	Метод FiltersRange.clear	268
6.91.2	Метод FiltersRange.eraseFilters	268
6.91.3	Метод FiltersRange.getCellRange	269
6.91.4	Метод FiltersRange.getFilters	269
6.91.5	Метод FiltersRange.setFilters	269
6.92	Класс FootnoteEndnote	270
6.92.1	Метод FootnoteEndnote.getPosition	270
6.92.2	Метод FootnoteEndnote.getRange	271
6.92.3	Метод FootnoteEndnote.getType	271
6.93	Перечисление FootnoteEndnoteType	271
6.94	Класс FootnotesEndnotes	271
6.95	Перечисление FormulaType	272
6.96	Класс FractionCellFormatting	272

6.97 Класс Frame	273
6.98 Класс FrozenRangePosition	274
6.98.1 Конструкторы	274
6.98.2 Метод FrozenRangePosition.create	274
6.98.3 Метод FrozenRangePosition.createFrozenArea	275
6.98.4 Метод FrozenRangePosition.createFrozenCols	275
6.98.5 Метод FrozenRangePosition.createFrozenRows	275
6.98.6 Метод FrozenRangePosition.isArea	276
6.98.7 Метод FrozenRangePosition.isCols	276
6.98.8 Метод FrozenRangePosition.isRows	276
6.98.9 Метод FrozenRangePosition.isRowsCols	276
6.99 Класс HeaderFooter	276
6.99.1 Метод HeaderFooter.getBlocks	277
6.99.2 Метод HeaderFooter.getRange	277
6.99.3 Метод HeaderFooter.getType	277
6.100 Перечисление HeaderFooterType	277
6.101 Класс HeadersFooters	278
6.101.1 Метод HeadersFooters.getEnumerator	278
6.102 Перечисление HorizontalAnchorAlignment	279
6.103 Перечисление HorizontalRelativeTo	279
6.104 Класс HorizontalTextAnchoredPosition	280
6.104.1 HorizontalTextAnchoredPosition.__eq__	280
6.104.2 HorizontalTextAnchoredPosition.__ne__	281
6.105 Класс HtmlFragments	282
6.106 Класс Hyperlink	282
6.106.1 Hyperlink.__eq__	282
6.106.2 Hyperlink.__ne__	283
6.107 Класс IconSetConditionalFormatOperator	283
6.107.1 Метод IconSetConditionalFormatOperator.getEntries	285
6.107.2 Метод IconSetConditionalFormatOperator.getType	285
6.107.3 Метод IconSetConditionalFormatOperator.isValueShown	285

6.107.4	Метод IconSetConditionalFormatOperator.setEntries	286
6.107.5	Метод IconSetConditionalFormatOperator.setValueShown	286
6.108	Класс Image	286
6.108.1	Метод Image.getFrame	286
6.108.2	Метод Image.remove	287
6.108.3	Метод Image.replaceURL	287
6.109	Класс Images	287
6.109.1	Метод Images.getEnumerator	288
6.110	Класс InlineFrame	288
6.110.1	Метод InlineFrame.getDimensions	289
6.110.2	Метод InlineFrame.getPosition	289
6.110.3	Метод InlineFrame.getWrapType	289
6.110.4	Метод InlineFrame.setDimensions	290
6.110.5	Метод InlineFrame.setPosition	290
6.110.6	Метод InlineFrame.setWrapType	292
6.111	Класс InputFieldControl	292
6.112	Класс Insets	292
6.112.1	Insets.__eq__	293
6.112.2	Insets.__ne__	293
6.113	Класс LineEndingProperties	294
6.113.1	LineEndingProperties.__eq__	295
6.113.2	LineEndingProperties.__ne__	295
6.114	Перечисление LineEndingStyle	296
6.115	Класс LineProperties	297
6.115.1	LineProperties.__eq__	298
6.115.2	LineProperties.__ne__	298
6.116	Класс LineSpacing	299
6.116.1	LineSpacing.__eq__	299
6.116.2	LineSpacing.__ne__	300
6.117	Перечисление LineSpacingRule	300
6.118	Перечисление LineStyle	302

6.119	Перечисление ListSchema	303
6.120	Класс LoadDocumentSettings	305
6.121	Класс LocaleInfo	305
6.122	Класс MediaObject	306
6.122.1	Метод MediaObject.getFrame	306
6.122.2	Метод MediaObject.toChart	307
6.122.3	Метод MediaObject.toImage	307
6.123	Класс MediaObjects	307
6.123.1	Метод MediaObjects.getEnumerator	308
6.124	Класс Message	308
6.124.1	Перечисление Message.Severity	308
6.124.2	Метод Message.getSeverity	309
6.124.3	Метод Message.getText	309
6.124.4	Метод Message.makeError	309
6.124.5	Метод Message.makeInfo	309
6.124.6	Метод Message.makeWarning	309
6.125	Класс Messenger	309
6.125.1	Метод Messenger.notify	310
6.125.2	Метод Messenger.subscribe	310
6.126	Класс NamedExpression	310
6.126.1	Метод NamedExpression.getCellRange	310
6.126.2	Метод NamedExpression.getExpression	310
6.126.3	Метод NamedExpression.getName	311
6.126.4	Метод NamedExpression.setName	311
6.127	Класс NamedExpressions	311
6.127.1	Метод NamedExpressions.addExpression	311
6.127.2	Метод NamedExpressions.get	311
6.127.3	Метод NamedExpressions.getEnumerator	312
6.127.4	Метод NamedExpressions.removeExpression	312
6.128	Класс NullaryConditionalFormatOperator	312
6.128.1	Метод NullaryConditionalFormatOperator.getCondition	313

6.128.2	Метод NullaryConditionalFormatOperator.getType	314
6.129	Класс NumberCellFormatting	314
6.130	Перечисление PageFieldOrder	315
6.131	Класс PageNumbers	315
6.131.1	Метод PageNumbers.contains	316
6.131.2	Метод PageNumbers.getLast	316
6.131.3	Метод PageNumbers.__eq__	316
6.131.4	Метод PageNumbers.__ne__	316
6.132	Перечисление PageOrientation	317
6.133	Перечисление PageParity	317
6.134	Класс PageProperties	318
6.134.1	PageProperties.__eq__	318
6.134.2	PageProperties.__ne__	319
6.135	Класс Paragraph	319
6.135.1	Метод Paragraph.decreaseListLevel	320
6.135.2	Метод Paragraph.getListLevel	320
6.135.3	Метод Paragraph.getListSchema	320
6.135.4	Метод Paragraph.getParagraphProperties	321
6.135.5	Метод Paragraph.getResolvedStyle	321
6.135.6	Метод Paragraph.getStyle	322
6.135.7	Метод Paragraph.increaseListLevel	322
6.135.8	Метод Paragraph.setListLevel	322
6.135.9	Метод Paragraph.setListSchema	323
6.135.10	Метод Paragraph.setParagraphProperties	323
6.135.11	Метод Paragraph.setStyle	324
6.136	Класс ParagraphProperties	324
6.136.1	ParagraphProperties.__eq__	327
6.136.2	ParagraphProperties.__ne__	328
6.137	Класс Paragraphs	328
6.137.1	Метод Paragraphs.decreaseListLevel	329
6.137.2	Метод Paragraphs.getEnumerator	329

6.137.3	Метод Paragraphs.increaseListLevel	329
6.137.4	Метод Paragraphs.setListLevel	330
6.137.5	Метод Paragraphs.setListSchema	330
6.138	Класс PercentageCellFormatting	330
6.139	Класс PivotTable	331
6.139.1	Метод PivotTable.areAllFiltersInDefaultState	331
6.139.2	Метод PivotTable.changeSourceRange	331
6.139.3	Метод PivotTable.createPivotTableEditor	331
6.139.4	Метод PivotTable.getColumnFields	332
6.139.5	Метод PivotTable.getConditionalLabelFilter	332
6.139.6	Метод PivotTable.getConditionalValueFilter	333
6.139.7	Метод PivotTable.getFieldCategories	333
6.139.8	Метод PivotTable.getFieldItems	334
6.139.9	Метод PivotTable.getFieldItemsByName	334
6.139.10	Метод PivotTable.getFieldsList	334
6.139.11	Метод PivotTable.getFilter	335
6.139.12	Метод PivotTable.getFilters	335
6.139.13	Метод PivotTable.getPageFields	335
6.139.14	Метод PivotTable.getPivotRange	335
6.139.15	Метод PivotTable.getPivotTableCaptions	336
6.139.16	Метод PivotTable.getPivotTableLayoutSettings	336
6.139.17	Метод PivotTable.getRowFields	336
6.139.18	Метод PivotTable.getSortingParams	337
6.139.19	Метод PivotTable.getSourceRange	337
6.139.20	Метод PivotTable.getSourceRangeAddress	337
6.139.21	Метод PivotTable.getValueFields	338
6.139.22	Метод PivotTable.isColumnGrandTotalEnabled	338
6.139.23	Метод PivotTable.isRowGrandTotalEnabled	338
6.139.24	Метод PivotTable.remove	339
6.139.25	Метод PivotTable.update	339
6.140	Класс PivotTableCaptions	339

6.141	Класс PivotTableCategoryField	340
6.142	Класс PivotTableConditionalLabelFilter	340
6.142.1	Метод PivotTableConditionalLabelFilter.getFieldName	341
6.142.2	Метод PivotTableConditionalLabelFilter.getOperation	341
6.142.3	Метод PivotTableConditionalLabelFilter.isInDefaultState	341
6.142.4	Метод PivotTableConditionalLabelFilter.reset	341
6.142.5	Метод PivotTableConditionalLabelFilter.setOperation	341
6.143	Класс PivotTableConditionalLabelFilterOperation	342
6.144	Перечисление PivotTableConditionalLabelFilterOperationType	343
6.145	Класс PivotTableConditionalValueFilter	343
6.145.1	Метод PivotTableConditionalValueFilter.getDefaultOperation	343
6.145.2	Метод PivotTableConditionalValueFilter.getExpectedOperandType	344
6.145.3	Метод PivotTableConditionalValueFilter.getFieldName	345
6.145.4	Метод PivotTableConditionalValueFilter.getOperation	345
6.145.5	Метод PivotTableConditionalValueFilter.isInDefaultState	345
6.145.6	Метод PivotTableConditionalValueFilter.reset	345
6.145.7	Метод PivotTableConditionalValueFilter.setOperation	345
6.146	Перечисление PivotTableConditionalValueFilterOperandType	346
6.147	Класс PivotTableConditionalValueFilterOperation	346
6.148	Перечисление PivotTableConditionalValueFilterOperationType	347
6.149	Класс PivotTableEditor	348
6.149.1	Метод PivotTableEditor.addField	348
6.149.2	Метод PivotTableEditor.apply	349
6.149.3	Метод PivotTableEditor.disableField	349
6.149.4	Метод PivotTableEditor.enableField	349
6.149.5	Метод PivotTableEditor.moveField	349
6.149.6	Метод PivotTableEditor.removeField	350
6.149.7	Метод PivotTableEditor.reorderField	350
6.149.8	Метод PivotTableEditor.resetAllFilters	350
6.149.9	Метод PivotTableEditor.setCaptions	351
6.149.10	Метод PivotTableEditor.setFilter	351

6.149.11	Метод PivotTableEditor.setFilters	352
6.149.12	Метод PivotTableEditor.setGrandTotalSettings	352
6.149.13	Метод PivotTableEditor.setLayoutSettings	352
6.149.14	Метод PivotTableEditor.setSortingByLabel	353
6.149.15	Метод PivotTableEditor.setSortingByValue	353
6.149.16	Метод PivotTableEditor.setSummarizeFunction	355
6.150	Класс PivotTableField	355
6.151	Класс PivotTableFieldCategories	355
6.151.1	Метод PivotTableFieldCategories.getEnumerator	355
6.152	Перечисление PivotTableFieldCategory	356
6.153	Класс PivotTableFieldProperties	356
6.154	Класс PivotTableFilter	357
6.154.1	Метод PivotTableFilter.getCount	357
6.154.2	Метод PivotTableFilter.getFieldName	357
6.154.3	Метод PivotTableFilter.getName	358
6.154.4	Метод PivotTableFilter.isHidden	358
6.154.5	Метод PivotTableFilter.isInDefaultState	358
6.154.6	Метод PivotTableFilter.reset	358
6.154.7	Метод PivotTableFilter.setHidden	359
6.155	Класс PivotTableFilters	359
6.155.1	Метод PivotTableFilters.getEnumerator	359
6.156	Перечисление PivotTableFunction	360
6.157	Класс PivotTableItem	360
6.157.1	Метод PivotTableItem.getAlias	361
6.157.2	Метод PivotTableItem.getItemType	361
6.157.3	Метод PivotTableItem.getName	361
6.157.4	Метод PivotTableItem.isCollapsed	361
6.158	Класс PivotTableItemForCategory	361
6.159	Класс PivotTableItems	362
6.159.1	Метод PivotTableItems.getEnumerator	362
6.160	Перечисление PivotTableItemType	363

6.161	Класс PivotTableLayoutSettings	364
6.162	Класс PivotTablePageField	365
6.163	Перечисление PivotTableReportLayout	365
6.164	Класс PivotTableSlicePath	366
6.165	Класс PivotTablesManager	366
6.165.1	Метод PivotTablesManager.create	366
6.166	Перечисление PivotTableSortingOrder	366
6.167	Класс PivotTableSortingParams	367
6.168	Перечисление PivotTableSortingType	367
6.169	Перечисление PivotTableUpdateResult	368
6.170	Класс PivotTableValueField	369
6.171	Класс PointU	370
6.171.1	PointU.toString	370
6.172	Класс Position	370
6.172.1	Метод Position.compare	370
6.172.2	Метод Position.getBefore	371
6.172.3	Метод Position.getCell	372
6.172.4	Метод Position.getCurrentRange	372
6.172.5	Метод Position.getNextPosition	373
6.172.6	Метод Position.getNextRange	374
6.172.7	Метод Position.getParagraph	375
6.172.8	Метод Position.getPreviousPosition	375
6.172.9	Метод Position.getPreviousRange	376
6.172.10	Метод Position.insertBookmark	377
6.172.11	Метод Position.insertEndnote	377
6.172.12	Метод Position.insertFootnote	377
6.172.13	Метод Position.insertHyperlink	378
6.172.14	Метод Position.insertImage	378
6.172.15	Метод Position.insertLineBreak	378
6.172.16	Метод Position.insertPageBreak	379
6.172.17	Метод Position.insertSectionBreak	379

6.172.18	Метод Position.insertTable	379
6.172.19	Метод Position.insertText	380
6.172.20	Метод Position.removeBackward	380
6.172.21	Метод Position.removeForward	380
6.172.22	Position.__eq__	380
6.172.23	Position.__ne__	381
6.173	Перечисление PredefinedTextStyle	381
6.174	PresentationExportSettings	382
6.175	Класс PrintingScope	383
6.175.1	Метод PrintingScope.getCellRange	383
6.175.2	Метод PrintingScope.usePrintArea	383
6.176	Перечисление PrintingScope.Type	384
6.177	Класс PrintTitles	384
6.177.1	Метод PrintTitles.create	385
6.177.2	Метод PrintTitles.createPrintTitlesCols	385
6.177.3	Метод PrintTitles.createPrintTitlesRows	386
6.177.4	Метод PrintTitles.isCols	386
6.177.5	Метод PrintTitles.isRows	386
6.177.6	Метод PrintTitles.isRowsCols	386
6.178	Класс Range	387
6.178.1	Конструктор Range	388
6.178.2	Метод Range.extractText	389
6.178.3	Метод Range.getBegin	389
6.178.4	Метод Range.getBlocksEnumerator	390
6.178.5	Метод Range.getComments	390
6.178.6	Метод Range.getContentEnd	390
6.178.7	Метод Range.getEnd	391
6.178.8	Метод Range.getFootnotesEndnotes	392
6.178.9	Метод Range.getImages	392
6.178.10	Метод Range.getInlineObjects	392
6.178.11	Метод Range.getParagraphs	393

6.178.12	Метод Range.getTextProperties	393
6.178.13	Метод Range.getTrackedChangesEnumerator	394
6.178.14	Метод Range.isContentLocked	394
6.178.15	Метод Range.lockContent	394
6.178.16	Метод Range.removeContent	395
6.178.17	Метод Range.replaceText	395
6.178.18	Метод Range.setHyperlink	396
6.178.19	Метод Range.setTextProperties	396
6.178.20	Метод Range.unlockContent	397
6.178.21	Метод Range.__eq__	397
6.178.22	Метод Range.__ne__	397
6.179	Класс RangeBorders	398
6.180	Класс RectU	398
6.180.1	RectU.toString	398
6.181	Класс SaveDocumentSettings	399
6.182	Перечисление ScaleFrom	399
6.183	Класс ScientificCellFormatting	399
6.184	Класс Script	400
6.184.1	Метод Script.getBody	400
6.184.2	Метод Script.getName	401
6.184.3	Метод Script.setBody	401
6.184.4	Метод Script.setName	401
6.185	Класс Scripting	402
6.185.1	Метод Scripting.runScript	402
6.186	Перечисление ScriptPosition	402
6.187	Класс Scripts	403
6.187.1	Метод Scripts.getEnumerator	403
6.187.2	Метод Scripts.getScript	403
6.187.3	Метод Scripts.removeScript	404
6.187.4	Метод Scripts.setScript	404
6.188	Класс Search	404

6.188.1	Метод Search.findText	404
6.189	Класс Section	405
6.189.1	Метод Section.getFooters	406
6.189.2	Метод Section.getHeaders	406
6.189.3	Метод Section.getPageOrientation	406
6.189.4	Метод Section.getPageProperties	406
6.189.5	Метод Section.getRange	407
6.189.6	Метод Section.setPageOrientation	407
6.189.7	Метод Section.setPageProperties	407
6.190	Перечисление SectionBreakType	407
6.191	Класс Sections	408
6.191.1	Метод Sections.getEnumerator	408
6.192	Класс Shape	408
6.192.1	Метод Shape.getShapeProperties	408
6.192.2	Метод Shape.setShapeProperties	409
6.193	Класс ShapeProperties	409
6.194	Перечисление ShapeTextLayout	410
6.195	Класс SizeU	410
6.195.1	Метод SizeU.toString	410
6.196	Класс SortingConditions	411
6.196.1	Метод SortingConditions.add	411
6.196.2	Метод SortingConditions.clear	412
6.197	Перечисление SortingDirection	412
6.198	Класс Table	412
6.198.1	Метод Table.clearColumnGroups	413
6.198.2	Метод Table.clearRowGroups	413
6.198.3	Метод Table.createFiltersRange	413
6.198.4	Метод Table.duplicate	414
6.198.5	Метод Table.find	414
6.198.6	Метод Table.freeze	415
6.198.7	Метод Table.getCell	415

6.198.8	Метод Table.getCellRange	416
6.198.9	Метод Table.getCharts	416
6.198.10	Метод Table.getColumnsCount	416
6.198.11	Метод Table.getColumnWidth	417
6.198.12	Метод Table.getConditionalFormatRules	417
6.198.13	Метод Table.getFiltersRange	418
6.198.14	Метод Table.getFrozenRange	418
6.198.15	Метод Table.getImages	419
6.198.16	Метод Table.getLastNonEmptyCellInColumn	419
6.198.17	Метод Table.getLastNonEmptyCellInRow	419
6.198.18	Метод Table.getMediaObjects	420
6.198.19	Метод Table.getName	420
6.198.20	Метод Table.getNamedExpressions	420
6.198.21	Метод Table.getPrintAreas	421
6.198.22	Метод Table.getPrintTitles	421
6.198.23	Метод Table.getProtectionProperties	421
6.198.24	Метод Table.getRowHeight	422
6.198.25	Метод Table.getRowsCount	422
6.198.26	Метод Table.getShowZeroValue	422
6.198.27	Метод Table.getUsedRange	423
6.198.28	Метод Table.groupColumns	423
6.198.29	Метод Table.groupRows	423
6.198.30	Метод Table.insertColumnAfter	424
6.198.31	Метод Table.insertColumnBefore	424
6.198.32	Метод Table.insertImage	425
6.198.33	Метод Table.insertRowAfter	426
6.198.34	Метод Table.insertRowBefore	426
6.198.35	Метод Table.isColumnVisible	427
6.198.36	Метод Table.isProtected	427
6.198.37	Метод Table.isRowVisible	428
6.198.38	Метод Table.isVisible	428

6.198.39	Метод Table.moveTo	428
6.198.40	Метод Table.remove	429
6.198.41	Метод Table.removeColumn	429
6.198.42	Метод Table.removeProtection	429
6.198.43	Метод Table.removeRow	430
6.198.44	Метод Table.removeVisibleColumns	430
6.198.45	Метод Table.removeVisibleRows	430
6.198.46	Метод Table.setColumnsVisible	431
6.198.47	Метод Table.setColumnWidth	431
6.198.48	Метод Table.setName	431
6.198.49	Метод Table.setPrintArea	432
6.198.50	Метод Table.setPrintAreas	432
6.198.51	Метод Table.setPrintTitles	433
6.198.52	Метод Table.setProtection	433
6.198.53	Метод Table.setRowHeight	434
6.198.54	Метод Table.setRowsVisible	434
6.198.55	Метод Table.setShowZeroValue	435
6.198.56	Метод Table.setVisible	435
6.198.57	Метод Table.ungroupColumns	435
6.198.58	Метод Table.ungroupRows	436
6.198.59	Table.__eq__	436
6.198.60	Table.__ne__	436
6.199	Класс TableFilters	436
6.199.1	Метод TableFilters.clear	437
6.199.2	Метод TableFilters.erase	437
6.199.3	Метод TableFilters.getAsConditionalFilter	437
6.199.4	Метод TableFilters.getAsValueFilter	438
6.199.5	Метод TableFilters.getFilterType	438
6.199.6	Метод TableFilters.setFilter	439
6.200	Класс TableProtectionProperties	440
6.201	Класс TableSearchSettings	441

6.201.1	Перечисление TableSearchSettings.MatchBehaviour	442
6.201.2	Перечисление TableSearchSettings.SearchProperty	442
6.202	Класс TextAnchoredPosition	442
6.202.1	TextAnchoredPosition.__eq__	443
6.202.2	TextAnchoredPosition.__ne__	444
6.203	Класс TextConditionalFormatOperator	445
6.203.1	Метод TextConditionalFormatOperator.getArgument	446
6.203.2	Метод TextConditionalFormatOperator.getCondition	446
6.203.3	Метод TextConditionalFormatOperator.getType	446
6.204	Класс TextExportSettings	446
6.205	Перечисление TextLayout	447
6.206	Класс TextOrientation	448
6.206.1	Метод TextOrientation.getAngle	448
6.206.2	TextOrientation.isStackedChars	448
6.206.3	TextOrientation.__eq__	448
6.206.4	TextOrientation.__ne__	449
6.207	Класс TextProperties	449
6.207.1	TextProperties.__eq__	451
6.207.2	TextProperties.__ne__	451
6.208	Класс TextStyle	452
6.208.1	Метод TextStyle.createChild	452
6.208.2	Метод TextStyle.getName	452
6.208.3	Метод TextStyle.getNextParagraphStyle	453
6.208.4	Метод TextStyle.getParagraphProperties	453
6.208.5	Метод TextStyle.getParent	453
6.208.6	Метод TextStyle.getTextProperties	454
6.208.7	Метод TextStyle.setName	454
6.208.8	Метод TextStyle.setNextParagraphStyle	454
6.208.9	Метод TextStyle.setParagraphProperties	455
6.208.10	Метод TextStyle.setTextProperties	455
6.209	Класс TextStyles	456

6.209.1	Метод TextStyles.create	456
6.209.2	Метод TextStyles.get	456
6.209.3	Метод TextStyles.getEnumerator	457
6.210	Класс TextToColumnsSettings	457
6.210.1	Перечисление TextToColumnsSettings.TextQualifier	458
6.211	Перечисление TextUnit	459
6.212	Перечисление TextWrapType	460
6.213	Перечисление ThemeColorID	460
6.214	Перечисление TimePatterns	461
6.215	Класс TimeZone	461
6.216	Класс TopBottomConditionalFormatOperator	462
6.216.1	Метод TopBottomConditionalFormatOperator.getCondition	463
6.216.2	Метод TopBottomConditionalFormatOperator.getType	463
6.216.3	Метод TopBottomConditionalFormatOperator.getUsePercent	463
6.216.4	Метод TopBottomConditionalFormatOperator.getValue	463
6.217	Класс TrackedChange	464
6.217.1	Метод TrackedChange.getInfo	464
6.217.2	Метод TrackedChange.getRange	465
6.217.3	Метод TrackedChange.getType	465
6.218	Класс TrackedChangeInfo	465
6.219	Перечисление TrackedChangeType	466
6.220	Класс UnaryConditionalFormatOperator	466
6.220.1	Метод UnaryConditionalFormatOperator.getArgument	467
6.220.2	Метод UnaryConditionalFormatOperator.getCondition	468
6.220.3	Метод UnaryConditionalFormatOperator.getType	468
6.221	Класс UniquenessConditionalFormatOperator	468
6.221.1	Метод UniquenessConditionalFormatOperator.getCondition	469
6.221.2	Метод UniquenessConditionalFormatOperator.getType	470
6.222	Класс UserInfo	470
6.222.1	Метод UserInfo.__eq__	470
6.222.2	Метод UserInfo.__ne__	471

6.223	Перечисление ValueFieldsOrientation	471
6.224	Класс ValuesTableFilter	471
6.224.1	Метод ValuesTableFilter.add	472
6.224.2	Метод ValuesTableFilter.clear	472
6.224.3	Метод ValuesTableFilter.remove	472
6.225	Перечисление VerticalAlignment	473
6.226	Перечисление VerticalAnchorAlignment	474
6.227	Перечисление VerticalRelativeTo	474
6.228	Класс VerticalTextAnchoredPosition	475
6.228.1	VerticalTextAnchoredPosition.__eq__	475
6.228.2	VerticalTextAnchoredPosition.__ne__	476
6.229	Класс WorkbookExportSettings	476
6.230	Класс WorksheetHtmlExportSettings	477
6.231	Перечисление WorksheetPrinterFitType	478
6.232	Исключения	478
6.232.1	Класс BaseError	478
6.232.2	Класс ApplicationCreateError	478
6.232.3	Класс DocumentCreateError	479
6.232.4	Класс DocumentExportError	479
6.232.5	Класс DocumentLoadError	479
6.232.6	Класс DocumentModificationError	479
6.232.7	Класс DocumentSaveError	479
6.232.8	Класс ForbiddenActionError	479
6.232.9	Класс IncorrectArgumentError	480
6.232.10	Класс InvalidObjectError	480
6.232.11	Класс NoSuchElementError	480
6.232.12	Класс NotImplementedError	480
6.232.13	Класс OutOfRangeError	480
6.232.14	Класс ParseError	480
6.232.15	Класс PivotTableError	481
6.232.16	Класс PositionDocumentMismatchError	481

6.232.17 Класс ScriptExecutionError	481
6.232.18 Класс UnknownError	481
7 Механизм контроля версий Document API	482

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

В данном разделе представлены общие сведения о библиотеке MyOffice Document API для языка программирования Python.

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

Этот раздел содержит описание поставляемого дистрибутива, процесс его установки, а также инструкции для сборки и распространения разработанных приложений.

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-3.6.0-cp38-abi3-win_amd64.whl
Linux	MyOfficeSDKDocumentAPI-3.6.0-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-3.6.0-cp38-abi3-win_amd64.whl.



Внимание! Следует убедиться в актуальности установленной версии Python. Для использования MyOffice Document API 3.6.0 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-3.6.0-cp38-abi3-linux_x86_64.whl.



Внимание! Следует убедиться в актуальности установленной версии Python. Для использования MyOffice Document API 3.6.0 необходима версия Python 3.8.6.

2.4 Проверка работоспособности

Для проверки работоспособности MyOffice Document API необходимо выполнить тестовый пример.

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

```
from MyOfficeSDKDocumentAPI import DocumentAPI as myOfficeSDK

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

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

python basic-app.py

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

MyOffice Document API считается работоспособным, если приложение выполнено успешно.



Здесь и далее вместо DocumentAPI использован алиас myOfficeSDK.

2.5 Распространение разработанных приложений

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

Для запуска разработанного приложения на компьютере пользователя должны присутствовать:

- интерпретатор Python, версии 3.8.6;
- установленный пакет MyOffice Document API для языка программирования Python.

3 ОБЪЕКТНАЯ МОДЕЛЬ МОЙОФИС SDK

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

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

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

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

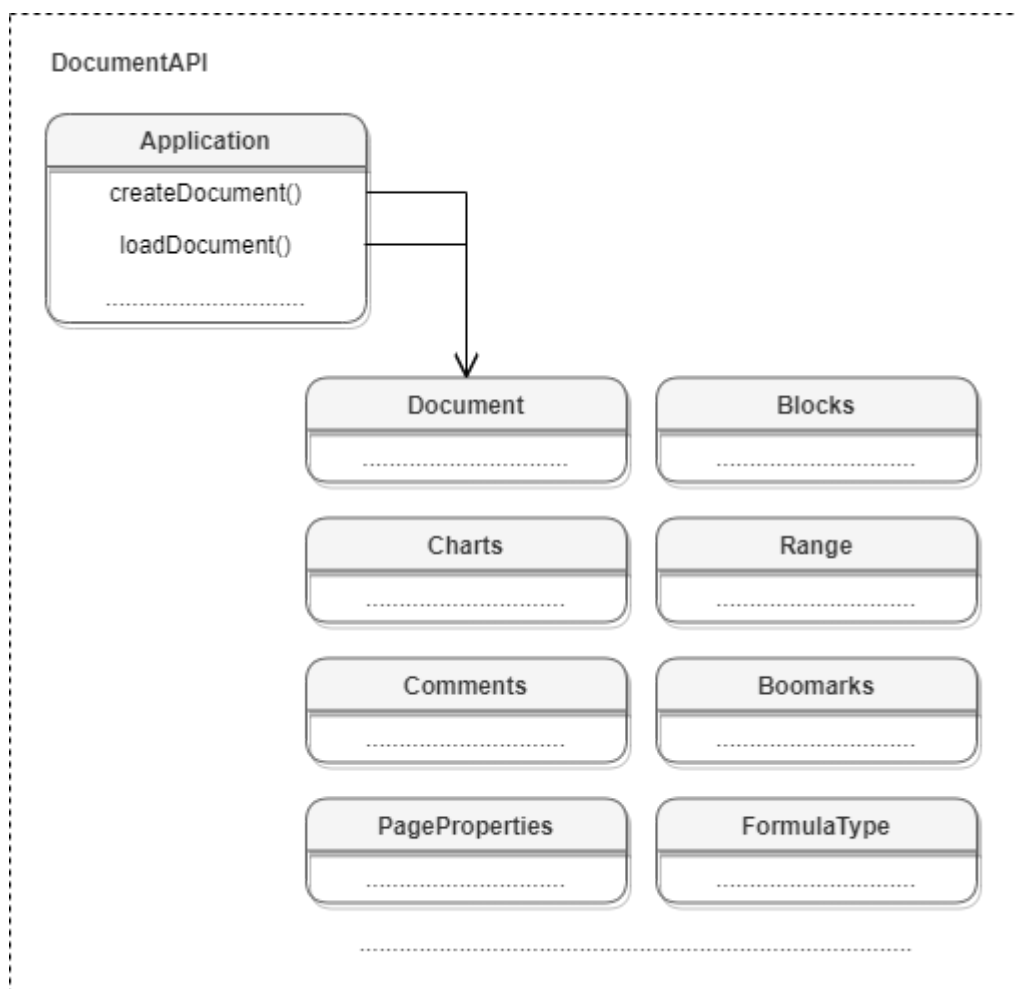


Рисунок 1 – Объектная модель МойОфис SDK.

4 РАБОТА С ДОКУМЕНТАМИ

В данном разделе представлены возможные сценарии использования библиотеки MyOffice Document API для взаимодействия с текстовыми и табличными документами.

4.1 Работа с текстовым документом

Данный раздел содержит специфичные для текстовых документов действия, доступные с помощью библиотеки MyOffice Document API.

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.

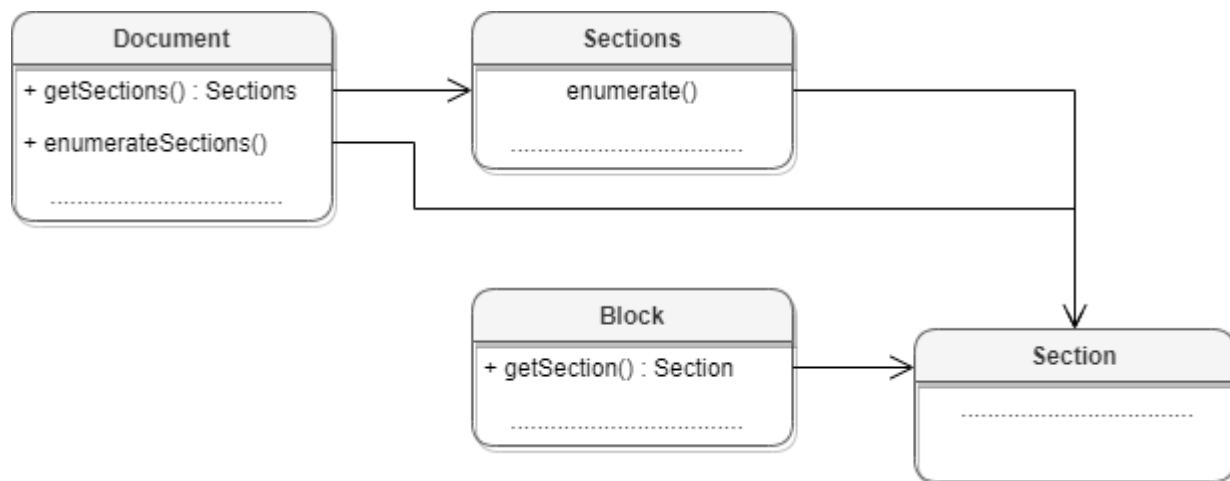


Рисунок 2 – Объектная модель классов для работы с секциями

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)
    
```

4.1.3.1 Работа с колонтитулами раздела

Для получения колонтитулов раздела следует использовать методы [Section.getHeaders\(\)](#) или [Section.getFooters\(\)](#).

Пример

```
section = document.getBlocks().getBlock(0).getSection()
headers = section.getHeaders()
for header in headers:
    print(header.getRange().extractText())

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

4.1.3.2 Управление ориентацией и свойствами страниц раздела

Для установки ориентации страницы можно использовать метод [Section.setPageOrientation\(\)](#) секции, полученной из блока документа.

```
section = document.getBlocks().getBlock(0).getSection()
section.setPageOrientation(myOfficeSDK.PageOrientation_Portrait)
```

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

```
pageProps = myOfficeSDK.PageProperties()
pageProps.width = 350
pageProps.height = 800

section = document.getBlocks().getBlock(0).getSection()
section.setPageProperties(pageProps)
```

Ориентация страниц может быть установлена для каждого раздела документа. Список разделов документа может быть получен из объекта [Document](#).

```
sections = document.getSections()
for section in sections:
    section.setPageOrientation(myOfficeSDK.PageOrientation_Portrait)
```

Ориентация страниц объекта [Section](#) может быть получена с использованием метода [Section.getPageOrientation\(\)](#).

```
section = document.getBlocks().getParagraph(0).getSection()  
orientation = section.getPageOrientation()
```

Свойства страниц объекта [Section](#) могут быть получены с использованием метода [Section.getPageProperties\(\)](#).

```
section = document.getBlocks().getParagraph(0).getSection()  
prop = section.getPageProperties()
```

4.1.4 Встроенные объекты в текстовом документе

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

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

- [вставка изображений](#) в текстовый документ;
- [перечисление графических объектов](#), находящихся в текстовом документе, определение их [типа](#) и [геометрических размеров](#);
- [перемещение графических объектов текстового документа, изменение их размеров](#).

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

4.1.4.1 Вставка изображения

Для вставки изображения используется метод [Position.insertImage\(\)](#).

Вставка изображения в текстовый документ

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

Вставка изображения в колонтитулы текстового документа

```
sections = document.getSections()  
for section in sections:
```

```
footers = section.getFooters()
for footer in footers:
    pos = footer.getRange().getBegin()
    pos.insertImage("logo.jpg", myOfficeSDK.SizeU(100, 50))
```

4.1.4.2 Перечисление встроенных объектов

Перечисление графических объектов в текстовом документе

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

Перечисление изображений в текстовом документе

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

Перечисление изображений в таблице текстового документа

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

4.1.5 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены на страницах документа.

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

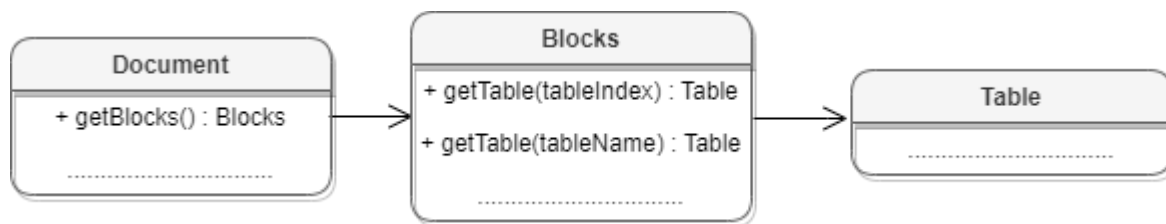


Рисунок 3 – Объектная модель для работы с таблицами

Получение таблицы текстового документа

Для получения таблицы используется метод [Blocks.getTable\(\)](#). В качестве аргумента используется индекс или имя таблицы.

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

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

Перечисление таблиц текстового документа

Для перечисления таблиц текстового документа можно использовать метод [Blocks.getTablesEnumerator\(\)](#).

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

Вставка таблицы в текстовый документ

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

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

Переименование таблицы

Для переименования таблицы используется метод [Table.setName\(\)](#).

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

Удаление таблицы

Для удаления таблицы используется метод [Table.remove\(\)](#).

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

4.1.6 Работа с закладками

Основным классом для работы с закладками является [Bookmarks](#). Список закладок документа возвращает метод [Document.getBookmarks\(\)](#). Метод [Bookmarks.getBookmarkRange\(\)](#) возвращает диапазон текста, метод [Bookmarks.removeBookmark\(\)](#) удаляет закладку по имени. Для создания закладки используется метод [Position.insertBookmark\(\)](#).

Доступны следующие операции с закладками:

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

Вставка закладки в указанное местоположение

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

Удаление закладки с заданным именем

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

Поиск закладки по имени

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

Замена текстового содержимого закладки

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

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

Вставка текста в закладку

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

Удаление содержимого закладки

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

Получение текстового содержимого закладки

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

Вставка таблицы в закладку

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

4.1.7 Рецензирование документов

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

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

Данные механизмы используются на стадии рецензирования или согласования документа с последующим внесением замечаний. Функции объектной модели для работы со средствами рецензирования позволяют получить детальную информацию о каждом

изменении: автор изменения, дата внесения изменения, оригинальный текст, измененный текст.

Для включения или отключения режима рецензирования используется метод [Document.setChangesTrackingEnabled\(\)](#). Для проверки текущего статуса данного режима используется метод [Document.isChangesTrackingEnabled\(\)](#).

Пример

```
document.setChangesTrackingEnabled(True)
print(document.isChangesTrackingEnabled())
```

Инструменты рецензирования применяются к диапазону документа, по этой причине методы доступа к ним находятся в классе [Range](#) (см. Рисунок 4).

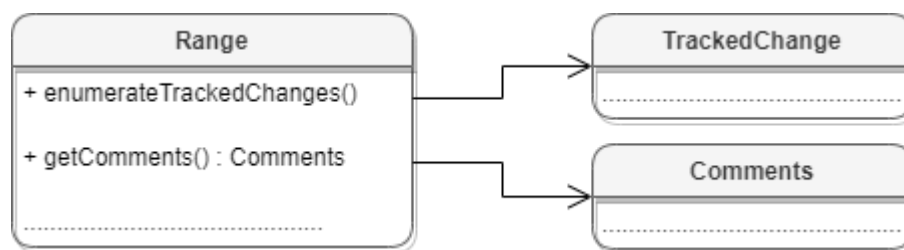


Рисунок 4 – Инструменты рецензирования документа

4.1.8 Работа с элементами управления

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

Используйте метод [Document.getContentControls\(\)](#) чтобы получить элементы управления из текущего документа:

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

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

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

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

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

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

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

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

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

4.2 Работа с табличным документом

Данный раздел содержит специфичные для табличных документов действия, доступные с помощью библиотеки MyOffice Document API.

4.2.1 Создание и открытие табличного документа

Метод [Application.createDocument](#) создает документ. В качестве параметра используется класс [DocumentType](#) или [DocumentSettings](#). Для создания табличного документа необходимо выбрать тип `DocumentType_Workbook`.

Пример создания табличного документа

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

Метод [Application.loadDocument](#) открывает документ. В качестве параметра используется путь к документу. Дополнительно может быть использован параметр [LoadDocumentSettings](#).

Примеры загрузки табличного документа

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

documentSettings = myOfficeSDK.DocumentSettings()
documentSettings.documentType = myOfficeSDK.DocumentType_Workbook
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_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)
```

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.

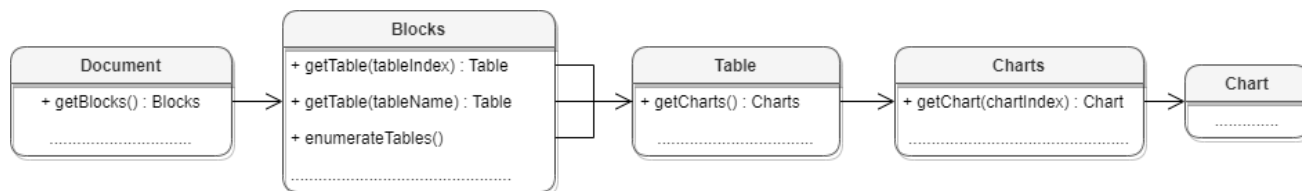


Рисунок 5 – Объектная модель классов для работы с диаграммами

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)

leftTopCellPosition = myOfficeSDK.CellPosition(0, 0)
rightBottomCellPosition = myOfficeSDK.CellPosition(1, 1)
srcCellRangePosition = myOfficeSDK.CellRangePosition(leftTopCellPosition,
rightBottomCellPosition)

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

sourceRange.copyInto(destRange)
```

Для перемещения ячеек следует воспользоваться методом [CellRange.moveInto\(\)](#):

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

Методы [CellRange.copyInto\(\)](#) и [CellRange.moveInto\(\)](#) также позволяют копировать и перемещать ячейки между документами и листами документа:

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

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

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

sourceRange.copyInto(targetRange)
```

4.2.5 Работа с формулами

Метод [Cell.setFormula](#) позволяет поместить формулу в ячейку таблицы:

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

Также при создании формулы можно использовать [именованные диапазоны](#) для обозначения группы ячеек.

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

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

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

```
if document.getCalculationMode() == myOfficeSDK.CalculationMode_Auto:
    document.setCalculationMode(myOfficeSDK.CalculationMode_Manual)
```

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

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

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

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

4.2.6 Проверка данных

Табличный редактор позволяет настроить проверку значений ячеек, чтобы разрешить ввод только корректных данных и исключить ошибки. Также вы можете проверить

правильность уже введенных значений. Поддерживаются следующие виды проверок: проверка по списку допустимых значений и проверка данных в формате **Дата**.

Проверка данных по списку значений

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

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

```
dvList = myOfficeSDK.DataValidation()  
dvList.setType(myOfficeSDK.DataValidationType_List)  
dvList.setFormula1("UK; Italy; Germany; Austria; Brazil")  
dvList.setShowDropDown(True)  
  
cellRange.setDataValidation(dvList)
```

Проверка данных в формате Дата

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

1. Создайте экземпляр класса [DataValidation](#).
2. Вызовите метод [DataValidation.setType\(\)](#) с параметром [DataValidationType Date](#) для создания проверки дат.
3. Используйте метод [DataValidation.setOperator\(\)](#), чтобы задать оператор сравнения.

4. Передайте дату для сравнения в метод [DataValidation.setFormula1\(\)](#). Метод принимает значения в виде строки в формате мм/дд/гггг, а также формулы, именованные диапазоны и адреса.
5. Если выбран оператор сравнения [Between](#) или [NotBetween](#), задайте вторую дату для промежутка с помощью метода [DataValidation.setFormula2\(\)](#).
6. Используя метод [CellRange.setDataValidation\(\)](#) примените проверку данных к диапазону ячеек.

```
dvDate = myOfficeSDK.DataValidation()  
dvDate.setType(myOfficeSDK.DataValidationType_Date)  
dvDate.setOperator(myOfficeSDK.DataValidationOperator_Between)  
dvDate.setFormula1("06/01/2024")  
dvDate.setFormula2("08/31/2024")  
  
cellRange.setDataValidation(dvDate)
```

Отображение сообщений об ошибках

Добавьте визуальное предупреждение о вводе недопустимых значений:

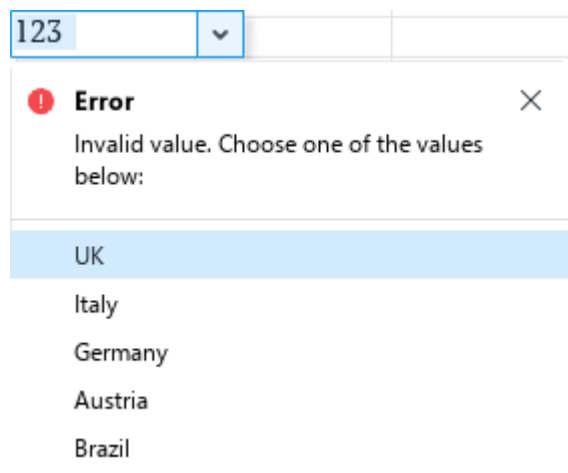


Рисунок 7 – Ошибка проверки данных

1. Вызовите метод [DataValidation.setShowErrorMessage\(\)](#) с параметром True для показа сообщения об ошибке.
2. Задайте поведение редактора при вводе недопустимых значений с помощью метода [DataValidation.setErrorStyle\(\)](#): запретить ввод недопустимых значений ([DataValidationErrorStyle Stop](#)) или разрешить ввод после показа предупреждения ([DataValidationErrorStyle Warning](#)).
3. Передайте заголовок сообщения в метод [DataValidation.setErrorTitle\(\)](#).

4. Используйте метод [DataValidation.setErrorMessage\(\)](#) для задания сообщения об ошибке.

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки сообщения об ошибке:  
validation.setShowErrorMessage(True)  
validation.setErrorStyle(myOfficeSDK.DataValidationErrorStyle_Stop)  
validation.setErrorTitle("Error")  
validation.setErrorMessage("Invalid value. Choose one of the values below:")  
  
cellRange.setDataValidation(validation)
```

Отображение подсказок

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

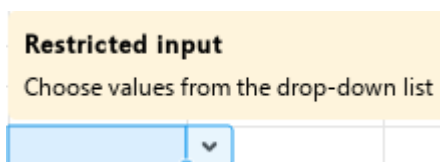


Рисунок 9 – Подсказка при вводе значения в ячейку с проверкой

1. Вызовите метод [DataValidation.setShowInputMessage\(\)](#) с параметром True для показа подсказки.
2. Передайте заголовок подсказки в метод [DataValidation.setPromptTitle\(\)](#).
3. Используйте метод [DataValidation.setPrompt\(\)](#) для задания сообщения подсказки.

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки подсказки:  
validation.setShowInputMessage(True)  
validation.setPromptTitle("Restricted input")  
validation.setPrompt("Choose values from the drop-down list")  
  
cellRange.setDataValidation(validation)
```

Обработка результатов проверки

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

```
if cell.getDataValidation() is None:
    print("No validation applied")
elif cell.checkDataValidation().isValid():
    print("Validation passed")
else:
    validation = cell.checkDataValidation().getDataValidation()
    print(validation.getErrorMessage())
```

Получение настроек проверок

Чтобы получить настройки проверки данных для конкретной ячейки, используйте метод [Cell.getDataValidation\(\)](#).

Удаление проверок

Вы можете использовать метод [CellRange.clearDataValidations\(\)](#), чтобы убрать проверку данных из диапазона ячеек.

4.2.7 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:

- [вставка изображений](#) в табличный документ;
- [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\(\)](#).

4.2.7.1 Вставка изображения

Для вставки изображения используется метод [Table.insertImage\(\)](#).

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

4.2.7.2 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)
```

4.2.8 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам [Table](#) осуществляется из [Blocks](#) (см. Рисунок 10). В табличном документе таблицами являются листы документа.

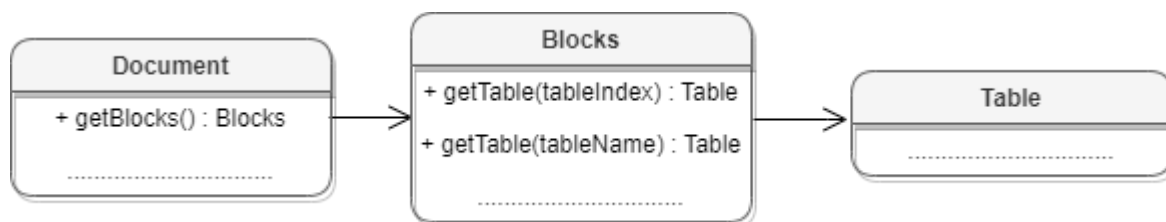


Рисунок 10 – Объектная модель для работы с таблицами

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()
```

4.2.9 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 11.

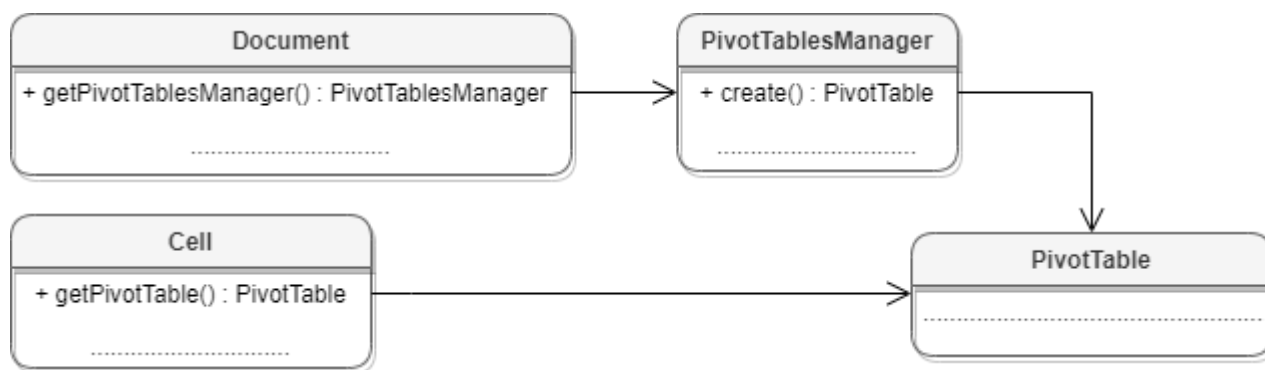


Рисунок 11 – Сводные таблицы

4.2.9.1 Создание сводной таблицы

Метод [PivotTablesManager.create](#) используется для добавления сводной таблицы в документ.

Пример

```

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

4.2.9.2 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())
```

4.2.9.3 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())
```

4.2.9.4 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())
```

4.2.9.5 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())
```

4.2.9.6 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)
```

4.2.9.7 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()
```

4.2.9.8 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)
```

4.2.9.9 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)
```

4.2.9.10 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)
```

4.2.9.11 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)
```

4.2.9.12 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)
```

4.2.9.13 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:
```

4.2.10 Работа с фильтрами

Работа с фильтрами возможна только в табличном документе. Диаграмма взаимодействия объектов приведена на рисунке 12.

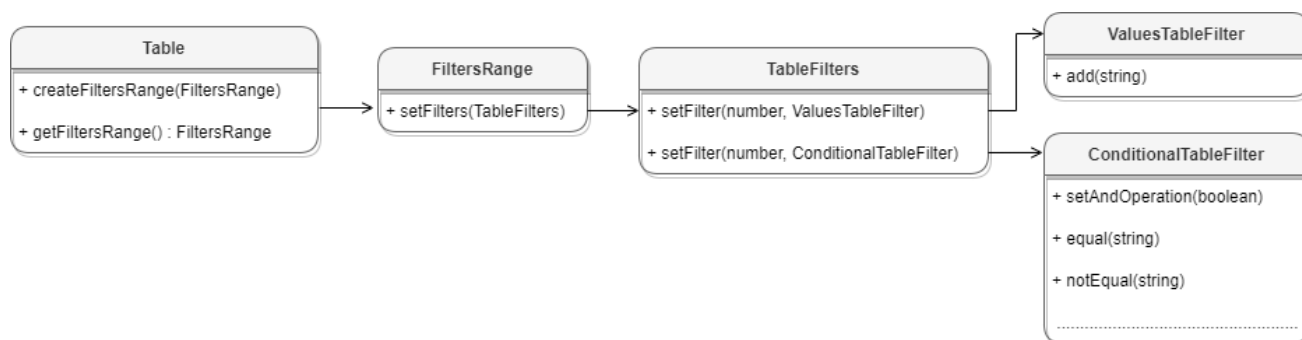


Рисунок 12 – Объектная модель таблиц для работы с фильтрами

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

Далее создаются фильтры (возможные варианты: [ValuesTableFilter](#), [ConditionalTableFilter](#)).

Фильтры помещаются в структуру [TableFilters](#).

Далее фильтры [TableFilters](#) помещаются в диапазон [FiltersRange](#) посредством использования метода [FiltersRange.setFilters\(\)](#).

Пример работы с фильтрами в табличном документе

```

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

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

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

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

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

```
filtersRange.setFilters(tableFilters)
```

4.3 Встроенные объекты

В данном разделе содержатся общие для текстовых и табличных документов методы взаимодействия со встроенными объектами.

4.3.1 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")
```

4.3.2 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)
```

4.4 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 13).

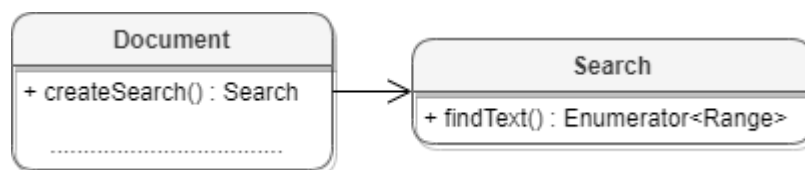


Рисунок 13 – Объектная модель для поиска в документе

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())
```

Для поиска ячеек в табличном документе используйте методы [Table.find](#) и [CellRange.find](#).

Пример поиска ячеек в табличном документе

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

searchProps = myOfficeSDK.TableSearchSettings()
searchProps.caseSensitive = myOfficeSDK.CaseSensitive_No
searchProps.matchBehaviour = myOfficeSDK.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = myOfficeSDK.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = True

results = sheet.find("*eye", searchProps)
for cell in results:
    print(cell.getFormattedValue()) # Steeleye Stout
```

Пример форматирования результатов поиска

```
sheet = document.getBlocks().getTable(0)
search = myOfficeSDK.createSearch(document)
template = "MyOffice"
ranges = search.findText(template, sheet)
for range in ranges:
    props = range.getTextProperties()
    props.bold = True
    range.setTextProperties(props)
```

4.5 Работа с макрокомандами

Класс `Scripts` предоставляет доступ к списку макрокоманд документа. На рисунке 14 изображена объектная модель классов, относящихся к работе с макрокомандами.

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

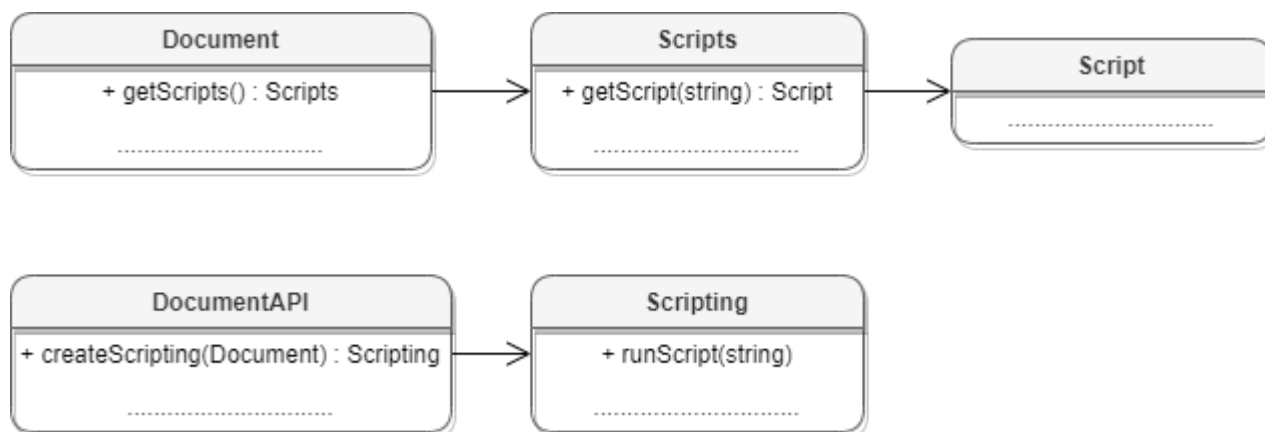


Рисунок 14 – Объектная модель классов для работы с макрокомандами

Доступны следующие операции:

- [получение списка макрокоманд](#);
- [добавление макрокоманды](#);
- [получение макрокоманды по имени](#);
- [удаление макрокоманды](#);
- [запуск макрокоманды](#).

4.6 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 15).

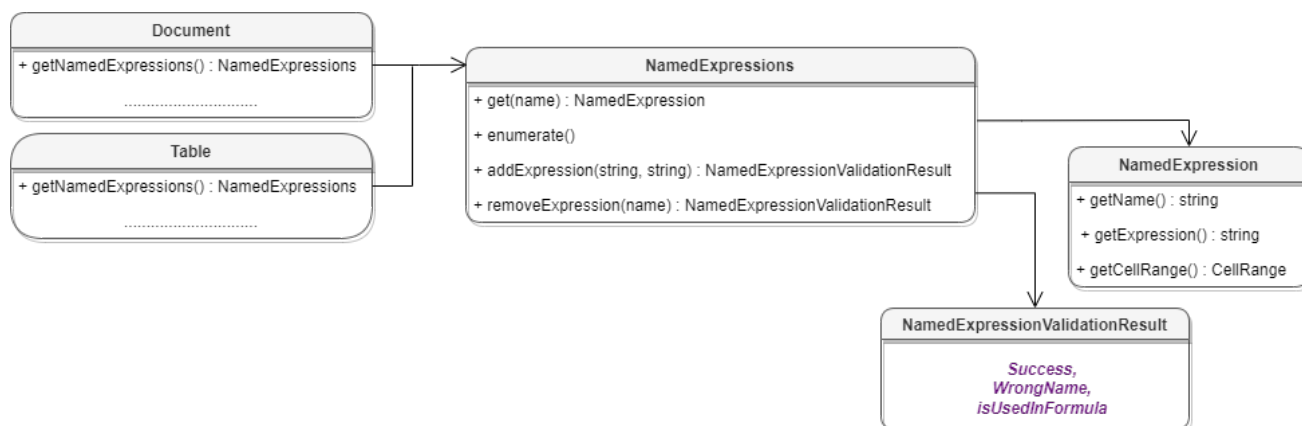


Рисунок 15 – Классы для работы с именованными диапазонами

4.6.1 Доступ к именованным диапазонам

Доступ к именованным диапазонам осуществляется посредством методов [Document.getNamedExpressions\(\)](#) и [Table.getNamedExpressions\(\)](#).

Примеры

```

namedExpressions = document.getNamedExpressions()

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

```

4.6.2 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())

```

4.6.3 Добавление именованного диапазона

Для добавления именованного диапазона используется метод [NamedExpressions.addExpression\(\)](#).

Пример

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

4.6.4 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()
```

4.6.5 Переименование именованного диапазона

Для переименования именованного диапазона используется метод [NamedExpression.setName\(\)](#).

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

4.6.6 Удаление именованного диапазона

Для удаления именованного диапазона используется метод [NamedExpressions.removeExpression\(\)](#).

Пример

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

4.7 Работа со строками и столбцами таблиц

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

4.7.1 Группировка строк и колонок таблицы

Следующий набор методов позволяет группировать строки и колонки таблицы: [Table.groupRows\(\)](#), [Table.ungroupRows\(\)](#), [Table.clearRowGroups\(\)](#), [Table.groupColumns\(\)](#), [Table.ungroupColumns\(\)](#), [Table.clearColumnGroups\(\)](#).

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

Методы могут вызвать исключения [DocumentAPI.OutOfRangeError](#) и [DocumentAPI.IncorrectArgumentError](#) в случае использования индексов, выходящих за рамки таблицы.

4.7.2 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)
```

4.8 Работа с ячейками таблиц

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

4.8.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 16):

- непосредственно из таблицы, используя метод [Table.getCell\(\)](#);
- из диапазона ячеек методом перечисления [CellRange.enumerate\(\)](#).

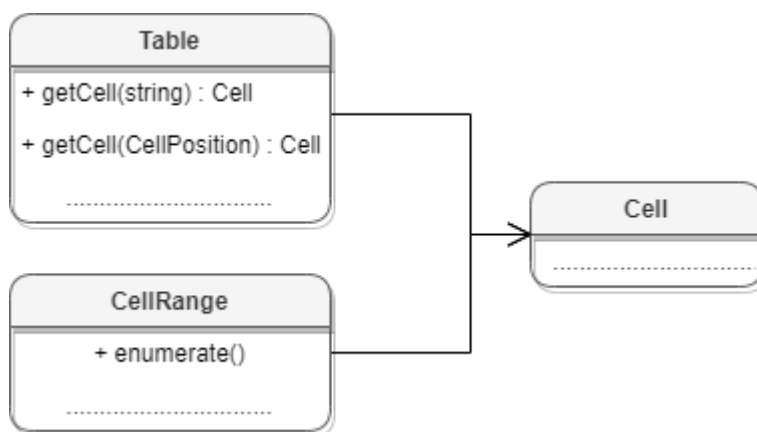


Рисунок 16 – Объектная модель для работы с ячейками таблиц

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("List1")
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("List1")
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())
```

4.8.2 Форматирование ячеек

При работе с ячейками таблиц можно использовать следующие варианты форматирования:

- форматирование параметров ячейки [CellProperties](#), например, цвет фона, угол поворота текста;
- форматирование [абзаца ячейки](#), например, отступы абзаца, межстрочный интервал текста;
- форматирование [текста](#), например, цвет текста, начертание;
- задание параметров [границ ячеек](#).

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

Пример установки и получения свойств абзаца ячейки

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

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

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

Пример настроек текста ячейки

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

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

cell.setTextProperties(textProperties)
```

4.8.3 Форматирование границ ячеек

Для оформления границ ячеек используется класс [Borders](#) (см. Рисунок 17). 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.

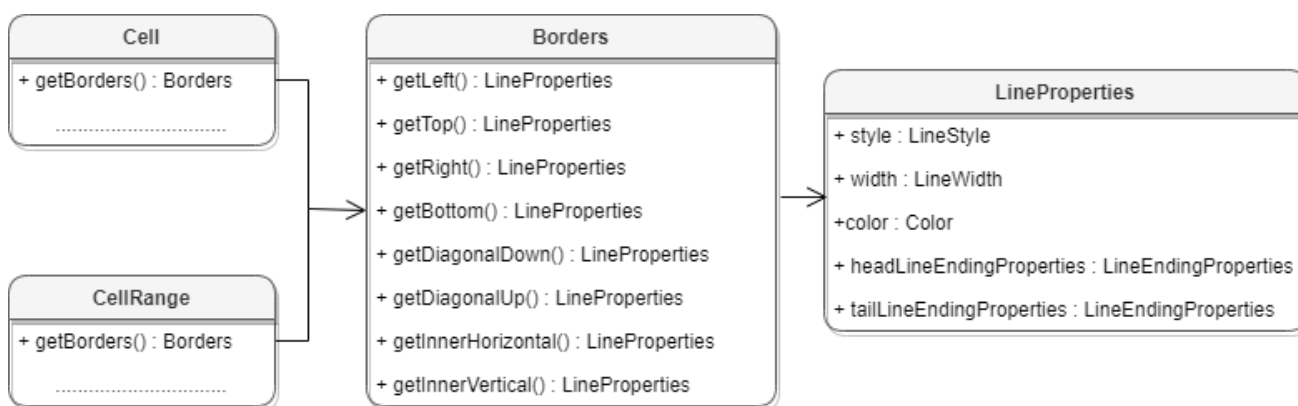


Рисунок 17 – Классы для работы с границами ячеек

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

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

4. Установить границы ячеек с помощью [Cell.setBorders\(\)](#) или [CellRange.setBorders\(\)](#).

Пример настройки границ ячеек

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

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

4.8.4 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()
```

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

Пример

```
cell = firstSheet.getCell("A2")
if cell.isInMergedRange():
    mergedRange = cell.getMergedRange()
```

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()
```

4.9 Защита документов

Данный раздел содержит способы защиты содержимого текстовых и табличных документов:

- [Защита диапазона текстового документа](#)
- [Защита листа табличного документа](#)
- [Защита структуры табличного документа](#)

4.9.1 Защита диапазона текстового документа

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

Используйте метод [Range.lockContent\(\)](#) для установки защиты:

```
# Защита первого абзаца документа:
textRange =
document.getRange().getBegin().getCurrentRange(myOfficeSDK.TextUnit_Paragraph)
textRange.lockContent()
```

При попытке редактирования защищенного фрагмента возникает исключение [DocumentModificationError](#):

```
pos = textRange.getContentEnd().getPreviousPosition(10)
pos.insertText("My Text") # DocumentModificationError
```

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

```
print(textRange.isContentLocked()) # True
print(document.getRange().isContentLocked()) # True
```

Для снятия защиты, вызовите метод [Range.unlockContent\(\)](#) защищенного фрагмента или диапазона текста, который его содержит:

```
textRange.unlockContent()  
# или  
document.getRange().unlockContent()
```

Также вы можете снять защиту только с части диапазона:

```
# Снятие защиты с последнего предложения защищенного абзаца:  
textRange.getContentEnd().getPreviousRange(myOfficeSDK.TextUnit_Sentence).unlockContent()
```

4.9.2 Защита листа табличного документа

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

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

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

```
sheet = document.getBlocks().getTable(0)  
# Разрешает редактировать все ячейки листа:  
cells = sheet.getCellRange("A1:Z300")  
cellProtection = myOfficeSDK.CellProtectionProperties()  
cellProtection.lockedForChanges = False  
cellProtection.formulasNotDisplayed = False  
cells.setProtectionProperties(cellProtection)  
# Задаёт ячейки, которые будут защищены от изменения:  
protectedCells = sheet.getCellRange("A1:H8")  
cellProtection1 = myOfficeSDK.CellProtectionProperties()  
cellProtection1.lockedForChanges = True  
cellProtection1.formulasNotDisplayed = False  
protectedCells.setProtectionProperties(cellProtection1)  
# Запрещает вставку и удаление строк и столбцов:  
tableProtection = myOfficeSDK.TableProtectionProperties()  
tableProtection.deleteColumns = False  
tableProtection.deleteRows = False
```

```
tableProtection.filterData = True
tableProtection.formatCells = True
tableProtection.formatColumns = True
tableProtection.formatRows = True
tableProtection.insertAndEditObjects = True
tableProtection.insertAndEditPivotTables = True
tableProtection.insertColumns = False
tableProtection.insertLinks = True
tableProtection.insertRows = False
tableProtection.selectProtectedCells = True
tableProtection.sortData = True
# Устанавливает защиту на лист и пароль для ее снятия:
sheet.setProtection(tableProtection, "password")
```

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

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

```
cell = sheet.getCell("F6")
print(cell.isProtected()) # True
```

Текущие настройки защиты ячеек возвращают методы [`Cell.getProtectionProperties\(\)`](#) и [`CellRange.getProtectionProperties\(\)`](#).

Статус защиты листа документа можно получить используя метод [`Table.isProtected\(\)`](#):

```
print(sheet.isProtected()) # True
```

Метод [`Table.getProtectionProperties\(\)`](#) позволяет получить настройки защиты листа.

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

```
sheet.removeProtection("password")
```

4.9.3 Защита структуры табличного документа

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

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

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

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

```
sheet = document.getBlocks().getTable(0)
sheet.remove() # SpreadsheetProtectionError
```

Используйте метод [Document.isStructureProtected\(\)](#) для получения состояния защиты структуры документа:

```
print(document.isStructureProtected()) # True
```

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

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

4.10 Локализация документов

Библиотеки Document API по умолчанию работают с документами используя английскую локализацию. Поэтому, при взаимодействии с документом через API, значения могут отличаться от отображаемых в интерфейсе редактора.

Пример получения даты для стандартной локализации

```
newDocument = application.createDocument(myOfficeSDK.DocumentType_Workbook)
sheet = newDocument.getRange().getBegin().insertTable(10, 10, "Sheet1")
cellRange = sheet.getCellRange("A1")
cellRange.insertCurrentDateTime(myOfficeSDK.DateTimeFormat_DateTime)
cell = sheet.getCell("A1")
print(cell.getFormattedValue()) # 05/23/2025 10:53 AM
```

Используйте поле [LocaleInfo.localeName](#), чтобы задать локализацию при открытии или создании документа. Это позволит работать с документами используя локализованные значения.

Пример получения даты для русской локализации

```
settings = myOfficeSDK.DocumentSettings()
settings.documentType = myOfficeSDK.DocumentType_Workbook
settings.localeInfo = myOfficeSDK.LocaleInfo()
settings.localeInfo.localeName = "ru_RU"

newDocument = application.createDocument(settings)
sheet = newDocument.getRange().getBegin().insertTable(10, 10, "Sheet1")
cellRange = sheet.getCellRange("A1")
cellRange.insertCurrentDateTime(myOfficeSDK.DateTimeFormat_DateTime)
cell = sheet.getCell("A1")
print(cell.getFormattedValue()) # 23.05.2025 10:53
```

5 ГЛОБАЛЬНЫЕ МЕТОДЫ

В данном разделе приведено описание глобальных методов библиотеки MyOffice Document API для языка программирования Python.

5.1 Глобальный метод createSearch

Метод инициализирует механизм поиска для текущего документа. Возвращает объект [Search](#), с помощью которого выполняются поисковые запросы.

Пример

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

5.2 Глобальный метод createScripting

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

Пример

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

5.3 Глобальный метод exportWorksheetToHtml

Метод позволяет сгенерировать HTML документ на основе заданного листа табличного документа. Данный метод возвращает HTML элементы и CSS стили этих элементов, которые впоследствии могут быть встроены в вашу HTML разметку.

Вызов

```
static HtmlFragments exportWorksheetToHtml(table, settings)
```

Параметры

- table: лист табличного документа, тип [Table](#).
- settings: настройки генерации HTML документа, тип [WorksheetHtmlExportSettings](#).

Возвращает

- HTML элементы, тип [HtmlFragments](#).

Ограничения

- Структура возвращаемых элементов может быть изменена в будущих релизах.
- Поддерживается только генерация текста и стилей для ячеек, строк и столбцов.

Пример

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

settings = myOfficeSDK.WorksheetHtmlExportSettings()
settings.exportHeaders = False
settings.exportMissingBorders = False
html = myOfficeSDK.exportWorksheetToHtml(sheet, settings)

with open("style.css", "w") as fileCSS:
    fileCSS.write(html.rootCss)

header = "<head>\r\n  <link rel=\"stylesheet\" \r\nhref=\"style.css\">\r\n</head>\r\n"

with open("doc.html", "w") as fileHTML:
    fileHTML.write(header + html.body)
```

– None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_AboveAverage:
        aboveOperator =
myOfficeSDK.castToAboveAverageConditionalFormat(rule.getOperator())
```

5.4.2 Глобальный метод castToBinaryConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к [BinaryConditionalFormatOperator](#).

Вызов

```
static BinaryConditionalFormatOperator
castToBinaryConditionalFormat(conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

– оператор для правил "Между" и "Не между", тип [BinaryConditionalFormatOperator](#).

– None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_Binary:
        aboveOperator =
myOfficeSDK.castToBinaryConditionalFormat(rule.getOperator())
```

5.4.3 Глобальный метод castToColorScaleConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к [ColorScaleConditionalFormatOperator](#).

Вызов

```
static ColorScaleConditionalFormatOperator  
castToColorScaleConditionalFormat(conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

– оператор для правила "Цветовая шкала", тип [ColorScaleConditionalFormatOperator](#).

– None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)  
rules = sheet.getConditionalFormatRules()  
for rule in rules:  
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_ColorScale:  
        aboveOperator =  
myOfficeSDK.castToColorScaleConditionalFormat(rule.getOperator())
```

5.4.4 Глобальный метод castToDataBarConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к [DataBarConditionalFormatOperator](#).

Вызов

```
static DataBarConditionalFormatOperator  
castToDataBarConditionalFormat(conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

– оператор для правила "Гистограмма", тип [DataBarConditionalFormatOperator](#).

– None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_DataBar:
        aboveOperator =
myOfficeSDK.castToDataBarConditionalFormat(rule.getOperator())
```

5.4.5 Глобальный метод castToIconSetConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к [IconSetConditionalFormatOperator](#).

Вызов

```
static IconSetConditionalFormatOperator
castToIconSetConditionalFormat(conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

– оператор для правила "Значки", тип [IconSetConditionalFormatOperator](#).
– None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_IconSet:
        aboveOperator =
myOfficeSDK.castToIconSetConditionalFormat(rule.getOperator())
```

5.4.6 Глобальный метод castToNullaryConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к [NullaryConditionalFormatOperator](#).

Вызов

```
static NullaryConditionalFormatOperator
castToNullaryConditionalFormat(conditionalFormatOperator)
```

Параметры

- conditionalFormatOperator: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

- оператор для правил без параметров, тип [NullaryConditionalFormatOperator](#).
- None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_Nullary:
        aboveOperator =
myOfficeSDK.castToNullaryConditionalFormat(rule.getOperator())
```

5.4.7 Глобальный метод castToTextConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к [TextConditionalFormatOperator](#).

Вызов

```
static TextConditionalFormatOperator
castToTextConditionalFormat(conditionalFormatOperator)
```

Параметры

- conditionalFormatOperator: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

- оператор для правила "Текст", тип [TextConditionalFormatOperator](#).
- None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_Text:
        aboveOperator =
myOfficeSDK.castToTextConditionalFormat(rule.getOperator())
```

5.4.8 Глобальный метод `castToTopBottomConditionalFormat`

Метод позволяет привести базовый оператор условного форматирования к [TopBottomConditionalFormatOperator](#).

Вызов

```
static TopBottomConditionalFormatOperator  
castToTopBottomConditionalFormat(conditionalFormatOperator)
```

Параметры

– `conditionalFormatOperator`: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

– оператор для правила "Наибольшие и наименьшие значения", тип [TopBottomConditionalFormatOperator](#).

– `None`, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)  
rules = sheet.getConditionalFormatRules()  
for rule in rules:  
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_TopBottom:  
        aboveOperator =  
myOfficeSDK.castToTopBottomConditionalFormat(rule.getOperator())
```

5.4.9 Глобальный метод `castToUnaryConditionalFormat`

Метод позволяет привести базовый оператор условного форматирования к [UnaryConditionalFormatOperator](#).

Вызов

```
static UnaryConditionalFormatOperator  
castToUnaryConditionalFormat(conditionalFormatOperator)
```

Параметры

– `conditionalFormatOperator`: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

- оператор для правил "Больше", "Меньше", "Равно" и "Не равно", тип [UnaryConditionalFormatOperator](#).
- None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_Unary:
        aboveOperator =
myOfficeSDK.castToUnaryConditionalFormat(rule.getOperator())
```

5.4.10 Глобальный метод castToUniquenessConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к [UniquenessConditionalFormatOperator](#).

Вызов

```
static UniquenessConditionalFormatOperator
castToUniquenessConditionalFormat(conditionalFormatOperator)
```

Параметры

- conditionalFormatOperator: оператор условного форматирования, тип [ConditionalFormatOperator](#).

Возвращает

- оператор для правила "Уникальные и повторяющиеся значения", тип [UniquenessConditionalFormatOperator](#).
- None, если приведение невозможно.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_Uniqueness:
        aboveOperator =
myOfficeSDK.castToUniquenessConditionalFormat(rule.getOperator())
```

5.4.11 Глобальный метод `createAboveAverageConditionalFormatOperator`

Метод создает оператор, который содержит настройки применения форматирования для правил "Выше среднего" и "Ниже среднего". Пример использования смотри в разделе [AboveAverageConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createAboveAverageConditionalFormatOperator(condition, stdDev)
```

Параметры

- `condition`: условие применения форматирования, тип [ConditionalFormatAboveAverageCondition](#).
- `stdDev`: (необязательный) количество стандартных отклонений, тип `int`.

Возвращает

- оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.12 Глобальный метод `createBinaryConditionalFormatOperator`

Метод создает оператор, который содержит настройки применения форматирования для правил "Между" и "Не между". Пример использования смотри в разделе [BinaryConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createBinaryConditionalFormatOperator(condition, firstArgument, secondArgument)
```

Параметры

- `condition`: условие применения форматирования, тип [ConditionalFormatBinaryCondition](#).
- `firstArgument`: первый аргумент, тип `string`.
- `secondArgument`: второй аргумент, тип `string`.

Возвращает

- оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.13 Глобальный метод `createColorScaleConditionalFormatOperator`

Метод создает оператор, который содержит настройки применения форматирования для правила "Цветовая шкала". Пример использования смотри в разделе [ColorScaleConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createColorScaleConditionalFormatOperator(entries)
```

Параметры

– entries: набор правил применения цветов, тип [ConditionalFormatColorScaleEntries](#).

Возвращает

– оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.14 Глобальный метод `createDataBarConditionalFormatOperator`

Метод создает оператор, который содержит настройки применения форматирования для правила "Гистограмма". Пример использования смотри в разделе [DataBarConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createDataBarConditionalFormatOperator(params)
```

Параметры

– params: настройки гистограммы, тип [ConditionalFormatDataBarParams](#).

Возвращает

– оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.15 Глобальный метод `createIconSetConditionalFormatOperator`

Метод создает оператор, который содержит настройки применения форматирования для правила "Значки". Пример использования смотри в разделе [IconSetConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createIconSetConditionalFormatOperator(entries, isValueShown)
```

Параметры

- entries: набор правил отображения значков, тип [ConditionalFormatIconSetEntries](#).
- isValueShown: True, чтобы показывать значение ячейки при примененном форматировании, в ином случае – False.

Возвращает

- оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.16 Глобальный метод createNullaryConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правил без параметров. Пример использования смотри в разделе [NullaryConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createNullaryConditionalFormatOperator(condition)
```

Параметры

- condition: условие применения форматирования, тип [ConditionalFormatNullaryCondition](#).

Возвращает

- оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.17 Глобальный метод createTextConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правила "Текст". Пример использования смотри в разделе [TextConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createTextConditionalFormatOperator(condition, argument)
```

Параметры

– condition: условие применения форматирования, тип [ConditionalFormatTextCondition](#).

– argument: аргумент условия, тип string.

Возвращает

– оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.18 Глобальный метод createTopBottomConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правила "Наибольшие и наименьшие значения". Пример использования смотри в разделе [TopBottomConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createTopBottomConditionalFormatOperator(condition, value, usePercent)
```

Параметры

– condition: условие применения форматирования, тип [ConditionalFormatTopBottomCondition](#).

– value: количество/процент значений для выделения, тип int.

– usePercent: True, чтобы выделять определенный процент значений; чтобы выделять определенное количество значений – False.

Возвращает

– оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.19 Глобальный метод createUnaryConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Пример использования смотри в разделе [UnaryConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createUnaryConditionalFormatOperator(condition, argument)
```

Параметры

— condition: условие применения форматирования, тип [ConditionalFormatUnaryCondition](#).

— argument: аргумент условия, тип string.

Возвращает

— оператор условного форматирования, тип [ConditionalFormatOperator](#).

5.4.20 Глобальный метод createUniquenessConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правила "Уникальные и повторяющиеся значения". Пример использования смотри в разделе [UniquenessConditionalFormatOperator](#).

Вызов

```
static ConditionalFormatOperator  
createUniquenessConditionalFormatOperator(condition)
```

Параметры

— condition: условие применения форматирования, тип [ConditionalFormatUniquenessCondition](#).

Возвращает

— оператор условного форматирования, тип [ConditionalFormatOperator](#).

6 СПРАВОЧНИК КЛАССОВ, СТРУКТУР И МЕТОДОВ

В данном разделе приведено описание классов, структур и методов библиотеки MyOffice Document API для языка программирования Python в алфавитном порядке.

6.1 Класс AboveAverageConditionalFormatOperator

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

Конструкторы

```
AboveAverageConditionalFormatOperator(ConditionalFormatAboveAverageCondition condition)
```

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

Пример

Количество
2
5
10
1
3
4
6
15
20
2
7

Рисунок 18 – Пример создания правила "Выше среднего"

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

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

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

```
255, 0, 150)))  
aboveStyle.cellProperties = cellProperties  
  
aboveOperator =  
myOfficeSDK.createAboveAverageConditionalFormatOperator(myOfficeSDK.ConditionalFo  
rmatAboveAverageCondition_Above)  
  
aboveRule = myOfficeSDK.ConditionalFormatRule(aboveOperator, aboveStyle,  
cellRangePosition, False)  
rules.addRule(aboveRule)
```

6.1.1 Метод AboveAverageConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
ConditionalFormatAboveAverageCondition getCondition()
```

Возвращает

— условие применения форматирования, тип [ConditionalFormatAboveAverageCondition](#).

6.1.2 Метод AboveAverageConditionalFormatOperator.getStdDev

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

Вызов

```
int getStdDev()
```

Возвращает

- количество стандартных отклонений, тип `int`.
- `None`, если отклонения не заданы.

6.1.3 Метод AboveAverageConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.2 Класс AbsoluteFrame

Класс `AbsoluteFrame` описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 19). Предназначен для получения и изменения свойств позиции медиаобъектов. Используется в табличном документе.

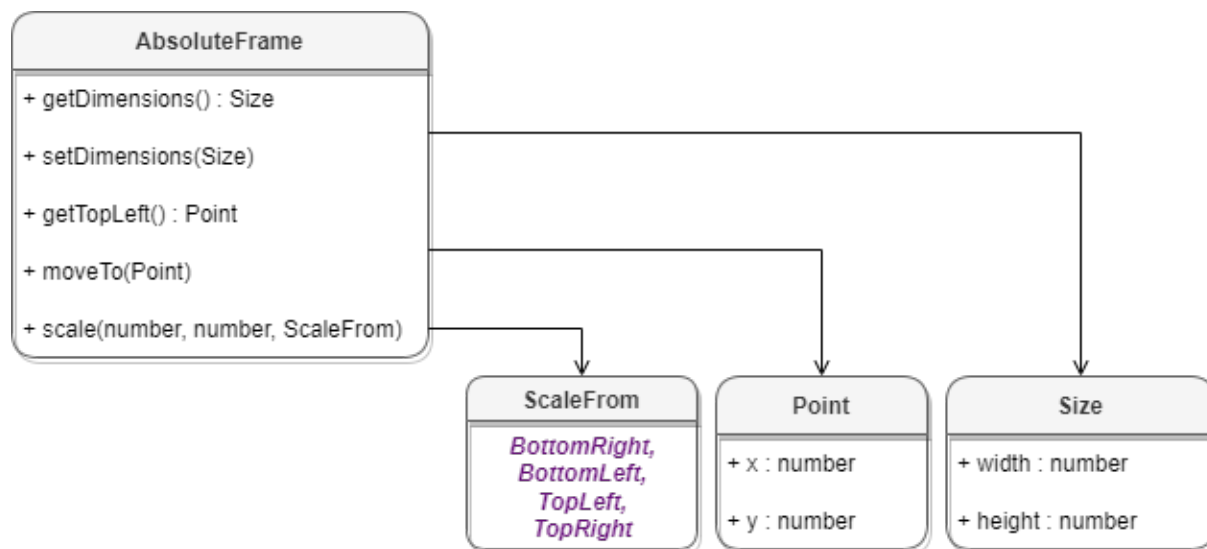


Рисунок 19 – Объектная модель класса `AbsoluteFrame`

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())
    
```

6.2.1 Метод `AbsoluteFrame.getDimensions`

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

6.2.2 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)
```

6.2.3 Метод `AbsoluteFrame.moveTo`

Метод перемещает объект в заданную позицию.

Пример

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

6.2.4 Метод `AbsoluteFrame.scale`

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

Вызов

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

Параметры

- `widthScale` – коэффициент масштабирования по горизонтали, тип - числовой;
- `heightScale` – коэффициент масштабирования по вертикали, тип - числовой;
- `scaleFrom` – точка, сохраняющая позицию при масштабировании, тип - [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)
```

6.2.5 Метод `AbsoluteFrame.setDimensions`

Метод задает размеры (изменяет размер) медиаобъекта.

Вызов

```
setDimensions(size)
```

Параметры

size – размеры встроенного объекта, тип - [SizeU](#).

Пример

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

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

6.3 Класс `AccountingCellFormatting`

Класс содержит параметры финансового формата ячеек таблицы и используется в качестве аргумента метода [Cell.setFormat\(\)](#).

Таблица 3 – Описание полей класса `AccountingCellFormatting`

Поле	Тип	Описание
<code>AccountingCellFormatting.decimalPlaces</code>	int	Количество десятичных позиций
<code>AccountingCellFormatting.symbol</code>	string	Символ денежной единицы
<code>AccountingCellFormatting.localeCode</code>	int	Идентификатор кода языка (MS-LCID)

Поле	Тип	Описание
AccountingCellFormatting.fillSymbol	string	Символ заполнения
AccountingCellFormatting.useThousandsSeparator	bool	Использовать разделитель для тысячных
AccountingCellFormatting.currencySignPlacement	CurrencySignPlacement	Тип размещения знака валюты

Пример

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

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

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

6.4 Перечисление Alignment

Перечисление Alignment содержит варианты горизонтального выравнивания текста, в том числе в ячейке таблицы. Используется в поле [ParagraphProperties.alignment](#).

Таблица 4 – Варианты горизонтального выравнивания текста

Значение	Описание
Alignment_Default	Выравнивание текста по умолчанию
Alignment_Left	Выравнивание текста по левому краю
Alignment_Center	Выравнивание текста по центру
Alignment_Right	Выравнивание по правому краю
Alignment_Justify	Выравнивание по ширине
Alignment_Distributed	Распределенное выравнивание, при применении которого между словами добавляются пробелы так, чтобы оба края каждой строки были выровнены по обеим сторонам. Последняя строка в абзаце также

Значение	Описание
	выравнивается по обеим сторонам, но если строка состоит из одного слова, то выравнивание по правой стороне не осуществляется
Alignment_Fill	Распределение текста по горизонтали – заполнение строки текстом

Пример

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

6.5 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.

6.5.1 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")
```

6.5.2 Метод `Application.getMessenger`

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

Пример

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

6.5.3 Метод Application.loadDocument

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

Используется один из следующих вариантов метода:

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

Примеры загрузки текстового документа

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

Примеры загрузки табличного документа

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

6.6 Класс BinaryConditionalFormatOperator

Класс `BinaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил "Между" и "Не между".

Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).
В качестве аргументов могут использоваться как значения, так и формулы.

Конструктор

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

Пример

Количество
2
5
10
1
3
4
6
15
20
2
7

Рисунок 20 – Пример создания правила "Не между"

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

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

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

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

binaryRule = myOfficeSDK.ConditionalFormatRule(binaryOperator, binaryStyle,
cellRangePosition, False)
rules.addRule(binaryRule)
```

6.6.1 Метод `BinaryConditionalFormatOperator.getCondition`

Метод возвращает условие применения форматирования.

Вызов

```
ConditionalFormatBinaryCondition getCondition()
```

Возвращает

– условие применения форматирования, тип [ConditionalFormatBinaryCondition](#).

6.6.2 Метод `BinaryConditionalFormatOperator.getFirstArgument`

Метод возвращает первый аргумент условия.

Вызов

```
string getFirstArgument()
```

Возвращает

– первый аргумент, тип `string`.

6.6.3 Метод `BinaryConditionalFormatOperator.getSecondArgument`

Метод возвращает второй аргумент условия.

Вызов

```
string getSecondArgument()
```

Возвращает

– второй аргумент, тип `string`.

6.6.4 Метод `BinaryConditionalFormatOperator.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.7 Класс Block

Класс Block является базовой для всех блоков документа. От нее наследуются классы [Paragraph](#), [Table](#), [Shape](#), [Field](#) (см. Рисунок 21).

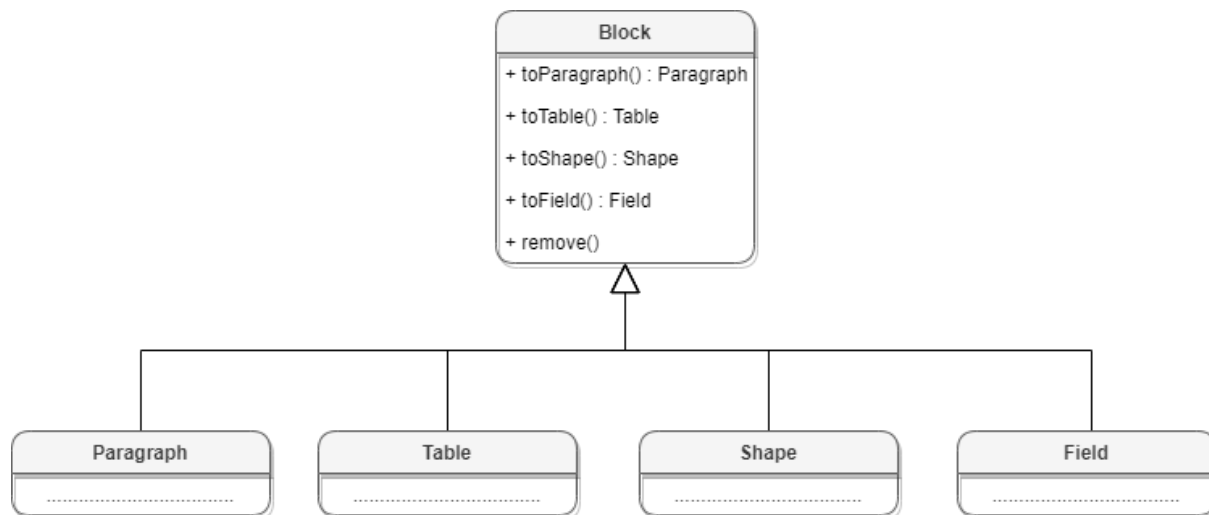


Рисунок 21 – Объектная модель класса Block

6.7.1 Метод Block.getRange

Возвращает диапазон [Range](#), в котором содержится данный блок.

Пример

```

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

6.7.2 Метод Block.getSection

Метод возвращает раздел [Section](#), содержащий блок.

Пример

```

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

6.7.3 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()
```

6.7.4 Методы `toParagraph`, `toTable`, `toShape`, `toField`

Преобразует объект [Block](#) в объект соответствующего типа.

Пример

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

6.8 Класс `Blocks`

Класс `Blocks` обеспечивает доступ к блокам [Block](#) документа или диапазона документа (см. Рисунок 22). Объект класса `Blocks` может быть получен вызовом метода [Document.getBlocks](#) или [HeaderFooter.getBlocks](#).

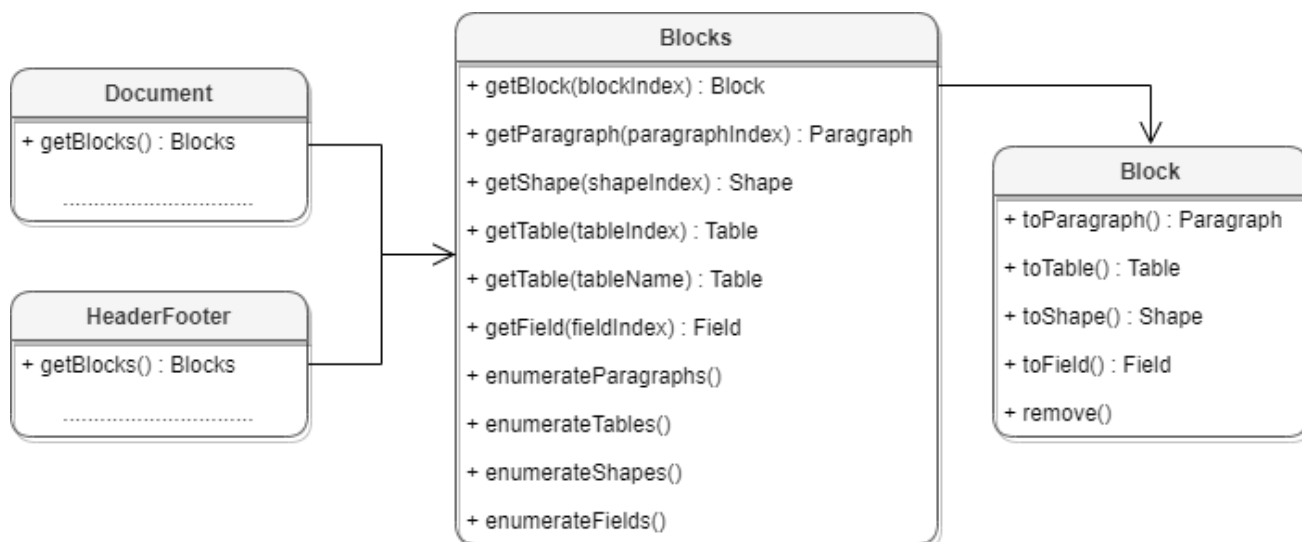


Рисунок 22 – Объектная модель класса Blocks

6.8.1 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())

```

6.8.2 Метод **Blocks.GetEnumerator**

Позволяет реализовать перечисление объектов [Block](#).

Пример

```

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

```

6.8.3 Метод `Blocks.getField`

Возвращает объект типа [Field](#) по заданному индексу.

Пример

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

6.8.4 Метод `Blocks.getFieldsEnumerator`

Позволяет перечислить объекты типа [Field](#).

Пример

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

6.8.5 Метод `Blocks.getParagraph`

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

Пример

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

6.8.6 Метод `Blocks.getParagraphsEnumerator`

Позволяет реализовать перечисление абзацев [Paragraph](#).

Пример

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

6.8.7 Метод `Blocks.getShape`

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

Пример

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

6.8.8 Метод `Blocks.getShapesEnumerator`

Позволяет перечислить объекты типа [Shape](#).

Пример

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

6.8.9 Метод `Blocks.getTable`

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

Пример

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

В качестве параметра метода также можно указать имя таблицы.

Пример

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

6.8.10 Метод `Blocks.getTablesEnumerator`

Позволяет перечислить объекты типа [Table](#).

Пример

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

6.9 Класс `Bookmarks`

Предоставляет доступ к операциям с закладками в документе (см. [Работа с закладками](#)).

6.9.1 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())
```

6.9.2 Метод `Bookmarks.removeBookmark`

Удаляет закладку по ее названию.

Пример

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

6.10 Класс `Borders`

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

Таблица 5 – Описание методов класса Borders

Метод	Описание
<code>Borders setLeft(lineProperties)</code>	Установка левой границы ячейки, метод возвращает Borders
<code>Borders setRight(lineProperties)</code>	Установка правой границы ячейки, метод возвращает Borders
<code>Borders setTop(lineProperties)</code>	Установка верхней границы ячейки, метод возвращает Borders
<code>Borders setBottom(lineProperties)</code>	Установка нижней границы ячейки, метод возвращает Borders
<code>Borders setDiagonalDown(lineProperties)</code>	Установка диагональной линии, метод возвращает Borders 
<code>Borders setDiagonalUp(lineProperties)</code>	Установка диагональной линии, метод возвращает Borders 
<code>Borders setOuter(lineProperties)</code>	Установка внешних границ ячейки, метод возвращает Borders
<code>Borders setDiagonals(lineProperties)</code>	Установка обеих типов диагональных линий одновременно, метод возвращает Borders
<code>Borders setInnerHorizontal(lineProperties)</code>	Установка внутренних горизонтальных границ ячейки, метод возвращает Borders
<code>Borders setInnerVertical(lineProperties)</code>	Установка внутренних вертикальных границ ячейки, метод возвращает Borders
<code>Borders setInner(lineProperties)</code>	Установка внутренних границ ячейки, метод возвращает Borders
<code>Borders setAll(lineProperties)</code>	Установка всех границ ячейки, метод возвращает Borders
<code>LineProperties getLeft()</code>	Получение левой границы ячейки
<code>LineProperties getRight()</code>	Получение правой границы ячейки
<code>LineProperties getTop()</code>	Получение верхней границы ячейки
<code>LineProperties getBottom()</code>	Получение нижней границы ячейки
<code>LineProperties getDiagonalDown()</code>	Получение диагональной линии
<code>LineProperties getDiagonalUp()</code>	Получение диагональной линии

Метод	Описание
<code>LineProperties getInnerHorizontal()</code>	Получение внутренних горизонтальных границ ячейки
<code>LineProperties getInnerVertical()</code>	Получение внутренних вертикальных границ ячейки
<code>bool isEmpty()</code>	Возвращает True, если не установлена ни одна граница 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).setRight(lineProperties)
borders.setTop(lineProperties).setBottom(lineProperties)
cell.setBorders(borders)
```

6.11 Перечисление CalculationMode

Режимы пересчета формул в документе представлены в таблице 6. Перечисление `CalculationMode` используется в методах [Document.getCalculationMode\(\)](#) и [Document.setCalculationMode\(\)](#).

Таблица 6 – Режимы пересчета формул

Значение	Описание
<code>CalculationMode_Auto</code>	Формулы пересчитываются автоматически при изменении данных.
<code>CalculationMode_Manual</code>	Формулы пересчитываются вручную.

6.12 Перечисление CaseSensitive

Параметры настройки учета регистра при поиске (см. метод [Search.FindText\(\)](#) и поле [TableSearchSettings.caseSensitive](#)) представлены в таблице 7.

Таблица 7 – Параметры регистра при поиске

Значение	Описание
CaseSensitive_Yes	Поиск с учетом регистра
CaseSensitive_No	Поиск без учета регистра

6.13 Класс Cell

Класс `Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 23).

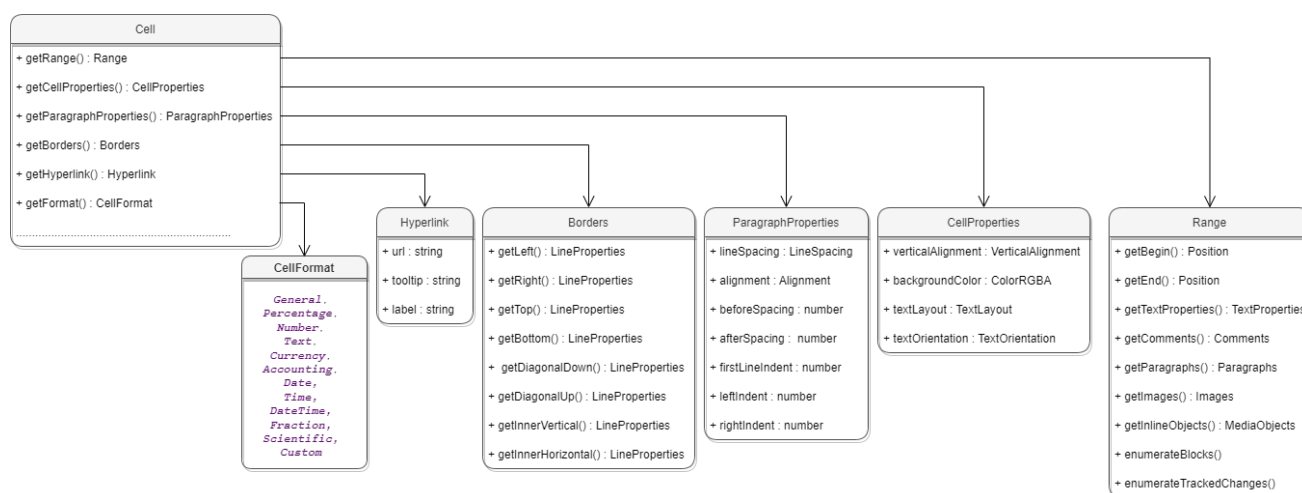


Рисунок 23 – Объектная модель ячейки таблиц

6.13.1 Метод `Cell.checkDataValidation`

Метод проверяет значение ячейки согласно заданной в ней проверке данных.

Вызов

```
DataValidationResult checkDataValidation()
```

Возвращает

– результат проверки значения ячейки, тип [DataValidationResult](#).

Пример

```

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

6.13.2 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)
```

6.13.3 Метод `Cell.getCellProperties`

Позволяет получить свойства [CellProperties](#) ячейки.

Пример

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

6.13.4 Метод `Cell.getColumnIndex`

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

Пример

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("A3")
print(cell.getColumnIndex()) # 0
print(cell.getRowIndex()) # 2
```

6.13.5 Метод `Cell.getCurrentRegion`

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

Вызов

```
CellRange getCurrentRegion()
```

Возвращает

— диапазон ячеек, тип [CellRange](#).

Пример

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

6.13.6 Метод `Cell.getCustomFormat`

Возвращает строку формата ячейки.

Пример

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

6.13.7 Метод `Cell.getDataValidation`

Метод возвращает настройки проверки данных для текущей ячейки.

Вызов

```
DataValidation getDataValidation()
```

Возвращает

– настройки проверки данных, тип [DataValidation](#).

Пример

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("E4")
cellDV = cell.getDataValidation()
print(cellDV.getPrompt())
print(cellDV.getFormula1())
```

6.13.8 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())
```

6.13.9 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())
```

6.13.10 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())
```

6.13.11 Метод Cell.getHyperlink

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

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

6.13.12 Метод Cell.getMergedRange

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

Вызов

```
CellRange getMergedRange()
```

Возвращает

— диапазон ячеек, тип [CellRange](#).

Пример

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

6.13.13 Метод Cell.getNote

Метод возвращает текст заметки для текущей ячейки.

Вызов

```
string getNote()
```

Возвращает

— текст заметки.

Пример

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

Методы [Cell.setNote](#) и [Cell.removeNote](#) позволяют добавить и удалить заметку.

6.13.14 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)
```

6.13.15 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())
```

6.13.16 Метод `Cell.getProtectionProperties`

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

Вызов

```
CellProtectionProperties getProtectionProperties()
```

Возвращает

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

Пример

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

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

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

6.13.17 Метод `Cell.getRange`

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

6.13.18 Метод `Cell.getRawValue`

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

Пример

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

6.13.19 Метод `Cell.getResolvedBorders`

Метод возвращает границы ячейки с учетом границ окружающих ячеек.

Вызов

```
Borders getResolvedBorders()
```

Возвращает

– границы ячейки, тип [Borders](#).

Пример

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

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

cell = sheet.getCell("A2")
cell.getBorders().getRight() # None
cell.getResolvedBorders().getRight().width # 1,5
```

6.13.20 Метод `Cell.getRowIndex`

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

Пример

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

```
print(cell.getColumnIndex()) # 0
print(cell.getRowIndex()) # 2
```

6.13.21 Метод Cell.getTable

Метод возвращает таблицу [Table](#), в которой находится текущая ячейка.

Вызов

```
Table getTable()
```

Пример

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

6.13.22 Метод Cell.getTextProperties

Метод возвращает текущие настройки форматирования текста для ячейки.

Вызов

```
TextProperties getTextProperties()
```

Возвращает

— свойства форматирования текста, тип [TextProperties](#).

Пример

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

props = cell.getTextProperties()
props.backgroundColor = myOfficeSDK.ColorRGBA(0, 0, 255, 200)
props.textColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))

cell.setTextProperties(props)
```

Метод [Cell.setTextProperties](#) позволяет задать настройки форматирования текста в ячейке.

6.13.23 Метод Cell.isEmpty

Метод позволяет определить содержит ли ячейка данные.

Вызов

```
bool isEmpty()
```

Возвращает

– True, если ячейка не содержит данные, в ином случае – False.

Пример

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

cell1.isEmpty() # False
cell1.isEmpty() # False

cell2.isEmpty() # False
cell2.isEmpty() # True

cell3.isEmpty() # True
cell3.isEmpty() # True
```

6.13.24 Метод Cell.isEmpty

Метод позволяет определить содержит ли ячейка данные или настройки форматирования.

Вызов

```
bool isEmpty()
```

Возвращает

– True, если ячейка не содержит данные или настройки форматирования, в ином случае – False.

Пример

```
sheet = document.getBlocks().getTable(0)
cell11 = sheet.getCell("A1")
cell12 = sheet.getCell("A2")
cell13 = sheet.getCell("A3")
cell11.setText("123")
props = myOfficeSDK.TextProperties()
props.bold = True
cell12.setTextProperties(props)

cell11.isEmpty() # False
cell11.isContentEmpty() # False

cell12.isEmpty() # False
cell12.isContentEmpty() # True

cell13.isEmpty() # True
cell13.isContentEmpty() # True
```

6.13.25 Метод Cell.isInMergedRange

Метод позволяет определить находится ли ячейка в объединенном диапазоне.

Вызов

```
bool isInMergedRange()
```

Возвращает

– True, если ячейка находится в объединенном диапазоне, в ином случае – False.

Пример

```
sheet = document.getBlocks().getTable(0)
sheet.getCellRange("A1:C1").merge()
cell11 = sheet.getCell("B1")
cell12 = sheet.getCell("B2")

print(cell11.isInMergedRange()) # True
print(cell12.isInMergedRange()) # False
```

6.13.26 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())
```

6.13.27 Метод `Cell.isProtected`

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

Вызов

```
bool isProtected()
```

6.13.28 Метод `Cell.removeNote`

Метод удаляет заметку из текущей ячейки.

Вызов

```
removeNote()
```

Пример

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("A3")
if cell.getNote() != None:
    cell.removeNote()
```

Методы [Cell.setNote](#) и [Cell.getNote](#) позволяют добавить заметку и получить её текст.

6.13.29 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())
```

6.13.30 Метод Cell.setBorders

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

6.13.31 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)
```

6.13.32 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")
```

6.13.33 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())
```

6.13.34 Метод Cell.setFormat

Метод устанавливает формат ячейки. Существуют несколько вариантов использования метода.

Варианты вызова метода

```
setFormat(cellFormat)
```

Где **cellFormat** – формат ячейки типа [CellFormat](#).

```
setFormat(accountingCellFormatting)
```

Где **accountingCellFormatting** – формат ячейки типа [AccountingCellFormatting](#).

```
setFormat(percentageCellFormatting)
```

Где **percentageCellFormatting** – формат ячейки типа [PercentageCellFormatting](#).

```
setFormat(numberCellFormatting)
```

Где **numberCellFormatting** – формат ячейки типа [NumberCellFormatting](#).

```
setFormat(currencyCellFormatting)
```

Где **currencyCellFormatting** – формат ячейки типа [CurrencyCellFormatting](#).

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

Где **dateTimeCellFormatting** – формат ячейки типа [DateTimeCellFormatting](#),
typeFormat - формат даты/времени типа [CellFormat](#).

```
setFormat(fractionCellFormatting)
```

Где **fractionCellFormatting** – формат ячейки типа [FractionCellFormatting](#).

```
setFormat(scientificCellFormatting)
```

Где **scientificCellFormatting** – формат ячейки типа [ScientificCellFormatting](#).

Примеры использования

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

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

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

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

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

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

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

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

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

6.13.35 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](#).

6.13.36 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())
```

6.13.37 Метод `Cell.setNote`

Метод добавляет заметку в текущую ячейку.

Вызов

```
setNote(noteText)
```

Параметры

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

Пример

```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("A3")
if cell.getNote() == None:
    cell.setNote("New Note")
```

Методы [Cell.getNote](#) и [Cell.removeNote](#) позволяют получить текст заметки или удалить её.

6.13.38 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())
```

6.13.39 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)
```

6.13.40 Метод Cell.setProtectionProperties

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

Вызов

```
setProtectionProperties(protectionProps)
```

Параметры

— protectionProps: свойства защиты ячейки, тип [CellProtectionProperties](#).

Пример

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

cellProps = cell.getProtectionProperties()
```

```
cellProps.lockedForChanges = False
cellProps.formulasNotDisplayed = False

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

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

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

6.13.41 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())
```

6.13.42 Метод `Cell.setTextProperties`

Метод задает настройки форматирования текста для ячейки.

Вызов

```
setTextProperties(props)
```

Параметры

— `props`: свойства форматирования текста, тип [TextProperties](#).

Пример

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

props = cell.getTextProperties()
props.backgroundColor = myOfficeSDK.ColorRGBA(0, 0, 255, 200)
```

```
props.textColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))

cell.setTextProperties(props)
```

Метод [Cell.getTextProperties](#) позволяет получить текущие настройки форматирования текста в ячейке.

6.13.43 Метод Cell.unmerge

Разъединяет несколько ячеек, которые были объединены ранее.

Пример

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

6.14 Перечисление CellFormat

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

Таблица 8 – Поддерживаемые форматы ячеек таблицы

Значение	Описание
CellFormat_General	Формат ячейки «Общий». В этом формате в ячейке отображаются первые 9 символов числа, остальные доступны для просмотра в строке формул. Для дробных чисел в формате «Общий» незначащие нули в дробной части не отображаются. Числа, состоящие более чем из 12 символов, переводятся в экспоненциальную форму после завершения ввода в ячейку.
CellFormat_Percentage	Формат ячейки «Процентный». Этот формат используется для представления чисел как процентов. При применении формата «Процентный» введенное число умножается на 100 и обозначается знаком «%».
CellFormat_Number	Формат ячейки «Числовой».

Значение	Описание
	Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя.
CellFormat_Text	Формат ячейки «Текстовый».
CellFormat_Currency	Формат ячейки «Денежный». Этот формат используется для представления чисел со знаком или кодом валюты.
CellFormat_Accounting	Формат ячейки «Финансовый». Этот формат применяется для чисел, используемых в бухгалтерских документах. В формате «Финансовый» введенное число автоматически дополняется названием валюты, которая соответствует настройкам системы компьютера. Отрицательные числа в формате «Финансовый» заключаются в круглые скобки в ячейке, а в строке формул остаются в том виде, в котором они были введены.
CellFormat_Date	Формат ячейки «Дата». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ.
CellFormat_Time	Формат ячейки «Время». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.
CellFormat_DateTime	Формат ячейки «Дата + Время»
CellFormat_Fraction	Формат ячейки «Дробный». Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель).
CellFormat_Scientific	Формат ячейки «Экспоненциальный». Экспоненциальный (или научный) формат используется для представления больших чисел в короткой форме. Все введенные числа длиной более 12 символов автоматически переводятся в этот формат. В ячейке число в формате «Экспоненциальный» представлено следующим образом: – целая часть, всегда состоящая из одной цифры; – разделитель целой и дробной части; – дробная часть, по умолчанию состоящая из двух цифр;

Значение	Описание
	– показатель степени числа 10 в виде Е<знак показателя степени><показатель степени> .
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	CellFormat.General	1
2	CellFormat.Percentage	100,00%
3	CellFormat.Number	1,00

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

6.15 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](#).

Конструкторы

```
CellPosition()
```

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

- rowArg – индекс строки ячейки (индексация начинается с нуля), тип int;
- columnArg – индекс колонки ячейки, тип int.

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)
```

6.15.1 Поле `CellPosition.column`

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

Пример

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

6.15.2 Поле `CellPosition.row`

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

Пример

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

6.15.3 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())
```

6.15.4 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")
```

6.15.5 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")
```

6.16 Класс CellProperties

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

Для задания свойств ячеек используются методы [Cell.setCellProperties\(\)](#) и [CellRange.setCellProperties\(\)](#). Для получения свойств ячеек используются методы [Cell.getCellProperties\(\)](#) и [CellRange.getCellProperties\(\)](#). Иерархия классов и полей `CellProperties` отображена на рисунке 24.

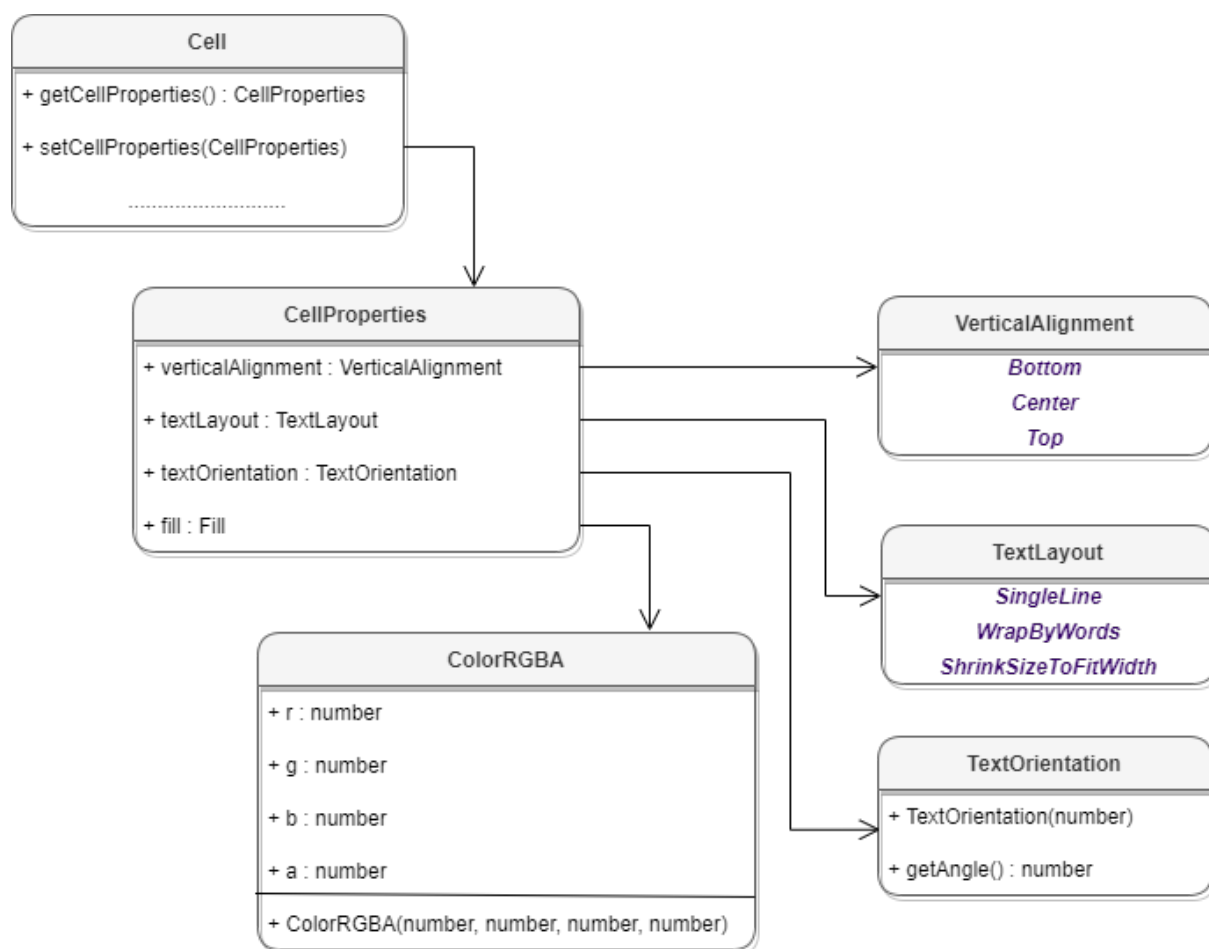


Рисунок 24 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 9 – Описание полей класса `CellProperties`

Поле	Тип	Описание
<code>CellProperties.verticalAlignment</code>	VerticalAlignment	Вертикальное выравнивание в ячейке
<code>CellProperties.textLayout</code>	TextLayout	Способ отображения значения ячейки
<code>CellProperties.fill</code>	Fill	Цвет фона ячейки
<code>CellProperties.textOrientation</code>	TextOrientation	Ориентация текста в ячейке (угол поворота)

Пример

```

cellProps = cell.getCellProperties()
cellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center
cellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth
cellProps.fill = myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255,

```

```
255, 0, 255)))  
cellProps.textOrientation = myOfficeSDK.TextOrientation(45)
```

6.16.1 CellProperties.__eq__

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

Пример

```
firstCellProps = myOfficeSDK.CellProperties()  
firstCellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center  
firstCellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth  
firstCellProps.fill =  
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))  
firstCellProps.textOrientation = myOfficeSDK.TextOrientation(45)  
  
secondCellProps = myOfficeSDK.CellProperties()  
secondCellProps.verticalAlignment = myOfficeSDK.VerticalAlignment_Center  
secondCellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth  
secondCellProps.fill =  
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))  
secondCellProps.textOrientation = myOfficeSDK.TextOrientation(45)  
  
if firstCellProps.__eq__(secondCellProps):  
    print("Equals")
```

6.16.2 CellProperties.__ne__

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

Пример

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

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

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

6.17 Класс CellProtectionProperties

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

Таблица 10 – Описание полей класса `CellProtectionProperties`

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

Пример

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

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

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

6.18 Класс `CellRange`

Класс `CellRange` описывает диапазон ячеек таблицы.

Пример

```
firstSheet = document.getBlocks().getTable("List1")
cellRange = firstSheet.getCellRange("B3:C4")
```

6.18.1 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))
```

6.18.2 Метод `CellRange.clearDataValidations`

Метод убирает проверку данных из текущего диапазона ячеек.

Вызов

```
clearDataValidations()
```

Пример

```
sheet = document.getBlocks().getTable(0)
sheet.getUsedRange().clearDataValidations()
```

6.18.3 Метод `CellRange.clearFormat`

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

Вызов

```
clearFormat(skipHiddenCells)
```

Параметры

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

6.18.4 Метод `CellRange.clearStyle`

Метод очищает форматирование диапазона ячеек.

Вызов

```
clearStyle(skipHiddenCells)
```

Параметры

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

6.18.5 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](#).

6.18.6 Метод `CellRange.copyInto`

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

Данный метод реализован только в табличных документах.

Пример (только для табличного документа)

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

Пример копирования ячеек между листами

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

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

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

sourceRange.copyInto(destRange)
```

При копировании ячеек в качестве новой позиции достаточно указать верхнюю левую ячейку нового диапазона, однако если необходимо продублировать исходный блок ячеек, в качестве параметра следует использовать диапазон, превышающий размеры исходного диапазона, но кратный его размерам. Например, при копировании диапазона "A1:B2" (размер 2x2) в диапазон "B5:E6" (размер 2x4) блок исходных ячеек продублируется два раза (см. Рисунок 25).

	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 25 – Копирование ячеек табличного документа

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

6.18.7 Метод `CellRange.find`

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

Вызов

```
CellsIterator find(string, settings)
```

Параметры

- string: поисковый запрос, тип string.
- settings: (необязательный) параметры поиска, тип [TableSearchSettings](#).

Возвращает

- список ячеек, соответствующих поисковому запросу, тип CellsIterator.

Пример

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

searchProps = myOfficeSDK.TableSearchSettings()
searchProps.caseSensitive = myOfficeSDK.CaseSensitive_No
searchProps.matchBehaviour = myOfficeSDK.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = myOfficeSDK.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = True

results = cellRange.find("*eye", searchProps)
for cell in results:
    print(cell.getFormattedValue()) # Steeleye Stout
```

6.18.8 Метод `CellRange.getAddress`

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

Вызов

```
string getAddress(addressSettings)
```

Параметры

— addressSettings: настройки форматирования адреса, тип [CellRangeAddressSettings](#).

Возвращает

— адрес диапазона ячеек.

Пример

```
sheet = document.getBlocks().getTable(0)
cellRange = sheet.getCellRange("A1:C11")

addressSettings = myOfficeSDK.CellRangeAddressSettings()
addressSettings.isFileNameRequired = True
addressSettings.isWorksheetNameRequired = True
addressSettings.addressFormat = myOfficeSDK.CellRangeAddressFormat_A1
addressSettings.isAbsoluteRow = True
addressSettings.isAbsoluteColumn = False

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

6.18.9 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())
```

6.18.10 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())
```

6.18.11 Метод `CellRange.getCellProperties`

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

Пример

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
cellProperties = cellRange.getCellProperties()
print(cellProperties.fill.getColor().getRGBAColor().r)
```

6.18.12 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())
```

6.18.13 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())
```

6.18.14 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())
```

6.18.15 Метод `CellRange.getParagraphProperties`

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

Вызов

```
ParagraphProperties getParagraphProperties(skipHiddenCells)
```

Параметры

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

Возвращает

- свойства форматирования абзацев, тип [ParagraphProperties](#).

Пример

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

paragraphProperties = cellRange.getParagraphProperties()
paragraphProperties.alignment = myOfficeSDK.Alignment_Center

cellRange.setParagraphProperties(paragraphProperties, False)
```

Свойства возвращаемого объекта [ParagraphProperties](#) могут быть `None`, если выбранный диапазон содержит ячейки с разными параметрами. Метод [CellRange.setParagraphProperties](#) позволяет задать настройки форматирования абзацев, находящихся в диапазоне ячеек.

6.18.16 Метод `CellRange.getProtectionProperties`

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

Вызов

```
CellProtectionProperties getProtectionProperties()
```

Возвращает

- [CellProtectionProperties](#): свойства защиты диапазона ячеек (`None`, если диапазон содержит только ячейки сводной таблицы).

Пример

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

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

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

Свойства возвращаемого объекта [CellProtectionProperties](#) могут быть None, если текущий диапазон содержит ячейки с разными параметрами.

6.18.17 Метод `CellRange.getTable`

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

Пример

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

6.18.18 Метод `CellRange.getTableRange`

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

6.18.19 Метод `CellRange.getTextProperties`

Метод возвращает текущие настройки форматирования текста для диапазона ячеек.

Вызов

```
TextProperties getTextProperties(skipHiddenCells)
```

Параметры

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

Возвращает

- свойства форматирования текста, тип [TextProperties](#).

Пример

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

props = cellRange.getTextProperties()
props.backgroundColor = myOfficeSDK.ColorRGBA(0, 0, 255, 200)
props.textColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))

cellRange.setTextProperties(props)
```

Свойства возвращаемого объекта [TextProperties](#) могут быть None, если выбранный диапазон содержит ячейки с разными параметрами. Метод [CellRange.setTextProperties](#) позволяет задать настройки форматирования текста в диапазоне ячеек.

6.18.20 Метод CellRange.insert

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

Вызов

```
insert(shiftAxis)
```

Параметры

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

Пример

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

cellRange.insert(myOfficeSDK.CellShiftAxis_Vertical)
```

6.18.21 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)
```

6.18.22 Метод `CellRange.intersect`

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

Вызов

```
CellRange intersect(other)
```

Параметры

– other: диапазон ячеек, тип [CellRange](#).

Возвращает

– диапазон пересечения, тип [CellRange](#).

– None, если диапазоны ячеек не пересекаются.

Пример

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

6.18.23 Метод `CellRange.isEmpty`

Метод позволяет определить содержат ли ячейки диапазона данные.

Вызов

```
bool isEmpty()
```

Возвращает

– True, если все ячейки диапазона не содержат данные, в ином случае – False.

Пример

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

range1.isEmpty() # False
range1.isContentEmpty() # False

range2.isEmpty() # False
range2.isContentEmpty() # True

range3.isEmpty() # True
range3.isContentEmpty() # True
```

6.18.24 Метод `CellRange.isEmpty`

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

Вызов

```
bool isEmpty()
```

Возвращает

— True, если все ячейки диапазона не содержат данные или настройки форматирования, в ином случае — False.

Пример

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

range1.isEmpty() # False
```

```
range1.isEmpty() # False
range1.isContentEmpty() # False

range2.isEmpty() # False
range2.isContentEmpty() # True

range3.isEmpty() # True
range3.isContentEmpty() # True
```

6.18.25 Метод `CellRange.isProtected`

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

Вызов

```
bool isProtected()
```

Метод `isProtected()` возвращает `None`, если часть ячеек диапазона защищена от редактирования, а часть – нет.

6.18.26 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()
```

6.18.27 Метод `CellRange.moveInto`

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

Данный метод реализован только в табличных документах.

Пример (только для табличного документа)

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

Пример перемещения ячеек между документами

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

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

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

sourceRange.moveInto(destRange)
```

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

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

6.18.28 Метод CellRange.remove

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

Вызов

```
remove(shiftAxis)
```

Параметры

– shiftAxis: направление сдвига ячеек (влево или вверх), тип [CellShiftAxis](#).

Пример

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

cellRange.remove(myOfficeSDK.CellShiftAxis_Horizontal)
```

6.18.29 Метод `CellRange.removeContent`

Метод очищает содержимое диапазона ячеек.

Вызов

```
removeContent(skipHiddenCells)
```

Параметры

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

6.18.30 Метод `CellRange.setArrayFormula`

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

Вызов

```
setArrayFormula(arrayFormula)
```

Параметры

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

Примеры

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

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

6.18.31 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)
```

6.18.32 Метод `CellRange.setCellProperties`

Метод предназначен для установки свойств [CellProperties](#) ячеек диапазона.

Пример

```
firstSheet = document.getBlocks().getTable("List11")
cellRange = firstSheet.getCellRange("B3:C4")
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 255, 0, 255)))
cellRange.setCellProperties(cellProperties)
```

6.18.33 Метод `CellRange.setDataValidation`

Метод устанавливает проверку данных для текущего диапазона ячеек.

Вызов

```
setDataValidation(dataValidation)
```

Параметры

— `dataValidation`: настройки проверки данных, тип [DataValidation](#).

Пример

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

cellRange.setDataValidation(dvList)
```

6.18.34 Метод `CellRange.setParagraphProperties`

Метод задает настройки форматирования абзацев, находящихся в диапазоне ячеек.

Вызов

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

Параметры

- `paraProperties`: свойства форматирования абзацев, тип [ParagraphProperties](#).
- `skipHiddenCells`: (необязательный, по умолчанию `True`) не применять настройки к отфильтрованным и скрытым ячейкам диапазона, тип `bool`.

Пример

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

paragraphProperties = cellRange.getParagraphProperties()
paragraphProperties.alignment = myOfficeSDK.Alignment_Center

cellRange.setParagraphProperties(paragraphProperties, False)
```

Метод [CellRange.getParagraphProperties](#) позволяет получить текущие настройки форматирования абзацев, находящихся в диапазоне ячеек.

6.18.35 Метод `CellRange.setProtectionProperties`

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

Вызов

```
setProtectionProperties(protectionProps)
```

Параметры

– protectionProps: свойства защиты диапазона ячеек, тип [CellProtectionProperties](#).

Пример

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

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

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

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

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

6.18.36 Метод `CellRange.setTextProperties`

Метод задает настройки форматирования текста для диапазона ячеек.

Вызов

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

Параметры

- textProperties: свойства форматирования текста, тип [TextProperties](#).
- skipHiddenCells: (необязательный, по умолчанию True) не применять настройки к отфильтрованным и скрытым ячейкам диапазона, тип `bool`.

Пример

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

```
props = cellRange.getTextProperties()  
props.backgroundColor = myOfficeSDK.ColorRGBA(0, 0, 255, 200)  
props.textColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))  
  
cellRange.setTextProperties(props)
```

Метод [CellRange.getTextProperties](#) позволяет получить текущие настройки форматирования текста в диапазоне ячеек.

6.18.37 Метод CellRange.sort

Метод сортирует строки текущего диапазона в соответствии с заданными условиями.

Вызов

```
sort(sortingConditions)
```

Параметры

– `sortingConditions`: коллекция условий сортировки ячеек, тип [SortingConditions](#).

Пример

```
sheet = document.getBlocks().getTable(0)  
range = sheet.getCellRange("A1:C11")  
  
conditions = myOfficeSDK.SortingConditions()  
conditions.add(0, myOfficeSDK.SortingDirection_Descending)  
conditions.add(1, myOfficeSDK.SortingDirection_Ascending)  
  
range.sort(conditions)
```

Метод вызывает `DocumentModificationError` в следующих случаях:

- Индекс столбца выходит за пределы текущего диапазона
- Диапазон содержит объединенные ячейки
- Диапазон содержит формулы
- В диапазоне присутствуют ячейки сводной таблицы
- Диапазон заблокирован от изменений

6.18.38 Метод CellRange.textToColumns

Метод разделяет текст в каждой ячейке диапазона на отдельные горизонтальные ячейки. Символы-разделители и настройки разделения задаются с помощью объекта

[TextToColumnsSettings](#). Данный метод можно использовать только у диапазона шириной в одну ячейку.

Вызов

```
textToColumns(settings)
```

Параметры

– settings: параметры разделения текста по ячейкам, тип [TextToColumnsSettings](#).

Пример

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

Рисунок 27 – Разделение текста диапазона по ячейкам

```
sheet = document.getBlocks().getTable(0)
cell11 = sheet.getCell("A1")
cell12 = sheet.getCell("A2")
cell13 = sheet.getCell("A3")
cell11.setText("1, Laptop, Moscow, 1500, 8, 'fragile, express'")
cell12.setText("2, UPS, 'Saint Petersburg', 600, 12, 'heavy, premium'")
cell13.setText("3, Mouse, Kazan, 200, 5, 'free shipping, standard'")

settings = myOfficeSDK.TextToColumnsSettings()
settings.useCommaDelimiter = True
settings.useSpaceDelimiter = True
settings.treatMultipleDelimitersAsOne = True
settings.textQualifier =
myOfficeSDK.TextToColumnsSettings.TextQualifier_SingleQuotes

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

6.19 Перечисление CellRangeAddressFormat

Перечисление `CellRangeAddressFormat` содержит форматы отображения адреса диапазона ячеек. Используется в поле [CellRangeAddressSettings.addressFormat](#).

Таблица 11 – Описание форматов отображения адреса

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

6.20 Класс CellRangeAddressSettings

Класс `CellRangeAddressSettings` предназначен для настройки параметров отображения адреса диапазона ячеек. Данный класс используется в методе [`CellRange.getAddress\(\)`](#).

Таблица 12 – Описание полей класса CellRangeAddressSettings

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

Пример

```
sheet = document.getBlocks().getTable(0)
cellRange = sheet.getCellRange("A1:C11")

addressSettings = myOfficeSDK.CellRangeAddressSettings()
addressSettings.isFileNameRequired = True
addressSettings.isWorksheetNameRequired = True
addressSettings.addressFormat = myOfficeSDK.CellRangeAddressFormat_A1
addressSettings.isAbsoluteRow = True
addressSettings.isAbsoluteColumn = False

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

6.21 Класс CellRangePosition

Класс `CellRangePosition` представляет положение диапазона ячеек в таблице. Используется в методах [Table.getCellRange\(\)](#), [CellRange.getTableRange\(\)](#) и [Chart.setRange\(\)](#). По умолчанию диапазон включает одну ячейку в позиции 0,0 что соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.

Конструкторы

```
CellRangePosition()
```

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

– `topLeftArg` – позиция левой верхней ячейки диапазона, тип [CellPosition](#);

– `bottomRightArg` – позиция правой нижней ячейки диапазона, тип [CellPosition](#).

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

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

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

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

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

Таблица 13 – Поля класса `CellRangePosition`

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

Примеры

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

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

6.21.1 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())
```

6.21.2 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")
```

6.21.3 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")
```

6.22 Перечисление CellShiftAxis

Перечисление `CellShiftAxis` содержит направления сдвига ячеек при вставке или удалении диапазона. Используется в методах [CellRange.insert\(\)](#) и [CellRange.remove\(\)](#).

Таблица 14 – Описание направлений сдвига ячеек

Значение	Описание
<code>CellShiftAxis_Horizontal</code>	Сдвигает ячейки по горизонтали (вправо при вставке, влево при удалении)
<code>CellShiftAxis_Vertical</code>	Сдвигает ячейки по вертикали (вниз при вставке, вверх при удалении)

Пример

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

cellRange.insert(myOfficeSDK.CellShiftAxis_Vertical)
```

6.23 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 28.

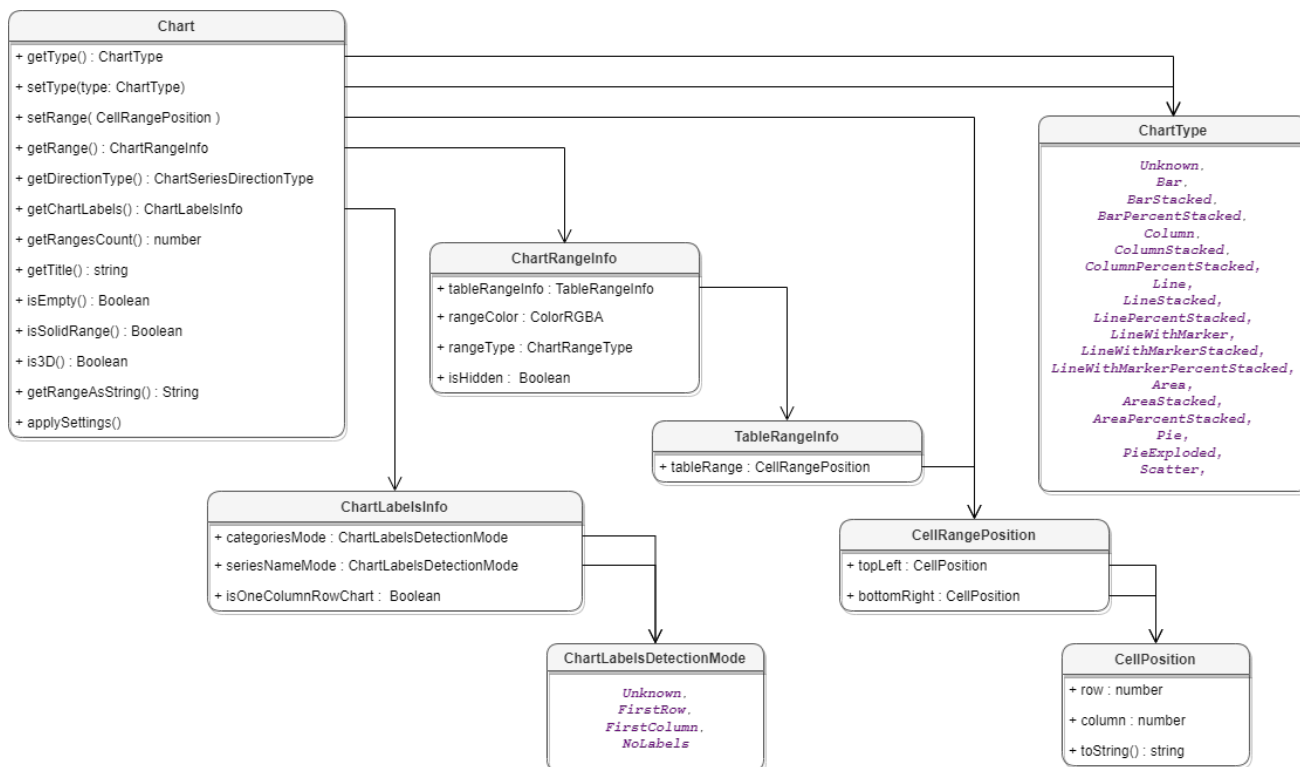


Рисунок 28 – Объектная модель класса Chart

6.23.1 Метод Chart.applySettings

Метод позволяет обновить параметры текущей выбранной диаграммы.

Вызов

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

Параметры

- cellRange – обновленный диапазон исходных данных диаграммы [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)
```

6.23.2 Метод Chart.getChartLabels

Метод возвращает коллекцию меток диаграммы типа [ChartLabelsInfo](#).

Пример

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

6.23.3 Метод Chart.getDirectionType

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

Пример

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

6.23.4 Метод Chart.getFrame

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

Пример

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

6.23.5 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)
```

6.23.6 Метод Chart.getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

Пример

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

6.23.7 Метод Chart.getRangesCount

Метод возвращает количество серий диаграммы.

Пример

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

6.23.8 Метод Chart.getTitle

Метод возвращает заголовок диаграммы.

Пример

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

6.23.9 Метод Chart.getType

Метод возвращает тип диаграммы [ChartType](#).

Пример

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

6.23.10 Метод Chart.is3D

Метод возвращает True, если диаграмма трехмерная.

Пример

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

6.23.11 Метод Chart.isEmpty

Метод возвращает True, если диаграмма не содержит значений.

Пример

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

6.23.12 Метод Chart.isSolidRange

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

Пример

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

6.23.13 Метод Chart.setRange

Метод задает диапазон [CellRangePosition](#) ячеек с исходными данными для диаграммы.

Пример

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

6.23.14 Метод Chart.setRect

Метод задает область расположения диаграммы, параметр `rect` – новая область.



Внимание ! Метод устаревший (deprecated), оставлен для обратной совместимости и не рекомендован к использованию.

Пример

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

6.23.15 Метод Chart.setType

Метод устанавливает тип диаграммы [ChartType](#). В качестве параметра передается новый тип диаграммы.

Пример

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

6.24 Перечисление ChartLabelsDetectionMode

Перечисление `ChartLabelsDetectionMode` содержит режимы автоматического определения меток диаграмм. Используется в полях [ChartLabelsInfo.categoriesMode](#) и [ChartLabelsInfo.seriesNameMode](#).

Таблица 15 – Режимы автоматического определения меток диаграмм

Значение	Описание
<code>ChartLabelsDetectionMode_Unknown</code>	Неопределенный тип
<code>ChartLabelsDetectionMode_FirstRow</code>	Метка на первой строке
<code>ChartLabelsDetectionMode_FirstColumn</code>	Метка на первой колонке
<code>ChartLabelsDetectionMode_NoLabels</code>	Не отрисовывать метки

Пример

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

```
chartLabels = chart.getChartLabels()
print(chartLabels.categoriesMode, chartLabels.seriesNameMode)
```

6.25 Класс ChartLabelsInfo

Класс ChartLabelsInfo описывает настройки автоматического определения меток диаграммы. Используется в методах [Chart.applySettings\(\)](#) и [Chart.getChartLabels\(\)](#).

Конструктор

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

Параметры

- 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.

Таблица 16 – Описание полей класса ChartLabelsInfo

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

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)
```

6.26 Класс ChartRangeInfo

Класс `ChartRangeInfo` описывает серию диаграммы. Используется в методе [Chart.getRange\(\)](#).

Конструктор

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

Таблица 17 – Описание полей класса `ChartRangeInfo`

Поле	Тип	Описание
<code>ChartRangeInfo.cellRange</code>	CellRange	Диапазон ячеек диаграммы
<code>ChartRangeInfo.rangeColor</code>	ColorRGBA	Цвет для отрисовки серии
<code>ChartRangeInfo.isHidden</code>	<code>bool</code>	Задаёт видимость серии диаграммы
<code>ChartRangeInfo.rangeType</code>	ChartRangeType	Тип диапазона диаграммы

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);
```

6.27 Перечисление ChartRangeType

Перечисление `ChartRangeType` содержит типы диапазонов исходных данных диаграммы. Используется в поле [ChartRangeInfo.rangeType](#).

Таблица 18 – Типы диапазонов

Значение	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])
```

6.28 Класс Charts

Класс `Charts` обеспечивает доступ к списку диаграмм (см. Рисунок 29) табличного документа. Доступ к списку диаграмм осуществляется с помощью метода [`Table.getCharts\(\)`](#).

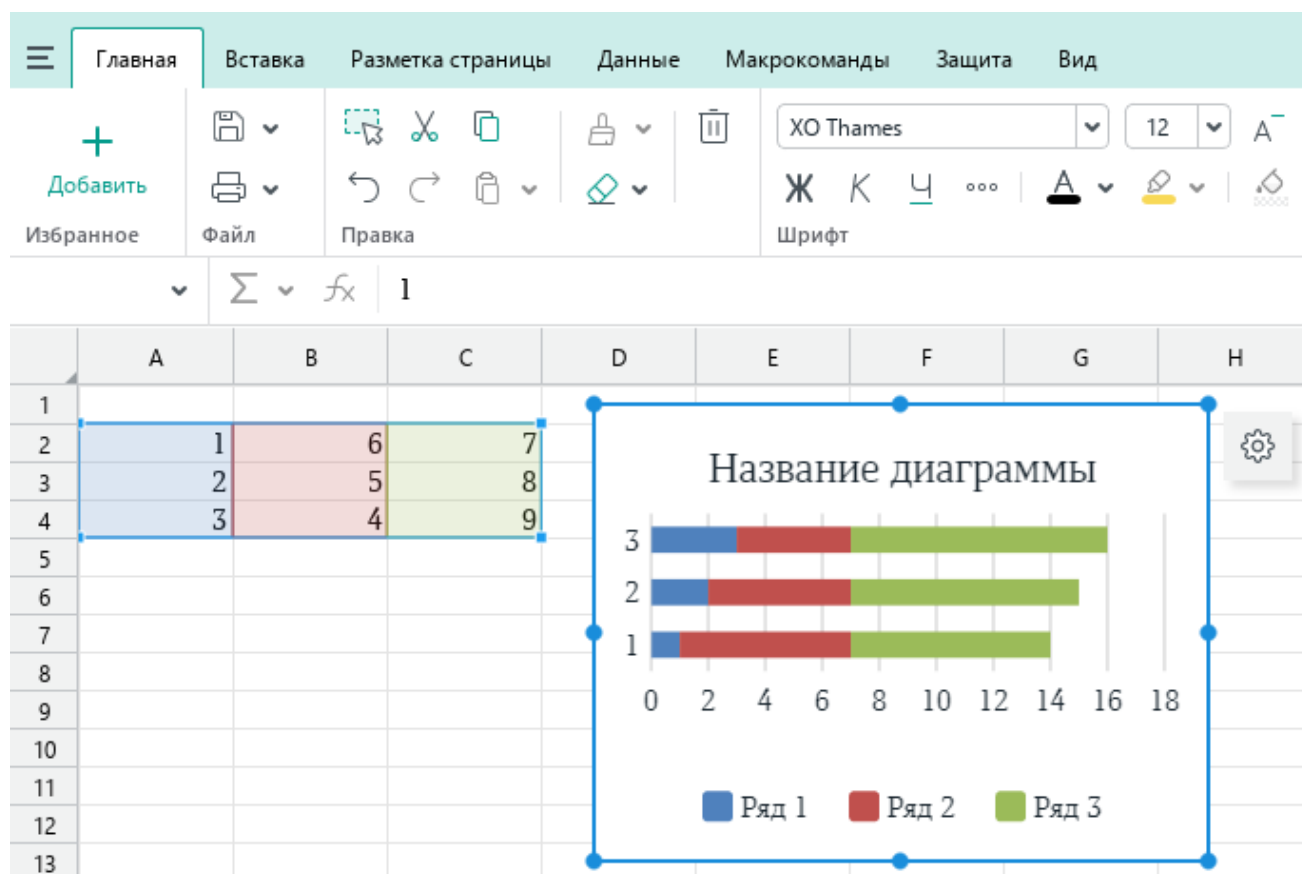


Рисунок 29 – Пример отображения диаграммы в приложении «МояТаблица»

6.28.1 Метод Charts.getChart

Метод возвращает диаграмму [Chart](#) по индексу chartIndex в коллекции диаграмм.

Пример

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

6.28.2 Метод Charts.getChartIndexByDrawingIndex

Метод возвращает индекс диаграммы по индексу отрисовки drawingIndex.

Пример

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

```
chartIndexByDrawingIndex = charts.getChartIndexByDrawingIndex(0)
print(chartIndexByDrawingIndex)
```

6.28.3 Метод Charts.getChartsCount

Метод возвращает общее количество диаграмм в табличном документе.

Пример

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

6.29 Перечисление ChartSeriesDirectionType

Перечисление ChartSeriesDirectionType содержит направления серий диаграмм. Используется в методах [Chart.applySettings\(\)](#) и [Chart.getDirectionType\(\)](#).

Таблица 19 – Направления серий диаграмм

Значение	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])
```

6.30 Перечисление ChartType

Перечисление ChartType содержит все поддерживаемые типы диаграмм. Используется в методах [Chart.getType\(\)](#) и [Chart.setType\(\)](#).

Таблица 20 – Типы диаграмм

Значение	Описание
ChartType_Unknown	Неопределенный тип

Значение	Описание
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())
```

6.31 Класс CheckBoxControl

Представляет собой элемент управления "Флажок" в документе. Является наследником класса [ContentControl](#). Методы `CheckBoxControl.getValue()` и

`CheckBoxControl.SetValue(bool)` позволяют получить и задать значение этого элемента управления.

Пример

```
controls = document.getContentControls()  
checkBox = controls.FindByTitle("check1").toCheckBox()  
checkBox.SetValue(not checkBox.GetValue())
```

6.32 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)
```

6.32.1 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)
```

6.32.2 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)
```

6.32.3 Метод `Color.getTransforms`

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

6.32.4 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)
```

6.32.5 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")
```

6.32.6 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")
```

6.33 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).

Конструкторы

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

Таблица 21 – Описание полей класса `ColorRGBA`

Поле	Тип	Описание
r	int	Значение от 0 до 255 для установки интенсивности красного цвета.
g	int	Значение от 0 до 255 для установки интенсивности зеленого цвета.
b	int	Значение от 0 до 255 для установки интенсивности голубого цвета.
a	int	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)
```

6.33.1 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")
```

6.33.2 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")
```

6.34 Класс `ColorScaleConditionalFormatOperator`

Класс `ColorScaleConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Цветовая шкала". Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
ColorScaleConditionalFormatOperator(ConditionalFormatColorScaleEntries
entries)
```

Пример

Итог
1500
2500
800
1200
1200
2400
1800
750
500
1800
1050

Рисунок 30 – Пример создания правила "Цветовая шкала" с тремя пороговыми значениями

```

sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

entries = myOfficeSDK.ConditionalFormatColorScaleEntries()
value1 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Min, "0", False)
entry1 = myOfficeSDK.ConditionalFormatColorScaleEntry(value1,
myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
entries.addEntry(entry1)
value2 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Percent, "50", False)
entry2 = myOfficeSDK.ConditionalFormatColorScaleEntry(value2,
myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 220, 56, 150)))
entries.addEntry(entry2)
value3 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Max, "0", False)
entry3 = myOfficeSDK.ConditionalFormatColorScaleEntry(value3,
myOfficeSDK.Color(myOfficeSDK.ColorRGBA(0, 255, 0, 150)))
entries.addEntry(entry3)

cellRange = sheet.getCellRange("G2:G12")
cellRangePosition = cellRange.getTableRange()

colorScaleOperator =

```

```
myOfficeSDK.createColorScaleConditionalFormatOperator(entries)

colorScaleRule = myOfficeSDK.ConditionalFormatRule()
colorScaleRule.setRange(cellRangePosition)
colorScaleRule.setOperator(colorScaleOperator)
rules.addRule(colorScaleRule)
```

6.34.1 Метод ColorScaleConditionalFormatOperator.getEntries

Метод возвращает набор правил применения цветов для текущего оператора.

Вызов

```
ConditionalFormatColorScaleEntries getEntries()
```

Возвращает

– набор правил применения цветов, тип [ConditionalFormatColorScaleEntries](#).

6.34.2 Метод ColorScaleConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.34.3 Метод ColorScaleConditionalFormatOperator.setEntries

Метод задает набор правил применения цветов для текущего оператора.

Вызов

```
setEntries(entries)
```

Параметры

– entries: набор правил применения цветов, тип [ConditionalFormatColorScaleEntries](#).

6.35 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)
```

6.35.1 Метод ColorTransforms.apply

Метод устанавливает цвет трансформации [ColorRGBA](#).

Пример

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

6.36 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](#).

6.36.1 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())
```

6.36.2 Метод `Comment.getInfo`

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

Пример

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

6.36.3 Метод `Comment.getRange`

Метод возвращает диапазон документа [Range](#), которому соответствует комментарий.

Пример

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

6.36.4 Метод `Comment.getReplies`

Метод предоставляет доступ к ответам на комментарии. Ответы находятся в такой же таблице [Comments](#), как и сами комментарии документа.

Пример

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

6.36.5 Метод `Comment.getText`

Метод возвращает текст комментария.

Пример

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

6.36.6 Метод `Comment.isResolved`

Метод возвращает значение `True`, если комментарий принят.

Пример

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

6.37 Класс `Comments`

Класс `Comments` содержит коллекцию комментариев диапазона (см. Рисунок 31).

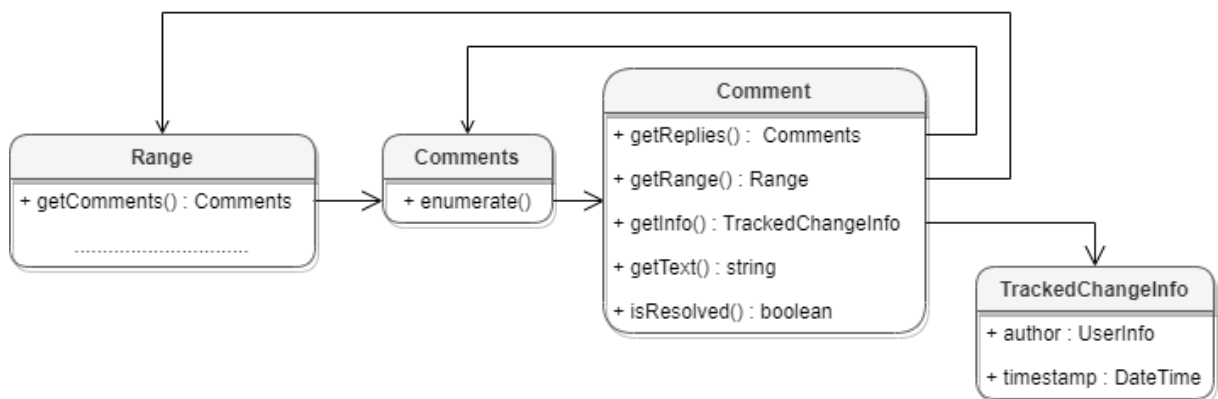


Рисунок 31 – Объектная модель классов для работы с комментариями

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

Пример

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

6.37.1 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)
```

6.38 Перечисление `ConditionalFormatAboveAverageCondition`

Перечисление `ConditionalFormatAboveAverageCondition` содержит условия применения форматирования для правил "Выше среднего" и "Ниже среднего". Используется в методе [AboveAverageConditionalFormatOperator.getCondition\(\)](#).

Таблица 22 – Условия применения форматирования

Значение	Описание
<code>ConditionalFormatAboveAverageCondition_Above</code>	Выше среднего
<code>ConditionalFormatAboveAverageCondition_EqualAbove</code>	Выше среднего или равно
<code>ConditionalFormatAboveAverageCondition_Below</code>	Ниже среднего
<code>ConditionalFormatAboveAverageCondition_EqualBelow</code>	Ниже среднего или равно

6.39 Перечисление `ConditionalFormatBinaryCondition`

Перечисление `ConditionalFormatBinaryCondition` содержит условия применения форматирования для правил "Между" и "Не между". Используется в методе [BinaryConditionalFormatOperator.getCondition\(\)](#).

Таблица 23 – Условия применения форматирования

Значение	Описание
<code>ConditionalFormatBinaryCondition_Between</code>	Между, включительно
<code>ConditionalFormatBinaryCondition_NotBetween</code>	Не между

6.40 Класс ConditionalFormatCellStyle

Класс ConditionalFormatCellStyle предназначен для настройки параметров отображения ячеек, которые попадают под условие форматирования. Эти настройки применяются к правилам, созданным с помощью следующих операторов: [AboveAverageConditionalFormatOperator](#), [BinaryConditionalFormatOperator](#), [NullaryConditionalFormatOperator](#), [TextConditionalFormatOperator](#), [TopBottomConditionalFormatOperator](#), [UnaryConditionalFormatOperator](#) и [UniquenessConditionalFormatOperator](#). Данный класс используется в методах [ConditionalFormatRule.getStyle\(\)](#), [ConditionalFormatRule.setStyle\(\)](#), [ConditionalFormatRuleProxy.getStyle\(\)](#) и [ConditionalFormatRuleProxy.setStyle\(\)](#).

Таблица 24 – Описание полей класса ConditionalFormatCellStyle

Поле	Тип	Описание
ConditionalFormatCellStyle.borders	Borders	Настройки границ ячеек
ConditionalFormatCellStyle.cellFormat	CellFormat	Формат ячеек
ConditionalFormatCellStyle.cellProperties	CellProperties	Настройки форматирования содержимого ячеек
ConditionalFormatCellStyle.textProperties	TextProperties	Параметры текста в ячейках

6.41 Класс ConditionalFormatColorScaleEntries

Класс ConditionalFormatColorScaleEntries представляет собой набор правил применения цветов для условного форматирования "Цветовая шкала". Правила в наборе должны быть расположены в порядке возрастания пороговых значений. Используется в методах [ColorScaleConditionalFormatOperator.getEntries\(\)](#) и [ColorScaleConditionalFormatOperator.setEntries\(\)](#).

Пример

```
entries = myOfficeSDK.ConditionalFormatColorScaleEntries()
value1 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Min, "0", False)
entry1 = myOfficeSDK.ConditionalFormatColorScaleEntry(value1,
```

```
myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
entries.addEntry(entry1)
# ...
colorScaleOperator = myOfficeSDK.ColorScaleConditionalFormatOperator(entries)
```

6.41.1 Метод ConditionalFormatColorScaleEntries.addEntry

Метод добавляет правило применения цвета в текущий набор.

Вызов

```
addEntry(entry)
```

Параметры

– entry: правило применения цвета, тип [ConditionalFormatColorScaleEntry](#).

6.41.2 Метод ConditionalFormatColorScaleEntries.getEntriesCount

Метод возвращает количество правил в текущем наборе.

Вызов

```
int getEntriesCount()
```

Возвращает

– количество правил, тип int.

6.41.3 Метод ConditionalFormatColorScaleEntries.getEntry

Метод возвращает правило по его индексу.

Вызов

```
ConditionalFormatColorScaleEntry getEntry(index)
```

Параметры

– index: индекс правила, индексация начинается с нуля, тип int.

Возвращает

– правило применения цвета, тип [ConditionalFormatColorScaleEntry](#).

6.41.4 Метод ConditionalFormatColorScaleEntries.setEntry

Метод заменяет правило под заданным индексом.

Вызов

```
setEntry(index, entry)
```

Параметры

— index: индекс правила для замены, индексация начинается с нуля, тип `int`.

— entry: новое правило применения цвета, тип [ConditionalFormatColorScaleEntry](#).

6.42 Класс ConditionalFormatColorScaleEntry

Класс `ConditionalFormatColorScaleEntry` представляет собой правило применения цвета для условного форматирования "Цветовая шкала". Используется в методах [ConditionalFormatColorScaleEntries.addEntry\(\)](#), [ConditionalFormatColorScaleEntries.getEntry\(\)](#) и [ConditionalFormatColorScaleEntries.setEntry\(\)](#).

Конструктор

```
ConditionalFormatColorScaleEntry(ConditionalFormatValueObject valueObject,  
Color color)
```

Пример

```
entries = myOfficeSDK.ConditionalFormatColorScaleEntries()  
value1 =  
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject  
Type_Min, "0", False)  
entry1 = myOfficeSDK.ConditionalFormatColorScaleEntry(value1,  
myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))  
entries.addEntry(entry1)
```

6.42.1 Метод ConditionalFormatColorScaleEntry.getColor

Метод возвращает цвет, который используется в текущем правиле.

Вызов

```
Color getColor()
```

Возвращает

— используемый цвет, тип [Color](#).

6.42.2 Метод `ConditionalFormatColorScaleEntry.getValueObject`

Метод возвращает пороговое значение для текущего правила.

Вызов

```
ConditionalFormatValueObject getValueObject()
```

Возвращает

– пороговое значение для текущего правила, тип [ConditionalFormatValueObject](#).

6.42.3 Метод `ConditionalFormatColorScaleEntry.setColor`

Метод позволяет задать цвет для текущего правила.

Вызов

```
setColor(color)
```

Параметры

– `color`: цвет для правила, тип [Color](#).

6.42.4 Метод `ConditionalFormatColorScaleEntry.setValueObject`

Метод позволяет задать пороговое значение для текущего правила.

Вызов

```
setValueObject(valueObject)
```

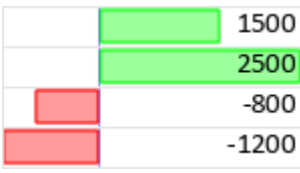
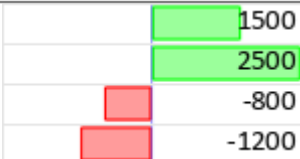
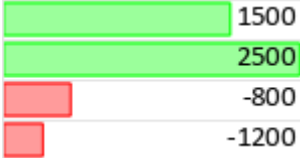
Параметры

– `valueObject`: пороговое значение для текущего правила, тип [ConditionalFormatValueObject](#).

6.43 Перечисление `ConditionalFormatDataBarAxisPosition`

Перечисление `ConditionalFormatDataBarAxisPosition` содержит варианты расположения оси между положительными и отрицательными значениями гистограммы. Используется в поле [ConditionalFormatDataBarParams.axisPosition](#).

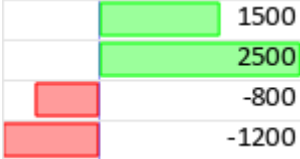
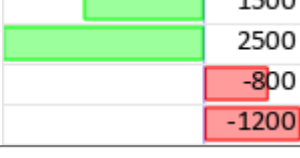
Таблица 25 – Варианты расположения оси

Значение	Описание	Изображение
ConditionalFormatDataBarAxisPosition_Auto	Ось расположена в зависимости от соотношения максимального положительного и отрицательного значений	
ConditionalFormatDataBarAxisPosition_Middle	Ось расположена посередине ячейки	
ConditionalFormatDataBarAxisPosition_None	Ось отсутствует, положительные и отрицательные значения направлены в одну сторону	

6.44 Перечисление ConditionalFormatDataBarDirection

Перечисление ConditionalFormatDataBarDirection содержит варианты направления гистограммы в ячейке. Используется в поле [ConditionalFormatDataBarParams.direction](#).

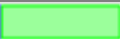
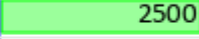
Таблица 26 – Варианты направления гистограммы

Значение	Описание	Изображение
ConditionalFormatDataBarDirection_LeftToRight	Гистограмма для положительных значений рисуется слева направо	
ConditionalFormatDataBarDirection_RightToLeft	Гистограмма для положительных значений рисуется справа налево	

6.45 Перечисление ConditionalFormatDataBarFillType

Перечисление ConditionalFormatDataBarDirection содержит варианты заливки гистограммы в ячейке. Используется в поле [ConditionalFormatDataBarParams.fillType](#).

Таблица 27 – Варианты заливки гистограммы

Значение	Описание	Изображение
ConditionalFormatDataBarFillType_Solid	Сплошная заливка	 1500
		 2500
		 -800
		 -1200
ConditionalFormatDataBarFillType_Gradient	Градиент	 1500
		 2500
		 -800
		 -1200

6.46 Класс ConditionalFormatDataBarParams

Класс ConditionalFormatDataBarParams предназначен для настройки параметров гистограммы условного форматирования. Используется при создании экземпляра класса [DataBarConditionalFormatOperator](#).

Конструктор

```
ConditionalFormatDataBarParams(ConditionalFormatValueObject lowerThreshold,
ConditionalFormatValueObject upperThreshold)
```

Таблица 28 – Описание полей класса ConditionalFormatDataBarParams

Поле	Тип	Описание
ConditionalFormatDataBarParams.lowerThreshold	ConditionalFormatValueObject	Нижнее пороговое значение
ConditionalFormatDataBarParams.upperThreshold	ConditionalFormatValueObject	Верхнее пороговое значение
ConditionalFormatDataBarParams.barFill	Color	Цвет заливки гистограммы для положительных значений
ConditionalFormatDataBarParams.negativeBarFill	Color	Цвет заливки гистограммы для отрицательных значений
ConditionalFormatDataBarParams.axisColor	Color	Цвет оси между положительными и отрицательными значениями
ConditionalFormatDataBarParams.borderColor	Color	Цвет границ гистограммы для

Поле	Тип	Описание
		положительных значений
ConditionalFormatDataBarParams.negativeBorderColor	Color	Цвет границ гистограммы для отрицательных значений
ConditionalFormatDataBarParams.minLength	float	Минимальная длина гистограммы, в процентах от ширины ячейки
ConditionalFormatDataBarParams.maxLength	float	Максимальная длина гистограммы, в процентах от ширины ячейки
ConditionalFormatDataBarParams.axisPosition	ConditionalFormatDataBarAxisPosition	Расположение оси между положительными и отрицательными значениями
ConditionalFormatDataBarParams.direction	ConditionalFormatDataBarDirection	Направление гистограммы
ConditionalFormatDataBarParams.fillType	ConditionalFormatDataBarFillType	Тип заливки гистограммы
ConditionalFormatDataBarParams.isValueVisible	bool	Показывать значение в ячейке с гистограммой

6.47 Класс ConditionalFormatDocumentRules

Класс ConditionalFormatDocumentRules представляет собой коллекцию правил условного форматирования для табличного документа. Элементы коллекции представлены объектами класса [ConditionalFormatRuleProxy](#). Используется в методе [Document.getConditionalFormatRules\(\)](#).

Пример

```
rules = document.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_AboveAverage:
        aboveOperator =
myOfficeSDK.castToAboveAverageConditionalFormat(rule.getOperator())
```

6.47.1 Метод ConditionalFormatDocumentRules.removeAllRules

Метод удаляет все правила условного форматирования из коллекции.

Вызов

```
removeAllRules()
```

Пример

```
rules = document.getConditionalFormatRules()
rules.removeAllRules()
```

6.48 Перечисление ConditionalFormatIconSet

Перечисление ConditionalFormatIconSet содержит наборы значков, которые используются в условном форматировании. Используется в методах [ConditionalFormatIconSetEntry.getIconSet\(\)](#) и [ConditionalFormatIconSetEntry.setIconSet\(\)](#).

Таблица 29 – Наборы значков условного форматирования

Значение	Набор значков
ConditionalFormatIconSet_ThreeArrows	
ConditionalFormatIconSet_ThreeArrowsGray	
ConditionalFormatIconSet_ThreeFlags	
ConditionalFormatIconSet_ThreeTrafficLights1	
ConditionalFormatIconSet_ThreeTrafficLights2	
ConditionalFormatIconSet_ThreeSigns	
ConditionalFormatIconSet_ThreeSymbols	
ConditionalFormatIconSet_ThreeSymbols2	
ConditionalFormatIconSet_ThreeStars	
ConditionalFormatIconSet_ThreeTriangles	
ConditionalFormatIconSet_ThreeSmilies	Не поддерживается
ConditionalFormatIconSet_ThreeColorSmilies	Не поддерживается
ConditionalFormatIconSet_FourArrows	

Значение	Набор значков
ConditionalFormatIconSet_FourArrowsGray	
ConditionalFormatIconSet_FourRedToBlack	
ConditionalFormatIconSet_FourRating	
ConditionalFormatIconSet_FourTrafficLights	
ConditionalFormatIconSet_FiveArrows	
ConditionalFormatIconSet_FiveArrowsGray	
ConditionalFormatIconSet_FiveRating	
ConditionalFormatIconSet_FiveQuarters	
ConditionalFormatIconSet_FiveBoxes	
ConditionalFormatIconSet_NoIcons	Пустой набор

Пример

```
entries = myOfficeSDK.ConditionalFormatIconSetEntries()
value1 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Percent, "0", False)
entry1 = myOfficeSDK.ConditionalFormatIconSetEntry(value1,
myOfficeSDK.ConditionalFormatIconSet_FiveBoxes, 0)
entries.addEntry(entry1)
```

6.49 Класс ConditionalFormatIconSetEntries

Класс `ConditionalFormatIconSetEntries` представляет собой набор правил отображения значков для условного форматирования. Правила в наборе должны быть расположены в порядке возрастания пороговых значений. Используется в методах [IconSetConditionalFormatOperator.getEntries\(\)](#) и [IconSetConditionalFormatOperator.setEntries\(\)](#).

Пример

```
entries = myOfficeSDK.ConditionalFormatIconSetEntries()
value1 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Percent, "0", False)
entry1 = myOfficeSDK.ConditionalFormatIconSetEntry(value1,
```

```
myOfficeSDK.ConditionalFormatIconSet_FiveBoxes, 0)
entries.addEntry(entry1)
# ...
iconSetOperator = myOfficeSDK.IconSetConditionalFormatOperator(entries, True)
```

6.49.1 Метод ConditionalFormatIconSetEntries.addEntry

Метод добавляет правило отображения значка в текущий набор.

Вызов

```
addEntry(entry)
```

Параметры

– entry: правило отображения значка, тип [ConditionalFormatIconSetEntry](#).

6.49.2 Метод ConditionalFormatIconSetEntries.getEntriesCount

Метод возвращает количество правил в текущем наборе.

Вызов

```
int getEntriesCount()
```

Возвращает

– количество правил, тип int.

6.49.3 Метод ConditionalFormatIconSetEntries.getEntry

Метод возвращает правило по его индексу.

Вызов

```
ConditionalFormatIconSetEntry getEntry(index)
```

Параметры

– index: индекс правила, индексация начинается с нуля, тип int.

Возвращает

– правило отображения значка, тип [ConditionalFormatIconSetEntry](#).

6.49.4 Метод ConditionalFormatIconSetEntries.setEntry

Метод заменяет правило под заданным индексом.

Вызов

```
setEntry(index, entry)
```

Параметры

- index: индекс правила для замены, индексация начинается с нуля, тип `int`.
- entry: новое правило отображения значка, тип [ConditionalFormatIconSetEntry](#).

6.50 Класс ConditionalFormatIconSetEntry

Класс `ConditionalFormatIconSetEntry` представляет собой правило отображения значка для условного форматирования. Используется в методах [ConditionalFormatIconSetEntries.addEntry\(\)](#), [ConditionalFormatIconSetEntries.getEntry\(\)](#) и [ConditionalFormatIconSetEntries.setEntry\(\)](#).

Конструкторы

```
ConditionalFormatIconSetEntry()
```

```
ConditionalFormatIconSetEntry(ConditionalFormatValueObject valueObject,  
ConditionalFormatIconSet iconSet, byte iconIndex)
```

Пример

```
entries = myOfficeSDK.ConditionalFormatIconSetEntries()  
value1 =  
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject  
Type_Percent, "0", False)  
entry1 = myOfficeSDK.ConditionalFormatIconSetEntry(value1,  
myOfficeSDK.ConditionalFormatIconSet_FiveBoxes, 0)  
entries.addEntry(entry1)
```

6.50.1 Метод ConditionalFormatIconSetEntry.getIconIndex

Метод возвращает индекс используемого значка из текущего набора.

Вызов

```
int getIconIndex()
```

Возвращает

- индекс значка в наборе, индексация начинается с нуля, тип `int`.

6.50.2 Метод `ConditionalFormatIconSetEntry.getIconSet`

Метод возвращает набор, значок из которого используется в текущем правиле.

Вызов

```
ConditionalFormatIconSet getIconSet()
```

Возвращает

– набор значков, тип [ConditionalFormatIconSet](#).

6.50.3 Метод `ConditionalFormatIconSetEntry.getValueObject`

Метод возвращает пороговое значение для текущего правила.

Вызов

```
ConditionalFormatValueObject getValueObject()
```

Возвращает

– пороговое значение для текущего правила, тип [ConditionalFormatValueObject](#).

6.50.4 Метод `ConditionalFormatIconSetEntry.setIconIndex`

Метод позволяет задать значок по его индексу в наборе.

Вызов

```
setIconIndex(iconIndex)
```

Параметры

– `iconIndex`: индекс значка в наборе, индексация начинается с нуля, тип `int`.

6.50.5 Метод `ConditionalFormatIconSetEntry.setIconSet`

Метод позволяет задать набор значков для использования в текущем правиле.

Вызов

```
setIconSet(iconSet)
```

Параметры

– `iconSet`: набор значков, тип [ConditionalFormatIconSet](#).

6.50.6 Метод ConditionalFormatIconSetEntry.setValueObject

Метод позволяет задать пороговое значение для текущего правила.

Вызов

```
setValueObject(valueObject)
```

Параметры

– valueObject: пороговое значение для текущего правила, тип [ConditionalFormatValueObject](#).

6.51 Перечисление ConditionalFormatNullaryCondition

Перечисление ConditionalFormatNullaryCondition содержит условия применения форматирования для правил без параметров. Используется в методе [NullaryConditionalFormatOperator.getCondition\(\)](#).

Таблица 30 – Условия применения форматирования

Значение	Описание
ConditionalFormatNullaryCondition_IsBlank	Пустая ячейка
ConditionalFormatNullaryCondition_IsNotBlank	Непустая ячейка
ConditionalFormatNullaryCondition_IsError	Ячейка с ошибкой формулы
ConditionalFormatNullaryCondition_IsNotError	Ячейка без ошибки формулы
ConditionalFormatNullaryCondition_Yesterday	Вчера
ConditionalFormatNullaryCondition_Today	Сегодня
ConditionalFormatNullaryCondition_Tomorrow	Завтра
ConditionalFormatNullaryCondition_InLast7Days	Последние 7 дней
ConditionalFormatNullaryCondition_LastWeek	Прошлая неделя
ConditionalFormatNullaryCondition_ThisWeek	Эта неделя
ConditionalFormatNullaryCondition_NextWeek	Следующая неделя
ConditionalFormatNullaryCondition_LastMonth	Прошлый месяц
ConditionalFormatNullaryCondition_ThisMonth	Этот месяц
ConditionalFormatNullaryCondition_NextMonth	Следующий месяц

6.52 Класс ConditionalFormatOperator

Класс ConditionalFormatOperator представляет собой базовый класс для операторов условного форматирования. Используется в методах [ConditionalFormatRule.getOperator\(\)](#), [ConditionalFormatRule.setOperator\(\)](#), [ConditionalFormatRuleProxy.getOperator\(\)](#) и [ConditionalFormatRuleProxy.setOperator\(\)](#).

Наследники

- [AboveAverageConditionalFormatOperator](#)
- [BinaryConditionalFormatOperator](#)
- [ColorScaleConditionalFormatOperator](#)
- [DataBarConditionalFormatOperator](#)
- [IconSetConditionalFormatOperator](#)
- [NullaryConditionalFormatOperator](#)
- [TextConditionalFormatOperator](#)
- [TopBottomConditionalFormatOperator](#)
- [UnaryConditionalFormatOperator](#)
- [UniquenessConditionalFormatOperator](#)

6.52.1 Метод ConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.53 Перечисление ConditionalFormatOperatorType

Перечисление ConditionalFormatOperatorType содержит типы операторов условного форматирования. Используется в методах [ConditionalFormatOperator.getType\(\)](#) и [ConditionalFormatRuleProxy.getType\(\)](#).

Таблица 31 – Типы операторов условного форматирования

Значение	Описание
ConditionalFormatOperatorType_Nullary	Оператор без аргументов
ConditionalFormatOperatorType_Unary	Больше, меньше, равно, не равно
ConditionalFormatOperatorType_Binary	Между и не между
ConditionalFormatOperatorType_Text	"Текст"
ConditionalFormatOperatorType_TopBottom	Наибольшие и наименьшие значения
ConditionalFormatOperatorType_AboveAverage	Выше и ниже среднего
ConditionalFormatOperatorType_Uniqueness	Уникальные и повторяющиеся значения
ConditionalFormatOperatorType_IconSet	"Значки"
ConditionalFormatOperatorType_ColorScale	"Цветовая шкала"
ConditionalFormatOperatorType_DataBar	"Гистограмма"

6.54 Класс ConditionalFormatRule

Класс `ConditionalFormatRule` представляет собой правило условного форматирования. Используется в методе [ConditionalFormatTableRules.addRule\(\)](#).

Конструкторы

```
ConditionalFormatRule()
```

```
ConditionalFormatRule(ConditionalFormatOperator conditionalFormatOperator,
ConditionalFormatCellStyle cellStyle, CellRangePosition cellRangePosition, bool
stopCalculations)
```

```
ConditionalFormatRule(ConditionalFormatOperator conditionalFormatOperator,
ConditionalFormatCellStyle cellStyle, CellRangePositions cellRangePositions, bool
stopCalculations)
```

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

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

aboveStyle = myOfficeSDK.ConditionalFormatCellStyle()
```

```
cellProperties = myOfficeSDK.CellProperties()  
cellProperties.fill = myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(0,  
255, 0, 150)))  
aboveStyle.cellProperties = cellProperties  
  
aboveOperator =  
myOfficeSDK.createAboveAverageConditionalFormatOperator(myOfficeSDK.ConditionalFo  
rmatAboveAverageCondition_Above)  
  
aboveRule = myOfficeSDK.ConditionalFormatRule(aboveOperator, aboveStyle,  
cellRangePosition, False)  
rules.addRule(aboveRule)
```

6.54.1 Метод ConditionalFormatRule.getOperator

Метод возвращает оператор условного форматирования.

Вызов

```
ConditionalFormatOperator getOperator()
```

Возвращает

— оператор условного форматирования, тип [ConditionalFormatOperator](#).

6.54.2 Метод ConditionalFormatRule.getRanges

Метод возвращает диапазоны ячеек таблицы, к которым применяется текущее правило.

Вызов

```
CellRangePositions getRanges()
```

Возвращает

— диапазоны ячеек таблицы, тип `CellRangePositions`.

6.54.3 Метод ConditionalFormatRule.getStopCalculations

Метод позволяет определить, будут ли применяться другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
bool getStopCalculations()
```

Возвращает

- True, если другие правила не применяются к ячейкам, которые попадают под текущее правило, в ином случае – False.

6.54.4 Метод `ConditionalFormatRule.getStyle`

Метод возвращает настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
ConditionalFormatCellStyle getStyle()
```

Возвращает

- настройки параметров отображения ячеек, тип [ConditionalFormatCellStyle](#).

6.54.5 Метод `ConditionalFormatRule.getUUID`

Метод возвращает уникальный идентификатор для текущего правила.

Вызов

```
string getUUID()
```

Возвращает

- уникальный идентификатор правила, тип `string`.

6.54.6 Метод `ConditionalFormatRule.setOperator`

Метод позволяет задать оператор условного форматирования.

Вызов

```
setOperator(conditionalFormatOperator)
```

Параметры

- `conditionalFormatOperator`: оператор условного форматирования, тип [ConditionalFormatOperator](#) или его наследники.

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
entries = myOfficeSDK.ConditionalFormatIconSetEntries()
// ...
iconSetOperator = myOfficeSDK.createIconSetConditionalFormatOperator(entries,
True)
```

```
cellRange = sheet.getCellRange("G2:G12")
cellRangePosition = cellRange.getTableRange()

iconSetRule = myOfficeSDK.ConditionalFormatRule()
iconSetRule.setOperator(iconSetOperator)
iconSetRule.setRange(cellRangePosition)
rules.addRule(iconSetRule)
```

6.54.7 Метод `ConditionalFormatRule.setRange`

Метод позволяет задать диапазон ячеек таблицы, к которому будет применено текущее правило.

Вызов

```
setRange(cellRangePosition)
```

Параметры

– `cellRangePosition`: диапазон ячеек таблицы, тип [CellRangePosition](#).

6.54.8 Метод `ConditionalFormatRule.setRanges`

Метод позволяет задать диапазоны ячеек таблицы, к которым будет применено текущее правило.

Вызов

```
setRanges(cellRangePositions)
```

Параметры

– `cellRangePositions`: диапазоны ячеек таблицы, тип `CellRangePositions`.

6.54.9 Метод `ConditionalFormatRule.setStopCalculations`

Метод позволяет задать, применять ли другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
setStopCalculations(stopCalculations)
```

Параметры

- stopCalculations: True, чтобы не применять другие правила к ячейкам, которые попадают под текущее правило, в ином случае – False.

6.54.10 Метод ConditionalFormatRule.setStyle

Метод позволяет задать настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
setStyle(style)
```

Параметры

- style: настройки параметров отображения ячеек, тип [ConditionalFormatCellStyle](#).

6.54.11 Метод ConditionalFormatRule.setUUID

Метод позволяет задать уникальный идентификатор для текущего правила.

Вызов

```
setUUID(uuid)
```

Параметры

- uuid: уникальный идентификатор правила, тип string.

6.55 Класс ConditionalFormatRuleProxy

Класс ConditionalFormatRuleProxy представляет собой правило условного форматирования, как элемент коллекции [ConditionalFormatDocumentRules](#) или [ConditionalFormatTableRules](#). Используется в методе [ConditionalFormatTableRules.getRule\(\)](#).

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
firstRule = rules.getRule(0)
firstRule.remove()
```

6.55.1 Метод ConditionalFormatRuleProxy.getData

Метод возвращает объект [ConditionalFormatRule](#) для текущего правила.

Вызов

```
ConditionalFormatRule getData()
```

Возвращает

– правило условного форматирования, тип [ConditionalFormatRule](#).

6.55.2 Метод ConditionalFormatRuleProxy.getIndex

Метод возвращает индекс правила в коллекции.

Вызов

```
int getIndex()
```

Возвращает

– индекс правила в коллекции, индексация начинается с нуля, тип `int`.

6.55.3 Метод ConditionalFormatRuleProxy.getOperator

Метод возвращает оператор условного форматирования.

Вызов

```
ConditionalFormatOperator getOperator()
```

Возвращает

– оператор условного форматирования, тип [ConditionalFormatOperator](#).

6.55.4 Метод ConditionalFormatRuleProxy.getRanges

Метод возвращает диапазоны ячеек таблицы, к которым применяется текущее правило.

Вызов

```
CellRangePositions getRanges()
```

Возвращает

– диапазоны ячеек таблицы, тип `CellRangePositions`.

6.55.5 Метод `ConditionalFormatRuleProxy.getStopCalculations`

Метод позволяет определить, будут ли применяться другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
bool getStopCalculations()
```

Возвращает

– True, если другие правила не применяются к ячейкам, которые попадают под текущее правило, в ином случае – False.

6.55.6 Метод `ConditionalFormatRuleProxy.getStyle`

Метод возвращает настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
ConditionalFormatCellStyle getStyle()
```

Возвращает

– настройки параметров отображения ячеек, тип [ConditionalFormatCellStyle](#).

6.55.7 Метод `ConditionalFormatRuleProxy.getTableName`

Метод возвращает название листа, который содержит текущее правило.

Вызов

```
string getTableName()
```

Возвращает

– название листа, тип `string`.

6.55.8 Метод `ConditionalFormatRuleProxy.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.55.9 Метод `ConditionalFormatRuleProxy.remove`

Метод удаляет текущее правило из коллекции.

Вызов

```
remove( )
```

6.55.10 Метод `ConditionalFormatRuleProxy.setOperator`

Метод позволяет задать оператор условного форматирования.

Вызов

```
setOperator(conditionalFormatOperator)
```

Параметры

– `conditionalFormatOperator`: оператор условного форматирования, тип [ConditionalFormatOperator](#) или его наследники.

6.55.11 Метод `ConditionalFormatRuleProxy.setRange`

Метод позволяет задать диапазон ячеек таблицы, к которому будет применено текущее правило.

Вызов

```
setRange(cellRangePosition)
```

Параметры

– `cellRangePosition`: диапазон ячеек таблицы, тип [CellRangePosition](#).

6.55.12 Метод `ConditionalFormatRuleProxy.setRanges`

Метод позволяет задать диапазоны ячеек таблицы, к которым будет применено текущее правило.

Вызов

```
setRanges(cellRangePositions)
```

Параметры

– `cellRangePositions`: диапазоны ячеек таблицы, тип `CellRangePositions`.

6.55.13 Метод `ConditionalFormatRuleProxy.setStopCalculations`

Метод позволяет задать, применять ли другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
setStopCalculations(stopCalculations)
```

Параметры

- `stopCalculations`: `True`, чтобы не применять другие правила к ячейкам, которые попадают под текущее правило, в ином случае – `False`.

6.55.14 Метод `ConditionalFormatRuleProxy.setStyle`

Метод позволяет задать настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
setStyle(style)
```

Параметры

- `style`: настройки параметров отображения ячеек, тип [ConditionalFormatCellStyle](#).

6.56 Класс `ConditionalFormatTableRules`

Класс `ConditionalFormatTableRules` представляет собой коллекцию правил условного форматирования для листа табличного документа. Элементы коллекции представлены объектами класса [ConditionalFormatRuleProxy](#). Используется в методе [Table.getConditionalFormatRules\(\)](#).

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_AboveAverage:
        aboveOperator =
myOfficeSDK.castToAboveAverageConditionalFormat(rule.getOperator())
```

6.56.1 Метод ConditionalFormatTableRules.addRule

Метод добавляет заданное правило в коллекцию.

Вызов

```
addRule(rule)
```

Параметры

– rule: правило условного форматирования, тип [ConditionalFormatRule](#).

Пример

```
aboveOperator =  
myOfficeSDK.createAboveAverageConditionalFormatOperator(myOfficeSDK.ConditionalFo  
rmatAboveAverageCondition_Above)  
aboveRule = myOfficeSDK.ConditionalFormatRule(aboveOperator, aboveStyle,  
cellRangePosition, False)  
rules.addRule(aboveRule)
```

6.56.2 Метод ConditionalFormatTableRules.getRule

Метод возвращает правило по его индексу.

Вызов

```
ConditionalFormatRuleProxy getRule(index)
```

Параметры

– index: индекс правила, индексация начинается с нуля, тип int.

Возвращает

– Правило условного форматирования, тип [ConditionalFormatRuleProxy](#).

Пример

```
sheet = document.getBlocks().getTable(0)  
rules = sheet.getConditionalFormatRules()  
rule = rules.getRule(rules.getRuleCount() - 1)
```

6.56.3 Метод ConditionalFormatTableRules.getRuleCount

Метод возвращает количество правил в коллекции.

Вызов

```
int getRuleCount()
```

Возвращает

– количество правил в коллекции, тип `int`.

6.56.4 Метод `ConditionalFormatTableRules.removeAllRules`

Метод удаляет все правила условного форматирования из коллекции.

Вызов

```
removeAllRules()
```

6.56.5 Метод `ConditionalFormatTableRules.removeRulesFromRange`

Метод удаляет правила условного форматирования, которые попадают в заданный диапазон ячеек.

Вызов

```
removeRulesFromRange(cellRangePosition)
```

Параметры

– `cellRangePosition`: диапазон ячеек таблицы, тип [CellRangePosition](#).

Пример

```
sheet = document.getBlocks().getTable(0)
cellRange = sheet.getCellRange("A2:F12")
cellRangePosition = cellRange.getTableRange()
rules = sheet.getConditionalFormatRules()
rules.removeRulesFromRange(cellRangePosition)
```

6.57 Перечисление `ConditionalFormatTextCondition`

Перечисление `ConditionalFormatTextCondition` содержит условия применения форматирования для правила "Текст". Используется в методе [TextConditionalFormatOperator.getCondition\(\)](#).

Таблица 32 – Условия применения форматирования

Значение	Описание
<code>ConditionalFormatTextCondition_ContainsText</code>	Содержит
<code>ConditionalFormatTextCondition_DoesNotContainText</code>	Не содержит
<code>ConditionalFormatTextCondition_BeginsWith</code>	Начинается с
<code>ConditionalFormatTextCondition_EndsWith</code>	Заканчивается на

6.58 Перечисление ConditionalFormatTopBottomCondition

Перечисление ConditionalFormatTopBottomCondition содержит условия применения форматирования для правила "Наибольшие и наименьшие значения". Используется в методе [TopBottomConditionalFormatOperator.getCondition\(\)](#).

Таблица 33 – Условия применения форматирования

Значение	Описание
ConditionalFormatTopBottomCondition_Top	Наибольшие значения
ConditionalFormatTopBottomCondition_Bottom	Наименьшие значения

6.59 Перечисление ConditionalFormatUnaryCondition

Перечисление ConditionalFormatUnaryCondition содержит условия применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Используется в методе [UnaryConditionalFormatOperator.getCondition\(\)](#).

Таблица 34 – Условия применения форматирования

Значение	Описание
ConditionalFormatUnaryCondition_Equal	Равно
ConditionalFormatUnaryCondition_NotEqual	Не равно
ConditionalFormatUnaryCondition_Greater	Больше
ConditionalFormatUnaryCondition_GreaterOrEqual	Больше или равно
ConditionalFormatUnaryCondition_Less	Меньше
ConditionalFormatUnaryCondition_LessOrEqual	Меньше или равно
ConditionalFormatUnaryCondition_Expression	Позволяет использовать формулу в качестве аргумента

Пример

Итог	Статус
1500	Доставлено
2500	
800	Доставлено
1200	Доставлено
1200	Доставлено
2400	
1800	Доставлено
750	
500	
1800	Доставлено
1050	

Рисунок 32 – Пример создания правила с формулой

```

sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

unaryStyle = myOfficeSDK.ConditionalFormatCellStyle()
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
unaryStyle.cellProperties = cellProperties

cellRange = sheet.getCellRange("G2:G12")
cellRangePosition = cellRange.getTableRange()

unaryOperator =
myOfficeSDK.UnaryConditionalFormatOperator(myOfficeSDK.ConditionalFormatUnaryCondi
tion_Expression, "=ISBLANK(H2)")

unaryRule = myOfficeSDK.ConditionalFormatRule(unaryOperator, unaryStyle,
cellRangePosition, False)
rules.addRule(unaryRule)

```

6.60 Перечисление ConditionalFormatUniquenessCondition

Перечисление ConditionalFormatUniquenessCondition содержит условия применения форматирования для правила "Уникальные и повторяющиеся значения". Используется в методе [UniquenessConditionalFormatOperator.getCondition\(\)](#).

Таблица 35 – Условия применения форматирования

Значение	Описание
ConditionalFormatUniquenessCondition_Duplicate	Повторяющиеся значения
ConditionalFormatUniquenessCondition_Unique	Уникальные значения

6.61 Класс ConditionalFormatValueObject

Класс ConditionalFormatValueObject представляет собой пороговое значение для применения подправил условного форматирования. Используется в правилах, созданных с помощью следующих операторов: [ColorScaleConditionalFormatOperator](#), [DataBarConditionalFormatOperator](#) и [IconSetConditionalFormatOperator](#).

Конструкторы

```
ConditionalFormatValueObject()
```

```
ConditionalFormatValueObject(ConditionalFormatValueObjectType type, string value, bool isStrictCompare)
```

Пример

```
value1 =  
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject  
Type_Percent, "0", False)  
entry1 = myOfficeSDK.ConditionalFormatIconSetEntry(value1,  
myOfficeSDK.ConditionalFormatIconSet_FiveBoxes, 0)
```

6.61.1 Метод ConditionalFormatValueObject.getType

Метод возвращает тип текущего порогового значения.

Вызов

```
ConditionalFormatValueObjectType getType()
```

Возвращает

– тип порогового значения, тип [ConditionalFormatValueObjectType](#).

6.61.2 Метод ConditionalFormatValueObject.getValue

Метод возвращает значение порога.

Вызов

```
string getValue()
```

Возвращает

– значение порога, тип `string`.

6.61.3 Метод `ConditionalFormatValueObject.isStrictCompare`

Метод позволяет определить, используется ли строгое сравнение в текущем пороговом значении (" $>$ " вместо " $>=$ ").

Вызов

```
bool isStrictCompare()
```

Возвращает

– `True`, если используется строгое сравнение, в ином случае – `False`.

6.61.4 Метод `ConditionalFormatValueObject.setStrictCompare`

Метод позволяет использовать строгое сравнение в текущем пороговом значении (" $>$ " вместо " $>=$ ").

Вызов

```
setStrictCompare(isStrictCompare)
```

Параметры

– `isStrictCompare`: `True`, чтобы использовать строгое сравнение, в ином случае – `False`.

6.61.5 Метод `ConditionalFormatValueObject.setType`

Метод позволяет задать тип текущего порогового значения.

Вызов

```
setType(type)
```

Параметры

– `type`: тип порогового значения, тип [ConditionalFormatValueObjectType](#).

6.61.6 Метод `ConditionalFormatValueObject.setValue`

Метод позволяет задать значение порога.

Вызов

```
setValue(value)
```

Параметры

– value: значение порога, тип string.

6.62 Перечисление ConditionalFormatValueObjectType

Перечисление ConditionalFormatValueObjectType содержит типы пороговых значений, которые используются в правилах условного форматирования. Используется в методах [ConditionalFormatValueObject.getType\(\)](#) и [ConditionalFormatValueObject.setType\(\)](#).

Таблица 36 – Типы пороговых значений

Значение	Описание
ConditionalFormatValueObjectType_Percentile	Процентиль
ConditionalFormatValueObjectType_Value	Фиксированное число
ConditionalFormatValueObjectType_Percent	Процент от наибольшего числа
ConditionalFormatValueObjectType_Formula	Результат формулы
ConditionalFormatValueObjectType_Min	Наименьшее число
ConditionalFormatValueObjectType_Max	Наибольшее число
ConditionalFormatValueObjectType_AutoMin	Автоматически устанавливает нижнее значение (только для правила "Гистограмма")
ConditionalFormatValueObjectType_AutoMax	Автоматически устанавливает верхнее значение (только для правила "Гистограмма")

Для корректного задания пороговых значений, при использовании типов Min, Max, AutoMin и AutoMax, необходимо дополнительно указать значение порога – 0.

Примеры

```
value1 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Percent, "0", False)
entry1 = myOfficeSDK.ConditionalFormatIconSetEntry(value1,
myOfficeSDK.ConditionalFormatIconSet_FiveBoxes, 0)
```

```
value1 = myOfficeSDK.ConditionalFormatValueObject()  
value1.setType(myOfficeSDK.ConditionalFormatValueObjectType_AutoMin)  
value1.setValue("0")  
value2 = myOfficeSDK.ConditionalFormatValueObject()  
value2.setType(myOfficeSDK.ConditionalFormatValueObjectType_AutoMax)  
value2.setValue("0")  
  
dataBarParams = myOfficeSDK.ConditionalFormatDataBarParams(value1, value2)
```

6.63 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](#).

6.63.1 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)
```

6.63.2 Методы добавления условий

Эти методы добавляют в фильтр условия сравнения со значениями, которые передаются в качестве аргумента. Если значение ячейки не соответствует указанным критериям, строки будут скрыты при применении фильтра.



- Критерии **match** и **notMatch** не могут быть сохранены для документов формата OXML.
- Критерии **aboveAverage**, **belowAverage**, **topValues**, **bottomValues**, **topPercent** и **bottomPercent** могут быть сохранены для документов формата OXML только если они являются единственными в фильтре.

```
equal(string value)  
notEqual(string value)  
less(string value)  
lessOrEqual(string value)  
greater(string value)  
greaterOrEqual(string value)  
match(string value)  
notMatch(string value)  
begins(string value)  
notBegins(string value)  
ends(string value)  
notEnds(string value)  
contains(string value)  
notContains(string value)  
aboveAverage()  
belowAverage()  
topValues(int numValues)  
bottomValues(int numValues)  
topPercent(int percentage)  
bottomPercent(int percentage)
```

Пример

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

Пример использования приведен в разделе [Работа с фильтрами](#).

6.64 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()
```

6.65 Перечисление ContainingTableFilter

Перечисление ContainingTableFilter содержит возможные типы фильтров столбцов таблицы. Используется в методе [TableFilters.getFilterType\(\)](#).

Таблица 37 – Типы фильтров

Значение	Описание
ContainingTableFilter_None	Фильтр отсутствует
ContainingTableFilter_Values	Фильтр по значению (ValuesTableFilter)
ContainingTableFilter_Conditional	Фильтр по условию (ConditionalTableFilter)
ContainingTableFilter_Color	Не поддерживается

Пример

```
if filters.getFilterType(0) == myOfficeSDK.ContainingTableFilter_Values:  
    valuesFilter = filters.getAsValueFilter(0)
```

6.66 Класс ContentControl

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

Наследники

- [CheckBoxControl](#)
- [DatePickerControl](#)
- [DropListControl](#)
- [InputFieldControl](#)

6.66.1 Метод ContentControl.canEdit

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

6.66.2 Метод ContentControl.getTitle

Метод возвращает название элемента управления.

6.66.3 Методы toCheckBox, toInputField, toDatePicker, toDropList

Методы преобразуют объект [ContentControl](#) в объект соответствующего типа. Возвращают None, если преобразование невозможно.

Пример

```
controls = document.getContentControls()

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

6.67 Класс ContentControls

Предоставляет доступ к операциям с элементами управления в документе (см. [Работа с элементами управления](#)). Метод `ContentControls.findByTitle(string)` позволяет получить элемент управления [ContentControl](#) по его названию.

Пример

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

6.68 Класс CurrencyCellFormatting

Класс содержит параметры для денежного формата ячеек таблицы.

Таблица 38 – Описание полей класса CurrencyCellFormatting

Поле	Тип	Описание
CurrencyCellFormatting.decimalPlaces	int	Количество десятичных позиций
CurrencyCellFormatting.symbol	string	Символ денежной единицы
CurrencyCellFormatting.localeCode	int	Идентификатор кода языка (MS-LCID)
CurrencyCellFormatting.useRedForNegative	bool	Использовать красный цвет для отрицательных значений
CurrencyCellFormatting.useBracketsForNegative	bool	Использовать скобки для отрицательных значений
CurrencyCellFormatting.hideSign	bool	Скрывать знак «минус» для отрицательных значений
CurrencyCellFormatting.useThousandsSeparator	bool	Использовать разделитель для тысячных
CurrencyCellFormatting.currencySignPlacement	CurrencySignPlacement	Варианты размещения знака валюты

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())
```

6.69 Перечисление CurrencySignPlacement

Перечисление `CurrencySignPlacement` содержит варианты размещения знака валюты. Используется в полях [LocaleInfo.currencyFormat](#) и [CurrencyCellFormatting.currencySignPlacement](#).

Таблица 39 – Описание вариантов расположения знаков валюты

Значение	Описание
<code>CurrencySignPlacement_Prefix</code>	Символ валюты перед значением (\$12.00)
<code>CurrencySignPlacement_Suffix</code>	Символ валюты после значения (12.00 P)

6.70 Класс DataBarConditionalFormatOperator

Класс `DataBarConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Гистограмма". Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
DataBarConditionalFormatOperator(ConditionalFormatDataBarParams params)
```

Пример



Рисунок 33 – Пример создания правила "Гистограмма"

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

cellRange = sheet.getCellRange("G2:G12")
cellRangePosition = cellRange.getTableRange()

value1 = myOfficeSDK.ConditionalFormatValueObject()
value1.setType(myOfficeSDK.ConditionalFormatValueObjectType_AutoMin)
value1.setValue("0")
value2 = myOfficeSDK.ConditionalFormatValueObject()
value2.setType(myOfficeSDK.ConditionalFormatValueObjectType_AutoMax)
value2.setValue("0")

dataBarParams = myOfficeSDK.ConditionalFormatDataBarParams(value1, value2)
dataBarParams.barFill = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(0, 255, 0, 100))
dataBarParams.borderColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(0, 255, 0, 255))
dataBarParams.negativeBarFill = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 100))
dataBarParams.negativeBorderColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 255))
dataBarParams.axisPosition =
myOfficeSDK.ConditionalFormatDataBarAxisPosition_Middle
dataBarParams.axisColor = myOfficeSDK.Color(myOfficeSDK.ColorRGBA(0, 0, 255, 100))
dataBarParams.fillType = myOfficeSDK.ConditionalFormatDataBarFillType_Gradient
dataBarParams.direction =
myOfficeSDK.ConditionalFormatDataBarDirection_LeftToRight
```

```
dataBarParams.isValueVisible = True

dataBarOperator =
myOfficeSDK.createDataBarConditionalFormatOperator(dataBarParams)

dataBarRule = myOfficeSDK.ConditionalFormatRule()
dataBarRule.setRange(cellRangePosition)
dataBarRule.setOperator(dataBarOperator)
rules.addRule(dataBarRule)
```

6.70.1 Метод `DataBarConditionalFormatOperator.GetAxisColor`

Метод возвращает цвет оси между положительными и отрицательными значениями.

Вызов

```
Color getAxisColor()
```

Возвращает

— цвет оси, тип [Color](#).

6.70.2 Метод `DataBarConditionalFormatOperator.GetAxisPosition`

Метод возвращает позицию оси между положительными и отрицательными значениями.

Вызов

```
ConditionalFormatDataBarAxisPosition getAxisPosition()
```

Возвращает

— расположение оси, тип [ConditionalFormatDataBarAxisPosition](#).

6.70.3 Метод `DataBarConditionalFormatOperator.getBarFill`

Метод возвращает цвет заливки гистограммы для положительных значений.

Вызов

```
Color getBarFill()
```

Возвращает

— цвет заливки для положительных значений, тип [Color](#).

6.70.4 Метод `DataBarConditionalFormatOperator.getBorderColor`

Метод возвращает цвет границ гистограммы для положительных значений.

Вызов

```
Color getBorderColor()
```

Возвращает

– цвет границ для положительных значений, тип [Color](#).

6.70.5 Метод `DataBarConditionalFormatOperator.getDirection`

Метод возвращает направление гистограммы в ячейках.

Вызов

```
ConditionalFormatDataBarDirection getDirection()
```

Возвращает

– направление гистограммы, тип [ConditionalFormatDataBarDirection](#).

6.70.6 Метод `DataBarConditionalFormatOperator.getFillType`

Метод возвращает тип заливки гистограммы.

Вызов

```
ConditionalFormatDataBarFillType getFillType()
```

Возвращает

– тип заливки гистограммы, тип [ConditionalFormatDataBarFillType](#).

6.70.7 Метод `DataBarConditionalFormatOperator.getLowerThreshold`

Метод возвращает нижнее пороговое значение гистограммы.

Вызов

```
ConditionalFormatValueObject getLowerThreshold()
```

Возвращает

– нижнее пороговое значение гистограммы, тип [ConditionalFormatValueObject](#).

6.70.8 Метод `DataBarConditionalFormatOperator.getMaxLength`

Метод возвращает максимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
float getMaxLength()
```

Возвращает

– максимальная длина гистограммы, в процентах, тип `float`.

6.70.9 Метод `DataBarConditionalFormatOperator.getMinLength`

Метод возвращает минимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
float getMinLength()
```

Возвращает

– минимальная длина гистограммы, в процентах, тип `float`.

6.70.10 Метод `DataBarConditionalFormatOperator.getNegativeBarFill`

Метод возвращает цвет заливки гистограммы для отрицательных значений.

Вызов

```
Color getNegativeBarFill()
```

Возвращает

– цвет заливки для отрицательных значений, тип [Color](#).

6.70.11 Метод `DataBarConditionalFormatOperator.getNegativeBorderColor`

Метод возвращает цвет границ гистограммы для отрицательных значений.

Вызов

```
Color getNegativeBorderColor()
```

Возвращает

– цвет границ для отрицательных значений, тип [Color](#).

6.70.12 Метод `DataBarConditionalFormatOperator.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.70.13 Метод `DataBarConditionalFormatOperator.getUpperThreshold`

Метод возвращает верхнее пороговое значение гистограммы.

Вызов

```
ConditionalFormatValueObject getUpperThreshold()
```

Возвращает

– верхнее пороговое значение гистограммы, тип [ConditionalFormatValueObject](#).

6.70.14 Метод `DataBarConditionalFormatOperator.getValueVisibility`

Метод позволяет определить показывается ли значение в ячейке с гистограммой.

Вызов

```
bool getValueVisibility()
```

Возвращает

– True, если значение показывается в ячейке с гистограммой, в ином случае – False.

6.70.15 Метод `DataBarConditionalFormatOperator.setAxisColor`

Метод позволяет задать цвет оси между положительными и отрицательными значениями.

Вызов

```
setAxisColor(color)
```

Параметры

– color: цвет оси, тип [Color](#).

6.70.16 Метод `DataBarConditionalFormatOperator.setAxisPosition`

Метод позволяет задать позицию оси между положительными и отрицательными значениями.

Вызов

```
setAxisPosition(position)
```

Параметры

– position: расположение оси, тип [ConditionalFormatDataBarAxisPosition](#).

6.70.17 Метод `DataBarConditionalFormatOperator.setBarFill`

Метод позволяет задать цвет заливки гистограммы для положительных значений.

Вызов

```
setBarFill(fill)
```

Параметры

– fill: цвет заливки для положительных значений, тип [Color](#).

6.70.18 Метод `DataBarConditionalFormatOperator.setBorderColor`

Метод позволяет задать цвет границ гистограммы для положительных значений.

Вызов

```
setBorderColor(color)
```

Параметры

– color: цвет границ для положительных значений, тип [Color](#).

6.70.19 Метод `DataBarConditionalFormatOperator.setDirection`

Метод позволяет задать направление гистограммы в ячейках.

Вызов

```
setDirection(direction)
```

Параметры

– direction: направление гистограммы, тип [ConditionalFormatDataBarDirection](#).

6.70.20 Метод `DataBarConditionalFormatOperator.setFillType`

Метод позволяет задать тип заливки гистограммы.

Вызов

```
setFillType(fillType)
```

Параметры

– `fillType`: тип заливки гистограммы, тип [ConditionalFormatDataBarFillType](#).

6.70.21 Метод `DataBarConditionalFormatOperator.setLowerThreshold`

Метод позволяет задать нижнее пороговое значение гистограммы.

Вызов

```
setLowerThreshold(lowerThreshold)
```

Параметры

– `lowerThreshold`: нижнее пороговое значение гистограммы, тип [ConditionalFormatValueObject](#).

6.70.22 Метод `DataBarConditionalFormatOperator.setMaxLength`

Метод задает максимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
setMaxLength(float maxLength)
```

Параметры

– `maxLength`: максимальная длина гистограммы, в процентах, тип `float`.

6.70.23 Метод `DataBarConditionalFormatOperator.setMinLength`

Метод задает минимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
setMinLength(float minLength)
```

Параметры

– `minLength`: минимальная длина гистограммы, в процентах, тип `float`.

6.70.24 Метод `DataBarConditionalFormatOperator.setNegativeBarFill`

Метод позволяет задать цвет заливки гистограммы для отрицательных значений.

Вызов

```
setNegativeBarFill(fill)
```

Параметры

– `fill`: цвет заливки для отрицательных значений, тип [Color](#).

6.70.25 Метод `DataBarConditionalFormatOperator.setNegativeBorderColor`

Метод позволяет задать цвет границ гистограммы для отрицательных значений.

Вызов

```
setNegativeBorderColor(color)
```

Параметры

– `color`: цвет границ для отрицательных значений, тип [Color](#).

6.70.26 Метод `DataBarConditionalFormatOperator.setUpperThreshold`

Метод позволяет задать верхнее пороговое значение гистограммы.

Вызов

```
setUpperThreshold(upperThreshold)
```

Параметры

– `upperThreshold`: верхнее пороговое значение гистограммы, тип [ConditionalFormatValueObject](#).

6.70.27 Метод `DataBarConditionalFormatOperator.setValueVisibility`

Метод задает видимость значения в ячейке с гистограммой.

Вызов

```
setValueVisibility(visible)
```

Параметры

– `visible`: `True`, чтобы показывать значение в ячейке с гистограммой, в ином случае – `False`.

6.71 Класс `DataValidation`

Класс `DataValidation` представляет собой настройки проверки данных (см. [Проверка данных](#)). Используется в методах [CellRange.setDataValidation\(\)](#), [Cell.getDataValidation\(\)](#) и [DataValidationResult.getDataValidation\(\)](#).

6.71.1 Метод `DataValidation.clear`

Метод очищает все настройки текущей проверки данных.

Вызов

```
clear()
```

Пример

```
cell = sheet.getCell("C4")
validation = cell.getDataValidation()
if not validation.isEmpty():
    validation.clear()
```

6.71.2 Метод `DataValidation.getAllowBlank`

Метод возвращает разрешен ли ввод пустых значений.

Вызов

```
bool getAllowBlank()
```

Возвращает

— `True`, если проверка данных разрешает ввод пустых значений, в ином случае — `False`.

6.71.3 Метод `DataValidation.getErrorMessage`

Метод возвращает текст сообщения об ошибке.

Вызов

```
string getErrorMessage()
```

Возвращает

— текст сообщения об ошибке, тип `string`.

6.71.4 Метод `DataValidation.getErrorStyle`

Метод возвращает значение, которое определяет поведение редактора при вводе недопустимых значений.

Вызов

```
DataValidationErrorStyle getErrorStyle()
```

Возвращает

— поведение редактора при вводе недопустимых значений, тип [DataValidationErrorStyle](#).

6.71.5 Метод `DataValidation.getErrorTitle`

Метод возвращает заголовок сообщения об ошибке.

Вызов

```
string getErrorTitle()
```

Возвращает

— заголовок сообщения об ошибке, тип `string`.

6.71.6 Метод `DataValidation.getFormula1`

Метод возвращает первый аргумент проверки данных. При проверке по списку значений, первым и единственным аргументом может быть:

- формула, результатом которой является диапазон ячеек (`"=INDIRECT(' '[Source.xods]Data' !$E$2:$E$6)"`);
- именованный диапазон (`"=Countries"`);
- адрес диапазона ячеек (`"='Data' !$E$2:$E$6"`);
- значения, разделенные точкой с запятой (`"UK; Italy; Germany; Austria; Brazil"`).

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY() - 7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=StartDate"`);
- адрес ячейки (`"='Data' !C2"`);
- дата в формате мм/дд/гггг (`"12/31/2024"`).

Вызов

```
string getFormula1()
```

Возвращает

– первый аргумент проверки данных, тип `string`.

6.71.7 Метод `DataValidation.getFormula2`

Метод возвращает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов [Between](#) и [NotBetween](#). Второй аргумент может принимать следующие значения:

- формула, результатом которой является дата ("`=TODAY( )+7`");
- именованный диапазон, состоящий из одной ячейки с датой ("`=EndDate`");
- адрес ячейки ("`= 'Data' !C2`");
- дата в формате мм/дд/гггг ("`12/31/2024`").

Вызов

```
string getFormula2()
```

Возвращает

– второй аргумент проверки данных, тип `string`.

6.71.8 Метод `DataValidation.getOperator`

Метод возвращает оператор сравнения введенного значения с аргументом/аргументами. Используется только при проверке дат.

Вызов

```
DataValidationOperator getOperator()
```

Возвращает

– оператор сравнения, тип [DataValidationOperator](#).

6.71.9 Метод `DataValidation.getPrompt`

Метод возвращает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
string getPrompt()
```

Возвращает

– текст подсказки, тип `string`.

6.71.10 Метод `DataValidation.getPromptTitle`

Метод возвращает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
string getPromptTitle()
```

Возвращает

– заголовок подсказки, тип `string`.

6.71.11 Метод `DataValidation.getShowDropDown`

Метод возвращает значение, которое определяет показывать ли выпадающий список значений. Используется только при проверке данных по списку значений.

Вызов

```
bool getShowDropDown()
```

Возвращает

– `True`, если выпадающий список значений показывается, в ином случае – `False`.

6.71.12 Метод `DataValidation.getShowErrorMessage`

Метод возвращает значение, которое определяет показывать ли сообщение об ошибке при вводе недопустимых значений.

Вызов

```
bool getShowErrorMessage()
```

Возвращает

– `True`, если сообщение об ошибке показывается, в ином случае – `False`.

6.71.13 Метод `DataValidation.getShowInputMessage`

Метод возвращает значение, которое определяет показывать ли подсказку при вводе данных в ячейку с проверкой.

Вызов

```
bool getShowInputMessage()
```

Возвращает

– True, если подсказка показывается, в ином случае – False.

6.71.14 Метод `DataValidation.getType`

Метод возвращает тип проверки данных.

Вызов

```
DataValidationType getType()
```

Возвращает

– тип проверки данных, тип [DataValidationType](#).

6.71.15 Метод `DataValidation.isEmpty`

Метод позволяет определить наличие заданных настроек проверки данных в текущем объекте [DataValidation](#).

Вызов

```
bool isEmpty()
```

Возвращает

– True, если объект не содержит заданных настроек проверки данных, в ином случае – False.

Пример

```
cell = sheet.getCell("C4")
validation = cell.getDataValidation()
if not validation.isEmpty():
    validation.clear()
```

6.71.16 Метод `DataValidation.setAllowBlank`

Метод позволяет разрешить ввод пустых значений.

Вызов

```
setAllowBlank(allowBlank)
```

Параметры

– allowBlank: True, чтобы разрешить ввод пустых значений, в ином случае – False.

6.71.17 Метод `DataValidation.setErrorMessage`

Метод задает текст сообщения об ошибке.

Вызов

```
setErrorMessage(errorMessage)
```

Параметры

– `errorMessage`: текст сообщения об ошибке, тип `string`.

Пример

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки сообщения об ошибке:  
validation.setShowErrorMessage(True)  
validation.setErrorStyle(myOfficeSDK.DataValidationErrorStyle_Stop)  
validation.setErrorTitle("Error")  
validation.setErrorMessage("Invalid value. Choose one of the values below:")  
  
cellRange.setDataValidation(validation)
```

6.71.18 Метод `DataValidation.setErrorStyle`

Метод задает поведение редактора при вводе недопустимых значений.

Вызов

```
setErrorStyle(errorStyle)
```

Параметры

– `errorStyle`: поведение редактора при вводе недопустимых значений, тип [`DataValidationErrorStyle`](#).

Пример

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки сообщения об ошибке:  
validation.setShowErrorMessage(True)  
validation.setErrorStyle(myOfficeSDK.DataValidationErrorStyle_Stop)  
validation.setErrorTitle("Error")  
validation.setErrorMessage("Invalid value. Choose one of the values below:")  
  
cellRange.setDataValidation(validation)
```

6.71.19 Метод `DataValidation.setErrorTitle`

Метод задает заголовок сообщения об ошибке.

Вызов

```
setErrorTitle(errorTitle)
```

Параметры

– `errorTitle`: заголовок сообщения об ошибке, тип `string`.

Пример

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки сообщения об ошибке:  
validation.setShowErrorMessage(True)  
validation.setErrorStyle(myOfficeSDK.DataValidationErrorStyle_Stop)  
validation.setErrorTitle("Error")  
validation.setErrorMessage("Invalid value. Choose one of the values below:")  
  
cellRange.setDataValidation(validation)
```

6.71.20 Метод `DataValidation.setFormula1`

Метод задает первый аргумент проверки данных. При проверке по списку значений, первым и единственным аргументом может быть:

- формула, результатом которой является диапазон ячеек
("=INDIRECT(' '[Source.xods]Data'!\$E\$2:\$E\$6)");
- именованный диапазон ("=Countries");
- адрес диапазона ячеек ("='Data'!\$E\$2:\$E\$6");
- значения, разделенные точкой с запятой ("UK; Italy; Germany; Austria; Brazil").

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата ("=TODAY()-7");
- именованный диапазон, состоящий из одной ячейки с датой ("=StartDate");
- адрес ячейки ("='Data'!C2");
- дата в формате мм/дд/гггг ("12/31/2024").

Вызов

```
setFormula1(formula1)
```

Параметры

— formula1: первый аргумент проверки данных, тип string.

Пример

```
dvList = myOfficeSDK.DataValidation()  
dvList.setType(myOfficeSDK.DataValidationType_List)  
dvList.setFormula1("UK; Italy; Germany; Austria; Brazil")  
dvList.setShowDropDown(True)  
  
cellRange.setDataValidation(dvList)
```

6.71.21 Метод DataValidation.setFormula2

Метод задает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов [Between](#) и [NotBetween](#). Второй аргумент может принимать следующие значения:

- формула, результатом которой является дата ("=TODAY()+7");
- именованный диапазон, состоящий из одной ячейки с датой ("=EndDate");
- адрес ячейки ("='Data'!C2");
- дата в формате мм/дд/гггг ("12/31/2024").

Вызов

```
setFormula2(formula2)
```

Параметры

— formula2: второй аргумент проверки данных, тип string.

Пример

```
dvDate = myOfficeSDK.DataValidation()  
dvDate.setType(myOfficeSDK.DataValidationType_Date)  
dvDate.setOperator(myOfficeSDK.DataValidationOperator_Between)  
dvDate.setFormula1("06/01/2024")  
dvDate.setFormula2("08/31/2024")  
  
cellRange.setDataValidation(dvDate)
```

6.71.22 Метод DataValidation.setOperator

Метод задает оператор сравнения введенного значения с аргументом/аргументами. Используется только при проверке дат.

Вызов

```
setOperator(dataValidationOperator)
```

Параметры

— dataValidationOperator: оператор сравнения, тип [DataValidationOperator](#).

Пример

```
dvDate = myOfficeSDK.DataValidation()  
dvDate.setType(myOfficeSDK.DataValidationType_Date)  
dvDate.setOperator(myOfficeSDK.DataValidationOperator_Between)  
dvDate.setFormula1("06/01/2024")  
dvDate.setFormula2("08/31/2024")  
  
cellRange.setDataValidation(dvDate)
```

6.71.23 Метод DataValidation.setPrompt

Метод задает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
setPrompt(prompt)
```

Параметры

— prompt: текст подсказки, тип string.

Пример

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки подсказки:  
validation.setShowInputMessage(True)  
validation.setPromptTitle("Restricted input")  
validation.setPrompt("Choose values from the drop-down list")  
  
cellRange.setDataValidation(validation)
```

6.71.24 Метод DataValidation.setPromptTitle

Метод задает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
setPromptTitle(promptTitle)
```

Параметры

– promptTitle: заголовок подсказки, тип string.

Пример

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки подсказки:  
validation.setShowInputMessage(True)  
validation.setPromptTitle("Restricted input")  
validation.setPrompt("Choose values from the drop-down list")  
  
cellRange.setDataValidation(validation)
```

6.71.25 Метод DataValidation.setShowDropDown

Метод определяет показывать ли выпадающий список значений. Используется только при проверке данных по списку значений.

Вызов

```
setShowDropDown(showDropDown)
```

Параметры

– showDropDown: True, чтобы показывать выпадающий список значений, в ином случае – False.

Пример

```
dvList = myOfficeSDK.DataValidation()  
dvList.setType(myOfficeSDK.DataValidationType_List)  
dvList.setFormula1("UK; Italy; Germany; Austria; Brazil")  
dvList.setShowDropDown(True)  
  
cellRange.setDataValidation(dvList)
```

6.71.26 Метод DataValidation.setShowErrorMessage

Метод определяет показывать ли сообщение об ошибке при вводе недопустимых значений.

Вызов

```
setShowErrorMessage(showErrorMessage)
```

Параметры

– showErrorMessage: True, чтобы показывать сообщение об ошибке, в ином случае – False.

Пример

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки сообщения об ошибке:  
validation.setShowErrorMessage(True)  
validation.setErrorStyle(myOfficeSDK.DataValidationErrorStyle_Stop)  
validation.setErrorTitle("Error")  
validation.setErrorMessage("Invalid value. Choose one of the values below:")  
  
cellRange.setDataValidation(validation)
```

6.71.27 Метод DataValidation.setShowInputMessage

Метод определяет показывать ли подсказку при вводе данных в ячейку с проверкой.

Вызов

```
setShowInputMessage(showInputMessage)
```

Параметры

– showInputMessage: True, чтобы показывать подсказку, в ином случае – False.

Пример

```
validation = myOfficeSDK.DataValidation()  
# ...  
# Настройки подсказки:  
validation.setShowInputMessage(True)  
validation.setPromptTitle("Restricted input")  
validation.setPrompt("Choose values from the drop-down list")  
  
cellRange.setDataValidation(validation)
```

6.71.28 Метод DataValidation.setType

Метод задает тип проверки данных.

Вызов

```
setType(type)
```

Параметры

– type: тип проверки данных, тип [DataValidationType](#).

Пример

```
dvDate = myOfficeSDK.DataValidation()  
dvDate.setType(myOfficeSDK.DataValidationType_Date)  
dvDate.setOperator(myOfficeSDK.DataValidationOperator_Between)  
dvDate.setFormula1("06/01/2024")  
dvDate.setFormula2("08/31/2024")  
  
cellRange.setDataValidation(dvDate)
```

6.72 Перечисление DataValidationErrorStyle

Перечисление `DataValidationErrorStyle` содержит варианты поведения редактора при вводе недопустимых значений. Используется в методах [DataValidation.getErrorStyle\(\)](#) и [DataValidation.setErrorStyle\(\)](#).

Таблица 40 – Варианты поведения редактора при вводе недопустимых значений

Значение	Описание
<code>DataValidationErrorStyle_Stop</code>	Запрещает ввод в ячейку недопустимого значения
<code>DataValidationErrorStyle_Warning</code>	Позволяет ввести в ячейку недопустимое значение после соответствующего предупреждения
<code>DataValidationErrorStyle_Information</code>	Поведение идентично значению <code>Warning</code>

6.73 Перечисление DataValidationOperator

Перечисление `DataValidationOperator` содержит операторы сравнения введенной даты с аргументом/аргументами. Используется в методах [DataValidation.getOperator\(\)](#) и [DataValidation.setOperator\(\)](#).

Таблица 41 – Описание операторов сравнения дат

Значение	Описание
<code>DataValidationOperator_Between</code>	Дата должна попадать в интервал между двумя аргументами включительно
<code>DataValidationOperator_NotBetween</code>	Дата должна быть вне интервала между двумя аргументами

Значение	Описание
<code>DataValidationOperator_Equal</code>	Дата должна совпадать со значением первого аргумента
<code>DataValidationOperator_NotEqual</code>	Дата не должна совпадать со значением первого аргумента
<code>DataValidationOperator_LessThan</code>	Дата должна быть раньше первого аргумента
<code>DataValidationOperator_LessThanOrEqualTo</code>	Дата должна быть раньше первого аргумента или совпадать с ним
<code>DataValidationOperator_GreaterThan</code>	Дата должна быть позже первого аргумента
<code>DataValidationOperator_GreaterThanOrEqualTo</code>	Дата должна быть позже первого аргумента или совпадать с ним

6.74 Класс `DataValidationResult`

Класс `DataValidationResult` представляет собой результат проверки значения ячейки (см. [Проверка данных](#)). Используется в методе [Cell.checkDataValidation\(\)](#).

6.74.1 Метод `DataValidationResult.getDataValidation`

Метод возвращает настройки проверки данных, если ячейка содержит недопустимое значение.

Вызов

```
DataValidation getDataValidation()
```

Возвращает

- настройки проверки данных, тип [DataValidation](#).
- `None`, если ячейка содержит допустимое значение.

Пример

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

6.74.2 Метод `DataValidationResult.isValid`

Метод возвращает является ли значение ячейки допустимым при текущих настройках проверки данных. При проверке данных по списку текстовых значений, необходимо установить формат ячейки **Текст**.

Вызов

```
bool isValid()
```

Возвращает

– True, если значение ячейки допустимо или проверка данных отсутствует, в ином случае – False.

Пример

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

6.75 Перечисление `DataValidationType`

Перечисление `DataValidationType` содержит типы проверки данных. Используется в методах [DataValidation.getType\(\)](#) и [DataValidation.setType\(\)](#).

Таблица 42 – Описание типов проверки данных

Значение	Описание
<code>DataValidationType_None</code>	Проверка данных отсутствует
<code>DataValidationType_Integer</code>	Не поддерживается
<code>DataValidationType_Fractional</code>	Не поддерживается
<code>DataValidationType_List</code>	Проверка данных по списку доступных значений
<code>DataValidationType_Date</code>	Проверка данных в формате Дата
<code>DataValidationType_Time</code>	Не поддерживается
<code>DataValidationType_TextLength</code>	Не поддерживается
<code>DataValidationType_Custom</code>	Не поддерживается

6.76 Перечисление DatePatterns

Форматы даты представлены в таблице 43. Пример использования см. в главе [DateTimeCellFormatting](#).

Таблица 43 – Форматы даты

Значение	Описание
DatePatterns_DayMonthTextLongYearLong	'mmmm dd, yyyy' для языка en_US
DatePatterns_FullDate	'день недели, mmmm dd, yyyy' для языка en_US
DatePatterns_DayMonthNumberLongYearLong	'mm/dd/yyyy' для языка en_US
DatePatterns_DayMonthNumberLongYearShort	'mm/dd/yy' для языка en_US
DatePatterns_DayMonthNumberShortYearShort	'm/dd/yy' для языка en_US
DatePatterns_DayMonthTextShort	'dd-mmm' для языка en_US
DatePatterns_MonthTextShortYearShort	'mmm-yy' для языка en_US
DatePatterns_DayMonthTextShortYearShort	'mmm dd, yy' для языка en_US
DatePatterns_DayMonthYearLongNonLocalizableHyphen	Нелокализуемый шаблон 'dd-mm-yyyy'
DatePatterns_DayMonthYearLongNonLocalizableSlash	Нелокализуемый шаблон 'dd/mm/yyyy'

6.77 Класс DatePickerControl

Представляет собой элемент управления "Выбор даты", используемый для ввода дат в документе. Является наследником класса [ContentControl](#). Методы `DatePickerControl.GetValue()` и `DatePickerControl.SetValue(DateTime)` позволяют получить и задать значение этого элемента управления.

Пример

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

6.78 Класс DateTime

Класс `DateTime` предоставляет дату и время с точностью до секунды. Используется для поля [TrackedChangeInfo.timeStamp](#).

Таблица 44 – Описание полей класса `DateTime`

Поле	Тип	Описание
<code>DateTime.year</code>	<code>int</code>	Год
<code>DateTime.month</code>	<code>int</code>	Месяц
<code>DateTime.day</code>	<code>int</code>	День
<code>DateTime.hour</code>	<code>int</code>	Часы
<code>DateTime.minute</code>	<code>int</code>	Минуты
<code>DateTime.second</code>	<code>int</code>	Секунды

6.78.1 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")
```

6.78.2 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")
```

6.79 Класс DateTimeCellFormatting

Класс содержит параметры для формата ячеек таблицы типа Дата и Время, используется в качестве аргумента метода [Cell.setFormat\(\)](#).

Таблица 45 – Описание полей класса DateTimeCellFormatting

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

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())
```

6.80 Перечисление DateTimeFormat

В таблице 46 представлены варианты форматирования даты и времени. Используется в качестве параметра метода [CellRange.insertCurrentDateTime\(\)](#).

Таблица 46 – Варианты форматирования даты и времени

Значение	Описание
DateTimeFormat_DateTime	Дата и время
DateTimeFormat_Date	Дата
DateTimeFormat_Time	Время

6.81 Класс Document

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

Пример

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

6.81.1 Метод Document.areMirroredMarginsEnabled

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

Пример

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

6.81.2 Метод Document.calculateAllFormulas

Метод пересчитывает все формулы в документе. Используйте этот метод после изменения данных табличного документа, если функция автоматического пересчета формул отключена. Чтобы узнать текущее состояние этой функции, воспользуйтесь методом [Document.getCalculationMode\(\)](#).

Также вы можете использовать метод [Document.calculateOutdatedFormulas\(\)](#) для пересчета только формул, данные которых были изменены, и метод `calculate()` объектов [Table](#), [CellRange](#) или [Cell](#) для пересчета формул, находящихся в определенном диапазоне.

6.81.3 Метод Document.calculateOutdatedFormulas

Метод пересчитывает формулы, данные которых были изменены. Используйте этот метод после изменения данных табличного документа, если функция автоматического пересчета формул отключена. Чтобы узнать текущее состояние этой функции, воспользуйтесь методом [Document.getCalculationMode\(\)](#).

Также вы можете использовать метод `calculate()` объектов [Table](#), [CellRange](#) или [Cell](#) для пересчета формул, находящихся в определенном диапазоне.

6.81.4 Метод `Document.exportAs`

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

Расширенные версии метода позволяют указать дополнительные настройки экспорта документа:

- для текстовых документов – класс [TextExportSettings](#);
- для табличных документов – класс [WorkbookExportSettings](#);
- для презентационных документов – класс [PresentationExportSettings](#).

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

Примеры

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

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

```
workbookSettings = myOfficeSDK.WorkbookExportSettings()  
workbookSettings.sheetNames = myOfficeSDK.VectorString(['List1', 'List3'])  
workbookSettings.printingScope =  
myOfficeSDK.PrintingScope(myOfficeSDK.PrintingScope.Type_PrintArea)  
workbookSettings.pageProperties = myOfficeSDK.PageProperties(100, 200)  
workbookSettings.scale = 90  
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1, workbookSettings)
```

```
presentationExportSettings = myOfficeSDK.PresentationExportSettings()  
presentationExportSettings.skipHiddenSlides = False  
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1,  
presentationExportSettings)
```

6.81.5 Метод `Document.getAbsoluteFilePath`

Метод возвращает строку, содержащую абсолютный путь к текущему документу. Получаемый путь имеет ОС - зависимый формат (например, содержит символы "/" для Unix и "\" для Windows).

Пример

```
print(document.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.

6.81.6 Метод Document.getBlocks

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

Пример

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

6.81.7 Метод Document.getBookmarks

Метод предоставляет доступ к списку закладок [Bookmarks](#).

Пример

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

6.81.8 Метод Document.getCalculationMode

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

6.81.9 Метод `Document.getComments`

Метод обеспечивает доступ к комментариям документа, возвращает объект [Comments](#).

Пример

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

6.81.10 Метод `Document.getConditionalFormatRules`

Метод позволяет получить коллекцию правил условного форматирования для табличного документа.

Вызов

```
ConditionalFormatDocumentRules getConditionalFormatRules()
```

Возвращает

— коллекция правил условного форматирования для документа, тип [ConditionalFormatDocumentRules](#).

Пример

```
rules = document.getConditionalFormatRules()
for rule in rules:
    if rule.getType() == myOfficeSDK.ConditionalFormatOperatorType_AboveAverage:
        aboveOperator =
myOfficeSDK.castToAboveAverageConditionalFormat(rule.getOperator())
```

6.81.11 Метод `Document.getContentControls`

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

Пример

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

6.81.12 Метод `Document.getFormulaType`

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

Пример

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

6.81.13 Метод `Document.getNamedExpressions`

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

Пример

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

6.81.14 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()
```

6.81.15 Метод `Document.getRange`

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

Пример

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

6.81.16 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())
```

6.81.17 Метод Document.getSections

Возвращает объект типа [Sections](#).

Пример

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

6.81.18 Метод Document.getSectionsEnumerator

Возвращает список объектов типа [Section](#).

Пример

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

6.81.19 Метод Document.getTextStyles

Метод возвращает объект, который позволяет взаимодействовать со стилями абзацев в документе.

Вызов

```
TextStyles getTextStyles()
```

Возвращает

— объект для работы со стилями абзацев, тип [TextStyles](#).

Пример

```
styles = document.getTextStyles()
for style in styles.getEnumerator():
    print(style.getName())
```

6.81.20 Метод `Document.isCalculatedOnSave`

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

6.81.21 Метод `Document.isChangesTrackingEnabled`

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

Пример

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

6.81.22 Метод `Document.isStructureProtected`

Метод возвращает состояние защиты от изменения структуры табличного документа (см. [Защита структуры табличного документа](#)).

Вызов

```
bool isStructureProtected()
```

6.81.23 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 34.

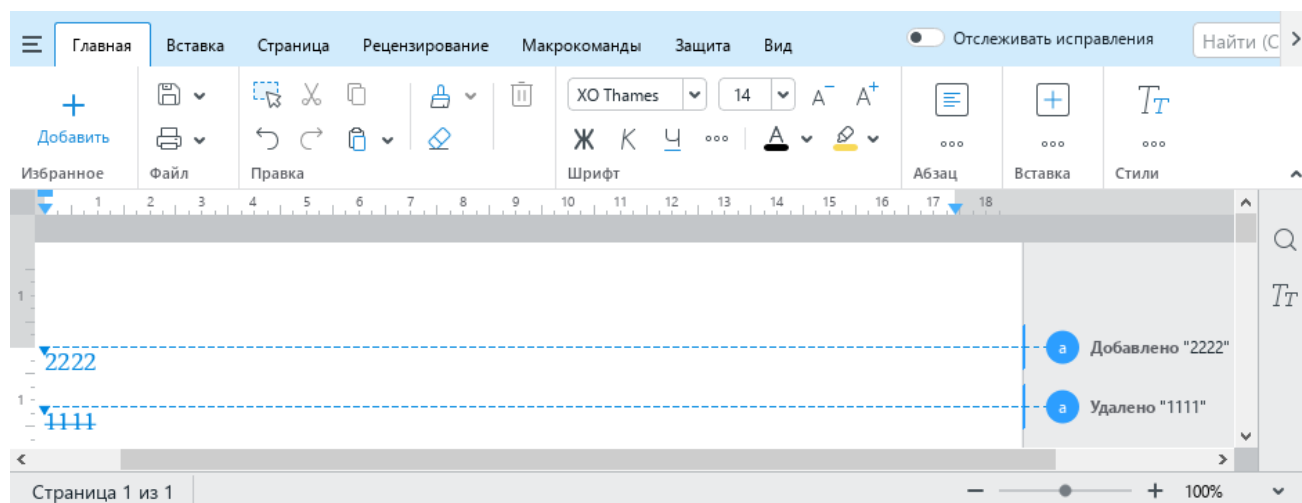


Рисунок 34 – Результат выполнения метода merge

6.81.24 Метод `Document.removeStructureProtection`

Метод снимает защиту от изменений структуры табличного документа (см. [Защита структуры табличного документа](#)).

Вызов

```
removeStructureProtection(password)
```

Параметры

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

Пример

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

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

6.81.25 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)
```

6.81.26 Метод `Document.setCalculatedOnSave`

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

6.81.27 Метод `Document.setCalculationMode`

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

6.81.28 Метод `Document.setChangesTrackingEnabled`

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

Пример

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

6.81.29 Метод `Document.setFormulaType`

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

Пример

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

6.81.30 Метод `Document.setMirroredMarginsEnabled`

Метод позволяет включать и выключать зеркальные поля в документе.

Пример

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

6.81.31 Метод `Document.setPageOrientation`

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

Пример

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

6.81.32 Метод `Document.setPageProperties`

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

Пример

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

6.81.33 Метод `Document.setStructureProtection`

Метод устанавливает защиту от изменений структуры табличного документа (см. [Защита структуры табличного документа](#)).

Вызов

```
setStructureProtection(password)
```

Параметры

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

Пример

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

Если метод `setStructureProtection()` применяется к документу с уже защищенной структурой, возникает исключение `SpreadsheetProtectionError`.

6.82 Перечисление DocumentFormat

В таблице 47 приведены поддерживаемые форматы документов, которые используются в поле [SaveDocumentSettings.documentFormat](#).

Таблица 47 – Форматы документов

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

6.83 Класс DocumentSettings

Класс `DocumentSettings` предоставляет общие настройки документа и используется в методе [Application.createDocument](#) и поле [LoadDocumentSettings.commonDocumentSettings](#).

Таблица 48 – Описание полей класса DocumentSettings

Поле	Тип	Описание
DocumentSettings.documentType	DocumentType	Тип документа
DocumentSettings.userInfo	UserInfo	Информация о пользователе
DocumentSettings.localeInfo	LocaleInfo	Информация о локализации
DocumentSettings.timeZone	TimeZone	Информация о временной зоне
DocumentSettings.formulaType	FormulaType	Система адресации ячеек

6.84 Перечисление DocumentType

В таблице 49 приведены поддерживаемые типы документов, которые используются при создании документа [Application.createDocument\(\)](#), [DocumentSettings](#).

Таблица 49 - Типы документов

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

6.85 Класс DropListControl

Представляет собой элемент управления "Выпадающий список" в документе. Является наследником класса [ContentControl](#). Методы `DropListControl.getValue()` и `DropListControl.setValue(int)` позволяют получить и задать значение этого элемента управления. Метод `DropListControl.getChoices()` возвращает коллекцию элементов, находящихся в выпадающем списке.

Пример

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

6.86 Класс DSVSettings

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

Таблица 50 – Описание полей класса DSVSettings

Поле	Тип	Описание
DSVSettings.autofit	bool	Признак необходимости автоматического подстраивания ширины столбца под размер данных в ячейке
DSVSettings.startBlockIndex	int	Индекс первого листа документа для сохранения
DSVSettings.lastBlockIndex	int	Индекс последнего листа документа для сохранения

Пример

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

6.86.1 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")
```

6.86.2 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")
```

6.87 Перечисление Encoding

В таблице 51 приведены поддерживаемые кодировки документов. Используется в [LoadDocumentSettings](#).

Таблица 51 - Кодировки документов

Значение	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

Значение	Кодировка
Encoding_CP866	CP866

6.88 Перечисление ExportFormat

В таблице 52 приведены поддерживаемые форматы экспорта документов, см. [Document.exportAs\(\)](#).

Таблица 52 - Форматы экспорта документов

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

6.89 Класс Field

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

6.90 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")
```

6.90.1 Метод **Fill.getColor**

Метод возвращает цвет заполнения [Color](#).

6.90.2 Метод **Fill.getUrl**

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

6.90.3 Метод **Fill.isNoFill**

Метод возвращает True, если заполнения нет.

6.91 Класс **FiltersRange**

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

6.91.1 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()
```

6.91.2 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()
```

6.91.3 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](#).

6.91.4 Метод `FiltersRange.getFilters`

Метод возвращает фильтры столбцов, которые находятся в текущем диапазоне фильтрации.

Вызов

```
TableFilters getFilters()
```

Возвращает

- фильтры столбцов, тип [TableFilters](#).
- None, если диапазон фильтрации недействителен.

Пример

```
filtersRange = sheet.createFiltersRange(cellRange)
# ...
filters = filtersRange.getFilters()
```

6.91.5 Метод `FiltersRange.setFilters`

Метод `FiltersRange.setFilters` устанавливает фильтры [TableFilters](#) для столбцов диапазона. Фильтрация выполняется с использованием логической операции AND по столбцам. Нумерация столбцов начинается относительно левой позиции диапазона фильтрации. Если номер столбца в фильтрах превышает диапазон фильтрации, то фильтр будет успешно добавлен, но фильтрация для этого столбца будет пропущена. Такое поведение

полезно, если необходимо изменить размер диапазона фильтрации при этом сохранить ранее определенные фильтры. Диапазон фильтрации должен существовать до вызова этого метода. В настоящее время DocumentAPI позволяет создавать диапазон фильтрации только для всего листа табличного документа. Для создания или изменения размера существующего диапазона используется метод [Table.createFiltersRange\(\)](#).

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](#).

6.92 Класс FootnoteEndnote

Класс `FootnoteEndnote` представляет собой сноску в текстовом документе.

6.92.1 Метод FootnoteEndnote.getPosition

Метод возвращает позицию сноски в тексте.

Вызов

```
Position getPosition()
```

Возвращает

— позиция сноски в тексте, тип [Position](#).

Пример

```
range = document.getRange()
notes = range.getFootnotesEndnotes()
for note in notes:
    note.getPosition().removeForward()
```

6.92.2 Метод `FootnoteEndnote.getRange`

Метод возвращает диапазон содержимого сноски.

Вызов

```
Range getRange()
```

Возвращает

— диапазон содержимого сноски, тип [Range](#).

Пример

```
range = document.getRange()  
notes = range.getFootnotesEndnotes()  
for note in notes:  
    print(note.getRange().extractText())
```

6.92.3 Метод `FootnoteEndnote.getType`

Метод возвращает тип текущей сноски.

Вызов

```
FootnoteEndnoteType getType()
```

Возвращает

— тип сноски, тип [FootnoteEndnoteType](#).

6.93 Перечисление `FootnoteEndnoteType`

Перечисление `FootnoteEndnoteType` содержит типы сносок в текстовом документе. Используется в методе [FootnoteEndnote.getType\(\)](#).

Таблица 53 – Описание типов сносок

Значение	Описание
<code>FootnoteEndnoteType_Footnote</code>	Сноска
<code>FootnoteEndnoteType_Endnote</code>	Концевая сноска

6.94 Класс `FootnotesEndnotes`

Класс `FootnotesEndnotes` представляет собой коллекцию сносок и концевых сносок. Используется в методе [Range.getFootnotesEndnotes\(\)](#).

Пример

```
range = document.getRange()
notes = range.getFootnotesEndnotes()
for note in notes:
    print(note.getRange().extractText())
```

6.95 Перечисление FormulaType

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 54. Используется в [Document.getFormulaType\(\)](#), [Document.setFormulaType\(\)](#), [DocumentSettings](#).

Таблица 54 – Системы адресации ячеек в табличном документе

Значение	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.

6.96 Класс FractionCellFormatting

Класс содержит параметры для дробного формата ячеек таблицы. Используется в качестве аргумента метода [Cell.setFormat\(\)](#).

Таблица 55 – Описание полей класса FractionCellFormatting

Поле	Тип	Описание
FractionCellFormatting.minNumeratorDigits	int	Количество позиций числителя
FractionCellFormatting.minDenominatorDigits	int	Количество позиций знаменателя
FractionCellFormatting.denominatorValue	int	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())
```

6.97 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 35).

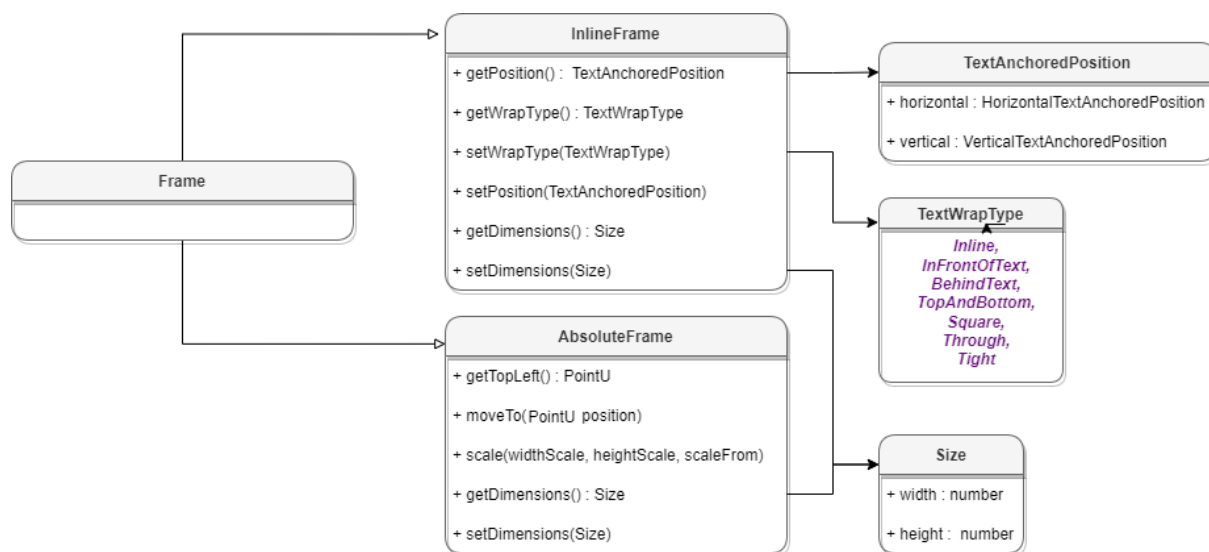


Рисунок 35 – Объектная модель класса `Frame`

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")
```

6.98 Класс FrozenRangePosition

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

6.98.1 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())
```

6.98.2 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())
```

6.98.3 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())
```

6.98.4 Метод `FrozenRangePosition.createFrozenCols`

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

Вызов

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

Пример

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

6.98.5 Метод `FrozenRangePosition.createFrozenRows`

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

Вызов

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

Пример

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

6.98.6 Метод `FrozenRangePosition.isArea`

Возвращает `True` если диапазон является непрерывной областью.

Пример

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

6.98.7 Метод `FrozenRangePosition.isCols`

Возвращает `True` если диапазон состоит из колонок.

Пример

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

6.98.8 Метод `FrozenRangePosition.isRows`

Возвращает `True` если диапазон состоит из строк.

Пример

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

6.98.9 Метод `FrozenRangePosition.isRowsCols`

Возвращает `True` если диапазон содержит строки и колонки.

Пример

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

6.99 Класс `HeaderFooter`

Класс `HeaderFooter` определяет колонтитул текстового документа.

6.99.1 Метод `HeaderFooter.getBlocks`

Метод предоставляет доступ к блокам ([Blocks](#)), которые содержатся в колонтитуле.

Пример

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

6.99.2 Метод `HeaderFooter.getRange`

Метод предоставляет диапазон ([Range](#)) с содержанием верхнего или нижнего колонтитулов.

Пример

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

6.99.3 Метод `HeaderFooter.getType`

Метод предоставляет информацию о типе колонтитула ([HeaderFooterType](#)).

Пример

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

6.100 Перечисление `HeaderFooterType`

Перечисление `HeaderFooterType` содержит типы колонтитулов. Используется в методе [HeaderFooter.getType\(\)](#).

Таблица 56 – Типы колонтитулов

Значение	Описание
HeaderFooterType_Header	Верхний колонтитул
HeaderFooterType_Footer	Нижний колонтитул

Пример

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

6.101 Класс HeadersFooters

Класс HeadersFooters представляет коллекцию верхних и нижних колонтитулов раздела (см. Рисунок 36). Доступ к колонтитулам осуществляется посредством методов [Section.getHeaders\(\)](#), [Section.getFooters\(\)](#).

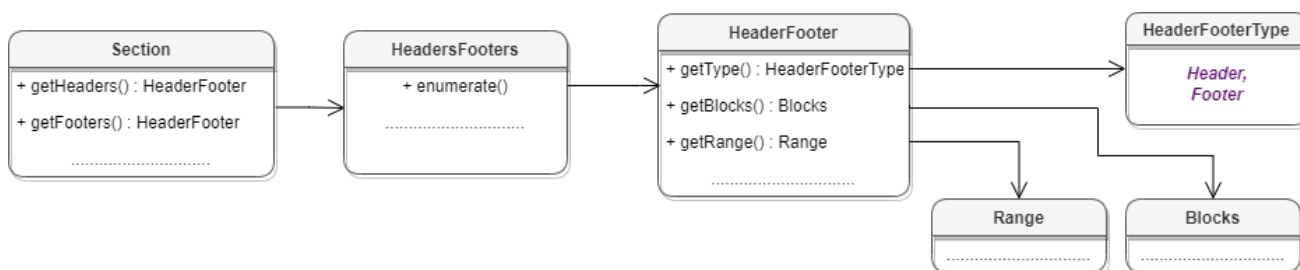


Рисунок 36 – Классы для работы с колонтитулами

6.101.1 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())
```

6.102 Перечисление HorizontalAnchorAlignment

В таблице 57 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали (см. описание класса [HorizontalTextAnchoredPosition](#)).

Таблица 57 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

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

6.103 Перечисление HorizontalRelativeTo

В таблице 58 представлены типы размещения объекта относительно закрепленной позиции по горизонтали (см. описание класса [HorizontalTextAnchoredPosition](#)).

Таблица 58 – Типы размещения объекта относительно закрепленной позиции по горизонтали

Значение	Описание
HorizontalRelativeTo_Character	Символ
HorizontalRelativeTo_Column	Столбец
HorizontalRelativeTo_ColumnLeftMargin	Левое поле столбца
HorizontalRelativeTo_ColumnRightMargin	Правое поле столбца
HorizontalRelativeTo_ColumnInsideMargin	Внутреннее поле столбца
HorizontalRelativeTo_ColumnOutsideMargin	Внешнее поле столбца
HorizontalRelativeTo_Page	Страница
HorizontalRelativeTo_PageContent	Содержимое страницы

Значение	Описание
HorizontalRelativeTo_PageLeftMargin	Левое поле страницы
HorizontalRelativeTo_PageRightMargin	Правое поле страницы
HorizontalRelativeTo_PageInsideMargin	Внутреннее поле страницы
HorizontalRelativeTo_PageOutsideMargin	Внешнее поле страницы

6.104 Класс HorizontalTextAnchoredPosition

Класс `HorizontalTextAnchoredPosition` предназначен для управления относительным положением объекта со смещением или выравниванием по горизонтали (см. описание класса [TextAnchoredPosition](#)).

Таблица 59 – Описание полей класса `HorizontalTextAnchoredPosition`

Поле	Тип	Описание
<code>HorizontalTextAnchoredPosition.relativeTo</code>	HorizontalRelativeTo	Тип размещения объекта относительно закрепленной позиции по горизонтали
<code>HorizontalTextAnchoredPosition.offset</code>	<code>float</code>	Смещение объекта
<code>HorizontalTextAnchoredPosition.alignment</code>	HorizontalAnchorAlignment	Тип выравнивания объекта относительно закрепленной позиции по горизонтали

6.104.1 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")
```

6.104.2 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.alignment =
myOfficeSDK.HorizontalAnchorAlignment_Center
secondHorizontalTextAnchoredPosition.offset = 20

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

6.105 Класс HtmlFragments

Класс `HtmlFragments` содержит HTML элементы, сгенерированные в результате работы метода [`exportWorksheetToHtml\(\)`](#).

Таблица 60 – Описание полей класса `HtmlFragments`

Поле	Тип	Описание
<code>HtmlFragments.rootCss</code>	<code>string</code>	Содержит CSS стили для сгенерированных элементов и соответствующие селекторы
<code>HtmlFragments.body</code>	<code>string</code>	Содержит сгенерированные HTML элементы внутри <code>div</code> контейнера с <code>class="myoffice-worksheet"</code>

6.106 Класс Hyperlink

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

Таблица 61 – Описание полей класса `Hyperlink`

Поле	Тип	Описание
<code>Hyperlink.url</code>	<code>string</code>	Адрес ссылки
<code>Hyperlink.tooltip</code>	<code>string</code>	Текст подсказки
<code>Hyperlink.label</code>	<code>string</code>	Текст описания

Пример

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

6.106.1 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")
```

6.106.2 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")
```

6.107 Класс `IconSetConditionalFormatOperator`

Класс `IconSetConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Значки". Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
IconSetConditionalFormatOperator(ConditionalFormatIconSetEntries entries,
bool isValueShown)
```

Пример

Итог	
	1500
	2500
	800
	1200
	1200
	2400
	1800
	750
	500
	1800
	1050

Рисунок 37 – Пример создания правила "Значки"

```

sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

entries = myOfficeSDK.ConditionalFormatIconSetEntries()
value1 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Percent, "0", False)
entry1 = myOfficeSDK.ConditionalFormatIconSetEntry(value1,
myOfficeSDK.ConditionalFormatIconSet_ThreeArrows, 0)
entries.addEntry(entry1)
value2 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Percent, "33", False)
entry2 = myOfficeSDK.ConditionalFormatIconSetEntry(value2,
myOfficeSDK.ConditionalFormatIconSet_ThreeArrows, 1)
entries.addEntry(entry2)
value3 =
myOfficeSDK.ConditionalFormatValueObject(myOfficeSDK.ConditionalFormatValueObject
Type_Percent, "67", False)
entry3 = myOfficeSDK.ConditionalFormatIconSetEntry(value3,
myOfficeSDK.ConditionalFormatIconSet_ThreeArrows, 2)
entries.addEntry(entry3)

iconSetOperator = myOfficeSDK.createIconSetConditionalFormatOperator(entries,
True)

cellRange = sheet.getCellRange("G2:G12")
cellRangePosition = cellRange.getTableRange()

```

```
iconSetRule = myOfficeSDK.ConditionalFormatRule()  
iconSetRule.setOperator(iconSetOperator)  
iconSetRule.setRange(cellRangePosition)  
rules.addRule(iconSetRule)
```

6.107.1 Метод `IconSetConditionalFormatOperator.getEntries`

Метод возвращает набор правил отображения значков для текущего оператора.

Вызов

```
ConditionalFormatIconSetEntries getEntries()
```

Возвращает

– набор правил отображения значков, тип [ConditionalFormatIconSetEntries](#).

6.107.2 Метод `IconSetConditionalFormatOperator.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.107.3 Метод `IconSetConditionalFormatOperator.isValueShown`

Метод позволяет определить, показывается ли значение ячейки при примененном условном форматировании.

Вызов

```
isValueShown()
```

Возвращает

– True, если значение ячейки показывается при примененном форматировании, в ином случае – False.

6.107.4 Метод `IconSetConditionalFormatOperator.setEntries`

Метод задает набор правил отображения значков для текущего оператора.

Вызов

```
setEntries(entries)
```

Параметры

– `entries`: набор правил отображения значков, тип [ConditionalFormatIconSetEntries](#).

6.107.5 Метод `IconSetConditionalFormatOperator.setValueShown`

Метод позволяет настроить отображение значения ячейки при примененном условном форматировании.

Вызов

```
setValueShown(isValueShown)
```

Параметры

– `isValueShown`: `True`, чтобы показывать значение ячейки при примененном форматировании, в ином случае – `False`.

6.108 Класс `Image`

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

6.108.1 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())
```

6.108.2 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
```

6.108.3 Метод `Image.replaceURL`

Метод заменяет текущее изображение.

Вызов

```
replaceURL(newURL)
```

Параметры

– `newURL`: путь к новому файлу изображения, тип `string`.

Пример

```
range = document.getRange()
images = range.getImages()
for image in images:
    image.replaceURL("logo2025.png")
    image.getFrame().setDimensions(myOfficeSDK.SizeU(150, 150))
```

6.109 Класс `Images`

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

6.109.1 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())
```

6.110 Класс `InlineFrame`

Класс `InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 38). Предназначен для получения и изменения свойств позиции графических объектов. Используется в текстовом документе.

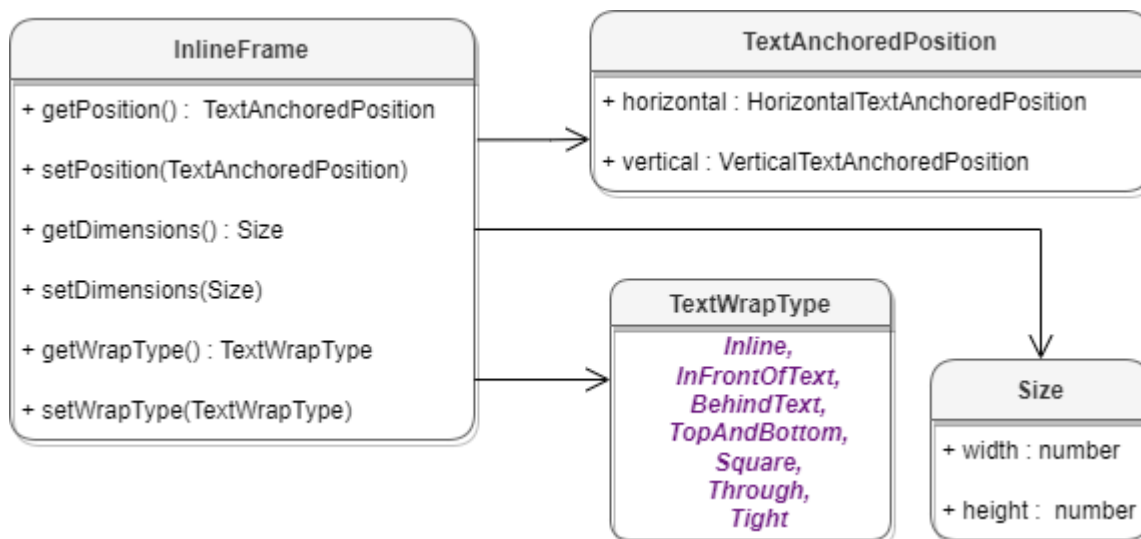


Рисунок 38 – Объектная модель класса `InlineFrame`

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")
```

6.110.1 Метод `InlineFrame.getDimensions`

Метод возвращает задает размеры встроенного объекта, тип - [SizeU](#).

Пример

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

6.110.2 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)
```

6.110.3 Метод `InlineFrame.getWrapType`

Метод возвращает вариант обтекания текстом встроенного объекта (см. [TextWrapType](#)).

Пример

```
mediaObjects = document.getRange().getInlineObjects()  
mediaObjectsEnumerator = mediaObjects.getEnumerator()
```

```
for mediaObject in mediaObjectsEnumerator:
    frame = mediaObject.getFrame()
    if (isinstance(frame, myOfficeSDK.InlineFrame)):
        wrapType = frame.getWrapType()
        print(wrapType)
```

6.110.4 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))
```

6.110.5 Метод `InlineFrame.setPosition`

Метод задает положение встроенного объекта, тип аргумента [TextAnchoredPosition](#). Новая позиция может быть установлена только для встроенных объектов, тип переноса текста которых не является типом [TextWrapType.Inline](#).

Примеры

```
// Позиция встроенного объекта не может быть задана,
// если стиль переноса текста - inline.
// Сначала его следует изменить на тип, отличный от inline.
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    wrapType = frame.getWrapType()
    if (wrapType == myOfficeSDK.TextWrapType_Inline):
        frame.setWrapType(myOfficeSDK.TextWrapType_TopAndBottom)
```

Используя классы [HorizontalTextAnchoredPosition](#), [VerticalTextAnchoredPosition](#), можно задать положение встроенных объектов в текстовом документе с учетом относительного смещения.

```
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    position = myOfficeSDK.TextAnchoredPosition()
```

```
position.horizontal =
myOfficeSDK.HorizontalTextAnchoredPosition(myOfficeSDK.HorizontalRelativeTo_Page)
position.horizontal.offset = 10
position.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageTopMargin)
position.vertical.offset = 15
frame.setPosition(position)
```

Используя классы [HorizontalTextAnchoredPosition](#), [VerticalTextAnchoredPosition](#), можно задать положение встроенных объектов в текстовом документе с учетом относительного выравнивания.

```
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    frame.setWrapType(myOfficeSDK.TextWrapType_TopAndBottom)
    position = myOfficeSDK.TextAnchoredPosition()
    position.horizontal =
myOfficeSDK.HorizontalTextAnchoredPosition(myOfficeSDK.HorizontalRelativeTo_Page)
    position.horizontal.alignment = myOfficeSDK.HorizontalAnchorAlignment_Center
    position.vertical =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageTopMargin)
    position.vertical.alignment = myOfficeSDK.VerticalAnchorAlignment_Top
    frame.setPosition(position)
```

Используя типы смещения [HorizontalRelativeTo.Column](#) и [VerticalRelativeTo.Page](#), можно установить абсолютное положение встроенного объекта в текстовом документе.

```
frame = mediaObject.getFrame()
if (isinstance(frame, myOfficeSDK.InlineFrame)):
    frame.setWrapType(myOfficeSDK.TextWrapType_TopAndBottom)
    position = myOfficeSDK.TextAnchoredPosition()
    position.horizontal =
myOfficeSDK
.HorizontalTextAnchoredPosition(myOfficeSDK.HorizontalRelativeTo_Column)
    position.horizontal.offset = 125
    position.vertical =
myOfficeSDK.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Page)
```

```
position.vertical.offset = 545
frame.setPosition(position)
```

6.110.6 Метод `InlineFrame.setWrapType`

Метод устанавливает вариант обтекания текстом встроенного объекта (см. [TextWrapType](#)).

Пример

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

6.111 Класс `InputFieldControl`

Представляет собой элемент управления "Поле ввода" в документе. Является наследником класса [ContentControl](#). Методы `InputFieldControl.getValue()` и `InputFieldControl.setValue(string)` позволяют получить и задать значение этого элемента управления.

Пример

```
controls = document.getContentControls()
inputField = controls.findByTitle("input").toInputField()
inputField.setValue(inputField.getValue().replace(' ', '_'))
```

6.112 Класс `Insets`

Класс `Insets` предназначен для задания полей, например, страницы. Поля класса `Insets` представлены в таблице 62. Используется в поле `margins` класса [PageProperties](#).

Таблица 62 – Описание полей класса `Insets`

Поле	Тип	Описание
<code>Insets.left</code>	float	Левая граница поля
<code>Insets.top</code>	float	Верхняя граница поля

Поле	Тип	Описание
Insets.right	float	Правая граница поля
Insets.bottom	float	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)
```

6.112.1 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")
```

6.112.2 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")
```

6.113 Класс LineEndingProperties

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

Таблица 63 – Описание полей класса LineEndingProperties

Поле	Тип	Описание
LineEndingProperties.style	LineEndingStyle	Стиль окончания линии
LineEndingProperties.relativeExtent	Size	Размер окончания линии относительно ее ширины

Пример

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

lineProperties = myOfficeSDK.LineProperties()
lineProperties.headLineEndingProperties = myOfficeSDK.LineEndingProperties()
lineProperties.headLineEndingProperties.style = myOfficeSDK.LineEndingStyle_Arrow
lineProperties.headLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
lineProperties.headLineEndingProperties.relativeExtent.width = 2
lineProperties.headLineEndingProperties.relativeExtent.height = 2
```

```
lineProperties.tailLineEndingProperties = myOfficeSDK.LineEndingProperties()
lineProperties.tailLineEndingProperties.style = myOfficeSDK.LineEndingStyle_Arrow
lineProperties.tailLineEndingProperties.relativeExtent = myOfficeSDK.SizeU()
lineProperties.tailLineEndingProperties.relativeExtent.width = 2
lineProperties.tailLineEndingProperties.relativeExtent.height = 2

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

borders = myOfficeSDK.Borders()
borders.setTop(lineProperties)
cell.setBorders(borders)
```

6.113.1 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")
```

6.113.2 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")
```

6.114 Перечисление LineEndingStyle

В таблице 64 приведены типы окончания линии. Используется в поле `style` класса [LineEndingProperties](#).

Таблица 64 – Типы окончания линии

Значение	Описание
LineEndingStyle_Arrow	
LineEndingStyle_Diamond	
LineEndingStyle_Oval	
LineEndingStyle_Stealth	
LineEndingStyle_Triangle	
LineEndingStyle_None	

Пример

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

lineProperties = myOfficeSDK.LineProperties()
```

```
lineProperties.headLineEndingProperties = myOfficeSDK.LineEndingProperties()
lineProperties.headLineEndingProperties.style = myOfficeSDK.LineEndingStyle_Arrow

borders = myOfficeSDK.Borders()
borders.setTop(lineProperties)
cell.setBorders(borders)
```

6.115 Класс LineProperties

Класс `LineProperties` предназначен для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 39).

Таблица 65 – Описание полей класса `LineProperties`

Поле	Тип	Описание
<code>LineProperties.style</code>	LineStyle	Тип линии
<code>LineProperties.width</code>	float	Толщина линии
<code>LineProperties.color</code>	Color	Цвет линии
<code>LineProperties.headLineEndingProperties</code>	LineEndingProperties	Тип начала линии
<code>LineProperties.tailLineEndingProperties</code>	LineEndingProperties	Тип окончания линии

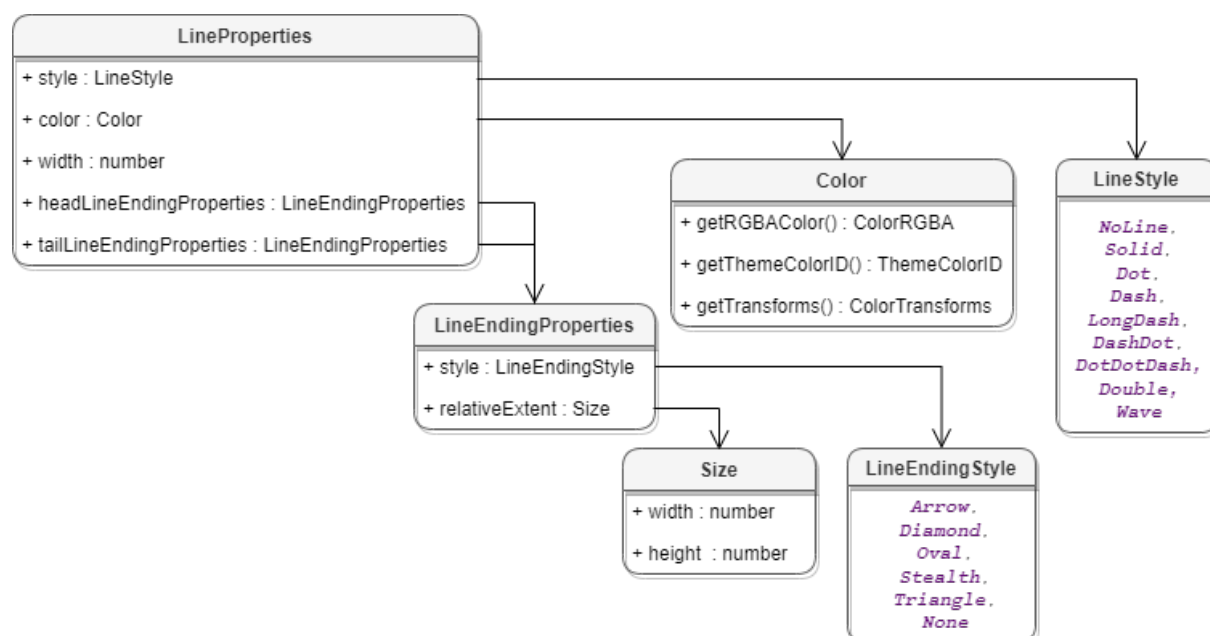


Рисунок 39 – Свойства границ ячеек

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)
```

6.115.1 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")
```

6.115.2 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")

```

6.116 Класс LineSpacing

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

Таблица 66 – Параметры межстрочного интервала

Поле	Тип	Описание
<code>LineSpacing.value</code>	<code>float</code>	Значение межстрочного интервала
<code>LineSpacing.rule</code>	LineSpacingRule	Правило формирования межстрочного интервала

Пример

```

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

```

6.116.1 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")
```

6.116.2 LineSpacing.__ne__

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

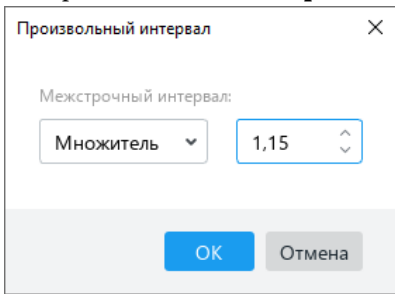
Пример

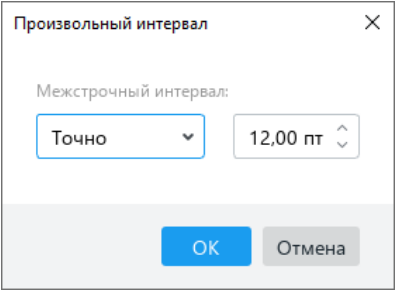
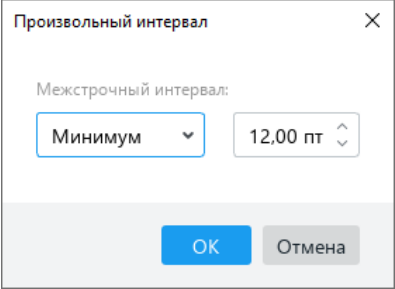
```
lineSpacingExact = myOfficeSDK.LineSpacing(10, myOfficeSDK.LineSpacingRule_Exact)
lineSpacingMultiple = myOfficeSDK.LineSpacing(10,
myOfficeSDK.LineSpacingRule_Multiple)
if lineSpacingExact.__ne__(lineSpacingMultiple):
    print("Ne")
```

6.117 Перечисление LineSpacingRule

Перечисление `LineSpacingRule` содержит типы межстрочного интервалов. В таблице 67 представлены описания правил формирования межстрочного интервала текстового абзаца.

Таблица 67 – Виды межстрочного интервала

Значение	Описание
<code>LineSpacingRule_Multiple</code>	Установка произвольного межстрочного интервала с использованием множителя. При вызове необходимо указать значение множителя, например: <pre>pPr.lineSpacing = myOfficeSDK.LineSpacing(1.15, myOfficeSDK.LineSpacingRule_Multiple)</pre> В данном примере используется значение множителя 1.15. Действие команды аналогично ручной настройке межстрочного интервала в диалоге «Произвольный интервал». 

Значение	Описание
LineSpacingRule_Exact	Установка произвольного межстрочного интервала с использованием точного значения. При вызове необходимо указать точное значение, например: <pre>pPr.lineSpacing = myOfficeSDK.LineSpacing(12.0, myOfficeSDK.LineSpacingRule_Exact)</pre> В данном примере используется точное значение 12.0. Действие команды аналогично ручной настройке межстрочного интервала в диалоге «Произвольный интервал». 
LineSpacingRule_AtLeast	Установка произвольного межстрочного интервала с использованием минимального значения. При вызове необходимо указать минимальное значение, например: <pre>pPr.lineSpacing = myOfficeSDK.LineSpacing(12.0, myOfficeSDK.LineSpacingRule_AtLeast)</pre> В данном примере используется минимальное значение 12.0. Действие команды аналогично ручной настройке межстрочного интервала в диалоге «Произвольный интервал». 

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)
```

6.118 Перечисление LineStyle

В таблице 68 приведены типы линий. Используется в поле style класса [LineProperties](#).

Таблица 68 – Типы линий

Значение	Описание
LineStyle_NoLine	Нет линии
LineStyle_Solid	
LineStyle_Dot	
LineStyle_Dash	
LineStyle_LongDash	
LineStyle_DashDot	
LineStyle_DotDotDash	
LineStyle_Double	
LineStyle_Wave	

Пример

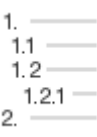
```
firstSheet = document.getBlocks().getTable(0)
cell = firstSheet.getCell("C3")

lineProperties = myOfficeSDK.LineProperties()
lineProperties.style = myOfficeSDK.LineStyle_Solid
```

6.119 Перечисление ListSchema

Перечисление ListSchema содержит типы схем форматирования списков, которые могут быть применены к абзацам текста. Данные значения используются в методах [Paragraph.getListSchema\(\)](#), [Paragraph.setListSchema\(\)](#).

Таблица 69 – Описания схем списков

Значение	Тип схемы списка	Изображение
ListSchema_Unknown	Неизвестно.	
ListSchema_UnknownBullet	Список без маркера.	Соответствует варианту «нет»
ListSchema_UnknownNumbering	Нумерация без номера.	Соответствует варианту «нет»
ListSchema_BulletCircleSolid	Список с маркерами в виде круга.	
ListSchema_BulletCircleContour	Список с маркерами в виде окружности.	
ListSchema_BulletSquareSolid	Список с маркерами в виде квадрата	
ListSchema_BulletDiamondDots	Список с маркерами в виде четырех ромбов.	
ListSchema_BulletHyphen	Список с маркерами в виде дефиса.	
ListSchema_BulletConcaveArrowSolid	Список с маркерами в виде вогнутой стрелки.	
ListSchema_BulletCheckmark	Список с маркерами в виде галочки.	
ListSchema_EnumeratorDecimalDot	Десятичная нумерация с точкой.	

Значение	Тип схемы списка	Изображение
ListSchema_Unknown	Неизвестно.	
ListSchema_EnumeratorDecimalDotMultiLevel	Многоуровневая десятичная нумерация с точкой.	1.1 _____ a. _____ b. _____ i. _____ 1.2 _____
ListSchema_EnumeratorDecimalBracket	Десятичная нумерация со скобкой.	1) _____ a) _____ b) _____ i) _____ 2) _____
ListSchema_EnumeratorLatinUppercaseDot	Нумерация латинскими прописными буквами с точкой.	A. _____ 1. _____ 2. _____ i. _____ B. _____
ListSchema_EnumeratorLatinLowercaseDot	Нумерация латинскими строчными буквами с точкой.	a. _____ 1. _____ 2. _____ i. _____ b. _____
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)
```

6.120 Класс LoadDocumentSettings

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

Таблица 70 – Описание полей класса `LoadDocumentSettings`

Поле	Тип	Описание
<code>LoadDocumentSettings.commonDocumentSettings</code>	DocumentSettings	Общие настройки документа
<code>LoadDocumentSettings.encoding</code>	Encoding	Кодировка документа
<code>LoadDocumentSettings.dsvSettings</code>	DSVSettings	Настройки для работы с файлами CSV и DSV
<code>LoadDocumentSettings.documentPassword</code>	string	Пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows.

6.121 Класс LocaleInfo

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

Таблица 71 – Описание полей класса `LocaleInfo`

Поле	Тип	Описание
<code>LocaleInfo.localeName</code>	string	Название локализации, представлено в формате <language> <REGION> , где языковой код соответствует стандарту ISO-639, а код региона стандарту ISO-3166.
<code>LocaleInfo.decimalSeparator</code>	string	Десятичный разделитель, отделяет целые и дробные части чисел.

Поле	Тип	Описание
<code>LocaleInfo.thousandSeparator</code>	string	Символ, разделяющий группы цифр в числовых значениях.
<code>LocaleInfo.listSeparator</code>	string	Символ, отделяющий элементы в списке.
<code>LocaleInfo.currencySymbol</code>	string	Символ валюты, используемой в текущей стране или регионе.
<code>LocaleInfo.currencyFormat</code>	CurrencySignPlacement	Расположение знака валюты в текущем регионе.
<code>LocaleInfo.shortDatePattern</code>	string	Заданный «короткий» формат отображения даты в текущем регионе (например, 'm/d/yy' для en_US).
<code>LocaleInfo.longDatePattern</code>	string	Заданный «длинный» формат отображения даты в текущем регионе (например, 'dddd, mmmm d, уууу' для en_US).
<code>LocaleInfo.timePattern</code>	string	Заданный формат отображения времени в текущем регионе (например, 'h:mm AM/PM' для en_US).

6.122 Класс `MediaObject`

Класс `MediaObject` представляет собой встроенный объект документа.

6.122.1 Метод `MediaObject.getFrame`

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

Пример

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

6.122.2 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")
```

6.122.3 Метод `MediaObject.toImage`

Метод возвращает изображение [Image](#), связанное со встроенным объектом. Если объект не является изображением, метод возвращает `nil`.

Пример

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

6.123 Класс `MediaObjects`

Класс `MediaObjects` предназначен для доступа к коллекции графических объектов. Объект может быть получен вызовом методов [Range.getInlineObjects\(\)](#), [Table.getMediaObjects\(\)](#) (см. Рисунок 40).

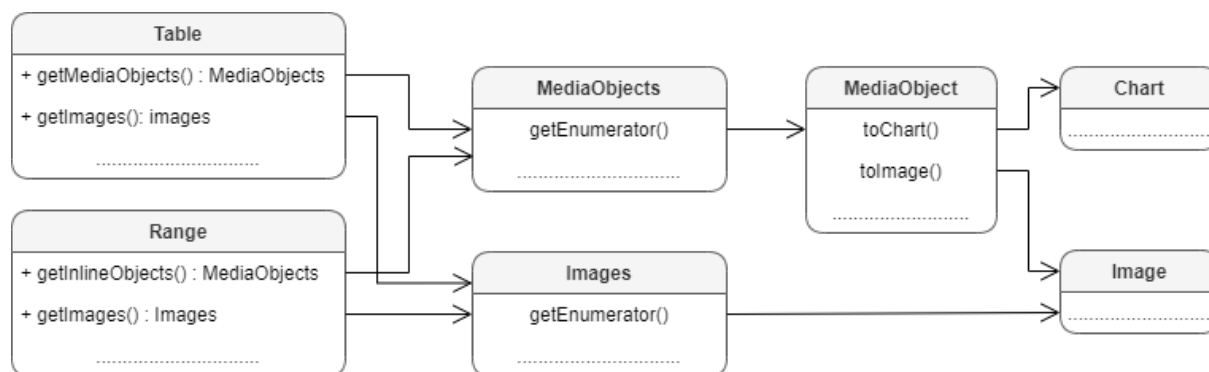


Рисунок 40 – Графические объекты

6.123.1 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")
```

6.124 Класс Message

Класс Message предназначен для формирования событий лога.

6.124.1 Перечисление Message.Severity

Перечисление Message.Severity (Таблица 72) описывает уровни сообщений лога (информация, предупреждение, ошибка).

Таблица 72 – Описание уровней лога `Message.Severity`

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

6.124.2 Метод `Message.getSeverity`

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

6.124.3 Метод `Message.getText`

Метод возвращает текст сообщения.

6.124.4 Метод `Message.makeError`

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

6.124.5 Метод `Message.makeInfo`

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

6.124.6 Метод `Message.makeWarning`

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

6.125 Класс `Messenger`

Служит для организации логов приложений.

6.125.1 Метод **Messenger.notify**

Метод используется для создания события лога

Пример

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

6.125.2 Метод **Messenger.subscribe**

Метод служит для подписки на события лога.

Пример

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

6.126 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())
```

6.126.1 Метод **NamedExpression.getCellRange**

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

6.126.2 Метод **NamedExpression.getExpression**

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

6.126.3 Метод `NamedExpression.getName`

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

6.126.4 Метод `NamedExpression.setName`

Метод позволяет переименовать именованный диапазон.

Вызов

```
setName(newName)
```

Параметры

– `newName`: новое имя диапазона, тип `string`.

Пример

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

6.127 Класс `NamedExpressions`

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

6.127.1 Метод `NamedExpressions.addExpression`

Добавляет новый диапазон в список именованных диапазонов.

Пример

```
expressionName = "Продажи"  
expressionValue = "=Формула покупки!$A$6:$A$14"  
namedExpressions.addExpression(expressionName, expressionValue)  
namedExpression = namedExpressions.get(expressionName)  
if namedExpression != None:  
    print(namedExpression.getName())
```

6.127.2 Метод `NamedExpressions.get`

Возвращает именованный диапазон [NamedExpression](#) по имени, в случае, если оно существует.

Пример

```
firstSheet = document.getBlocks().getTable(0)
namedExpressions = firstSheet.getNamedExpressions()
namedExpression = namedExpressions.get(expressionName)
if namedExpression != None:
    print(namedExpression.getName())
```

6.127.3 Метод `NamedExpressions.getEnumerator`

Позволяет получить доступ ко всему списку именованных диапазонов.

Пример

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

6.127.4 Метод `NamedExpressions.removeExpression`

Удаляет именованный диапазон по заданному имени.

Пример

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

6.128 Класс `NullaryConditionalFormatOperator`

Класс `NullaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил без параметров. Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
NullaryConditionalFormatOperator(ConditionalFormatNullaryCondition
condition)
```

Пример

Дата заказа
10 января 2025 г.
15 февраля 2025 г.
1 марта 2025 г.
20 апреля 2025 г.
5 мая 2025 г.
18 марта 2025 г.
2 июня 2025 г.
15 июля 2025 г.
1 августа 2025 г.
20 сентября 2025 г.
1 октября 2025 г.

Рисунок 41 – Пример создания правила для дат в прошлом месяце

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

nullaryStyle = myOfficeSDK.ConditionalFormatCellStyle()
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
nullaryStyle.cellProperties = cellProperties

cellRange = sheet.getCellRange("B2:B12")
cellRangePosition = cellRange.getTableRange()

nullaryOperator =
myOfficeSDK.createNullaryConditionalFormatOperator(myOfficeSDK.ConditionalFormatN
ullaryCondition_LastMonth)

nullaryRule = myOfficeSDK.ConditionalFormatRule(nullaryOperator, nullaryStyle,
cellRangePosition, False)
rules.addRule(nullaryRule)
```

6.128.1 Метод NullaryConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
ConditionalFormatNullaryCondition getCondition()
```

Возвращает

– условие применения форматирования, тип [ConditionalFormatNullaryCondition](#).

6.128.2 Метод `NullaryConditionalFormatOperator.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.129 Класс `NumberCellFormatting`

Класс содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода [Cell.setFormat\(\)](#).

Таблица 73 – Описание полей класса `NumberCellFormatting`

Поле	Тип	Описание
<code>NumberCellFormatting.decimalPlaces</code>	<code>int</code>	Количество десятичных позиций
<code>NumberCellFormatting.useThousandsSeparator</code>	<code>bool</code>	Использовать разделитель для тысячных
<code>NumberCellFormatting.useRedForNegative</code>	<code>bool</code>	Использовать красный цвет для отрицательных значений
<code>NumberCellFormatting.useBracketsForNegative</code>	<code>bool</code>	Использовать скобки для отрицательных значений
<code>NumberCellFormatting.hideSign</code>	<code>bool</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())
```

6.130 Перечисление PageFieldOrder

Перечисление `PageFieldOrder` содержит виды отображения полей из области фильтров. Используется в поле [PivotTableLayoutSettings.pageFieldOrder](#).

Таблица 74 – Виды отображения полей из области фильтров

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

6.131 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
pageNumbers = myOfficeSDK.PageNumbers([1, 13, 25])
```

```
# диапазон страниц
pageNumbers = myOfficeSDK.PageNumbers(1, 20)
```

6.131.1 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))
```

6.131.2 Метод PageNumbers.getLast

Метод `PageNumbers.getLast` возвращает последний номер страницы.

Пример

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

6.131.3 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")
```

6.131.4 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")
```

6.132 Перечисление PageOrientation

Перечисление PageOrientation содержит варианты ориентации страницы документа. Может быть использована для получения / установки ориентации страниц для секции или документа в методах [Section.setPageOrientation\(\)](#), [Section.getPageOrientation\(\)](#).

Таблица 75 – Варианты ориентации страницы документа

Значение	Описание
PageOrientation_Landscape	Альбомная ориентация
PageOrientation_Portrait	Книжная ориентация

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())
```

6.133 Перечисление PageParity

Варианты выбора страниц для экспорта и печати представлены в таблице 76. Используется в [PageNumbers](#).

Таблица 76 – Варианты выбора страниц для экспорта и печати

Значение	Описание
PageParity_Odd	Только нечетные страницы
PageParity_Even	Только четные страницы
PageParity_All	Все страницы

6.134 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). Используется в [Document.setPageProperties\(\)](#), [Section.getPageProperties\(\)](#), [Section.setPageProperties\(\)](#).

Таблица 77 – Описание полей класса PageProperties

Поле	Тип	Описание
PageProperties.height	float	Высота страницы
PageProperties.width	float	Ширина страницы
PageProperties.margins	Insets	Поля страницы

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)
```

6.134.1 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")
```

6.134.2 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")
```

6.135 Класс Paragraph

Класс `Paragraph` предоставляет доступ к свойствам абзаца (см. Рисунок 42).

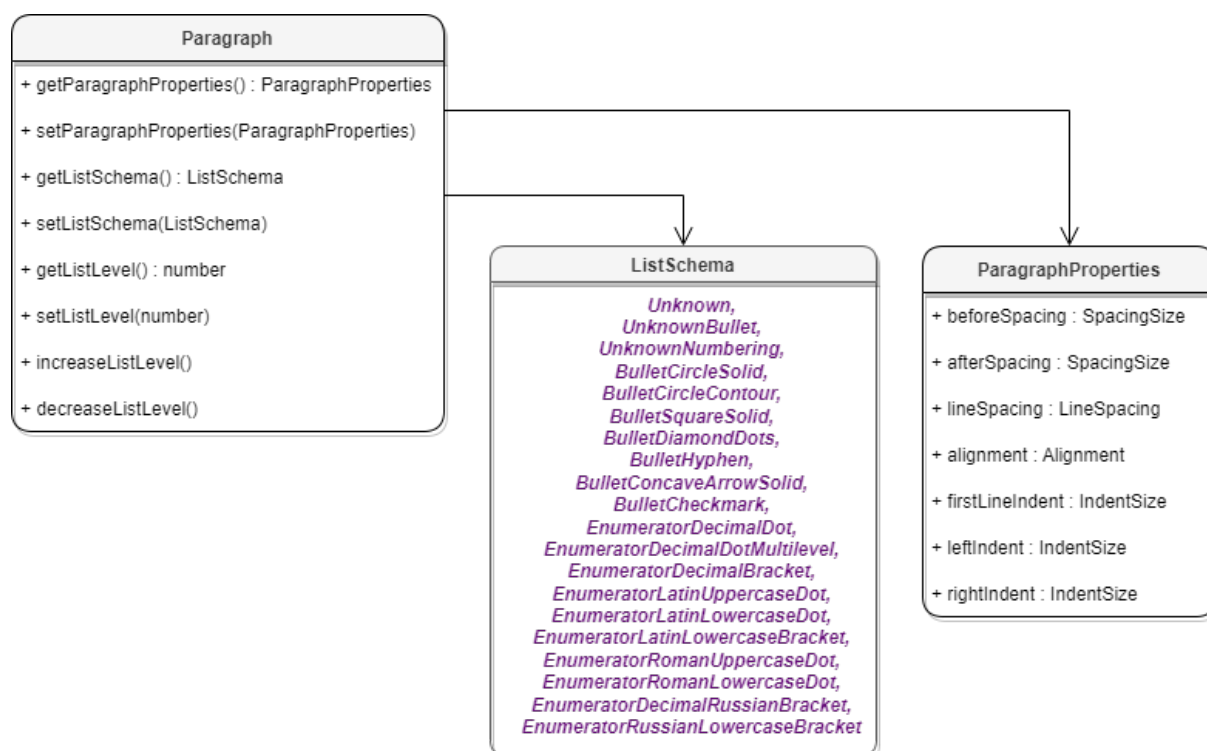


Рисунок 42 – Объектная модель классов для работы со свойствами абзаца

6.135.1 Метод Paragraph.decreaseListLevel

Метод позволяет уменьшить на единицу глубину вложенности элемента списка. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе. Если схема нумерации для абзаца не установлена, будет вызвано исключение [DocumentModificationError](#).

Пример

```
blocks = document.getBlocks()
paragraph = blocks.getParagraph(0)
if paragraph != None:
    try:
        paragraph.decreaseListLevel()
        print(paragraph.getListLevel())
    except myOfficeSDK.DocumentModificationError as err:
        print(err)
```

6.135.2 Метод Paragraph.getListLevel

Метод позволяет получить глубину вложенности элемента списка. Данный метод используется только в текстовом документе для абзаца, который является частью маркированного или нумерованного списка (для абзаца установлен тип списка [ListSchema](#)). В противном случае метод вернет None.

Пример

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

6.135.3 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())
```

6.135.4 Метод Paragraph.getParagraphProperties

Метод предоставляет доступ к классу, определяющему такие свойства абзаца [ParagraphProperties](#), как выравнивание текста, межстрочные интервалы, отступы и т. д.

Пример для текстового документа

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

6.135.5 Метод Paragraph.getResolvedStyle

Метод возвращает комбинированный стиль текущего абзаца. Комбинированным стилем является стиль всего абзаца, общий стиль всех его частей или стиль по умолчанию. Данный стиль отображается в интерфейсе редактора.

Вызов

```
TextStyle getResolvedStyle()
```

Возвращает

– комбинированный стиль текущего абзаца, тип [TextStyle](#).

Пример

```
paragraph = document.getRange().getBegin().getParagraph()
print(paragraph.getResolvedStyle().getName())
```

6.135.6 Метод Paragraph.getStyle

Метод возвращает стиль текущего абзаца.

Вызов

```
TextStyle getStyle()
```

Возвращает

– стиль текущего абзаца, тип [TextStyle](#).

Пример

```
paragraphs = document.getRange().getParagraphs()
for paragraph in paragraphs:
    print(paragraph.getStyle().getName())
```

6.135.7 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)
```

6.135.8 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)
```

6.135.9 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)
```

6.135.10 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)
```

6.135.11 Метод Paragraph.setStyle

Метод задает стиль абзаца.

Вызов

```
setStyle(textStyle)
```

Параметры

– textStyle: стиль абзаца, тип [TextStyle](#).

Пример

```
styles = document.GetTextStyles()  
normalStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Normal)  
paragraphs = document.getRange().getParagraphs()  
for paragraph in paragraphs:  
    paragraph.setStyle(normalStyle)
```

6.136 Класс ParagraphProperties

Класс ParagraphProperties предназначен для управления свойствами форматирования абзаца (см. Рисунок 43). Класс ParagraphProperties используется в методах

[Paragraph.getParagraphProperties\(\)](#),
[Paragraph.setParagraphProperties\(\)](#), [Cell.getParagraphProperties\(\)](#),
[Cell.setParagraphProperties\(\)](#), [CellRange.getParagraphProperties\(\)](#) и
[CellRange.setParagraphProperties\(\)](#).

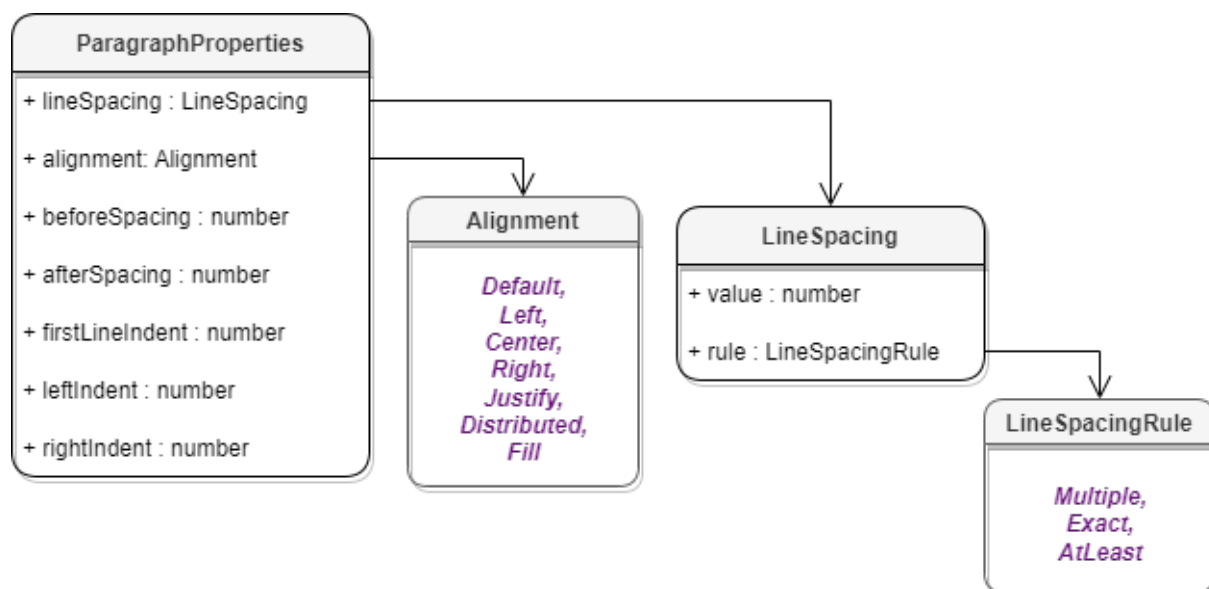
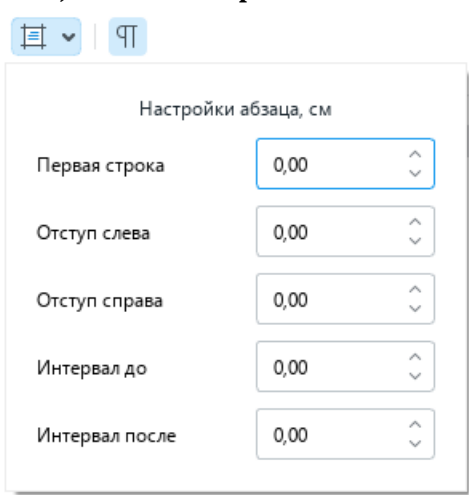


Рисунок 43 – Объектная модель классов для работы со свойствами абзаца

Описание полей класса [ParagraphProperties](#) представлено в таблице 78.

Таблица 78 – Описание полей класса ParagraphProperties

Поле	Тип	Описание
ParagraphProperties.beforeSpacing	float	Установка величины расстояния до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца, (см. рисунок ниже), в поле Интервал до. 
ParagraphProperties.afterSpacing	float	Установка величины расстояния после абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в

Поле	Тип	Описание
		диалоговом окне Настройки абзаца , в поле Интервал после .
ParagraphProperties.lineSpacing	LineSpacing	Расстояние между строк одного абзаца (межстрочный интервал).
ParagraphProperties.alignment	Alignment	Выравнивание текстового фрагмента по горизонтали.
ParagraphProperties.firstLineIndent	float	Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , (см. рисунок выше), в поле Первая строка .
ParagraphProperties.leftIndent	float	Расстояние от левого поля документа до абзаца (отступ слева). При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , (см. рисунок выше), в поле Отступ слева .
ParagraphProperties.rightIndent	float	Расстояние от правого поля документа до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , (см. рисунок выше), в поле Отступ справа .

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)
```

6.136.1 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")
```

6.136.2 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")
```

6.137 Класс Paragraphs

Класс `Paragraphs` предоставляет доступ к коллекции абзацев типа [Paragraph](#) (см. Рисунок 44). Коллекция абзацев может быть получена из объекта [Range](#) посредством использования метода [Range.getParagraphs\(\)](#).

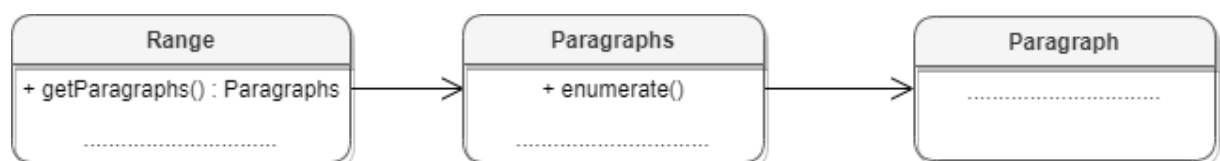


Рисунок 44 – Объектная модель для работы со списком абзацев

Пример для текстового документа

```
range = document.getRange()  
paragraphs = range.getParagraphs()
```

Пример для табличного документа

```
firstSheet = document.getBlocks().getTable(0)  
cell = firstSheet.getCell("B2")  
range = cell.getRange()  
paragraphs = range.getParagraphs()
```

6.137.1 Метод Paragraphs.decreaseListLevel

Метод уменьшает уровень списка на единицу. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе.

Пример

```
docRange = document.getRange()  
paragraphs = docRange.getParagraphs()  
paragraphs.decreaseListLevel()
```

6.137.2 Метод Paragraphs.getEnumerator

Метод позволяет перечислить коллекцию абзацев.

Пример

```
docRange = document.getRange()  
paragraphs = docRange.getParagraphs()  
paragraphsEnumerator = paragraphs.getEnumerator()  
for paragraph in paragraphsEnumerator:  
    print(paragraph.getRange().extractText())
```

6.137.3 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()
```

6.137.4 Метод Paragraphs.setListLevel

Метод устанавливает глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример

```
docRange = document.getRange()  
paragraphs = docRange.getParagraphs()  
paragraphs.setListLevel(1)
```

6.137.5 Метод Paragraphs.setListSchema

Метод устанавливает тип маркированного или нумерованного списка [ListSchema](#). Данный метод используется только в текстовом документе.

Пример

```
docRange = document.getRange()  
paragraphs = docRange.getParagraphs()  
paragraphs.setListSchema(myOfficeSDK.ListSchema_BulletCheckmark)
```

6.138 Класс PercentageCellFormatting

Содержит параметр для процентного формата ячеек таблицы, используется в качестве аргумента метода [Cell.setFormat\(\)](#).

Таблица 79 – Описание полей класса PercentageCellFormatting

Поле	Тип	Описание
PercentageCellFormatting.decimalPlaces	int	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())
```

6.139 Класс PivotTable

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

6.139.1 Метод PivotTable.areAllFiltersInDefaultState

Метод позволяет определить, применен ли к сводной таблице хоть один фильтр.

Вызов

```
bool areAllFiltersInDefaultState()
```

Возвращает

– True, если к сводной таблице не применен ни один фильтр, в ином случае – False.

6.139.2 Метод PivotTable.changeSourceRange

Метод позволяет задать новый диапазон исходных данных сводной таблицы без обновления самой таблицы. Параметр `sourceRange` – строка, представляющая новый диапазон таблицы.

Пример

```
pivotTable.changeSourceRange("I3:K5")  
sourceRange = pivotTable.getSourceRange()  
print(sourceRange.getBeginColumn() + ", " + sourceRange.getLastColumn())
```

6.139.3 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()
```

6.139.4 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)
```

6.139.5 Метод PivotTable.getConditionalLabelFilter

Метод возвращает примененный условный фильтр по подписи.

Вызов

```
PivotTableConditionalLabelFilter getConditionalLabelFilter(fieldName)
```

Параметры

— fieldName: название поля в сводной таблице, тип string.

Возвращает

— условный фильтр по подписи, тип [PivotTableConditionalLabelFilter](#).

— None, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

Пример

```
labelOperation = myOfficeSDK.PivotTableConditionalLabelFilterOperation()  
labelOperation.operationType =  
myOfficeSDK.PivotTableConditionalLabelFilterOperationType_Contains  
labelOperation.operand = "to"
```

```
labelFilter = pivotTable.getConditionalLabelFilter("Product Name")
labelFilter.setOperation(labelOperation)

tableEditor.setFilter(labelFilter).apply()
```

6.139.6 Метод PivotTable.getConditionalValueFilter

Метод возвращает примененный условный фильтр по значению.

Вызов

```
PivotTableConditionalValueFilter getConditionalValueFilter(fieldName)
```

Параметры

– fieldName: название поля в сводной таблице, тип string.

Возвращает

- условный фильтр по значению, тип [PivotTableConditionalValueFilter](#).
- None, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

Пример

```
valueOperation =
myOfficeSDK.PivotTableConditionalValueFilter.getDefaultOperation(myOfficeSDK.PivotTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

valueFilter = pivotTable.getConditionalValueFilter("Product Name")
valueFilter.setOperation(valueOperation)

tableEditor.setFilter(valueFilter).apply()
```

6.139.7 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)
```

6.139.8 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())
```

6.139.9 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())
```

6.139.10 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)
```

6.139.11 Метод `PivotTable.getFilter`

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

Пример

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

6.139.12 Метод `PivotTable.getFilters`

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

Пример

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

6.139.13 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)
```

6.139.14 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())
```

6.139.15 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)
```

6.139.16 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)
```

6.139.17 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)
```

6.139.18 Метод `PivotTable.getSortingParams`

Метод возвращает параметры сортировки, примененные к заданному полю сводной таблицы.

Вызов

```
PivotTableSortingParams getSortingParams(fieldName)
```

Параметры

– `fieldName`: название поля таблицы, тип `string`.

Возвращает

– параметры сортировки, тип [PivotTableSortingParams](#).

– `None`, если параметры сортировки не удалось получить.

Пример

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("L1")
pivotTable = cell.getPivotTable()
sorting = pivotTable.getSortingParams("Product")
print(sorting.sortingType)
print(sorting.sortingOrder)
```

6.139.19 Метод `PivotTable.getSourceRange`

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

Пример

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

6.139.20 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())
```

6.139.21 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)
```

6.139.22 Метод `PivotTable.isColumnGrandTotalEnabled`

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

Пример

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

6.139.23 Метод `PivotTable.isRowGrandTotalEnabled`

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

Пример

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

6.139.24 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()
```

6.139.25 Метод `PivotTable.update`

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

Пример

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

6.140 Класс `PivotTableCaptions`

Класс `PivotTableCaptions` хранит все пользовательские заголовки сводной таблицы. Может быть получен вызовом [PivotTable.getPivotTableCaptions\(\)](#).

Таблица 80 – Описание полей класса `PivotTableCaptions`

Поле	Тип	Описание
<code>PivotTableCaptions.errorCaption</code>	string	Подпись для значений, которые возвращают ошибку
<code>PivotTableCaptions.emptyCaption</code>	string	Подпись для значений, которые возвращают пустое значение
<code>PivotTableCaptions.grandTotalCaption</code>	string	Подпись общих итогов
<code>PivotTableCaptions.valuesHeaderCaption</code>	string	Подпись поля из области значений; это поле отображается в отчете в случае, если в сводной таблице

Поле	Тип	Описание
		находится более двух полей из области значений, и макет имеет тип 'табличный' или 'структурный'
<code>PivotTableCaptions.rowHeaderCaption</code>	string	Подпись заголовка строк (видна только при включенном компактном макете, это макет по умолчанию)
<code>PivotTableCaptions.columnHeaderCaption</code>	string	Подпись заголовка колонок (видна только при включенном компактном макете, это макет по умолчанию)

6.141 Класс PivotTableCategoryField

Класс `PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. таблицу 81). Объект может быть получен посредством вызовов [PivotTable.getRowFields\(\)](#), [PivotTable.getColumnFields\(\)](#).

Таблица 81 – Описание полей `PivotTableCategoryField`

Поле	Тип	Описание
<code>PivotTableCategoryField.fieldProperties</code>	PivotTableFieldProperties	Свойства поля
<code>PivotTableCategoryField.subtotalFunctions</code>	Коллекция объектов PivotTableFunction	Список функций для вычисления подытога

6.142 Класс PivotTableConditionalLabelFilter

Класс `PivotTableConditionalLabelFilter` представляет собой условный фильтр по подписи. Фильтрация производится по строковым значениям, которые содержит поле сводной таблицы. Используется в методах [PivotTable.getConditionalLabelFilter\(\)](#) и [PivotTableEditor.setFilter\(\)](#).

6.142.1 Метод `PivotTableConditionalLabelFilter.getFieldName`

Метод возвращает имя поля, с которым ассоциирован фильтр.

Вызов

```
string getFieldName()
```

Возвращает

– имя поля, тип `string`.

6.142.2 Метод `PivotTableConditionalLabelFilter.getOperation`

Метод возвращает текущие настройки фильтра.

Вызов

```
PivotTableConditionalLabelFilterOperation getOperation()
```

Возвращает

– настройки фильтра, тип [PivotTableConditionalLabelFilterOperation](#).

6.142.3 Метод `PivotTableConditionalLabelFilter.isInDefaultState`

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

Вызов

```
bool isInDefaultState()
```

Возвращает

– `True`, если фильтр не содержит настроек, в ином случае – `False`.

6.142.4 Метод `PivotTableConditionalLabelFilter.reset`

Метод сбрасывает настройки текущего фильтра.

Вызов

```
reset()
```

6.142.5 Метод `PivotTableConditionalLabelFilter.setOperation`

Метод устанавливает настройки фильтра.

Вызов

```
setOperation(operation)
```

Параметры

– operation:	настройки	фильтра,	тип
	<code>PivotTableConditionalLabelFilterOperation.</code>		

Пример

```
labelOperation = myOfficeSDK.PivotTableConditionalLabelFilterOperation()  
labelOperation.operationType =  
myOfficeSDK.PivotTableConditionalLabelFilterOperationType_Contains  
labelOperation.operand = "to"  
  
labelFilter = pivotTable.getConditionalLabelFilter("Product Name")  
labelFilter.setOperation(labelOperation)  
  
tableEditor.setFilter(labelFilter).apply()
```

6.143 Класс PivotTableConditionalLabelFilterOperation

Класс `PivotTableConditionalLabelFilterOperation` представляет собой настройки условного фильтра по подписи. Используется в методах [PivotTableConditionalLabelFilter.getOperation\(\)](#) и [PivotTableConditionalLabelFilter.setOperation\(\)](#).

Таблица 82 – Описание полей класса PivotTableConditionalLabelFilterOperation

Поле	Тип	Описание
PivotTableConditionalLabelFilterOperation.operationType	PivotTableConditionalLabelFilterOperationType	Тип операции сравнения
PivotTableConditionalLabelFilterOperation.operand	string	Значение для сравнения

Пример

```
labelOperation = myOfficeSDK.PivotTableConditionalLabelFilterOperation()  
labelOperation.operationType =  
myOfficeSDK.PivotTableConditionalLabelFilterOperationType_Contains  
labelOperation.operand = "to"  
  
labelFilter = pivotTable.getConditionalLabelFilter("Product Name")  
labelFilter.setOperation(labelOperation)  
  
tableEditor.setFilter(labelFilter).apply()
```

6.144 Перечисление PivotTableConditionalLabelFilterOperationType

Перечисление `PivotTableConditionalLabelFilterOperationType` содержит типы операций сравнения, применяемые к фильтру по подписи. Используется в поле [PivotTableConditionalLabelFilterOperation.operationType](#).

Таблица 83 – Описание типов операций сравнения

Значение	Описание
<code>PivotTableConditionalLabelFilterOperationType_Equal</code>	Равные
<code>PivotTableConditionalLabelFilterOperationType_NotEqual</code>	Не равные
<code>PivotTableConditionalLabelFilterOperationType_BeginsWith</code>	Начинаются с
<code>PivotTableConditionalLabelFilterOperationType_NotBeginsWith</code>	Не начинается с
<code>PivotTableConditionalLabelFilterOperationType_EndsWith</code>	Заканчиваются на
<code>PivotTableConditionalLabelFilterOperationType_NotEndsWith</code>	Не заканчиваются на
<code>PivotTableConditionalLabelFilterOperationType_Contains</code>	Содержат
<code>PivotTableConditionalLabelFilterOperationType_NotContains</code>	Не содержат

6.145 Класс PivotTableConditionalValueFilter

Класс `PivotTableConditionalValueFilter` представляет собой условный фильтр по значению. Фильтрация производится по значениям, которые соответствуют полю в строке или столбце сводной таблицы. Используется в методах [PivotTable.getConditionalValueFilter\(\)](#) и [PivotTableEditor.setFilter\(\)](#).

6.145.1 Метод PivotTableConditionalValueFilter.getDefaultOperation

Метод возвращает настройки условного фильтра соответствующие заданной операции сравнения.

Вызов

```
static PivotTableConditionalValueFilterOperation
getDefaultOperation(operationType)
```

Параметры

– operationType: тип операции сравнения, тип [PivotTableConditionalValueFilterOperationType](#).

Возвращает

– настройки условного фильтра по значению, тип [PivotTableConditionalValueFilterOperation](#).

Пример

```
valueOperation =
myOfficeSDK.PivotTableConditionalValueFilter.getDefaultOperation(myOfficeSDK.Pivo
tTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

valueFilter = pivotTable.getConditionalValueFilter("Product Name")
valueFilter.setOperation(valueOperation)

tableEditor.setFilter(valueFilter).apply()
```

6.145.2 Метод PivotTableConditionalValueFilter.getExpectedOperandType

Метод возвращает тип данных, который должен использоваться с заданной операцией сравнения.

Вызов

```
static PivotTableConditionalValueFilterOperandType
getExpectedOperandType(operationType)
```

Параметры

– operationType: тип операции сравнения, тип [PivotTableConditionalValueFilterOperationType](#).

Возвращает

– тип данных, тип [PivotTableConditionalValueFilterOperandType](#).

6.145.3 Метод `PivotTableConditionalValueFilter.getFieldName`

Метод возвращает имя поля, с которым ассоциирован фильтр.

Вызов

```
string getFieldName()
```

Возвращает

– имя поля, тип `string`.

6.145.4 Метод `PivotTableConditionalValueFilter.getOperation`

Метод возвращает текущие настройки фильтра.

Вызов

```
PivotTableConditionalValueFilterOperation getOperation()
```

Возвращает

– настройки фильтра, тип [PivotTableConditionalValueFilterOperation](#).

6.145.5 Метод `PivotTableConditionalValueFilter.isInDefaultState`

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

Вызов

```
bool isInDefaultState()
```

Возвращает

– `True`, если фильтр не содержит настроек, в ином случае – `False`.

6.145.6 Метод `PivotTableConditionalValueFilter.reset`

Метод сбрасывает настройки текущего фильтра.

Вызов

```
reset()
```

6.145.7 Метод `PivotTableConditionalValueFilter.setOperation`

Метод устанавливает настройки фильтра.

Вызов

```
setOperation(operation)
```

Параметры

– operation:	настройки	фильтра,	тип
	PivotTableConditionalValueFilterOperation.		

Пример

```
valueOperation =
myOfficeSDK.PivotTableConditionalValueFilter.getDefaultOperation(myOfficeSDK.PivotTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

valueFilter = pivotTable.getConditionalValueFilter("Product Name")
valueFilter.setOperation(valueOperation)

tableEditor.setFilter(valueFilter).apply()
```

6.146 Перечисление PivotTableConditionalValueFilterOperandType

Перечисление `PivotTableConditionalValueFilterOperandType` содержит типы данных, с которыми могут работать различные операции сравнения. Используется в методе `PivotTableConditionalValueFilter.getExpectedOperandType()`.

Таблица 84 – Описание типов данных

Значение	Описание
PivotTableConditionalValueFilterOperandType_PositiveInt	Положительное целое число
PivotTableConditionalValueFilterOperandType_Percent	Число с плавающей запятой в диапазоне от 0 до 100
PivotTableConditionalValueFilterOperandType_Double	Дробное число
PivotTableConditionalValueFilterOperandType_NonNegativeDouble	Положительное дробное число

6.147 Класс PivotTableConditionalValueFilterOperation

Класс `PivotTableConditionalValueFilterOperation` представляет собой настройки условного фильтра по значению. Используется в методах [`PivotTableConditionalValueFilter.getDefaultOperation\(\)`](#), [`PivotTableConditionalValueFilter.getOperation\(\)`](#) и [`PivotTableConditionalValueFilter.setOperation\(\)`](#).

Таблица 85 – Описание полей класса PivotTableConditionalValueFilterOperation

Поле	Тип	Описание
PivotTableConditionalValueFilterOperation.operationType	PivotTableConditionalValueFilterOperationType	Тип операции сравнения
PivotTableConditionalValueFilterOperation.valueFieldIndex	int	Индекс поля, к которому применяется фильтр
PivotTableConditionalValueFilterOperation.operand1	string	Первое значение для сравнения
PivotTableConditionalValueFilterOperation.operand2	string	Второе значение для операции сравнения Between

Пример

```

valueOperation =
myOfficeSDK.PivotTableConditionalValueFilter.getDefaultOperation(myOfficeSDK.PivotTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

valueFilter = pivotTable.getConditionalValueFilter("Product Name")
valueFilter.setOperation(valueOperation)

tableEditor.setFilter(valueFilter).apply()

```

6.148 Перечисление PivotTableConditionalValueFilterOperationType

Перечисление PivotTableConditionalValueFilterOperationType содержит типы операций сравнения, применяемые к фильтру по значению. Используется в поле [PivotTableConditionalValueFilterOperation.operationType](#).

Таблица 86 – Описание типов операций сравнения

Значение	Описание
PivotTableConditionalValueFilterOperationType_Equal	Равные
PivotTableConditionalValueFilterOperationType_NotEqual	Не равные
PivotTableConditionalValueFilterOperationType_Greater	Больше
PivotTableConditionalValueFilterOperationType_GreaterOrEqual	Больше или равные
PivotTableConditionalValueFilterOperationType_Less	Меньше
PivotTableConditionalValueFilterOperationType_LessOrEqual	Меньше или равные
PivotTableConditionalValueFilterOperationType_Between	Между
PivotTableConditionalValueFilterOperationType_TopPercent	Не поддерживается
PivotTableConditionalValueFilterOperationType_BottomPercent	Не поддерживается
PivotTableConditionalValueFilterOperationType_TopSum	Не поддерживается
PivotTableConditionalValueFilterOperationType_BottomSum	Не поддерживается
PivotTableConditionalValueFilterOperationType_TopValues	Не поддерживается
PivotTableConditionalValueFilterOperationType_BottomValues	Не поддерживается

6.149 Класс PivotTableEditor

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

6.149.1 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()
```

6.149.2 Метод `PivotTableEditor.apply`

Метод обновляет сводную таблицу с заданными свойствами и возвращает результат [PivotTableUpdateResult](#).

Пример

```
pivotTableEditor = pivotTable.createPivotTableEditor()  
if myOfficeSDK.PivotTableUpdateResult_Success == pivotTableEditor.apply():  
    print("Successfully applied")
```

6.149.3 Метод `PivotTableEditor.disableField`

Метод удаляет поле из всех областей. Параметр `fieldName` - имя поля (тип - строка). Метод возвращает объект [PivotTableEditor](#).

Пример

```
pivotTableEditor = pivotTable.createPivotTableEditor()  
pivotTableEditor = pivotTableEditor.disableField("Age")  
pivotTableEditor.apply()
```

6.149.4 Метод `PivotTableEditor.enableField`

Метод добавляет поле в область, зависящую от типа поля. Параметр `fieldName` - имя поля. Метод возвращает объект [PivotTableEditor](#).

Пример

```
pivotTableEditor = pivotTable.createPivotTableEditor()  
pivotTableEditor = pivotTableEditor.enableField("Age")  
pivotTableEditor.apply()
```

6.149.5 Метод `PivotTableEditor.moveField`

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

Пример

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

6.149.6 Метод `PivotTableEditor.removeField`

Метод удаляет поле из категории. Параметры: `fieldName` - имя поля, `fromCategory` - область, из которой удаляется поле (тип - [PivotTableFieldCategory](#)). Метод возвращает объект [PivotTableEditor](#).

Пример

```
pivotTableEditor = pivotTable.createPivotTableEditor()  
pivotTableEditor = pivotTableEditor.removeField("BB",  
myOfficeSDK.PivotTableFieldCategory_Values)  
pivotTableEditor.apply()
```

6.149.7 Метод `PivotTableEditor.reorderField`

Метод изменяет позицию поля в пределах категории. Параметры: `fieldName` - имя поля, `category` - область (тип - [PivotTableFieldCategory](#)), `toIndex` - новая позиция поля. Метод возвращает объект [PivotTableEditor](#).

Пример

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

6.149.8 Метод `PivotTableEditor.resetAllFilters`

Метод сбрасывает все фильтры, примененные к сводной таблице, и обновляет её.

Вызов

```
PivotTableEditor resetAllFilters()
```

Возвращает

– текущий редактор сводных таблиц, тип [PivotTableEditor](#).

6.149.9 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()
```

6.149.10 Метод `PivotTableEditor.setFilter`

Метод задает фильтр [PivotTableFilter](#), [PivotTableConditionalLabelFilter](#) или [PivotTableConditionalValueFilter](#) сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`. Метод возвращает объект [PivotTableEditor](#).

Перегрузки

```
PivotTableEditor setFilter(PivotTableFilter filter)
```

```
PivotTableEditor setFilter(PivotTableConditionalLabelFilter filter)
```

```
PivotTableEditor setFilter(PivotTableConditionalValueFilter filter)
```

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()
```

6.149.11 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)
```

6.149.12 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 = pivotTableEditor.setGrandTotalSettings(True, True)  
pivotTableEditor.apply()
```

6.149.13 Метод `PivotTableEditor.setLayoutSettings`

Метод устанавливает настройки отображения [PivotTableLayoutSettings](#) сводной таблицы, возвращает объект [PivotTableEditor](#).

Пример

```
pivotTableLayoutSettings = pivotTable.getPivotTableLayoutSettings()  
pivotTableLayoutSettings.reportLayout =  
myOfficeSDK.PivotTableReportLayout_Tabular  
  
pivotTableEditor = pivotTable.createPivotTableEditor()  
pivotTableEditor = pivotTableEditor.setLayoutSettings(pivotTableLayoutSettings)  
pivotTableEditor.apply()
```

6.149.14 Метод `PivotTableEditor.setSortingByLabel`

Метод позволяет отсортировать данные в сводной таблице по названию строк и столбцов.

Вызов

```
PivotTableEditor setSortingByLabel(fieldName, order)
```

Параметры

- `fieldName`: название поля для сортировки, тип `string`.
- `order`: порядок сортировки, тип [PivotTableSortingOrder](#).

Возвращает

- текущий редактор сводных таблиц, тип [PivotTableEditor](#).

Пример

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("L1")
pivotTable = cell.getPivotTable()
editor = pivotTable.createPivotTableEditor()
editor.setSortingByLabel("Product", myOfficeSDK.PivotTableSortingOrder_Ascending)
editor.apply()
```

6.149.15 Метод `PivotTableEditor.setSortingByValue`

Метод позволяет отсортировать данные в сводной таблице по значениям и итогам.

Вызов

```
PivotTableEditor setSortingByValue(fieldName, order, valueFieldName,
sortingByValueSlice, sortingFieldSubtotal, valueSliceSubtotal)
```

Параметры

- `fieldName`: название поля для сортировки, тип `string`.
- `order`: порядок сортировки, тип [PivotTableSortingOrder](#).
- `valueFieldName`: название поля значений, которое содержит данные для сортировки, тип `string`.
- `sortingByValueSlice`: (необязательный) срез данных для сортировки по значениям, содержит название строки или столбца с данными или путь до него во вложенной структуре, тип `VectorString`.

- `sortingFieldSubtotal`: (необязательный) функция для сортировки по подытогам, применяется к заданному в параметре `fieldName` полю, тип [PivotTableFunction](#).
- `valueSliceSubtotal`: (необязательный) функция для сортировки по подытогам, применяется к заданному в параметре `sortByValueSlice` полю, тип [PivotTableFunction](#).

Возвращает

- текущий редактор сводных таблиц, тип [PivotTableEditor](#).

Пример сортировки по общим итогам

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("L1")
pivotTable = cell.getPivotTable()
editor = pivotTable.createPivotTableEditor()
editor.setSortingByValue("Manager", myOfficeSDK.PivotTableSortingOrder_Ascending,
"Sum of Total")
editor.apply()
```

Пример сортировки по значениям во вложенном столбце "Russia > Furniture"

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("L1")
pivotTable = cell.getPivotTable()
editor = pivotTable.createPivotTableEditor()
editor.setSortingByValue("Manager",
myOfficeSDK.PivotTableSortingOrder_Descending, "Sum of Total",
myOfficeSDK.VectorString(["Russia", "Furniture"]))
editor.apply()
```

Пример сортировки по промежуточным итогам в столбце "Russia"

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("L1")
pivotTable = cell.getPivotTable()
editor = pivotTable.createPivotTableEditor()
editor.setSortingByValue("Manager",
myOfficeSDK.PivotTableSortingOrder_Descending, "Sum of Total",
myOfficeSDK.VectorString(["Russia"]), None, myOfficeSDK.PivotTableFunction_Auto)
editor.apply()
```

6.149.16 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)
```

6.150 Класс `PivotTableField`

Класс `PivotTableField` содержит свойства полей сводной таблицы (см. таблицу 87). Объект может быть получен посредством вызова [PivotTable.getFieldsList\(\)](#).

Таблица 87 – Описание полей класса `PivotTableField`

Поле	Тип	Описание
<code>PivotTableField.fieldProperties</code>	PivotTableFieldProperties	Свойства полей сводной таблицы
<code>PivotTableField.fieldCategories</code>	PivotTableFieldCategories	Категории полей сводной таблицы
<code>PivotTableField.customFormula</code>	string	Вычисляемая формула

6.151 Класс `PivotTableFieldCategories`

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

6.151.1 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())
```

6.152 Перечисление PivotTableFieldCategory

Перечисление PivotTableFieldCategory содержит флаги, которые задают категорию области полей. Пример использования см. в [PivotTable.getFieldCategories\(\)](#).

Таблица 88 – Категории области полей

Значение	Описание
PivotTableFieldCategory_Pages	Область фильтров
PivotTableFieldCategory_Rows	Область строк
PivotTableFieldCategory_Columns	Область колонок
PivotTableFieldCategory_Values	Область значений

6.153 Класс PivotTableFieldProperties

Класс PivotTableFieldProperties содержит свойства поля [PivotTableField](#) сводной таблицы (см. таблицу 89).

Таблица 89 – Описание полей класса PivotTableFieldProperties

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

6.154 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()
```

6.154.1 Метод PivotTableFilter.getCount

Возвращает количество фильтруемых полей.

Пример

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

6.154.2 Метод PivotTableFilter.getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

Пример

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

6.154.3 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))
```

6.154.4 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))
```

6.154.5 Метод `PivotTableFilter.isInDefaultState`

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

Вызов

```
bool isInDefaultState()
```

Возвращает

– `True`, если фильтр не содержит настроек, в ином случае – `False`.

6.154.6 Метод `PivotTableFilter.reset`

Метод сбрасывает настройки текущего фильтра.

Вызов

```
reset()
```

6.154.7 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)
```

6.155 Класс `PivotTableFilters`

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

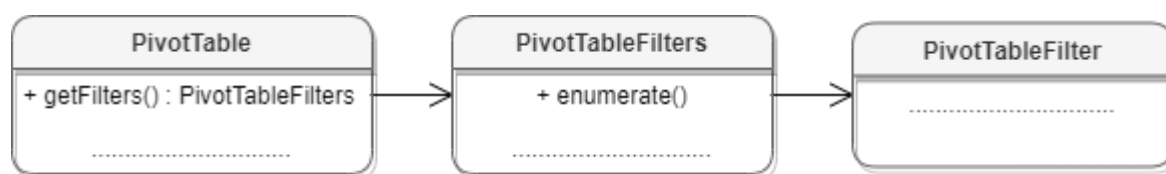


Рисунок 45 – Объектная модель классов для работы с фильтрами

6.155.1 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())
```

6.156 Перечисление PivotTableFunction

Перечисление PivotTableFunction содержит функции, которые могут быть использованы в сводных таблицах. Объект используется в качестве поля subtotalFunctions класса [PivotTableCategoryField](#).

Таблица 90 – Функции сводных таблиц

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

6.157 Класс PivotTableItem

PivotTableItem описывает элемент сводной таблицы (см. Рисунок 46). См. пример в главе [PivotTableItems.getEnumerator](#).

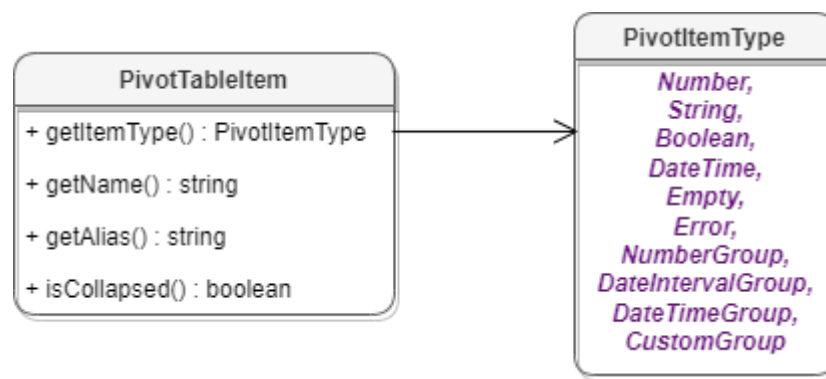


Рисунок 46 – Класс PivotTableItem

6.157.1 Метод PivotTableItem.getAlias

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

6.157.2 Метод PivotTableItem.getItemType

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

6.157.3 Метод PivotTableItem.getName

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

6.157.4 Метод PivotTableItem.isCollapsed

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

6.158 Класс PivotTableItemForCategory

Класс PivotTableItemForCategory представляет собой элемент коллекции [PivotTableSlicePath](#).

Таблица 91 – Описание полей класса PivotTableItemForCategory

Поле	Тип	Описание
PivotTableItemForCategory.fieldName	string	Название поля сводной таблицы
PivotTableItemForCategory.item	PivotTableItem	Элемент сводной таблицы

6.159 Класс PivotTableItems

Класс обеспечивает доступ к списку элементов сводной таблицы (см. Рисунок 47).

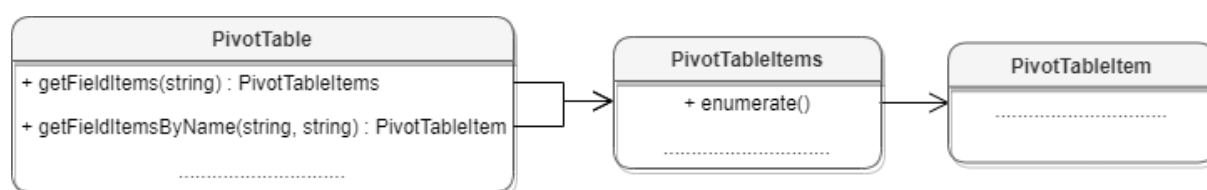


Рисунок 47 – Объектная модель классов для работы с элементами сводных таблиц

6.159.1 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.GetItemTypeInfo())
        print(pivotTableItem.isCollapsed())
    
```

6.160 Перечисление PivotTableItemType

Перечисление PivotTableItemType содержит возможные типы элементов сводной таблицы.

Таблица 92 – Типы элементов сводной таблицы

Значение	Описание
PivotTableItemType_Number	Числовой
PivotTableItemType_String	Строковый
PivotTableItemType_Boolean	Логический
PivotTableItemType_DateTime	Дата / время
PivotTableItemType_Empty	Пустой тип
PivotTableItemType_Error	Ошибка
PivotTableItemType_NumberGroup	Интервальная группировка
PivotTableItemType_DateIntervalGroup	Интервальная группировка по датам
PivotTableItemType_DateTimeGroup	Группировка по дате / времени
PivotTableItemType_CustomGroup	Пользовательская (произвольная) группировка

Пример

```
sheet = document.getBlocks().getTable("Лист1")
cell = sheet.getCell("A1")
pivotTable = cell.getPivotTable()
if pivotTable != None:
    pivotTableItems = pivotTable.getFieldItems("Age")
    pivotTableItemsEnumerator = pivotTableItems.getEnumerator()
    for pivotTableItem in pivotTableItemsEnumerator:
        print(pivotTableItem.getType())
```

6.161 Класс PivotTableLayoutSettings

Класс PivotTableLayoutSettings содержит настройки отображения сводной таблицы. Данный объект может быть получен в результате вызова [PivotTable.getPivotTableLayoutSettings\(\)](#) и установлен методом [PivotTableEditor.setLayoutSettings\(\)](#).

Таблица 93 – Описание полей класса PivotTableLayoutSettings

Поле	Тип	Описание
PivotTableLayoutSettings. reportLayout	PivotTableReportLayout	Настройка вида макета сводной таблицы (компактный, табличный, структурный).
PivotTableLayoutSettings. valueFieldsOrientation	ValueFieldsOrientation	Настраивает положение значений в случае, если в сводной таблице более двух полей значений.
PivotTableLayoutSettings. pageFieldOrder	PageFieldOrder	Настройка порядка полей фильтров (вниз, затем поперек или сначала поперек, потом вниз)
PivotTableLayoutSettings. indentForCompactLayout	float	Размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей)
PivotTableLayoutSettings. pageFieldWrapCount	int	Настройка связана с pageFieldOrder, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д)
PivotTableLayoutSettings. isMergeAndCenterLabelsEnabled	bool	Настройка позволяет объединить ячейки заголовков

Поле	Тип	Описание
PivotTableLayoutSettings. useGridDropZones	bool	Флаг, отвечающий за отображение классического вида (как в Excel 2003). Влияет только на расположение полей в отчете
PivotTableLayoutSettings. displayFieldCaptions	bool	Флаг, отвечающий за отображение заголовков полей

6.162 Класс PivotTablePageField

Содержит свойства поля из области фильтров (см. таблицу 94). Объект может быть получен посредством вызова [PivotTable.getPageFields\(\)](#).

Таблица 94 – Описание полей класса PivotTablePageField

Поле	Тип	Описание
PivotTablePageField. fieldProperties	PivotTableFieldProperties	Свойства поля

6.163 Перечисление PivotTableReportLayout

Перечисление PivotTableReportLayout описывает внешний вид отчетов сводной таблицы. Используется в качестве поля reportLayout класса [PivotTableLayoutSettings](#).

Таблица 95 – Варианты внешнего вида отчетов

Значение	Описание
PivotTableReportLayout_Compact	Компактный вид
PivotTableReportLayout_Tabular	Табличный вид
PivotTableReportLayout_Outline	Структурный вид

6.164 Класс PivotTableSlicePath

Класс `PivotTableSlicePath` представляет собой путь до столбца или строки в сводной таблице. Данный класс является коллекцией объектов [PivotTableItemForCategory](#). Используется в поле [PivotTableSortingParams.sortByValueSlice](#).

6.165 Класс PivotTablesManager

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

Пример

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

6.165.1 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)
```

6.166 Перечисление PivotTableSortingOrder

Перечисление `PivotTableSortingOrder` позволяет задать порядок сортировки данных в сводной таблице. Используется в методах [PivotTableEditor.setSortingByLabel\(\)](#) и [PivotTableEditor.setSortingByValue\(\)](#) и в поле [PivotTableSortingParams.sortingOrder](#).

Таблица 96 – Варианты порядка сортировки данных в сводной таблице

Значение	Описание
PivotTableSortingOrder_Ascending	Сортировка по возрастанию
PivotTableSortingOrder_Descending	Сортировка по убыванию

6.167 Класс PivotTableSortingParams

Класс PivotTableSortingParams содержит настройки сортировки данных в сводной таблице. Данный класс используется в методе [PivotTable.getSortingParams\(\)](#).

Таблица 97 – Описание полей класса PivotTableSortingParams

Поле	Тип	Описание
PivotTableSortingParams.sortByValueSlice	PivotTableSlicePath	Срез данных для сортировки по значениям, который содержит путь до столбца или строки с данными
PivotTableSortingParams.sortFieldSubtotal	PivotTableFunction	Функция для сортировки по подытогам, которая применена к текущему полю
PivotTableSortingParams.sortingOrder	PivotTableSortingOrder	Порядок сортировки
PivotTableSortingParams.sortingType	PivotTableSortingType	Тип сортировки
PivotTableSortingParams.valueFieldNameForSorting	string	Название поля значений, которое содержит данные для сортировки
PivotTableSortingParams.valueSliceSubtotal	PivotTableFunction	Функция для сортировки по подытогам, которая применена к sortByValueSlice полю

6.168 Перечисление PivotTableSortingType

Перечисление PivotTableSortingType содержит типы сортировки данных в сводной таблице. Используется в поле [PivotTableSortingParams.sortingType](#).

Таблица 98 – Типы сортировки данных в сводной таблице

Значение	Описание
PivotTableSortingType_Manual	Ручная сортировка
PivotTableSortingType_Label	Сортировка по названию строк или столбцов
PivotTableSortingType_Value	Сортировка по значениям и итогам

6.169 Перечисление PivotTableUpdateResult

В таблице 99 приведены значения, которые соответствуют возможным результатам обновления сводной таблицы (см. методы [PivotTable.update\(\)](#), [PivotTableEditor.apply\(\)](#)).

Таблица 99 – Результаты обновления сводной таблицы

Значение	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

Значение	Описание
PivotTableUpdateResult_EmptyDataSourceHeaders	Пустые заголовки исходных данных
PivotTableUpdateResult_NoReferenceUnderDefine	Попытка обновить или создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу
PivotTableUpdateResult_NoSuchItem	Элемент не найден
PivotTableUpdateResult_CannotExpandCollapseLeafItem	Не удастся раскрыть свернутый элемент
PivotTableUpdateResult_AnotherPivotInsideDataSource	Найдена другая сводная таблица в этом же диапазоне
PivotTableUpdateResult_Canceled	Обновление сводной таблицы отменено
PivotTableUpdateResult_NoSuchSheetInExternalDocument	Не найден лист во внешнем документе
PivotTableUpdateResult_InvalidSortingParams	Неправильно заданы настройки сортировки

6.170 Класс PivotTableValueField

PivotTableValueField содержит свойства поля сводной таблицы, использующегося как значение столбец (см. таблицу 100). Объект класса может быть получен посредством вызова [PivotTable.GetValueFields\(\)](#).

Таблица 100 – Описание полей класса PivotTableValueField

Поле	Тип	Описание
PivotTableValueField.baseFieldName	string	Оригинальное поле на основе которого было создано данное поле.
PivotTableValueField.valueFieldName	string	Автоматический уникальный псевдоним такой как “Sum of %имя поля%”.
PivotTableValueField.cellNumberFormat	CellFormat	Числовой формат для конкретного поля значений.
PivotTableValueField.totalFunction	PivotTableFunction	Агрегирующая функция поля значений (SUM, COUNT, MAX и т.д.).
PivotTableValueField.customFormula	string	Вычисляемая формула для поля значений.

6.171 Класс PointU

Класс PointU представляет точку двухмерном пространстве.

Таблица 101 – Описание полей класса PointU

Поле	Тип	Описание
PointU.x	float	Координата точки по оси x
PointU.y	float	Координата точки по оси y

Пример

```
point = myOfficeSDK.PointU(50, 50)
print("x=", point.x, ", height=", point.y) # x= 50.0 , height= 50.0
```

6.171.1 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)
```

6.172 Класс Position

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

6.172.1 Метод Position.compare

Метод сравнивает заданную позицию с текущей.

Вызов

```
int compare(other)
```

Параметры

– other: позиция для сравнения с текущей, тип [Position](#).

Возвращает

- положительное расстояние, если текущая позиция больше заданной.
- отрицательное расстояние, если текущая позиция меньше заданной.

- 0, если позиции равны.
- None, если позиции нельзя сравнить (позиции в разных документах, в тексте и в колонтитуле и т.д.).

Примеры для текстового документа

```
table1 = document.getRange().getBegin().insertTable(2, 2, "table1")
positionDoc = document.getRange().getBegin()
positionTable = table1.getRange().getBegin()
positionCell = table1.getCell(myOfficeSDK.CellPosition(0,
0)).getRange().getBegin()

if positionDoc.compare(positionTable) + positionTable.compare(positionCell) == 0:
    # True
```

```
document.getRange().getBegin().insertText("Some text")
positionBegin = document.getRange().getBegin()
positionEnd = document.getRange().getEnd()

print(positionBegin.compare(positionEnd)) # -10
print(positionEnd.compare(positionBegin)) # 10
```

Пример для табличного документа

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

firstCellEnd = sheet.getCell("A1").getRange().getEnd()
secondCellBegin = sheet.getCell("B1").getRange().getBegin()

print(firstCellEnd.compare(secondCellBegin)) # 0
```

6.172.2 Метод Position.getBefore

Метод возвращает позицию в текстовом документе, которая находится перед заданной таблицей.

Вызов

```
static Position getBefore(table)
```

Параметры

- table: таблица в текстовом документе, тип [Table](#).

Возвращает

- позиция перед таблицей, тип [Position](#).

Пример

```
table = document.getRange().getBegin().insertTable(3, 3, "table1")
document.getRange().getBegin().insertText("Text in the first cell")
position = myOfficeSDK.Position.getBefore(table)
position.insertSectionBreak(myOfficeSDK.SectionBreakType_Continuous)
document.getRange().getBegin().insertText("Text before the table")
```

6.172.3 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()
```

6.172.4 Метод Position.getCurrentRange

Метод возвращает диапазон, в котором находится текущая позиция. Размер диапазона зависит от выбранной единицы текста.

Вызов

```
Range getCurrentRange(textUnit)
```

Параметры

— textUnit: единица текста, определяющая размер диапазона, тип [TextUnit](#).

Возвращает

— диапазон документа, содержащий текущую позицию, тип [Range](#).

Пример для текстового документа

```
docBegin = document.getRange().getBegin()
sentence = docBegin.getCurrentRange(myOfficeSDK.TextUnit_Sentence)
word = docBegin.getCurrentRange(myOfficeSDK.TextUnit_Word)
character = docBegin.getCurrentRange(myOfficeSDK.TextUnit_Character)

print(sentence.extractText())
# Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incidunt ut labore et dolore magna aliqua.
```

```
print(word.extractText()) # Lorem
print(character.extractText()) # L
```

Пример для табличного документа

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("B2")
cellBegin = cell.getRange().getBegin()
sentence = cellBegin.getCurrentRange(myOfficeSDK.TextUnit_Sentence)

print(sentence.extractText()) # First sentence.
```

6.172.5 Метод `Position.getNextPosition`

Метод возвращает позицию, которая находится на заданном расстоянии после текущей. Если результат выходит за пределы текущего абзаца, метод возвращает конец абзаца.

Вызов

```
Position getNextPosition(count)
```

Параметры

– `count`: (необязательный, по умолчанию 1) расстояние до следующей позиции.

Возвращает

– следующая позиция, тип [Position](#).

Пример для текстового документа

```
document.getRange().getBegin().insertText("New text")
document.getRange().getBegin().getNextPosition(2).insertText("!")
print(document.getRange().extractText()) # Ne!w text
```

Пример для табличного документа

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("B2")
cell.setText("New text")
cell.getRange().getBegin().getNextPosition(2).insertText("!")

print(cell.getFormattedValue()) # Ne!w text
```

6.172.6 Метод `Position.getNextRange`

Метод возвращает диапазон, который расположен после текущего диапазона. Размеры текущего и следующего диапазонов зависят от выбранной единицы текста. Если выбранная единица текста отлична от `Paragraph`, то работа метода `Position.getNextRange` ограничена текущим абзацем.

Вызов

```
Range getNextRange(textUnit)
```

Параметры

– `textUnit`: единица текста, определяющая размеры диапазонов, тип [TextUnit](#).

Возвращает

– следующий диапазон документа, тип [Range](#).

Пример для текстового документа

```
docBegin = document.getRange().getBegin()
nextSentence = docBegin.getNextRange(myOfficeSDK.TextUnit_Sentence)
nextWord =
docBegin.getNextRange(myOfficeSDK.TextUnit_Word).getBegin().getNextRange(myOffice
SDK.TextUnit_Word)
nextCharacter = docBegin.getNextRange(myOfficeSDK.TextUnit_Character)

print(nextSentence.extractText())
# Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
print(nextWord.extractText()) # ipsum
print(nextCharacter.extractText()) # o
```

Пример для табличного документа

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("B2")
cellBegin = cell.getRange().getBegin()
sentence = cellBegin.getNextRange(myOfficeSDK.TextUnit_Sentence)

print(sentence.extractText()) # Second sentence.
```

6.172.7 Метод `Position.getParagraph`

Метод возвращает абзац ([Paragraph](#)) для текущей позиции.

Пример для текстового документа

```
position = document.getRange().getBegin()  
paragraphAtPosition = position.getParagraph()
```

Пример для табличного документа

```
sheet = document.getBlocks().getTable(0)  
cell = sheet.getCell("B2")  
position = cell.getRange().getBegin()  
paragraphAtPosition = position.getParagraph()
```

6.172.8 Метод `Position.getPreviousPosition`

Метод возвращает позицию, которая находится на заданном расстоянии перед текущей. Если результат выходит за пределы текущего абзаца, метод возвращает начало абзаца.

Вызов

```
Position getPreviousPosition(count)
```

Параметры

– `count`: (необязательный, по умолчанию 1) расстояние до предыдущей позиции.

Возвращает

– предыдущая позиция, тип [Position](#).

Пример для текстового документа

```
document.getRange().getBegin().insertText("New text")  
document.getRange().getContentEnd().getPreviousPosition(2).insertText("!")  
print(document.getRange().extractText()) # New te!xt
```

Пример для табличного документа

```
sheet = document.getBlocks().getTable(0)  
cell = sheet.getCell("B2")  
cell.setText("New text")  
cell.getRange().getContentEnd().getPreviousPosition(2).insertText("!")  
  
print(cell.getFormattedValue()) # New te!xt
```

6.172.9 Метод `Position.getPreviousRange`

Метод возвращает диапазон, который расположен перед текущим диапазоном. Размеры текущего и предыдущего диапазонов зависят от выбранной единицы текста. Если выбранная единица текста отлична от `Paragraph`, то работа метода `Position.getPreviousRange` ограничена текущим абзацем.

Вызов

```
Range getPreviousRange(textUnit)
```

Параметры

– `textUnit`: единица текста, определяющая размеры диапазонов, тип [TextUnit](#).

Возвращает

– предыдущий диапазон документа, тип [Range](#).

Пример для текстового документа

```
sentenceBegin =
document.getRange().getBegin().getNextRange(myOfficeSDK.TextUnit_Sentence).getBegin()
previousSentence = sentenceBegin.getPreviousRange(myOfficeSDK.TextUnit_Sentence)

previousWord = sentenceBegin.getPreviousRange(myOfficeSDK.TextUnit_Word)
while previousWord.getContentEnd().compare(previousWord.getBegin()) == 1:
    previousWord =
previousWord.getBegin().getPreviousRange(myOfficeSDK.TextUnit_Word)

previousCharacter =
sentenceBegin.getPreviousRange(myOfficeSDK.TextUnit_Character)
previousCharacter =
previousCharacter.getBegin().getPreviousRange(myOfficeSDK.TextUnit_Character)
previousCharacter =
previousCharacter.getBegin().getPreviousRange(myOfficeSDK.TextUnit_Character)

print(previousSentence.extractText())
# Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
# incididunt ut labore et dolore magna aliqua.
print(previousWord.extractText()) # aliqua
print(previousCharacter.extractText()) # a
```

Пример для табличного документа

```
sheet = document.getBlocks().getTable(0)
cell = sheet.getCell("B2")
```

```
cellEnd = cell.getRange().getContentEnd()  
sentence = cellEnd.getPreviousRange(myOfficeSDK.TextUnit_Sentence)  
  
print(sentence.extractText()) # Second sentence.
```

6.172.10 Метод **Position.insertBookmark**

Вставляет закладку с наименованием в текущую позицию.

Пример

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

6.172.11 Метод **Position.insertEndnote**

Метод вставляет концевую сноску в текущую позицию.

Вызов

```
Position insertEndnote()
```

Возвращает

— позиция содержимого концевой сноски, тип [Position](#).

Пример

```
range = document.getRange()  
sentence = range.getBegin().getCurrentRange(myOfficeSDK.TextUnit_Sentence)  
pos = sentence.getContentEnd().insertEndnote()  
pos.insertText("Endnote")
```

6.172.12 Метод **Position.insertFootnote**

Метод вставляет сноску в текущую позицию.

Вызов

```
Position insertFootnote()
```

Возвращает

— позиция содержимого сноски, тип [Position](#).

Пример

```
range = document.getRange()  
sentence = range.getBegin().getCurrentRange(myOfficeSDK.TextUnit_Sentence)
```

```
pos = sentence.getContentEnd().insertFootnote()  
pos.insertText("Footnote")
```

6.172.13 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")
```

6.172.14 Метод `Position.insertImage`

Вставляет изображение из файла в текущую позицию. Возвращает объект [Image](#), содержащий вставленное изображение.

Параметры

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

Пример

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

6.172.15 Метод `Position.insertLineBreak`

Метод предназначен для вставки перевода строки.

Пример

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

6.172.16 Метод `Position.insertPageBreak`

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

Пример

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

6.172.17 Метод `Position.insertSectionBreak`

Вставляет разрыв раздела в текущую позицию. В качестве параметра выступает тип [SectionBreakType](#). При вызове без параметра используется значение `SectionBreakType_NextPage`.

Пример

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

```
endPosition.insertSectionBreak(myOfficeSDK.SectionBreakType_EvenPage)
```

6.172.18 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")
```

6.172.19 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")
```

6.172.20 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)
```

6.172.21 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)
```

6.172.22 `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")
```

6.172.23 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")
```

6.173 Перечисление `PredefinedTextStyle`

Перечисление `PredefinedTextStyle` содержит стандартные стили абзаца. Используется в методах [TextStyles.create\(\)](#) и [TextStyles.get\(\)](#).

Таблица 102 – Описание стандартных стилей абзацев

Значение	Описание
<code>PredefinedTextStyle_Normal</code>	Обычный
<code>PredefinedTextStyle_Heading1</code>	Заголовок 1
<code>PredefinedTextStyle_Heading2</code>	Заголовок 2
<code>PredefinedTextStyle_Heading3</code>	Заголовок 3
<code>PredefinedTextStyle_Heading4</code>	Заголовок 4
<code>PredefinedTextStyle_Heading5</code>	Заголовок 5
<code>PredefinedTextStyle_Heading6</code>	Заголовок 6
<code>PredefinedTextStyle_Heading7</code>	Заголовок 7
<code>PredefinedTextStyle_Heading8</code>	Заголовок 8

Значение	Описание
PredefinedTextStyle_Heading9	Заголовок 9
PredefinedTextStyle_Title	Название документа
PredefinedTextStyle_Subtitle	Подзаголовок
PredefinedTextStyle_HeaderFooter	Колонтитул
PredefinedTextStyle_Footnote	Сноска
PredefinedTextStyle_Endnote	Концевая сноска
PredefinedTextStyle_Contents1	Содержание 1
PredefinedTextStyle_Contents2	Содержание 2
PredefinedTextStyle_Contents3	Содержание 3
PredefinedTextStyle_Contents4	Содержание 4
PredefinedTextStyle_Contents5	Содержание 5
PredefinedTextStyle_Contents6	Содержание 6
PredefinedTextStyle_Contents7	Содержание 7
PredefinedTextStyle_Contents8	Содержание 8
PredefinedTextStyle_Contents9	Содержание 9
PredefinedTextStyle_Hyperlink	Ссылка

6.174 PresentationExportSettings

Класс `PresentationExportSettings` предоставляет настройки, необходимые для экспорта презентационных документов (см. [Document.exportAs](#)).

Таблица 103 – Описание полей класса `PresentationExportSettings`

Поле	Тип	Описание
<code>PresentationExportSettings.skipHiddenSlides</code>	<code>bool</code>	Исключает из созданного документа скрытые слайды

Пример

```
presentationExportSettings = myOfficeSDK.PresentationExportSettings()  
presentationExportSettings.skipHiddenSlides = False  
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1,  
presentationExportSettings)
```

6.175 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)
```

6.175.1 Метод **PrintingScope.getCellRange**

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

6.175.2 Метод **PrintingScope.usePrintArea**

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

6.176 Перечисление `PrintingScope.Type`

Варианты выбора диапазона страниц для экспорта и печати представлены в таблице 104. Используется в [PrintingScope](#).

Таблица 104 – Диапазон страниц для экспорта и печати

Значение	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)
```

6.177 Класс `PrintTitles`

Класс `PrintTitles` представляет собой сквозной заголовок для экспорта документов. Сквозной заголовок – это диапазон ячеек, который будет отображаться на каждой странице экспортируемого листа. Данный класс используется в методах [Table.getPrintTitles\(\)](#) и [Table.setPrintTitles\(\)](#).

Конструкторы

```
PrintTitles()
```

```
PrintTitles(int top, int left, int bottom, int right)
```

Таблица 105 – Описание полей класса `PrintTitles`

Поле	Тип	Описание
<code>PrintTitles.bottomRow</code>	<code>int</code>	Индекс нижней строки диапазона
<code>PrintTitles.leftColumn</code>	<code>int</code>	Индекс левой колонки диапазона
<code>PrintTitles.rightColumn</code>	<code>int</code>	Индекс правой колонки диапазона
<code>PrintTitles.topRow</code>	<code>int</code>	Индекс верхней строки диапазона

Пример

```
sheet = document.getBlocks().getTable(0)
titles = myOfficeSDK.PrintTitles(4, 4, 7, 7)
sheet.setPrintTitles(titles)

document.exportAs("exportedDoc.pdf", myOfficeSDK.ExportFormat_PDFA1)
```

6.177.1 Метод `PrintTitles.create`

Метод создает новый сквозной заголовок с заданными координатами.

Вызов

```
static PrintTitles create(top, left, bottom, right)
```

Параметры

- `top`: индекс верхней строки диапазона.
- `left`: индекс левой колонки диапазона.
- `bottom`: индекс нижней строки диапазона.
- `right`: индекс правой колонки диапазона.

Возвращает

- сквозной заголовок, тип [PrintTitles](#).

Пример

```
titles = myOfficeSDK.PrintTitles.create(4, 4, 7, 7)
print(titles.isRowsCols()) # True
```

6.177.2 Метод `PrintTitles.createPrintTitlesCols`

Метод создает новый сквозной заголовок состоящий из целых колонок.

Вызов

```
static PrintTitles createPrintTitlesCols(first, last)
```

Параметры

- `first`: индекс первой колонки диапазона.
- `last`: индекс последней колонки диапазона.

Возвращает

- сквозной заголовок, тип [PrintTitles](#).

Пример

```
titles = myOfficeSDK.PrintTitles.createPrintTitlesCols(0, 2)
print(titles.isCols()) # True
```

6.177.3 Метод `PrintTitles.createPrintTitlesRows`

Метод создает новый сквозной заголовок состоящий из целых строк.

Вызов

```
static PrintTitles createPrintTitlesRows(first, last)
```

Параметры

- `first`: индекс первой строки диапазона.
- `last`: индекс последней строки диапазона.

Возвращает

- сквозной заголовок, тип [PrintTitles](#).

Пример

```
titles = myOfficeSDK.PrintTitles.createPrintTitlesRows(0, 0)
print(titles.isRows()) # True
```

6.177.4 Метод `PrintTitles.isCols`

Метод позволяет определить состоит ли сквозной заголовок только из целых колонок.

Вызов

```
bool isCols()
```

Возвращает

- `True`, если сквозной заголовок состоит из целых колонок, в ином случае – `False`.

6.177.5 Метод `PrintTitles.isRows`

Метод позволяет определить состоит ли сквозной заголовок только из целых строк.

Вызов

```
bool isRows()
```

Возвращает

- `True`, если сквозной заголовок состоит из целых строк, в ином случае – `False`.

6.177.6 Метод `PrintTitles.isRowsCols`

Метод позволяет определить состоит ли сквозной заголовок только из целых строк или столбцов.

Вызов

```
bool isRowsCols()
```

Возвращает

- True, если сквозной заголовок не состоит из целых строк или колонок, в ином случае
- False.

6.178 Класс Range

Класс Range предоставляет доступ к диапазону документа. На рисунке 48 изображена объектная модель классов, относящихся к работе с диапазонами.

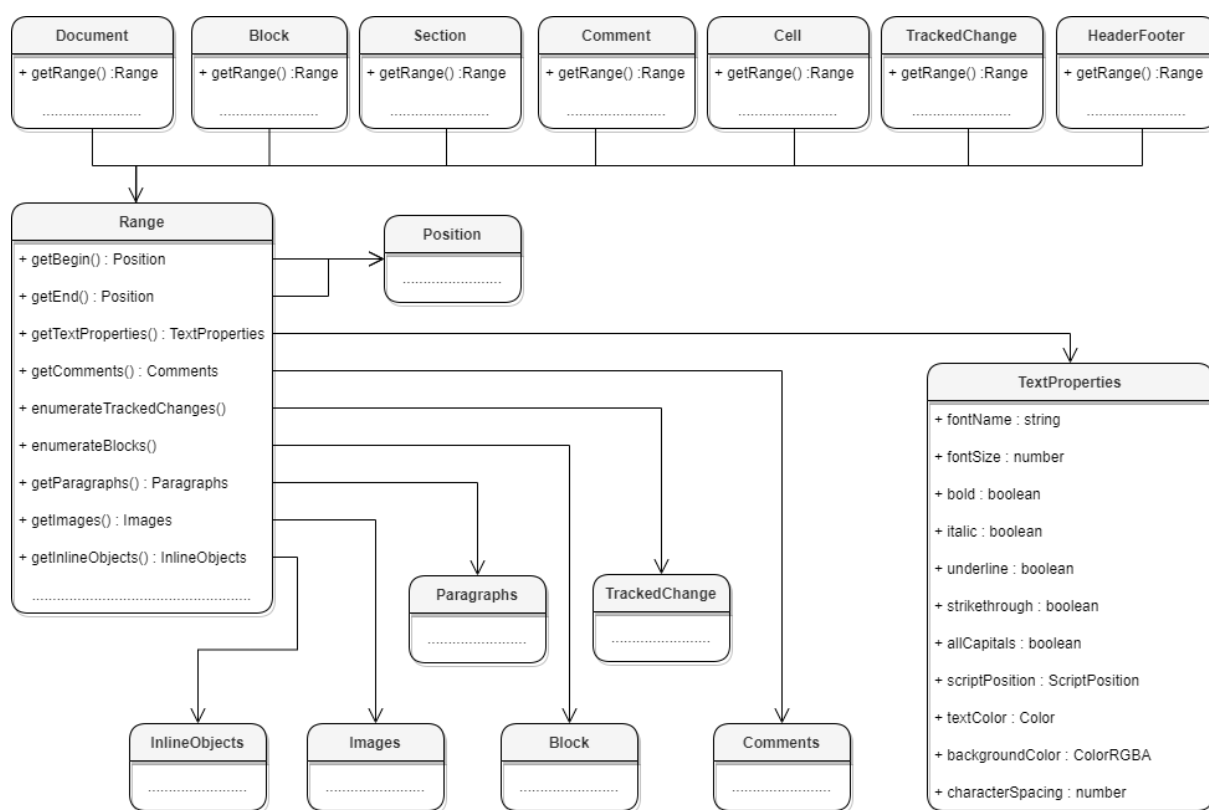


Рисунок 48 – Объектная модель для работы с классом Range

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())
```

6.178.1 Конструктор Range

Для создания объекта [Range](#) вы можете использовать следующий конструктор:

```
documentRange = myOfficeSDK.Range(begin, end)
```

Параметры

— begin: начальная позиция диапазона, тип [Position](#);

— end: конечная позиция диапазона, тип [Position](#).

6.178.2 Метод `Range.extractText`

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и другие объекты игнорируются.

Пример для текстового документа

```
docRange = document.getRange()  
print(docRange.ExtractText())
```

Пример для табличного документа

```
table = document.getBlocks().getTable(0)  
if table != None:  
    cell = table.getCell("B2")  
    cellRange = cell.getRange()  
    print(cellRange.ExtractText())
```

6.178.3 Метод `Range.getBegin`

Метод возвращает позицию в начале диапазона.

Пример для текстового документа

```
beginDocPosition = document.getRange().getBegin()  
beginDocPosition.insertText("API")
```

Пример для табличного документа

```
table = document.getBlocks().getTable(0)  
if table != None:  
    cell = table.getCell("B2")  
    cellRange = cell.getRange()  
    beginDocPos = cellRange.getBegin()  
    beginDocPos.insertText("API")
```

6.178.4 Метод `Range.getBlocksEnumerator`

Предоставляет возможность итерации по блокам.

Пример для текстового документа

```
docRange = document.getRange()
blocksEnumerator = docRange.getBlocksEnumerator()
for block in blocksEnumerator:
    print(block.getRange().extractText())
```

Пример для табличного документа

```
table = document.getBlocks().getTable(0)
if table != None:
    cell = table.getCell("B2")
    cellRange = cell.getRange()
    blocksEnumerator = cellRange.getBlocksEnumerator()
    for block in blocksEnumerator:
        print(block.getRange().extractText())
```

6.178.5 Метод `Range.getComments`

Обеспечивает доступ к комментариям в диапазоне.

Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам. Если дат нет, то порядок комментариев не определен.

Пример

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

6.178.6 Метод `Range.getContentEnd`

Метод возвращает позицию в конце содержимого текущего диапазона.

Вызов

```
Position getContentEnd()
```

Возвращает

— позиция конца диапазона, тип [Position](#).

Метод `Range.getContentEnd` может использоваться для добавления информации в конец содержимого абзаца/ячейки. Чтобы получить позицию конца диапазона с переходом к следующему абзацу/ячейке, вызовите метод [Range.getEnd](#).

Пример для текстового документа

```
document.getRange().getBegin().insertText("Text")
document.getRange().getContentEnd().insertText("#")
print(document.getRange().extractText()) # Text#\n
```

Пример для табличного документа

```
cell = document.getBlocks().getTable(0).getCell("B2")
cell.setText("Text")
cellEnd = cell.getRange().getContentEnd()
cellEnd.insertText("#")
print(cell.getRawValue()) # Text#
```

6.178.7 Метод `Range.getEnd`

Метод возвращает позицию в конце диапазона, включая символ перехода на новую строку/ячейку.

Метод `Range.getEnd` может использоваться для перехода к следующему абзацу/ячейке. Чтобы получить позицию конца диапазона для добавления в него информации, вызовите метод [Range.getContentEnd](#).

Пример для текстового документа

```
document.getRange().getBegin().insertText("First Paragraph")
endDocPosition = document.getRange().getEnd()
endDocPosition.insertText("Second Paragraph")
print(document.getRange().extractText()) # First Paragraph\nSecond Paragraph\n
```

Пример для табличного документа

```
sheet = document.getBlocks().getTable(0)
sheet.getCell("B2").setText("Text")
cellEnd = sheet.getCell("B2").getRange().getEnd()
cellEnd.insertText("#")
print(sheet.getCell("B2").getRowValue()) # Text
print(sheet.getCell("C2").getRowValue()) # #
```

6.178.8 Метод `Range.getFootnotesEndnotes`

Метод возвращает коллекцию сносок и концевых сносок в текущем диапазоне.

Вызов

```
FootnotesEndnotes getFootnotesEndnotes()
```

Возвращает

— коллекция сносок и концевых сносок, тип [FootnotesEndnotes](#).

Пример

```
range = document.getRange()  
notes = range.getFootnotesEndnotes()  
for note in notes:  
    print(note.getRange().extractText())
```

6.178.9 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())
```

6.178.10 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())
```

6.178.11 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())
```

6.178.12 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)
```

6.178.13 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())
```

6.178.14 Метод `Range.isContentLocked`

Метод возвращает значение `True`, если изменения содержимого диапазона запрещены (см. [Защита диапазона текстового документа](#)).

Пример для текстового документа

```
docRange = document.getRange()  
print(docRange.isContentLocked())
```

Пример для таблицы внутри текстового документа

```
table = document.getBlocks().getTable(0)  
if table != None:  
    cell = table.getCell("B2")  
    cellRange = cell.getRange()  
    print(cellRange.isContentLocked())
```

6.178.15 Метод `Range.lockContent`

Метод запрещает изменения содержимого диапазона (см. [Защита диапазона текстового документа](#)).



Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа

```
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())
```

6.178.16 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())
```

6.178.17 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")
```

6.178.18 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")
```

6.178.19 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)
```

6.178.20 Метод `Range.unlockContent`

Метод разрешает изменения содержимого диапазона (см. [Защита диапазона текстового документа](#)).



Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа

```
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())
```

6.178.21 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")
```

6.178.22 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")
```

6.179 Класс RangeBorders

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

6.180 Класс RectU

Класс RectU описывает прямоугольник в двухмерном пространстве.

Таблица 106 – Описание полей класса RectU

Поле	Тип	Описание
RectU.topLeft	PointU	Координата левой верхней вершины прямоугольника
RectU.bottomRight	PointU	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
```

6.180.1 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)]
```

6.181 Класс SaveDocumentSettings

Класс `SaveDocumentSettings` предоставляет настройки, используемые для сохранения документа в файл, см. [Document.saveAs\(\)](#).

Таблица 107 – Описание полей класса `SaveDocumentSettings`

Поле	Тип	Описание
<code>SaveDocumentSettings.documentFormat</code>	DocumentFormat	Формат документа
<code>SaveDocumentSettings.documentType</code>	DocumentType	Тип документа
<code>SaveDocumentSettings.documentPassword</code>	string	Пароль для защиты электронного документа от несанкционированного доступа
<code>SaveDocumentSettings.isTemplate</code>	bool	Флаг, обозначающий, что документ должен быть сохранен как шаблон
<code>SaveDocumentSettings.dsvSettings</code>	DSVSettings	Настройки для сохранения документа в формате DSV

6.182 Перечисление ScaleFrom

В таблице 108 представлены позиции якоря при масштабировании объекта `AbsoluteFrame`. Используется в [AbsoluteFrame.scale\(\)](#).

Таблица 108 – Варианты для якоря масштабирования

Значение	Описание
<code>ScaleFrom_BottomRight</code>	Правый нижний угол
<code>ScaleFrom_BottomLeft</code>	Левый нижний угол
<code>ScaleFrom_TopLeft</code>	Левый верхний угол
<code>ScaleFrom_TopRight</code>	Правый верхний угол

6.183 Класс ScientificCellFormatting

Класс содержит параметры для экспоненциального формата ячеек таблицы. Используется в качестве аргумента метода [Cell.setFormat\(\)](#).

Таблица 109 – Описание полей класса ScientificCellFormatting

Поле	Тип	Описание
ScientificCellFormatting.decimalPlaces	int	Количество десятичных позиций
ScientificCellFormatting.minExponentDigits	int	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())
```

6.184 Класс Script

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

6.184.1 Метод Script.getBody

Метод возвращает текст макрокоманды в виде строки.

Пример

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

6.184.2 Метод Script.getName

Метод возвращает имя макрокоманды.

Пример

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

6.184.3 Метод Script.setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

Пример

```
scripts = document.getScripts()

scriptName = "Enumerate scripts for document"
scriptBody = "local scripts = document:getScripts()\nfor script in\nscripts: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() + "'")
```

6.184.4 Метод Script.setName

Метод устанавливает имя для макрокоманды.

Пример

```
scripts = document.getScripts()

scriptName = "Enumerate scripts for document"
scriptCode = "local scripts = document:getScripts()"
scripts.setScript(scriptName, scriptCode)

script = scripts.getScript(scriptName)
if script != None:
    newScriptName = "New script name"
```

```
script.setName(newScriptName)
print("Script was renamed to '" + script.getName() + "'")
```

6.185 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)
```

6.185.1 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)
```

6.186 Перечисление ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 110. Используется в качестве поля scriptPosition класса [TextProperties](#).

Таблица 110 – Типы надстрочного и подстрочного форматирования

Значение	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)
```

6.187 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())
```

6.187.1 Метод `Scripts.getEnumerator`

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

Пример

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

6.187.2 Метод `Scripts.getScript`

Метод возвращает объект класса [Script](#), описывающий макрокоманду. В качестве аргумента используется имя макрокоманды.

Пример

```
scripts = document.getScripts()  
script = scripts.getScript("ScriptName")  
if script == None:  
    print("Script not found")  
else:  
    print("Script body:" + script.getBody())
```

6.187.3 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")
```

6.187.4 Метод `Scripts.setScript`

Метод добавляет макрокоманду в текущий документ. Если макрокоманда с таким именем уже существует, будет обновлено ее содержимое.

Пример

```
scripts = document.getScripts()

scriptName = "Enumerate scripts for document"
scriptCode = "local scripts = document:getScripts()\nfor script in\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")
```

6.188 Класс `Search`

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

6.188.1 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](#).

6.189 Класс Section

Класс `Section` представляет собой раздел в документе.

6.189.1 Метод `Section.getFooters`

Метод возвращает коллекцию [HeadersFooters](#) нижних колонтитулов раздела.

Пример

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

6.189.2 Метод `Section.getHeaders`

Метод возвращает коллекцию [HeadersFooters](#) верхних колонтитулов раздела.

Пример

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

6.189.3 Метод `Section.getPageOrientation`

Метод возвращает ориентацию страниц раздела.

Пример

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

6.189.4 Метод `Section.getPageProperties`

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

Пример

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

6.189.5 Метод `Section.getRange`

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

Пример

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

6.189.6 Метод `Section.setPageOrientation`

Метод задает ориентацию страниц раздела.

Пример

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

6.189.7 Метод `Section.setPageProperties`

Метод устанавливает параметры [PageProperties](#) страниц, находящихся в разделе.

Пример

```
for section in sectionsEnumerator:
    section.setPageProperties(myOfficeSDK.PageProperties(100, 50))
    pageProperties = section.getPageProperties()
    print(pageProperties.height)
```

6.190 Перечисление `SectionBreakType`

Перечисление `SectionBreakType` содержит варианты разрыва страниц, используется в [Position.InsertSectionBreak\(\)](#).

Таблица 111 – Варианты разрыва страниц

Значение	Описание
<code>SectionBreakType_NextPage</code>	Следующий раздел начинается с новой страницы
<code>SectionBreakType_Continuous</code>	Следующий раздел продолжается на текущей странице без вставки разрыва страницы

Значение	Описание
SectionBreakType_EvenPage	Следующий раздел начинается на ближайшей четной странице
SectionBreakType_OddPage	Следующий раздел начинается на ближайшей нечетной странице

6.191 Класс Sections

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

6.191.1 Метод `Sections.getEnumerator`

Метод позволяет перечислить коллекцию секций документа.

Пример

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

6.192 Класс Shape

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

6.192.1 Метод `Shape.getShapeProperties`

Метод возвращает свойства фигуры [ShapeProperties](#).

Пример

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

6.192.2 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)
```

6.193 Класс `ShapeProperties`

Класс `ShapeProperties` описывает свойства фигуры. Используется в методах [Shape.getShapeProperties\(\)](#) и [Shape.setShapeProperties\(\)](#).

Таблица 112 – Описание полей класса `ShapeProperties`

Поле	Тип	Описание
<code>ShapeProperties.verticalAlignment</code>	VerticalAlignment	Вертикальное выравнивание
<code>ShapeProperties.borderProperties</code>	LineProperties	Свойства границ фигуры
<code>ShapeProperties.fill</code>	Fill	Свойства заполнения фигуры
<code>ShapeProperties.shapeTextLayout</code>	ShapeTextLayout	Свойства текста внутри фигуры

Пример

```
ShapeProperties shapeProperties = shape.getShapeProperties();
shapeProperties.verticalAlignment = ...
shapeProperties.borderProperties = ...
shapeProperties.fill = ...
shapeProperties.shapeTextLayout = ...
shape.setShapeProperties(shapeProperties);
```

6.194 Перечисление ShapeTextLayout

Перечисление ShapeTextLayout позволяет задать свойства текста, находящегося внутри фигуры. Используется в качестве поля shapeTextLayout класса [ShapeProperties](#).

Таблица 113 – Свойства текста внутри фигуры

Значение	Описание
ShapeTextLayout_DoNotAutoFit	Размещение текста в фигуре по умолчанию
ShapeTextLayout_FitShapeExtentToText	Расширение фигуры под текст
ShapeTextLayout_FitTextToShape	Заполнение фигуры текстом

6.195 Класс SizeU

Класс SizeU представляет размер объекта в двухмерном пространстве.

Таблица 114 – Описание полей классе SizeU

Поле	Тип	Описание
SizeU.width	float	Ширина объекта
SizeU.height	float	Высота объекта

Пример

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

6.195.1 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)
```

6.196 Класс `SortingConditions`

Представляет собой коллекцию условий для сортировки строк. Класс `SortingConditions` используется в методе [CellRange.sort\(\)](#).

Конструктор по умолчанию:

```
SortingConditions()
```

Конструктор копирования:

```
SortingConditions(SortingConditions other)
```

Порядок условий в коллекции определяет порядок применения сортировки. Например, можно задать дополнительное условие для другого столбца, чтобы отсортировать строки с одинаковыми значениями в первом столбце.

Пример

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

conditions = myOfficeSDK.SortingConditions()
conditions.add(0, myOfficeSDK.SortingDirection_Descending)
conditions.add(1, myOfficeSDK.SortingDirection_Ascending)

range.sort(conditions)
```

6.196.1 Метод `SortingConditions.add`

Метод добавляет заданное условие сортировки в коллекцию.

Вызов

```
add(index, sortingDirection)
```

Параметры

- `index`: индекс столбца диапазона для применения сортировки.
- `sortingDirection`: порядок сортировки, тип [SortingDirection](#).

Пример

```
conditions = myOfficeSDK.SortingConditions()
conditions.add(0, myOfficeSDK.SortingDirection_Descending)
conditions.add(1, myOfficeSDK.SortingDirection_Ascending)
```

6.196.2 Метод `SortingConditions.clear`

Метод очищает коллекцию условий.

6.197 Перечисление `SortingDirection`

В таблице 115 представлены варианты порядка сортировки строк. Используется в методе [`SortingConditions.add\(\)`](#).

Таблица 115 – Варианты порядка сортировки строк

Значение	Описание
<code>SortingDirection_Ascending</code>	Сортировка по возрастанию
<code>SortingDirection_Descending</code>	Сортировка по убыванию

Пример

```
conditions = myOfficeSDK.SortingConditions()
conditions.add(0, myOfficeSDK.SortingDirection_Descending)
conditions.add(1, myOfficeSDK.SortingDirection_Ascending)
```

6.198 Класс `Table`

Класс `Table` предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 49).

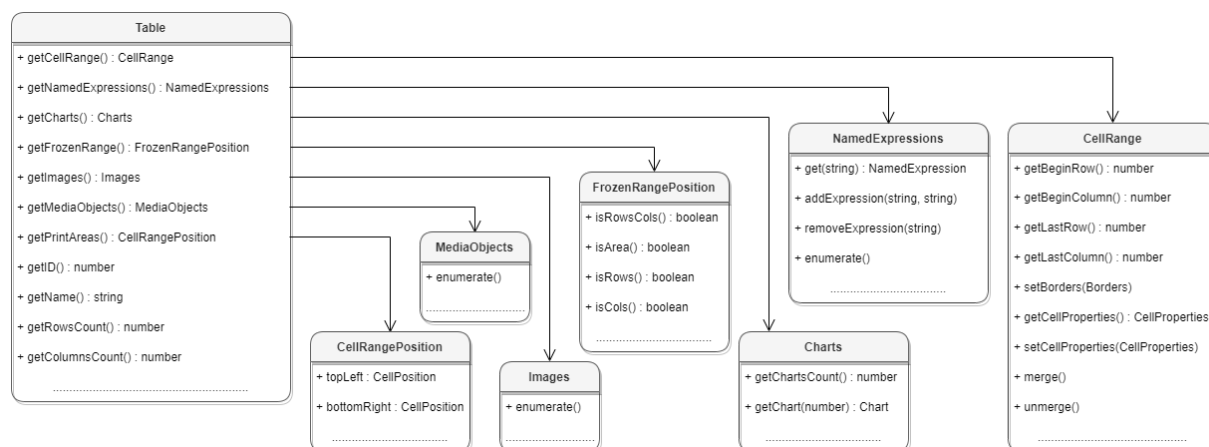


Рисунок 49 – Структура полей класса `Table`

6.198.1 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.

6.198.2 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.

6.198.3 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](#).

6.198.4 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()
```

6.198.5 Метод Table.find

Метод выполняет поиск ячеек, соответствующих заданному запросу.

Вызов

```
CellsIterator find(string, settings)
```

Параметры

- `string`: поисковый запрос, тип `string`.
- `settings`: (необязательный) параметры поиска, тип [TableSearchSettings](#).

Возвращает

- список ячеек, соответствующих поисковому запросу, тип `CellsIterator`.

Пример

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

searchProps = myOfficeSDK.TableSearchSettings()
searchProps.caseSensitive = myOfficeSDK.CaseSensitive_No
searchProps.matchBehaviour = myOfficeSDK.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = myOfficeSDK.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = True

results = sheet.find("*eye", searchProps)
for cell in results:
    print(cell.getFormattedValue()) # Steeleye Stout
```

6.198.6 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())
```

6.198.7 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())
```

6.198.8 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)
```

6.198.9 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())
```

6.198.10 Метод Table.getColumnsCount

Метод позволяет получить количество столбцов таблицы.

Пример

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

6.198.11 Метод `Table.getColumnWidth`

Метод возвращает ширину столбца таблицы в пунктах (1/72 дюйма).

Вызов

```
float getColumnWidth(columnIndex)
```

Параметры

- `columnIndex` – индекс столбца в таблице, для которого возвращается значение ширины. Индексация столбцов начинается с нуля.

Возвращает

- ширина столбца в пунктах (1/72 дюйма).

Пример

```
table = document.getRange().getBegin().insertTable(2, 2, "NewTable")
width = table.getColumnWidth(1)
```

Задать ширину столбца таблицы позволяет метод [Table.setColumnWidth](#).

6.198.12 Метод `Table.getConditionalFormatRules`

Метод позволяет получить коллекцию правил условного форматирования для текущего листа.

Вызов

```
ConditionalFormatTableRules getConditionalFormatRules()
```

Возвращает

- коллекция правил условного форматирования для листа, тип [ConditionalFormatTableRules](#).

Пример

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

topBottomStyle = myOfficeSDK.ConditionalFormatCellStyle()
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
topBottomStyle.cellProperties = cellProperties

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

```
topBottomOperator =  
myOfficeSDK.createTopBottomConditionalFormatOperator(myOfficeSDK.ConditionalForma  
tTopBottomCondition_Top, 20, True)  
  
topBottomRule = myOfficeSDK.ConditionalFormatRule(topBottomOperator,  
topBottomStyle, cellRangePosition, False)  
rules.addRule(topBottomRule)
```

6.198.13 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](#).

6.198.14 Метод Table.getFrozenRange

Существует возможность закрепления диапазона строк и столбцов. Такие диапазоны всегда остаются видимыми на экране в случае, когда пользователь осуществляет навигацию по таблице.

Метод `getFrozenRange` возвращает закрепленный диапазон [FrozenRangePosition](#).

Пример

```
table = document.getBlocks().getTable(0)  
frozenRangePosition = myOfficeSDK.FrozenRangePosition.createFrozenCols(0, 2)  
table.freeze(frozenRangePosition)  
print(table.getFrozenRange().isCols())
```

6.198.15 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())
```

6.198.16 Метод Table.getLastNonEmptyCellInColumn

Возвращает индекс последней заполненной ячейки в столбце. Последней заполненной считается ячейка, после которой все ячейки пустые.

Вызов

```
int getLastNonEmptyCellInColumn(columnIndex)
```

Параметры

– columnIndex: индекс столбца.

Возвращает

- индекс строки, которая содержит последнюю заполненную ячейку.
- None, если все ячейки в столбце пустые.

Пример

```
sheet = document.getBlocks().getTable(0)
print(sheet.getLastNonEmptyCellInColumn(0)) # 10
print(sheet.getLastNonEmptyCellInRow(0)) # 2
```

6.198.17 Метод Table.getLastNonEmptyCellInRow

Возвращает индекс последней заполненной ячейки в строке. Последней заполненной считается ячейка, после которой все ячейки пустые.

Вызов

```
int getLastNonEmptyCellInRow(rowIndex)
```

Параметры

— rowIndex: индекс строки.

Возвращает

— индекс столбца, который содержит последнюю заполненную ячейку.

— None, если все ячейки в строке пустые.

Пример

```
sheet = document.getBlocks().getTable(0)
print(sheet.getLastNonEmptyCellInColumn(0)) # 10
print(sheet.getLastNonEmptyCellInRow(0)) # 2
```

6.198.18 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)
```

6.198.19 Метод Table.getName

Метод позволяет получить наименование листа табличного документа.

Пример

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

6.198.20 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())
```

6.198.21 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())
```

6.198.22 Метод Table.getPrintTitles

Метод возвращает сквозной заголовок для текущего листа.

Вызов

```
PrintTitles getPrintTitles()
```

Возвращает

- СКВОЗНОЙ ЗАГОЛОВОК, тип [PrintTitles](#).
- None, если СКВОЗНОЙ ЗАГОЛОВОК не установлен.

Пример

```
sheet = document.getBlocks().getTable(0)  
printTitles = sheet.getPrintTitles()  
print(printTitles.isCols())  
print(printTitles.isRows())  
print(printTitles.isRowsCols())
```

6.198.23 Метод Table.getProtectionProperties

Метод возвращает параметры защиты от изменений листа табличного документа (см. [Защита листа табличного документа](#)).

Вызов

```
TableProtectionProperties getProtectionProperties()
```

Возвращает

- [TableProtectionProperties](#): свойства защиты листа документа (None, если защита листа не установлена).

6.198.24 Метод `Table.getRowHeight`

Метод возвращает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов

```
float getRowHeight(rowIndex)
```

Параметры

- `rowIndex` – индекс строки в таблице, для которой возвращается значение высоты. Индексация строк начинается с нуля.

Возвращает

- высота строки в пунктах (1/72 дюйма).

Пример

```
table = document.getRange().getBegin().insertTable(2, 2, "NewTable")
height = table.getRowHeight(1)
```

Задать высоту строки таблицы позволяет метод [Table.setRowHeight](#).

6.198.25 Метод `Table.getRowsCount`

Метод позволяет получить количество строк таблицы.

Пример

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

6.198.26 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())
```

6.198.27 Метод `Table.getUsedRange`

Метод возвращает диапазон ячеек, используемый в текущем листе табличного документа. К используемым относятся ячейки, которые содержат:

- данные;
- заметки;
- форматирование;
- объединенные ячейки;
- фильтры;
- сводные таблицы;
- умные таблицы.

Вызов

```
CellRange getUsedRange()
```

Возвращает

– используемый диапазон ячеек, тип [CellRange](#).

Пример

```
sheet = document.getBlocks().getTable(0)
usedRange = sheet.getUsedRange()
```

6.198.28 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.

6.198.29 Метод `Table.groupRows`

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

Вызов

```
groupRows(rowIndex, rowCount)
```

Параметры

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

6.198.30 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)
```

6.198.31 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)
```

6.198.32 Метод Table.insertImage

Метод вставляет изображение из файла в заданные координаты на листе табличного документа.

Вызов

```
Image insertImage(url, rect)
```

Параметры

- url: путь к файлу, тип string.
- rect: координаты для вставки изображения, тип [RectU](#).

Возвращает

- вставленное изображение, тип [Image](#).

Пример

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

6.198.33 Метод Table.insertRowAfter

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

Вызов

```
insertRowAfter(rowIndex, copyRowStyle, rowCount)
```

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`.
- `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.insertRowAfter(0, False, 2)
```

6.198.34 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)
```

6.198.35 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](#).

6.198.36 Метод Table.isProtected

Метод возвращает состояние защиты от изменений листа табличного документа (см. [Защита листа табличного документа](#)).

Вызов

```
bool isProtected()
```

6.198.37 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](#).

6.198.38 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)
```

6.198.39 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)
```

6.198.40 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()
```

6.198.41 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)
```

6.198.42 Метод Table.removeProtection

Метод снимает защиту от изменений с листа табличного документа (см. [Защита листа табличного документа](#)).

Вызов

```
removeProtection(password)
```

Параметры

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

Пример

```
firstSheet = document.getBlocks().getTable(0)
firstSheet.removeProtection("password")
```

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

6.198.43 Метод `Table.removeRow`

Метод предназначен для удаления строки таблицы, начиная с заданного индекса.

Вызов

```
removeRow(rowIndex, rowCount)
```

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)
```

6.198.44 Метод `Table.removeVisibleColumns`

Метод удаляет видимые столбцы таблицы, находящиеся между заданными индексами (включительно).

Вызов

```
removeVisibleColumns(firstIndex, lastIndex)
```

Параметры

- `firstIndex`: индекс первого столбца. Индексация столбцов начинается с нуля.
- `lastIndex`: индекс последнего столбца.

6.198.45 Метод `Table.removeVisibleRows`

Метод удаляет видимые строки таблицы, находящиеся между заданными индексами (включительно).

Вызов

```
removeVisibleRows(firstIndex, lastIndex)
```

Параметры

- `firstIndex`: индекс первой строки. Индексация строк начинается с нуля.
- `lastIndex`: индекс последней строки.

6.198.46 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.

6.198.47 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)
```

Метод [Table.getColumnWidth](#) позволяет получить ширину столбца таблицы.

6.198.48 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)
```

6.198.49 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))
```

6.198.50 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())
```

6.198.51 Метод Table.setPrintTitles

Метод задает сквозной заголовок (диапазон ячеек, который будет отображаться на каждой странице экспортируемого документа) для текущего листа.

Вызов

```
setPrintTitles(range)
```

Параметры

– range: СКВОЗНОЙ ЗАГОЛОВОК, тип [PrintTitles](#).

Пример

```
sheet = document.getBlocks().getTable(0)
titles = myOfficeSDK.PrintTitles(4, 4, 7, 7)
sheet.setPrintTitles(titles)

document.exportAs("exportedDoc.pdf", myOfficeSDK.ExportFormat_PDFA1)
```

6.198.52 Метод Table.setProtection

Метод устанавливает защиту от изменений на лист табличного документа (см. [Защита листа табличного документа](#)).

Вызов

```
setProtection(tableProtectionProperties, password)
```

Параметры

– tableProtectionProperties: параметры защиты листа, тип [TableProtectionProperties](#);

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

Пример

```
tableProps = myOfficeSDK.TableProtectionProperties()
tableProps.deleteColumns = False
tableProps.deleteRows = False
tableProps.filterData = True
tableProps.formatCells = True
tableProps.formatColumns = True
tableProps.formatRows = True
tableProps.insertAndEditObjects = False
tableProps.insertAndEditPivotTables = False
tableProps.insertColumns = False
```

```
tableProps.insertLinks = True
tableProps.insertRows = False
tableProps.selectProtectedCells = True
tableProps.sortData = True

firstSheet = document.getBlocks().getTable(0)
firstSheet.setProtection(tableProps, "password")
```

Метод `setProtectionProperties()` объектов [Cell](#) и [CellRange](#) позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой защиты).

Если метод `setProtection()` применяется к уже защищенному листу, возникает исключение `SpreadsheetProtectionError`.

6.198.53 Метод `Table.setRowHeight`

Метод устанавливает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов

```
setRowHeight(rowIndex, height, heightRule)
```

Параметры

- `rowIndex` – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- `height` – высота строки в пунктах (1/72 дюйма).
- `heightRule` – точность значения (`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)
```

Метод [Table.getRowHeight](#) позволяет получить высоту строки таблицы.

6.198.54 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;
`columnsCount` – количество строк;
`visible` – видимость.

6.198.55 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)
```

6.198.56 Метод `Table.setVisible`

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

Пример

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

6.198.57 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.

6.198.58 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.

6.198.59 `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")
```

6.198.60 `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")
```

6.199 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](#).

6.199.1 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()
```

6.199.2 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)
```

6.199.3 Метод `TableFilters.getAsConditionalFilter`

Метод возвращает фильтр по условию, установленный в заданном столбце.

Вызов

```
ConditionalTableFilter getAsConditionalFilter(column)
```

Параметры

- column: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `int`.

Возвращает

- фильтр по условию, тип [ConditionalTableFilter](#).
- `None`, если фильтр отсутствует или он другого типа.

Пример

```
if filters.getFilterType(0) == myOfficeSDK.ContainingTableFilter_Conditional:  
    conditionalFilter = filters.getAsConditionalFilter(0)
```

6.199.4 Метод `TableFilters.getAsValueFilter`

Метод возвращает фильтр по значению, установленный в заданном столбце.

Вызов

```
ValuesTableFilter getAsValueFilter(column)
```

Параметры

- column: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `int`.

Возвращает

- фильтр по значению, тип [ValuesTableFilter](#).
- `None`, если фильтр отсутствует или он другого типа.

Пример

```
if filters.getFilterType(0) == myOfficeSDK.ContainingTableFilter_Values:  
    valuesFilter = filters.getAsValueFilter(0)
```

6.199.5 Метод `TableFilters.getFilterType`

Метод возвращает тип фильтра, который установлен в заданном столбце.

Вызов

```
ContainingTableFilter getFilterType(column)
```

Параметры

- column: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `int`.

Возвращает

- тип фильтра столбцов таблицы, тип [ContainingTableFilter](#).

Пример

```
if filters.getFilterType(0) == myOfficeSDK.ContainingTableFilter_Values:  
    valuesFilter = filters.getAsValueFilter(0)
```

6.199.6 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](#).

6.200 Класс TableProtectionProperties

Класс TableProtectionProperties предназначен для настройки параметров защиты листа в табличном документе (аналог раздела «Разрешенные действия» в меню «Управление защитой»). Данный класс используется в методах [Table.setProtection\(\)](#) и [Table.getProtectionProperties\(\)](#).

Таблица 116 – Описание полей класса TableProtectionProperties

Поле	Значение по умолчанию	Описание
TableProtectionProperties.deleteColumns	False	Разрешить удалять колонки
TableProtectionProperties.deleteRows	False	Разрешить удалять строки
TableProtectionProperties.filterData	False	Разрешить фильтровать данные
TableProtectionProperties.formatCells	False	Разрешить форматировать ячейки
TableProtectionProperties.formatColumns	False	Разрешить форматировать столбцы
TableProtectionProperties.formatRows	False	Разрешить форматировать строки
TableProtectionProperties.insertAndEditObjects	False	Разрешить вставлять и редактировать объекты: изображения, диаграммы, фигуры и текстовые поля
TableProtectionProperties.insertAndEditPivotTables	False	Разрешить вставлять и редактировать сводные таблицы
TableProtectionProperties.insertColumns	False	Разрешить вставлять столбцы
TableProtectionProperties.insertLinks	False	Разрешить вставлять и редактировать ссылки в ячейках
TableProtectionProperties.insertRows	False	Разрешить вставлять строки
TableProtectionProperties.selectProtectedCells	True	Разрешить выделение защищенных ячеек
TableProtectionProperties.sortData	False	Разрешить сортировать данные

Пример

```
tableProps = myOfficeSDK.TableProtectionProperties()
tableProps.deleteColumns = False
tableProps.deleteRows = False
tableProps.filterData = True
tableProps.formatCells = True
tableProps.formatColumns = True
tableProps.formatRows = True
tableProps.insertAndEditObjects = False
tableProps.insertAndEditPivotTables = False
tableProps.insertColumns = False
tableProps.insertLinks = True
tableProps.insertRows = False
tableProps.selectProtectedCells = True
tableProps.sortData = True

firstSheet = document.getBlocks().getTable(0)
firstSheet.setProtection(tableProps, "password")
```

6.201 Класс TableSearchSettings

Класс TableSearchSettings предназначен для настройки параметров поиска ячеек. Данный класс используется в методах [Table.find\(\)](#) и [CellRange.find\(\)](#).

Таблица 117 – Описание полей класса TableSearchSettings

Поле	Тип	Описание
TableSearchSettings.caseSensitive	CaseSensitive	Позволяет учитывать регистр при поиске
TableSearchSettings.matchBehaviour	TableSearchSettings.MatchBehaviour	Задаёт алгоритм сравнения запроса и значения
TableSearchSettings.searchIn	TableSearchSettings.SearchProperty	Позволяет выбрать значения, по которым производится поиск
TableSearchSettings.wholeWords	bool	Разбивает текст в ячейке на слова, которые считаются отдельными значениями для сравнения

Пример

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

searchProps = myOfficeSDK.TableSearchSettings()
searchProps.caseSensitive = myOfficeSDK.CaseSensitive_No
```

```
searchProps.matchBehaviour = myOfficeSDK.TableSearchSettings.MatchBehaviour_Glob
searchProps.searchIn = myOfficeSDK.TableSearchSettings.SearchProperty_Value
searchProps.wholeWords = True

results = sheet.find("*eye", searchProps)
for cell in results:
    print(cell.getFormattedValue()) # Steeleye Stout
```

6.201.1 Перечисление TableSearchSettings.MatchBehaviour

Перечисление MatchBehaviour содержит алгоритмы сравнения запроса и значения. Используется в поле [TableSearchSettings.matchBehaviour](#).

Таблица 118 – Описание алгоритмов сравнения

Значение	Описание
TableSearchSettings.MatchBehaviour_Partial	Значение частично содержит запрос
TableSearchSettings.MatchBehaviour_Total	Значение полностью соответствует запросу
TableSearchSettings.MatchBehaviour_Glob	Позволяет использовать шаблоны, содержащие символы * и ?, для создания запроса

6.201.2 Перечисление TableSearchSettings.SearchProperty

Перечисление SearchProperty содержит значения, по которым производится поиск в таблице. Используется в поле [TableSearchSettings.searchIn](#).

Таблица 119 – Описание вариантов поиска в таблице

Значение	Описание
TableSearchSettings.SearchProperty_Value	Поиск по значениям ячеек
TableSearchSettings.SearchProperty_Formula	Поиск по тексту формул в ячейках
TableSearchSettings.SearchProperty_Notes	Поиск по тексту заметок

6.202 Класс TextAnchoredPosition

Класс TextAnchoredPosition представляет позицию объекта на странице текстового документа (см. [InlineFrame.setPosition\(\)](#)).

Таблица 120 – Описание полей класса TextAnchoredPosition

Поле	Тип	Описание
TextAnchoredPosition.horizontal	HorizontalTexAnchoredPosition	Позиция по горизонтали
TextAnchoredPosition.vertical	VerticalTextAnchoredPosition	Позиция по вертикали

6.202.1 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.alignment =
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 = 10
secondTextAnchoredPosition.vertical =
myOfficeSDK

```

```
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_PageContent)
secondTextAnchoredPosition.vertical.alignment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondTextAnchoredPosition.vertical.offset = 10

if firstTextAnchoredPosition.__eq__(secondTextAnchoredPosition):
    print("Equals")
```

6.202.2 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.alignment =
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")
```

6.203 Класс TextConditionalFormatOperator

Класс TextConditionalFormatOperator представляет собой оператор, который содержит настройки применения форматирования для правила "Текст". Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
TextConditionalFormatOperator(ConditionalFormatTextCondition condition,
string argument)
```

Пример

Product Name
Laptop
Smartphone
Headphones
TV
Tablet
Gaming console
Monitor
Keyboard
Mouse
Camera
Watch

Рисунок 50 – Пример создания правила для текстовых значений

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

textStyle = myOfficeSDK.ConditionalFormatCellStyle()
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
textStyle.cellProperties = cellProperties

cellRange = sheet.getCellRange("A2:A12")
cellRangePosition = cellRange.getTableRange()
```

```
textOperator =  
myOfficeSDK.createTextConditionalFormatOperator(myOfficeSDK.ConditionalFormatText  
Condition_BeginsWith, "M")  
  
textRule = myOfficeSDK.ConditionalFormatRule(textOperator, textStyle,  
cellRangePosition, False)  
rules.addRule(textRule)
```

6.203.1 Метод `TextConditionalFormatOperator.getArgument`

Метод возвращает аргумент условия.

Вызов

```
string getArgument()
```

Возвращает

– аргумент условия, тип `string`.

6.203.2 Метод `TextConditionalFormatOperator.getCondition`

Метод возвращает условие применения форматирования.

Вызов

```
ConditionalFormatTextCondition getCondition()
```

Возвращает

– условие применения форматирования, тип [ConditionalFormatTextCondition](#).

6.203.3 Метод `TextConditionalFormatOperator.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.204 Класс `TextExportSettings`

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

Таблица 121 – Описание полей класса TextExportSettings

Поле	Тип	Описание
TextExportSettings.pageNumbers	PageNumbers	Настройки страниц для экспорта текстовых документов

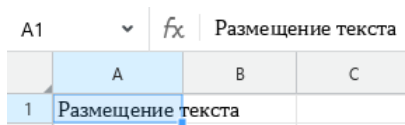
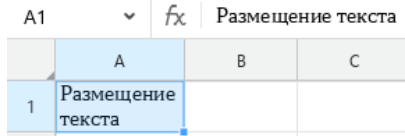
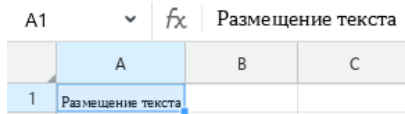
Пример

```
textExportSettings = myOfficeSDK.TextExportSettings()
textExportSettings.pageNumbers =
myOfficeSDK.PageNumbers(myOfficeSDK.PageParity_Even)
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFa1, textExportSettings)
```

6.205 Перечисление TextLayout

В таблице 122 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле textLayout таблицы [CellProperties](#).

Таблица 122 – Варианты размещения текста в ячейках таблицы

Значение	Описание	Отображение
TextLayout_SingleLine	Текст располагается в одну строку с наложением на соседние ячейки.	
TextLayout_WrapByWords	Текст внутри ячейки переносится по словам. Высота ряда увеличивается чтобы разместить текст полностью.	
TextLayout_ShrinkSizeToFitWidth	Текст располагается в одну линию, отображение масштабируется таким образом, чтобы полностью разместиться в ячейке без изменения ее размера. Размер шрифта не изменяется, данная настройка влияет только на отображение содержимого ячейки таблицы.	

Пример

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

```
cellProps = cell.getCellProperties()  
cellProps.textLayout = myOfficeSDK.TextLayout_ShrinkSizeToFitWidth  
cell.setCellProperties(cellProps)
```

6.206 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)
```

6.206.1 Метод TextOrientation.getAngle

Возвращает угол направления текста в ячейке. Значение угла указывается в градусах.

Пример

```
firstSheet = document.getBlocks().getTable("List1")  
cell = firstSheet.getCell("A3")  
cellProps = cell.getCellProperties()  
print(cellProps.textOrientation.isStackedChars())
```

6.206.2 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())
```

6.206.3 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")
```

6.206.4 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")
```

6.207 Класс TextProperties

Класс `TextProperties` содержит поля, задающие параметры текста. На рисунке 51 изображена объектная модель класса `TextProperties`.

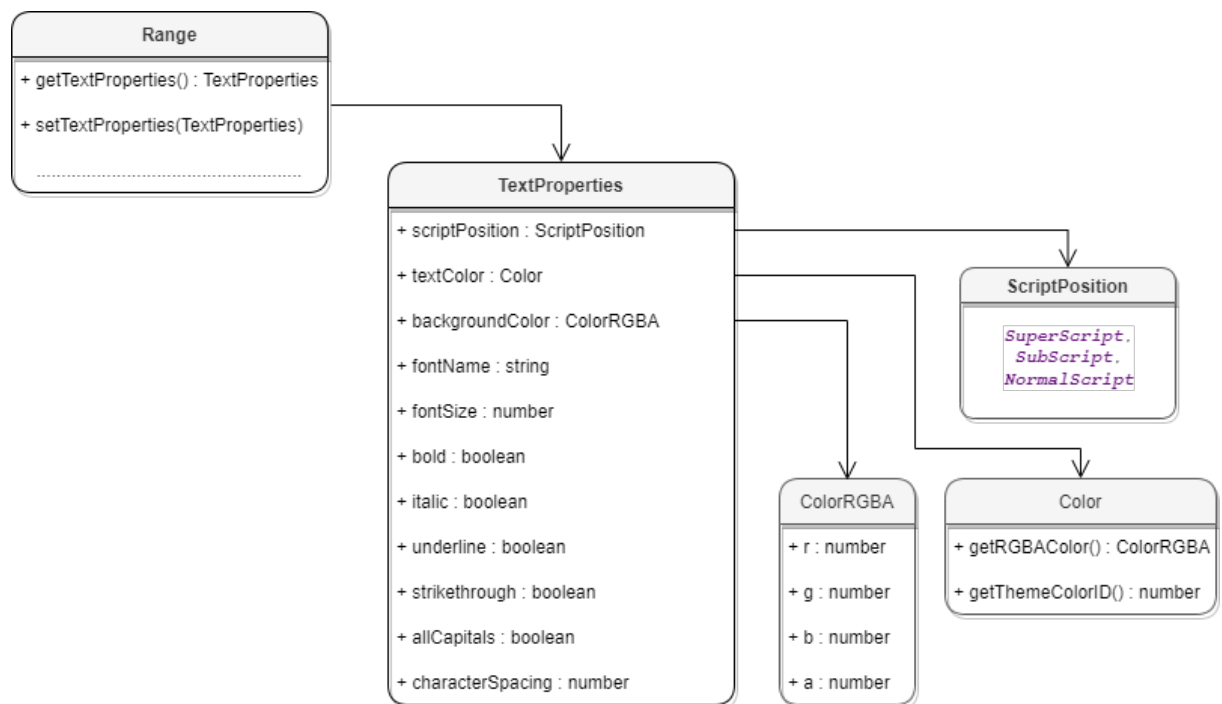


Рисунок 51 – Объектная модель для работы с классом `TextProperties`

Описание полей класса `TextProperties` представлено в таблице 123. Свойства `TextProperties` применяются к диапазону текста `Range` (методы [Range.getTextProperties\(\)](#), [Range.setTextProperties\(\)](#)), ячейке `Cell` (методы [Cell.getTextProperties\(\)](#), [Cell.setTextProperties\(\)](#)) и диапазону ячеек `CellRange` (методы [CellRange.getTextProperties\(\)](#), [CellRange.setTextProperties\(\)](#)).

Таблица 123 – Описание полей класса `TextProperties`

Поле	Тип	Описание
<code>TextProperties.fontName</code>	<code>string</code>	Наименование шрифта, использованного для оформления фрагмента документа.
<code>TextProperties.fontSize</code>	<code>float</code>	Размер шрифта, использованного для оформления фрагмента документа.
<code>TextProperties.bold</code>	<code>bool</code>	Значение <code>True</code> устанавливает жирное начертание для указанного фрагмента текста.
<code>TextProperties.italic</code>	<code>bool</code>	Значение <code>True</code> устанавливает начертание курсивом для указанного фрагмента текста.
<code>TextProperties.underline</code>	<code>bool</code>	Значение <code>True</code> устанавливает подчеркивание для указанного фрагмента текста.
<code>TextProperties.strikethrough</code>	<code>bool</code>	Значение <code>True</code> устанавливает начертание «зачеркнутый» для указанного фрагмента текста.
<code>TextProperties.allCapitals</code>	<code>bool</code>	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>	ScriptPosition	Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме.
<code>TextProperties.textColor</code>	Color	Цвет указанного фрагмента документа.

Поле	Тип	Описание
<code>TextProperties.backgroundColor</code>	ColorRGBA	Цвет фона указанного фрагмента документа.
<code>TextProperties.characterSpacing</code>	<code>float</code>	Размер межсимвольного интервала.

Поля класса `TextProperties` могут быть `None`, если фрагмент документа содержит текст с разными параметрами.

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)
```

6.207.1 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")
```

6.207.2 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")
```

6.208 Класс TextStyle

Класс `TextStyle` представляет собой стиль абзаца. Используется в методах [Paragraph.getStyle\(\)](#), [Paragraph.getResolvedStyle\(\)](#), [Paragraph.setStyle\(\)](#), [TextStyles.create\(\)](#), [TextStyles.get\(\)](#), [TextStyle.getParent\(\)](#), [TextStyle.getNextParagraphStyle\(\)](#) и [TextStyle.setNextParagraphStyle\(\)](#).

6.208.1 Метод TextStyle.createChild

Метод создает новый стиль на основе текущего стиля абзаца.

Вызов

```
createChild(name)
```

Параметры

— name: название нового стиля, тип `string`.

Пример

```
styles = document.getTextStyles()
normalStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Normal)
normalStyle.createChild("myStyle")
myStyle = styles.get("myStyle")
parentStyle = myStyle.getParent()
print(myStyle.getName()) # myStyle
print(parentStyle.getName()) # Normal
```

6.208.2 Метод TextStyle.getName

Метод возвращает название текущего стиля.

Вызов

```
string getName()
```

Возвращает

– название стиля, тип `string`.

6.208.3 Метод `TextStyle.getNextParagraphStyle`

Метод возвращает стиль следующего абзаца.

Вызов

```
TextStyle getNextParagraphStyle()
```

Возвращает

– стиль следующего абзаца, тип [TextStyle](#).

Пример

```
styles = document.getTextStyles()  
titleStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Title)  
print(titleStyle.getNextParagraphStyle().getName()) # Normal
```

6.208.4 Метод `TextStyle.getParagraphProperties`

Метод возвращает настройки форматирования абзаца.

Вызов

```
ParagraphProperties getParagraphProperties()
```

Возвращает

– настройки форматирования абзаца, тип [ParagraphProperties](#).

6.208.5 Метод `TextStyle.getParent`

Метод возвращает стиль, на котором основан текущий стиль абзаца.

Вызов

```
TextStyle getParent()
```

Возвращает

– родительский стиль абзаца, тип [TextStyle](#).

– `None`, если текущий стиль не основан ни на каком стиле.

Пример

```
styles = document.getTextStyles()
normalStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Normal)
normalStyle.createChild("myStyle")
myStyle = styles.get("myStyle")
parentStyle = myStyle.getParent()
print(myStyle.getName()) # myStyle
print(parentStyle.getName()) # Normal
```

6.208.6 Метод `TextStyle.getTextProperties`

Метод возвращает настройки форматирования текста.

Вызов

```
TextProperties getTextProperties()
```

Возвращает

– настройки форматирования текста, тип [TextProperties](#).

6.208.7 Метод `TextStyle.setName`

Метод задает название стиля.

Вызов

```
setName(newName)
```

Параметры

– `newName`: название стиля, тип `string`.

Пример

```
styles = document.getTextStyles()
normalStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Normal)
normalStyle.setName("myStyle")
```

6.208.8 Метод `TextStyle.setNextParagraphStyle`

Метод задает стиль следующего абзаца.

Вызов

```
setNextParagraphStyle(paragraphStyle)
```

Параметры

– paragraphStyle: стиль следующего абзаца, тип [TextStyle](#).

Пример

```
styles = document.getTextStyles()
titleStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Title)
titleStyle.setNextParagraphStyle(styles.get(myOfficeSDK.PredefinedTextStyle_Subtitle))
```

6.208.9 Метод TextStyle.setParagraphProperties

Метод задает настройки форматирования абзаца.

Вызов

```
setParagraphProperties(properties)
```

Параметры

– properties: настройки форматирования абзаца, тип [ParagraphProperties](#).

Пример

```
styles = document.getTextStyles()
normalStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Normal)
props = myOfficeSDK.ParagraphProperties()
props.beforeSpacing = 2.1
normalStyle.setParagraphProperties(props)
```

6.208.10 Метод TextStyle.setTextProperties

Метод задает настройки форматирования текста.

Вызов

```
setTextProperties(textProperties)
```

Параметры

– textProperties: настройки форматирования текста, тип [TextProperties](#).

Пример

```
styles = document.getTextStyles()
normalStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Normal)
props = myOfficeSDK.TextProperties()
props.fontSize = 20
normalStyle.setTextProperties(props)
```

6.209 Класс TextStyles

Класс TextStyles предоставляет доступ к стилям абзацев в документе. Используется в методе [Document.getTextStyles\(\)](#).

6.209.1 Метод TextStyles.create

Метод создает новый стиль абзаца с заданным названием или идентификатором. Если такой стиль уже существует, возникает [IncorrectArgumentError](#).

Вызов

```
TextStyle create(newStyleName)
```

```
TextStyle create(predefinedTextStyle)
```

Параметры

- newStyleName: название нового стиля, тип string.
- predefinedTextStyle: идентификатор нового стиля, тип [PredefinedTextStyle](#).

Возвращает

- созданный стиль абзаца, тип [TextStyle](#).

Пример

```
styles = document.getTextStyles()  
myStyle = styles.create("myStyle")  
myHeadingStyle = styles.create(myOfficeSDK.PredefinedTextStyle_Heading6)
```

6.209.2 Метод TextStyles.get

Метод возвращает стиль абзаца по его названию или идентификатору.

Вызов

```
TextStyle get(styleName)
```

```
TextStyle get(predefinedTextStyle)
```

Параметры

- styleName: название стиля, тип string.
- predefinedTextStyle: идентификатор стиля, тип [PredefinedTextStyle](#).

Возвращает

- стиль абзаца, тип [TextStyle](#).

– None, если стиля с заданным названием или идентификатором не существует.

Пример

```
styles = document.getTextStyles()
normalStyle = styles.get("Normal")
titleStyle = styles.get(myOfficeSDK.PredefinedTextStyle_Title)
print(normalStyle.getName())
print(titleStyle.getName())
```

6.209.3 Метод `TextStyles.getEnumerator`

Метод возвращает перечисление стилей абзацев.

Вызов

```
TextStylesEnumerator getEnumerator()
```

Возвращает

– перечисление стилей абзацев, тип `TextStylesEnumerator`.

Пример

```
styles = document.getTextStyles()
for style in styles.getEnumerator():
    print(style.getName())
```

6.210 Класс `TextToColumnsSettings`

Класс `TextToColumnsSettings` предназначен для настройки параметров разделения текста по ячейкам. Данный класс используется в методе [CellRange.textToColumns\(\)](#).

Таблица 124 – Описание полей класса `TextToColumnsSettings`

Поле	Тип	Описание
<code>TextToColumnsSettings.customDelimiter</code>	string	Позволяет задать пользовательский разделитель
<code>TextToColumnsSettings.textQualifier</code>	TextQualifier	Задаёт символ-ограничитель. Текст между двумя ограничителями записывается в одну ячейку, даже если содержит разделители
<code>TextToColumnsSettings.treatMultipleDelimitersAsOne</code>	bool	Объединять разделители, расположенные подряд
<code>TextToColumnsSettings.useCommaDelimiter</code>	bool	Использовать запятую (,) в качестве разделителя

Поле	Тип	Описание
TextToColumnsSettings.usePercentDelimiter	bool	Использовать процент (%) в качестве разделителя
TextToColumnsSettings.useSemicolonDelimiter	bool	Использовать точку с запятой (;) в качестве разделителя
TextToColumnsSettings.useSlashDelimiter	bool	Использовать косую черту (/) в качестве разделителя
TextToColumnsSettings.useSpaceDelimiter	bool	Использовать пробел в качестве разделителя
TextToColumnsSettings.useTabDelimiter	bool	Использовать знак табуляции в качестве разделителя

Пример

```
settings = myOfficeSDK.TextToColumnsSettings()
settings.useCommaDelimiter = True
settings.useSpaceDelimiter = True
settings.treatMultipleDelimitersAsOne = True
settings.textQualifier =
myOfficeSDK.TextToColumnsSettings.TextQualifier_SingleQuotes
cellRange.textToColumns(settings)
```

6.210.1 Перечисление TextToColumnsSettings.TextQualifier

Перечисление TextQualifier содержит доступные символы-ограничители для горизонтального разделения текста по ячейкам. Текст между двумя ограничителями записывается в одну ячейку, даже если содержит разделители. Используется в поле [TextToColumnsSettings.textQualifier](#).

Таблица 125 – Варианты символов-ограничителей

Значение	Описание
TextToColumnsSettings.TextQualifier_None	Без ограничителей
TextToColumnsSettings.TextQualifier_SingleQuotes	Одинарные кавычки (')
TextToColumnsSettings.TextQualifier_DoubleQuotes	Двойные кавычки (")

Пример

```
settings = myOfficeSDK.TextToColumnsSettings()
settings.useCommaDelimiter = True
```

```
settings.useSpaceDelimiter = True
settings.treatMultipleDelimitersAsOne = True
settings.textQualifier =
myOfficeSDK.TextToColumnsSettings.TextQualifier_SingleQuotes
cellRange.textToColumns(settings)
```

6.211 Перечисление TextUnit

В таблице 126 представлены варианты разделения текста на диапазоны. Используется в методах [Position.getCurrentRange\(\)](#), [Position.getNextRange\(\)](#) и [Position.getPreviousRange\(\)](#).

Таблица 126 – Варианты разделения текста на диапазоны

Значение	Описание
TextUnit_Character	Разделение по символам
TextUnit_Word	Разделение по словам (символы-разделители считаются отдельными словами)
TextUnit_Sentence	Разделение по предложениям (разделители после предложения считаются его частью)
TextUnit_Paragraph	Разделение по абзацам

Пример

```
secondSentence =
document.getRange().getBegin().getNextRange(myOfficeSDK.TextUnit_Sentence)
firstSentenceLastWord =
secondSentence.getBegin().getPreviousRange(myOfficeSDK.TextUnit_Word)
while
firstSentenceLastWord.getContentEnd().compare(firstSentenceLastWord.getBegin())
== 1:
    firstSentenceLastWord =
firstSentenceLastWord.getBegin().getPreviousRange(myOfficeSDK.TextUnit_Word)
thirdSentenceFirstLetter =
secondSentence.getContentEnd().getCurrentRange(myOfficeSDK.TextUnit_Character)

print(secondSentence.extractText())
# Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
print(firstSentenceLastWord.extractText())
```

```
# aliqua
print(thirdSentenceFirstLetter.extractText())
# D
```

6.212 Перечисление TextWrapType

В таблице 127 представлены варианты обтекания текстом встроенного объекта. Используется в [InlineFrame.setWrapType\(\)](#).

Таблица 127 – Варианты обтекания текстом встроенного объекта

Значение	Описание
TextWrapType_Inline	Встроенный объект располагается в тексте
TextWrapType_InFrontOfText	Встроенный объект располагается перед текстом
TextWrapType_BehindText	Встроенный объект располагается за текстом
TextWrapType_TopAndBottom	Текст располагается сверху и снизу от встроенного объекта
TextWrapType_Square	Текст располагается вокруг прямоугольной рамки встроенного объекта
TextWrapType_Through	Текст обтекает встроенный объект по сторонам и внутри

6.213 Перечисление ThemeColorID

В таблице 128 представлены типы идентификаторов цветов тем. Используется в [Color](#).

Таблица 128 – Типы идентификаторов цветов тем

Значение	Описание
ThemeColorID_Background1	Фон1
ThemeColorID_Text1	Текст1
ThemeColorID_Background2	Фон2
ThemeColorID_Text2	Текст2
ThemeColorID_Dark1	Темная1
ThemeColorID_Dark2	Темная2
ThemeColorID_Light1	Светлая1

Значение	Описание
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

6.214 Перечисление TimePatterns

Форматы времени представлены в таблице 129. Пример использования см. в главе [DateTimeCellFormatting](#).

Таблица 129 – Форматы времени

Значение	Описание
TimePatterns_ShortTime	'hh:mm AM/PM' для языка en_US
TimePatterns_LongTime	'hh:mm:ss AM/PM' для языка en_US

6.215 Класс TimeZone

Класс `TimeZone` предоставляет настройки, необходимые для экспорта текстовых документов, см. [DocumentSettings](#).

Таблица 130 – Описание полей класса `TimeZone`

Поле	Тип	Описание
<code>TimeZone.offsetInSecondsToUTC</code>	<code>int</code>	Смещение или разность между временем в данном часовом поясе и временем в формате UTC (Всемирное координированное время) в секундах

6.216 Класс TopBottomConditionalFormatOperator

Класс `TopBottomConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Наибольшие и наименьшие значения". Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
TopBottomConditionalFormatOperator(ConditionalFormatTopBottomCondition
condition, int value, bool usePercent)
```

Пример

Количество
2
5
10
1
3
4
6
15
20
2
7

Рисунок 52 – Пример создания правила для наибольших 20% значений

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

topBottomStyle = myOfficeSDK.ConditionalFormatCellStyle()
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
topBottomStyle.cellProperties = cellProperties

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

topBottomOperator =
myOfficeSDK.createTopBottomConditionalFormatOperator(myOfficeSDK.ConditionalForma
tTopBottomCondition_Top, 20, True)

topBottomRule = myOfficeSDK.ConditionalFormatRule(topBottomOperator,
```

```
topBottomStyle, cellRangePosition, False)  
rules.addRule(topBottomRule)
```

6.216.1 Метод TopBottomConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
ConditionalFormatTopBottomCondition getCondition()
```

Возвращает

– условие применения форматирования, тип [ConditionalFormatTopBottomCondition](#).

6.216.2 Метод TopBottomConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.216.3 Метод TopBottomConditionalFormatOperator.getUsePercent

Метод позволяет определить, выделяется ли определенный процент значений или определенное количество значений.

Вызов

```
bool getUsePercent()
```

Возвращает

– True, если условное форматирование выделяет определенный процент значений; если выделяется определенное количество значений – False.

6.216.4 Метод TopBottomConditionalFormatOperator.getValue

Метод возвращает количество/процент значений для выделения.

Вызов

```
int getValue()
```

Возвращает

– количество/процент значений для выделения, тип `int`.

6.217 Класс `TrackedChange`

Класс `TrackedChange` представляет отслеживаемое изменение в диапазоне документа (см. Рисунок 53).

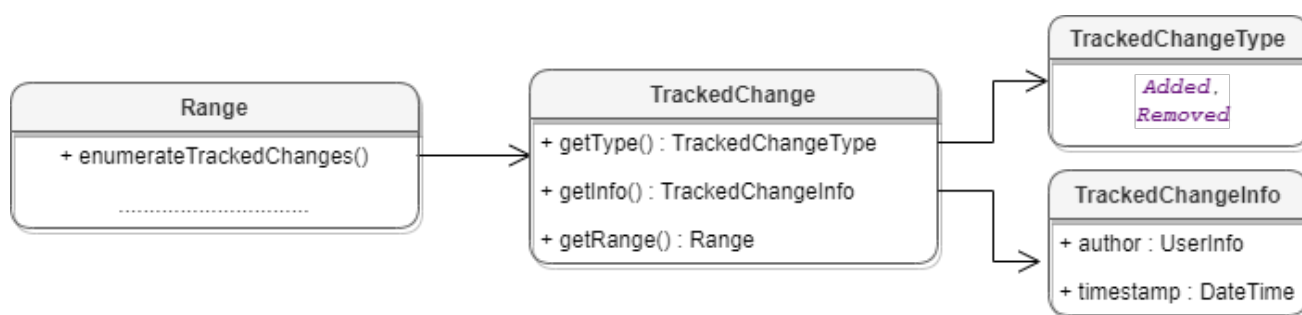


Рисунок 53 – Объектная модель классов для работы с отслеживаемыми изменениями

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

Пример

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

6.217.1 Метод `TrackedChange.getInfo`

Метод позволяет получить информацию об отслеживаемых изменениях [TrackedChangeInfo](#).

Пример

```

trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()
if trackedChangesEnumerator != None:
    for trackedChange in trackedChangesEnumerator:
        trackedChangeInfo = trackedChange.getInfo()
    
```

6.217.2 Метод `TrackedChange.getRange`

Метод возвращает объект [Range](#), который соответствует измененному диапазону внутри абзаца.

Пример

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()  
if trackedChangesEnumerator != None:  
    for trackedChange in trackedChangesEnumerator:  
        trackedChangeRange = trackedChange.getRange()  
        print(trackedChangeRange.extractText())
```

6.217.3 Метод `TrackedChange.getType`

Метод позволяет получить информацию о типе отслеживаемого изменения [TrackedChangeType](#).

Пример

```
trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator()  
if trackedChangesEnumerator != None:  
    for trackedChange in trackedChangesEnumerator:  
        trackedChangeType = trackedChange.getType()
```

6.218 Класс `TrackedChangeInfo`

Класс `TrackedChangeInfo` содержит информацию об отслеживаемых изменениях. Используется в [TrackedChange.getInfo](#), [Comment.getInfo](#).

Таблица 131 – Описание полей класса `TrackedChangeInfo`

Поле	Тип	Описание
<code>TrackedChangeInfo.author</code>	UserInfo	Автор изменений
<code>TrackedChangeInfo.timeStamp</code>	DateTime	Дата и время изменений

Пример

```
trackedChangeInfo = trackedChange.getInfo()  
author = trackedChangeInfo.author  
if author != None:  
    print(author.name)
```

```
timestamp = trackedChangeInfo.timestamp
if timestamp != None:
    print(timestamp.year, timestamp.month, timestamp.day)
```

6.219 Перечисление TrackedChangeType

Перечисление `TrackedChangeType` содержит типы отслеживаемых изменений, возвращается методом [TrackedChange.getType\(\)](#).

Таблица 132 – Типы отслеживаемых изменений

Значение	Описание
<code>TrackedChangeType_Added</code>	Добавленные изменения
<code>TrackedChangeType_Removed</code>	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")
```

6.220 Класс UnaryConditionalFormatOperator

Класс `UnaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
UnaryConditionalFormatOperator(ConditionalFormatUnaryCondition condition,
string argument)
```

Пример

Количество
2
5
10
1
3
4
6
15
20
2
7

Рисунок 54 – Пример создания правила для значений ≥ 10

```
sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

unaryStyle = myOfficeSDK.ConditionalFormatCellStyle()
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
unaryStyle.cellProperties = cellProperties

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

unaryOperator =
myOfficeSDK.createUnaryConditionalFormatOperator(myOfficeSDK.ConditionalFormatUnaryCondition_GreaterOrEqual, "10")

unaryRule = myOfficeSDK.ConditionalFormatRule(unaryOperator, unaryStyle,
cellRangePosition, False)
rules.addRule(unaryRule)
```

6.220.1 Метод UnaryConditionalFormatOperator.getArgument

Метод возвращает аргумент условия.

Вызов

```
string getArgument()
```

Возвращает

– аргумент условия, тип `string`.

6.220.2 Метод `UnaryConditionalFormatOperator.getCondition`

Метод возвращает условие применения форматирования.

Вызов

```
ConditionalFormatUnaryCondition getCondition()
```

Возвращает

– условие применения форматирования, тип [ConditionalFormatUnaryCondition](#).

6.220.3 Метод `UnaryConditionalFormatOperator.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.221 Класс `UniquenessConditionalFormatOperator`

Класс `UniquenessConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Уникальные и повторяющиеся значения". Используется при создании правила условного форматирования ([ConditionalFormatRule](#)).

Конструктор

```
UniquenessConditionalFormatOperator(ConditionalFormatUniquenessCondition condition)
```

Пример

Город
Москва
Санкт-Петербург
Алматы
Минск
Екатеринбург
Казань
Новосибирск
Самара
Астана
Екатеринбург
Минск

Рисунок 55 – Пример создания правила для повторяющихся значений

```

sheet = document.getBlocks().getTable(0)
rules = sheet.getConditionalFormatRules()

uniqueStyle = myOfficeSDK.ConditionalFormatCellStyle()
cellProperties = myOfficeSDK.CellProperties()
cellProperties.fill =
myOfficeSDK.Fill(myOfficeSDK.Color(myOfficeSDK.ColorRGBA(255, 0, 0, 150)))
uniqueStyle.cellProperties = cellProperties

cellRange = sheet.getCellRange("D2:D12")
cellRangePosition = cellRange.getTableRange()

uniqueOperator =
myOfficeSDK.createUniquenessConditionalFormatOperator(myOfficeSDK.ConditionalForm
atUniquenessCondition_Duplicate)

uniqueRule = myOfficeSDK.ConditionalFormatRule(uniqueOperator, uniqueStyle,
cellRangePosition, False)
rules.addRule(uniqueRule)

```

6.221.1 Метод UniquenessConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
ConditionalFormatUniquenessCondition getCondition()
```

Возвращает

– условие применения форматирования, тип [ConditionalFormatUniquenessCondition](#).

6.221.2 Метод `UniquenessConditionalFormatOperator.getType`

Метод возвращает тип оператора условного форматирования.

Вызов

```
ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип [ConditionalFormatOperatorType](#).

6.222 Класс `UserInfo`

Класс `UserInfo` предоставляет информацию о пользователе, используется в [TrackedChangeInfo](#), [DocumentSettings](#).

Таблица 133 – Описание полей класса `UserInfo`

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

6.222.1 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");
```

6.222.2 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")
```

6.223 Перечисление `ValueFieldsOrientation`

Класс `ValueFieldsOrientation` описывает варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений. Является полем класса [PivotTableLayoutSettings](#).

Таблица 134 – Варианты ориентации значений

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

6.224 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](#).

6.224.1 Метод `ValuesTableFilter.add`

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

Пример

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

6.224.2 Метод `ValuesTableFilter.clear`

Метод `ValuesTableFilter.clear` удаляет все элементы фильтра.

Пример

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

6.224.3 Метод `ValuesTableFilter.remove`

Метод удаляет заданное значение из фильтра.

Вызов

```
remove(value)
```

Параметры

– `value`: значение фильтра, тип `string`.

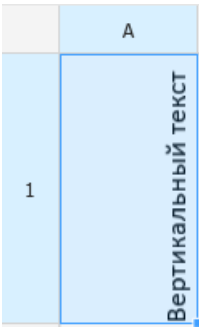
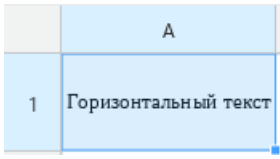
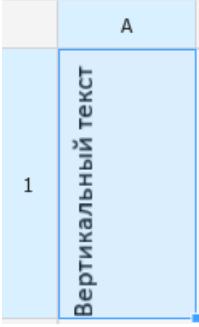
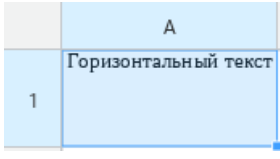
Пример

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

6.225 Перечисление VerticalAlignment

В таблице 135 представлены константы, описывающие варианты выравнивания текста по вертикали. Используется в [CellProperties](#), [ShapeProperties](#).

Таблица 135 – Виды выравнивания текста по вертикали

Значение	Представление в интерфейсе	
VerticalAlignment_Bottom		
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)
```

6.226 Перечисление VerticalAnchorAlignment

В таблице 136 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали (см. описание класса [VerticalTextAnchoredPosition](#)).

Таблица 136 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

Значение	Описание
VerticalAnchorAlignment_Top	По верхнему краю
VerticalAnchorAlignment_Bottom	По нижнему краю
VerticalAnchorAlignment_Center	По центру
VerticalAnchorAlignment_Inside, VerticalAnchorAlignment_Outside	По границам

6.227 Перечисление VerticalRelativeTo

В таблице 137 представлены типы размещения объекта относительно закрепленной позиции по вертикали (см. описание класса [VerticalTextAnchoredPosition](#)).

Таблица 137 – Типы размещения объекта относительно закрепленной позиции по вертикали

Значение	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

6.228 Класс VerticalTextAnchoredPosition

Класс VerticalTextAnchoredPosition предназначен для управления относительным положением объекта со смещением или выравниванием по вертикали (см. описание класса [TextAnchoredPosition](#)).

Таблица 138 – Описание полей класса VerticalTextAnchoredPosition

Поле	Тип	Описание
VerticalTextAnchoredPosition.relativeTo	VerticalRelativeTo	Тип размещения объекта относительно закрепленной позиции по вертикали
VerticalTextAnchoredPosition.offset	float	Смещение объекта
VerticalTextAnchoredPosition.alignment	VerticalAnchorAlignment	Тип выравнивания объекта относительно закрепленной позиции по вертикали

6.228.1 VerticalTextAnchoredPosition.__eq__

Метод __eq__ используется для определения эквивалентности двух объектов типа TextAnchoredPosition.

Пример

```
firstVerticalTextAnchoredPosition =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Paragraph)
firstVerticalTextAnchoredPosition.alignment =
myOfficeSDK.VerticalAnchorAlignment_Center
firstVerticalTextAnchoredPosition.offset = 10

secondVerticalTextAnchoredPosition =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Paragraph)
```

```
secondVerticalTextAnchoredPosition.alignment =
myOfficeSDK.VerticalAnchorAlignment_Center
secondVerticalTextAnchoredPosition.offset = 10

if firstVerticalTextAnchoredPosition.__eq__(secondVerticalTextAnchoredPosition):
    print("Equals")
```

6.228.2 VerticalTextAnchoredPosition.__ne__

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

Пример

```
firstVerticalTextAnchoredPosition =
myOfficeSDK
.VerticalTextAnchoredPosition(myOfficeSDK.VerticalRelativeTo_Paragraph)
firstVerticalTextAnchoredPosition.alignment =
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")
```

6.229 Класс WorkbookExportSettings

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

Таблица 139 – Описание полей класса `WorkbookExportSettings`

Поле	Тип	Описание
<code>WorkbookExportSettings.sheetNames</code>	<code>VectorString</code>	Представляет коллекцию имен листов для экспорта. Если коллекция пуста, экспортируются все листы.

Поле	Тип	Описание
WorkbookExportSettings.printingScope	PrintingScope	Представляет область печати (весь документ, область печати, пользовательский диапазон и т. д.).
WorkbookExportSettings.pageProperties	PageProperties	Представляют свойства страницы для выходного документа (высота и ширина страницы в пунктах).
WorkbookExportSettings.scale	float	Представляет масштаб экспорта выходного документа в процентах (например, 50,0%, 150,63%, 400,0% и т. д.).
WorkbookExportSettings.drawNoneBorders	bool	Добавляет стандартные границы ячеек в итоговый документ.
WorkbookExportSettings.printEmptyPages	bool	Включает пустые страницы в итоговый документ.
WorkbookExportSettings.pageID	int	Индекс страницы, с которой начинается экспорт.
WorkbookExportSettings.fitType	WorksheetPrinterFitType	Вариант масштабирования при экспорте табличных документов.

Пример

```
workbookSettings = myOfficeSDK.WorkbookExportSettings()
workbookSettings.sheetNames = myOfficeSDK.VectorString(['List1', 'List3'])
workbookSettings.printingScope =
myOfficeSDK.PrintingScope(myOfficeSDK.PrintingScope.Type_PrintArea)
workbookSettings.pageProperties = myOfficeSDK.PageProperties(100, 200)
workbookSettings.scale = 90
workbookSettings.printEmptyPages = False
workbookSettings.drawNoneBorders = False
workbookSettings.fitType = myOfficeSDK.WorksheetPrinterFitType_FitToPage
filePath = "C:\\work\\Sheet.pdf"
document.exportAs(filePath, myOfficeSDK.ExportFormat_PDFA1, workbookSettings)
```

6.230 Класс WorksheetHtmlExportSettings

Класс `WorksheetHtmlExportSettings` предназначен для настройки генерации HTML документа на основе листа табличного документа. Данный класс используется в методе [exportWorksheetToHtml\(\)](#).

Таблица 140 – Описание полей класса WorksheetHtmlExportSettings

Поле	Тип	Описание
WorksheetHtmlExportSettings.exportHeaders	bool	Добавляет заголовки строк и столбцов
WorksheetHtmlExportSettings.exportMissingBorders	bool	Добавляет линии сетки

6.231 Перечисление WorksheetPrinterFitType

Перечисление WorksheetPrinterFitType содержит варианты масштабирования при экспорте табличных документов. Используется в поле [WorkbookExportSettings.fitType](#).

Таблица 141 – Варианты масштабирования при экспорте табличных документов

Значение	Описание
WorksheetPrinterFitType_ActualSize	Фактический размер
WorksheetPrinterFitType_ByPageScale	По масштабу страницы
WorksheetPrinterFitType_ByPageBreaksOnly	По разрыву страниц
WorksheetPrinterFitType_FitToPage	Вписать в страницу
WorksheetPrinterFitType_FitToWidth	Вписать по ширине
WorksheetPrinterFitType_FitToHeight	Вписать по высоте

6.232 Исключения

6.232.1 Класс BaseError

Класс BaseError является базовым классом для всех исключений SDK.

```
class BaseError(Exception)
```

6.232.2 Класс ApplicationCreateError

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

```
class ApplicationCreateError(BaseError)
```

6.232.3 Класс DocumentCreateError

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

```
class DocumentCreateError(BaseError)
```

6.232.4 Класс DocumentExportError

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

```
class DocumentExportError(BaseError)
```

6.232.5 Класс DocumentLoadError

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

```
class DocumentLoadError(BaseError)
```

6.232.6 Класс DocumentModificationError

Исключение `DocumentModificationError` вызывается, когда невозможно выполнить операцию по изменению документа. Например, оно возникает при попытке применить методы [Paragraph.setListLevel\(\)](#), [Paragraph.increaseListLevel\(\)](#), [Paragraph.decreaseListLevel\(\)](#) для абзаца, который не является списком.

```
class DocumentModificationError(BaseError)
```

6.232.7 Класс DocumentSaveError

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

```
class DocumentSaveError(BaseError)
```

6.232.8 Класс ForbiddenActionError

Исключение `ForbiddenActionError` вызывается в случае выполнения запрещенной операции.

```
class ForbiddenActionError(BaseError)
```

6.232.9 Класс `IncorrectArgumentError`

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

```
class IncorrectArgumentError(BaseError)
```

6.232.10 Класс `InvalidObjectError`

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

```
class InvalidObjectError(BaseError)
```

6.232.11 Класс `NoSuchElementError`

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

```
class NoSuchElementError(BaseError)
```

6.232.12 Класс `NotImplementedError`

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

```
class NotImplementedError(BaseError)
```

6.232.13 Класс `OutOfRangeError`

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

```
class OutOfRangeError(BaseError)
```

6.232.14 Класс `ParseError`

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

```
class ParseError(BaseError)
```

6.232.15 Класс PivotTableError

Исключение `PivotTableError` вызывается в случае ошибки при работе со сводными таблицами. Например, использование фильтра, который не может быть применен к сводной таблице.

```
class PivotTableError(BaseError)
```

6.232.16 Класс PositionDocumentMismatchError

Исключение `PositionDocumentsMismatchError` вызывается в случае, когда несколько позиций относятся к различным документам и не могут быть использованы в одной операции. Например, при попытке пользователя создать диапазон `Range`, включающий позиции `Position`, принадлежащие нескольким различным документам, и выполнить операцию для такого диапазона.

```
class PositionDocumentMismatchError(BaseError)
```

6.232.17 Класс ScriptExecutionError

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

```
class ScriptExecutionError(BaseError)
```

6.232.18 Класс UnknownError

Исключение `UnknownError` вызывается в случае, когда критическое исключение возникло по неизвестной причине. Приложение должно быть завершено, поскольку возникло неопределенное состояние ядра Document API.

```
class UnknownError(BaseError)
```

7 МЕХАНИЗМ КОНТРОЛЯ ВЕРСИЙ DOCUMENT API

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.)
```