

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

**НАСТОЛЬНЫЕ ПРИЛОЖЕНИЯ
«МОЙОФИС ТЕКСТ», «МОЙОФИС ТАБЛИЦА»**

3.3

**СПРАВОЧНИК МАКРОКОМАНД НА ЯЗЫКЕ
ПРОГРАММИРОВАНИЯ LUA**

Версия 2

На 353 листах

Дата публикации: 21.03.2025

**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	27
1.1	Назначение	27
1.2	Макрокоманды ПО МойОфис	27
1.3	Перечень эксплуатационной документации	28
2	Работа с макрокомандами	29
2.1	Редактор макрокоманд	29
2.1.1	Окно редактора макрокоманд	29
2.1.2	Создание макрокоманд	30
2.1.3	Выполнение макрокоманд	30
2.1.4	Редактирование макрокоманд	30
2.1.5	Удаление макрокоманд	31
2.1.6	Отладка макрокоманд	31
2.2	Пример подготовки и запуска макрокоманды	34
2.2.1	Редактирование и запуск макрокоманды в текстовом документе	34
2.2.2	Описание примера работы макрокоманды	34
2.3	Преобразование макрокоманд на языке программирования VBA	36
3	Объектная модель МойОфис	37
4	Работа с документами	38
4.1	Работа с текстовым документом	38
4.1.1	Разделы (секции) документа	38
4.1.1.1	Работа с колонтитулами раздела	39
4.1.1.2	Управление ориентацией и свойствами страниц раздела	39
4.1.2	Работа с таблицами текстового документа	40
4.1.3	Работа с закладками	42
4.1.4	Рецензирование документов	43
4.1.5	Работа с графическими объектами в текстовом документе	44
4.1.6	Работа с элементами управления	47
4.2	Работа с табличным документом	48
4.2.1	Работа с текстом в табличном документе	48
4.2.2	Копирование ячеек в табличном документе	48
4.2.3	Диаграммы	49
4.2.4	Работа с формулами	49

МойОфис

4.2.5	Проверка данных	51
4.2.6	Работа с графическими объектами в табличном документе	54
4.2.7	Работа с листами табличного документа	56
4.2.8	Работа со сводными таблицами	58
4.2.8.1	Создание сводной таблицы	58
4.2.8.2	Получение диапазона исходных данных сводной таблицы	59
4.2.8.3	Получение диапазона размещения сводной таблицы	59
4.2.8.4	Получение флагов отображения общих итогов для строк и колонок	59
4.2.8.5	Получение заголовков сводной таблицы	60
4.2.8.6	Получение и применение фильтра для сводной таблицы	60
4.2.8.7	Получение полей из области фильтров	60
4.2.8.8	Получение полей из области значений	61
4.2.8.9	Получение полей из области строк	61
4.2.8.10	Получение полей из области колонок	62
4.2.8.11	Получение настроек отображения сводной таблицы	62
4.2.8.12	Обновление сводной таблицы	62
4.2.9	Работа с фильтрами	63
4.2.10	Обработка событий табличного документа	64
4.3	Поиск в документе	66
4.4	Работа с макрокомандами	67
4.5	Работа с именованными диапазонами	68
5	Работа со строками и столбцами таблиц	69
5.1	Группировка строк и колонок таблицы	69
5.2	Управление видимостью строк / колонок	69
6	Работа с ячейками таблиц	71
6.1	Доступ к ячейкам	71
6.2	Форматирование ячеек	73
6.3	Форматирование границ ячеек	75
6.4	Объединение и разделение ячеек таблицы	76
7	Справочник таблиц DocumentAPI	77
7.1	Таблица DocumentAPI.AbsoluteFrame	77
7.1.1	Метод AbsoluteFrame:getDimensions	77
7.1.2	Метод AbsoluteFrame:getTopLeft	78
7.1.3	Метод AbsoluteFrame:moveTo	78

7.1.4	Метод AbsoluteFrame:scale	78
7.1.5	Метод AbsoluteFrame:setDimensions	79
7.2	Таблица DocumentAPI.AccountingCellFormatting	79
7.3	Таблица DocumentAPI.Alignment	80
7.4	Таблица DocumentAPI.Block	81
7.4.1	Метод Block:getRange	81
7.4.2	Метод Block:getSection	81
7.4.3	Метод Block:remove	82
7.4.4	Методы toParagraph, toTable, toShape, toField	82
7.5	Таблица DocumentAPI.Blocks	82
7.5.1	Метод Blocks:enumerate	83
7.5.2	Метод Blocks:enumerateParagraphs	83
7.5.3	Метод Blocks:enumerateTables	83
7.5.4	Метод Blocks:getBlock	83
7.5.5	Метод Blocks:getField	83
7.5.6	Метод Blocks:getParagraph	84
7.5.7	Метод Blocks:getShape	84
7.5.8	Метод Blocks:getTable	84
7.6	Таблица DocumentAPI.Bookmarks	84
7.6.1	Метод Bookmarks:getBookmarkRange	84
7.6.2	Метод Bookmarks:removeBookmark	85
7.7	Таблица DocumentAPI.Borders	85
7.8	Таблица DocumentAPI.CalculationMode	86
7.9	Таблица DocumentAPI.CaseSensitive	87
7.10	Таблица DocumentAPI.Cell	87
7.10.1	Метод Cell:checkDataValidation	87
7.10.2	Метод Cell:getBorders	88
7.10.3	Метод Cell:getCellProperties	88
7.10.4	Метод Cell:getColumnIndex	88
7.10.5	Метод Cell:getCurrentRegion	88
7.10.6	Метод Cell:getCustomFormat	89
7.10.7	Метод Cell:getDataValidation	89
7.10.8	Метод Cell:getFormat	89
7.10.9	Метод Cell:getFormattedValue	90

МойОфис

7.10.10	Метод Cell:getFormulaAsString	90
7.10.11	Метод Cell:getHyperlink	90
7.10.12	Метод Cell:getMergedRange	90
7.10.13	Метод Cell:getNote	91
7.10.14	Метод Cell:getParagraphProperties	91
7.10.15	Метод Cell:getPivotTable	91
7.10.16	Метод Cell:getProtectionProperties	92
7.10.17	Метод Cell:getRange	92
7.10.18	Метод Cell:getRawValue	92
7.10.19	Метод Cell:getRowIndex	93
7.10.20	Метод Cell:getTable	93
7.10.21	Метод Cell:getTextProperties	93
7.10.22	Метод Cell:isInMergedRange	94
7.10.23	Метод Cell:isPivotTableRoot	94
7.10.24	Метод Cell:isProtected	94
7.10.25	Метод Cell:removeNote	94
7.10.26	Метод Cell:setBool	95
7.10.27	Метод Cell:setBorders	95
7.10.28	Метод Cell:setCellProperties	95
7.10.29	Метод Cell:setContent	95
7.10.30	Метод Cell:setCustomFormat	96
7.10.31	Метод Cell:setFormat	96
7.10.32	Метод Cell:setFormattedValue	98
7.10.33	Метод Cell:setFormula	99
7.10.34	Метод Cell:setNote	99
7.10.35	Метод Cell:setNumber	99
7.10.36	Метод Cell:setParagraphProperties	99
7.10.37	Метод Cell:setProtectionProperties	100
7.10.38	Метод Cell:setText	101
7.10.39	Метод Cell:setTextProperties	101
7.10.40	Метод Cell:unmerge	101
7.11	Таблица DocumentAPI.CellFormat	102
7.12	Таблица DocumentAPI.CellPosition	104
7.12.1	Поле CellPosition.column	104

МойОфис

7.12.2	Поле CellPosition.row	105
7.12.3	Метод CellPosition.toString	105
7.12.4	Метод CellPosition:___eq	105
7.13	Таблица DocumentAPI.CellProperties	105
7.13.1	Метод CellProperties:___eq	107
7.14	Таблица DocumentAPI.CellProtectionProperties	107
7.15	Таблица DocumentAPI.CellRange	108
7.15.1	Метод CellRange:autoFill	108
7.15.2	Метод CellRange:clearDataValidations	108
7.15.3	Метод CellRange:containsCell	109
7.15.4	Метод CellRange:copyInto	109
7.15.5	Метод CellRange:enumerate	110
7.15.6	Метод CellRange:find	110
7.15.7	Метод CellRange:getAddress	111
7.15.8	Метод CellRange:getBeginColumn	112
7.15.9	Метод CellRange:getBeginRow	112
7.15.10	Метод CellRange:getCellProperties	112
7.15.11	Метод CellRange:getLastColumn	112
7.15.12	Метод CellRange:getLastRow	113
7.15.13	Метод CellRange:getParagraphProperties	113
7.15.14	Метод CellRange:getProtectionProperties	114
7.15.15	Метод CellRange:getTable	114
7.15.16	Метод CellRange:getTableRange	114
7.15.17	Метод CellRange:getTextProperties	115
7.15.18	Метод CellRange:insert	115
7.15.19	Метод CellRange:insertCurrentDateTime	116
7.15.20	Метод CellRange:isProtected	116
7.15.21	Метод CellRange:merge	116
7.15.22	Метод CellRange:moveInto	117
7.15.23	Метод CellRange:remove	117
7.15.24	Метод CellRange:setBorders	118
7.15.25	Метод CellRange:setCellProperties	118
7.15.26	Метод CellRange:setDataValidation	118
7.15.27	Метод CellRange:setParagraphProperties	119

МойОфис

7.15.28	Метод CellRange:setProtectionProperties	119
7.15.29	Метод CellRange:setTextProperties	120
7.15.30	Метод CellRange:sort	121
7.16	Таблица DocumentAPI.CellRangeAddressFormat	121
7.17	Таблица DocumentAPI.CellRangeAddressSettings	122
7.18	Таблица DocumentAPI.CellRangePosition	123
7.18.1	Метод CellRangePosition:toString	124
7.18.2	Метод CellRangePosition:___eq	124
7.19	Таблица DocumentAPI.CellRanges	125
7.19.1	Метод CellRanges:addRange	125
7.19.2	Метод CellRanges:clear	125
7.19.3	Метод CellRanges:enumerate	125
7.20	Таблица DocumentAPI.CellShiftAxis	126
7.21	Таблица DocumentAPI.Chart	126
7.21.1	Метод Chart:applySettings	127
7.21.2	Метод Chart:getChartLabels	128
7.21.3	Метод Chart:getDirectionType	128
7.21.4	Метод Chart:getRange	128
7.21.5	Метод Chart:getRangeAsString	129
7.21.6	Метод Chart:getRangesCount	129
7.21.7	Метод Chart:getTitle	129
7.21.8	Метод Chart:getType	129
7.21.9	Метод Chart:is3D	129
7.21.10	Метод Chart:isEmpty	130
7.21.11	Метод Chart:isSolidRange	130
7.21.12	Метод Chart:setRange	130
7.21.13	Метод Chart:setRect	130
7.21.14	Метод Chart:setType	130
7.22	Таблица DocumentAPI.ChartLabelsDetectionMode	131
7.23	Таблица DocumentAPI.ChartLabelsInfo	131
7.24	Таблица DocumentAPI.ChartRangeInfo	132
7.25	Таблица DocumentAPI.ChartRangeType	133
7.26	Таблица DocumentAPI.Charts	133
7.26.1	Метод Charts:getChart	134

7.26.2	Метод Charts:getChartIndexByDrawingIndex	134
7.26.3	Метод Charts:getChartsCount	135
7.27	Таблица DocumentAPI.ChartSeriesDirectionType	135
7.28	Таблица DocumentAPI.ChartType	135
7.29	Таблица DocumentAPI.CheckBoxControl	136
7.30	Таблица DocumentAPI.Color	137
7.30.1	Метод Color:getRGBAColor	137
7.30.2	Метод Color:getThemeColorID	137
7.30.3	Метод Color:getTransforms	137
7.30.4	Метод Color:setTransforms	138
7.30.5	Метод Color:___eq	138
7.31	Таблица DocumentAPI.ColorRGBA	138
7.31.1	Метод ColorRGBA:___eq	139
7.32	Таблица DocumentAPI.ColorTransforms	139
7.32.1	Метод ColorTransforms:apply	140
7.33	Таблица DocumentAPI.Comment	140
7.33.1	Метод Comment:getInfo	140
7.33.2	Метод Comment:getRange	140
7.33.3	Метод Comment:getReplies	141
7.33.4	Метод Comment:getText	141
7.33.5	Метод Comment:isResolved	141
7.34	Таблица DocumentAPI.Comments	141
7.34.1	Метод Comments:enumerate	142
7.35	Таблица DocumentAPI.ConditionalTableFilter	142
7.35.1	Метод ConditionalTableFilter:setAndOperation	143
7.35.2	Методы добавления условий	143
7.36	Таблица DocumentAPI.ContentControl	144
7.36.1	Метод ContentControl:canEdit	144
7.36.2	Метод ContentControl:getTitle	144
7.36.3	Методы toCheckBox, toInputField, toDatePicker, toDropList	145
7.37	Таблица DocumentAPI.ContentControls	145
7.38	Таблица DocumentAPI.CurrencyCellFormatting	145
7.39	Таблица DocumentAPI.CurrencySignPlacement	146
7.40	Таблица DocumentAPI.DataValidation	147

7.40.1	Метод <code>DataValidation:clear</code>	147
7.40.2	Метод <code>DataValidation:getAllowBlank</code>	147
7.40.3	Метод <code>DataValidation:getErrorMessage</code>	147
7.40.4	Метод <code>DataValidation:getErrorStyle</code>	147
7.40.5	Метод <code>DataValidation:getErrorTitle</code>	148
7.40.6	Метод <code>DataValidation:getFormula1</code>	148
7.40.7	Метод <code>DataValidation:getFormula2</code>	149
7.40.8	Метод <code>DataValidation:getOperator</code>	149
7.40.9	Метод <code>DataValidation:getPrompt</code>	149
7.40.10	Метод <code>DataValidation:getPromptTitle</code>	149
7.40.11	Метод <code>DataValidation:getShowDropDown</code>	150
7.40.12	Метод <code>DataValidation:getShowErrorMessage</code>	150
7.40.13	Метод <code>DataValidation:getShowInputMessage</code>	150
7.40.14	Метод <code>DataValidation:getType</code>	150
7.40.15	Метод <code>DataValidation:isEmpty</code>	151
7.40.16	Метод <code>DataValidation:setAllowBlank</code>	151
7.40.17	Метод <code>DataValidation:setErrorMessage</code>	151
7.40.18	Метод <code>DataValidation:setErrorStyle</code>	152
7.40.19	Метод <code>DataValidation:setErrorTitle</code>	152
7.40.20	Метод <code>DataValidation:setFormula1</code>	153
7.40.21	Метод <code>DataValidation:setFormula2</code>	153
7.40.22	Метод <code>DataValidation:setOperator</code>	154
7.40.23	Метод <code>DataValidation:setPrompt</code>	155
7.40.24	Метод <code>DataValidation:setPromptTitle</code>	155
7.40.25	Метод <code>DataValidation:setShowDropDown</code>	156
7.40.26	Метод <code>DataValidation:setShowErrorMessage</code>	156
7.40.27	Метод <code>DataValidation:setShowInputMessage</code>	157
7.40.28	Метод <code>DataValidation:setType</code>	157
7.41	Таблица <code>DocumentAPI.DataValidationErrorStyle</code>	157
7.42	Таблица <code>DocumentAPI.DataValidationOperator</code>	158
7.43	Таблица <code>DocumentAPI.DataValidationResult</code>	158
7.43.1	Метод <code>DataValidationResult:getDataValidation</code>	158
7.43.2	Метод <code>DataValidationResult:isValid</code>	159
7.44	Таблица <code>DocumentAPI.DataValidationType</code>	159

7.45	Таблица DocumentAPI.DatePatterns	160
7.46	Таблица DocumentAPI.DatePickerControl	161
7.47	Таблица DocumentAPI.DateTime	161
7.47.1	Метод DateTime: __eq	161
7.48	Таблица DocumentAPI.DateTimeCellFormatting	162
7.49	Таблица DocumentAPI.DateTimeFormat	162
7.50	Таблица DocumentAPI.document	162
7.50.1	Метод document:areMirroredMarginsEnabled	163
7.50.2	Метод document:calculateAllFormulas	163
7.50.3	Метод document:calculateOutdatedFormulas	163
7.50.4	Метод document:disableEvents	164
7.50.5	Метод document:enableEvents	164
7.50.6	Метод document:enumerateSections	164
7.50.7	Метод document:getAbsolutePath	164
7.50.8	Метод document:getBlocks	165
7.50.9	Метод document:getBookmarks	165
7.50.10	Метод document:getCalculationMode	165
7.50.11	Метод document:getComments	165
7.50.12	Метод document:getContentControls	166
7.50.13	Метод document:getFormulaType	166
7.50.14	Метод document:getNamedExpressions	166
7.50.15	Метод document:getPivotTablesManager	166
7.50.16	Метод document:getRange	167
7.50.17	Метод document:getScripts	167
7.50.18	Метод document:getSections	167
7.50.19	Метод document:isCalculatedOnSave	167
7.50.20	Метод document:isChangesTrackingEnabled	167
7.50.21	Метод document:isEventsEnabled	168
7.50.22	Метод document:isStructureProtected	168
7.50.23	Метод document:removeStructureProtection	168
7.50.24	Метод document:setCalculatedOnSave	169
7.50.25	Метод document:setCalculationMode	169
7.50.26	Метод document:setChangesTrackingEnabled	169
7.50.27	Метод document:setFormulaType	169

МойОфис

7.50.28	Метод document:setMirroredMarginsEnabled	170
7.50.29	Метод document:setPageOrientation	170
7.50.30	Метод document:setPageProperties	170
7.50.31	Метод document:setStructureProtection	170
7.51	Таблица DocumentAPI.DropListControl	171
7.52	Таблица DocumentAPI.Field	171
7.53	Таблица DocumentAPI.Fill	171
7.53.1	Метод Fill:getColor	172
7.53.2	Метод Fill:getUrl	172
7.53.3	Метод Fill:isNoFill	172
7.54	Таблица DocumentAPI.FiltersRange	172
7.54.1	Метод FiltersRange:clear	172
7.54.2	Метод FiltersRange:eraseFilters	172
7.54.3	Метод FiltersRange:getCellRange	172
7.54.4	Метод FiltersRange:setFilters	173
7.55	Таблица DocumentAPI.FractionCellFormatting	173
7.56	Таблица DocumentAPI.FrozenRangePosition	174
7.56.1	Конструкторы	174
7.56.2	Метод FrozenRangePosition:create	175
7.56.3	Метод FrozenRangePosition:createFrozenArea	175
7.56.4	Метод FrozenRangePosition:createFrozenCols	175
7.56.5	Метод FrozenRangePosition:createFrozenRows	175
7.56.6	Метод FrozenRangePosition:isArea	176
7.56.7	Метод FrozenRangePosition:isCols	176
7.56.8	Метод FrozenRangePosition:isRows	176
7.56.9	Метод FrozenRangePosition:isRowsCols	176
7.56.10	Метод FrozenRangePosition:___eq	176
7.57	Таблица DocumentAPI.HeaderFooter	177
7.57.1	Метод HeaderFooter:getBlocks	177
7.57.2	Метод HeaderFooter:getRange	177
7.57.3	Метод HeaderFooter:getType	177
7.58	Таблица DocumentAPI.HeaderFooterType	178
7.59	Таблица DocumentAPI.HeadersFooters	178
7.59.1	Метод HeadersFooters:enumerate	178

7.60	Таблица DocumentAPI.HorizontalAnchorAlignment	179
7.61	Таблица DocumentAPI.HorizontalRelativeTo	179
7.62	Таблица DocumentAPI.HorizontalTextAnchoredPosition	180
7.62.1	Метод HorizontalTextAnchoredPosition: __eq	181
7.63	Таблица DocumentAPI.Hyperlink	181
7.63.1	Метод Hyperlink: __eq	182
7.64	Таблица DocumentAPI.Image	182
7.64.1	Метод Image:getFrame	182
7.64.2	Метод Image:remove	183
7.65	Таблица DocumentAPI.Images	183
7.65.1	Метод Images:enumerate	183
7.66	Таблица DocumentAPI.InlineFrame	184
7.66.1	Метод InlineFrame:getDimensions	184
7.66.2	Метод InlineFrame:getPosition	185
7.66.3	Метод InlineFrame:getWrapType	185
7.66.4	Метод InlineFrame:setDimensions	185
7.66.5	Метод InlineFrame:setPosition	185
7.66.6	Метод InlineFrame:setWrapType	186
7.67	Таблица DocumentAPI.InputFieldControl	186
7.68	Таблица DocumentAPI.Insets	187
7.69	Таблица DocumentAPI.LineEndingProperties	187
7.69.1	Метод LineEndingProperties: __eq	188
7.70	Таблица DocumentAPI.LineEndingStyle	189
7.71	Таблица DocumentAPI.LineProperties	189
7.71.1	Поле LineProperties.color	190
7.71.2	Поле LineProperties.headLineEndingProperties	190
7.71.3	Поле LineProperties.style	191
7.71.4	Поле LineProperties.tailLineEndingProperties	191
7.71.5	Поле LineProperties.width	191
7.71.6	Метод LineProperties: __eq	191
7.72	Таблица DocumentAPI.LineSpacing	191
7.73	Таблица DocumentAPI.LineSpacingRule	192
7.74	Таблица DocumentAPI.LineStyle	194
7.75	Таблица DocumentAPI.ListSchema	195

7.76	Таблица DocumentAPI.MediaObject	196
7.76.1	Метод MediaObject:getFrame	196
7.76.2	Метод MediaObject:toChart	196
7.76.3	Метод MediaObject:toImage	197
7.77	Таблица DocumentAPI.MediaObjects	197
7.77.1	Метод MediaObjects:enumerate	198
7.78	Таблица DocumentAPI.NamedExpression	199
7.78.1	Метод NamedExpression:getCellRange	199
7.78.2	Метод NamedExpression:getExpression	199
7.78.3	Метод NamedExpression:getName	199
7.79	Таблица DocumentAPI.NamedExpressions	199
7.79.1	Метод NamedExpressions:addExpression	199
7.79.2	Метод NamedExpressions:enumerate	200
7.79.3	Метод NamedExpressions:get	200
7.79.4	Метод NamedExpressions:removeExpression	200
7.80	Таблица DocumentAPI.NamedExpressionsValidationResult	201
7.81	Таблица DocumentAPI.NumberCellFormatting	201
7.82	Таблица DocumentAPI.PageFieldOrder	202
7.83	Таблица DocumentAPI.PageOrientation	202
7.84	Таблица DocumentAPI.PageProperties	203
7.84.1	Метод PageProperties: __eq	203
7.85	Таблица DocumentAPI.Paragraph	204
7.85.1	Метод Paragraph:decreaseListLevel	204
7.85.2	Метод Paragraph:getListLevel	205
7.85.3	Метод Paragraph:getListSchema	205
7.85.4	Метод Paragraph:getParagraphProperties	205
7.85.5	Метод Paragraph:increaseListLevel	206
7.85.6	Метод Paragraph:setListLevel	206
7.85.7	Метод Paragraph:setListSchema	206
7.85.8	Метод Paragraph:setParagraphProperties	206
7.86	Таблица DocumentAPI.ParagraphProperties	207
7.87	Таблица DocumentAPI.Paragraphs	210
7.87.1	Метод Paragraphs:decreaseListLevel	211
7.87.2	Метод Paragraphs:enumerate	211

7.87.3	Метод Paragraphs:increaseListLevel	212
7.87.4	Метод Paragraphs:setListLevel	212
7.87.5	Метод Paragraphs:setListSchema	212
7.88	Таблица DocumentAPI.PercentageCellFormatting	212
7.89	Таблица DocumentAPI.PivotTable	213
7.89.1	Метод PivotTable:areAllFiltersInDefaultState	213
7.89.2	Метод PivotTable:changeSourceRange	213
7.89.3	Метод PivotTable:createPivotTableEditor	214
7.89.4	Метод PivotTable:getColumnFields	214
7.89.5	Метод PivotTable:getConditionalLabelFilter	214
7.89.6	Метод PivotTable:getConditionalValueFilter	215
7.89.7	Метод PivotTable:getFieldCategories	215
7.89.8	Метод PivotTable:getFieldItems	215
7.89.9	Метод PivotTable:getFieldItemsByName	216
7.89.10	Метод PivotTable:getFieldsList	216
7.89.11	Метод PivotTable:getFilter	216
7.89.12	Метод PivotTable:getFilters	216
7.89.13	Метод PivotTable:getPageFields	217
7.89.14	Метод PivotTable:getPivotRange	217
7.89.15	Метод PivotTable:getPivotTableCaptions	217
7.89.16	Метод PivotTable:getPivotTableLayoutSettings	217
7.89.17	Метод PivotTable:getRowFields	218
7.89.18	Метод PivotTable:getSourceRange	218
7.89.19	Метод PivotTable:getSourceRangeAddress	218
7.89.20	Метод PivotTable:getValueFields	219
7.89.21	Метод PivotTable:isColumnGrandTotalEnabled	219
7.89.22	Метод PivotTable:isRowGrandTotalEnabled	219
7.89.23	Метод PivotTable:remove	219
7.89.24	Метод PivotTable:update	220
7.90	Таблица DocumentAPI.PivotTableCaptions	220
7.91	Таблица DocumentAPI.PivotTableCategoryField	220
7.92	Таблица DocumentAPI.PivotTableConditionalLabelFilter	221
7.92.1	Метод PivotTableConditionalLabelFilter:getFieldName	221
7.92.2	Метод PivotTableConditionalLabelFilter:getOperation	221

7.92.3	Метод PivotTableConditionalLabelFilter:isInDefaultState	221
7.92.4	Метод PivotTableConditionalLabelFilter:reset	222
7.92.5	Метод PivotTableConditionalLabelFilter:setOperation	222
7.93	Таблица DocumentAPI.PivotTableConditionalLabelFilterOperation	222
7.94	Таблица DocumentAPI.PivotTableConditionalLabelFilterOperationType	223
7.95	Таблица DocumentAPI.PivotTableConditionalValueFilter	224
7.95.1	Метод PivotTableConditionalValueFilter:getDefaultOperation	224
7.95.2	Метод PivotTableConditionalValueFilter:getExpectedOperandType	225
7.95.3	Метод PivotTableConditionalValueFilter:getFieldName	225
7.95.4	Метод PivotTableConditionalValueFilter:getOperation	225
7.95.5	Метод PivotTableConditionalValueFilter:isInDefaultState	225
7.95.6	Метод PivotTableConditionalValueFilter:reset	226
7.95.7	Метод PivotTableConditionalValueFilter:setOperation	226
7.96	Таблица DocumentAPI.PivotTableConditionalValueFilterOperandType	226
7.97	Таблица DocumentAPI.PivotTableConditionalValueFilterOperation	227
7.98	Таблица DocumentAPI.PivotTableConditionalValueFilterOperationType	228
7.99	Таблица DocumentAPI.PivotTableEditor	228
7.99.1	Метод PivotTableEditor:addField	228
7.99.2	Метод PivotTableEditor:apply	229
7.99.3	Метод PivotTableEditor:disableField	229
7.99.4	Метод PivotTableEditor:enableField	229
7.99.5	Метод PivotTableEditor:moveField	230
7.99.6	Метод PivotTableEditor:removeField	230
7.99.7	Метод PivotTableEditor:reorderField	230
7.99.8	Метод PivotTableEditor:resetAllFilters	231
7.99.9	Метод PivotTableEditor:setCaptions	231
7.99.10	Метод PivotTableEditor:setFilter	231
7.99.11	Метод PivotTableEditor:setFilters	232
7.99.12	Метод PivotTableEditor:setGrandTotalSettings	232
7.99.13	Метод PivotTableEditor:setLayoutSettings	232
7.99.14	Метод PivotTableEditor:setSummarizeFunction	233
7.100	Таблица DocumentAPI.PivotTableField	233
7.101	Таблица DocumentAPI.PivotTableFieldCategories	234
7.101.1	Метод PivotTableFieldCategories:enumerate	234

7.102 Таблица DocumentAPI.PivotTableFieldCategory	234
7.103 Таблица DocumentAPI.PivotTableFieldProperties	234
7.104 Таблица DocumentAPI.PivotTableFilter	235
7.104.1 Метод PivotTableFilter:getCount	235
7.104.2 Метод PivotTableFilter:getFieldName	236
7.104.3 Метод PivotTableFilter:getName	236
7.104.4 Метод PivotTableFilter:isHidden	236
7.104.5 Метод PivotTableFilter:isInDefaultState	236
7.104.6 Метод PivotTableFilter:reset	237
7.104.7 Метод PivotTableFilter:setHidden	237
7.105 Таблица DocumentAPI.PivotTableFilters	237
7.105.1 Метод PivotTableFilters:enumerate	237
7.106 Таблица DocumentAPI.PivotTableFunction	238
7.107 Таблица DocumentAPI.PivotTableItem	238
7.107.1 Метод PivotTableItem:getAlias	239
7.107.2 Метод PivotTableItem:getItemType	239
7.107.3 Метод PivotTableItem:getName	239
7.107.4 Метод PivotTableItem:isCollapsed	239
7.108 Таблица DocumentAPI.PivotTableItems	239
7.108.1 Метод PivotTableItems:enumerate	240
7.109 Таблица DocumentAPI.PivotTableItemType	240
7.110 Таблица DocumentAPI.PivotTableLayoutSettings	241
7.111 Таблица DocumentAPI.PivotTablePageField	241
7.112 Таблица DocumentAPI.PivotTableReportLayout	242
7.113 Таблица DocumentAPI.PivotTablesManager	242
7.113.1 Метод PivotTablesManager:create	242
7.114 Таблица DocumentAPI.PivotTableUpdateResult	243
7.115 Таблица DocumentAPI.PivotTableValueField	244
7.116 Таблица DocumentAPI.PointU	244
7.116.1 Метод PointU:toString	245
7.117 Таблица DocumentAPI.Position	245
7.117.1 Метод Position:compare	245
7.117.2 Метод Position:getCell	246
7.117.3 Метод Position:getCurrentRange	246

МойОфис

7.117.4	Метод Position:getNextPosition	247
7.117.5	Метод Position:getNextRange	247
7.117.6	Метод Position:getParagraph	248
7.117.7	Метод Position:getPreviousPosition	248
7.117.8	Метод Position:getPreviousRange	249
7.117.9	Метод Position:insertBookmark	250
7.117.10	Метод Position:insertHyperlink	250
7.117.11	Метод Position:insertImage	250
7.117.12	Метод Position:insertLineBreak	251
7.117.13	Метод Position:insertPageBreak	251
7.117.14	Метод Position:insertSectionBreak	252
7.117.15	Метод Position:insertTable	252
7.117.16	Метод Position:insertText	252
7.117.17	Метод Position:removeBackward	253
7.117.18	Метод Position:removeForward	253
7.117.19	Метод Position:___eq	253
7.118	Таблица DocumentAPI.PrintDocumentResult	253
7.119	Таблица DocumentAPI.PrintSettings	254
7.120	Таблица DocumentAPI.Range	255
7.120.1	Конструктор DocumentAPI.Range	257
7.120.2	Метод Range:enumerateBlocks	257
7.120.3	Метод Range:enumerateTrackedChanges	258
7.120.4	Метод Range:extractText	258
7.120.5	Метод Range:getBegin	258
7.120.6	Метод Range:getComments	259
7.120.7	Метод Range:getContentEnd	259
7.120.8	Метод Range:getEnd	260
7.120.9	Метод Range:getImages	260
7.120.10	Метод Range:getInlineObjects	261
7.120.11	Метод Range:getParagraphs	261
7.120.12	Метод Range:getTextProperties	261
7.120.13	Метод Range:isContentLocked	262
7.120.14	Метод Range:lockContent	262
7.120.15	Метод Range:removeContent	263

МойОфис

7.120.16	Метод Range:replaceText	263
7.120.17	Метод Range:setHyperlink	263
7.120.18	Метод Range:setTextProperties	264
7.120.19	Метод Range:unlockContent	264
7.121	Таблица DocumentAPI.RangeBorders	265
7.122	Таблица DocumentAPI.RectU	265
7.122.1	Метод RectU:toString	265
7.123	Таблица DocumentAPI.ScaleFrom	266
7.124	Таблица DocumentAPI.ScientificCellFormatting	266
7.125	Таблица DocumentAPI.Script	267
7.125.1	Метод Script:getBody	267
7.125.2	Метод Script:getName	267
7.125.3	Метод Script:setBody	267
7.125.4	Метод Script:setName	267
7.126	Таблица DocumentAPI.Scripting	268
7.126.1	Метод Scripting:runScript	268
7.127	Таблица DocumentAPI.ScriptPosition	268
7.128	Таблица DocumentAPI.Scripts	268
7.128.1	Метод Scripts:enumerate	269
7.128.2	Метод Scripts:getScript	269
7.128.3	Метод Scripts:removeScript	269
7.128.4	Метод Scripts:setScript	269
7.129	Таблица DocumentAPI.Search	270
7.129.1	Метод Search:findText	270
7.130	Таблица DocumentAPI.Section	271
7.130.1	Метод Section:getFooters	271
7.130.2	Метод Section:getHeaders	271
7.130.3	Метод Section:getPageOrientation	271
7.130.4	Метод Section:getPageProperties	272
7.130.5	Метод Section:getRange	272
7.130.6	Метод Section:setPageOrientation	272
7.130.7	Метод Section:setPageProperties	272
7.131	Таблица DocumentAPI.Sections	273
7.131.1	Метод Sections:enumerate	273

7.132 Таблица DocumentAPI.Shape	273
7.132.1 Метод Shape:getShapeProperties	273
7.132.2 Метод Shape:setShapeProperties	273
7.133 Таблица DocumentAPI.ShapeProperties	274
7.133.1 Поле ShapeProperties:borderProperties	274
7.133.2 Поле ShapeProperties:fill	274
7.133.3 Поле ShapeProperties:shapeTextLayout	274
7.133.4 Поле ShapeProperties:verticalAlignment	274
7.134 Таблица DocumentAPI.ShapeTextLayout	274
7.135 Таблица DocumentAPI.SizeU	275
7.135.1 Метод SizeU:toString	275
7.136 Таблица DocumentAPI.SortingConditions	275
7.136.1 Метод SortingConditions:add	276
7.136.2 Метод SortingConditions:clear	276
7.137 Таблица DocumentAPI.SortingDirection	276
7.138 Таблица DocumentAPI.Table	277
7.138.1 Метод Table:clearColumnGroups	277
7.138.2 Метод Table:clearRowGroups	277
7.138.3 Метод Table:createFiltersRange	278
7.138.4 Метод Table:duplicate	278
7.138.5 Метод Table:find	279
7.138.6 Метод Table:freeze	279
7.138.7 Метод Table:getCell	279
7.138.8 Метод Table:getCellRange	280
7.138.9 Метод Table:getCharts	280
7.138.10 Метод Table:getColumnsCount	281
7.138.11 Метод Table:getColumnWidth	281
7.138.12 Метод Table:getFiltersRange	281
7.138.13 Метод Table:getFrozenRange	282
7.138.14 Метод Table:getImages	282
7.138.15 Метод Table:getLastNonEmptyCellInColumn	282
7.138.16 Метод Table:getLastNonEmptyCellInRow	283
7.138.17 Метод Table:getMediaObjects	283
7.138.18 Метод Table:getName	284

7.138.19 Метод Table:getNameExpressions	284
7.138.20 Метод Table:getPrintAreas	284
7.138.21 Метод Table:getProtectionProperties	284
7.138.22 Метод Table:getRowHeight	284
7.138.23 Метод Table:getRowCount	285
7.138.24 Метод Table:getShowZeroValue	285
7.138.25 Метод Table:getUsedRange	285
7.138.26 Метод Table:groupColumns	286
7.138.27 Метод Table:groupRows	286
7.138.28 Метод Table:insertColumnAfter	286
7.138.29 Метод Table:insertColumnBefore	287
7.138.30 Метод Table:insertRowAfter	288
7.138.31 Метод Table:insertRowBefore	288
7.138.32 Метод Table:isColumnVisible	289
7.138.33 Метод Table:isProtected	289
7.138.34 Метод Table:isRowVisible	290
7.138.35 Метод Table:isVisible	290
7.138.36 Метод Table:moveTo	290
7.138.37 Метод Table:remove	291
7.138.38 Метод Table:removeColumn	291
7.138.39 Метод Table:removeProtection	291
7.138.40 Метод Table:removeRow	292
7.138.41 Метод Table:removeVisibleColumns	292
7.138.42 Метод Table:removeVisibleRows	292
7.138.43 Метод Table:setColumnsVisible	292
7.138.44 Метод Table:setColumnWidth	293
7.138.45 Метод Table:setName	293
7.138.46 Метод Table:setPrintArea	294
7.138.47 Метод Table:setPrintAreas	294
7.138.48 Метод Table:setProtection	295
7.138.49 Метод Table:setRowHeight	296
7.138.50 Метод Table:setRowsVisible	296
7.138.51 Метод Table:setShowZeroValue	297
7.138.52 Метод Table:setVisible	297

7.138.53	Метод Table:ungroupColumns	297
7.138.54	Метод Table:ungroupRows	298
7.138.55	Метод Table:___eq	298
7.139	Таблица DocumentAPI.TableFilters	298
7.139.1	Метод TableFilters:clear	298
7.139.2	Метод TableFilters:erase	299
7.139.3	Метод TableFilters:setFilter	299
7.140	Таблица DocumentAPI.TableProtectionProperties	300
7.141	Таблица DocumentAPI.TableRangeInfo	301
7.142	Таблица DocumentAPI.TableSearchSettings	301
7.142.1	Таблица TableSearchSettings.MatchBehaviour	302
7.142.2	Таблица TableSearchSettings.SearchProperty	303
7.143	Таблица DocumentAPI.TextAnchoredPosition	303
7.143.1	Метод TextAnchoredPosition:___eq	304
7.144	Таблица DocumentAPI.TextLayout	304
7.145	Таблица DocumentAPI.TextOrientation	305
7.145.1	Метод TextOrientation:getAngle	305
7.145.2	Метод TextOrientation:isStackedChars	305
7.145.3	Метод TextOrientation:___eq	305
7.146	Таблица DocumentAPI.TextProperties	306
7.147	Таблица DocumentAPI.TextUnit	308
7.148	Таблица DocumentAPI.TextWrapType	309
7.149	Таблица DocumentAPI.ThemeColorID	309
7.150	Таблица DocumentAPI.TimePatterns	310
7.151	Таблица DocumentAPI.TrackedChange	310
7.151.1	Метод TrackedChange:getInfo	311
7.151.2	Метод TrackedChange:getRange	311
7.151.3	Метод TrackedChange:getType	312
7.152	Таблица DocumentAPI.TrackedChangeInfo	312
7.152.1	Метод TrackedChangeInfo:___eq	312
7.153	Таблица DocumentAPI.TrackedChangeType	313
7.154	Таблица DocumentAPI.ValueFieldsOrientation	313
7.155	Таблица DocumentAPI.ValuesTableFilter	313
7.155.1	Метод ValuesTableFilter:add	314

7.155.2	Метод ValuesTableFilter:clear	314
7.156	Таблица DocumentAPI.VectorString	314
7.156.1	Метод VectorString:back	314
7.156.2	Метод VectorString:clear	315
7.156.3	Метод VectorString:empty	315
7.156.4	Метод VectorString:front	315
7.156.5	Метод VectorString:max_size	315
7.156.6	Метод VectorString:pop_back	316
7.156.7	Метод VectorString:push_back	316
7.156.8	Метод VectorString:size	316
7.156.9	Метод VectorString: __getitem	316
7.156.10	Метод VectorString: __setitem	317
7.157	Таблица DocumentAPI.VectorUInt	317
7.157.1	Метод VectorUInt:back	317
7.157.2	Метод VectorUInt:clear	317
7.157.3	Метод VectorUInt:empty	318
7.157.4	Метод VectorUInt:front	318
7.157.5	Метод VectorUInt:max_size	318
7.157.6	Метод VectorUInt:pop_back	318
7.157.7	Метод VectorUInt:push_back	318
7.157.8	Метод VectorUInt:size	319
7.157.9	Метод VectorUInt: __getitem	319
7.157.10	Метод VectorUInt: __setitem	319
7.158	Таблица DocumentAPI.VerticalAlignment	319
7.159	Таблица DocumentAPI.VerticalAnchorAlignment	320
7.160	Таблица DocumentAPI.VerticalRelativeTo	321
7.161	Таблица DocumentAPI.VerticalTextAnchoredPosition	321
7.161.1	Метод VerticalTextAnchoredPosition: __eq	322
7.162	Таблица DocumentAPI.WorksheetPrinterFitType	323
8	Справочник функций DocumentAPI	324
8.1	Функция DocumentAPI.createSearch	324
8.2	Функция DocumentAPI.createScripting	324
9	Справочник таблиц EditorAPI	325
9.1	Таблица EditorAPI.SelectionDirection	325

МойОфис

9.2	Таблица EditorAPI.SelectionMode	325
9.3	Таблица EditorAPI.TableSelectionUnit	326
9.4	Таблица EditorAPI.TextSelectionUnit	326
10	Справочник функций EditorAPI	327
10.1	Функция EditorAPI.changeSelection	327
10.2	Функция EditorAPI.getActiveWorksheet	327
10.3	Функция EditorAPI.getSelection	328
10.4	Функция EditorAPI.isPrinterAvailable	328
10.5	Функция EditorAPI.messageBox	329
10.6	Функция EditorAPI.printDocument	329
10.7	Функция EditorAPI.setActiveWorksheet	330
10.8	Функция EditorAPI.setSelection	330
10.9	Функция EditorAPI.showPrintDialog	330
11	Справочник методов EventsAPI	332
11.1	Метод EventsAPI.subscribeWorkbookBeforeClose	332
11.2	Метод EventsAPI.subscribeWorkbookBeforeSave	332
11.3	Метод EventsAPI.subscribeWorkbookOpen	333
11.4	Метод EventsAPI.subscribeWorksheetActivate	333
11.5	Метод EventsAPI.subscribeWorksheetChange	333
11.6	Метод EventsAPI.subscribeWorksheetDeactivate	334
11.7	Метод EventsAPI.subscribeWorksheetSelectionChange	334
12	Функции для работы со строками в формате Юникод (UTF-8)	336
12.1	Функция utf8.char	336
12.2	Функция utf8.charpattern	336
12.3	Функция utf8.codepoint	337
12.4	Функция utf8.codes	337
12.5	Функция utf8.compare	337
12.6	Функция utf8.isalpha	338
12.7	Функция utf8.isdigit	339
12.8	Функция utf8.islower	339
12.9	Функция utf8.isupper	339
12.10	Функция utf8.len	340
12.11	Функция utf8.lower	340
12.12	Функция utf8.next	341

12.13	Функция utf8.offset	341
12.14	Функция utf8.substr	341
12.15	Функция utf8.upper	342
13	Функции для работы с регулярными выражениями	343
13.1	Функция Re.create	343
13.2	Функция Re.match	343
13.2.1	Флаги, используемые в Re.match	343
13.3	Функция Re.replace	345
13.4	Функция Re.search	345
13.5	Флаги, используемые для замены	346
14	Функции для работы с датой и временем	348
14.1	Функция os.clock	349
14.2	Функция os.date	349
14.3	Функция os.difftime	350
14.4	Функция os.time	351
15	Класс Matches	352
15.1	Метод getFirst	352
15.2	Метод getLength	352
15.3	Метод getSize	352
15.4	Метод getString	352
15.5	Метод _tostring	353

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

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

Таблица 1 - Сокращения и расшифровки

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

1 Общие сведения

1.1 Назначение

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

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

Данные приложения входят в состав следующих продуктов:

- «МойОфис Стандартный 3»;
- «МойОфис Профессиональный 2»;
- «МойОфис Профессиональный 3»;
- «МойОфис Схема»;
- «МойОфис Образование»;
- «МойОфис Стандартный. Домашняя версия»;
- «МойОфис для дома».

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

1.2 Макрокоманды ПО МойОфис

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

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

Справочное руководство по языку программирования Lua опубликовано по ссылкам:

<https://lua.org.ru> (ru), <https://devdocs.io> (en).

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

Структура данных, представляющая текущий открытый документ в ПО МойОфис, в терминах языка программирования Lua является таблицей [DocumentAPI.Document](#) со своим набором методов. Иные структуры данных для работы с отдельными ячейками, областями, диаграммами, свойствами текста и т. д. также являются таблицами с необходимым набором полей и методов.

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

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

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

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

1.3 Перечень эксплуатационной документации

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

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

2 Работа с макрокомандами

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

2.1 Редактор макрокоманд

2.1.1 Окно редактора макрокоманд

Для работы с макрокомандами используется редактор макрокоманд. Чтобы открыть окно редактора, в приложении «МойОфис Текст» выберите пункт командного меню **Инструменты > Редактор макрокоманд** или в приложении «МойОфис Таблица» выберите пункт командного меню **Инструменты > Макрокоманды > Редактор макрокоманд**.

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

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

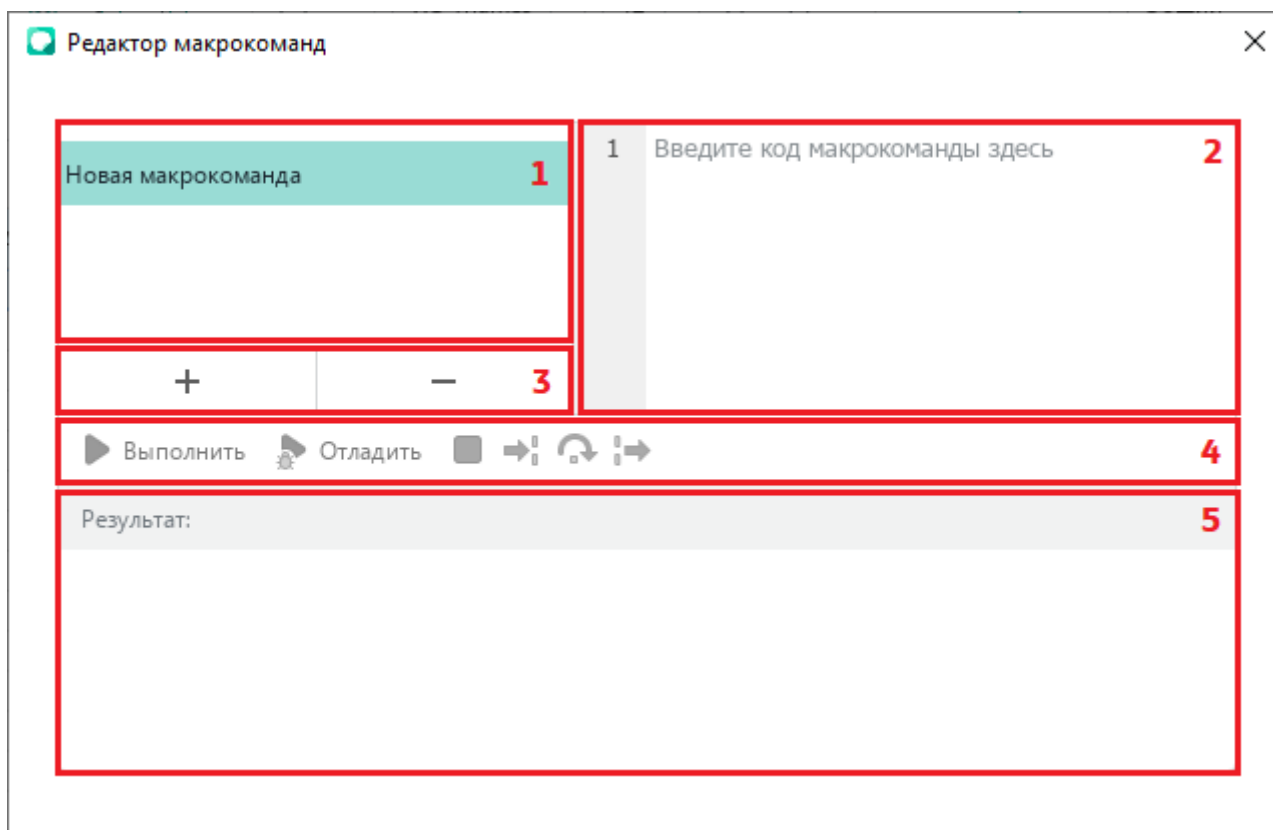


Рисунок 1 – Окно редактора макроккоманд

2.1.2 Создание макроккоманд

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

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

2.1.3 Выполнение макроккоманд

Чтобы выполнить макроккоманду, выберите ее в перечне макроккоманд и нажмите кнопку **▶ Выполнить** (см. Рисунок 1).

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

2.1.4 Редактирование макроккоманд




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

2.1.5 Удаление макрокоманд

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

2.1.6 Отладка макрокоманд

Для отладки макрокоманды выполните следующие действия:

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

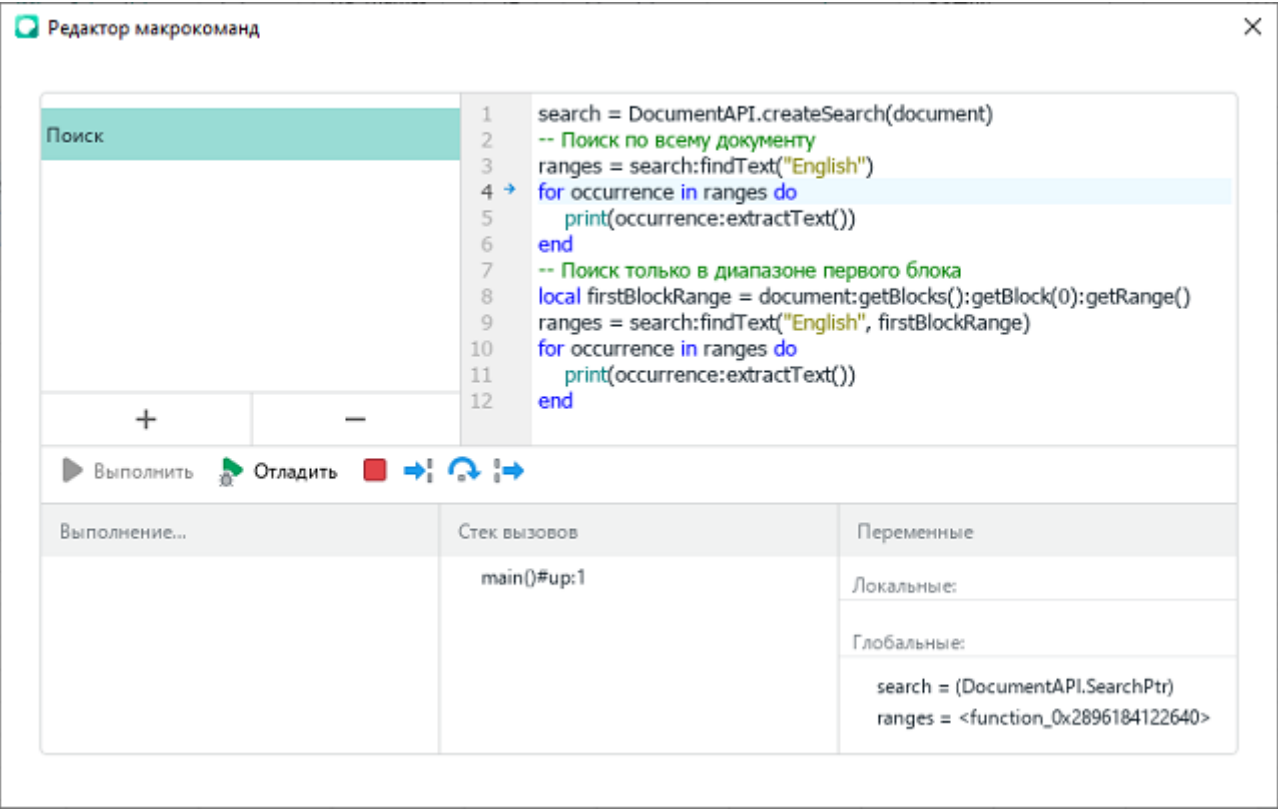






Рисунок 2 – Окно редактора макрокоманд в начале отладки


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

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

-  – для выполнения одного шага отладки или захода в тело функции, если таковая есть в текущей позиции отладки;
-  – для выполнения одного шага отладки без захода в тело функции;
-  – для продолжения выполнения макрокоманды до момента выхода из функции, в которой отладчик находится в текущей позиции.

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

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

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

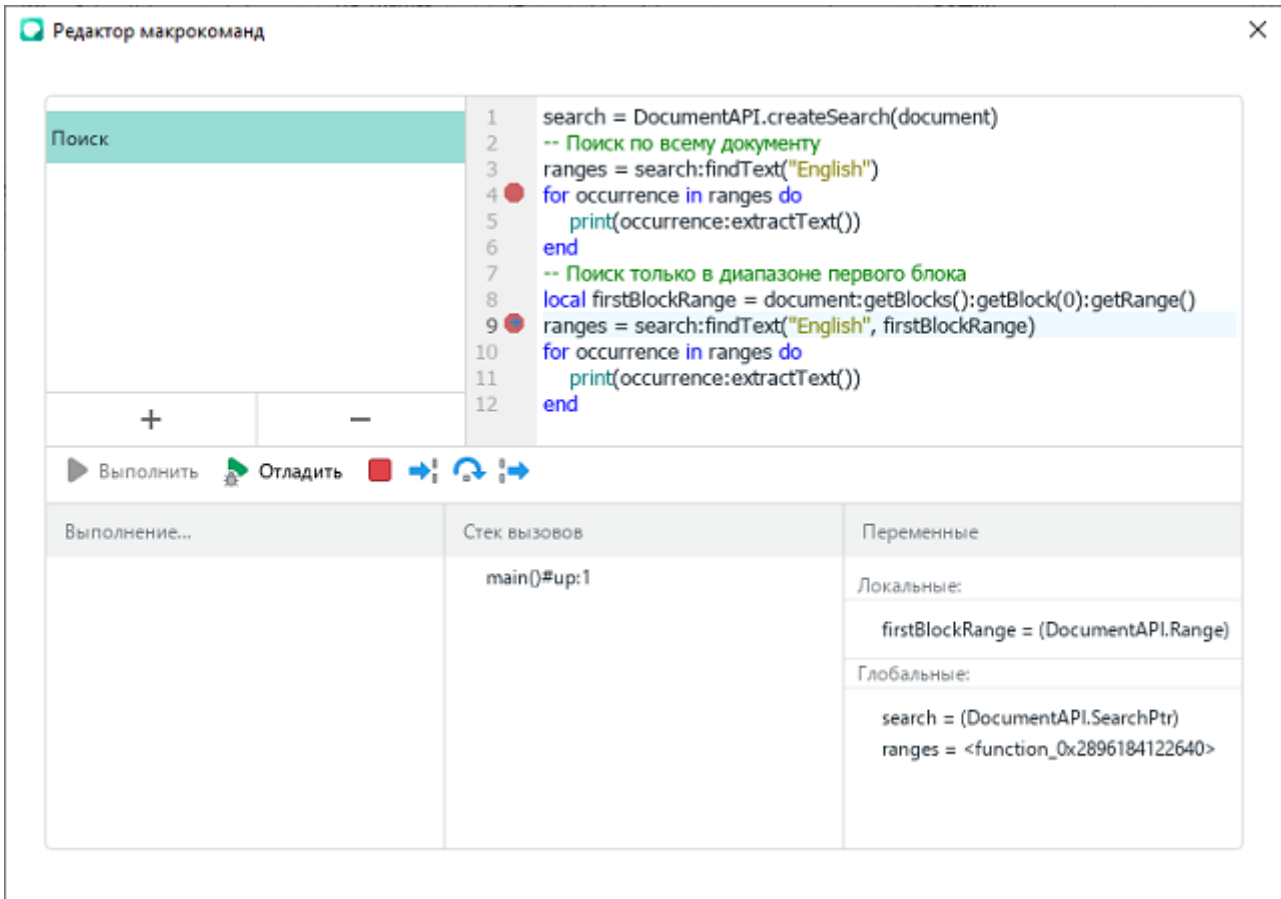


Рисунок 3 – Окно редактора макрокоманд в процессе отладки макрокоманд

Отладка завершается при достижении конца макрокоманды.

2.2 Пример подготовки и запуска макроккоманды


2.2.1 Редактирование и запуск макроккоманды в текстовом документе

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

Чтобы создать и запустить макроккоманду, необходимо выполнить следующие действия:

1. Запустить приложение «МойОфис Текст» и открыть редактор макроккоманд. Для этого в командном меню выбрать пункт **Инструменты > Макроккоманды > Редактор макроккоманд**.
2. Нажать кнопку **+** для создания новой макроккоманды. Указать имя макроккоманды. По умолчанию новой макроккоманде присваивается имя «Без имени».
3. Ввести текст макроккоманды:

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

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

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

2.2.2 Описание примера работы макроккоманды

Ниже приведено описание макроккоманды, текст которой приведен в разделе [Редактирование и запуск макроккоманды в текстовом документе](#).

С помощью последовательности вызовов устанавливается курсор в начало документа:

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

Таблица [DocumentAPI.Document](#) представляет текущий открытый текстовый документ.

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

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

Следующая последовательность вызовов настраивает форматирование для документа:

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

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

Таблица [DocumentAPI.TextProperties](#) позволяет управлять такими характеристиками как наименование и размер шрифта, цвет, начертание и т.п.

В данном примере устанавливаются две настройки форматирования:

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

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

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

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

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

2.3 Преобразование макрокоманд на языке программирования VBA

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

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

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

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

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

3 Объектная модель МойОфис

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

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

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

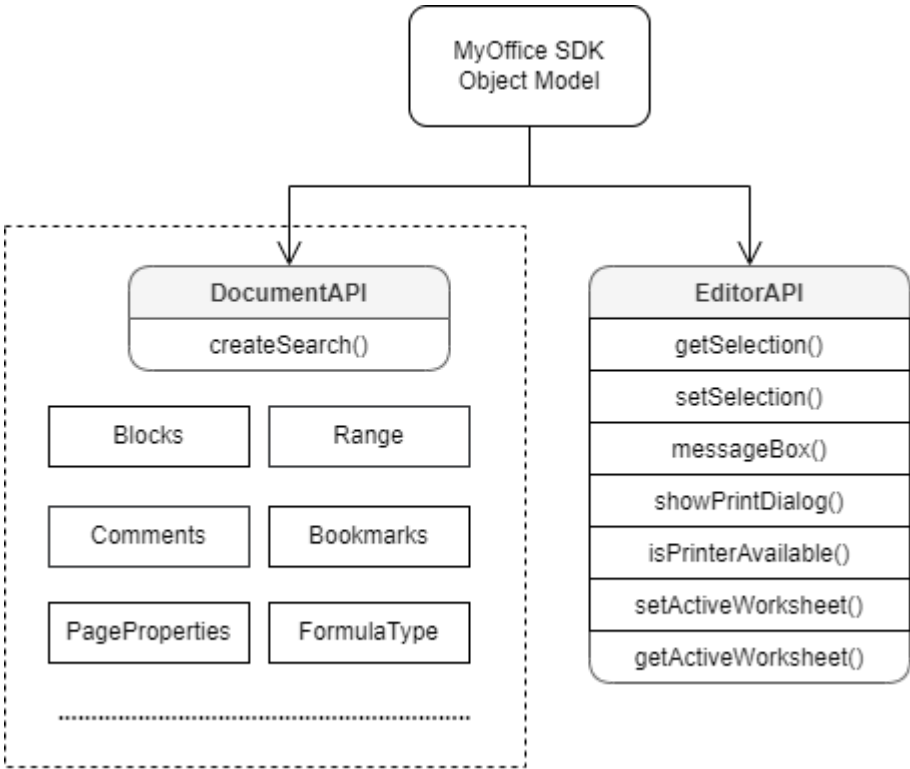


Рисунок 4 – Объектная модель МойОфис SDK.

4 Работа с документами

4.1 Работа с текстовым документом

4.1.1 Разделы (секции) документа

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

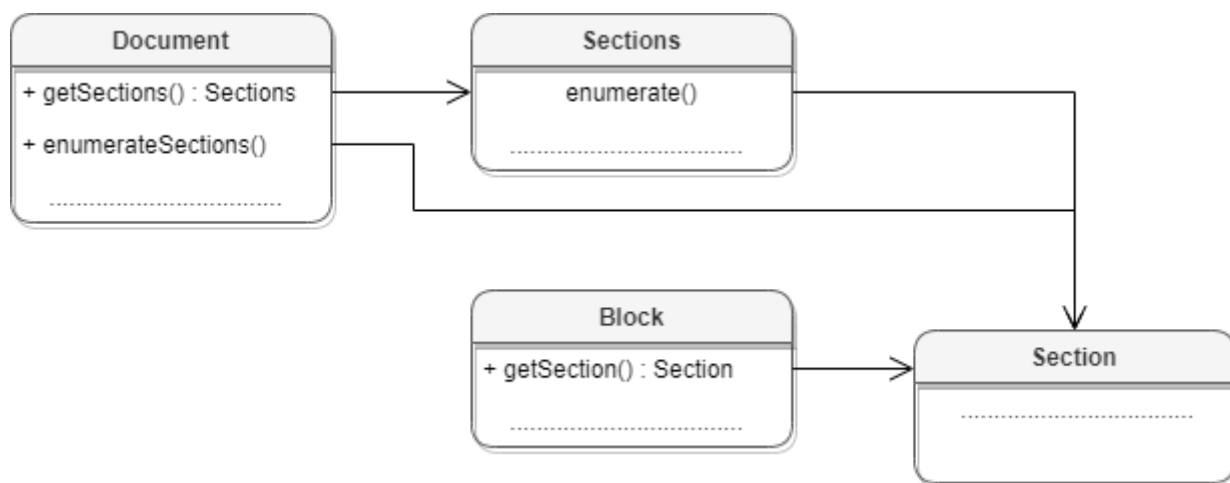


Рисунок 5 – Объектная модель таблиц для работы с секциями

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

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

- получение таблицы [DocumentAPI.Sections](#) с помощью вызова [document:getSections\(\)](#);
- перечисление всех доступных секций [DocumentAPI.Section](#) с помощью вызова [document:enumerateSections\(\)](#);
- получение секции [DocumentAPI.Section](#) вызовом метода [Block.getSection\(\)](#) для блока, который входит в секцию.

Примеры

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

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

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

4.1.1.1 Работа с колонтитулами раздела

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

Пример

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

4.1.1.2 Управление ориентацией и свойствами страниц раздела

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

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

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

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

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

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

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

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

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

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

4.1.2 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены являются листы документа.

Доступ к объектам [Table](#) осуществляется из [Blocks](#) (см. Рисунок 6).

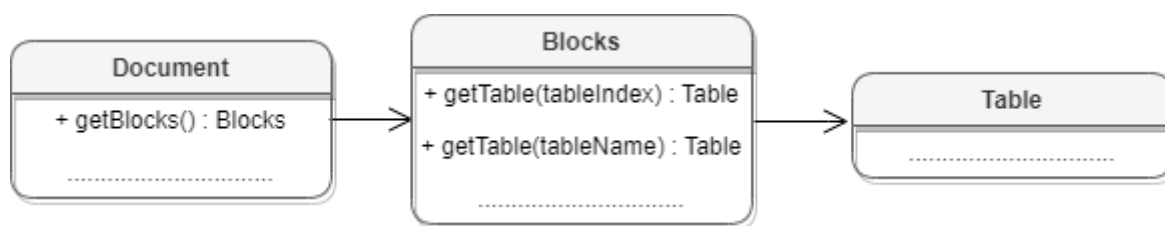


Рисунок 6 – Объектная модель для работы с таблицами

Для работы с таблицами доступны следующие операции:

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

Ниже приведены примеры работы с таблицами в текстовых документах:

Перечисление таблиц документа

Для перечисления таблиц текстового документа используется метод [Blocks::getTablesEnumerator\(\)](#).

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

Получение таблицы текстового документа

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

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

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

Вставка таблицы в текстовый документ

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

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

Вставка текста в таблицу текстового документа

Для вставки текста в таблицу текстового документа используется метод [Position::insertText\(\)](#).

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

Переименование таблицы

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

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

Удаление таблицы

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

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

4.1.3 Работа с закладками

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

Доступны следующие операции с закладками:

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

- получение текстового содержимого закладки;
- вставка таблицы в закладку.

Вставка закладки в указанное местоположение

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

Удаление закладки с заданным именем

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

Поиск закладки по имени

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

Замена текстового содержимого закладки

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

Вставка текста в закладку

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

Удаление содержимого закладки

```
sig_rng:removeContent()
```

Получение текстового содержимого закладки

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

Вставка таблицы в закладку

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

4.1.4 Рецензирование документов

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

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

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

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

Пример

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

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

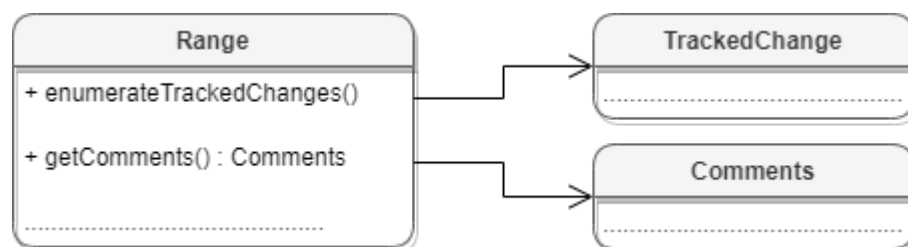


Рисунок 7 – Инструменты рецензирования документа

4.1.5 Работа с графическими объектами в текстовом документе

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

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

- Перечисление графических объектов, находящихся в документе, определение их типа и геометрических размеров.

- Вставка изображений в текстовый документ. Место вставки определяется типом [Position](#).
- Перемещение графических объектов, изменение их размеров и масштаба.

Перечисление графических объектов в текстовом документе.

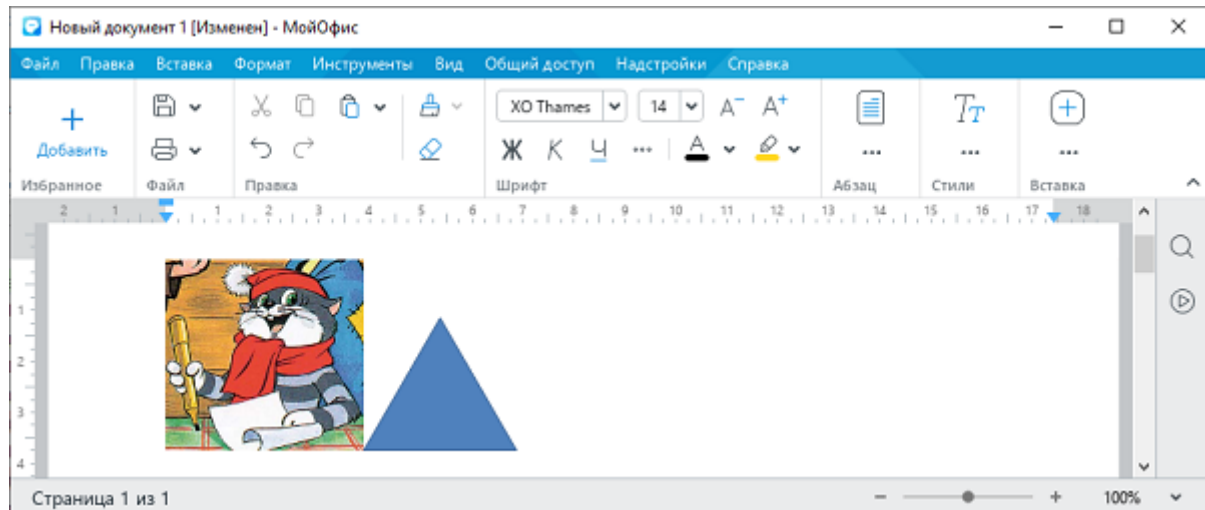


Рисунок 8 – Графические объекты в текстовом документе

Вариант 1: перечисление графических объектов в текстовом документе

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

Вывод

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

Вариант 2: перечисление изображений в текстовом документе

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

Вывод

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

Перечисление графических объектов в таблицах текстового документа

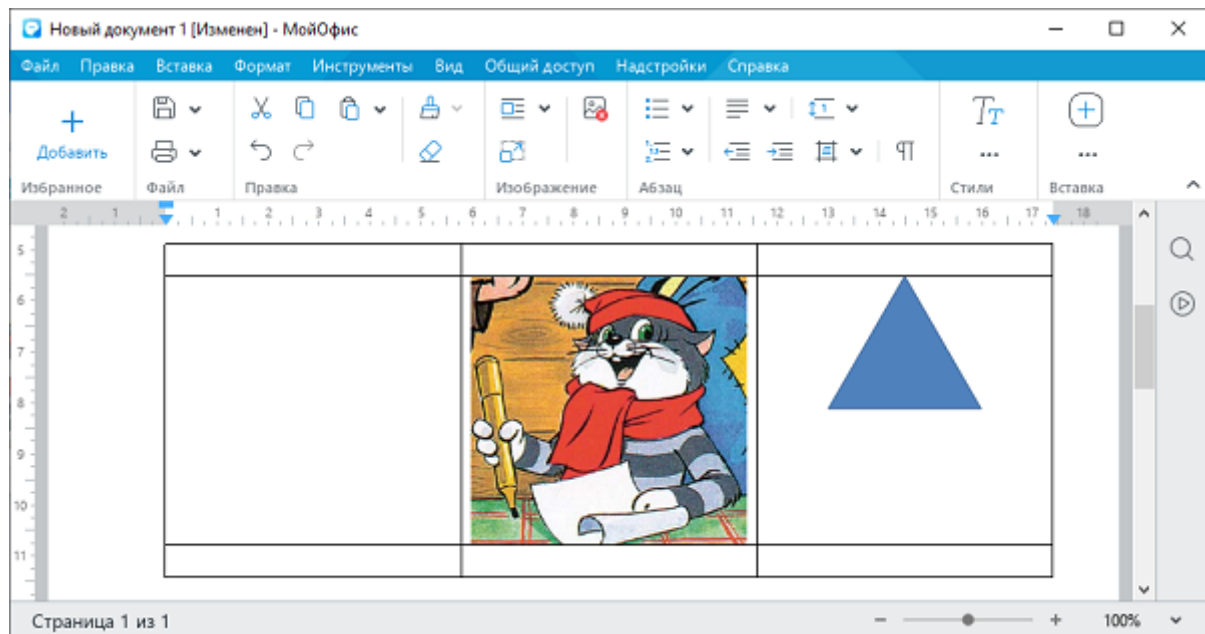


Рисунок 9 – Графические объекты в таблице текстового документа

Вариант 1: перечисление графических объектов в таблице текстового документа

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

Вывод

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

Вариант 2: перечисление изображений в таблице текстового документа

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

Вывод

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

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

Вставка изображения в текстовый документ

Вариант 1: вставка изображения в позицию диапазона текстового документа

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

Вариант 2: вставка изображения в ячейку таблицы текстового документа

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

4.1.6 Работа с элементами управления

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

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

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

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

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

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

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

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

```
checkBox.setValue(not(checkBox.getValue()))
```

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

4.2 Работа с табличным документом

4.2.1 Работа с текстом в табличном документе

Вставка текста в табличный документ

Для вставки текста в ячейку табличного документа используется метод [Position::insertText\(\)](#).

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

4.2.2 Копирование ячеек в табличном документе

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

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

```
local leftTopCellPosition = DocumentAPI.CellPosition(0, 0)  
local rightBottomCellPosition = DocumentAPI.CellPosition(1, 1)  
local srcCellRangePosition = DocumentAPI.CellRangePosition(leftTopCellPosition,  
rightBottomCellPosition)  
  
local strTargetRange = "A1:B2"
```



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

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

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

4.2.3 Диаграммы

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

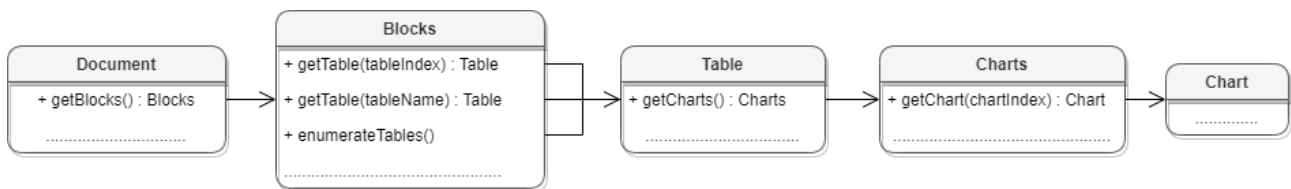


Рисунок 10 – Объектная модель таблиц для работы с диаграммами

Доступ к списку диаграмм производится через таблицу [DocumentAPI.Table](#), соответствующую листу табличного документа.

Пример

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



Создание и удаление диаграмм в текущей версии не поддерживается.

4.2.4 Работа с формулами

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

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

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

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

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

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

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

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

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

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

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

4.2.5 Проверка данных

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

Проверка данных по списку значений

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

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

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

Проверка данных в формате Дата

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

1. Создайте экземпляр таблицы [DataValidation](#).
2. Позовите метод [DataValidation:setType\(\)](#) с параметром [DataValidationType_Date](#) для создания проверки дат.
3. Используйте метод [DataValidation:setOperator\(\)](#), чтобы задать оператор сравнения.

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

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

Отображение сообщений об ошибках

Добавьте визуальное предупреждение о вводе недопустимых значений:

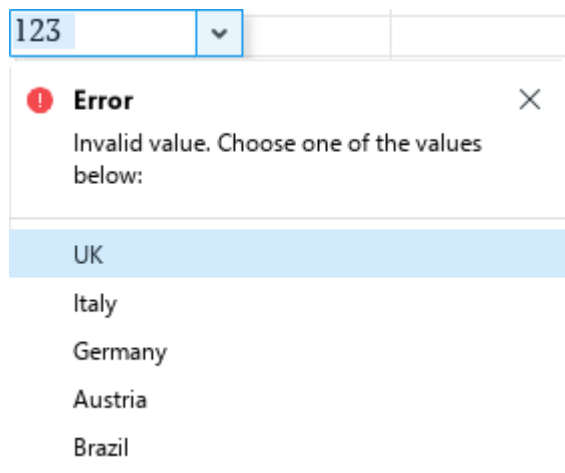


Рисунок 12 – Ошибка проверки данных

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

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

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

Отображение подсказок

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

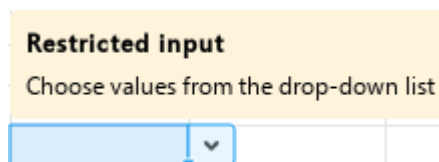


Рисунок 14 – Подсказка при вводе значения в ячейку с проверкой

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

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

Обработка результатов проверки

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

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

Получение настроек проверок

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

Удаление проверок

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

4.2.6 Работа с графическими объектами в табличном документе

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

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

- Перечисление графических объектов, находящихся в документе, определение их типа и геометрических размеров.
- Вставка изображений в текстовый документ. Место вставки определяется типом [Position](#).
- Перемещение графических объектов, изменение их размеров и масштаба.

Перечисление графических объектов в табличном документе

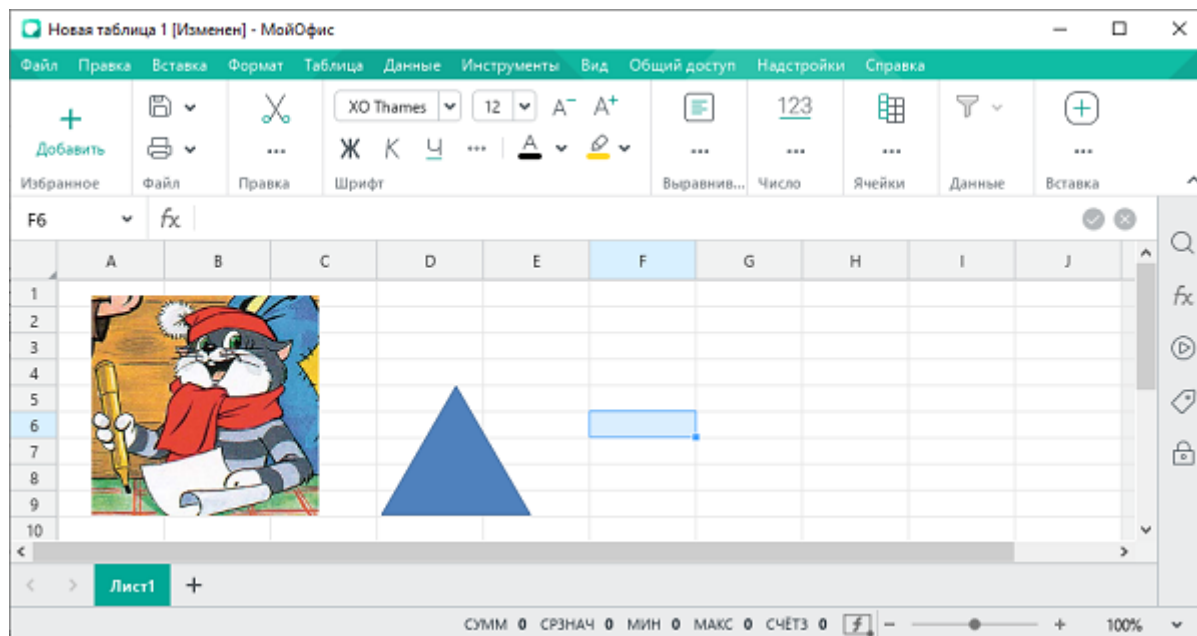


Рисунок 15 – Графические объекты в табличном документе

Вариант 1: перечисление графических объектов в табличном документе

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

Вывод

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

Вариант 2: перечисление изображений в табличном документе

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

Вывод

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

Вставка изображения в табличный документ

В текущей версии не поддерживается.

4.2.7 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам [Table](#) осуществляется из [Blocks](#) (см. Рисунок 16).

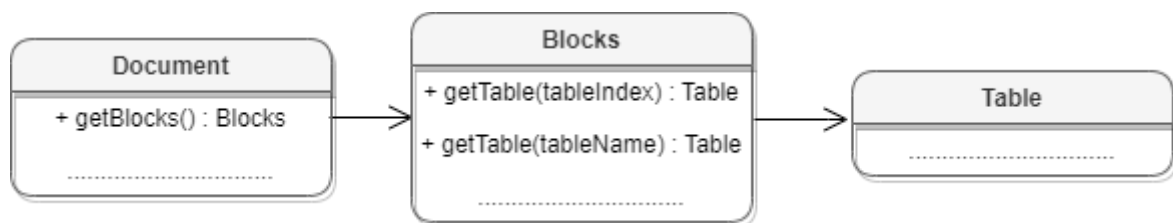


Рисунок 16 – Объектная модель для работы с таблицами

Получение листа табличного документа

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

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

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

Перечисление страниц табличного документа

Для перечисления листов табличного документа используется метод [Blocks::getTablesEnumerator\(\)](#).

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


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

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

Вставка страницы в табличный документ

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

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

Активация страницы, получение активной страницы

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

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

Переименование страницы

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

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

Скрытие и отображение страниц табличного документа

Для скрытия / отображения листа документа используется метод [Table::setVisible\(\)](#).

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

Копирование страницы

Для создания копии страницы используется метод [Table::duplicate\(\)](#).

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

Удаление страницы

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

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

4.2.8 Работа со сводными таблицами

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

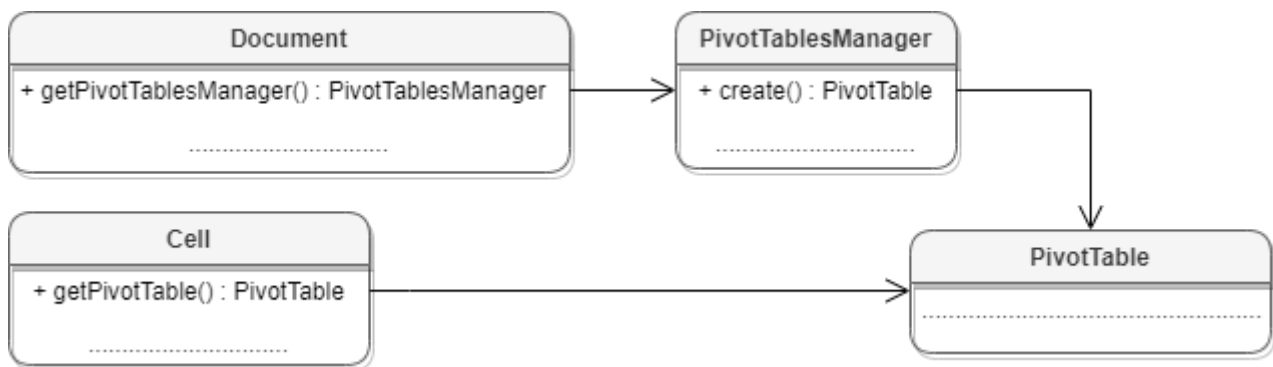


Рисунок 17 – Сводные таблицы

4.2.8.1 Создание сводной таблицы

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

Пример

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

```
local pivotCaptions = pivotTable:getPivotTableCaptions()  
pivotCaptions.grandTotalCaption = "Итого"  
pivotCaptions.rowHeaderCaption = "Продукт"  
pivotCaptions.columnHeaderCaption = "Страна"  
tableEditor:setCaptions(pivotCaptions)  
  
tableEditor:apply()
```

4.2.8.2 Получение диапазона исходных данных сводной таблицы

Для получения диапазона исходных данных сводной таблицы используется метод [PivotTable:getSourceRange\(\)](#).

Пример

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

4.2.8.3 Получение диапазона размещения сводной таблицы

Для получения диапазона размещения сводной таблицы используется метод [PivotTable:getPivotRange\(\)](#).

Пример

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

4.2.8.4 Получение флагов отображения общих итогов для строк и колонок

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

Пример

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

4.2.8.5 Получение заголовков сводной таблицы

Для получения заголовков сводной таблицы используется метод [PivotTable:getPivotTableCaptions\(\)](#).

Пример

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

4.2.8.6 Получение и применение фильтра для сводной таблицы

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

Пример

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

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

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

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

4.2.8.7 Получение полей из области фильтров

Для получения полей из области фильтров используется метод [PivotTable:getPageFields\(\)](#).

Пример

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

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

4.2.8.8 Получение полей из области значений

Для получения полей из области значений используется метод [PivotTable:getValueFields\(\)](#).

Пример

```
-- Получение полей из области значений
valueFields = pivotTable:getValueFields()
-- Перебираем все поля из области значений
for fieldIdx = 0, valueFields:size() - 1 do
    print(valueFields[fieldIdx].baseFieldName)
    print(valueFields[fieldIdx].valueFieldName)
    print(valueFields[fieldIdx].cellNumberFormat)
    print(valueFields[fieldIdx].totalFunction)
end
```

4.2.8.9 Получение полей из области строк

Для получения полей из области строк используется метод [PivotTable:getRowFields\(\)](#).

Пример

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

```
print(fieldProperties.subtotalAlias)
end
```

4.2.8.10 Получение полей из области колонок

Для получения полей из области колонок используется метод [PivotTable:getColumnFields\(\)](#).

Пример

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

4.2.8.11 Получение настроек отображения сводной таблицы

Для получения настроек отображения сводной таблицы используется метод [PivotTable:getPivotTableLayoutSettings\(\)](#).

Пример

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

4.2.8.12 Обновление сводной таблицы

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

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

4.2.9 Работа с фильтрами

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

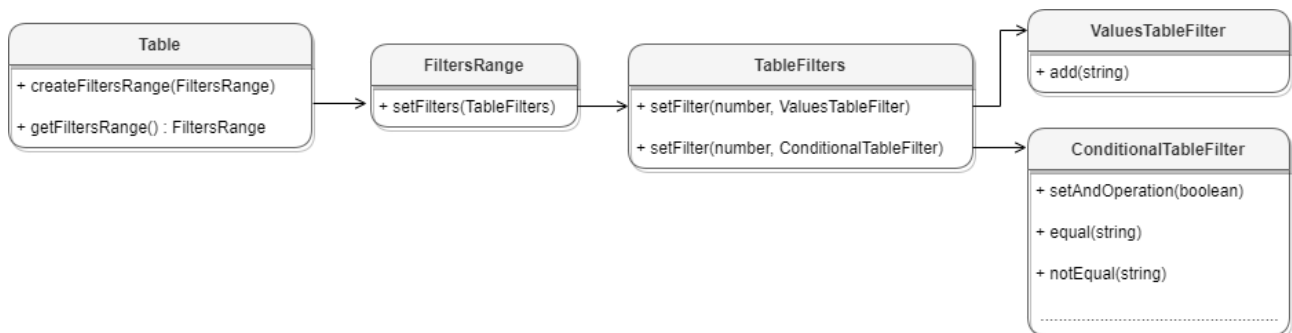


Рисунок 18 – Объектная модель таблиц для работы с фильтрами

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

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

Фильтры помещаются в структуру [TableFilters](#).

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

Пример работы с фильтрами в табличном документе

```
local tbl = EditorAPI.getActiveWorksheet()  
local range = DocumentAPI.CellRangePosition(1, 1, 8, 2)  
local filtersRange = tbl:createFiltersRange(range)  
  
local johnPaulFilter = DocumentAPI.ValuesTableFilter()  
johnPaulFilter:add("John")  
johnPaulFilter:add("Paul")  
  
local songFilter = DocumentAPI.ConditionalTableFilter()  
songFilter:setAndOperation(true)
```

```
songFilter:notEqual("")
songFilter:notBegins("TODO")

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

filtersRange:setFilters(tableFilters)
```

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

	A	B	C
1			
2		Lead vocal	Song
3		John	Help!
4		Paul	Drive My Car
6		John	In My Life
8		Paul	TODO: new title for 'Scrambled Eggs'
9		John	

Рисунок 19 – Применение фильтров

4.2.10 Обработка событий табличного документа

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



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

Таблица 2 - События табличного документа

Событие	Метод
Открытие документа	EventsAPI.subscribeWorkbookOpen
Событие перед сохранением документа	EventsAPI.subscribeWorkbookBeforeSave
Событие перед закрытием документа	EventsAPI.subscribeWorkbookBeforeClose
Изменение выделения ячеек	EventsAPI.subscribeWorksheetSelectionChange
Изменение содержимого ячеек	EventsAPI.subscribeWorksheetChange
Активация страницы документа	EventsAPI.subscribeWorksheetActivate
Деактивация страницы документа	EventsAPI.subscribeWorksheetDeactivate

Пример подписки на событие:

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

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

На данный момент существуют следующие ограничения:

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

4.3 Поиск в документе

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

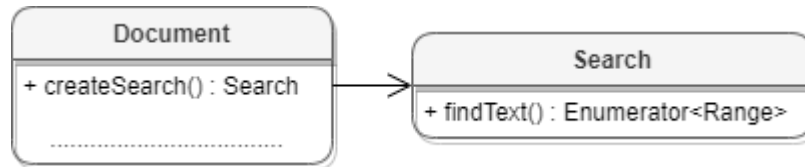


Рисунок 20 – Объектная модель для поиска в документе

Примеры поиска в документе

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

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

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

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

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

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

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

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

-- Поиск в таблице с учетом регистра
local table = document:getBlocks():getTable(0)
ranges = search:findText(text, table, DocumentAPI.CaseSensitive_Yes)

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

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

Пример поиска ячеек в табличном документе

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

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

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

4.4 Работа с макрокомандами

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

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

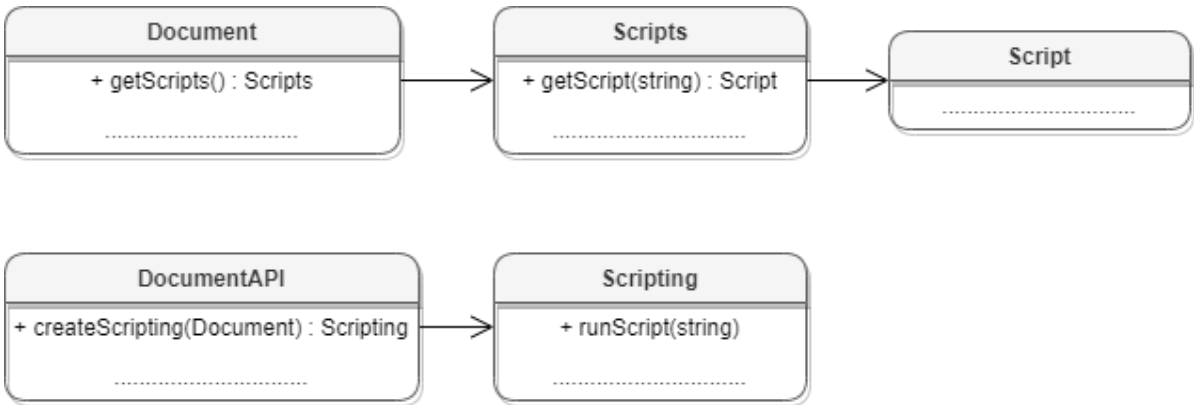


Рисунок 21 – Объектная модель таблиц для работы с макросами

4.5 Работа с именованными диапазонами

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

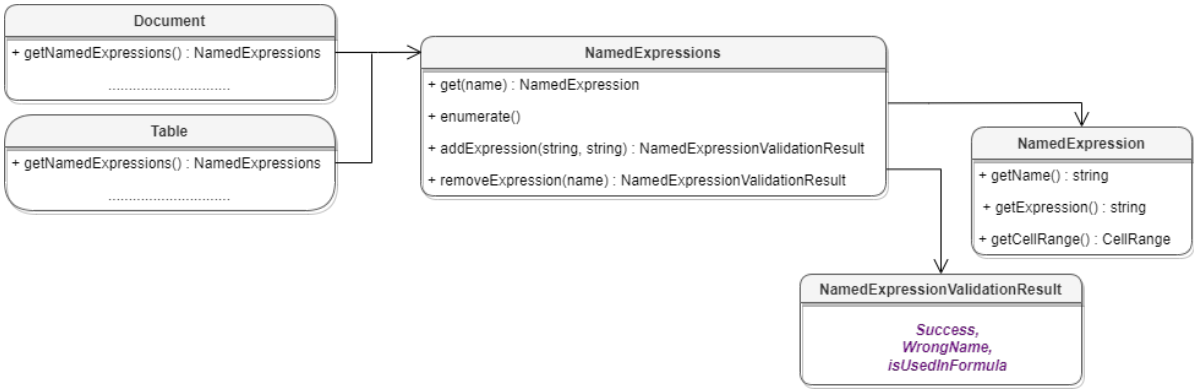


Рисунок 22 – Таблицы для работы с именованными диапазонами

5 Работа со строками и столбцами таблиц

5.1 Группировка строк и колонок таблицы

Следующий набор методов позволяет группировать строки и колонки таблицы:

[Table::groupRows\(\)](#), [Table::ungroupRows\(\)](#), [Table::clearRowGroups\(\)](#),
[Table::groupColumns\(\)](#), [Table::ungroupColumns\(\)](#),
[Table::clearColumnGroups\(\)](#).

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

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

5.2 Управление видимостью строк / колонок

Метод [Table::isRowVisible](#) позволяет определять видимость строки с заданным индексом.

Метод [Table::isColumnVisible](#) позволяет определять видимость столбца с заданным индексом.

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

Пример для текстового и табличного редактора

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

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

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

Пример для табличного редактора

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

```
local tbl = selection:getTable()

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

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

setSelectionVisible(false)
```

6 Работа с ячейками таблиц

6.1 Доступ к ячейкам

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

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

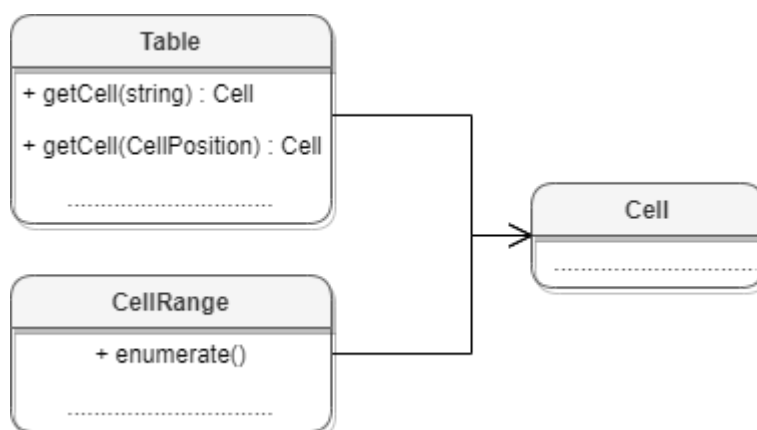


Рисунок 23 – Объектная модель для работы с ячейками таблиц

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

Пример

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

Второй вариант доступа к ячейке - перечисление диапазона ячеек методом [CellRange.enumerate\(\)](#).

Пример

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

Для определения того, входит ли ячейка в указанный диапазон, используется метод [CellRange.containsCell\(\)](#).

Примеры

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

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

local cell11 = table1:getCell("A1")
local cell12 = table1:getCell("C4")
local 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

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

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

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

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

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

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


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

Пример

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

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

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

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

Пример

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

Для получения значения ячейки используется метод [Cell.getFormattedValue\(\)](#).

Пример

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

6.2 Форматирование ячеек

При работе с ячейками таблиц можно использовать следующие варианты форматирования:

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

- форматирование [текста](#), например, цвет текста, начертание;
- задание параметров [границ ячеек](#).

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

Пример установки и получения свойств параграфа ячейки

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

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

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

Пример настроек текста ячейки

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

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

6.3 Форматирование границ ячеек

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

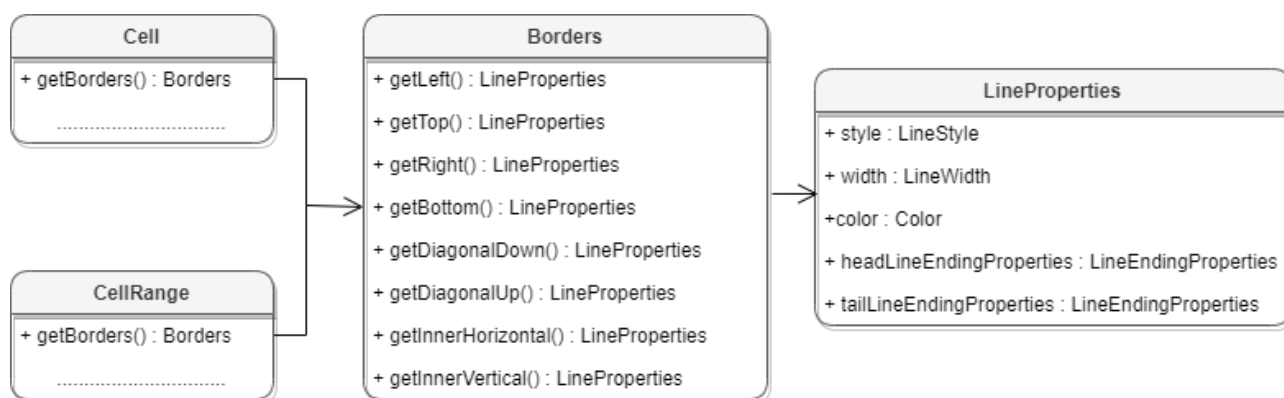


Рисунок 24 – Таблицы для работы с границами ячеек

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

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

Пример настройки границ ячеек

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

--Настроить параметры для рисования линии
local lineProp = DocumentAPI.LineProperties()
lineProp.style = DocumentAPI.LineStyle_Solid
```

```
lineProp.width = 1.5
local lc = DocumentAPI.ColorRGBA(55, 146, 179, 200)
lineProp.color = DocumentAPI.Color(lc)

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

--установка внешних границ
borders:setOuter(lineProp)

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

6.4 Объединение и разделение ячеек таблицы

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

Для объединения нескольких ячеек используйте метод [CellRange.merge\(\)](#).

Пример

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

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 используйте метод [Cell.unmerge\(\)](#).

Пример

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

7 Справочник таблиц DocumentAPI

7.1 Таблица DocumentAPI.AbsoluteFrame

Таблица `DocumentAPI.AbsoluteFrame` описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 25). Предназначена для получения и изменения свойств позиции медиаобъектов. Используется в табличном документе.

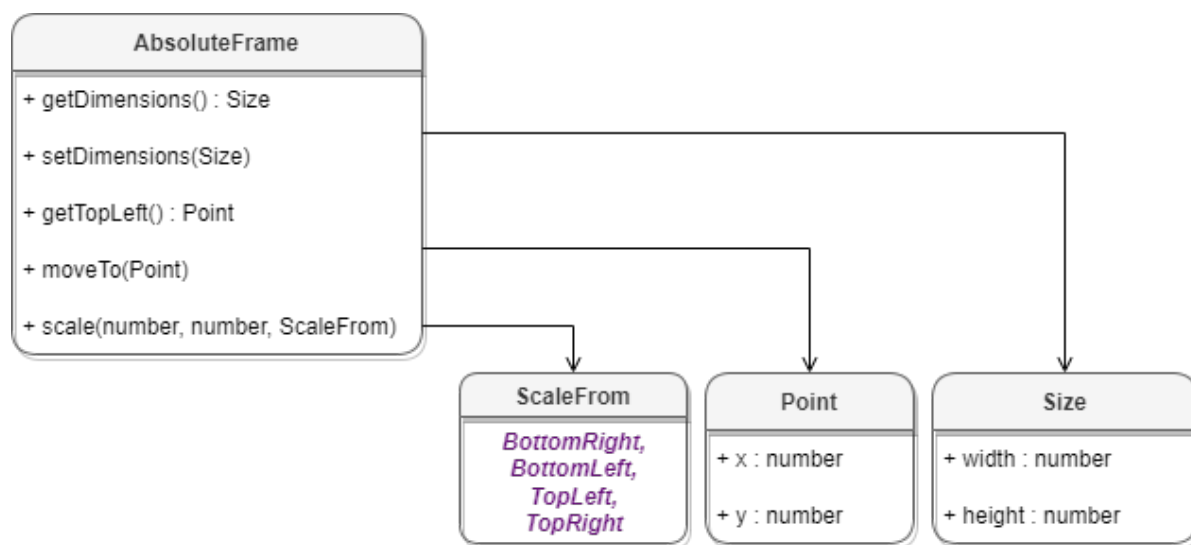


Рисунок 25 – Объектная модель таблицы `DocumentAPI.AbsoluteFrame`

Пример для табличного документа

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

7.1.1 Метод `AbsoluteFrame:getDimensions`

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

Пример

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

7.1.2 Метод `AbsoluteFrame:getTopLeft`

Метод возвращает позицию верхней левой точки медиаобъекта, тип - [DocumentAPI.PointU](#).

Пример

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

7.1.3 Метод `AbsoluteFrame:moveTo`

Метод перемещает объект в заданную позицию, тип аргумента - [DocumentAPI.PointU](#).

Пример

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

7.1.4 Метод `AbsoluteFrame:scale`

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

Вызов

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

Параметры

- `widthScale` – коэффициент масштабирования по горизонтали, тип - числовой;
- `heightScale` – коэффициент масштабирования по вертикали, тип - числовой;
- `scaleFrom` – точка, сохраняющая позицию при масштабировании, тип - [DocumentAPI.ScaleFrom](#).

Пример

```
-- Уменьшение масштаба всех медиаобъектов на 50%
sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
```

```
mediaObject:getFrame():scale(0.5, 0.5, DocumentAPI.ScaleFrom_TopLeft)
end
```

7.1.5 Метод `AbsoluteFrame:setDimensions`

Метод задает размеры (изменяет размер) медиаобъекта.

Вызов

```
setDimensions(size)
```

Параметры

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

Пример

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

7.2 Таблица `DocumentAPI.AccountingCellFormatting`

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

Описание полей таблицы `DocumentAPI.AccountingCellFormatting` представлено в таблице 3.

Таблица 3 – Описание полей таблицы `DocumentAPI.AccountingCellFormatting`

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

Пример

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




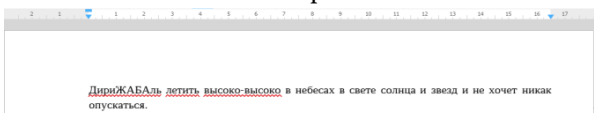
cell:setFormat(DocumentAPI.CellFormat_Accounting)
local accountingCellFormatting = DocumentAPI.AccountingCellFormatting()
accountingCellFormatting.decimalPlaces = 3
accountingCellFormatting.symbol = 'Pyб'

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

7.3 Таблица DocumentAPI.Alignment

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

Таблица 4 – Варианты выравнивания по горизонтали

Наименование константы	Описание
DocumentAPI.Alignment_Default	Выравнивание по умолчанию.
DocumentAPI.Alignment_Left	По левому краю 
DocumentAPI.Alignment_Center	По центру 
DocumentAPI.Alignment_Right	По правому краю 
DocumentAPI.Alignment_Justify	По ширине 

Пример для текстового документа

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


Пример для табличного документа

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

7.4 Таблица DocumentAPI.Block

Таблица `DocumentAPI.Block` является базовой для всех блоков документа. От нее наследуются таблицы [Paragraph](#), [Table](#), [Shape](#), [Field](#) (см. Рисунок 26).

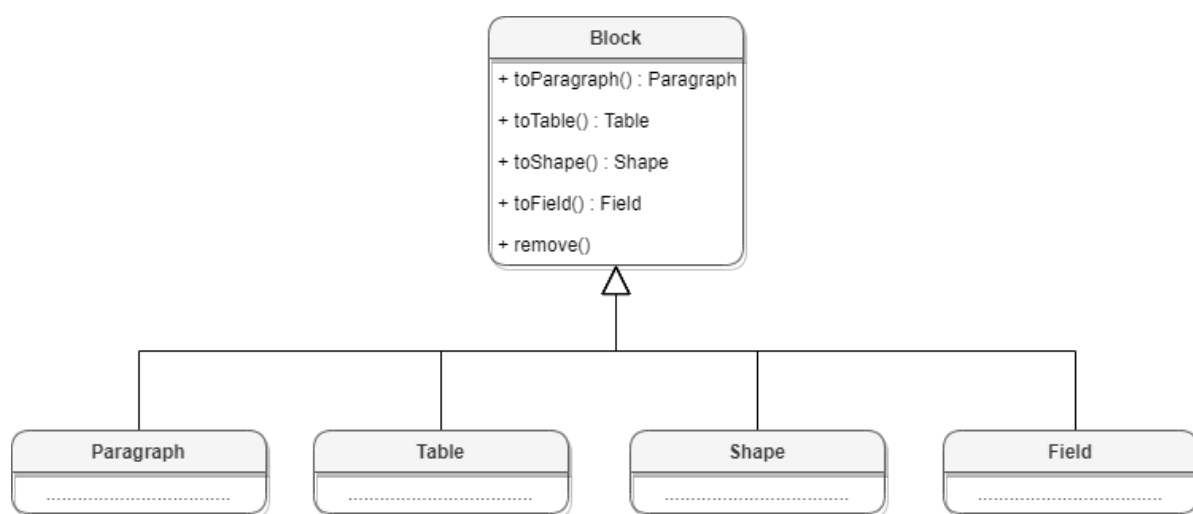


Рисунок 26 – Объектная модель таблицы `DocumentAPI.Block`

7.4.1 Метод `Block:getRange`

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

Пример

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

7.4.2 Метод `Block:getSection`

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

Пример

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

7.4.3 Метод `Block:remove`

Удаляет блок из документа. Текущий экземпляр объекта [DocumentAPI.Block](#) становится недействительным.

Пример

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

7.4.4 Методы `toParagraph`, `toTable`, `toShape`, `toField`

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

Пример

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

7.5 Таблица `DocumentAPI.Blocks`

Таблица `DocumentAPI.Blocks` обеспечивает доступ к блокам [DocumentAPI.Block](#) документа или диапазона документа (см. Рисунок 27). Таблица `DocumentAPI.Blocks` может быть получена вызовом метода [Document:getBlocks](#) или [HeaderFooter:getBlocks](#).

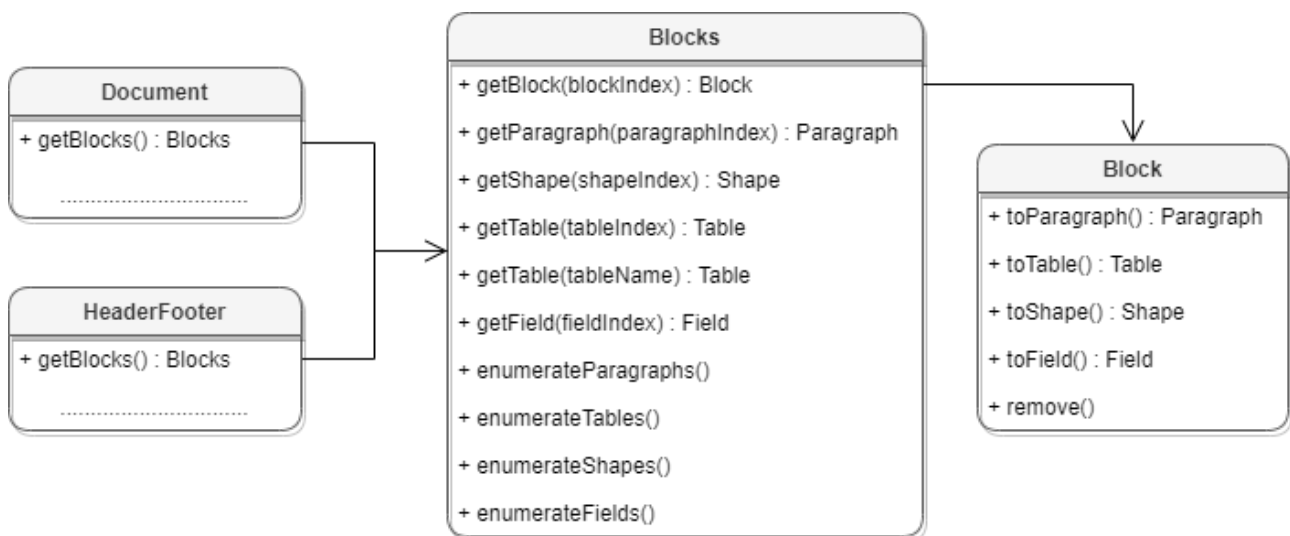


Рисунок 27 – Объектная модель таблицы `DocumentAPI.Blocks`

7.5.1 Метод `Blocks:enumerate`

Позволяет перечислить объекты типа [DocumentAPI.Block](#).

Пример

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

7.5.2 Метод `Blocks:enumerateParagraphs`

Позволяет реализовать перечисление абзацев [DocumentAPI.Paragraph](#).

Пример

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

7.5.3 Метод `Blocks:enumerateTables`

Позволяет перечислить объекты типа [DocumentAPI.Table](#).

Пример

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

7.5.4 Метод `Blocks:getBlock`

Возвращает объект типа [DocumentAPI.Block](#) по заданному индексу. Нумерация индексов начинается с нуля.

Пример

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

7.5.5 Метод `Blocks:getField`

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

Пример

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

7.5.6 Метод `Blocks:getParagraph`

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

Пример

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

7.5.7 Метод `Blocks:getShape`

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

Пример

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

7.5.8 Метод `Blocks:getTable`

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

Пример

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

В качестве параметра метода также можно указать имя таблицы.

Пример

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

7.6 Таблица `DocumentAPI.Bookmarks`

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

7.6.1 Метод `Bookmarks:getBookmarkRange`

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

Пример

```
local bookmarks = document:getBookmarks()  
local bookmarkRange = bookmarks:getBookmarkRange("Bookmark")
```

```
if (bookmarkRange ~= nil) then
    bookmarkRange:replaceText("Lua")
end
```

7.6.2 Метод Bookmarks:removeBookmark

Удаляет закладку по ее названию.

Пример

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

7.7 Таблица DocumentAPI.Borders

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

Таблица 5 – Описание методов таблицы `DocumentAPI.Borders`

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

Метод	Описание
LineProperties getRight()	Получение правой границы ячейки
LineProperties getTop()	Получение верхней границы ячейки
LineProperties getBottom()	Получение нижней границы ячейки
LineProperties getDiagonalDown()	Получение диагональной линии
LineProperties getDiagonalUp()	Получение диагональной линии
LineProperties getInnerHorizontal()	Получение внутренних горизонтальных границ ячейки
LineProperties getInnerVertical()	Получение внутренних вертикальных границ ячейки
bool isEmpty()	Возвращает true, если не установлена ни одна граница ячейки

Пример

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

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

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

cell:setBorders(borders)
```

7.8 Таблица DocumentAPI.CalculationMode

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

Таблица 6 – Режимы пересчета формул

Поле	Описание
DocumentAPI.CalculationMode_Auto	Формулы пересчитываются автоматически при изменении данных.
DocumentAPI.CalculationMode_Manual	Формулы пересчитываются вручную.

7.9 Таблица DocumentAPI.CaseSensitive

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

Таблица 7 – Описание полей таблицы DocumentAPI.CaseSensitive

Поле	Описание
DocumentAPI.CaseSensitive_Yes	Поиск с учетом регистра
DocumentAPI.CaseSensitive_No	Поиск без учета регистра

Пример

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

7.10 Таблица DocumentAPI.Cell

Таблица DocumentAPI.Cell предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 28).

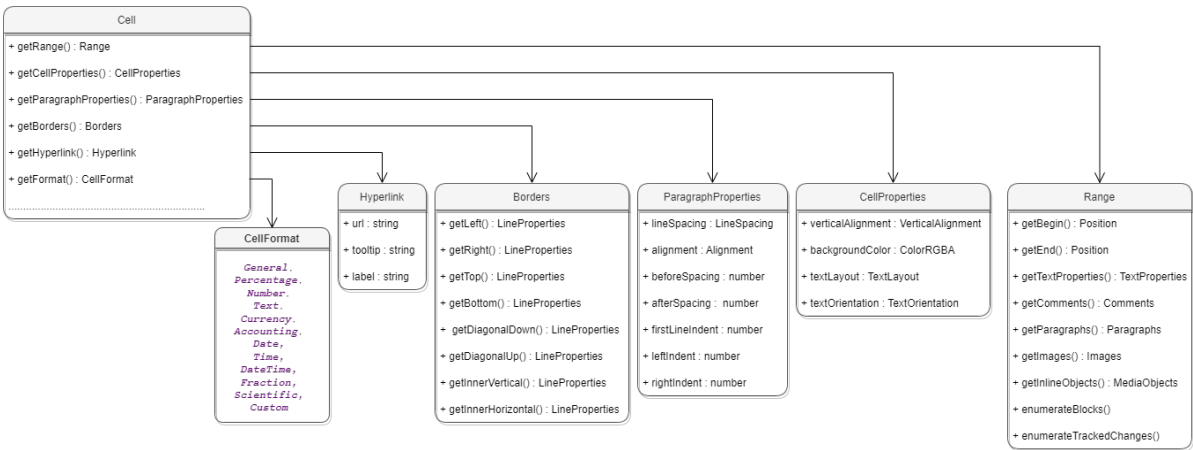


Рисунок 28 – Объектная модель ячейки таблиц

7.10.1 Метод Cell:checkDataValidation

Метод проверяет значение ячейки согласно заданной в ней проверке данных.

Вызов

```
DataValidationResult checkDataValidation()
```

Возвращает

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

Пример

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

7.10.2 Метод Cell:getBorders

Позволяет получить границы ячейки (тип [DocumentAPI.Borders](#)). Примеры использования приведены в разделе [DocumentAPI.Borders](#).

Пример

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

7.10.3 Метод Cell:getCellProperties

Позволяет получить свойства [DocumentAPI.CellProperties](#) ячейки.

Пример

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

7.10.4 Метод Cell:getColumnIndex

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

Пример

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

7.10.5 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

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

7.10.6 Метод Cell:getCustomFormat

Возвращает строку формата ячейки.

Пример

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

7.10.7 Метод Cell:getDataValidation

Метод возвращает настройки проверки данных для текущей ячейки.

Вызов

```
DataValidation getDataValidation()
```

Возвращает

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

Пример

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

7.10.8 Метод Cell:getFormat

Метод возвращает формат ячейки. Список поддерживаемых форматов ячеек приведен в разделе [DocumentAPI.CellFormat](#).

Пример

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

local cellFormatting = DocumentAPI.PercentageCellFormatting()
```

```
cell:setFormat(cellFormatting)
print("Формат: ", cell:getFormat()) -- 1
```

7.10.9 Метод Cell:getFormattedValue

Метод позволяет получить значение ячейки в текущем формате. Список поддерживаемых форматов см. в разделе [DocumentAPI.CellFormat](#).

Пример

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

7.10.10 Метод Cell:getFormulaAsString

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

Пример

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

7.10.11 Метод Cell:getHyperlink

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

```
local cell =
document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0, 0))
local hyperlink = cell:getHyperlink()
if (hyperlink ~= nil) then
    print(hyperlink)
end
```

7.10.12 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

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

7.10.13 Метод Cell:getNote

Метод возвращает текст заметки для текущей ячейки.

Вызов

```
string getNote()
```

Возвращает

— текст заметки.

Пример

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

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

7.10.14 Метод Cell:getParagraphProperties

Возвращает свойства абзаца [DocumentAPI.ParagraphProperties](#), находящегося в ячейке.

Пример

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

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

7.10.15 Метод Cell:getPivotTable

Возвращает сводную таблицу [DocumentAPI.PivotTable](#), относящуюся к ячейке.

Пример

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

7.10.16 Метод `Cell:getProtectionProperties`

Метод возвращает параметры защиты ячейки табличного документа.

Вызов

```
CellProtectionProperties getProtectionProperties()
```

Возвращает

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

Пример

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

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

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

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

7.10.17 Метод `Cell:getRange`

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

Пример

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

7.10.18 Метод `Cell:getRawValue`

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

Пример

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

7.10.19 Метод Cell:getRowIndex

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

Пример

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

7.10.20 Метод Cell:getTable

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

Вызов

```
Table getTable()
```

Пример

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

7.10.21 Метод Cell:getTextProperties

Метод возвращает текущие настройки форматирования текста для ячейки.

Вызов

```
TextProperties getTextProperties()
```

Возвращает

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

Пример

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

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

cell:setTextProperties(props)
```

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

7.10.22 Метод `Cell:isInMergedRange`

Метод позволяет определить находится ли ячейка в объединенном диапазоне.

Вызов

```
bool isInMergedRange()
```

Возвращает

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

Пример

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

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

7.10.23 Метод `Cell:isPivotTableRoot`

Метод позволяет определить является ли ячейка основанием сводной таблицы.

Пример

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

7.10.24 Метод `Cell:isProtected`

Метод возвращает статус защиты от редактирования ячейки в табличном документе.

Вызов

```
bool isProtected()
```

Методы [setProtectionProperties\(\)](#) и [getProtectionProperties\(\)](#) позволяют задать и получить параметры защиты ячейки ([CellProtectionProperties](#)).

7.10.25 Метод `Cell:removeNote`

Метод удаляет заметку из текущей ячейки.

Вызов

```
removeNote()
```

Пример

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

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

7.10.26 Метод Cell:setBool

Устанавливает для ячейки значение логического типа.

Пример

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

7.10.27 Метод Cell:setBorders

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

7.10.28 Метод Cell:setCellProperties

Позволяет установить свойства ячейки [DocumentAPI.CellProperties](#).

Пример

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

7.10.29 Метод Cell:setContent

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

Пример

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

7.10.30 Метод `Cell:setCustomFormat`

Устанавливает формат ячейки.

Пример

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

7.10.31 Метод `Cell:setFormat`

Метод устанавливает формат ячейки. Существуют несколько вариантов использования метода.

Варианты вызова метода

```
setFormat(cellFormat)
```

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

```
setFormat(accountingCellFormatting)
```

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

```
setFormat(percentageCellFormatting)
```

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

```
setFormat(numberCellFormatting)
```

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

```
setFormat(currencyCellFormatting)
```

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

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

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


```
setFormat(fractionCellFormatting)
```

Где **fractionCellFormatting** – формат ячейки типа

[DocumentAPI.FractionCellFormatting](#).

```
setFormat(scientificCellFormatting)
```

Где **scientificCellFormatting** – формат ячейки типа

[DocumentAPI.ScientificCellFormatting](#).

Примеры использования

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

cellA1:setNumber(2.3)

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

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

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

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

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

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

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

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

7.10.32 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 `DocumentAPI.CellFormat_Text`.

Список поддерживаемых форматов см. в разделе [DocumentAPI.CellFormat](#). Пример использования метода см. в разделе [Доступ к ячейкам](#).

7.10.33 Метод `Cell:setFormula`

Метод позволяет вставить формулу в ячейку табличного документа.

Пример

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

7.10.34 Метод `Cell:setNote`

Метод добавляет заметку в текущую ячейку.

Вызов

```
setNote(noteText)
```

Параметры

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

Пример

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

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

7.10.35 Метод `Cell:setNumber`

Устанавливает для ячейки значение числового типа.

Пример

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

7.10.36 Метод `Cell:setParagraphProperties`

Устанавливает свойства абзаца [DocumentAPI.ParagraphProperties](#), находящегося в ячейке.

Пример

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

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

7.10.37 Метод Cell:setProtectionProperties

Метод задает параметры защиты ячейки в табличном документе.

Вызов

```
setProtectionProperties(protectionProps)
```

Параметры

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

Пример

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

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

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

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

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

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

7.10.38 Метод Cell:setText

Устанавливает для ячейки значение строкового типа.

Пример

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

7.10.39 Метод Cell:setTextProperties

Метод задает настройки форматирования текста для ячейки.

Вызов

```
setTextProperties(props)
```

Параметры

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

Пример

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

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

cell:setTextProperties(props)
```

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

7.10.40 Метод Cell:unmerge

Разъединяет несколько ячеек, которые были объединены ранее. Примеры объединения и разъединения ячеек см. в разделе [Объединение и разделение ячеек таблицы](#).

Пример

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

7.11 Таблица DocumentAPI.CellFormat

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

Таблица 8 – Поддерживаемые форматы ячеек таблицы

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

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

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

Примеры использования

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

Результат выполнения данного примера приведен на рисунке 29.

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

Рисунок 29 – Результат установки формата

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

7.12 Таблица DocumentAPI.CellPosition

Таблица DocumentAPI.CellPosition позволяет задать координаты ячейки электронной таблицы или таблицы в составе текстового документа.

Позиция ячейки A1 имеет координаты (0, 0).

Также для указания адреса ячейки в качестве параметра метода `getCell` можно использовать строку вида «A1».

Примеры

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

```
local table = document:getBlocks():getTable(0) -- первый лист документа
local cell = table:getCell("A3") -- ячейка A3
```

7.12.1 Поле CellPosition.column

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

Пример

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

7.12.2 Поле `CellPosition.row`

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

Пример

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

7.12.3 Метод `CellPosition.toString`

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

Пример

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

7.12.4 Метод `CellPosition.__eq`

Метод используется для определения эквивалентности двух объектов `CellPosition`.

Пример

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

7.13 Таблица `DocumentAPI.CellProperties`

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

Для задания свойств ячейки используется метод [Cell.setCellProperties\(\)](#). Для получения свойств ячейки используется метод [Cell.getCellProperties\(\)](#). Иерархия таблиц и полей `DocumentAPI.CellProperties` отображена на рисунке 30.

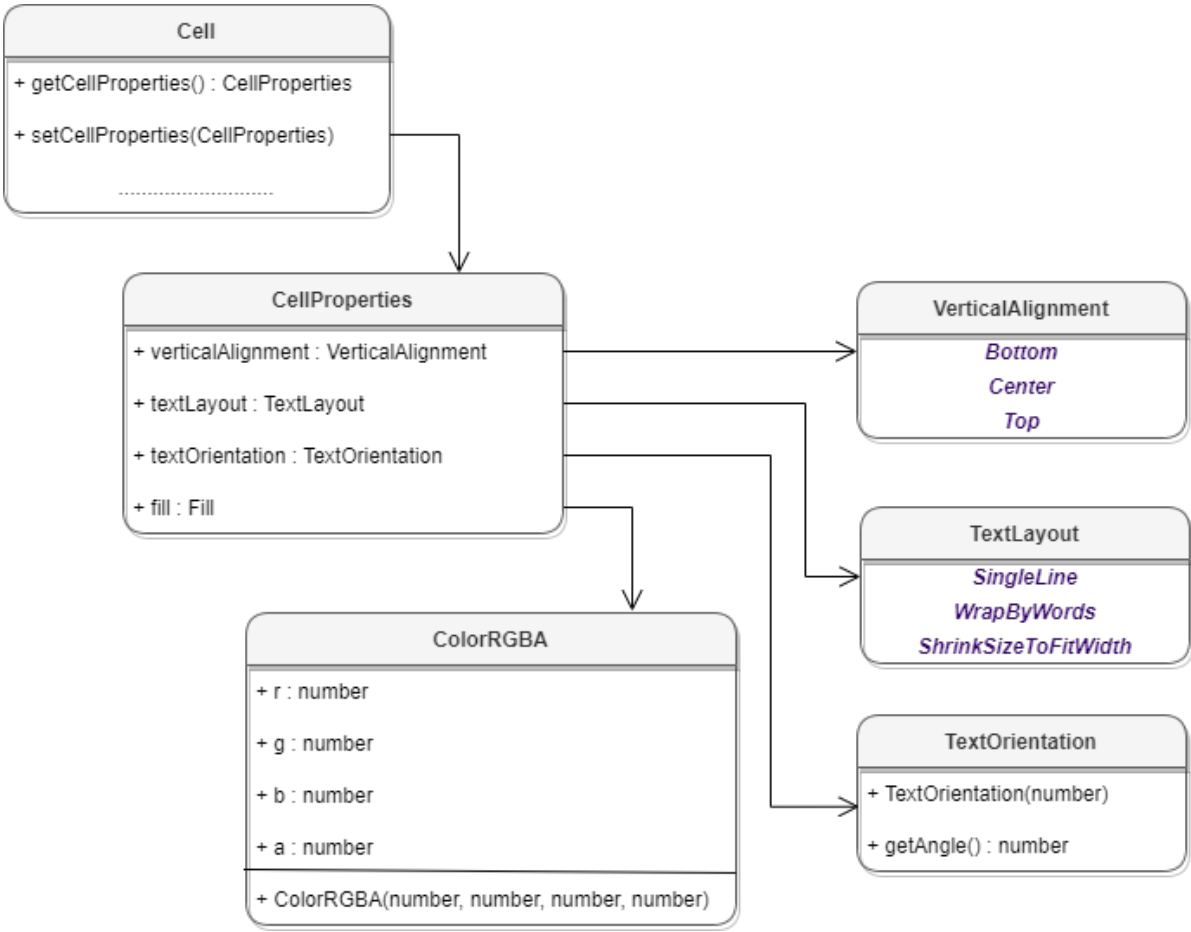


Рисунок 30 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 9 – Описание полей таблицы DocumentAPI.CellProperties

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

Пример

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

props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
```

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

cell:setCellProperties(props)
```

7.13.1 Метод CellProperties:__eq

Метод используется для определения эквивалентности значений двух объектов CellProperties.

Пример

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

7.14 Таблица DocumentAPI.CellProtectionProperties

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

Таблица 10 – Описание полей таблицы DocumentAPI.CellProtectionProperties

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

Пример

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

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

```
cellProps.formulasNotDisplayed = false

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

7.15 Таблица DocumentAPI.CellRange

Таблица DocumentAPI.CellRange описывает диапазон ячеек таблицы.

Пример

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

7.15.1 Метод CellRange:autoFill

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

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

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

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

Пример

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

7.15.2 Метод CellRange:clearDataValidations

Метод убирает проверку данных из текущего диапазона ячеек.

Вызов

```
clearDataValidations()
```

Пример

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

7.15.3 Метод `CellRange:containsCell`

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

Примеры

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

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

Дополнительный пример использования метода `CellRange:containsCell` приведен в разделе [Доступ к ячейкам](#).

7.15.4 Метод `CellRange:copyInto`

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

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

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

Пример (только для табличного документа)

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

При копировании ячеек в качестве новой позиции достаточно указать верхнюю левую ячейку нового диапазона, однако, если необходимо продублировать исходный блок ячеек, в качестве параметра следует использовать диапазон, превышающий размеры исходного диапазона, но кратный его размерам. Например, при копировании диапазона "A1:B2" (размер 2x2) в диапазон "B5:E6" (размер 2x4) блок исходных ячеек продублируется два раза (см. Рисунок 31).

	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 31 – Копирование ячеек табличного документа

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

7.15.5 Метод `CellRange:enumerate`

Метод возвращает коллекцию ячеек в диапазоне.

Пример

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

7.15.6 Метод `CellRange:find`

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

Вызов

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

Параметры

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

Возвращает

- список ячеек, соответствующих поисковому запросу.

Пример

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

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

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

7.15.7 Метод CellRange:getAddress

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

Вызов

```
string getAddress(addressSettings)
```

Параметры

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

Возвращает

– адрес диапазона ячеек.

Пример

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

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

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

7.15.8 Метод `CellRange:getBeginColumn`

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

Пример

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

7.15.9 Метод `CellRange:getBeginRow`

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

Пример

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

7.15.10 Метод `CellRange:getCellProperties`

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

Пример

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

7.15.11 Метод `CellRange:getLastColumn`

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

Пример


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

7.15.12 Метод `CellRange:getLastRow`

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

Пример

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

7.15.13 Метод `CellRange:getParagraphProperties`

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

Вызов

```
ParagraphProperties getParagraphProperties(skipHiddenCells)
```

Параметры

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

Возвращает

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

Пример

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

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

cellRange:setParagraphProperties(paragraphProperties, false)
```

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

7.15.14 Метод `CellRange:getProtectionProperties`

Метод возвращает параметры защиты диапазона ячеек табличного документа.

Вызов

```
CellProtectionProperties getProtectionProperties()
```

Возвращает

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

Пример

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

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

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

Свойства возвращаемого объекта [CellProtectionProperties](#) могут быть `nil`, если текущий диапазон содержит ячейки с разными параметрами. Метод [setProtectionProperties\(\)](#) позволяет задать параметры защиты диапазона ячеек. Вы также можете использовать метод [isProtected\(\)](#), чтобы узнать, защищены ли ячейки диапазона от редактирования.

7.15.15 Метод `CellRange:getTable`

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

Пример

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

7.15.16 Метод `CellRange:getTableRange`

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

7.15.17 Метод `CellRange:getTextProperties`

Метод возвращает текущие настройки форматирования текста для диапазона ячеек.

Вызов

```
TextProperties getTextProperties(skipHiddenCells)
```

Параметры

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

Возвращает

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

Пример

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

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

cellRange:setTextProperties(props)
```

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

7.15.18 Метод `CellRange:insert`

Метод вставляет пустые ячейки на место текущего диапазона и сдвигает существующие ячейки диапазона в заданном направлении.

Вызов

```
insert(shiftAxis)
```

Параметры

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

Пример

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

cellRange:insert(DocumentAPI.CellShiftAxis_Vertical)
```

7.15.19 Метод `CellRange:insertCurrentDateTime`

Метод служит для установки значения даты/времени

[DocumentAPI.DateTimeFormat](#) диапазона ячеек.

Пример

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

7.15.20 Метод `CellRange:isProtected`

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

Вызов

```
bool isProtected()
```

Метод `isProtected()` возвращает `nil`, если часть ячеек диапазона защищена от редактирования, а часть – нет. Методы [setProtectionProperties\(\)](#) и [getProtectionProperties\(\)](#) позволяют задать и получить параметры защиты диапазона ячеек ([CellProtectionProperties](#)).

7.15.21 Метод `CellRange:merge`

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

Пример

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

7.15.22 Метод `CellRange:moveInto`

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

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

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

Пример (только для табличного документа)

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

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

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

7.15.23 Метод `CellRange:remove`

Метод удаляет ячейки текущего диапазона и сдвигает на их место оставшиеся ячейки в заданном направлении.

Вызов

```
remove(shiftAxis)
```

Параметры

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

Пример

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

cellRange:remove(DocumentAPI.CellShiftAxis_Horizontal)
```

7.15.24 Метод `CellRange:setBorders`

Метод предназначен для установки границ диапазона ячеек. Отдельные границы устанавливаются с помощью методов таблицы [DocumentAPI.Borders](#).

Пример

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

7.15.25 Метод `CellRange:setCellProperties`

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

Пример

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

7.15.26 Метод `CellRange:setDataValidation`

Метод устанавливает проверку данных для текущего диапазона ячеек.

Вызов

```
setDataValidation(DataValidation dataValidation)
```

Параметры

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

Пример

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

cellRange:setDataValidation(dvList)
```

7.15.27 Метод CellRange:setParagraphProperties

Метод задает настройки форматирования абзацев, находящихся в диапазоне ячеек.

Вызов

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

Параметры

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

Пример

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

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

cellRange:setParagraphProperties(paragraphProperties, false)
```

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

7.15.28 Метод CellRange:setProtectionProperties

Метод задает параметры защиты диапазона ячеек в табличном документе.

Вызов

```
setProtectionProperties(protectionProps)
```

Параметры

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

Пример

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

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

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

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

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

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

7.15.29 Метод `CellRange:setTextProperties`

Метод задает настройки форматирования текста для диапазона ячеек.

Вызов

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

Параметры

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

Пример

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

local props = cellRange:getTextProperties()
```



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

cellRange.setTextProperties(props)
```

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

7.15.30 Метод CellRange.sort

Метод сортирует строки текущего диапазона в соответствии с заданными условиями.

Вызов

```
sort(sortingConditions)
```

Параметры

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

Пример

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

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

range.sort(conditions)
```

Метод вызывает `DocumentModificationError` в следующих случаях:

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

7.16 Таблица DocumentAPI.CellRangeAddressFormat

Таблица `CellRangeAddressFormat` определяет формат отображения адреса диапазона ячеек. Используется в поле [CellRangeAddressSettings.addressFormat](#).

Таблица 11 – Описание форматов отображения адреса

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

7.17 Таблица DocumentAPI.CellRangeAddressSettings

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

Таблица 12 – Описание полей таблицы `CellRangeAddressSettings`

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

Пример

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

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

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

7.18 Таблица DocumentAPI.CellRangePosition

Таблица `DocumentAPI.CellRangePosition` представляет положение диапазона ячеек в таблице.

Для создания нового объекта `DocumentAPI.CellRangePosition` используется один из следующих конструкторов:

```
DocumentAPI.CellRangePosition(DocumentAPI.CellPosition leftTopPosition,  
DocumentAPI.CellPosition rightBottomPosition)
```

Где:

- `leftTopPosition` – позиция левой верхней ячейки диапазона;
- `rightBottomPosition` – позиция правой нижней ячейки диапазона.

```
DocumentAPI.CellRangePosition(leftColumn: number, topRow: number, rightColumn:  
number, bottomRow: number)
```

Где:

- `leftColumn` – индекс левого столбца диапазона, индексирование происходит с нуля;
- `topRow` – индекс верхней строки диапазона (индексирование производится с нуля);
- `rightColumn` – индекс правого столбца диапазона (индексирование производится с нуля);
- `bottomRow` – индекс нижней строки диапазона (индексирование производится с нуля).

Объект `DocumentAPI.CellRangePosition` используется в качестве поля `tableRange` таблицы [DocumentAPI.TableRangeInfo](#), а также в методах [Table.getCellRange\(\)](#), [Chart.setRange\(\)](#). По умолчанию диапазон включает одну ячейку в позиции 0,0 что соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.

Описание полей таблицы `DocumentAPI.CellRangePosition` представлено в таблице 13.

Таблица 13 – Поля таблицы `DocumentAPI.CellRangePosition`

Поле	Тип	Описание
<code>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.
bottomRight	CellPosition	Содержит позицию правой нижней ячейки таблицы прямоугольного диапазона.

Примеры

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

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

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

7.18.1 Метод CellRangePosition:toString

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

Пример

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

7.18.2 Метод CellRangePosition:__eq

Метод используется для определения эквивалентности двух объектов CellRangePosition.

Пример

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

7.19 Таблица DocumentAPI.CellRanges

Таблица `DocumentAPI.CellRanges` представляет собой коллекцию диапазонов ячеек ([DocumentAPI.CellRange](#)).

Конструктор

```
DocumentAPI.CellRanges(document)
```

7.19.1 Метод CellRanges:addRange

Метод добавляет заданный диапазон ячеек в коллекцию.

Вызов

```
addRange(range)
```

Параметры

— `range`: диапазон ячеек, тип [CellRange](#).

Пример

```
local sheet = document:getBlocks():getTable(0)
local cellrange = sheet:getCellRange("A1:A3")
local cellrange1 = sheet:getCellRange("C1:C3")

ranges = DocumentAPI.CellRanges(document)
ranges:addRange(cellrange)
ranges:addRange(cellrange1)
```

7.19.2 Метод CellRanges:clear

Метод очищает текущую коллекцию диапазонов ячеек.

Вызов

```
clear()
```

7.19.3 Метод CellRanges:enumerate

Метод возвращает перечисление диапазонов ячеек.

Вызов

```
function enumerate()
```

Возвращает

– перечисление диапазонов ячеек.

Пример

```
local ranges = EditorAPI.getSelection()
for range in ranges:enumerate() do
    print(range:getTableRange():toString())
end
```

7.20 Таблица DocumentAPI.CellShiftAxis

Таблица `CellShiftAxis` определяет направление сдвига ячеек при вставке или удалении диапазона. Используется в методах [CellRange.insert\(\)](#) и [CellRange.remove\(\)](#).

Таблица 14 – Описание направлений сдвига ячеек

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

Пример

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

cellRange:insert(DocumentAPI.CellShiftAxis_Vertical)
```

7.21 Таблица DocumentAPI.Chart

Таблица `DocumentAPI.Chart` представляет диаграмму в табличном документе и описывает все ее элементы (заголовков, легенда, тип, данные, диапазон и т.д.). Объектная модель `DocumentAPI.Chart` приведена на рисунке 32.

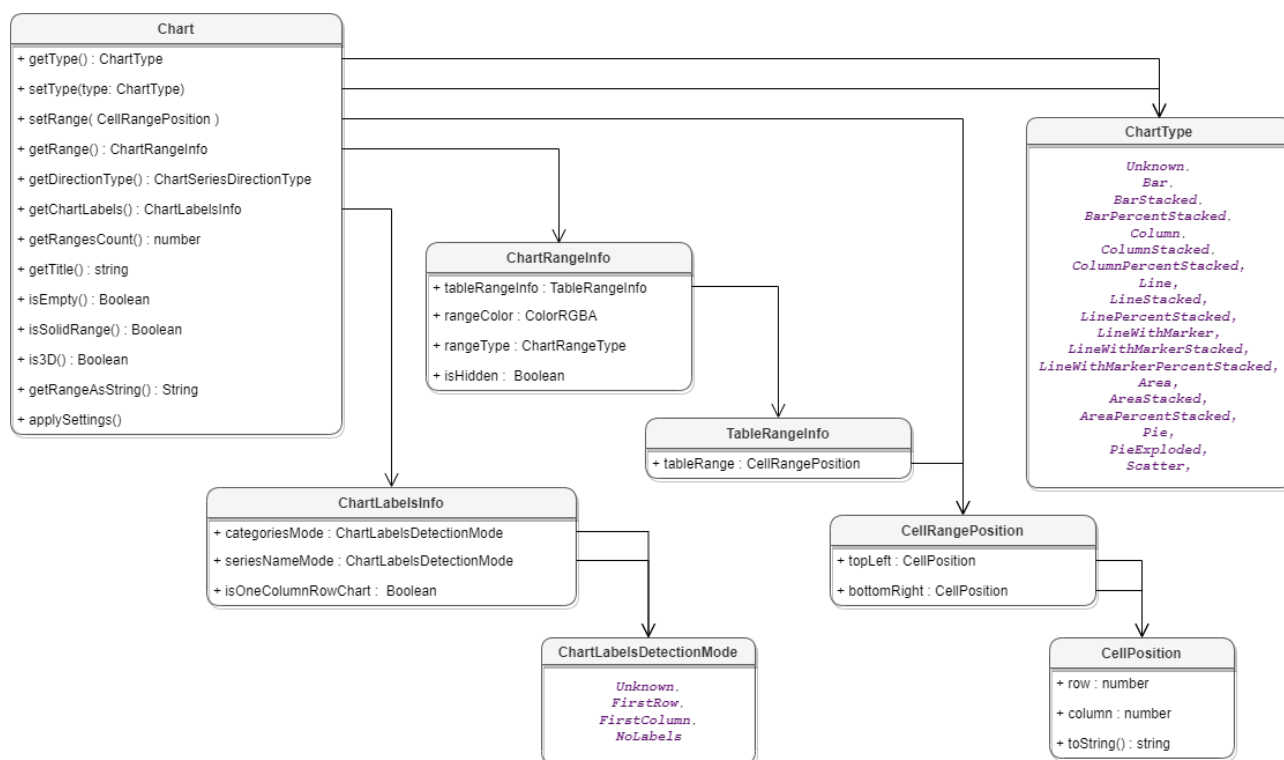


Рисунок 32 – Объектная модель таблицы DocumentAPI.Chart

7.21.1 Метод Chart:applySettings

Метод позволяет обновить параметры текущей выбранной диаграммы.

Вызов

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

Параметры

– cellRange – обновленный диапазон исходных данных диаграммы

[DocumentAPI.CellRange](#);

– directionType – направление серий

[DocumentAPI.ChartSeriesDirectionType](#);

– title – заголовок диаграммы (тип - строка);

– labelsInfo – информация о метках диаграммы [DocumentAPI.ChartLabelsInfo](#).

Пример

```

local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()

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

```
local directionType = DocumentAPI.ChartSeriesDirectionType_ByRow
local title = 'Title'
local chartLabelsInfo =
DocumentAPI.ChartLabelsInfo(DocumentAPI.ChartLabelsDetectionMode_FirstRow,
DocumentAPI.ChartLabelsDetectionMode_FirstRow, false)

charts:getChart(0):applySettings(cellRange, directionType, title,
chartLabelsInfo)
```

7.21.2 Метод Chart:getChartLabels

Метод возвращает коллекцию меток диаграммы типа

[DocumentAPI.ChartLabelsInfo](#).

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local chart = charts:getChart(0)
local chartLabelsInfo = chart:getChartLabels()
print(chartLabelsInfo.categoriesMode, chartLabelsInfo.seriesNameMode,
chartLabelsInfo.isOneColumnRowChart)
```

7.21.3 Метод Chart:getDirectionType

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

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getDirectionType())
```

7.21.4 Метод Chart:getRange

Метод возвращает диапазон ячеек [DocumentAPI.ChartRangeInfo](#) с исходными данными диаграммы. Параметр `rangeIndex` – индекс диапазона.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRange(0).rangeType)
```


7.21.5 Метод Chart:getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRangeAsString())
```

7.21.6 Метод Chart:getRangesCount

Метод возвращает количество серий диаграммы.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRangesCount())
```

7.21.7 Метод Chart:getTitle

Метод возвращает заголовок диаграммы.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getTitle())
```

7.21.8 Метод Chart:getType

Метод возвращает тип диаграммы [DocumentAPI.ChartType](#).

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getType())
```

7.21.9 Метод Chart:is3D

Метод возвращает true, если диаграмма трехмерная.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):is3D())
```

7.21.10 Метод Chart:isEmpty

Метод возвращает true, если диаграмма не содержит значений.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):isEmpty())
```

7.21.11 Метод Chart:isSolidRange

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

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):isSolidRange())
```

7.21.12 Метод Chart:setRange

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

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local cellRangePosition = DocumentAPI.CellRangePosition(0, 0, 5, 5)
charts:getChart(0):setRange(cellRangePosition)
```

7.21.13 Метод Chart:setRect

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



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

7.21.14 Метод Chart:setType

Метод устанавливает тип диаграммы [DocumentAPI.ChartType](#). Параметр chartType - новый тип диаграммы.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
charts:getChart(0):setType(DocumentAPI.ChartType_LineStacked)
print(charts:getChart(0):getType())
```

7.22 Таблица DocumentAPI.ChartLabelsDetectionMode

Таблица `DocumentAPI.ChartLabelsDetectionMode` описывает режимы автоматического определения меток диаграмм. Описание полей таблицы представлено в таблице 15.

Таблица 15 – Описание полей таблицы `DocumentAPI.ChartLabelsDetectionMode`

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

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local chart = charts:getChart(0)
local chartLabels = chart:getChartLabels()
print(chartLabels.categoriesMode, chartLabels.seriesNameMode)
```

7.23 Таблица DocumentAPI.ChartLabelsInfo

Таблица `DocumentAPI.ChartLabelsInfo` описывает настройки автоматического определения меток диаграммы. Инициализируется конструктором, в который передаются параметры:

- `categoriesMode` – режим автоматического определения меток для категорий, тип [`DocumentAPI.ChartLabelsDetectionMode`](#);
- `seriesNameMode` – режим автоматического определения меток для серий, тип [`DocumentAPI.ChartLabelsDetectionMode`](#);
- `oneColumnRow` – передается `true`, если диапазон диаграммы содержит только одну строку или одну колонку.

Описание полей таблицы представлено в таблице 16.

Таблица 16 – Описание полей таблицы DocumentAPI.ChartLabelsInfo

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

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local chart = charts:getChart(0)
local chartLabelsInfo = chart:getChartLabels()
print(chartLabelsInfo.categoriesMode, chartLabelsInfo.seriesNameMode,
chartLabelsInfo.isOneColumnRowChart)
```

7.24 Таблица DocumentAPI.ChartRangeInfo

Таблица DocumentAPI.ChartRangeInfo описывает серию диаграммы.

Инициализируется конструктором, в который передаются следующие параметры:

- tableRangeInfo - диапазон ячеек, тип [DocumentAPI.TableRangeInfo](#);
- color – цвет серии диаграммы, тип [DocumentAPI.ColorRGBA](#);
- hidden – видимость серии, тип Boolean;
- rangeType – тип диапазона исходных данных диаграммы, тип [DocumentAPI.ChartRangeType](#).

Описание полей таблицы представлено в таблице 17.

Таблица 17 – Описание полей таблицы DocumentAPI.ChartRangeInfo

Поле	Описание	Тип
tableRangeInfo	Исходный диапазон ячеек для серии	DocumentAPI.TableRangeInfo
rangeColor	Цвет для отрисовки серии	DocumentAPI.ColorRGBA
isHidden	Задаёт видимость серии диаграммы	Boolean
rangeType	Тип диапазона диаграммы	DocumentAPI.ChartRangeType

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local rangeInfo = charts:getChart(0):getRange(0)
print(rangeInfo.tableRangeInfo, rangeInfo.rangeColor, rangeInfo.isHidden,
rangeInfo.rangeType)
```

7.25 Таблица DocumentAPI.ChartRangeType

Таблица DocumentAPI.ChartRangeType описывает тип диапазона исходных данных диаграммы. Описание полей таблицы представлено в таблице 18.

Таблица 18 – Описание полей таблицы DocumentAPI.ChartRangeType

Поле	Описание
DocumentAPI.ChartRangeType_Series	Серии
DocumentAPI.ChartRangeType_SeriesName	Имена серий
DocumentAPI.ChartRangeType_Categories	Категории
DocumentAPI.ChartRangeType_DataPoint	Разметка данных

Пример

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

rangeTypes = {"Series", "SeriesName", "Categories", "DataPoint" }
print(rangeTypes[rangeInfo.rangeType + 1])
```

7.26 Таблица DocumentAPI.Charts

Таблица DocumentAPI.Charts обеспечивает доступ к списку диаграмм (см. Рисунок 33) табличного документа. Доступ к списку диаграмм осуществляется с помощью метода [Table.getCharts\(\)](#).

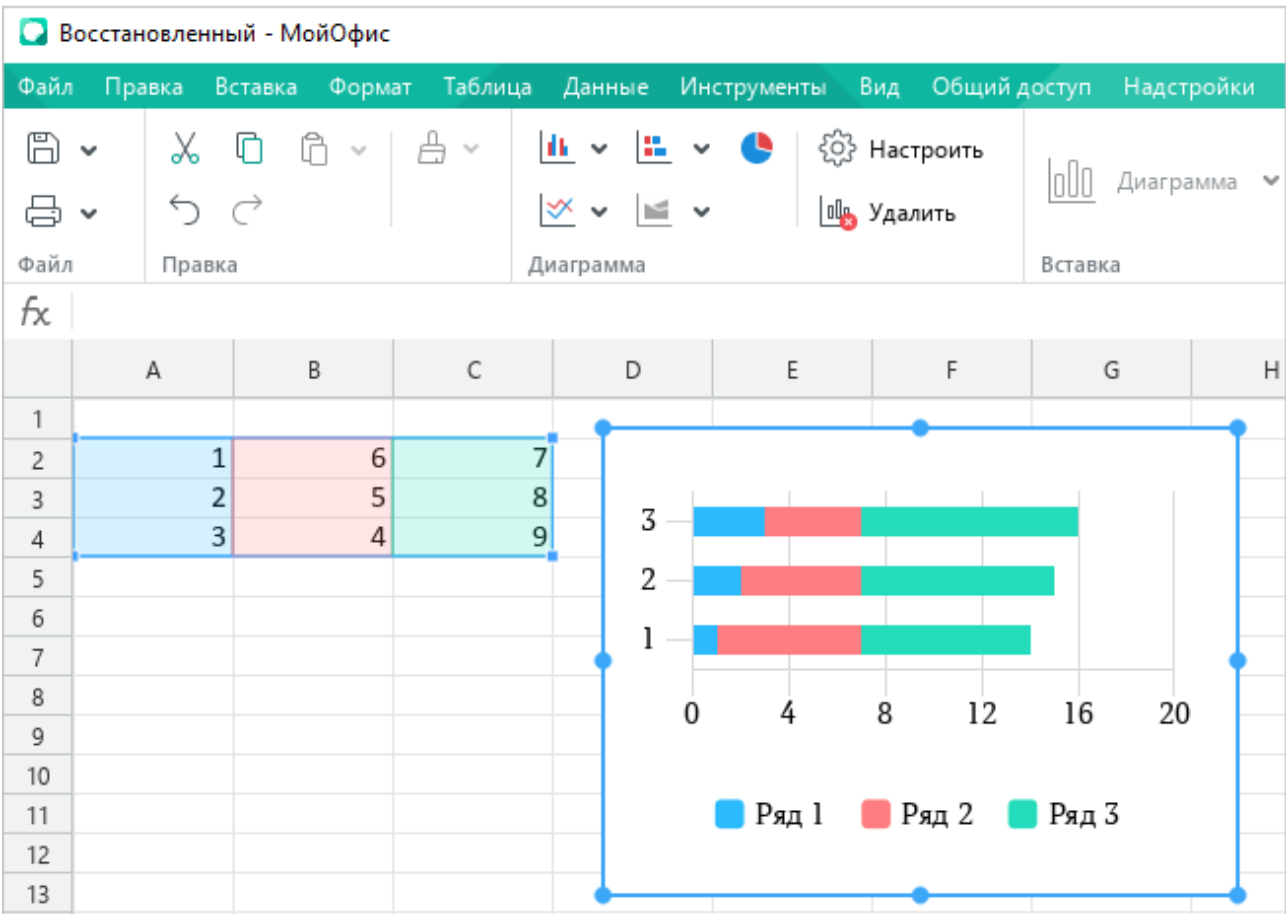


Рисунок 33 – Пример отображения диаграммы в МойОфис Таблица.

7.26.1 Метод Charts:getChart

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

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getRangeAsString())
```

7.26.2 Метод Charts:getChartIndexByDrawingIndex

Метод возвращает индекс диаграммы по индексу отрисовки `drawingIndex`.

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChartIndexByDrawingIndex(0))
```

7.26.3 Метод Charts:getChartsCount

Метод возвращает общее количество диаграмм в табличном документе.

Пример

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

7.27 Таблица DocumentAPI.ChartSeriesDirectionType

Таблица DocumentAPI.ChartSeriesDirectionType описывает направление серий диаграмм. Описание полей таблицы представлено в таблице 19.

Таблица 19 – Описание полей таблицы DocumentAPI.ChartSeriesDirectionType

Поле	Описание
DocumentAPI.ChartSeriesDirectionType_Unknown	Неопределенный тип
DocumentAPI.ChartSeriesDirectionType_ByRow	Серии направлены по строкам
DocumentAPI.ChartSeriesDirectionType_ByColumn	Серии направлены по колонкам

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getDirectionType())
```

7.28 Таблица DocumentAPI.ChartType

Таблица DocumentAPI.ChartType описывает все поддерживаемые типы диаграмм. Описание полей таблицы представлено в таблице 20.

Таблица 20 – Описание полей таблицы DocumentAPI.ChartType

Поле	Описание
DocumentAPI.ChartType_Unknown	Неопределенный тип
DocumentAPI.ChartType_Bar	Линейчатая диаграмма с группировкой
DocumentAPI.ChartType_BarStacked	Линейчатая диаграмма с накоплением
DocumentAPI.ChartType_BarPercentStacked	Линейчатая нормированная диаграмма с накоплением
DocumentAPI.ChartType_Column	Гистограмма с группировкой
DocumentAPI.ChartType_ColumnStacked	Гистограмма с накоплением

Поле	Описание
DocumentAPI.ChartType_ColumnPercentStacked	Нормированная гистограмма с накоплением
DocumentAPI.ChartType_Line	Стандартный график
DocumentAPI.ChartType_LineStacked	График с накоплением
DocumentAPI.ChartType_LinePercentStacked	Нормированный график с накоплением
DocumentAPI.ChartType_LineWithMarker	Стандартный график с маркерами
DocumentAPI.ChartType_LineWithMarkerStacked	График с накоплением и маркерами
DocumentAPI.ChartType_LineWithMarkerPercentStacked	Нормированный график с накоплением и маркерами
DocumentAPI.ChartType_Area	Стандартная диаграмма с областями
DocumentAPI.ChartType_AreaStacked	Диаграмма с областями с накоплением
DocumentAPI.ChartType_AreaPercentStacked	Нормированная диаграмма с областями с накоплением
DocumentAPI.ChartType_Pie	Круговая диаграмма
DocumentAPI.ChartType_PieExploded	Круговая диаграмма с отделенными секторами
DocumentAPI.ChartType_Scatter	Диаграмма рассеяния

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
print(charts:getChart(0):getType())
```

7.29 Таблица DocumentAPI.CheckBoxControl

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

Пример

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


7.30 Таблица DocumentAPI.Color

Таблица `DocumentAPI.Color` представляет либо цветовой объект RGBA, либо заданные цвета идентификатора темы. В качестве параметров конструктора используются таблицы [DocumentAPI.ColorRGBA](#), [DocumentAPI.ThemeColorID](#).

Пример

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

7.30.1 Метод Color:getRGBAColor

Метод возвращает цвет [DocumentAPI.ColorRGBA](#).

Пример

```
local color = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))
local rgbaColor = color:getRGBAColor()
if rgbaColor then
    print(rgbaColor.r, rgbaColor.g, rgbaColor.b, rgbaColor.a)
end
```

7.30.2 Метод Color:getThemeColorID

Метод возвращает цвет идентификатора темы [DocumentAPI.ThemeColorID](#).

Пример

```
local color = DocumentAPI.Color(DocumentAPI.ThemeColorID_Text1)
local themeColorID = color:getThemeColorID()
if themeColorID then
    print(themeColorID)
end
```

7.30.3 Метод Color:getTransforms

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

Пример

```
local color = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))
print(color:getTransforms())
```

7.30.4 Метод Color:setTransforms

Метод устанавливает трансформацию цвета [ColorTransforms](#).

Пример

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

7.30.5 Метод Color:__eq

Метод используется для определения эквивалентности двух значений цвета.

Пример

```
local color = DocumentAPI.Color(DocumentAPI.ThemeColorID_Text1)
if color:__eq(color) then
    print("Equals")
end
```

7.31 Таблица DocumentAPI.ColorRGBA

Таблица `DocumentAPI.ColorRGBA` предназначена для задания цвета текста, линии, фона и т.д. Используется четырехканальный формат, содержащий данные для красного (r), зеленого (g), голубого (b) цветов и альфа-канала (a).

Для создания нового объекта используется один из конструкторов:

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

Описание полей таблицы `DocumentAPI.ColorRGBA` представлено в таблице 21.

Таблица 21 – Описание таблицы `DocumentAPI.ColorRGBA`

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

Примеры использования

```
local rgba = DocumentAPI.ColorRGBA()  
rgba.r = 0  
rgba.g = 0  
rgba.b = 255  
rgba.a = 200  
print("r=", rgba.r, ", g=", rgba.g, ", b=", rgba.b, ", a=", rgba.a)  -- r=0,  
g=0, b=255, a=200
```

```
local rgba = DocumentAPI.ColorRGBA(55, 146, 179, 200)  
print("r=", rgba.r, ", g=", rgba.g, ", b=", rgba.b, ", a=", rgba.a)  -- r=55,  
g=146, b=179, a=200
```

```
local line_prop = DocumentAPI.LineProperties()  
line_prop.color = DocumentAPI.Color(rgba)
```

7.31.1 Метод ColorRGBA: __eq

Метод используется для определения эквивалентности двух значений цвета DocumentAPI.ColorRGBA.

Пример

```
colorRGBA = DocumentAPI.Color(DocumentAPI.ColorRGBA(55, 146, 179, 200))  
if colorRGBA:__eq(colorRGBA) then  
    print("Equals")  
end
```

7.32 Таблица DocumentAPI.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

```
local color = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))  
colorTransforms = DocumentAPI.ColorTransforms()  
colorTransforms:apply(DocumentAPI.ColorRGBA(55, 146, 179, 200))  
color:setTransforms(colorTransforms)  
print(color:getTransforms())
```

7.32.1 Метод `ColorTransforms:apply`

Метод устанавливает цвет трансформации [ColorRGBA](#).

Пример

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

7.33 Таблица `DocumentAPI.Comment`

Таблица `DocumentAPI.Comment` предоставляет доступ к следующим свойствам комментария:

- диапазон текста [DocumentAPI.Range](#), который описывает комментарий;
- текст комментария;
- информация о комментарии [DocumentAPI.TrackedChangeInfo](#);
- признак того, что комментарий принят;
- список ответов на комментарий [DocumentAPI.Comments](#).

7.33.1 Метод `Comment:getInfo`

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

Пример

```
local commentsList = document:getRange():getComments()  
for comment in commentsList:enumerate() do  
    local commentInfo = comment:getInfo()  
    local name = commentInfo.author.name  
    print("Автор комментария: ", name)  
end
```

7.33.2 Метод `Comment:getRange`

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

Пример

```
local commentsList = document:getRange():getComments()  
for comment in commentsList:enumerate() do  
    print("Диапазон комментария: ", comment:getRange():extractText())  
end
```

7.33.3 Метод `Comment:getReplies`

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

Пример

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    local commentReplies = comment:getReplies()
    for reply in commentReplies:enumerate() do
        local name = reply.author.name
        print("Ответ на комментарий " .. name .. ": ", reply:getText())
    end
end
end
```

7.33.4 Метод `Comment:getText`

Метод возвращает текст комментария.

Пример

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    print("Текст комментария: ", comment:getText())
end
end
```

7.33.5 Метод `Comment:isResolved`

Метод возвращает значение `true`, если комментарий принят.

Пример

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    print("Комментарий принят:", comment:isResolved())
end
end
```

7.34 Таблица `DocumentAPI.Comments`

Таблица `DocumentAPI.Comments` содержит коллекцию комментариев диапазона (см. Рисунок 34).

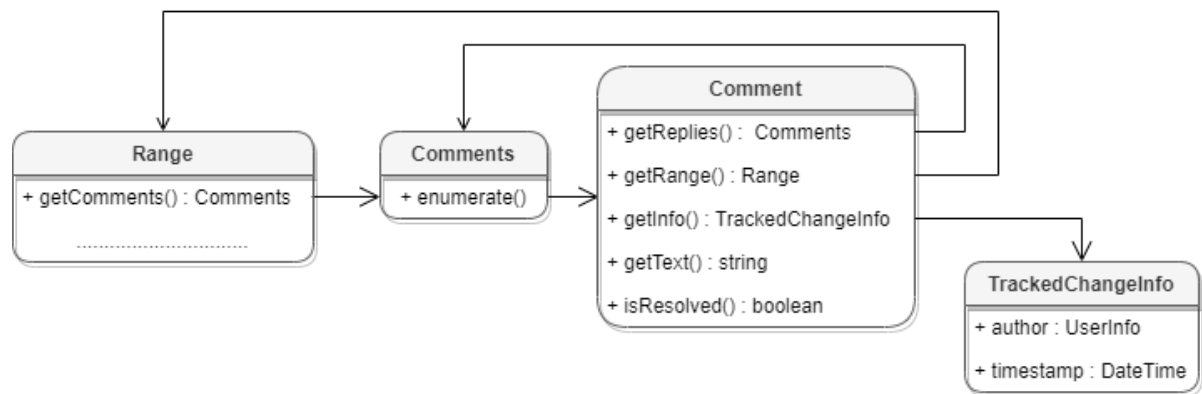


Рисунок 34 – Объектная модель таблиц для работы с комментариями

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

Пример

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    local commentInfo = comment:getInfo()
    local name = commentInfo.author.name
    print("Комментарий " .. name .. ": ", comment:getText())
end
```

7.34.1 Метод Comments:enumerate

Метод возвращает коллекцию комментариев всего документа.

Пример

```
local comments = document:getComments()
for comment in comments:enumerate() do
    print(comment:getText())
end
```

7.35 Таблица DocumentAPI.ConditionalTableFilter

Таблица ConditionalTableFilter реализует фильтр, содержащий предикат(ы) для фильтрации строк. Согласно схеме XML, можно использовать одно или два условия, которые объединяются с помощью логической операции AND или OR. На самом деле поддерживается больше критериев, но рекомендуется использовать только один или два. Если не было добавлено ни одного унарного условия, этот фильтр очищает любой другой фильтр, примененный к определенному столбцу. Этот фильтр сохраняется в документе, но

редактор не полностью его поддерживает. Фильтр может быть загружен и применен редактором, но если документ изменяется, фильтр будет изменен на тип [ValuesTableFilter](#). Если этот фильтр применяется через API, и документ не изменяется после применения фильтра, он будет сохранен как `ConditionalTableFilter`.

Конструктор по умолчанию, логическая операция фильтра, по умолчанию - OR:

```
ConditionalTableFilter()
```

Конструктор с параметром, задающим логическую операцию фильтра

```
ConditionalTableFilter(bool andOperation);
```

Параметр

- `andOperation` – логическая операция фильтра, по умолчанию - OR. В дальнейшем может быть изменена методом [ConditionalTableFilter:setAndOperation](#).

Конструктор копирования:

```
ConditionalTableFilter(ConditionalTableFilter other);
```

Пример использования приведен в разделе [Работа с фильтрами](#).

7.35.1 Метод `ConditionalTableFilter:setAndOperation`

Метод `ConditionalTableFilter:setAndOperation` устанавливает логическую операцию AND. Логическая операция применяется, если определено более одного унарного критерия. Логическая операция по умолчанию - OR.

Пример

```
local songFilter = DocumentAPI.ConditionalTableFilter(johnPaulFilter)
songFilter:setAndOperation(true)
```

7.35.2 Методы добавления условий

Эти методы добавляют в фильтр условия сравнения со значениями, которые передаются в качестве аргумента. Если значение ячейки не соответствует указанным критериям, строки будут скрыты при применении фильтра.



Критерии **match** и **notMatch** не могут быть сохранены для документов формата OXML.

```
equal(string value)
notEqual(string value)
less(string value)
lessOrEqual(string value)
greater(string value)
greaterOrEqual(string value)
match(string value)
notMatch(string value)
begins(string value)
notBegins(string value)
ends(string value)
notEnds(string value)
contains(string value)
notContains(string value)
```

Пример

```
local songFilter = DocumentAPI.ConditionalTableFilter(johnPaulFilter)
songFilter:notEqual(" ")
songFilter:notBegins("TODO")
```

Пример использования приведен в разделе [Работа с фильтрами](#).

7.36 Таблица DocumentAPI.ContentControl

Базовая таблица для элементов управления (см. [Работа с элементами управления](#)).

Наследники

- [CheckBoxControl](#)
- [DatePickerControl](#)
- [DropListControl](#)
- [InputFieldControl](#)

7.36.1 Метод ContentControl:canEdit

Метод возвращает true, если значение элемента управления может быть изменено, и false в обратном случае.

7.36.2 Метод ContentControl:getTitle

Метод возвращает название элемента управления.

7.36.3 Методы toCheckBox, toInputField, toDatePicker, toDropList

Методы преобразуют объект [ContentControl](#) в объект соответствующего типа. Возвращают nil, если преобразование невозможно.

Пример

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

7.37 Таблица DocumentAPI.ContentControls

Предоставляет доступ к операциям с элементами управления в документе (см. [Работа с элементами управления](#)). Метод ContentControls:findByTitle(string) позволяет получить элемент управления [ContentControl](#) по его названию.

Пример

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

7.38 Таблица DocumentAPI.CurrencyCellFormatting

Таблица содержит параметры для денежного формата ячеек таблицы. Описание полей таблицы DocumentAPI.CurrencyCellFormatting представлено в таблице 22.

Таблица 22 – Описание полей таблицы DocumentAPI.CurrencyCellFormatting

Поле	Описание
DocumentAPI.CurrencyCellFormatting.decimalPlaces	Количество десятичных позиций
DocumentAPI.CurrencyCellFormatting.symbol	Символ денежной единицы
DocumentAPI.CurrencyCellFormatting.localeCode	Идентификатор кода языка (MS-LCID)
DocumentAPI.CurrencyCellFormatting.useThousandsSeparator	Использовать разделитель для тысячных
DocumentAPI.CurrencyCellFormatting.useRedForNegative	Использовать красный цвет для отрицательных значений

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

Данная таблица используется в качестве аргумента метода [Cell:setFormat\(\)](#), см. пример.

Пример

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

local currencyCellFormatting = DocumentAPI.CurrencyCellFormatting()
currencyCellFormatting.decimalPlaces = 2
currencyCellFormatting.useThousandsSeparator = true
currencyCellFormatting.useRedForNegative = true
currencyCellFormatting.useBracketsForNegative = true
currencyCellFormatting.hideSign = false
currencyCellFormatting.currencySignPlacement =
DocumentAPI.CurrencySignPlacement_Suffix

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

7.39 Таблица DocumentAPI.CurrencySignPlacement

Варианты размещения знака валюты представлены в таблице 23. Данный тип используется в поле `currencyFormat` таблицы `DocumentAPI.LocaleInfo`, а также в поле `currencySignPlacement` таблицы [DocumentAPI.CurrencyCellFormatting](#) (см.пример в ее описании).

Таблица 23 – Описание полей таблицы `DocumentAPI.CurrencySignPlacement`

Поле	Описание	Пример
DocumentAPI.CurrencySignPlacement_Prefix	Размещение знака валюты до значения.	\$12.00
DocumentAPI.AccountingCellFormatting_Suffix	Размещение знака валюты после значения.	12,00 Р

7.40 Таблица DocumentAPI.DataValidation

Таблица DataValidation представляет собой настройки проверки данных (см. [Проверка данных](#)). Используется в методах [CellRange:setDataValidation\(\)](#), [Cell:getDataValidation\(\)](#) и [DataValidationResult:getDataValidation\(\)](#).

7.40.1 Метод DataValidation:clear

Метод очищает все настройки текущей проверки данных.

Вызов

```
clear()
```

Пример

```
local cell = sheet:getCell("C4")
local validation = cell:getDataValidation()
if not validation:isEmpty() then
    validation:clear()
end
```

7.40.2 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.

7.40.3 Метод DataValidation:getErrorMessage

Метод возвращает текст сообщения об ошибке.

Вызов

```
string getErrorMessage()
```

Возвращает

– текст сообщения об ошибке, тип string.

7.40.4 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](#).

7.40.5 Метод DataValidation:getErrorTitle

Метод возвращает заголовок сообщения об ошибке.

Вызов

```
string getErrorTitle()
```

Возвращает

— заголовок сообщения об ошибке, тип string.

7.40.6 Метод DataValidation:getFormula1

Метод возвращает первый аргумент проверки данных. При проверке по списку значений, первым и единственным аргументом может быть:

- формула, результатом которой является диапазон ячеек ("=INDIRECT(" '[Source.xods]Data' !\$E\$2:\$E\$6")");
- именованный диапазон ("=Countries");
- адрес диапазона ячеек ("='Data' !\$E\$2:\$E\$6");
- значения, разделенные точкой с запятой ("UK; Italy; Germany; Austria; Brazil").

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата ("=TODAY()-7");
- именованный диапазон, состоящий из одной ячейки с датой ("=StartDate");
- адрес ячейки ("='Data' !C2");
- дата в поддерживаемом формате ("31.12.2024").

Вызов

```
string getFormula1()
```

Возвращает

— первый аргумент проверки данных, тип string.

7.40.7 Метод `DataValidation:getFormula2`

Метод возвращает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов [Between](#) и [NotBetween](#). Второй аргумент может принимать следующие значения:

- формула, результатом которой является дата ("`=TODAY()+7`");
- именованный диапазон, состоящий из одной ячейки с датой ("`=EndDate`");
- адрес ячейки ("`'Data'!C2`");
- дата в поддерживаемом формате ("`31.12.2024`").

Вызов

```
string getFormula2()
```

Возвращает

– второй аргумент проверки данных, тип `string`.

7.40.8 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](#).

7.40.9 Метод `DataValidation:getPrompt`

Метод возвращает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
string getPrompt()
```

Возвращает

– текст подсказки, тип `string`.

7.40.10 Метод `DataValidation:getPromptTitle`

Метод возвращает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
string getPromptTitle()
```

Возвращает

– заголовок подсказки, тип string.

7.40.11 Метод **DataValidation:showDropDown**

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.

7.40.12 Метод **DataValidation:showErrorMessage**

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.

7.40.13 Метод **DataValidation:showInputMessage**

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.

7.40.14 Метод **DataValidation:getType**

Метод возвращает тип проверки данных.

Вызов

```
DataValidationType getType()
```

Возвращает

– тип проверки данных, тип [DataValidationType](#).

7.40.15 Метод `DataValidation:isEmpty`

Метод позволяет определить наличие заданных настроек проверки данных в текущем объекте [DataValidation](#).

Вызов

```
bool isEmpty()
```

Возвращает

– true, если объект не содержит заданных настроек проверки данных, в ином случае – false.

Пример

```
local cell = sheet:getCell("C4")
local validation = cell:getDataValidation()
if not validation:isEmpty() then
    validation:clear()
end
```

7.40.16 Метод `DataValidation:setAllowBlank`

Метод позволяет разрешить ввод пустых значений.

Вызов

```
setAllowBlank(allowBlank)
```

Параметры

– allowBlank: true, чтобы разрешить ввод пустых значений, в ином случае – false.

7.40.17 Метод `DataValidation:setErrorMessage`

Метод задает текст сообщения об ошибке.

Вызов

```
setErrorMessage(errorMessage)
```

Параметры

– errorMessage: текст сообщения об ошибке, тип string.

Пример

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки сообщения об ошибке:
```

```
validation:setShowErrorMessage(true)
validation:setErrorStyle(DocumentAPI.DataValidationErrorStyle_Stop)
validation:setErrorTitle("Error")
validation:setErrorMessage("Invalid value. Choose one of the values below:")

cellRange:setDataValidation(validation)
```

7.40.18 Метод `DataValidation:setErrorStyle`

Метод задает поведение редактора при вводе недопустимых значений.

Вызов

```
setErrorStyle(errorStyle)
```

Параметры

– `errorStyle`: поведение редактора при вводе недопустимых значений, тип [DataValidationErrorStyle](#).

Пример

```
local validation = DocumentAPI.DataValidation()
-- ...
-- Настройки сообщения об ошибке:
validation:setShowErrorMessage(true)
validation:setErrorStyle(DocumentAPI.DataValidationErrorStyle_Stop)
validation:setErrorTitle("Error")
validation:setErrorMessage("Invalid value. Choose one of the values below:")

cellRange:setDataValidation(validation)
```

7.40.19 Метод `DataValidation:setErrorTitle`

Метод задает заголовок сообщения об ошибке.

Вызов

```
setErrorTitle(errorTitle)
```

Параметры

– `errorTitle`: заголовок сообщения об ошибке, тип `string`.

Пример

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



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

cellRange:setDataValidation(validation)
```

7.40.20 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' !E2:E6)"`);
- именованный диапазон (`"=Countries"`);
- адрес диапазона ячеек (`"='Data' !E2:E6"`);
- значения, разделенные точкой с запятой (`"UK; Italy; Germany; Austria; Brazil"`).

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()-7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=StartDate"`);
- адрес ячейки (`"='Data' !C2"`);
- дата в поддерживаемом формате (`"31.12.2024"`).

Вызов

```
setFormula1(formula1)
```

Параметры

– `formula1`: первый аргумент проверки данных, тип `string`.

Пример

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

cellRange:setDataValidation(dvList)
```

7.40.21 Метод DataValidation:setFormula2

Метод задает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов [Between](#) и [NotBetween](#). Второй аргумент может принимать

следующие значения:

- формула, результатом которой является дата ("=TODAY()+7");
- именованный диапазон, состоящий из одной ячейки с датой ("=EndDate");
- адрес ячейки ("='Data'!C2");
- дата в поддерживаемом формате ("31.12.2024").

Вызов

```
setFormula2(formula2)
```

Параметры

– formula2: второй аргумент проверки данных, тип string.

Пример

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

7.40.22 Метод DataValidation:setOperator

Метод задает оператор сравнения введенного значения с аргументом/аргументами. Используется только при проверке дат.

Вызов

```
setOperator(dataValidationOperator)
```

Параметры

– dataValidationOperator: оператор сравнения, тип [DataValidationOperator](#).

Пример

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

7.40.23 Метод `DataValidation:setPrompt`

Метод задает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
setPrompt(prompt)
```

Параметры

– `prompt`: текст подсказки, тип `string`.

Пример

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

7.40.24 Метод `DataValidation:setPromptTitle`

Метод задает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
setPromptTitle(promptTitle)
```

Параметры

– `promptTitle`: заголовок подсказки, тип `string`.

Пример

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

7.40.25 Метод `DataValidation:setShowDropDown`

Метод определяет показывать ли выпадающий список значений. Используется только при проверке данных по списку значений.

Вызов

```
setShowDropDown(showDropDown)
```

Параметры

– `showDropDown`: `true`, чтобы показывать выпадающий список значений, в ином случае – `false`.

Пример

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

7.40.26 Метод `DataValidation:setShowErrorMessage`

Метод определяет показывать ли сообщение об ошибке при вводе недопустимых значений.

Вызов

```
setShowErrorMessage(showErrorMessage)
```

Параметры

– `showErrorMessage`: `true`, чтобы показывать сообщение об ошибке, в ином случае – `false`.

Пример

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

7.40.27 Метод `DataValidation:setShowInputMessage`

Метод определяет показывать ли подсказку при вводе данных в ячейку с проверкой.

Вызов

```
setShowInputMessage( showInputMessage )
```

Параметры

– `showInputMessage`: `true`, чтобы показывать подсказку, в ином случае – `false`.

Пример

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

7.40.28 Метод `DataValidation:setType`

Метод задает тип проверки данных.

Вызов

```
setType( type )
```

Параметры

– `type`: тип проверки данных, тип [DataValidationType](#).

Пример

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

7.41 Таблица `DocumentAPI.DataValidationErrorStyle`

Таблица `DataValidationErrorStyle` определяет поведение редактора при вводе недопустимых значений. Используется в методах [DataValidation:getErrorStyle\(\)](#) и [DataValidation:setErrorStyle\(\)](#).

Таблица 24 – Описание поведений редактора при вводе недопустимых значений

Наименование константы	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>

7.42 Таблица `DocumentAPI.DataValidationOperator`

Таблица `DataValidationOperator` определяет оператор сравнения введенной даты с аргументом/аргументами. Используется в методах [DataValidation:getOperator\(\)](#) и [DataValidation:setOperator\(\)](#).

Таблица 25 – Описание операторов сравнения дат

Наименование константы	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
<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_LessThanOrEqual</code>	Дата должна быть раньше первого аргумента или совпадать с ним
<code>DataValidationOperator_GreaterThan</code>	Дата должна быть позже первого аргумента
<code>DataValidationOperator_GreaterThanOrEqual</code>	Дата должна быть позже первого аргумента или совпадать с ним

7.43 Таблица `DocumentAPI.DataValidationResult`

Таблица `DataValidationResult` представляет собой результат проверки значения ячейки (см. [Проверка данных](#)). Используется в методе [Cell:checkDataValidation\(\)](#).

7.43.1 Метод `DataValidationResult:getDataValidation`

Метод возвращает настройки проверки данных, если ячейка содержит недопустимое значение.

Вызов

```
DataValidation getDataValidation()
```

Возвращает

- настройки проверки данных, тип [DataValidation](#).
- nil, если ячейка содержит допустимое значение.

Пример

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

7.43.2 Метод DataValidationResult.isValid

Метод возвращает является ли значение ячейки допустимым при текущих настройках проверки данных. При проверке данных по списку текстовых значений, необходимо установить формат ячейки **Текст**.

Вызов

```
bool isValid()
```

Возвращает

- true, если значение ячейки допустимо или проверка данных отсутствует, в ином случае – false.

Пример

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

7.44 Таблица DocumentAPI.DataValidationType

Таблица DataValidationType определяет тип проверки данных. Используется в методах [DataValidation:getType\(\)](#) и [DataValidation:setType\(\)](#).

Таблица 26 – Описание типов проверки данных

Наименование константы	Описание
DataValidationType_None	Проверка данных отсутствует
DataValidationType_Integer	Не поддерживается
DataValidationType_Fractional	Не поддерживается
DataValidationType_List	Проверка данных по списку доступных значений
DataValidationType_Date	Проверка данных в формате Дата
DataValidationType_Time	Не поддерживается
DataValidationType_TextLength	Не поддерживается
DataValidationType_Custom	Не поддерживается

7.45 Таблица DocumentAPI.DatePatterns

Форматы даты представлены в таблице 27. Пример использования см. в главе [DocumentAPI.DateTimeCellFormatting](#).

Таблица 27 – Форматы даты

Наименование константы	Описание
DocumentAPI.DatePatterns_DayMonthTextLongYearLong	'mmmm dd, yyyy' для языка en_US
DocumentAPI.DatePatterns_FullDate	'день недели, mmmm dd, yyyy' для языка en_US
DocumentAPI.DatePatterns_DayMonthNumberLongYearLong	'mm/dd/yyyy' для языка en_US
DocumentAPI.DatePatterns_DayMonthNumberLongYearShort	'mm/dd/yy' для языка en_US
DocumentAPI.DatePatterns_DayMonthNumberShortYearShort	'm/dd/yy' для языка en_US
DocumentAPI.DatePatterns_DayMonthTextShort	'dd-mmm' для языка en_US
DocumentAPI.DatePatterns_MonthTextShortYearShort	'mmm-yy' для языка en_US
DocumentAPI.DatePatterns_DayMonthTextShortYearShort	'mmm dd, yy' для языка en_US
DocumentAPI.DatePatterns_DayMonthYearLongNonLocalizableHyphen	Нелокализуемый шаблон 'dd-mm-yyyy'

Наименование константы	Описание
DocumentAPI.DatePatterns_DayMonthYearLongNonLocalizableSlash	Нелокализуемый шаблон 'dd/mm/yyyy'

7.46 Таблица DocumentAPI.DatePickerControl

Представляет собой поле с календарем, используемое для ввода дат в документе.

Является наследником таблицы [ContentControl](#). Методы

`DatePickerControl:getValue()` и `DatePickerControl:setValue(DateTime)`

позволяют получить и задать значение этого элемента управления.

Пример

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

7.47 Таблица DocumentAPI.DateTime

Таблица `DocumentAPI.DateTime` предоставляет дату и время с точностью до секунды. Используется для поля `TrackedChangeInfo.timeStamp`. Описание полей таблицы `DocumentAPI.DateTime` представлено в таблице 28.

Таблица 28 – Описание полей таблицы `DocumentAPI.DateTime`

Поле	Тип	Описание
DocumentAPI.DateTime.year	Дата	Год
DocumentAPI.DateTime.month	Дата	Месяц
DocumentAPI.DateTime.day	Дата	День
DocumentAPI.DateTime.hour	Время	Часы
DocumentAPI.DateTime.minute	Время	Минуты
DocumentAPI.DateTime.second	Время	Секунды

7.47.1 Метод `DateTime:__eq`

Метод используется для определения эквивалентности двух значений времени.

7.48 Таблица DocumentAPI.DateTimeCellFormatting

Таблица содержит параметры для формата ячеек таблицы типа Дата и Время, используется в качестве аргумента метода [Cell:setFormat\(\)](#). Описание полей таблицы DocumentAPI.DateTimeCellFormatting представлено в таблице 29.

Таблица 29 – Описание полей таблицы DocumentAPI.DateTimeCellFormatting

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

Пример

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

local dateTimeCellFormatting = DocumentAPI.DateTimeCellFormatting()
dateTimeCellFormatting.dateListID = DocumentAPI.DatePatterns_FullDate
dateTimeCellFormatting.timeListID = DocumentAPI.TimePatterns_LongTime

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

7.49 Таблица DocumentAPI.DateTimeFormat

В таблице 30 представлены варианты форматирования даты и времени. Используется в качестве параметра метода [CellRange.insertCurrentDateTime\(\)](#).

Таблица 30 – Варианты форматирования даты и времени

Наименование константы	Описание
DocumentAPI.DateTimeFormat_DateTime	Дата и время
DocumentAPI.DateTimeFormat_Date	Дата
DocumentAPI.DateTimeFormat_Time	Время

7.50 Таблица DocumentAPI.document

Таблица DocumentAPI.Document осуществляет доступ к содержимому открытого текстового или табличного документа.

Пример

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

7.50.1 Метод `document:areMirroredMarginsEnabled`

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

Пример

```
document:setMirroredMarginsEnabled(true)
print(document:areMirroredMarginsEnabled())
```

7.50.2 Метод `document:calculateAllFormulas`

Метод пересчитывает все формулы в документе. Используйте этот метод после изменения данных табличного документа, если функция автоматического пересчета формул отключена. Чтобы узнать текущее состояние этой функции, воспользуйтесь методом [Document:getCalculationMode\(\)](#).

Также вы можете использовать метод [Document:calculateOutdatedFormulas\(\)](#) для пересчета только формул, данные которых были изменены, и метод `calculate()` объектов [Table](#), [CellRange](#) или [Cell](#) для пересчета формул, находящихся в определенном диапазоне.

Пример

```
-- пересчет выделенного диапазона ячеек
EditorAPI.getSelection():calculate()

-- пересчет заданного диапазона ячеек
document:getBlocks():getTable(0):getCellRange("A2:A6"):calculate()

-- пересчет всего листа
document:getBlocks():getTable(0):calculate()

-- пересчет всего документа
document:calculateOutdatedFormulas()
```

7.50.3 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.

7.50.4 Метод `document:disableEvents`

Метод отключает обработку событий табличного документа (см. [Обработка событий табличного документа](#)).

Вызов

```
disableEvents()
```

Метод [enableEvents\(\)](#) позволяет включить обработку событий документа. Вы можете использовать метод [isEventsEnabled\(\)](#), чтобы узнать текущее состояние обработки событий.

7.50.5 Метод `document:enableEvents`

Метод включает обработку событий табличного документа (см. [Обработка событий табличного документа](#)).

Вызов

```
enableEvents()
```

Метод [disableEvents\(\)](#) позволяет отключить обработку событий документа. Вы можете использовать метод [isEventsEnabled\(\)](#), чтобы узнать текущее состояние обработки событий.

7.50.6 Метод `document:enumerateSections`

Возвращает таблицу объектов типа [DocumentAPI.Section](#).

Пример

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

7.50.7 Метод `document:getAbsolutePath`

Метод возвращает строку, содержащую абсолютный путь к текущему документу. Получаемый путь имеет ОС - зависимый формат (например, содержит символы "/" для Unix и "\" для Windows).

Пример

```
local documentPath = document:getAbsolutePath()  
print(documentPath)
```



Ограничения:

- Если документ был создан, но не сохранен, данный метод вернет пустую строку;
- Абсолютный путь может быть получен только для локальных файлов и не будет доступен для получения пути хранения облачного документа;
- В текущей реализации отсутствует возможность полноценного использования метода при совместном редактировании.

7.50.8 Метод `document:getBlocks`

Метод предоставляет доступ к таблице [DocumentAPI.Blocks](#) и далее к отдельным фрагментам (абзацам, таблицам и т. д.), из которых состоит документ.

Пример

```
local blocks = document:getBlocks()
```

7.50.9 Метод `document:getBookmarks`

Метод предоставляет доступ к таблице закладок [DocumentAPI.Bookmarks](#).
Используется только в текстовых документах.

Пример

```
local bookmarks = document:getBookmarks()
```

7.50.10 Метод `document:getCalculationMode`

Метод возвращает текущий режим пересчета формул в документе [CalculationMode](#).
Чтобы изменить этот режим, используйте метод [Document:setCalculationMode\(\)](#).

7.50.11 Метод `document:getComments`

Метод обеспечивает доступ к комментариям документа, возвращает таблицу [DocumentAPI.Comments](#).
Используется только в текстовых документах.

Пример

```
local comments = document:getComments()  
for comment in comments:enumerate() do  
    print(comment:getRange())  
    print(comment:getText())  
end
```

```
print(comment:getInfo().author)
print(comment:getInfo().timeStamp)
print(comment:isResolved())
print(comment:getReplies())
end
```

7.50.12 Метод document:getContentControls

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

Пример

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

7.50.13 Метод document:getFormulaType

Метод возвращает поддерживаемую адресацию ячеек `DocumentAPI.FormulaType` документа.

Пример

```
document:setFormulaType(DocumentAPI.FormulaType_R1C1)
print(document:getFormulaType())
```

7.50.14 Метод document:getNamedExpressions

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

Пример

```
namedExpressions = document:getNamedExpressions()
print(namedExpressions)
```

7.50.15 Метод document:getPivotTablesManager

Возвращает объект [DocumentAPI.PivotTablesManager](#), который используется для создания сводных таблиц. Метод может быть использован только в табличном редакторе.

Пример

```
local pivotTablesManager = document:getPivotTablesManager()
print(pivotTablesManager)
```

7.50.16 Метод `document:getRange`

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

Пример

```
local range = document:getRange()  
print(range:extractText())
```

7.50.17 Метод `document:getScripts`

Метод предоставляет доступ к таблице макрокоманд [DocumentAPI.Scripts](#), содержащихся в документе.

Пример

```
local scripts = document:getScripts()  
for script in document:getScripts():enumerate() do  
    print(script:getName())  
end
```

7.50.18 Метод `document:getSections`

Возвращает таблицу объектов типа [DocumentAPI.Sections](#).

Пример

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

7.50.19 Метод `document:isCalculatedOnSave`

Метод возвращает состояние функции пересчета формул при сохранении документа. Используйте метод [Document:setCalculatedOnSave\(\)](#), чтобы включить или отключить эту функцию.

7.50.20 Метод `document:isChangesTrackingEnabled`

Метод возвращает текущее состояние отслеживания изменений в документе (true - включены). Используется только в текстовых документах.

Пример

```
local trackingChanges = "Disabled"  
if document:isChangesTrackingEnabled() then
```

```
trackingChanges = "Enabled"  
end  
print(trackingChanges)
```

7.50.21 Метод document:isEventsEnabled

Метод возвращает состояние обработки событий для текущего документа (см. [Обработка событий табличного документа](#)).

Вызов

```
bool isEventsEnabled()
```

Методы [disableEvents\(\)](#) и [enableEvents\(\)](#) позволяют отключить и включить обработку событий документа.

7.50.22 Метод document:isStructureProtected

Метод возвращает состояние защиты от изменения структуры табличного документа.

Вызов

```
bool isStructureProtected()
```

Методы [setStructureProtection\(\)](#) и [removeStructureProtection\(\)](#) позволяют установить и снять защиту от изменений структуры.

7.50.23 Метод document:removeStructureProtection

Метод снимает защиту от изменений структуры табличного документа.

Вызов

```
removeStructureProtection(password)
```

Параметры

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

Пример

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

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

7.50.24 Метод `document:setCalculatedOnSave`

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

Пример

```
document:setCalculatedOnSave(false)
```

7.50.25 Метод `document:setCalculationMode`

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

Пример

```
document:setCalculationMode(DocumentAPI.CalculationMode_Manual)
```

7.50.26 Метод `document:setChangesTrackingEnabled`

Метод управляет состоянием отслеживания изменений в документе (включены или выключены). Используется только в текстовых документах.

Пример

```
if trackingChanges == "Disabled" then  
    document:setChangesTrackingEnabled(true)  
    if document:isChangesTrackingEnabled() then  
        trackingChanges = "Enabled"  
    end  
end
```

7.50.27 Метод `document:setFormulaType`

Метод устанавливает поддерживаемую адресацию ячеек `DocumentAPI.FormulaType` документа.

Пример

```
document:setFormulaType(DocumentAPI.FormulaType_A1)
```

7.50.28 Метод `document:setMirroredMarginsEnabled`

Метод позволяет включать и выключать зеркальные поля в документе.

Пример

```
document:setMirroredMarginsEnabled(true)  
print(document:areMirroredMarginsEnabled())
```

7.50.29 Метод `document:setPageOrientation`

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

Пример

```
document:setPageOrientation(DocumentAPI.PageOrientation_Landscape)
```

7.50.30 Метод `document:setPageProperties`

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

Пример

```
local properties = DocumentAPI.PageProperties()  
properties.width = 100  
properties.height = 200  
document:setPageProperties(properties)
```

7.50.31 Метод `document:setStructureProtection`

Метод устанавливает защиту от изменений структуры табличного документа.

Вызов

```
setStructureProtection(password)
```

Параметры

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

Пример

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

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

7.51 Таблица DocumentAPI.DropListControl

Представляет собой поле с выпадающим списком в документе. Является наследником таблицы [ContentControl](#). Методы `DropListControl:getValue()` и `DropListControl:setValue(int)` позволяют получить и задать значение этого элемента управления. Метод `DropListControl:getChoices()` возвращает коллекцию элементов, находящихся в выпадающем списке.

Пример

```
local controls = document:getContentControls()  
local comboBox = controls:findByTitle("select"):toDropList()  
comboBox:setValue(2)
```

7.52 Таблица DocumentAPI.Field

Таблица `Field` предназначена для реализации некоторых полей, например, содержания.

7.53 Таблица DocumentAPI.Fill

Таблица описывает свойства заполнения для [ShapeProperties](#), [CellProperties](#).

Варианты заполнения:

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

Примеры

```
-- Без заполнения  
shapeProperties.fill = DocumentAPI.Fill()
```

```
-- Заполнение цветом  
shapeProperties.fill =  
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(55, 146, 179, 200)))
```

```
-- Заполнение шаблоном из url  
shapeProperties.fill = DocumentAPI.Fill("https://fillpattern.url")
```

7.53.1 Метод `Fill:getColor`

Метод возвращает цвет заполнения [DocumentAPI.Color](#).

7.53.2 Метод `Fill:getUrl`

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

7.53.3 Метод `Fill:isNoFill`

Метод возвращает `true`, если заполнения нет.

7.54 Таблица `DocumentAPI.FiltersRange`

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

7.54.1 Метод `FiltersRange:clear`

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

Пример

```
local filtersRange = table:createFiltersRange(range)
.....
tableFilters:clear()
```

7.54.2 Метод `FiltersRange:eraseFilters`

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

Пример

```
local filtersRange = table:createFiltersRange(range)
.....
tableFilters:eraseFilters()
```

7.54.3 Метод `FiltersRange:getCellRange`

Метод `FiltersRange:getCellRange` возвращает диапазон ячеек [CellRange](#), содержащий текущий объект фильтрации. Возвращаемый диапазон включает строку заголовка.

Пример

```
local cellRange = filtersRange:getCellRange()  
print(cellRange:getBeginRow() .. ", " .. cellRange:getLastRow())
```

Более подробный пример приведен в разделе [Работа с фильтрами](#).

7.54.4 Метод `FiltersRange:setFilters`

Метод `FiltersRange:setFilters` устанавливает фильтры [TableFilters](#) для столбцов диапазона. Фильтрация выполняется с использованием логической операции AND по столбцам. Нумерация столбцов начинается относительно левой позиции диапазона фильтрации. Если номер столбца в фильтрах превышает диапазон фильтрации, то фильтр будет успешно добавлен, но фильтрация для этого столбца будет пропущена. Такое поведение полезно, если необходимо изменить размер диапазона фильтрации при этом сохранить ранее определенные фильтры. Диапазон фильтрации должен существовать до вызова этого метода. В настоящее время `DocumentAPI` позволяет создавать диапазон фильтрации только для всего листа табличного документа. Для создания или изменения размера существующего диапазона используется метод [Table:createFiltersRange\(\)](#).

Пример

```
local filtersRange = table:createFiltersRange(range)  
.....  
local tableFilters = DocumentAPI.TableFilters()  
tableFilters:setFilter(0, johnPaulFilter)  
tableFilters:setFilter(1, songFilter)  
.....  
filtersRange:setFilters(tableFilters)
```

Более подробный пример приведен в разделе [Работа с фильтрами](#).

7.55 Таблица `DocumentAPI.FractionCellFormatting`

Таблица содержит параметры для дробного формата ячеек таблицы. Данная таблица используется в качестве аргумента метода [Cell:setFormat\(\)](#). Описание полей таблицы `DocumentAPI.FractionCellFormatting` представлено в таблице 31.

Таблица 31 – Описание полей таблицы `DocumentAPI.FractionCellFormatting`

Поле	Описание
<code>DocumentAPI.FractionCellFormatting.minNumerator</code>	Количество позиций числителя

Поле	Описание
orDigits	
DocumentAPI.FractionCellFormatting.minDenominatorDigits	Количество позиций знаменателя
DocumentAPI.FractionCellFormatting.denominatorValue	Знаменатель

Пример

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("B2")

local fractionCellFormatting = DocumentAPI.FractionCellFormatting()
fractionCellFormatting.minNumeratorDigits = 2
fractionCellFormatting.minDenominatorDigits = 3
fractionCellFormatting.denominatorValue = 22

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

7.56 Таблица DocumentAPI.FrozenRangePosition

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

7.56.1 Конструкторы

Конструктор с параметрами по умолчанию.

```
FrozenRangePosition()
```

Конструктор, создающий диапазон ячеек. В качестве параметров используются координаты левой верхней и правой нижней точек области.

```
FrozenRangePosition(top, left, bottom, right)
```

Примеры

```
frozenRangePosition = DocumentAPI.FrozenRangePosition()
print(frozenRangePosition:isRowsCols())

frozenRangePosition = DocumentAPI.FrozenRangePosition(0, 2, 5, 5)
print(frozenRangePosition:isRowsCols())
```

7.56.2 Метод FrozenRangePosition:create

Создает объект заблокированной области таблицы FrozenRangePosition. В качестве параметров используются координаты левой верхней и правой нижней точек области.

Вызов

```
FrozenRangePosition create(top, left, bottom, right)
```

Пример

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

7.56.3 Метод FrozenRangePosition:createFrozenArea

Создает объект заблокированной области таблицы FrozenRangePosition. Область содержит все ячейки прямоугольника {0, 0, bottom, right}.

Вызов

```
FrozenRangePosition createFrozenArea(bottom, right)
```

Пример

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

7.56.4 Метод FrozenRangePosition:createFrozenCols

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

Вызов

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

Пример

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

7.56.5 Метод FrozenRangePosition:createFrozenRows

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

Вызов

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

Пример

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

7.56.6 Метод FrozenRangePosition:isArea

Возвращает true если диапазон является непрерывной областью.

Пример

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

7.56.7 Метод FrozenRangePosition:isCols

Возвращает true если диапазон состоит из колонок.

Пример

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

7.56.8 Метод FrozenRangePosition:isRows

Возвращает true если диапазон состоит из строк.

Пример

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

7.56.9 Метод FrozenRangePosition:isRowsCols

Возвращает true если диапазон содержит строки и колонки.

Пример

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

7.56.10 Метод FrozenRangePosition:__eq

Метод используется для определения эквивалентности двух объектов FrozenRangePosition.

Пример

```
local pos1 = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local pos2 = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
print(pos1:__eq(pos2)) -- true
```

7.57 Таблица DocumentAPI.HeaderFooter

Таблица `DocumentAPI.HeaderFooter` определяет колонтитул текстового документа.

7.57.1 Метод HeaderFooter:getBlocks

Метод предоставляет доступ к блокам ([DocumentAPI.Blocks](#)), которые содержатся в колонтитуле.

Пример

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    for block in header:getBlocks():enumerate() do
        print(block:getRange():extractText())
    end
end
```

7.57.2 Метод HeaderFooter:getRange

Метод предоставляет диапазон ([DocumentAPI.Range](#)) с содержанием верхнего или нижнего колонтитулов.

Пример

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    print(header:getRange():extractText())
end
```

7.57.3 Метод HeaderFooter:getType

Метод предоставляет информацию о типе колонтитула ([DocumentAPI.HeaderFooterType](#)).

Пример

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

7.58 Таблица DocumentAPI.HeaderFooterType

Типы колонтитулов представлены в таблице 32.

Таблица 32 – Типы колонтитулов

Наименование константы	Описание
DocumentAPI.HeaderFooterType_Header	Верхний колонтитул
DocumentAPI.HeaderFooterType_Footer	Нижний колонтитул

7.59 Таблица DocumentAPI.HeadersFooters

Таблица DocumentAPI.HeadersFooters представляет коллекцию верхних и нижних колонтитулов раздела текстового документа (см. Рисунок 35). Доступ к колонтитулам осуществляется посредством методов [Section.getHeaders\(\)](#), [Section.getFooters\(\)](#).

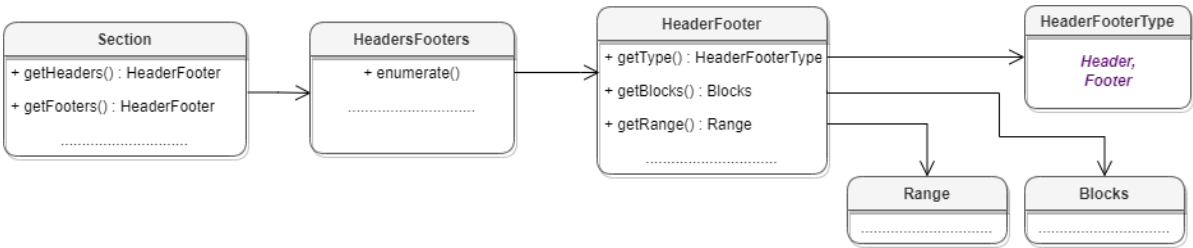


Рисунок 35 – Таблицы колонтитулов

7.59.1 Метод HeadersFooters:enumerate

Метод возвращает коллекцию колонтитулов.

Пример

```
local section = document:getBlocks():getBlock(0):getSection()  
local headers = section:getHeaders()  
for header in headers:enumerate() do
```

```
if (header:getType() == DocumentAPI.HeaderFooterType_Header) then
    print("Header") else print("Footer")
end
end
```

7.60 Таблица DocumentAPI.HorizontalAnchorAlignment

В таблице 33 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали. Используется в [DocumentAPI.HorizontalTextAnchoredPosition](#).

Таблица 33 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

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

7.61 Таблица DocumentAPI.HorizontalRelativeTo

В таблице 34 представлены типы размещения объекта относительно закрепленной позиции по горизонтали. Используется в [DocumentAPI.HorizontalTextAnchoredPosition](#).

Таблица 34 – Типы размещения объекта относительно закрепленной позиции по горизонтали

Наименование константы	Описание
DocumentAPI.HorizontalRelativeTo_Character	Символ
DocumentAPI.HorizontalRelativeTo_Column	Столбец
DocumentAPI.HorizontalRelativeTo_ColumnLeftMargin	Левое поле столбца
DocumentAPI.HorizontalRelativeTo_ColumnRightMargin	Правое поле столбца
DocumentAPI.HorizontalRelativeTo_ColumnInsideMargin	Внутреннее поле столбца
DocumentAPI.HorizontalRelativeTo_ColumnOutsideMargin	Внешнее поле столбца
DocumentAPI.HorizontalRelativeTo_Page	Страница
DocumentAPI.HorizontalRelativeTo_PageContent	Содержимое страницы
DocumentAPI.HorizontalRelativeTo_PageLeftMargin	Левое поле страницы

Наименование константы	Описание
DocumentAPI.HorizontalRelativeTo_PageRightMargin	Правое поле страницы
DocumentAPI.HorizontalRelativeTo_PageInsideMargin	Внутреннее поле страницы
DocumentAPI.HorizontalRelativeTo_PageOutsideMargin	Внешнее поле страницы

7.62 Таблица DocumentAPI.HorizontalTextAnchoredPosition

Таблица DocumentAPI.HorizontalTextAnchoredPosition (см. Рисунок 36) предназначена для управления относительным положением объекта со смещением или выравниванием по горизонтали. Пример использования см. в [InlineFrame.setPosition\(\)](#).

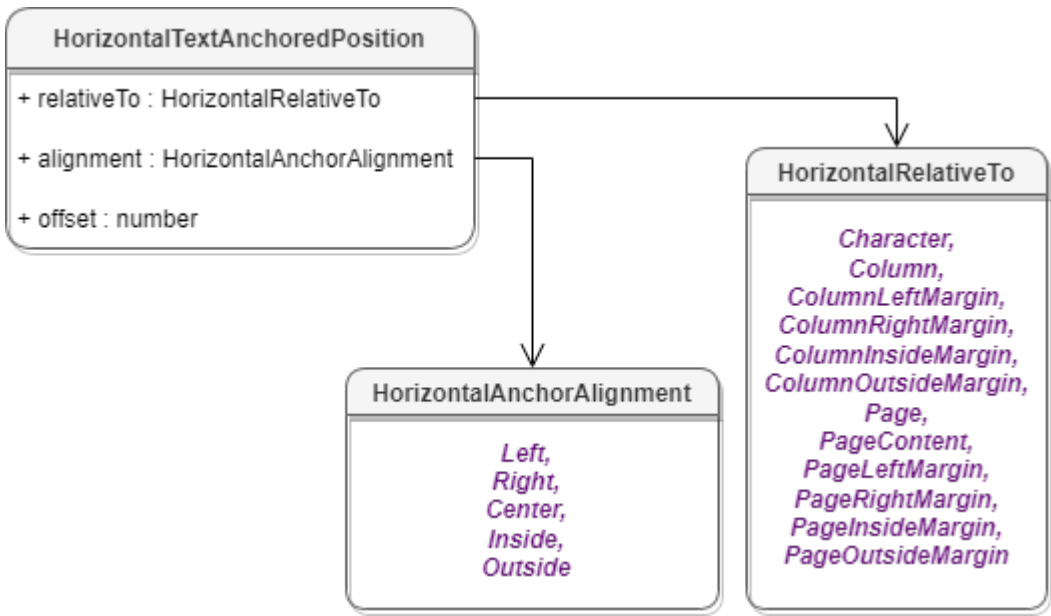


Рисунок 36 – Поля таблицы DocumentAPI.HorizontalTextAnchoredPosition

Описание полей таблицы DocumentAPI.HorizontalTextAnchoredPosition представлено в таблице 35.

Таблица 35 – Описание полей таблицы DocumentAPI.HorizontalTextAnchoredPosition

Поле	Описание
DocumentAPI.HorizontalTextAnchoredPosition.alignment	Тип выравнивания объекта относительно закрепленной позиции по HorizontalAnchorAlignment .
DocumentAPI.HorizontalTextAnchoredPosition.relativeTo	Тип размещения объекта относительно закрепленной позиции

Поле	Описание
	по горизонтали HorizontalRelativeTo .
<code>DocumentAPI.HorizontalTextAnchoredPosition.offset</code>	Смещение объекта.

7.62.1 Метод `HorizontalTextAnchoredPosition.__eq`

Метод используется для определения эквивалентности двух положений объекта по горизонтали.

Пример

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos1.horizontal.offset = 1

local pos2 = DocumentAPI.TextAnchoredPosition()
pos2.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos2.horizontal.offset = 1

print(pos1.horizontal:__eq(pos2.horizontal))
```

7.63 Таблица `DocumentAPI.Hyperlink`

Таблица `DocumentAPI.Hyperlink` описывает свойства ссылки. Может быть получена посредством вызова метода [Cell.getHyperlink\(\)](#).

Таблица 36 – Описание полей таблицы `DocumentAPI.Hyperlink`

Поле	Тип	Описание
<code>DocumentAPI.Hyperlink.url</code>	Строка	Адрес ссылки
<code>DocumentAPI.Hyperlink.tooltip</code>	Строка	Текст подсказки
<code>DocumentAPI.Hyperlink.label</code>	Строка	Текст описания

Пример

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

```
if (hyperlink ~= nil) then
    print(hyperlink.url, hyperlink.tooltip, hyperlink.label)
end
```

7.63.1 Метод Hyperlink:__eq

Метод используется для определения эквивалентности двух объектов Hyperlink.

Пример

```
table_0 = document:getBlocks():getTable(0)
cell_00 = table_0:getCell(DocumentAPI.CellPosition(0, 0))
cell_01 = table_0:getCell(DocumentAPI.CellPosition(0, 1))
local hyperlink_00 = cell_00:getHyperlink()
local hyperlink_01 = cell_01:getHyperlink()
if (hyperlink_00 and hyperlink_01) then
    print(hyperlink_00:__eq(hyperlink_01))
end
```

7.64 Таблица DocumentAPI.Image

Таблица DocumentAPI.Image представляет собой изображение, находящееся в текстовом или табличном документе.

7.64.1 Метод Image:getFrame

Метод аналогичен методу [MediaObject:getFrame\(\)](#), он возвращает свойства позиции изображения. В зависимости от текущего редактора метод возвращает разные типы рамок. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют тип [DocumentAPI.InlineFrame](#), табличные документы работают с абсолютной позицией и используют тип [DocumentAPI.AbsoluteFrame](#).

Пример для текстового документа

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    local image = mediaObject:toImage()
    if (image) then
        print(image:getFrame()) -- <userdata of type
'CO::API::Document::InlineFrame'>
    end
end
```

Пример для табличного документа

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
    local image = mediaObject:toImage()
    if (image) then
        print(image:getFrame()) -- <userdata of type
'CO::API::Document::AbsoluteFrame'>
    end
end
```

7.64.2 Метод Image:remove

Метод удаляет изображение из документа.

Пример для текстового документа

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    local image = mediaObject:toImage()
    if image then
        image:remove()
        break
    end
end
```

7.65 Таблица DocumentAPI.Images

Таблица `DocumentAPI.Images` используется для доступа к коллекции изображений.

Может быть получена вызовом методов [Table.getImages\(\)](#), [Range.getImages\(\)](#).

7.65.1 Метод Images:enumerate

Метод позволяет перечислить коллекцию изображений.

Пример для текстового документа

```
for image in EditorAPI.getSelection():getImages():enumerate() do
    print(image:getFrame():getWrapType())
end
```

Пример для табличного документа

```
local sheet = document:getBlocks():getTable(0)
local images = sheet:getImages()
for image in images:enumerate() do
```

```
print(image:getFrame():getTopLeft().x)
end
```

7.66 Таблица DocumentAPI.InlineFrame

Таблица `DocumentAPI.InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 37). Предназначена для получения и изменения свойств позиции графических объектов. Используется в текстовом документе.

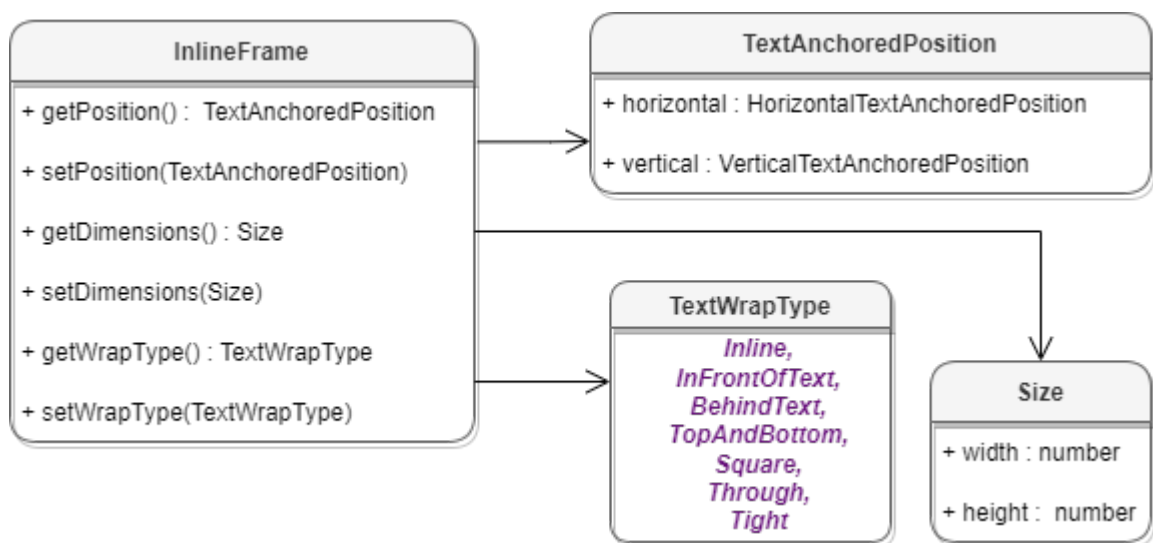


Рисунок 37 – Объектная модель таблицы `DocumentAPI.InlineFrame`

Пример для текстового документа

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    local inlineFrame = mediaObject:getFrame()
    print(inlineFrame:getDimensions())
    print(inlineFrame:getWrapType())
    print(inlineFrame:getPosition())
end
```

7.66.1 Метод `InlineFrame:getDimensions`

Метод возвращает задает размеры встроенного объекта, тип - [DocumentAPI.Size](#).

Пример

```
local mediaObjects = document:getRange():getImages()
for mediaObject in mediaObjects:enumerate() do
```



```
local inlineFrame = mediaObject:getFrame()  
local dimensions = inlineFrame:getDimensions()  
if (dimensions) then  
    print(dimensions.width, dimensions.height)  
end  
end
```

7.66.2 Метод `InlineFrame:getPosition`

Метод возвращает позицию встроенного объекта на странице в виде таблицы [DocumentAPI.TextAnchoredPosition](#).

Пример

```
local mediaObjects = document:getRange():getImages()  
for mediaObject in mediaObjects:enumerate() do  
    local inlineFrame = mediaObject:getFrame()  
    local textAnchoredPosition = inlineFrame:getPosition()  
    if (textAnchoredPosition) then  
        print(textAnchoredPosition.horizontal, textAnchoredPosition.vertical)  
    end  
end
```

7.66.3 Метод `InlineFrame:getWrapType`

Метод возвращает вариант обтекания текстом встроенного объекта (см. [DocumentAPI.TextWrapType](#)).

Пример

```
local mediaObjects = document:getRange():getImages()  
for mediaObject in mediaObjects:enumerate() do  
    print(mediaObject:getFrame():getWrapType())  
end
```

7.66.4 Метод `InlineFrame:setDimensions`

Метод задает размер [DocumentAPI.SizeU](#) встроенного объекта.

Пример

```
inlineFrame:setDimensions(DocumentAPI.SizeU(100, 100))
```

7.66.5 Метод `InlineFrame:setPosition`

Метод задает положение встроенного объекта, тип аргумента [DocumentAPI.TextAnchoredPosition](#). Новая позиция может быть установлена только для

встроенных объектов, тип переноса текста которых не является типом [DocumentAPI.TextWrapType Inline](#).

Пример

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    local inlineFrame = mediaObject:getFrame()
    if (inlineFrame:getWrapType() ~= DocumentAPI.TextWrapType_Inline) then
        local pos = DocumentAPI.TextAnchoredPosition()

        -- Установка смещения по горизонтали относительно края колонки
        pos.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
        pos.horizontal.offset = 10
        -- Установка смещения по вертикали относительно края страницы
        pos.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
        pos.vertical.offset = 10

        -- Установка позиции рамки графического объекта
        inlineFrame:setPosition(pos)
    end
end
```

7.66.6 Метод `InlineFrame:setWrapType`

Метод устанавливает вариант обтекания текстом встроенного объекта (см. [DocumentAPI.TextWrapType](#)).

Пример

```
local mediaObjects = document:getRange():getImages()
for mediaObject in mediaObjects:enumerate() do
    local inlineFrame = mediaObject:getFrame()
    inlineFrame:setWrapType(DocumentAPI.TextWrapType_InFrontOfText)
end
```

7.67 Таблица `DocumentAPI.InputFieldControl`

Представляет собой текстовое поле в документе. Является наследником таблицы [ContentControl](#). Методы `InputFieldControl:getValue()` и

`InputFieldControl:setValue(string)` позволяют получить и задать значение этого элемента управления.

Пример

```
local controls = document:getContentControls()  
local inputField = controls:findByTitle("input"):toInputField()  
local text = inputField:getValue():gsub(' ', '_')  
inputField:setValue(text)
```

7.68 Таблица `DocumentAPI.Insets`

Таблица `DocumentAPI.Insets` предназначена для задания полей, например, страницы. Данный тип используется в поле `margins` таблицы [DocumentAPI.PageProperties](#). Поля `DocumentAPI.Insets` представлены в таблице 37.

Таблица 37 – Описание полей таблицы `DocumentAPI.Insets`

Поле	Тип	Описание
<code>DocumentAPI.Insets.left</code>	number	Левая граница поля
<code>DocumentAPI.Insets.top</code>	number	Верхняя граница поля
<code>DocumentAPI.Insets.right</code>	number	Правая граница поля
<code>DocumentAPI.Insets.bottom</code>	number	Нижняя граница поля

Пример

```
local insets = DocumentAPI.Insets()  
insets.left = 10.0  
print(insets.left)
```

7.69 Таблица `DocumentAPI.LineEndingProperties`

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

Таблица 38 – Описание полей таблицы `DocumentAPI.LineEndingProperties`

Поле	Тип	Описание
<code>DocumentAPI.LineEndingProperties.style</code>	LineEndingStyle	Стиль окончания линии

Поле	Тип	Описание
DocumentAPI.LineEndingProperties.relativeExtent	Size	Размер окончания линии относительно ее ширины

Пример

```

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

lineProperties = DocumentAPI.LineProperties()
lineProperties.headLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.headLineEndingProperties.style =
DocumentAPI.LineEndingStyle_Arrow

lineProperties.headLineEndingProperties.relativeExtent = DocumentAPI.SizeU()
lineProperties.headLineEndingProperties.relativeExtent.width = 2
lineProperties.headLineEndingProperties.relativeExtent.height = 2

lineProperties.tailLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.tailLineEndingProperties.style =
DocumentAPI.LineEndingStyle_Arrow
lineProperties.tailLineEndingProperties.relativeExtent = DocumentAPI.SizeU()
lineProperties.tailLineEndingProperties.relativeExtent.width = 2
lineProperties.tailLineEndingProperties.relativeExtent.height = 2

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)

```

7.69.1 Метод LineEndingProperties: __eq

Метод используется для определения эквивалентности значений двух объектов LineEndingProperties.

Пример

```

lineEnding1 = DocumentAPI.LineEndingProperties()
lineEnding1.style = DocumentAPI.LineEndingStyle_Arrow
lineEnding2 = DocumentAPI.LineEndingProperties()
lineEnding2.style = DocumentAPI.LineEndingStyle_Diamond
print("Eq: " .. tostring(lineEnding1:__eq(lineEnding2)))

```

7.70 Таблица DocumentAPI.LineEndingStyle

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

Таблица 39 – Типы окончания линии

Наименование константы	Описание
DocumentAPI.LineEndingStyle_Arrow	
DocumentAPI.LineEndingStyle_Diamond	
DocumentAPI.LineEndingStyle_Oval	
DocumentAPI.LineEndingStyle_Stealth	
DocumentAPI.LineEndingStyle_Triangle	
DocumentAPI.LineEndingStyle_None	

Пример

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

lineProperties = DocumentAPI.LineProperties()
lineProperties.headLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.headLineEndingProperties.style = DocumentAPI.LineEndingStyle_Oval

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

7.71 Таблица DocumentAPI.LineProperties

Таблица DocumentAPI.LineProperties предназначена для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 38).

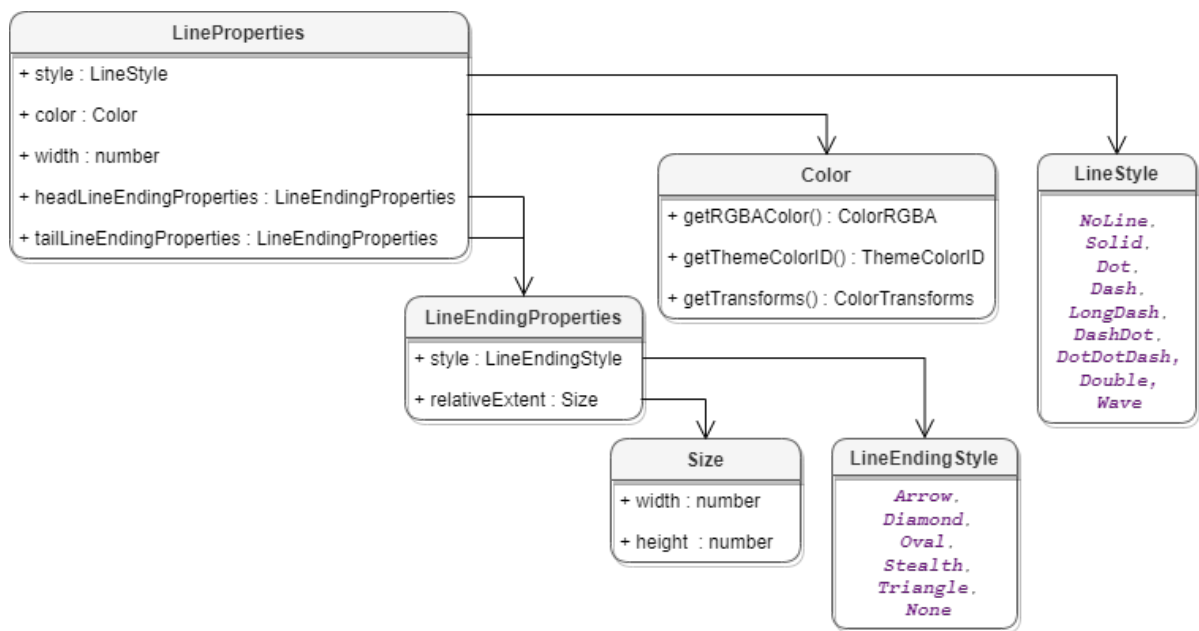


Рисунок 38 – Свойства границ ячеек

Пример

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

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

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
local brds = cell:setBorders(borders)
```

7.71.1 Поле LineProperties.color

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

7.71.2 Поле LineProperties.headLineEndingProperties

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

7.71.3 Поле `LineProperties.style`

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

7.71.4 Поле `LineProperties.tailLineEndingProperties`

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

7.71.5 Поле `LineProperties.width`

Поле предназначено для установки ширины линии. Тип - числовой.

7.71.6 Метод `LineProperties.__eq`

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

Пример

```
lineProperties1 = DocumentAPI.LineProperties()  
lineProperties1.style = DocumentAPI.LineStyle_Solid  
  
lineProperties2 = DocumentAPI.LineProperties()  
lineProperties2.style = DocumentAPI.LineStyle_Dot  
  
print("Eq: " .. tostring(lineProperties1.__eq(lineProperties2)))
```

7.72 Таблица `DocumentAPI.LineSpacing`

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

Таблица 40 – Параметры межстрочного интервала

Поле	Описание
<code>DocumentAPI.LineSpacing.value</code>	Значение межстрочного интервала.
<code>DocumentAPI.LineSpacing.rule</code>	Правило формирования межстрочного интервала DocumentAPI.LineSpacingRule .

Пример

```
-- Конструктор  
local lineSpacing = DocumentAPI.LineSpacing(1.5,
```

```
DocumentAPI.LineSpacingRule_Multiple)  
-- Обращение к полям  
lineSpacing.value = 1  
lineSpacing.rule = DocumentAPI.LineSpacingRule_Exact
```

7.73 Таблица DocumentAPI.LineSpacingRule

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

Таблица 41 – Виды межстрочного интервала

Наименование константы	Описание
DocumentAPI.LineSpacingRule_Multiple	<p>Установка произвольного межстрочного интервала с использованием множителя.</p> <p>При вызове необходимо указать значение множителя, например:</p> <pre>pPr.lineSpacing = DocumentAPI.LineSpacing(1.15, DocumentAPI.LineSpacingRule_Multiple)</pre> <p>В данном примере используется значение множителя 1.15.</p> <p>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «Произвольный интервал» (см. документ «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).</p> <div><p>Произвольный интервал</p><p>Межстрочный интервал:</p><div><div>Множитель</div><div>1,15</div></div><div><div>OK</div><div>Отмена</div></div></div>
DocumentAPI.LineSpacingRule_Exact	<p>Установка произвольного межстрочного интервала с использованием точного значения.</p> <p>При вызове необходимо указать точное значение, например:</p> <pre>pPr.lineSpacing = DocumentAPI.LineSpacing(12.0, DocumentAPI.LineSpacingRule_Exact)</pre> <p>В данном примере используется точное значение 12.0.</p> <p>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «Произвольный интервал» (см. документ «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).</p>

Наименование константы	Описание
	<div><p>Произвольный интервал</p><p>Межстрочный интервал:</p><div><div>Точно</div><div>12,00 пт</div></div><div><div>OK</div><div>Отмена</div></div></div>
DocumentAPI.LineSpacingRule_AtLeast	<p>Установка произвольного межстрочного интервала с использованием минимального значения.</p> <p>При вызове необходимо указать минимальное значение, например:</p> <pre>pPr.lineSpacing = DocumentAPI.LineSpacing(12.0, DocumentAPI.LineSpacingRule_AtLeast)</pre> <p>В данном примере используется точное значение 12.0.</p> <p>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «Произвольный интервал» (см. документ «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).</p> <div><p>Произвольный интервал</p><p>Межстрочный интервал:</p><div><div>Минимум</div><div>12,00 пт</div></div><div><div>OK</div><div>Отмена</div></div></div>









Пример

```
paragraph = document:getBlocks():getParagraph(0)
props = paragraph:getParagraphProperties()
props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
paragraph:setParagraphProperties(props)
```

7.74 Таблица DocumentAPI.LineStyle

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

Таблица 42 – Типы линий

Наименование константы	Описание
DocumentAPI.LineStyle_NoLine	Нет линии
DocumentAPI.LineStyle_Solid	
DocumentAPI.LineStyle_Dot	
DocumentAPI.LineStyle_Dash	
DocumentAPI.LineStyle_LongDash	
DocumentAPI.LineStyle_DashDot	
DocumentAPI.LineStyle_DotDotDash	
DocumentAPI.LineStyle_Double	
DocumentAPI.LineStyle_Wave	

Пример

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

lineProperties = DocumentAPI.LineProperties()
lineProperties.style = DocumentAPI.LineStyle_Wave

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

7.75 Таблица DocumentAPI.ListSchema

Типы схем форматирования списков, которые могут быть применены к абзацам текста, представлены в таблице 43. Данные константы используются в методах [Paragraph:getListSchema\(\)](#), [Paragraph:setListSchema\(\)](#).

Таблица 43 – Типы схем форматирования списков

Наименование константы	Описание
DocumentAPI.ListSchema_Unknown	Схема не определена
DocumentAPI.ListSchema_UnknownBullet	Список без маркера
DocumentAPI.ListSchema_UnknownNumbering	Нумерация без номера
DocumentAPI.ListSchema_BulletCircleSolid	Список с маркерами в виде заполненного круга
DocumentAPI.ListSchema_BulletCircleContour	Список с маркерами в виде окружности
DocumentAPI.ListSchema_BulletSquareSolid	Список с маркерами в виде квадрата
DocumentAPI.ListSchema_BulletDiamondDots	Список с маркерами в виде четырех ромбов
DocumentAPI.ListSchema_BulletHyphen	Список с маркерами в виде дефиса
DocumentAPI.ListSchema_BulletConcaveArrowSolid	Список с маркерами в виде вогнутой стрелки.
DocumentAPI.ListSchema_BulletCheckmark	Список с маркерами в виде галочки.
DocumentAPI.ListSchema_EnumeratorDecimalDot	Десятичная нумерация с точкой.
DocumentAPI.ListSchema_EnumeratorDecimalDotMultiLevel	Многоуровневая десятичная нумерация с точкой
DocumentAPI.ListSchema_EnumeratorDecimalBracket	Десятичная нумерация со скобкой
DocumentAPI.ListSchema_EnumeratorLatinUppercaseDot	Нумерация латинскими прописными буквами с точкой
DocumentAPI.ListSchema_EnumeratorLatinLowercaseDot	Нумерация латинскими строчными буквами с точкой
DocumentAPI.ListSchema_EnumeratorLatinLowercaseBracket	Нумерация латинскими строчными буквами со скобкой
DocumentAPI.ListSchema_EnumeratorRomanUppercaseDot	Нумерация римскими прописными цифрами с точкой
DocumentAPI.ListSchema_EnumeratorRomanLowercaseDot	Нумерация римскими строчными цифрами с точкой
DocumentAPI.ListSchema_EnumeratorDecimalRussianBracket	Десятичная нумерация через запятую со скобкой
DocumentAPI.ListSchema_EnumeratorRussianLowercaseBracket	Нумерация с русскими строчными буквами со скобкой

Пример

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

7.76 Таблица DocumentAPI.MediaObject

Таблица `DocumentAPI.MediaObject` представляет собой встроенный объект документа.

7.76.1 Метод MediaObject:getFrame

Метод возвращает свойства позиции встроенного объекта. В зависимости от текущего редактора метод возвращает разные типы таблиц. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют таблицу [DocumentAPI.InlineFrame](#), табличные документы работают с абсолютной позицией и используют таблицу [DocumentAPI.AbsoluteFrame](#).

Пример для текстового документа

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    print(mediaObject:getFrame()) -- <userdata of type
    'CO::API::Document::InlineFrame'>
end
```

Пример для табличного документа

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
    print(mediaObject:getFrame()) -- <userdata of type
    'CO::API::Document::AbsoluteFrame'>
end
```

7.76.2 Метод MediaObject:toChart

Метод возвращает диаграмму [DocumentAPI.Chart](#), связанную со встроенным объектом. Если объект не является диаграммой, метод возвращает `nil`.

Диаграммы реализованы только в табличных документах.

Пример для табличного документа

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
```

```
local chart = mediaObject:toChart()
if chart ~= nil then
    print("Текущий объект является диаграммой")
else
    print("Текущий объект является фигурой")
end
end
```

7.76.3 Метод MediaObject:toImage

Метод возвращает изображение [DocumentAPI.Image](#), связанное со встроенным объектом. Если объект не является изображением, метод возвращает nil.

Пример для текстового документа

```
for mediaObject in document:getRange():getInlineObjects():enumerate() do
    local image = mediaObject:toImage()
    if image then
        print("Текущий объект является изображением")
    else
        print("Текущий объект является фигурой")
    end
end
```

Пример для табличного документа

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
    local image = mediaObject:toImage()
    if image ~= nil then
        print("Текущий объект является изображением")
    else
        print("Текущий объект является фигурой")
    end
end
```

7.77 Таблица DocumentAPI.MediaObjects

Таблица `DocumentAPI.MediaObjects` предназначен для доступа к коллекции графических объектов. Может быть получена вызовом методов [Table.getMediaObjects\(\)](#) или [Range.getInlineObjects\(\)](#) (см. Рисунок 39).

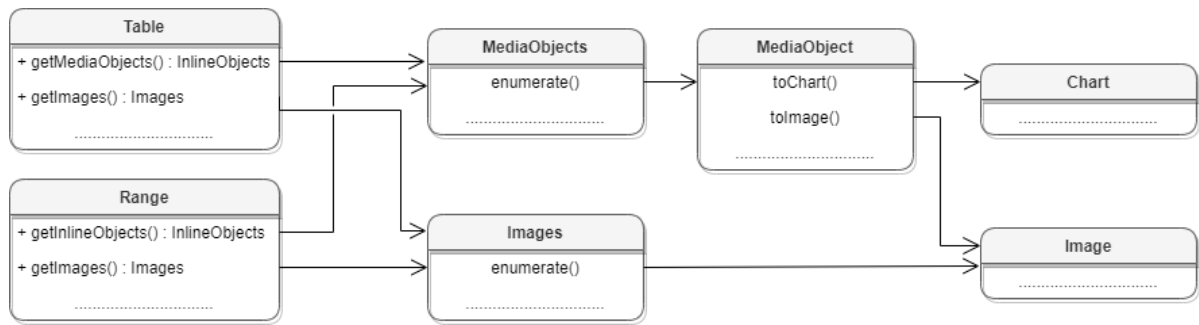


Рисунок 39 – Графические объекты

7.77.1 Метод MediaObjects:enumerate

Метод позволяет перечислить коллекцию встроенных объектов.

Примеры для текстового документа

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    print("Встроенный объект:", mediaObject)
end
```

```
local mediaObjects = EditorAPI.getSelection():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    print("Встроенный объект:", mediaObject)
end
```

Пример для табличного документа

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
    local image = mediaObject:toImage()
    if image ~= nil then
        print("Объект является изображением")
    else
        local chart = mediaObject:toChart()
        if chart ~= nil then
            print("Объект является диаграммой")
        else
            print("Объект является фигурой")
        end
    end
end
end
```

7.78 Таблица `DocumentAPI.NamedExpression`

Класс описывает структуру именованного диапазона.

Пример

```
local namedExpressions = sheet:getNameExpressions()  
for namedExpression in namedExpressions:enumerate() do  
    print(namedExpression:getName())  
    print(namedExpression:getExpression())  
    cellRange = namedExpression:getCellRange()  
    print(cellRange:getBeginRow(), cellRange:getLastRow())  
end
```

7.78.1 Метод `NamedExpression:getCellRange`

Возвращает диапазон ячеек [DocumentAPI.CellRange](#) именованного диапазона.

Пример см. в [DocumentAPI.NamedExpression](#).

7.78.2 Метод `NamedExpression:getExpression`

Возвращает текст диапазона (формулы). Пример см. в

[DocumentAPI.NamedExpression](#).

7.78.3 Метод `NamedExpression:getName`

Возвращает имя именованного диапазона. Пример см. в

[DocumentAPI.NamedExpression](#).

7.79 Таблица `DocumentAPI.NamedExpressions`

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

7.79.1 Метод `NamedExpressions:addExpression`

Добавляет новый диапазон в список именованных диапазонов, возвращает результат операции [NamedExpressionsValidationResult](#).

Пример

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

```
expressionValue)
if (validationResult == DocumentAPI.NamedExpressionsValidationResult_Success)
then
    print("Named expression was added")
end
```

7.79.2 Метод NamedExpressions:enumerate

Позволяет получить доступ ко всему списку именованных диапазонов.

Пример

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

7.79.3 Метод NamedExpressions:get

Возвращает именованный диапазон [NamedExpression](#) по имени name, если он существует.

Пример

```
local namedExpressions = document:getNamedExpressions()
local namedExpression = namedExpressions:get("Продажи")
if (namedExpression) then
    print(namedExpression:getName()) -- Продажи
else
    print("No named expression was found")
end
```

7.79.4 Метод NamedExpressions:removeExpression

Удаляет именованный диапазон по заданному имени, возвращает результат операции [NamedExpressionsValidationResult](#).

Пример

```
local namedExpression = namedExpressions:get(expressionName)
if (namedExpression) then
    local validationResult = namedExpressions:removeExpression(expressionName)
    if (validationResult ==
DocumentAPI.NamedExpressionsValidationResult_Success) then
        print("Named expression was removed")
    end
end
```



```
end  
end
```

7.80 Таблица DocumentAPI.NamedExpressionsValidationResult

Таблица `DocumentAPI.NamedExpressionsValidationResult` описывает результат операций [NamedExpressions:addExpression\(\)](#), [NamedExpressions:removeExpression\(\)](#). Описание полей таблицы представлено в таблице 44.

Таблица 44 – Описание полей таблицы `DocumentAPI.NamedExpressionsValidationResult`

Поле	Описание
<code>DocumentAPI.NamedExpressionsValidationResult_Success</code>	Операция выполнена успешно
<code>DocumentAPI.NamedExpressionsValidationResult_WrongName</code>	Неправильный формат имени
<code>DocumentAPI.NamedExpressionsValidationResult_isUsedInFormula</code>	Имя уже используется в формуле

7.81 Таблица DocumentAPI.NumberCellFormatting

Таблица содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода [Cell:setFormat\(\)](#). Описание полей таблицы `DocumentAPI.NumberCellFormatting` представлено в таблице 45.

Таблица 45 – Описание полей таблицы `DocumentAPI.NumberCellFormatting`

Поле	Описание
<code>DocumentAPI.NumberCellFormatting.decimalPlaces</code>	Количество десятичных позиций
<code>DocumentAPI.NumberCellFormatting.useThousandsSeparator</code>	Использовать разделитель для тысячных
<code>DocumentAPI.NumberCellFormatting.useRedForNegative</code>	Использовать красный цвет для отрицательных значений
<code>DocumentAPI.NumberCellFormatting.useBracketsForNegative</code>	Использовать скобки для отрицательных значений
<code>DocumentAPI.NumberCellFormatting.hideSign</code>	Скрывать знак «минус» для отрицательных значений

Пример

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A2")

local numberCellFormatting = DocumentAPI.NumberCellFormatting()
numberCellFormatting.decimalPlaces = 2
numberCellFormatting.useThousandsSeparator = true
numberCellFormatting.useRedForNegative = true
numberCellFormatting.useBracketsForNegative = true
numberCellFormatting.hideSign = false

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

7.82 Таблица DocumentAPI.PageFieldOrder

Таблица `DocumentAPI.PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем таблицы [DocumentAPI.PivotTableLayoutSettings](#). Описание полей таблицы представлено в таблице 46.

Таблица 46 – Описание полей таблицы `DocumentAPI.PageFieldOrder`

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

7.83 Таблица DocumentAPI.PageOrientation

Типы ориентации страницы представлены в таблице 47. Данная константа может быть использована для получения / установки ориентации страниц для секции или документа.

Таблица 47 – Типы ориентации страницы

Наименование константы	Описание
<code>DocumentAPI.PageOrientation_Landscape</code>	Альбомная ориентация страницы
<code>DocumentAPI.PageOrientation_Portrait</code>	Портретная ориентация страницы

Примеры

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

7.84 Таблица DocumentAPI.PageProperties

Таблица `DocumentAPI.PageProperties` предоставляет такие свойства страницы как высота, ширина, размеры полей. Описание полей приведено в таблице 48. Используется в [Document.setPageProperties\(\)](#), [Section.getPageProperties\(\)](#), [Section.setPageProperties\(\)](#). Применяется только в текстовом документе.

Таблица 48 – Описание полей таблицы `DocumentAPI.PageProperties`

Поле	Описание
<code>DocumentAPI.PageProperties.height</code>	Высота страницы
<code>DocumentAPI.PageProperties.width</code>	Ширина страницы
<code>DocumentAPI.PageProperties.margins</code>	Поля страницы, тип - Insets

Примеры

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

```
local pageProperties = DocumentAPI.PageProperties()  
pageProperties.width = 100  
pageProperties.height = 200  
document:setPageProperties(pageProperties)
```

```
local pageProperties = DocumentAPI.PageProperties(100, 200)  
document:setPageProperties(pageProperties)
```

7.84.1 Метод PageProperties: __eq

Метод позволяет использовать оператор сравнения `__eq` для определения эквивалентности содержимого двух структур [DocumentAPI.PageProperties](#).

Пример

```
local pageProperties1 = DocumentAPI.PageProperties(100, 200)
document:setPageProperties(pageProperties1)

local pageProperties2 = DocumentAPI.PageProperties(100, 200)
document:setPageProperties(pageProperties2)

print(pageProperties1:__eq(pageProperties2)) -- true
```

7.85 Таблица DocumentAPI.Paragraph

Таблица DocumentAPI.Paragraph предоставляет доступ к свойствам абзаца (см. Рисунок 40).

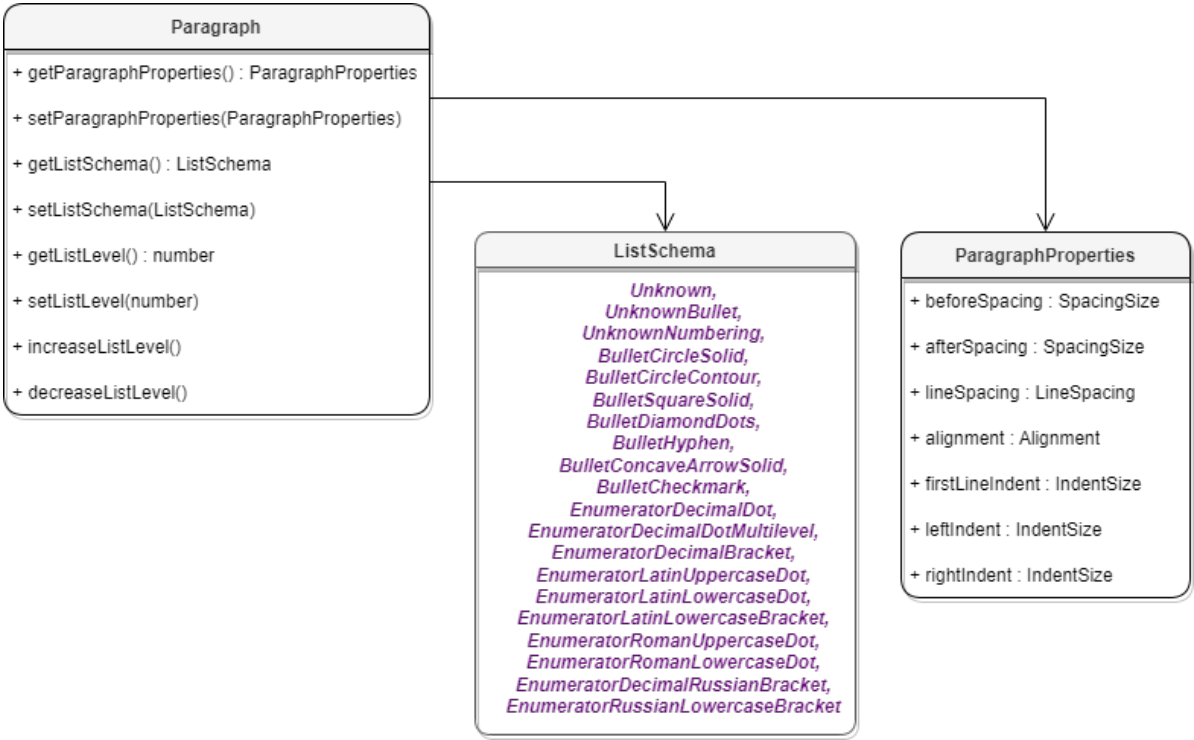


Рисунок 40 – Объектная модель таблиц для работы со свойствами параграфа

7.85.1 Метод Paragraph:decreaseListLevel

Метод позволяет уменьшить на единицу глубину вложенности элемента списка. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе.

Пример

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:decreaseListLevel()
```

7.85.2 Метод Paragraph:getListLevel

Метод позволяет получить глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример

```
local paragraph = document:getBlocks():getParagraph(0)
local level = paragraph:getListLevel()
```

7.85.3 Метод Paragraph:getListSchema

Метод возвращает схему форматирования абзаца [DocumentAPI.ListSchema](#) либо значение nil, если схема нумерации не установлена для абзаца. Данный метод используется только в текстовом документе.

Пример

```
local paragraph = document:getBlocks():getParagraph(0)
local schema = paragraph:getListSchema()
```

7.85.4 Метод Paragraph:getParagraphProperties

Метод предоставляет доступ к таблице свойств форматирования абзаца [DocumentAPI.ParagraphProperties](#), таким как выравнивание текста, межстрочные интервалы, отступы и т. д.

Пример для текстового документа

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
print(para_props.afterSpacing)
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()
```

```
for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    print(para_props.afterSpacing)
end
```

7.85.5 Метод Paragraph:increaseListLevel

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

Пример

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:increaseListLevel()
```

7.85.6 Метод Paragraph:setListLevel

Метод позволяет установить глубину вложенности элемента списка.

Значение может быть равным nil, если схема нумерации не установлена для абзаца. В этом случае будет установлено минимальное значение. Данный метод используется только в текстовом документе.

Пример

```
local paragraph = document:getBlocks():getParagraph(0)
local level = paragraph:setListLevel(1)
```

7.85.7 Метод Paragraph:setListSchema

Метод позволяет установить тип маркированного или нумерованного списка [DocumentAPI.ListSchema](#). Данный метод используется только в текстовом документе.

Пример

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

7.85.8 Метод Paragraph:setParagraphProperties

Метод предназначен для обновления таблицы свойств форматирования абзаца [DocumentAPI.ParagraphProperties](#).

Пример для текстового документа

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

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    para_props.alignment = DocumentAPI.Alignment_Right
    para:setParagraphProperties(para_props)
end
```

7.86 Таблица DocumentAPI.ParagraphProperties

Таблица `DocumentAPI.ParagraphProperties` предназначена для управления свойствами форматирования (см. Рисунок 41). Таблица `DocumentAPI.ParagraphProperties` используется в методах [Paragraph:getParagraphProperties\(\)](#), [Paragraph:setParagraphProperties\(\)](#), [Cell:getParagraphProperties\(\)](#), [Cell:setParagraphProperties\(\)](#), [CellRange:getParagraphProperties\(\)](#) и [CellRange:setParagraphProperties\(\)](#).

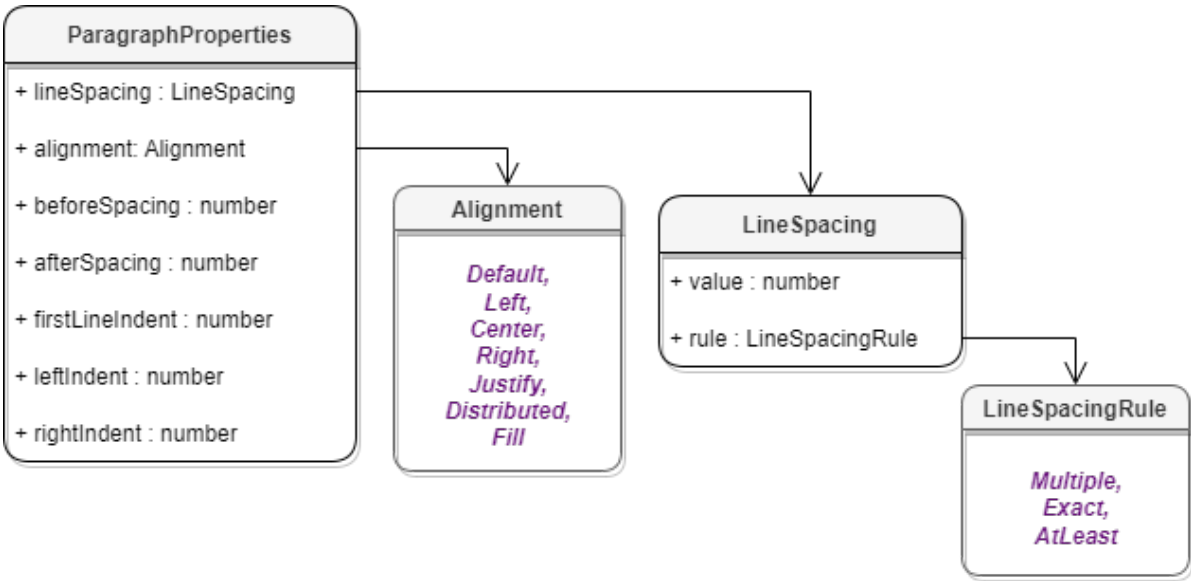
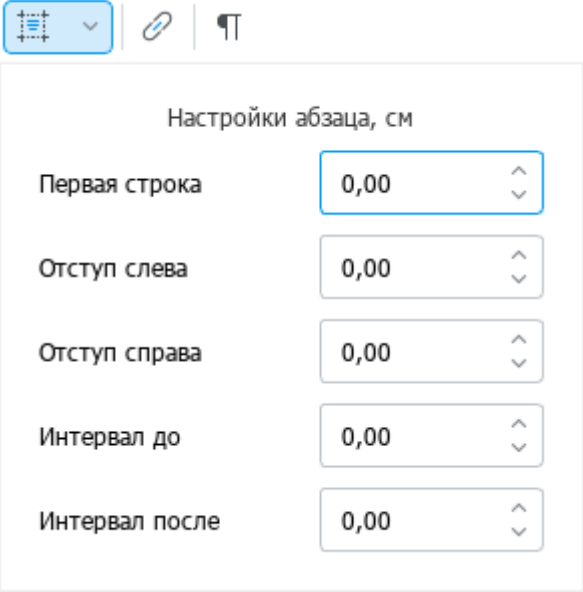


Рисунок 41 – Объектная модель таблиц для работы со свойствами параграфа

Описание полей таблицы [DocumentAPI.ParagraphProperties](#) представлено в таблице 49.

Таблица 49 – Описание полей таблицы DocumentAPI.ParagraphProperties

Поле	Описание
ParagraphProperties.beforeSpacing	Установка величины расстояния до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , (см. рисунок выше), в поле Интервал до (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).
ParagraphProperties.afterSpacing	Установка величины расстояния после абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , в поле Интервал после (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).

Поле	Описание
	 <p>Настройки абзаца, см</p> <p>Первая строка 0,00</p> <p>Отступ слева 0,00</p> <p>Отступ справа 0,00</p> <p>Интервал до 0,00</p> <p>Интервал после 0,00</p>
ParagraphProperties.lineSpacing	Расстояние между строк одного абзаца (межстрочный интервал), LineSpacing .
ParagraphProperties.alignment	Выравнивание текстового фрагмента по горизонтали. Список допустимых значений находится в разделе Alignment .
ParagraphProperties.firstLineIndent	Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , (см. рисунок выше), в поле Первая строка (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).
ParagraphProperties.leftIndent	Расстояние от левого поля документа до абзаца (отступ слева). При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , (см. рисунок выше), в поле Отступ слева (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).
ParagraphProperties.rightIndent	Расстояние от правого поля документа до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца , (см. рисунок выше), в поле Отступ справа (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).

Пример для текстового документа

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
--
para_props.afterSpacing = 28.3 -- значение соответствует 1 см
para_props.beforeSpacing = 28.3 -- значение соответствует 1 см
para_props.alignment = DocumentAPI.Alignment_Center
para_props.firstLineIndent = 28.3 -- значение соответствует 1 см
para_props.leftIndent = 28.3 -- значение соответствует 1см
para_props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
para_props.rightIndent = 28.3 -- значение соответствует 1см
--
para:setParagraphProperties(para_props)
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    para_props.afterSpacing = 28.3 -- значение соответствует 1 см
    para_props.beforeSpacing = 28.3 -- значение соответствует 1 см
    para_props.alignment = DocumentAPI.Alignment_Center
    para_props.firstLineIndent = 28.3 -- значение соответствует 1 см
    para_props.leftIndent = 28.3 -- значение соответствует 1см
    para_props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
    para_props.rightIndent = 28.3 -- значение соответствует 1см
    para:setParagraphProperties(para_props)
end
```

7.87 Таблица DocumentAPI.Paragraphs

Таблица `DocumentAPI.Paragraphs` предоставляет доступ к коллекции абзацев типа [DocumentAPI.Paragraph](#) (см. Рисунок 42). Коллекция абзацев может быть получена из таблицы [DocumentAPI.Range](#) посредством использования вызова [Range:getParagraphs\(\)](#).

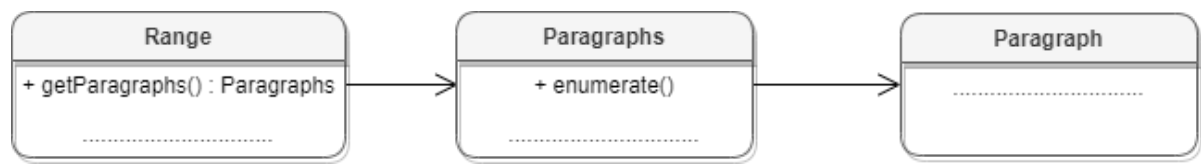


Рисунок 42 – Объектная модель для работы со списком абзацев

Пример для текстового документа

```
local paragraphs = document:getRange():getParagraphs()
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local paragraphs = cell:getRange():getParagraphs()
```

7.87.1 Метод Paragraphs:decreaseListLevel

Метод уменьшает уровень списка на единицу. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе.

Пример

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:decreaseListLevel()
```

7.87.2 Метод Paragraphs:enumerate

Метод позволяет перечислить коллекцию абзацев.

Пример для текстового документа

```
local paragraphs = document:getRange():getParagraphs()
for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    print(para_props.alignment)
end
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()
```

```
for para in paragraphs:enumerate() do
    local para_props = para:getParagraphProperties()
    print(para_props.alignment)
end
```

7.87.3 Метод Paragraphs:increaseListLevel

Метод увеличивает уровень списка на единицу. В случае, если максимальный уровень уже установлен, увеличения не происходит. Данный метод используется только в текстовом документе.

Пример

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:increaseListLevel()
```

7.87.4 Метод Paragraphs:setListLevel

Метод устанавливает глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:setListLevel(1)
```

7.87.5 Метод Paragraphs:setListSchema

Метод устанавливает тип маркированного или нумерованного списка [DocumentAPI.ListSchema](#). Данный метод используется только в текстовом документе.

Пример

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

7.88 Таблица DocumentAPI.PercentageCellFormatting

Содержит параметр для процентного формата ячеек таблицы, используется в качестве аргумента метода [Cell:setFormat\(\)](#). Описание полей таблицы DocumentAPI.PercentageCellFormatting представлено в таблице 50.

Таблица 50 – Описание полей таблицы DocumentAPI.PercentageCellFormatting

Поле	Описание
DocumentAPI.PercentageCellFormatting.decimalPlaces	Количество десятичных позиций

Пример

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

local percentageCellFormatting = DocumentAPI.PercentageCellFormatting()
percentageCellFormatting.decimalPlaces = 2

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

7.89 Таблица DocumentAPI.PivotTable

Таблица для представления сводной таблицы. Может быть получена из ячейки [Cell.getPivotTable\(\)](#), или при создании новой сводной таблицы [PivotTablesManager.create\(\)](#).

7.89.1 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.

7.89.2 Метод PivotTable:changeSourceRange

Метод позволяет задать новый диапазон исходных данных сводной таблицы без обновления самой таблицы. Параметр sourceRange – строка, представляющая новый диапазон таблицы.

Пример

```
pivotTable:changeSourceRange("I3:K5")
local cellRange = pivotTable:getSourceRange()
print(cellRange:getBeginRow(), cellRange:getLastRow())
```

7.89.3 Метод `PivotTable:createPivotTableEditor`

Метод возвращает объект [DocumentAPI.PivotTableEditor](#), который служит для обновления свойств и редактирования сводной таблицы.

Пример

```
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local pivotTableEditor = pivotTable:createPivotTableEditor()
```

7.89.4 Метод `PivotTable:getColumnFields`

Метод возвращает список полей [DocumentAPI.PivotTableCategoryField](#) из области колонок.

Пример

```
local columnFields = pivotTable:getColumnFields()
for fieldIdx = 0, columnFields:size() - 1 do
    print(columnFields[fieldIdx].fieldProperties.fieldName)
end
```

7.89.5 Метод `PivotTable:getConditionalLabelFilter`

Метод возвращает примененный условный фильтр по подписи.

Вызов

```
PivotTableConditionalLabelFilter getConditionalLabelFilter(fieldName)
```

Параметры

– `fieldName`: название поля в сводной таблице, тип `string`.

Возвращает

– условный фильтр по подписи, тип [PivotTableConditionalLabelFilter](#).

– `nil`, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

Пример

```
local labelOperation = DocumentAPI.PivotTableConditionalLabelFilterOperation()
labelOperation.operationType =
DocumentAPI.PivotTableConditionalLabelFilterOperationType_Contains
labelOperation.operand = "to"

local labelFilter = pivotTable:getConditionalLabelFilter("Product Name")
```

```
labelFilter:setOperation(labelOperation)

tableEditor:setFilter(labelFilter):apply()
```

7.89.6 Метод PivotTable:getConditionalValueFilter

Метод возвращает примененный условный фильтр по значению.

Вызов

```
PivotTableConditionalValueFilter getConditionalValueFilter(fieldName)
```

Параметры

– `fieldName`: название поля в сводной таблице, тип `string`.

Возвращает

– условный фильтр по значению, тип [PivotTableConditionalValueFilter](#).

– `nil`, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

Пример

```
local valueOperation =
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.PivotTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

local valueFilter = pivotTable:getConditionalValueFilter("Product Name")
valueFilter:setOperation(valueOperation)

tableEditor:setFilter(valueFilter):apply()
```

7.89.7 Метод PivotTable:getFieldCategories

Метод возвращает список категорий [DocumentAPI.PivotTableFieldCategories](#), содержащих заданное поле `fieldName`.

Пример

```
local fieldCategories = pivotTable:getFieldCategories("Age")
```

7.89.8 Метод PivotTable:getFieldItems

Метод возвращает все элементы [DocumentAPI.PivotTableItems](#) сводной таблицы по заданному имени поля `fieldName`.

Пример

```
local pivotTableItems = pivotTable:getFieldItems("Age")
print(pivotTableItems)
```

7.89.9 Метод PivotTable:getFieldItemsByName

Метод возвращает все элементы [DocumentAPI.PivotTableItems](#) из заданного поля fieldName по имени itemName.

Пример

```
local pivotTableItemsByName = pivotTable:getFieldItemsByName("Ultimate Question of Life", "42")
print(pivotTableItemsByName)
```

7.89.10 Метод PivotTable:getFieldsList

Метод возвращает список [DocumentAPI.PivotTableField](#) всех полей сводной таблицы.

Пример

```
local fieldsList = pivotTable:getFieldsList()
print(fieldsList:size())
for fieldIdx = 0, fieldsList:size() - 1 do
    print(fieldsList[fieldIdx].fieldProperties.fieldName)
end
```

7.89.11 Метод PivotTable:getFilter

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

Пример

```
local filter = pivotTable:getFilter("Age")
print(filter:getFieldName())
```

7.89.12 Метод PivotTable:getFilters

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

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
```



```
-- use filter  
end
```

7.89.13 Метод `PivotTable:getPageFields`

Метод возвращает список полей [DocumentAPI.PivotTablePageField](#) из области фильтров.

Пример

```
local pageFields = pivotTable:getPageFields()  
print(pageFields:size())
```

7.89.14 Метод `PivotTable:getPivotRange`

Метод возвращает диапазон ячеек [DocumentAPI.CellRange](#), в котором размещена сводная таблица.

Пример

```
local tbl = document:getBlocks():getTable(0)  
local cell = tbl:getCell("L8")  
local pivotTable = cell:getPivotTable()  
local cellRange = pivotTable:getPivotRange()  
print(cellRange:getBeginRow(), cellRange:getLastRow()) -- 7 10
```

7.89.15 Метод `PivotTable:getPivotTableCaptions`

Метод возвращает информацию [DocumentAPI.PivotTableCaptions](#) о всех заголовках сводной таблицы.

Пример

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

7.89.16 Метод `PivotTable:getPivotTableLayoutSettings`

Метод возвращает настройки отображения [DocumentAPI.PivotTableLayoutSettings](#) сводной таблицы.

Пример

```
local settings = pivotTable:getPivotTableLayoutSettings()
print(settings.reportLayout)
print(settings.valueFieldsOrientation)
print(settings.pageFieldOrder)
print(settings.indentForCompactLayout)
print(settings.pageFieldWrapCount)
```

7.89.17 Метод PivotTable:getRowFields

Метод возвращает список полей [DocumentAPI.PivotTableCategoryField](#) из области строк.

Пример

```
local rowFields = pivotTable:getRowFields()
for fieldIdx = 0, rowFields:size() - 1 do
    print(rowFields[fieldIdx].fieldProperties.fieldName)
end
```

7.89.18 Метод PivotTable:getSourceRange

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

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local cellRange = pivotTable:getSourceRange()
print(cellRange:getBeginRow(), cellRange:getLastRow()) -- 2 6
```

7.89.19 Метод PivotTable:getSourceRangeAddress

Метод возвращает текстовое представление диапазона исходных данных сводной таблицы.

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
if (pivotTable) then
    print(pivotTable:getSourceRangeAddress()) -- 'Sheet1'!I3:K7
end
```

7.89.20 Метод PivotTable:getValueFields

Метод возвращает список полей [DocumentAPI.PivotTableValueField](#) из области значений.

Пример

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

7.89.21 Метод PivotTable:isColumnGrandTotalEnabled

Метод возвращает true, если разрешено показывать общие итоги для столбцов.

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A3")
local pivotTable = cell:getPivotTable()
print(pivotTable:isColumnGrandTotalEnabled())
```

7.89.22 Метод PivotTable:isRowGrandTotalEnabled

Метод возвращает true, если разрешено показывать общие итоги для строк.

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A3")
local pivotTable = cell:getPivotTable()
print(pivotTable:isRowGrandTotalEnabled())
```

7.89.23 Метод PivotTable:remove

Метод удаляет сводную таблицу.

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
if (pivotTable) then
```

```
pivotTable:remove()  
end
```

7.89.24 Метод PivotTable:update

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

Пример

```
local updateResult = pivotTable:update()  
if (updateResult ~= DocumentAPI.PivotTableUpdateResult_Success) then  
    print(updateResult)  
end
```

7.90 Таблица DocumentAPI.PivotTableCaptions

Таблица `DocumentAPI.PivotTableCaptions` хранит все пользовательские заголовки сводной таблицы. Описание полей таблицы представлено в таблице 51.

Таблица 51 – Описание полей таблицы `DocumentAPI.PivotTableCaptions`

Поле	Описание
<code>PivotTableCaptions.errorCaption</code>	Алиас для значений, которые возвращают ошибку.
<code>PivotTableCaptions.emptyCaption</code>	Алиас для значений, которые возвращают пустое значение.
<code>PivotTableCaptions.grandTotalCaption</code>	Алиас общих итогов.
<code>PivotTableCaptions.valuesHeaderCaption</code>	Алиас поля из области значений; это поле отображается в отчете в случае, если в сводной таблице наличие более двух полей из области значений, и макет имеют тип 'outline' или 'tabular'.
<code>PivotTableCaptions.rowHeaderCaption</code>	Алиас заголовка строк (виден только при включенном компактном макете, это алиас по умолчанию).
<code>PivotTableCaptions.columnHeaderCaption</code>	Алиас заголовка колонок (виден только при включенном компактном макете, это алиас по умолчанию).

7.91 Таблица DocumentAPI.PivotTableCategoryField

`DocumentAPI.PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. таблицу 52). Таблица может быть

получена посредством вызовов [PivotTable:getRowFields\(\)](#),
[PivotTable:getColumnFields\(\)](#).

Таблица 52 – Описание полей таблицы DocumentAPI.PivotTableCategoryField

Поле	Описание
PivotTableCategoryField. fieldProperties	Свойства поля PivotTableFieldProperties
PivotTableCategoryField. subtotalFunctions	Список функций PivotTableFunction для вычисления подытога

7.92 Таблица DocumentAPI.PivotTableConditionalLabelFilter

Таблица PivotTableConditionalLabelFilter представляет собой условный фильтр по подписи. Фильтрация производится по строковым значениям, которые содержит поле сводной таблицы. Используется в методах [PivotTable:getConditionalLabelFilter\(\)](#) и [PivotTableEditor:setFilter\(\)](#).

7.92.1 Метод PivotTableConditionalLabelFilter:getFieldName

Метод возвращает имя поля, с которым ассоциирован фильтр.

Вызов

```
string getFieldname()
```

Возвращает

– имя поля, тип string.

7.92.2 Метод PivotTableConditionalLabelFilter:getOperation

Метод возвращает текущие настройки фильтра.

Вызов

```
PivotTableConditionalLabelFilterOperation getOperation()
```

Возвращает

– настройки фильтра, тип [PivotTableConditionalLabelFilterOperation](#).

7.92.3 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.

7.92.4 Метод PivotTableConditionalLabelFilter:reset

Метод сбрасывает настройки текущего фильтра.

Вызов

```
reset()
```

7.92.5 Метод PivotTableConditionalLabelFilter:setOperation

Метод устанавливает настройки фильтра.

Вызов

```
setOperation(operation)
```

Параметры

– operation:	настройки	фильтра,	тип
	PivotTableConditionalLabelFilterOperation.		

Пример

```
local labelOperation = DocumentAPI.PivotTableConditionalLabelFilterOperation()  
labelOperation.operationType =  
DocumentAPI.PivotTableConditionalLabelFilterOperationType_Contains  
labelOperation.operand = "to"  
  
local labelFilter = pivotTable:getConditionalLabelFilter("Product Name")  
labelFilter:setOperation(labelOperation)  
  
tableEditor:setFilter(labelFilter):apply()
```

7.93 Таблица DocumentAPI.PivotTableConditionalLabelFilterOperation

Таблица PivotTableConditionalLabelFilterOperation представляет собой настройки условного фильтра по подписи. Используется в методах [PivotTableConditionalLabelFilter:getOperation\(\)](#) и [PivotTableConditionalLabelFilter:setOperation\(\)](#).

Таблица 53 – Описание полей таблицы PivotTableConditionalLabelFilterOperation

Поле	Описание
PivotTableConditionalLabelFilterOperation.operationType	Тип операции сравнения, тип PivotTableConditionalLabelFilterOperationType
PivotTableConditionalLabelFilterOperation.operand	Значение для сравнения, тип string

Пример

```
local labelOperation = DocumentAPI.PivotTableConditionalLabelFilterOperation()  
labelOperation.operationType =  
DocumentAPI.PivotTableConditionalLabelFilterOperationType_Contains  
labelOperation.operand = "to"  
  
local labelFilter = pivotTable:getConditionalLabelFilter("Product Name")  
labelFilter:setOperation(labelOperation)  
  
tableEditor:setFilter(labelFilter):apply()
```

7.94 Таблица DocumentAPI.PivotTableConditionalLabelFilterOperationType

Таблица PivotTableConditionalLabelFilterOperationType определяет тип операции сравнения, применяемый к фильтру по подписи. Используется в поле [PivotTableConditionalLabelFilterOperation.operationType](#).

Таблица 54 – Описание типов операций сравнения

Наименование константы	Описание
PivotTableConditionalLabelFilterOperationType_Equal	Равные
PivotTableConditionalLabelFilterOperationType_NotEqual	Не равные
PivotTableConditionalLabelFilterOperationType_BeginsWith	Начинаются с
PivotTableConditionalLabelFilterOperationType_NotBeginsWith	Не начинается с
PivotTableConditionalLabelFilterOperationType_EndsWith	Заканчиваются на
PivotTableConditionalLabelFilterOperationType_NotEndsWith	Не заканчиваются на

Наименование константы	Описание
PivotTableConditionalLabelFilterOperationType_Contains	Содержат
PivotTableConditionalLabelFilterOperationType_NotContains	Не содержат

7.95 Таблица DocumentAPI.PivotTableConditionalValueFilter

Таблица `PivotTableConditionalValueFilter` представляет собой условный фильтр по значению. Фильтрация производится по значениям, которые соответствуют полю в строке или столбце сводной таблицы. Используется в методах [PivotTable:getConditionalValueFilter\(\)](#) и [PivotTableEditor:setFilter\(\)](#).

7.95.1 Метод PivotTableConditionalValueFilter:getDefaultOperation

Метод возвращает настройки условного фильтра соответствующие заданной операции сравнения.

Вызов

```
static PivotTableConditionalValueFilterOperation  
getDefaultOperation(operationType)
```

Параметры

– operationType: тип операции сравнения, тип [PivotTableConditionalValueFilterOperationType](#).

Возвращает

– настройки условного фильтра по значению, тип [PivotTableConditionalValueFilterOperation](#).

Пример

```
local valueOperation =  
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.Piv  
otTableConditionalValueFilterOperationType_Between)  
valueOperation.operand1 = "20"  
valueOperation.operand2 = "50"  
  
local valueFilter = pivotTable:getConditionalValueFilter("Product Name")  
valueFilter:setOperation(valueOperation)
```



```
tableEditor:setFilter(valueFilter):apply()
```

7.95.2 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](#).

7.95.3 Метод PivotTableConditionalValueFilter:getFieldName

Метод возвращает имя поля, с которым ассоциирован фильтр.

Вызов

```
string getFieldName()
```

Возвращает

– имя поля, тип string.

7.95.4 Метод PivotTableConditionalValueFilter:getOperation

Метод возвращает текущие настройки фильтра.

Вызов

```
PivotTableConditionalValueFilterOperation getOperation()
```

Возвращает

– настройки фильтра, тип [PivotTableConditionalValueFilterOperation](#).

7.95.5 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.

7.95.6 Метод PivotTableConditionalValueFilter:reset

Метод сбрасывает настройки текущего фильтра.

Вызов

```
reset()
```

7.95.7 Метод PivotTableConditionalValueFilter:setOperation

Метод устанавливает настройки фильтра.

Вызов

```
setOperation(operation)
```

Параметры

– operation: настройки фильтра, тип
[PivotTableConditionalValueFilterOperation](#).

Пример

```
local valueOperation =
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.PivotTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

local valueFilter = pivotTable:getConditionalValueFilter("Product Name")
valueFilter:setOperation(valueOperation)

tableEditor:setFilter(valueFilter):apply()
```

7.96 Таблица DocumentAPI.PivotTableConditionalValueFilterOperandType

Таблица PivotTableConditionalValueFilterOperandType определяет тип данных, с которыми могут работать различные операции сравнения. Используется в методе [PivotTableConditionalValueFilter:getExpectedOperandType\(\)](#).

Таблица 55 – Описание типов данных

Наименование константы	Описание
PivotTableConditionalValueFilterOperandType_PositiveInt	Положительное целое число
PivotTableConditionalValueFilterOperandType_Percent	Число с плавающей запятой в диапазоне от 0 до 100

Наименование константы	Описание
PivotTableConditionalValueFilterOperandType_Double	Дробное число
PivotTableConditionalValueFilterOperandType_NonNegativeDouble	Положительное дробное число

7.97 Таблица DocumentAPI.PivotTableConditionalValueFilterOperation

Таблица PivotTableConditionalValueFilterOperation представляет собой настройки условного фильтра по значению. Используется в методах [PivotTableConditionalValueFilter:getDefaultOperation\(\)](#), [PivotTableConditionalValueFilter:getOperation\(\)](#) и [PivotTableConditionalValueFilter:setOperation\(\)](#).

Таблица 56 – Описание полей таблицы PivotTableConditionalValueFilterOperation

Поле	Описание
PivotTableConditionalValueFilterOperation.operationType	Тип операции сравнения, тип PivotTableConditionalValueFilterOperationType
PivotTableConditionalValueFilterOperation.valueFieldIndex	Индекс поля, к которому применяется фильтр
PivotTableConditionalValueFilterOperation.operand1	Первое значение для сравнения, тип string
PivotTableConditionalValueFilterOperation.operand2	Второе значение для операции сравнения Between, тип string

Пример

```
local valueOperation =
DocumentAPI.PivotTableConditionalValueFilter.getDefaultOperation(DocumentAPI.PivotTableConditionalValueFilterOperationType_Between)
valueOperation.operand1 = "20"
valueOperation.operand2 = "50"

local valueFilter = pivotTable:getConditionalValueFilter("Product Name")
valueFilter:setOperation(valueOperation)

tableEditor:setFilter(valueFilter):apply()
```

7.98 Таблица DocumentAPI.PivotTableConditionalValueFilterOperationType

Таблица PivotTableConditionalValueFilterOperationType определяет тип операции сравнения, применяемый к фильтру по значению. Используется в поле [PivotTableConditionalValueFilterOperation.operationType](#).

Таблица 57 – Описание типов операций сравнения

Наименование константы	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

7.99 Таблица DocumentAPI.PivotTableEditor

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

7.99.1 Метод PivotTableEditor.addField

Метод добавляет новое поле в сводную таблицу, используя параметры:

- fieldName - имя поля;
- toCategory - категория поля (тип - [DocumentAPI.PivotTableFieldCategory](#));

— index - позиция в категории.

Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
pivotTableEditor = pivotTableEditor.addField("CC",  
DocumentAPI.PivotTableFieldCategory_Values)  
pivotTableEditor.apply()
```

7.99.2 Метод PivotTableEditor:apply

Метод обновляет сводную таблицу с заданными свойствами и возвращает результат [DocumentAPI.PivotTableUpdateResult](#).

Пример

```
local pivotTableEditor = pivotTable:createPivotTableEditor()  
if DocumentAPI.PivotTableUpdateResult_Success == pivotTableEditor.apply() then  
    print("Successfully applied");  
end
```

7.99.3 Метод PivotTableEditor:disableField

Метод удаляет поле из всех областей. Параметр fieldName - имя поля (тип - строка). Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
local pivotTableEditor = pivotTable:createPivotTableEditor()  
pivotTableEditor:disableField("Age")  
pivotTableEditor.apply()
```

7.99.4 Метод PivotTableEditor:enableField

Метод добавляет поле в область, зависящую от типа поля. Параметр fieldName - имя поля. Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
local pivotTableEditor = pivotTable:createPivotTableEditor()  
pivotTableEditor:enableField("Age")  
pivotTableEditor.apply()
```

7.99.5 Метод PivotTableEditor:moveField

Метод перемещает поле между категориями.

Параметры:

- fieldName - имя поля;
- toCategory - область, в которую перемещается поле (тип - [DocumentAPI.PivotTableFieldCategory](#));
- index - позиция в новой категории.

Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
pivotTableEditor = pivotTableEditor:moveField("BB",  
DocumentAPI.PivotTableFieldCategory_Values, 0)  
pivotTableEditor:apply()
```

7.99.6 Метод PivotTableEditor:removeField

Метод удаляет поле из категории.

Параметры:

- fieldName - имя поля,
- fromCategory - область, из которой удаляется поле (тип - [DocumentAPI.PivotTableFieldCategory](#)).

Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
pivotTableEditor = pivotTableEditor:removeField("Age",  
DocumentAPI.PivotTableFieldCategory_Values)  
pivotTableEditor:apply()
```

7.99.7 Метод PivotTableEditor:reorderField

Метод изменяет позицию поля в пределах категории.

Параметры:

- fieldName - имя поля;
- category - область (тип - [DocumentAPI.PivotTableFieldCategory](#));
- toIndex - новая позиция поля.

Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
pivotTableEditor = pivotTableEditor:reorderField("Age",  
DocumentAPI.PivotTableFieldCategory_Values, 0)  
pivotTableEditor:apply()
```

7.99.8 Метод PivotTableEditor:resetAllFilters

Метод сбрасывает все фильтры, примененные к сводной таблице, и обновляет её.

Вызов

```
PivotTableEditor resetAllFilters()
```

Возвращает

– текущий редактор сводных таблиц, тип [PivotTableEditor](#).

7.99.9 Метод PivotTableEditor:setCaptions

Метод задает заголовки сводной таблицы [DocumentAPI.PivotTableCaptions](#), возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
local pivotTableCaptions = pivotTable:getPivotTableCaptions()  
pivotTableCaptions.grandTotalCaption = "Общий итог за год"  
  
local pivotTableEditor = pivotTable:createPivotTableEditor()  
pivotTableEditor = pivotTableEditor:setCaptions(pivotTableCaptions)  
pivotTableEditor:apply()
```

7.99.10 Метод PivotTableEditor:setFilter

Метод задает фильтр [DocumentAPI.PivotTableFilter](#), [DocumentAPI.PivotTableConditionalLabelFilter](#) или [DocumentAPI.PivotTableConditionalValueFilter](#) сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`. Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Перегрузки

```
PivotTableEditor setFilter(PivotTableFilter filter)
```

```
PivotTableEditor setFilter(PivotTableConditionalLabelFilter filter)
```

```
PivotTableEditor setFilter(PivotTableConditionalValueFilter filter)
```

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    for filterIdx = 0, filter:getCount() - 1 do
        filter:setHidden(filterIdx, false)
        pivotTableEditor:setFilter(filter)
    end
end
pivotTableEditor:apply()
```

7.99.11 Метод PivotTableEditor:setFilters

Метод задает фильтры [DocumentAPI.PivotTableFilters](#) сводной таблицы. Если какой-то из фильтров не может быть применен, он пропускается. Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    for filterIdx = 0, filter:getCount() - 1 do
        filter:setHidden(filterIdx, false)
    end
end
pivotTableEditor:setFilters(filters)
pivotTableEditor:apply()
```

7.99.12 Метод PivotTableEditor:setGrandTotalSettings

Метод задает настройки отображения общего итога.

Параметры:

- isRowGrandTotalEnabled – показывать общие итоги для строк;
- isColGrandTotalEnabled – показывать общие итоги для столбцов.

Пример

```
local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor:setGrandTotalSettings(true, true)
```

7.99.13 Метод PivotTableEditor:setLayoutSettings

Метод	устанавливает	настройки	отображения
DocumentAPI.PivotTableLayoutSettings	сводной	таблицы,	возвращает объект

[DocumentAPI.PivotTableEditor.](#)

Пример

```
local layoutSettings = pivotTable:getPivotTableLayoutSettings()  
layoutSettings.reportLayout = DocumentAPI.PivotTableReportLayout_Tabular  
  
local pivotTableEditor = pivotTable:createPivotTableEditor()  
pivotTableEditor = pivotTableEditor:setLayoutSettings(layoutSettings)  
pivotTableEditor:apply()
```

7.99.14 Метод PivotTableEditor:setSummarizeFunction

Метод задает суммирующую функцию для поля из области значений.

Параметры

- valueFieldName - имя поля (тип - строка);
- summarizeFunction - суммирующая функция, тип - [DocumentAPI.PivotTableFunction.](#)

Метод возвращает объект [DocumentAPI.PivotTableEditor.](#)

Пример

```
pivotTableEditor = pivotTableEditor:setSummarizeFunction("Age",  
DocumentAPI.PivotTableFunction_Sum)  
pivotTableEditor:apply()
```

7.100 Таблица DocumentAPI.PivotTableField

Таблица DocumentAPI.PivotTableField содержит свойства полей сводной таблицы (см. таблицу 58). Таблица может быть получена посредством вызова [PivotTable:getFieldsList\(\)](#).

Таблица 58 – Описание полей таблицы DocumentAPI.PivotTableField

Поле	Описание
PivotTableField.fieldProperties	Свойства полей сводной таблицы PivotTableFieldProperties
PivotTableField.fieldCategories	Категории полей сводной таблицы PivotTableFieldCategories
PivotTableField.customFormula	Вычисляемая формула (строка)

7.101 Таблица `DocumentAPI.PivotTableFieldCategories`

Класс обеспечивает доступ к списку категорий поля сводной таблицы. Может быть получена посредством использования метода [`PivotTable.getFieldCategories\(\)`](#).

7.101.1 Метод `PivotTableFieldCategories:enumerate`

Метод для перечисления категорий поля [`DocumentAPI.PivotTableFieldCategory`](#).

Пример

```
local fieldCategories = pivotTable:getFieldCategories("Age")
for fieldCategory in fieldCategories:enumerate() do
    print(fieldCategory)
end
```

7.102 Таблица `DocumentAPI.PivotTableFieldCategory`

Таблица `DocumentAPI.PivotTableFieldCategory` описывает флаги, которые задают категорию области полей. Описание полей таблицы представлено в таблице 59.

Таблица 59 – Описание полей таблицы `DocumentAPI.PivotTableFieldCategory`

Поле	Описание
<code>DocumentAPI.PivotTableFieldCategory_Pages</code>	Область фильтров
<code>DocumentAPI.PivotTableFieldCategory_Rows</code>	Область строк
<code>DocumentAPI.PivotTableFieldCategory_Columns</code>	Область колонок
<code>DocumentAPI.PivotTableFieldCategory_Values</code>	Область значений

7.103 Таблица `DocumentAPI.PivotTableFieldProperties`

`DocumentAPI.PivotTableFieldProperties` содержит свойства поля [`DocumentAPI.PivotTableField`](#) сводной таблицы (см. таблицу 60).

Таблица 60 – Описание полей таблицы `DocumentAPI.PivotTableFieldProperties`

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

7.104 Таблица DocumentAPI.PivotTableFilter

Позволяет осуществить доступ к списку фильтров таблицы, каждый из которых обладает свойством видимости (Рис. 43).

. При любом изменении фильтров они должны быть применены к сводной таблице посредством использования методов [PivotTableEditor.setFilter\(\)](#), [PivotTableEditor.setFilters\(\)](#).

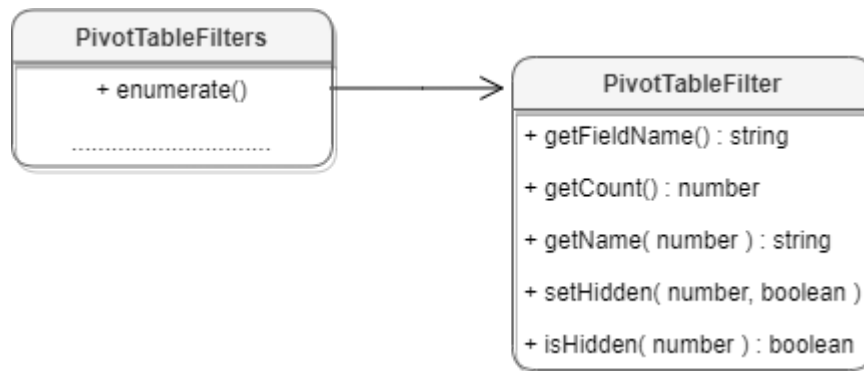


Рисунок 43 – Таблица DocumentAPI.PivotTableFilter

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    for filterIdx = 0, filter:getCount() - 1 do
        filter:setHidden(filterIdx, false)
    end
end
pivotTableEditor:setFilters(filters)
pivotTableEditor:apply()
```

7.104.1 Метод PivotTableFilter:getCount

Возвращает количество фильтруемых полей.

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    print(filter:getCount())
end
```

7.104.2 Метод PivotTableFilter:getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    for filterIdx = 0, filter:getCount() - 1 do
        print(filter:getFieldName())
    end
end
```

7.104.3 Метод PivotTableFilter:getName

Возвращает имя поля для заданного индекса.

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    for filterIdx = 0, filter:getCount() - 1 do
        print(filter:getName(filterIdx))
    end
end
```

7.104.4 Метод PivotTableFilter:isHidden

Возвращает видимость поля для заданного индекса `itemIndex`. Если `true`, то поле скрыто.

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    for filterIdx = 0, filter:getCount() - 1 do
        print(filter:isHidden(filterIdx))
    end
end
```

7.104.5 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.

7.104.6 Метод PivotTableFilter:reset

Метод сбрасывает настройки текущего фильтра.

Вызов

```
reset()
```

7.104.7 Метод PivotTableFilter:setHidden

Устанавливает видимость поля для заданного индекса. Параметры: `itemName` – индекс поля, `hidden` – видимость (true – поле скрыто).

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    for filterIdx = 0, filter:getCount() - 1 do
        print(filter:setName(filterIdx, false))
    end
end
```

7.105 Таблица DocumentAPI.PivotTableFilters

Таблица обеспечивает доступ к списку фильтров. Для получения `DocumentAPI.PivotTableFilters` используется метод [PivotTable:getFilters\(\)](#).

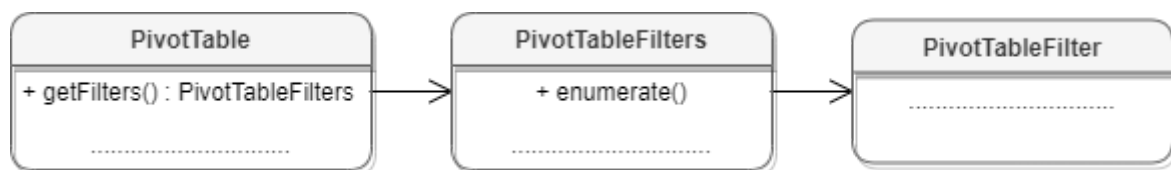


Рисунок 44 – Объектная модель таблиц для работы с фильтрами

7.105.1 Метод PivotTableFilters:enumerate

Метод используется для доступа к коллекции фильтров (см. [DocumentAPI.PivotTableFilter](#)).

Пример

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
```

```
print(filter:getName(0))
print(filter:getFieldName())
end
```

7.106 Таблица DocumentAPI.PivotTableFunction

Таблица DocumentAPI.PivotTableFunction описывает функции, которые могут быть использованы в сводных таблицах. Описание полей таблицы представлено в таблице 61. Таблица используется в качестве поля subtotalFunctions таблицы [DocumentAPI.PivotTableCategoryField](#).

Таблица 61 – Описание полей таблицы DocumentAPI.PivotTableFunction

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

7.107 Таблица DocumentAPI.PivotTableItem

DocumentAPI.PivotTableItem описывает элемент сводной таблицы (см. Рисунок 45). См. пример в главе [PivotTableItems:enumerate](#).

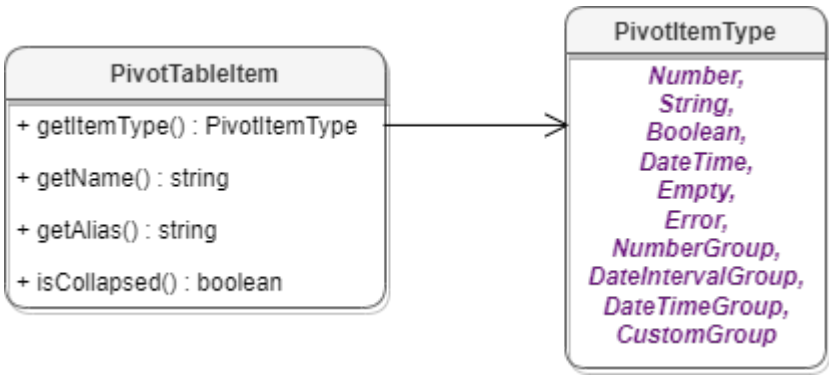


Рисунок 45 – Таблица DocumentAPI.PivotTableItem

7.107.1 Метод PivotTableItem:getAlias

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

7.107.2 Метод PivotTableItem:getItemType

Метод возвращает тип [DocumentAPI.PivotTableItemType](#) элемента сводной таблицы. См. пример в главе [PivotTableItems:enumerate](#).

7.107.3 Метод PivotTableItem:getName

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

7.107.4 Метод PivotTableItem:isCollapsed

Метод возвращает true, если элемент сводной таблицы свернут. См. пример в главе [PivotTableItems:enumerate](#).

7.108 Таблица DocumentAPI.PivotTableItems

Таблица обеспечивает доступ к списку элементов сводной таблицы (см. Рисунок 46).

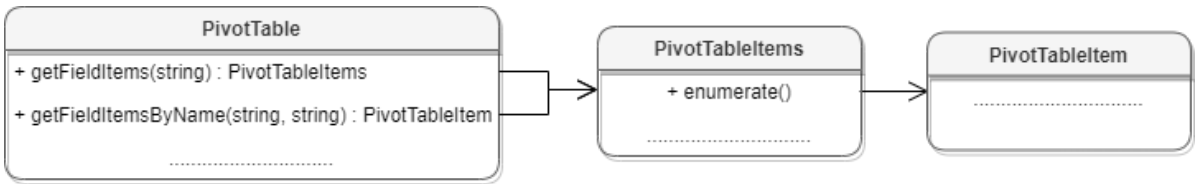


Рисунок 46 – Объектная модель таблиц для работы с элементами сводных таблиц

7.108.1 Метод PivotTableItems:enumerate

Используется для перечисления элементов сводной таблицы.

Пример

```
local fieldItems = pivotTable:getFieldItems("Age")
for fieldItem in fieldItems:enumerate() do
    print(fieldItem:getName())
    print(fieldItem:getAlias())
    print(fieldItem:getItemType())
    print(fieldItem:isCollapsed())
end
```

7.109 Таблица DocumentAPI.PivotTableItemType

Таблица DocumentAPI.PivotTableItemType содержит возможные типы элементов сводной таблицы. Описание полей таблицы представлено в таблице 62.

Таблица 62 – Описание полей таблицы DocumentAPI.PivotTableItemType

Поле	Описание
DocumentAPI.PivotTableItemType_Number	Числовой
DocumentAPI.PivotTableItemType_String	Строковый
DocumentAPI.PivotTableItemType_Boolean	Логический
DocumentAPI.PivotTableItemType_DateTime	Дата / время
DocumentAPI.PivotTableItemType_Empty	Пустой тип
DocumentAPI.PivotTableItemType_Error	Ошибка
DocumentAPI.PivotTableItemType_NumberGroup	Интервальная группировка
DocumentAPI.PivotTableItemType_DateIntervalGroup	Интервальная группировка по датам
DocumentAPI.PivotTableItemType_DateTimeGroup	Группировка по дате / времени
DocumentAPI.PivotTableItemType_CustomGroup	Пользовательская (произвольная) группировка

Пример

```
local fieldItems = pivotTable:getFieldItems("Age")
for fieldItem in fieldItems:enumerate() do
    if (fieldItem:getItemType() == DocumentAPI.PivotTableItemType_Number) then
        print("Numeric type")
    end
end
```



```
end  
end
```

7.110 Таблица DocumentAPI.PivotTableLayoutSettings

Таблица `DocumentAPI.PivotTableLayoutSettings` содержит настройки отображения сводной таблицы. Данная таблица может быть получена в результате вызова [PivotTable.getPivotTableLayoutSettings\(\)](#) и установлена методом [PivotTableEditor.setLayoutSettings\(\)](#). Описание полей таблицы представлено в таблице 63.

Таблица 63 – Описание полей таблицы `DocumentAPI.PivotTableLayoutSettings`

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

7.111 Таблица DocumentAPI.PivotTablePageField

Содержит свойства поля из области фильтров (см. таблицу 64). Таблица может быть получена посредством вызова [PivotTable.getPageFields\(\)](#).

Таблица 64 – Описание полей таблицы `DocumentAPI.PivotTablePageField`

Поле	Описание
<code>PivotTablePageField.fieldProperties</code>	Свойства поля PivotTableFieldProperties

7.112 Таблица `DocumentAPI.PivotTableReportLayout`

Таблица `DocumentAPI.PivotTableReportLayout` описывает внешний вид отчетов сводной таблицы. Является полем таблицы [DocumentAPI.PivotTableLayoutSettings](#). Описание полей таблицы представлено в таблице 65.

Таблица 65 – Описание полей таблицы `DocumentAPI.PivotTableReportLayout`

Поле	Описание
<code>DocumentAPI.PivotTableReportLayout_Compact</code>	Компактный вид
<code>DocumentAPI.PivotTableReportLayout_Tabular</code>	Табличный вид
<code>DocumentAPI.PivotTableReportLayout_Outline</code>	Структурный вид

7.113 Таблица `DocumentAPI.PivotTablesManager`

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

Пример

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

7.113.1 Метод `PivotTablesManager:create`

Метод создает сводную таблицу [PivotTable](#) на основе диапазона исходных данных [CellRange](#).

Если местоположение не задано, создается новый лист (таблица), и сводная таблица будет расположена по умолчанию.

Пример

```
local pivotTablesManager = document.getPivotTablesManager()
local tbl = document.getBlocks():getTable(0)
local cellRange = tbl.getCellRange("I3:K7")
local pivotTable = pivotTablesManager.create(cellRange, tbl.getCell("L8"))
```

7.114 Таблица DocumentAPI.PivotTableUpdateResult

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

Таблица 66 – Результаты обновления сводной таблицы

Наименование константы	Описание
DocumentAPI.PivotTableUpdateResult_Success	Успешное обновление таблицы
DocumentAPI.PivotTableUpdateResult_NoPivotTable	Сводная таблица не найдена
DocumentAPI.PivotTableUpdateResult_NoSuchFieldInCategory	Не найдено поле в категории
DocumentAPI.PivotTableUpdateResult_NoSuchFieldInPivotTable	Не найдено поле в сводной таблице
DocumentAPI.PivotTableUpdateResult_InvalidIndex	Ошибка в индексе
DocumentAPI.PivotTableUpdateResult_FieldAlreadyEnabled	Поле уже существует
DocumentAPI.PivotTableUpdateResult_MovingFieldToTheSameCategoryForbidden	Попытка перемещения поля в рамках текущей категории
DocumentAPI.PivotTableUpdateResult_InvalidFunction	Неправильная функция
DocumentAPI.PivotTableUpdateResult_InvalidCategory	Неправильная область
DocumentAPI.PivotTableUpdateResult_InvalidDataSourceRange	Ошибка диапазона исходных данных
DocumentAPI.PivotTableUpdateResult_NoDataRowsInDataSource	В исходных данных нет строк с данными
DocumentAPI.PivotTableUpdateResult_EmptyDataSourceHeaders	Пустые заголовки исходных данных
DocumentAPI.PivotTableUpdateResult_NoReferenceUnderDefine	Попытка обновить или создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу
DocumentAPI.PivotTableUpdateResult_NoSuchItem	Элемент не найден
DocumentAPI.PivotTableUpdateResult_CannotExpandCollapseLeafItem	Не удастся раскрыть свернутый элемент

Наименование константы	Описание
<code>DocumentAPI.PivotTableUpdateResult_AnotherPivotInsideDataSource</code>	Найдена другая сводная таблица в этом же диапазоне
<code>DocumentAPI.PivotTableUpdateResult_Canceled</code>	Обновление сводной таблицы отменено
<code>DocumentAPI.PivotTableUpdateResult_NoSuchSheetInExternalDocument</code>	Не найден лист во внешнем документе

7.115 Таблица `DocumentAPI.PivotTableValueField`

`DocumentAPI.PivotTableValueField` содержит свойства поля сводной таблицы, использующегося как значение столбец (см. таблицу 67). Таблица может быть получена посредством вызова [PivotTable:getValueFields\(\)](#).

Таблица 67 – Описание полей таблицы `DocumentAPI.PivotTableValueField`

Поле	Описание
<code>PivotTableValueField baseFieldName</code>	Оригинальное поле на основе которого было создано данное поле, тип - строка.
<code>PivotTableValueField valueFieldName</code>	Автоматический уникальный псевдоним такой как "Sum of %имя поля%", тип - строка.
<code>PivotTableValueField cellNumberFormat</code>	Числовой формат типа CellFormat для конкретного поля значений.
<code>PivotTableValueField totalFunction</code>	Агрегирующая функция PivotTableFunction поля значений (SUM, COUNT, MAX и т.д.).
<code>PivotTableValueField customFormula</code>	Вычисляемая формула для поля значений, тип - строка.

7.116 Таблица `DocumentAPI.PointU`

Таблица `DocumentAPI.PointU` представляет позицию объекта (x, y). Описание полей таблицы `DocumentAPI.PointU` представлено в таблице 68.

Таблица 68 – Описание полей таблицы `DocumentAPI.PointU`

Поле	Описание
<code>DocumentAPI.PointU.x</code>	Позиция x
<code>DocumentAPI.PointU.y</code>	Позиция y

Пример

```
local point = DocumentAPI.PointU(2, 3)
print("x=", point.x, ", y=", point.y)  --(x = 2.0, y = 3.0)
```

7.116.1 Метод PointU:toString

Возвращает информацию о позиции в виде строкового значения формата (width: <value>, height: <value>).

Пример

```
local point = DocumentAPI.PointU(2, 3)
print(point:toString())  --(x: 2.0, y: 3.0)
```

7.117 Таблица DocumentAPI.Position

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

7.117.1 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.
- nil, если позиции нельзя сравнить (позиции в разных документах, в тексте и в колонтитуле и т.д.).

Примеры для текстового документа

```
table1 = document:getRange():getBegin():insertTable(2, 2, "table1")
positionDoc = document:getRange():getBegin()
positionTable = table1:getRange():getBegin()
positionCell = table1:getCell(DocumentAPI.CellPosition(0,
0)):getRange():getBegin()
```

```
if (positionDoc:compare(positionTable) + positionTable:compare(positionCell) ==
0) then
    -- true
end
```

```
document:getRange():getBegin():insertText("Some text")
positionBegin = document:getRange():getBegin()
positionEnd = document:getRange():getEnd()

print(positionBegin:compare(positionEnd)) -- -10
print(positionEnd:compare(positionBegin)) -- 10
```

Пример для табличного документа

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

firstCellEnd = sheet:getCell("A1"):getRange():getEnd()
secondCellBegin = sheet:getCell("B1"):getRange():getBegin()

print(firstCellEnd:compare(secondCellBegin)) -- 0
```

7.117.2 Метод Position:getCell

Метод возвращает ячейку, в которой находится позиция, либо nil если позиция не находится в ячейке.

Пример для табличного документа

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")
local range = cell:getRange()
local pos = range:getBegin()
print(pos:getCell())
```

Пример для текстового документа

```
local range = document:getRange()
local pos = rng:getBegin()
print(pos:getCell())
```

7.117.3 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

```
local docBegin = document:getRange():getBegin()
local sentence = docBegin:getCurrentRange(DocumentAPI.TextUnit_Sentence)
local word = docBegin:getCurrentRange(DocumentAPI.TextUnit_Word)
local character = docBegin:getCurrentRange(DocumentAPI.TextUnit_Character)

print(sentence:extractText())
-- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.
print(word:extractText()) -- Lorem
print(character:extractText()) -- L
```

7.117.4 Метод Position:getNextPosition

Метод возвращает позицию, которая находится на заданном расстоянии после текущей. Если результат выходит за пределы текущего параграфа, метод возвращает конец параграфа.

Вызов

```
Position getNextPosition(count)
```

Параметры

– count: (необязательный, по умолчанию 1) расстояние до следующей позиции.

Возвращает

– следующая позиция, тип [Position](#).

Пример

```
document:getRange():getBegin():insertText("New text")
document:getRange():getBegin():getNextPosition(2):insertText("!")
print(document:getRange():extractText()) -- Ne!w text
```

7.117.5 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 параграфом.

Вызов

```
Range getNextRange(textUnit)
```

Параметры

– textUnit: единица текста, определяющая размеры диапазонов, тип [TextUnit](#).

Возвращает

– следующий диапазон документа, тип [Range](#).

Пример

```
local docBegin = document:getRange():getBegin()
local nextSentence = docBegin:getNextRange(DocumentAPI.TextUnit_Sentence)
local nextWord =
docBegin:getNextRange(DocumentAPI.TextUnit_Word):getBegin():getNextRange(DocumentAPI.TextUnit_Word)
local nextCharacter = docBegin:getNextRange(DocumentAPI.TextUnit_Character)

print(nextSentence:extractText())
-- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
print(nextWord:extractText()) -- ipsum
print(nextCharacter:extractText()) -- o
```

7.117.6 Метод Position:getParagraph

Метод возвращает параграф [Paragraph](#) для текущей позиции.

Пример

```
local position = document:getRange():getBegin()
local paragraphAtPosition = position:getParagraph()
```

7.117.7 Метод Position:getPreviousPosition

Метод возвращает позицию, которая находится на заданном расстоянии перед текущей. Если результат выходит за пределы текущего параграфа, метод возвращает начало параграфа.

Вызов

```
Position getPreviousPosition(count)
```

Параметры

- count: (необязательный, по умолчанию 1) расстояние до предыдущей позиции.

Возвращает

- предыдущая позиция, тип [Position](#).

Пример

```
document:getRange():getBegin():insertText("New text")
document:getRange():getContentEnd():getPreviousPosition(2):insertText("!")
print(document:getRange():extractText()) -- New te!xt
```

7.117.8 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 параграфом.

Вызов

```
Range getPreviousRange(textUnit)
```

Параметры

- textUnit: единица текста, определяющая размеры диапазонов, тип [TextUnit](#).

Возвращает

- предыдущий диапазон документа, тип [Range](#).

Пример

```
local sentenceBegin =
document:getRange():getBegin():getNextRange(DocumentAPI.TextUnit_Sentence):getBe
gin()
local previousSentence =
sentenceBegin:getPreviousRange(DocumentAPI.TextUnit_Sentence)
local previousWord = sentenceBegin:getPreviousRange(DocumentAPI.TextUnit_Word)
while previousWord:getContentEnd():compare(previousWord:getBegin()) == 1 do
    previousWord =
previousWord:getBegin():getPreviousRange(DocumentAPI.TextUnit_Word)
end
local previousCharacter =
sentenceBegin:getPreviousRange(DocumentAPI.TextUnit_Character)
previousCharacter =
previousCharacter:getBegin():getPreviousRange(DocumentAPI.TextUnit_Character)
previousCharacter =
previousCharacter:getBegin():getPreviousRange(DocumentAPI.TextUnit_Character)
```

```
print(previousSentence:extractText())  
-- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
tempor incididunt ut labore et dolore magna aliqua.  
print(previousWord:extractText()) -- aliqua  
print(previousCharacter:extractText()) -- a
```

7.117.9 Метод Position:insertBookmark

Вставляет закладку с наименованием в заданную позицию текстового документа.



Внимание ! Метод может быть использован только в текстовом редакторе.

Пример

```
document:getRange():getBegin():insertBookmark("Bookmark example")
```

7.117.10 Метод Position:insertHyperlink

Метод `insertHyperlink` вставляет ссылку в текущую позицию. В качестве параметров передаются адрес ссылки и текст ссылки.

Вызов

```
insertHyperlink( url, size )
```

Параметры

- url – адрес ссылки;
- label – текст ссылки.

Пример

```
document:getRange():getBegin():insertHyperlink("https://testhyperlink.com",  
"Hyperlink")
```

7.117.11 Метод Position:insertImage

Вставляет изображение в позицию текстового документа.



Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

Вызов

```
Image insertImage(url, size)
```

Параметры

- url – полный путь к локальному файлу, либо ссылка на сетевой ресурс;
- size – геометрические размеры изображения для вставки.

Возвращает

- [Image](#): вставленное изображение.

Примеры

```
local image = document:getRange():getBegin():insertImage("C://Tmp//123.jpg",  
DocumentAPI.SizeU(100, 100))
```

```
local image =  
document:getRange():getBegin():insertImage  
("https://www.images.ru/images/fish.jpg", DocumentAPI.SizeU(50, 50))
```

7.117.12 Метод Position:insertLineBreak

Метод предназначен для вставки перевода строки в указанную позицию текстового документа.



Внимание ! Метод может быть использован только в текстовом редакторе.

Пример

```
local rng = document:getRange()  
local end_pos = rng:getEnd()  
end_pos:insertLineBreak()
```

7.117.13 Метод Position:insertPageBreak

Метод предназначен для вставки разрыва страницы в указанную позицию текстового документа.



Внимание ! Метод может быть использован только в текстовом редакторе.

Пример

```
local rng = document:getRange()  
local end_pos = rng:getEnd()  
end_pos:insertPageBreak()
```

7.117.14 Метод `Position:insertSectionBreak`

Вставляет разрыв раздела в текущую позицию текстового документа.



Внимание ! Метод может быть использован только в текстовом редакторе.

Пример

```
document:getRange():getBegin():insertSectionBreak()
```

7.117.15 Метод `Position:insertTable`

Метод предназначен для вставки таблицы с заданным числом строк и столбцов в заданное местоположение в документе. Возвращает объект таблицы.

Следует учитывать, что при вставке таблицы к ее имени автоматически добавляется порядковый номер, начинающийся с единицы. Таким образом, вызов

```
t = position:insertTable(3, 3, "Table")
```

приведет к созданию в текстовом документе таблицы с именем «Table1».

Пример вставки таблицы в начало текстового документа

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

Пример вставки таблицы в конец текстового документа

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

В табличном документе данный метод используется для вставки нового рабочего листа.

Пример вставки нового листа в табличный документ

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

7.117.16 Метод `Position:insertText`

Метод предназначен для вставки текстовой строки в заданное местоположение в документе.

Пример

```
local rng = document:getRange()  
local begin_pos = rng:getBegin()  
begin_pos:insertText("Текст в начале строки")
```

7.117.17 Метод `Position:removeBackward`

Метод удаляет `count` объектов (символов, картинок и т.д.) до текущей позиции.

Пример

```
document:getRange():getEnd():removeBackward(3)
```

7.117.18 Метод `Position:removeForward`

Метод удаляет `count` объектов (символов, картинок и т.д.) после текущей позиции.

Пример

```
document:getRange():getBegin():removeForward(3)
```

7.117.19 Метод `Position:__eq`

Метод используется для определения эквивалентности значений двух местоположений в документе.

Пример

```
print(document:getRange():getBegin():__eq(document:getRange():getEnd()))
```

7.118 Таблица `DocumentAPI.PrintDocumentResult`

В таблице 69 представлены коды, возвращаемые после печати (см. [EditorAPI.showPrintDialog\(\)](#)).

Таблица 69 – Коды, возвращаемые после печати

Наименование константы	Описание
<code>DocumentAPI.PrintDocumentResult_Success</code>	Печать прошла успешно
<code>DocumentAPI.PrintDocumentResult_OneCopyPrinted</code>	Напечатана только одна копия из заданных
<code>DocumentAPI.PrintDocumentResult_CancelPrinting</code>	Печать была отменена
<code>DocumentAPI.PrintDocumentResult_NoPrinter</code>	Принтер не найден
<code>DocumentAPI.PrintDocumentResult_BlankDocument</code>	На печать отправлен пустой документ

7.119 Таблица DocumentAPI.PrintSettings

Таблица DocumentAPI.PrintSettings представляет установки, используемые при печати документов. Описание полей таблицы DocumentAPI.PrintSettings представлено в таблице 70. Используется в [EditorAPI.printDocument](#).

Таблица 70 – Описание полей таблицы DocumentAPI.PrintSettings

Поле	Тип	Описание
DocumentAPI.PrintSettings.printerName	string	Имя используемого принтера. Если не указано, то используется принтер по умолчанию. Если принтер с указанным именем недоступен, то возникает ошибка.
DocumentAPI.PrintSettings.landscapeOrientation	bool	Если значение равно true, то размер страницы поворачивается на 90 градусов. В настоящее время используется только для рабочих таблиц.
DocumentAPI.PrintSettings.leftMargin	number	Ширина левого поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
DocumentAPI.PrintSettings.topMargin	number	Ширина верхнего поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
DocumentAPI.PrintSettings.rightMargin	number	Ширина правого поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
DocumentAPI.PrintSettings.bottomMargin	number	Ширина нижнего поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
DocumentAPI.PrintSettings.parity	PageParity	Выбор страниц для печати.
DocumentAPI.PrintSettings.firstPage	number	Номер первой страницы для печати.

Поле	Тип	Описание
DocumentAPI.PrintSettings.lastPage	number	Номер последней страницы для печати.
DocumentAPI.PrintSettings.printSelection	bool	Область печати. Значение по умолчанию false.
DocumentAPI.PrintSettings.worksheetPrinterFitType	WorksheetPrinterFitType	Вариант масштабирования при печати табличных документов.
DocumentAPI.PrintSettings.copies	number	Количество копий при печати.
DocumentAPI.PrintSettings.collateCopies	bool	Если параметр имеет значение false, то печать каждой отдельной страницы будет повторена заданное количество копий раз до начала печати следующей страницы. Если параметр имеет значение true, то все страницы печатаются до запуска печати следующей копии этих страниц. Значение по умолчанию false.

7.120 Таблица DocumentAPI.Range

Таблица DocumentAPI.Range предоставляет доступ к диапазону документа. На рисунке 47 изображена объектная модель таблиц, относящихся к работе с диапазонами.

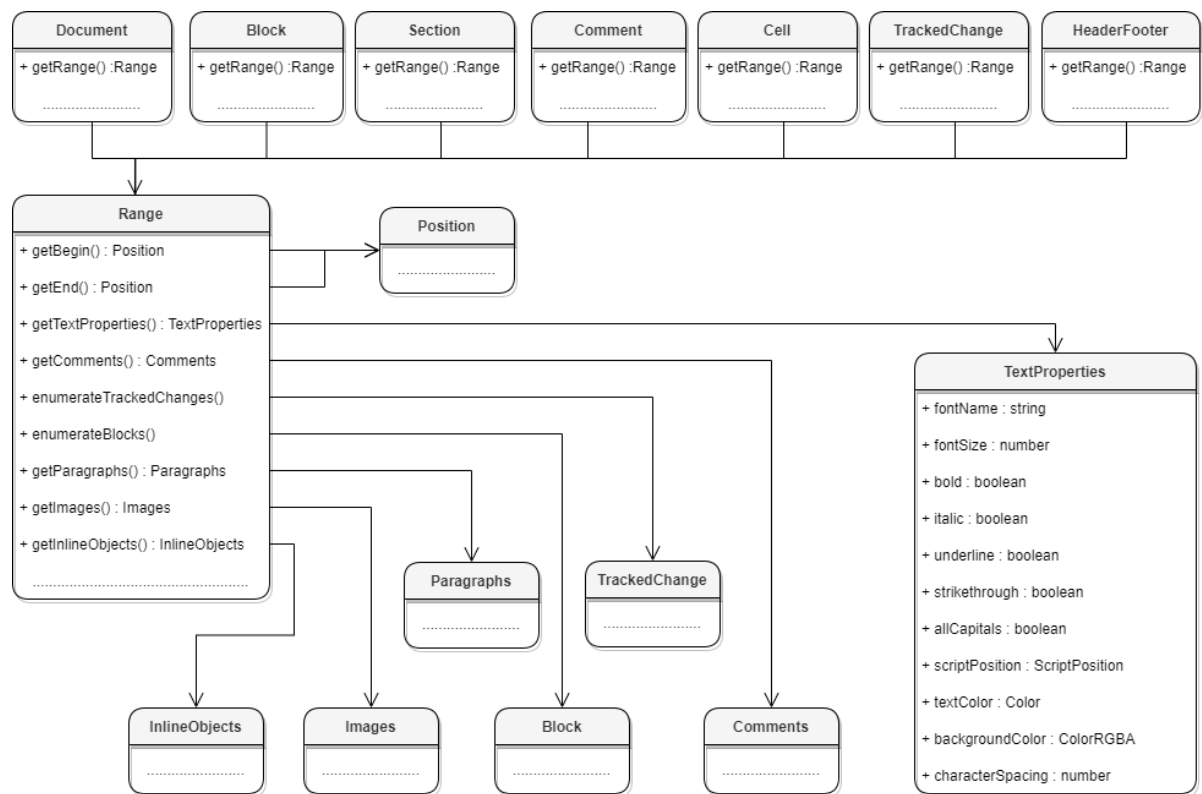


Рисунок 47 – Объектная модель для работы с таблицей DocumentAPI . Range

Варианты получения диапазона для текстового документа

```

-- диапазон всего документа
documentRange = document:getRange()

-- диапазон блока
block = document:getBlocks():getBlock(0)
blockRange = block:getRange()

-- диапазон секций
sections = document:getSections()
for section in sections:enumerate() do
    sectionRange = section:getRange()
end

-- диапазон комментариев
commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    commentRange = comment:getRange()
end

-- диапазон ячейки
table = document:getBlocks():getTable(0)
cell = table:getCell("B2")
cellRange = cell:getRange()

```



```
-- диапазон верхних колонтитулов
section = document:getBlocks():getBlock(0):getSection()
headers = section:getHeaders()
for header in headers:enumerate() do
    headerRange = header:getRange()
end
-- диапазон отслеживаемых изменений
local trackedChangesList = document:getRange():enumerateTrackedChanges()
for trackedChange in trackedChangesList do
    trackedChangeRange = trackedChange:getRange()
end
```

7.120.1 Конструктор DocumentAPI.Range

Для создания объекта [DocumentAPI.Range](#) вы можете использовать следующий конструктор:

```
documentRange = DocumentAPI.Range(begin, end)
```

Параметры

- begin: начальная позиция диапазона, тип [DocumentAPI.Position](#);
- end: конечная позиция диапазона, тип [DocumentAPI.Position](#).

7.120.2 Метод Range:enumerateBlocks

Предоставляет возможность итерации по блокам.

Пример для текстового документа

```
local range = document:getRange()
for block in range:enumerateBlocks() do
    print(block:getRange():extractText())
end
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
for block in range:enumerateBlocks() do
    print(block:getRange():extractText())
end
```

7.120.3 Метод Range:enumerateTrackedChanges

Предоставляет возможность итерации по отслеживаемым изменениям [DocumentAPI.TrackedChange](#). Метод может быть использован только в текстовых документах.

Пример

```
local changesList = document:getRange():enumerateTrackedChanges()  
for change in changesList do  
    print(change:getRange():extractText())  
end
```

7.120.4 Метод Range:extractText

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и другие объекты игнорируются.

Пример для текстового документа

```
local range = document:getRange() -- содержимое всего документа  
local text = range:extractText()  
print (text)
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)  
local cell = tbl:getCell("B2")  
local range = cell:getRange() -- содержимое ячейки B2  
print (range:extractText())
```

7.120.5 Метод Range:getBegin

Метод возвращает позицию в начале диапазона.

Пример для текстового документа

```
local range = document:getRange() -- содержимое всего документа  
local pos = range:getBegin() -- в начало документа  
pos:insertText("Привет")
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)  
local cell = tbl:getCell("B2")  
local range = cell:getRange() -- содержимое ячейки B2
```

```
local pos = range:getBegin() -- в начало ячейки
pos:insertText("Привет")
```

7.120.6 Метод Range:getComments

Обеспечивает доступ к комментариям в диапазоне.

Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам. Если дат нет, то порядок комментариев не определен.

Пример

```
local comments = document:getRange():getComments()
for comment in comments:enumerate() do
    print(comment:getRange())
    print(comment:getText())
    print(comment:getInfo().author)
    print(comment:getInfo().timeStamp)
    print(comment:isResolved())
    print(comment:getReplies())
end
```

7.120.7 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 параграфа/ячейки. Чтобы получить позицию конца диапазона с переходом к следующему параграфу/ячейке, позовите метод [Range:getEnd](#).

Пример для текстового документа

```
document:getRange():getBegin():insertText("Text")
document:getRange():getContentEnd():insertText("#")
print(document:getRange():extractText()) -- Text#\n
```

Пример для табличного документа

```
local cell = document:getBlocks():getTable(0):getCell("B2")
cell:setText("Text")
local cellEnd = cell:getRange():getContentEnd()
```

```
cellEnd:insertText("#")
print(cell:getRawValue()) -- Text#
```

7.120.8 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 параграфу/ячейке. Чтобы получить позицию конца диапазона для добавления в него информации, позовите метод [Range:getContentEnd](#).

Пример для текстового документа

```
document:getRange():getBegin():insertText("First Paragraph")
local endDocPosition = document:getRange():getEnd()
endDocPosition:insertText("Second Paragraph")
print(document:getRange():extractText()) -- First Paragraph\nSecond Paragraph\n
```

Пример для табличного документа

```
local sheet = document:getBlocks():getTable(0)
sheet:getCell("B2"):setText("Text")
local cellEnd = sheet:getCell("B2"):getRange():getEnd()
cellEnd:insertText("#")
print(sheet:getCell("B2"):getRawValue()) -- Text
print(sheet:getCell("C2"):getRawValue()) -- #
```

7.120.9 Метод Range:getImages

Обеспечивает доступ к изображениям ([DocumentAPI.Image](#)) в диапазоне.



Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

Примеры

```
local images = document:getRange():getImages()
for image in images:enumerate() do
    print(image:getFrame():getWrapType())
end

for image in EditorAPI.getSelection():getImages():enumerate() do
    print(image:getFrame():getWrapType())
end
```

7.120.10 Метод Range:getInlineObjects

Обеспечивает доступ к перечислению [DocumentAPI.MediaObjects](#) графических объектов диапазона.



Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

Пример

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    print(mediaObject)
end
```

7.120.11 Метод Range:getParagraphs

Обеспечивает доступ к абзацам [DocumentAPI.Paragraphs](#) в диапазоне.

Пример для текстового документа

```
local paragraphs = document:getRange():getParagraphs()
for para in paragraphs:enumerate() do
    print(para:getRange():extractText())
end
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
    print(para:getRange():extractText())
end
```

7.120.12 Метод Range:getTextProperties

Метод возвращает таблицу с текущими настройками форматирования для фрагмента текстового документа. Описание настроек форматирования осуществляется с помощью таблицы [DocumentAPI.TextProperties](#).

Пример для текстового документа

```
local range = document:getRange()  
local props = range:getTextProperties()  
print(props.italic)
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)  
local cell = tbl:getCell("B2")  
local range = cell:getRange()  
local props = range:getTextProperties()  
print(props.italic)
```

7.120.13 Метод Range:isContentLocked

Метод возвращает значение true, если изменения содержимого диапазона запрещены.

Пример для текстового документа

```
local range = document:getRange() -- содержимое всего документа  
if range:isContentLocked() then  
    print("Документ содержит заблокированное содержимое")  
end
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)  
local cell = tbl:getCell("B2")  
local range = cell:getRange() -- содержимое ячейки  
if range:isContentLocked() then  
    print("Ячейка содержит заблокированное содержимое")  
end
```

7.120.14 Метод Range:lockContent

Метод запрещает изменения содержимого диапазона.



Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа

```
local range = document:getRange() -- содержимое всего документа  
range:lockContent()
```

Пример для таблицы внутри текстового документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:lockContent()
```

7.120.15 Метод Range:removeContent

Метод полностью удаляет содержимое диапазона.

Пример для текстового документа

```
local range = document:getRange() -- содержимое всего документа
range:removeContent()
print (range:extractText())
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:removeContent()
print (range:extractText())
```

7.120.16 Метод Range:replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример для текстового документа

```
local range = document:getRange() -- содержимое всего документа
range:replaceText("НОВЫЙ текст")
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки таблицы
range:replaceText("НОВЫЙ текст")
```

7.120.17 Метод Range:setHyperlink

Метод setHyperlink вставляет ссылку в содержимое диапазона и заменяет его текст ТЕКСТОМ ССЫЛКИ.

Вызов

```
setHyperlink( url, label )
```

Параметры

- url – адрес ссылки;
- label – текст ссылки.

Пример для текстового документа

```
local range = document:getRange()  
range:setHyperlink("https://testhyperlink.com", "Hyperlink")
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)  
local cell = tbl:getCell("B2")  
local range = cell:getRange()  
range:setHyperlink("https://testhyperlink.com", "Hyperlink")  
print(cell:getFormattedValue())
```

7.120.18 Метод Range:setTextProperties

Метод применяет настройки форматирования [DocumentAPI.TextProperties](#) для диапазона.

Пример для текстового документа

```
local range = document:getRange()  
local props = range:getTextProperties()  
props.italic = true  
range:setTextProperties(props) -- текстовый фрагмент оформлен курсивом
```

Пример для табличного документа

```
local tbl = document:getBlocks():getTable(0)  
local cell = tbl:getCell("B2")  
local range = cell:getRange()  
local props = range:getTextProperties()  
props.italic = true  
range:setTextProperties(props)
```

7.120.19 Метод Range:unlockContent

Метод разрешает изменения содержимого диапазона.



Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа

```
local range = document:getRange() -- содержимое всего документа
range:unlockContent()
```

Пример для таблицы внутри текстового документа

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:unlockContent()
```

7.121 Таблица DocumentAPI.RangeBorders

Таблица DocumentAPI.RangeBorders оставлена для совместимости. Вместо нее необходимо использовать таблицу [DocumentAPI.Borders](#).

7.122 Таблица DocumentAPI.RectU

Таблица DocumentAPI.RectU представляет описание прямоугольной области. Описание полей таблицы DocumentAPI.RectU представлено в таблице 71.

Таблица 71 – Описание полей таблицы DocumentAPI.RectU

Поле	Описание
DocumentAPI.RectU.topLeft	Координаты левого верхнего угла области, тип CellPosition
DocumentAPI.RectU.rightBottom	Координаты правого нижнего угла области, тип CellPosition

Пример

```
local rect = DocumentAPI.RectU(2, 3, 4, 5)
print("tlx=", rect.topLeft.x, ", tly=", rect.topLeft.y, ", rbx=",
rect.bottomRight.x, ", rby=", rect.bottomRight.y) --(tlx = 2.0, tly = 3.0, brx
= 4.0, bry = 5.0)
```

7.122.1 Метод RectU:toString

Возвращает информацию о прямоугольной области в виде строкового описания координат [topLeft: (x: <value>, y: <value>), bottomRight: (x: <value>, y: <value>)].

Пример

```
local point = DocumentAPI.RectU(2, 3, 4, 5)
print(point.toString()) --[topLeft: (x: 2.0, y: 3.0), bottomRight: (x: 4.0, y: 5.0)]
```

7.123 Таблица DocumentAPI.ScaleFrom

В таблице 72 представлены позиции объекта, остающиеся неизменными при масштабировании объекта. Используется в [AbsoluteFrame.scale\(\)](#).

Таблица 72 – Неизменные позиции объекта при масштабировании

Наименование константы	Позиция
DocumentAPI.ScaleFrom_BottomRight	Правый нижний угол
DocumentAPI.ScaleFrom_BottomLeft	Левый нижний угол
DocumentAPI.ScaleFrom_TopLeft	Левый верхний угол
DocumentAPI.ScaleFrom_TopRight	Правый верхний угол

7.124 Таблица DocumentAPI.ScientificCellFormatting

Таблица содержит параметры для экспоненциального формата ячеек таблицы. Данная таблица используется в качестве аргумента метода [Cell:setFormat\(\)](#). Описание полей таблицы DocumentAPI.ScientificCellFormatting представлено в таблице 73.

Таблица 73 – Описание полей таблицы DocumentAPI.ScientificCellFormatting

Поле	Описание
DocumentAPI.ScientificCellFormatting.decimalPlaces	Количество десятичных позиций
DocumentAPI.ScientificCellFormatting.minExponentDigits	Минимальное количество позиций экспоненты

Пример

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("B2")

local scientificCellFormatting = DocumentAPI.ScientificCellFormatting()
scientificCellFormatting.decimalPlaces = 2
scientificCellFormatting.minExponentDigits = 3
```

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

7.125 Таблица DocumentAPI.Script

Таблица DocumentAPI.Script предназначена для управления отдельной макрокомандой. Таблица содержит поля Name и Body.

7.125.1 Метод Script:getBody

Метод возвращает текст макрокоманды в виде строки.

Пример

```
local scripts = document:getScripts()
local script = scripts:getScript("Enumerate scripts for document")
local scriptBody = script:getBody()
print(scriptBody)
```

7.125.2 Метод Script:getName

Метод возвращает имя макрокоманды.

Пример

```
local scripts = document:getScripts()
local sc = scripts:getScript("Enumerate scripts for document")
print(sc:getName())
```

7.125.3 Метод Script:setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

Пример

```
local scripts = document:getScripts()
local script = scripts:getScript("Enumerate scripts for document")
script:setBody("local scripts = document:getScripts()\nfor script in\nscripts:enumerate() do\nprint(script:getName())\nend")
```

7.125.4 Метод Script:setName

Метод устанавливает имя для макрокоманды.

Пример

```
local scripts = document:getScripts()
local sc = scripts:getScript("Enumerate scripts for document")
sc:setName("Enumerate scripts for current document")
```

7.126 Таблица DocumentAPI.Scripting

Таблица `DocumentAPI.Scripting` может быть получена путем вызова [DocumentAPI.createScripting\(\)](#) и содержит метод [runScript](#), который используется для запуска макрокоманды.

7.126.1 Метод Scripting:runScript

Метод предназначен для запуска макрокоманды, хранящейся в документе. В качестве аргумента передается имя макрокоманды.

Пример

```
scripting = DocumentAPI.createScripting(document)
scripting.runScript("Enumerate scripts for document")
```

7.127 Таблица DocumentAPI.ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 74. Используется в качестве поля `scriptPosition` таблицы [DocumentAPI.TextProperties](#).

Таблица 74 – Типы надстрочного и подстрочного форматирования

Наименование константы	Описание
<code>DocumentAPI.ScriptPosition_SuperScript</code>	Надстрочный знак (верхний индекс)
<code>DocumentAPI.ScriptPosition_SubScript</code>	Подстрочный знак (нижний индекс)
<code>DocumentAPI.ScriptPosition_NormalScript</code>	Без указания индекса

Пример

```
local props = DocumentAPI.TextProperties()
props.scriptPosition = DocumentAPI.ScriptPosition_SuperScript
range.setTextProperties(props)
```

7.128 Таблица DocumentAPI.Scripts

Таблица `DocumentAPI.Scripts` предоставляет доступ к списку макрокоманд документа. Коллекцию макрокоманд [DocumentAPI.Scripts](#) можно получить из документа посредством вызова метода `document:getScripts()`.

Пример

```
local scripts = document:getScripts()
for script in scripts:enumerate() do
```

```
print(script:getName())
print(script:getBody())
end
```

7.128.1 Метод Scripts:enumerate

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

Пример

```
for script in document:getScripts():enumerate() do
    print(script:getName())
end
```

7.128.2 Метод Scripts:getScript

Метод возвращает таблицу [DocumentAPI.Script](#), описывающую макрокоманду. В качестве аргумента используется имя макрокоманды.

Пример

```
local scripts = document:getScripts()
local script = scripts:getScript("Enumerate scripts for document")
print(script:getName())
```

7.128.3 Метод Scripts:removeScript

Метод удаляет макрокоманду из текущего документа. В качестве аргумента используется имя макрокоманды.

Пример

```
local scripts = document:getScripts()
scripts:removeScript("Enumerate scripts for document")
```

7.128.4 Метод Scripts:setScript

Метод добавляет макрокоманду в текущий документ. Если макрокоманда с таким именем уже существует, будет обновлено ее содержимое.

Пример

```
local scripts = document:getScripts()
local script_name = "Enumerate scripts for document"
local script_code = "local scripts = document:getScripts()\nfor script in\nscripts:enumerate() do\nprint(script:getName())\nend"
scripts:setScript(script_name, script_code)
```

7.129 Таблица DocumentAPI.Search

Таблица `DocumentAPI.Search` предоставляет доступ к механизму поиска фрагментов документа, открытого в редакторе текста или таблиц.

7.129.1 Метод Search:findText

Метод выполняет поиск строки без учета регистра во всем документе или выбранном диапазоне документа. Результат возвращается в виде диапазона [DocumentAPI.Range](#), содержащего искомый фрагмент.

Если строка не обнаружена, возвращается пустая таблица.

Возможно использование следующих вариантов метода:

```
Range findText(String text)
Range findText(String text, CaseSensitive caseSensitive)
Range findText(String text, Range range)
Range findText(String text, Range range, CaseSensitive caseSensitive)
Range findText(String text, CellRange cellRange)
Range findText(String text, CellRange cellRange, CaseSensitive caseSensitive)
Range findText(String text, Table tbl)
Range findText(String text, Table tbl, CaseSensitive caseSensitive)
```

Параметры

- `text` – строка для поиска;
- `caseSensitive` – поиск с учетом или без учета регистра, тип [DocumentAPI.CaseSensitive](#);
- `range` – диапазон, в котором будет производиться поиск, тип [DocumentAPI.Range](#);
- `cellRange` – диапазон ячеек, в котором будет производиться поиск, тип [DocumentAPI.CellRange](#);
- `table` – таблица, в которой будет производиться поиск, тип [DocumentAPI.Table](#).

Пример

```
search = DocumentAPI.createSearch(document)
-- Поиск по всему документу
ranges = search.findText("English")
for occurrence in ranges do
    print(occurrence.extractText())
end
```

Дополнительные примеры использования метода `Search:findText` приведены в разделе [Поиск в документе](#).

7.130 Таблица `DocumentAPI.Section`

Таблица `DocumentAPI.Section` представляет собой раздел в документе.

7.130.1 Метод `Section:getFooters`

Метод возвращает коллекцию [DocumentAPI.HeadersFooters](#) нижних колонтитулов данного раздела.

Пример

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

7.130.2 Метод `Section:getHeaders`

Метод возвращает коллекцию [DocumentAPI.HeadersFooters](#) верхних колонтитулов данного раздела.

Пример

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

7.130.3 Метод `Section:getPageOrientation`

Метод возвращает ориентацию страниц раздела.

Пример

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

7.130.4 Метод `Section:getPageProperties`

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

Пример

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

7.130.5 Метод `Section:getRange`

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

Пример

```
local sections = document:enumerateSections()  
for section in sections do  
    print(section:getRange():extractText())  
end
```

7.130.6 Метод `Section:setPageOrientation`

Метод задает ориентацию страниц раздела.

Пример

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

7.130.7 Метод `Section:setPageProperties`

Метод устанавливает параметры [DocumentAPI.PageProperties](#) страниц, находящихся в разделе.

Пример

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



```
properties.margins.left = 10
section:setPageProperties(properties)
```

7.131 Таблица DocumentAPI.Sections

Таблица DocumentAPI.Sections представляет интерфейс для доступа к коллекции секций документа. Может быть получена посредством вызова метода [document::getSections\(\)](#). Описание секции см. в разделе [DocumentAPI.Section](#).

7.131.1 Метод Sections:enumerate

Метод возвращает коллекцию секций документа.

Пример

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

7.132 Таблица DocumentAPI.Shape

Таблица Shape представляет собой фигуру, содержит методы для установки и получения ее свойств [DocumentAPI.ShapeProperties](#).

7.132.1 Метод Shape:getShapeProperties

Метод возвращает свойства фигуры [DocumentAPI.ShapeProperties](#).

Пример

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

7.132.2 Метод Shape:setShapeProperties

Метод устанавливает свойства фигуры [DocumentAPI.ShapeProperties](#).

Пример

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

```
shape_properties.verticalAlignment = DocumentAPI.VerticalAlignment_Center  
shape:setShapeProperties(shape_properties)
```

7.133 Таблица DocumentAPI.ShapeProperties

Таблица описывает свойства фигуры и содержит следующие поля:

- verticalAlignment - вертикальное выравнивание, тип [DocumentAPI.VerticalAlignment](#);
- borderProperties - свойства границ фигуры, тип [DocumentAPI.LineProperties](#);
- fill - свойства заполнения фигуры, тип [DocumentAPI.Fill](#);
- shapeTextLayout - свойства текста внутри фигуры, тип [DocumentAPI.ShapeTextLayout](#).

7.133.1 Поле ShapeProperties:borderProperties

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

7.133.2 Поле ShapeProperties:fill

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

7.133.3 Поле ShapeProperties:shapeTextLayout

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

7.133.4 Поле ShapeProperties:verticalAlignment

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

7.134 Таблица DocumentAPI.ShapeTextLayout

Таблица DocumentAPI.ShapeTextLayout описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 75. Используется в таблице [DocumentAPI.ShapeProperties](#).

Таблица 75 – Описание полей таблицы DocumentAPI.ShapeTextLayout

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

7.135 Таблица DocumentAPI.SizeU

Таблица `DocumentAPI.SizeU` представляет размер объекта в двухмерном пространстве. Описание полей таблицы `DocumentAPI.SizeU` представлено в таблице 76.

Таблица 76 – Описание полей таблицы `DocumentAPI.SizeU`

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

Пример

```
local size = DocumentAPI.SizeU(2, 3)
print("width=", size.width, ", height=", size.height)  --(width = 2,0, height = 3,0)
```

7.135.1 Метод SizeU.toString

Возвращает информацию о размерах в виде строкового значения формата (width: <value>, height: <value>).

Пример

```
local size = DocumentAPI.SizeU(2, 3)
print(size.toString())  --(width: 2.0, height: 3.0)
```

7.136 Таблица DocumentAPI.SortingConditions

Представляет собой коллекцию условий для сортировки строк. Таблица `SortingConditions` используется в методе [CellRange.sort\(\)](#).

Конструктор по умолчанию:

```
DocumentAPI.SortingConditions()
```

Конструктор копирования:

```
DocumentAPI.SortingConditions(sortingConditions)
```

Порядок условий в коллекции определяет порядок применения сортировки. Например, можно задать дополнительное условие для другого столбца, чтобы отсортировать строки с одинаковыми значениями в первом столбце.

Пример

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

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

range:sort(conditions)
```

7.136.1 Метод `SortingConditions:add`

Метод добавляет заданное условие сортировки в коллекцию.

Вызов

```
add(index, sortingDirection)
```

Параметры

- `index`: индекс столбца диапазона для применения сортировки.
- `sortingDirection`: порядок сортировки, тип [SortingDirection](#).

Пример

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

7.136.2 Метод `SortingConditions:clear`

Метод очищает коллекцию условий.

7.137 Таблица `DocumentAPI.SortingDirection`

В таблице 77 представлены варианты порядка сортировки строк. Используется в методе [SortingConditions:add\(\)](#).

Таблица 77 – Варианты порядка сортировки строк

Наименование константы	Описание
<code>SortingDirection_Ascending</code>	Сортировка по возрастанию

Наименование константы	Описание
SortingDirection_Descending	Сортировка по убыванию

Пример

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

7.138 Таблица DocumentAPI.Table

Таблица DocumentAPI.Table предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 48).

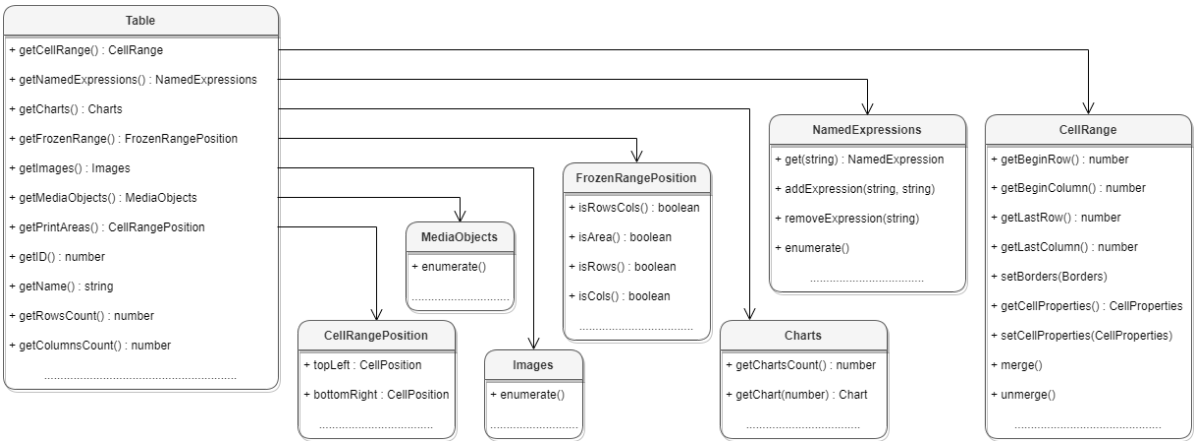


Рисунок 48 – Структура полей таблицы DocumentAPI.Table

7.138.1 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.

7.138.2 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.

7.138.3 Метод `Table:createFiltersRange`

Метод `Table:createFiltersRange` задает диапазон, который используется как диапазон фильтрации.

В качестве параметра используется диапазон ячеек типа [CellRangePosition](#). Метод возвращает [FiltersRange](#).

Разрешен только один диапазон фильтрации на таблицу. Это означает, что данный метод удаляет ранее определенный диапазон фильтрации. При этом есть исключение: если новый диапазон начинается с той же позиции, предыдущие фильтры будут сохранены.

Диапазон фильтрации должен включать дополнительную строку, которая используется как заголовок таблицы. Эта строка никогда не фильтруется.

Метод может быть использован только в табличных документах.

Пример

```
local sheet = document:getBlocks():getTable(0)
local cellRange = DocumentAPI.CellRangePosition(1, 1, 8, 2)
local filtersRange = sheet:createFiltersRange(cellRange)
```

Более подробный пример приведен в разделе [Работа с фильтрами](#).

7.138.4 Метод `Table:duplicate`

Для создания копии листа в табличном документе используется метод `duplicate`. Созданная копия листа размещается после копируемого листа. Метод может быть использован только в табличном документе.

Пример

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

7.138.5 Метод Table:find

Метод выполняет поиск ячеек, соответствующих заданному запросу.

Вызов

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

Параметры

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

Возвращает

- список ячеек, соответствующих поисковому запросу.

Пример

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

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

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

7.138.6 Метод Table:freeze

Метод freeze закрепляет заданную область [DocumentAPI.FrozenRangePosition](#) таблицы. Может быть использован только в табличном документе.

Пример

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local tbl = document:getBlocks():getTable(0)
tbl:freeze(frozenRangePosition)
print(tbl:getFrozenRange():isCols())
```

7.138.7 Метод Table:getCell

Метод позволяет получить доступ к отдельной ячейке таблицы. В качестве аргумента может выступать текстовое представление адреса ячейки, либо экземпляр таблицы

[DocumentApi.CellPosition](#).

Примеры

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
print(cell:getFormattedValue())
```

```
local cellPosition = DocumentAPI.CellPosition(2, 1)
local cell = tbl:getCell(cellPosition)
print(cell:getFormattedValue())
```

7.138.8 Метод Table:getCellRange

Метод позволяет получить доступ к диапазону ячеек таблицы [DocumentAPI.CellRange](#). В качестве аргумента может использоваться строка, описывающая диапазон ("A1:C4"), либо объект типа [DocumentAPI.CellRangePosition](#).

Примеры

```
local table = document:getBlocks():getTable(0)
local range = table:getCellRange("A1:C4")
for cell in range:enumerate() do
    print(cell:getFormattedValue())
end
```

```
local table = document:getBlocks():getTable(0)
local range = table:getCellRange(DocumentAPI.CellRangePosition(0, 0, 2, 2))
for cell in range:enumerate() do
    print(cell:getFormattedValue())
end
```

7.138.9 Метод Table:getCharts

Для получения списка диаграмм ([DocumentAPI.Charts](#)) таблицы используется метод Table:getCharts.

Пример

```
for tbl in document:getBlocks():enumerateTables() do
    print(tbl:getCharts():getChartsCount())
end
```


7.138.10 Метод `Table:getColumnsCount`

Метод позволяет получить количество столбцов таблицы.

Пример

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getColumnsCount())
```

7.138.11 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

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
width = tbl:getColumnWidth(1)
```

Задать ширину столбца таблицы позволяет метод [Table:setColumnWidth](#).

7.138.12 Метод `Table:getFiltersRange`

Метод `Table:getFiltersRange` возвращает текущий диапазон фильтрации, принадлежащий таблице. Рабочий лист табличного документа может содержать только один диапазон фильтрации.

Метод возвращает `FiltersRange`, если диапазон фильтрации существует.

Метод может быть использован только в табличных документах.

Пример

```
local filtersRange = sheet:getFiltersRange()
local cellRange = filtersRange:getCellRange()
print(cellRange:getBeginRow() .. ", " .. cellRange:getLastRow())
```

Более подробный пример приведен в разделе [Работа с фильтрами](#).

7.138.13 Метод Table:getFrozenRange

Существует возможность закрепления диапазона строк и столбцов. Такие диапазоны всегда остаются видимыми на экране в случае, когда пользователь осуществляет навигацию по таблице.

Метод `getFrozenRange` возвращает закрепленный диапазон

[DocumentAPI.FrozenRangePosition](#).

Пример

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local tbl = document:getBlocks():getTable(0)
tbl:freeze(frozenRangePosition)
print(tbl:getFrozenRange():isCols())
```

7.138.14 Метод Table:getImages

Для получения списка изображений ([DocumentAPI.Images](#)) таблицы используется метод `Table:getImages`.

Пример

```
tbl = document:getBlocks():getTable(0)
images = tbl:getImages()

for image in images:enumerate() do
    print(image)
end
```

7.138.15 Метод Table:getLastNonEmptyCellInColumn

Возвращает индекс последней заполненной ячейки в столбце. Последней заполненной считается ячейка, после которой все ячейки пустые.

Вызов

```
number getLastNonEmptyCellInColumn(columnIndex)
```

Параметры

— `columnIndex`: индекс столбца.

Возвращает

- индекс строки, которая содержит последнюю заполненную ячейку.
- `nil`, если все ячейки в столбце пустые.

Пример

```
local sheet = document:getBlocks():getTable(0)
print(sheet:getLastNonEmptyCellInColumn(0)) -- 10
print(sheet:getLastNonEmptyCellInRow(0)) -- 2
```

7.138.16 Метод Table:getLastNonEmptyCellInRow

Возвращает индекс последней заполненной ячейки в строке. Последней заполненной считается ячейка, после которой все ячейки пустые.

Вызов

```
number getLastNonEmptyCellInRow(rowIndex)
```

Параметры

– rowIndex: индекс строки.

Возвращает

- индекс столбца, который содержит последнюю заполненную ячейку.
- nil, если все ячейки в строке пустые.

Пример

```
local sheet = document:getBlocks():getTable(0)
print(sheet:getLastNonEmptyCellInColumn(0)) -- 10
print(sheet:getLastNonEmptyCellInRow(0)) -- 2
```

7.138.17 Метод Table:getMediaObjects

Для получения списка медиаобъектов ([DocumentAPI.MediaObjects](#)) таблицы используется метод Table:getMediaObjects.

Пример

```
tbl = document:getBlocks():getTable(0)
mediaObjects = tbl:getMediaObjects()

for mediaObject in mediaObjects:enumerate() do
    print(mediaObject)
end
```

7.138.18 Метод Table:getName

Метод позволяет получить наименование листа табличного документа.

Пример

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getName())
```

7.138.19 Метод Table:getNameExpressions

Метод используется для получения списка именованных диапазонов [DocumentAPI:NamedExpressions](#).

7.138.20 Метод Table:getPrintAreas

Метод Table:getPrintAreas возвращает текущие области печати - вектор элементов [DocumentAPI.CellRangePosition](#). См. [описание](#) методов вектора.

Пример

```
tbl = document:getBlocks():getTable(0)
tbl:setPrintArea(DocumentAPI.CellRangePosition(0, 0, 5, 5))

printAreas = tbl:getPrintAreas()
print(printAreas[0]:toString())
```

7.138.21 Метод Table:getProtectionProperties

Метод возвращает параметры защиты от изменений листа табличного документа.

Вызов

```
TableProtectionProperties getProtectionProperties()
```

Возвращает

– [TableProtectionProperties](#): свойства защиты листа документа (nil, если защита листа не установлена).

Используйте методы [setProtection\(\)](#) и [removeProtection\(\)](#), чтобы установить и снять защиту от изменений листа. Вы также можете использовать метод [isProtected\(\)](#), чтобы узнать, установлена ли защита на лист.

7.138.22 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

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
height = tbl:getRowHeight(1)
```

Задать высоту строки таблицы позволяет метод [Table:setRowHeight](#).

7.138.23 Метод Table:getRowCount

Метод позволяет получить количество строк таблицы.

Пример

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getRowCount())
```

7.138.24 Метод Table:getShowZeroValue

Для проверки режима отображения нулевых значений ячеек используется метод `getShowZeroValue`.

Пример

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(false)
print(tbl:getShowZeroValue())
```

7.138.25 Метод Table:getUsedRange

Метод возвращает диапазон ячеек, используемый в текущем листе табличного документа. К используемым относятся ячейки, которые содержат:

- данные;
- заметки;
- форматирование;
- объединенные ячейки;
- фильтры;
- сводные таблицы;
- умные таблицы.

Вызов

```
CellRange getUsedRange()
```

Возвращает

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

Пример

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

7.138.26 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.

7.138.27 Метод Table:groupRows

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

Вызов

```
groupRows(rowIndex, rowsCount)
```

Параметры

- rowIndex – индекс строки, начиная с которого будет начата группировка строк;
- rowsCount – количество строк для группировки.

7.138.28 Метод Table:insertColumnAfter

Метод предназначен для вставки нового столбца после указанной позиции в таблице.

Вызов

```
insertColumnAfter( columnIndex, copyColumnStyle, columnsCount )
```

Параметры

- columnIndex – индекс столбца в таблице, после которого производится вставка. Индексация столбцов начинается с нуля.

- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
-- Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
tbl:insertColumnAfter(0, false, 2)
```

7.138.29 Метод `Table:insertColumnBefore`

Метод предназначен для вставки нового столбца до указанной позиции в таблице.

Вызов

```
insertColumnBefore( columnIndex, copyColumnStyle, columnsCount )
```

Параметры

- `columnIndex` – индекс столбца в таблице, перед которым производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
-- Добавление двух столбцов в середину таблицы, без наследования настроек форматирования
tbl:insertColumnBefore(1, false, 2)
```

7.138.30 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
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl.insertRowAfter(0, false, 2)
```

7.138.31 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
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl:insertRowBefore(1, false, 2)
```

7.138.32 Метод Table:isColumnVisible

Метод `Table::isColumnVisible` позволяет определять видимость столбца по заданному индексу. Индексация столбцов начинается с нуля. Метод возвращает `true` если столбец отображается.

Для задания видимости столбцов таблицы применяется метод [Table::setColumnsVisible](#).

Вызов

```
isColumnVisible(columnIndex)
```

Параметр

`columnIndex` – индекс столбца.

Пример

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

Дополнительный пример использования метода `Table::isColumnVisible` приведен в разделе [Управление видимостью строк / колонок](#).

7.138.33 Метод Table:isProtected

Метод возвращает состояние защиты от изменений листа табличного документа.

Вызов

```
bool isProtected()
```

Используйте методы [setProtection\(\)](#) и [removeProtection\(\)](#), чтобы установить и снять защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод [getProtectionProperties\(\)](#).

7.138.34 Метод Table:isRowVisible

Метод `Table::isRowVisible` позволяет определять видимость строки по заданному индексу. Индексация строк начинается с нуля. Метод возвращает `true` если строка отображается.

Для задания видимости строк таблицы применяется метод [Table::setRowsVisible](#).

Вызов

```
isRowVisible(rowIndex)
```

Параметр

`rowIndex` – индекс строки.

Пример

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

Дополнительный пример использования метода `Table::isRowVisible` приведен в разделе [Управление видимостью строк / колонок](#).

7.138.35 Метод Table:isVisible

Метод возвращает значение `true`, если лист таблицы в табличном документе отображается в редакторе таблиц.

Пример

```
local tbl = document:getBlocks():getTable(0)
if not tbl:isVisible() then
    tbl:setVisible(true)
end
```

7.138.36 Метод Table:moveTo

Для перемещения листа таблицы по указанному индексу в табличном документе используется метод `moveTo`. Указанный индекс должен быть меньше или равен количеству листов в документе. Индексация листов начинается с нуля. Метод может быть использован только в табличном документе.

Пример

```
-- В табличном документе два листа с индексами 0 и 1.
-- Поменяем их местами.
```

```
local tbl = document:getBlocks():getTable(0)
tbl:moveTo(1)
```

7.138.37 Метод Table:remove

Для удаления таблицы в текстовом документе или листа в табличном документе используется метод `remove()`.

Пример

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

7.138.38 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.

7.138.39 Метод Table:removeProtection

Метод снимает защиту от изменений с листа табличного документа.

Вызов

```
removeProtection(password)
```

Параметры

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

Пример

```
local firstSheet = document:getBlocks():getTable(0)
firstSheet:removeProtection("password")
```

Используйте метод [setProtection\(\)](#), чтобы установить защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод [getProtectionProperties\(\)](#). Вы также можете использовать метод [isProtected\(\)](#), чтобы узнать, установлена ли защита на лист.

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

7.138.40 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.

7.138.41 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.

7.138.42 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.

7.138.43 Метод `Table:setColumnsVisible`

Метод `Table::setColumnsVisible` позволяет задавать видимость столбцов, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Метод предназначен для использования только в табличном редакторе.

Вызов

```
setColumnsVisible(first, columnsCount, visible)
```

Параметры

first – начальный индекс;

columnsCount – количество столбцов;

visible – видимость.

Пример использования в табличном редакторе

```
local tbl = document:getBlocks():getTable(0)
tbl:setColumnsVisible(0, 2, false)
```

Дополнительный пример использования метода `Table::setColumnsVisible` приведен в разделе [Управление видимостью строк / колонок](#).

7.138.44 Метод `Table:setColumnWidth`

Метод устанавливает ширину столбца таблицы в пунктах (1/72 дюйма).

Вызов

```
setColumnWidth(columnIndex, width)
```

Параметры

- columnIndex – индекс столбца в таблице, для которого устанавливается значение ширины. Индексация столбцов начинается с нуля.
- width – ширина столбца в пунктах (1/72 дюйма).

Пример

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Установить ширину столбца в 400 pt
tbl:setColumnWidth(1, 400)
```

Метод [Table:getColumnWidth](#) позволяет получить ширину столбца таблицы.

7.138.45 Метод `Table:setName`

Метод задает имя таблицы. В случае с табличным документом это имя будет являться заголовком листа документа. Данное значение должно быть уникальным, т.к. может использоваться для ссылки на таблицу, например, из формул.

Пример

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

Для текстовых документов использование данного метода также допустимо, наименование таблицы нигде не отображается, но в дальнейшем его можно использовать для доступа к таблице по имени.

Пример

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

7.138.46 Метод Table:setPrintArea

Метод служит для установки и сброса области печати [DocumentAPI.CellRangePosition](#).

Пример

```
local tbl = document:getBlocks():getTable(0)
tbl:setPrintArea(DocumentAPI.CellRangePosition(0, 0, 5, 5)) -- установить область
печати размером в пять строк и пять колонок, начиная с левого верхнего угла
таблицы
```

7.138.47 Метод Table:setPrintAreas

Метод Table:setPrintAreas задает множественные области печати или экспорта CellRangePositions, где CellRangePositions - вектор из элементов [CellRangePosition](#) (см. [описание](#) вектора).

Пример

```
tbl = document:getBlocks():getTable(0)
ranges = DocumentAPI.CellRangePositions()
ranges:push_back(DocumentAPI.CellRangePosition(0, 0, 5, 5))
ranges:push_back(DocumentAPI.CellRangePosition(1, 2, 5, 5))
tbl:setPrintAreas(ranges)

printAreas = tbl:getPrintAreas()
print(printAreas[0]:toString(), printAreas[1]:toString())
```

7.138.48 Метод Table:setProtection

Метод устанавливает защиту от изменений на лист табличного документа.

Вызов

```
setProtection(tableProtectionProperties, password)
```

Параметры

- tableProtectionProperties: параметры защиты листа, тип [TableProtectionProperties](#);
- password: (необязательный) пароль для установки защиты, тип string.

Пример

```
local tableProps = DocumentAPI.TableProtectionProperties()  
tableProps.deleteColumns = false  
tableProps.deleteRows = false  
tableProps.filterData = true  
tableProps.formatCells = true  
tableProps.formatColumns = true  
tableProps.formatRows = true  
tableProps.insertAndEditObjects = false  
tableProps.insertAndEditPivotTables = false  
tableProps.insertColumns = false  
tableProps.insertLinks = true  
tableProps.insertRows = false  
tableProps.selectProtectedCells = true  
tableProps.sortData = true  
  
local firstSheet = document:getBlocks():getTable(0)  
firstSheet:setProtection(tableProps, "password")
```

Используйте метод [removeProtection\(\)](#), чтобы снять защиту от изменений листа.

Получить текущие параметры защиты листа позволяет метод [getProtectionProperties\(\)](#). Вы также можете использовать метод [isProtected\(\)](#), чтобы узнать, установлена ли защита на лист.

Метод setProtectionProperties() объектов [Cell](#) и [CellRange](#) позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой защиты).

Если метод setProtection() применяется к уже защищенному листу, возникает исключение SpreadsheetProtectionError.

7.138.49 Метод `Table:setRowHeight`

Метод устанавливает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов

```
setRowHeight(rowIndex, height, heightRule)
```

Параметры

- `rowIndex` – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- `height` – высота строки в пунктах (1/72 дюйма).
- `heightRule` – точность значения (`DocumentAPI.RowHeightRule_Exact` – точно, `DocumentAPI.RowHeightRule_AtLeast` – не меньше).

Пример

```
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Установить высоту строки в 100 pt
tbl:setRowHeight(1, 100, DocumentAPI.RowHeightRule_Exact)
```

Метод [Table:getRowHeight](#) позволяет получить высоту строки таблицы.

7.138.50 Метод `Table:setRowsVisible`

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

Метод предназначен для использования только в табличном редакторе.

Вызов

```
setRowsVisible(first, rowsCount, visible)
```

Параметры

- `first` – начальный индекс;
- `columnsCount` – количество строк;
- `visible` – видимость.

Пример использования в табличном редакторе

```
local tbl = document:getBlocks():getTable(0)
tbl:setRowsVisible(0, 2, false)
```


Дополнительный пример использования метода `Table::setRowsVisible` приведен в разделе [Управление видимостью строк / колонок](#).

7.138.51 Метод `Table:setShowZeroValue`

Для упрощения чтения таблицы нулевые значения ячеек могут быть скрыты. Для управления скрыванием/показом ячеек используется метод `setShowZeroValue`. Метод может быть использован только в табличном документе.

Пример

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(true)
```

7.138.52 Метод `Table:setVisible`

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

Вызов

```
setVisible( visible )
```

Параметр

`visible` – параметр, задающий видимость листа. Если значение параметра `visible` равно `true`, то лист таблицы отображается в редакторе таблиц.

Пример

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

7.138.53 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.

7.138.54 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.

7.138.55 Метод Table:__eq

Метод используется для определения эквивалентности двух таблиц.

Пример

```
local tbl1 = document:getBlocks():getTable(0)
local tbl2 = document:getBlocks():getTable("Table Name")
if tbl1:__eq(tbl2) then
    print("tbl1 и tbl2 ссылаются на общую таблицу в документе")
end
```

7.139 Таблица DocumentAPI.TableFilters

TableFilters - это таблица, которая хранит фильтры столбцов. Фильтры можно применять к диапазону фильтрации [FiltersRange](#). При применении фильтров, соответствующие строки рабочего листа будут скрыты.

Конструктор по умолчанию:

```
local firstTableFilters = DocumentAPI.TableFilters()
```

Конструктор копирования:

```
local secondTableFilters = DocumentAPI.TableFilters(firstTableFilters)
```

Пример использования приведен в разделе [Работа с фильтрами](#).

7.139.1 Метод TableFilters:clear

Метод `TableFilters:clear` удаляет все фильтры столбцов, которые были сохранены ранее.

Пример

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

7.139.2 Метод TableFilters:erase

Метод TableFilters:erase удаляет фильтр из заданного столбца. В качестве параметра используется индекс столбца.

Пример

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

7.139.3 Метод TableFilters:setFilter

Метод TableFilters:setFilter устанавливает фильтр для конкретного столбца. В качестве параметров используются индекс столбца, а также используемый фильтр: [ValuesTableFilter](#) или [ConditionalTableFilter](#).

Пример

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

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

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

Более подробный пример приведен в разделе [Работа с фильтрами](#).

7.140 Таблица DocumentAPI.TableProtectionProperties

Таблица DocumentAPI.TableProtectionProperties предназначена для настройки параметров защиты листа в табличном документе (аналог раздела «Разрешенные действия» в меню «Управление защитой»). Данная таблица используется в методах [Table:setProtection\(\)](#) и [Table:getProtectionProperties\(\)](#).

Таблица 78 – Описание полей таблицы DocumentAPI.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

```
local tableProps = DocumentAPI.TableProtectionProperties()
tableProps.deleteColumns = false
tableProps.deleteRows = false
tableProps.filterData = true
tableProps.formatCells = true
tableProps.formatColumns = true
tableProps.formatRows = true
tableProps.insertAndEditObjects = false
tableProps.insertAndEditPivotTables = false
tableProps.insertColumns = false
tableProps.insertLinks = true
tableProps.insertRows = false
tableProps.selectProtectedCells = true
tableProps.sortData = true

local firstSheet = document:getBlocks():getTable(0)
firstSheet:setProtection(tableProps, "password")
```

7.141 Таблица DocumentAPI.TableRangeInfo

Таблица DocumentAPI.TableRangeInfo описывает диапазон ячеек таблицы.

Описание полей таблицы DocumentAPI.TableRangeInfo представлено в таблице 79.

Таблица 79 – Поля таблицы DocumentAPI.TableRangeInfo

Поле	Тип	Описание
tableRange	DocumentAPI.CellRangePosition	Диапазон ячеек

Пример

```
local tbl = document:getBlocks():getTable(0)
local charts = tbl:getCharts()
local rangeInfo = charts:getChart(0):getRange(0)
local tableRangeInfo = rangeInfo.tableRangeInfo
local tableRange = tableRangeInfo.tableRange
print("topLeft=", tableRange.topLeft.row, tableRange.topLeft.column)
print("topLeft=", tableRange.bottomRight.row, tableRange.bottomRight.column)
```

7.142 Таблица DocumentAPI.TableSearchSettings

Таблица TableSearchSettings предназначена для настройки параметров поиска ячеек. Данная таблица используется в методах [Table:find\(\)](#) и [CellRange:find\(\)](#).

Таблица 80 – Описание полей таблицы TableSearchSettings

Поле	Описание
TableSearchSettings.caseSensitive	Позволяет учитывать регистр при поиске, тип CaseSensitive
TableSearchSettings.matchBehaviour	Задаёт алгоритм сравнения запроса и значения, тип TableSearchSettings.MatchBehaviour
TableSearchSettings.searchIn	Позволяет выбрать значения, по которым производится поиск, тип TableSearchSettings.SearchProperty
TableSearchSettings.wholeWords	Разбивает текст в ячейке на слова, которые считаются отдельными значениями для сравнения, тип bool

Пример

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

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

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

7.142.1 Таблица TableSearchSettings.MatchBehaviour

Таблица MatchBehaviour определяет алгоритм сравнения запроса и значения. Используется в поле [TableSearchSettings.matchBehaviour](#).

Таблица 81 – Описание алгоритмов сравнения

Наименование константы	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

7.142.2 Таблица TableSearchSettings.SearchProperty

Таблица SearchProperty определяет значения, по которым производится поиск в таблице. Используется в поле [TableSearchSettings.searchIn](#).

Таблица 82 – Описание вариантов поиска в таблице

Наименование константы	Описание
TableSearchSettings.SearchProperty_Value	Поиск по значениям ячеек
TableSearchSettings.SearchProperty_Formula	Поиск по тексту формул в ячейках
TableSearchSettings.SearchProperty_Notes	Поиск по тексту заметок

7.143 Таблица DocumentAPI.TextAnchoredPosition

Таблица DocumentAPI.TextAnchoredPosition (см. Рисунок 49) представляет позицию объекта на странице текстового документа. Пример использования см. в разделе [InlineFrame.setPosition\(\)](#).

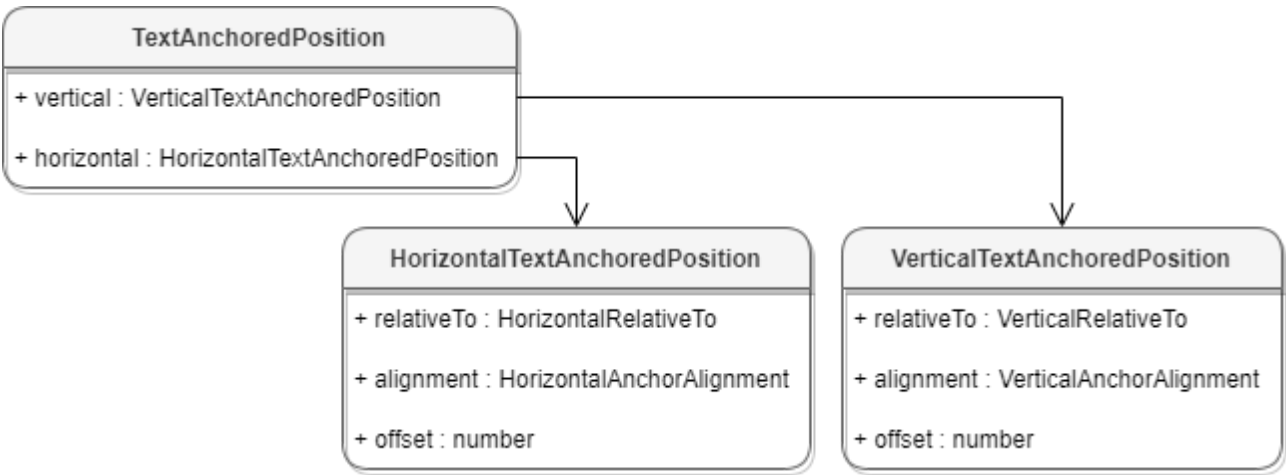


Рисунок 49 – Поля таблицы DocumentAPI.TextAnchoredPosition

Описание полей таблицы представлено в таблице 83.

Таблица 83 – Описание полей таблицы DocumentAPI.TextAnchoredPosition

Поле	Описание
DocumentAPI.TextAnchoredPosition.horizontal	Позиция по горизонтали HorizontalTexAnchoredPosition
DocumentAPI.TextAnchoredPosition.vertical	Позиция по вертикали VerticalTextAnchoredPosition

7.143.1 Метод `TextAnchoredPosition: __eq`

Метод используется для определения эквивалентности значений двух позиций объектов.

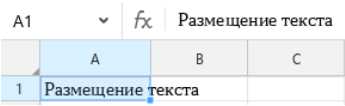
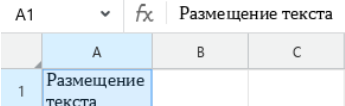
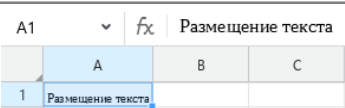
Пример

```
local pos1 = DocumentAPI.TextAnchoredPosition()  
pos1.horizontal =  
DocumentAPI.HorizontalTextAnchoredPosition  
(DocumentAPI.HorizontalRelativeTo_Column)  
pos1.horizontal.offset = 1  
  
local pos2 = DocumentAPI.TextAnchoredPosition()  
pos2.horizontal =  
DocumentAPI.HorizontalTextAnchoredPosition  
(DocumentAPI.HorizontalRelativeTo_Column)  
pos2.horizontal.offset = 1  
  
print(pos1: __eq(pos2))
```

7.144 Таблица `DocumentAPI.TextLayout`

В таблице 84 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле `textLayout` таблицы [CellProperties](#).

Таблица 84 – Варианты размещения текста в ячейках таблицы

Наименование константы	Описание	Отображение
<code>DocumentAPI.TextLayout_SingleLine</code>	Текст располагается в одну строку с наложением на соседние ячейки.	
<code>DocumentAPI.TextLayout_WrapByWords</code>	Текст внутри ячейки переносится по словам. Высота ряда увеличивается чтобы разместить текст полностью.	
<code>DocumentAPI.TextLayout_ShrinkSizeToFitWidth</code>	Текст располагается в одну линию, отображение масштабируется таким образом, чтобы полностью разместиться в ячейке без изменения ее размера. Размер шрифта не изменяется, данная настройка влияет только на отображение содержимого ячейки таблицы.	

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell_A1 = tbl:getCell("A1")
local props = cell_A1:getCellProperties()
props.textLayout = DocumentAPI.TextLayout_ShrinkSizeToFitWidth
cell_A1:setCellProperties(props)
```

7.145 Таблица DocumentAPI.TextOrientation

Таблица DocumentAPI.TextOrientation предоставляет доступ к свойствам ориентации текста в ячейке, фигуре и т. д (см. [DocumentAPI.CellProperties](#)).

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("D2") --(DocumentAPI.CellPosition(3,1))
local props = cell:getCellProperties()
props.textOrientation = DocumentAPI.TextOrientation(45)
cell:setCellProperties(props)
print(props.textOrientation:getAngle())
```

7.145.1 Метод TextOrientation:getAngle

Возвращает угол ориентации текста в ячейке. Значение угла указывается в градусах.

Пример

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

7.145.2 Метод TextOrientation:isStackedChars

Возвращает True в случае, если ориентация текста представляет собой вертикальный столбец.

Пример

```
local cellProperties = cell:getCellProperties()
print(cellProperties.textOrientation:isStackedChars())
```

7.145.3 Метод TextOrientation:__eq

Метод используется для определения эквивалентности значений двух объектов

TextOrientation.

Пример

```
print(DocumentAPI.TextOrientation(45):__eq(DocumentAPI.TextOrientation(45)))
```

7.146 Таблица DocumentAPI.TextProperties

Таблица DocumentAPI.TextProperties содержит поля, задающие параметры текста. На рисунке 50 изображена объектная модель таблицы DocumentAPI.TextProperties.

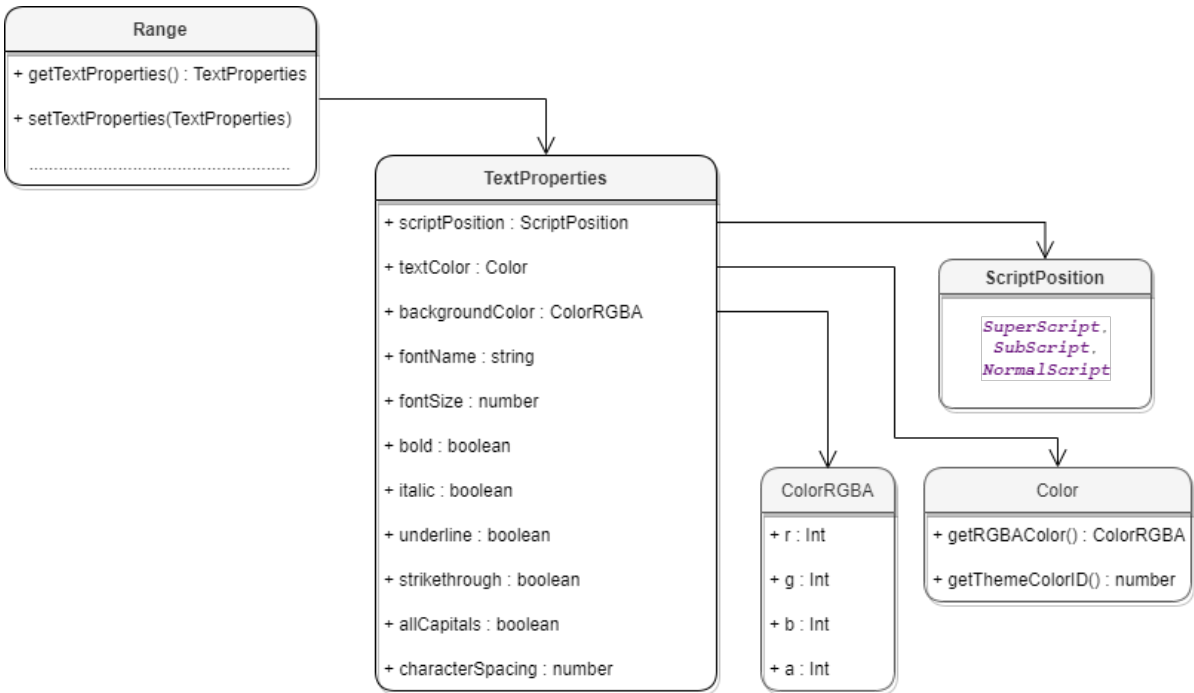


Рисунок 50 – Объектная модель для работы с таблицей
DocumentAPI.TextProperties

Описание полей таблицы DocumentAPI.TextProperties представлено в таблице 85. Свойства DocumentAPI.TextProperties применяются к диапазону текста DocumentAPI.Range (методы [Range.getTextProperties\(\)](#), [Range.setTextProperties\(\)](#)), ячейке DocumentAPI.Cell (методы [Cell.getTextProperties\(\)](#), [Cell.setTextProperties\(\)](#)) и диапазону ячеек DocumentAPI.CellRange (методы [CellRange.getTextProperties\(\)](#), [CellRange.setTextProperties\(\)](#)).

Таблица 85 – Описание полей таблицы DocumentAPI.TextProperties

Поле	Тип	Описание
TextProperties.fontName	Строковое	Наименование шрифта, использованного для оформления фрагмента документа.
TextProperties.fontSize	Числовое	Размер шрифта, использованного для оформления фрагмента документа.
TextProperties.bold	Логическое	Значение true устанавливает жирное начертание для указанного фрагмента текста.
TextProperties.italic	Логическое	Значение true устанавливает начертание курсивом для указанного фрагмента текста.
TextProperties.underline	Логическое	Значение true устанавливает подчеркивание для указанного фрагмента текста.
TextProperties.strikethrough	Логическое	Значение true устанавливает начертание «зачеркнутый» для указанного фрагмента текста.
TextProperties.allCapitals	Логическое	Значение true устанавливает все буквы указанного фрагмента текста как прописные. Значение false устанавливает все буквы указанного фрагмента текста как строчные.
TextProperties.scriptPosition	ScriptPosition	Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме.
TextProperties.textColor	Color	Цвет указанного фрагмента документа.
TextProperties.backgroundColor	ColorRGBA	Цвет фона указанного фрагмента документа.
TextProperties.characterSpacing	Числовое	Размер межсимвольного интервала.

Пример

```
local props = DocumentAPI.TextProperties()
props.fontName = "XO Oriel"
```

```
props.fontSize = 20

-- текст третьего абзаца
local range = document:getBlocks():getParagraph(2):getRange()

-- установить свойства фрагмента текста
range.setTextProperties(props)
```

7.147 Таблица DocumentAPI.TextUnit

В таблице 86 представлены варианты разделения текста на диапазоны. Используется в методах [Position:getCurrentRange\(\)](#), [Position:getNextRange\(\)](#) и [Position:getPreviousRange\(\)](#).

Таблица 86 – Варианты разделения текста на диапазоны

Наименование константы	Описание
DocumentAPI.TextUnit_Character	Разделение по символам
DocumentAPI.TextUnit_Word	Разделение по словам (символы-разделители считаются отдельными словами)
DocumentAPI.TextUnit_Sentence	Разделение по предложениям (разделители после предложения считаются его частью)
DocumentAPI.TextUnit_Paragraph	Разделение по параграфам

Пример

```
local secondSentence =
document:getRange():getBegin():getNextRange(DocumentAPI.TextUnit_Sentence)
local firstSentenceLastWord =
secondSentence:getBegin():getPreviousRange(DocumentAPI.TextUnit_Word)
while
firstSentenceLastWord:getContentEnd():compare(firstSentenceLastWord:getBegin())
== 1 do
    firstSentenceLastWord =
firstSentenceLastWord:getBegin():getPreviousRange(DocumentAPI.TextUnit_Word)
end
local thirdSentenceFirstLetter =
secondSentence:getContentEnd():getCurrentRange(DocumentAPI.TextUnit_Character)
```

```
print(secondSentence:extractText())
-- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
print(firstSentenceLastWord:extractText())
-- aliqua
print(thirdSentenceFirstLetter:extractText())
-- D
```

7.148 Таблица DocumentAPI.TextWrapType

В таблице 87 представлены варианты обтекания текстом встроенного объекта. Используется в [InlineFrame.setWrapType\(\)](#).

Таблица 87 – Варианты обтекания текстом встроенного объекта

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

7.149 Таблица DocumentAPI.ThemeColorID

В таблице 88 представлены типы идентификаторов цветов тем. Используется в [DocumentAPI.Color](#).

Таблица 88 – Типы идентификаторов цветов тем

Наименование константы	Описание
DocumentAPI.ThemeColorID_Background1	Фон1
DocumentAPI.ThemeColorID_Text1	Текст1
DocumentAPI.ThemeColorID_Background2	Фон2
DocumentAPI.ThemeColorID_Text2	Текст2
DocumentAPI.ThemeColorID_Dark1	Темная1

Наименование константы	Описание
DocumentAPI.ThemeColorID_Dark2	Темная2
DocumentAPI.ThemeColorID_Light1	Светлая1
DocumentAPI.ThemeColorID_Light2	Светлая2
DocumentAPI.ThemeColorID_Accent1	Акцент1
DocumentAPI.ThemeColorID_Accent2	Акцент2
DocumentAPI.ThemeColorID_Accent3	Акцент3
DocumentAPI.ThemeColorID_Accent4	Акцент4
DocumentAPI.ThemeColorID_Accent5	Акцент5
DocumentAPI.ThemeColorID_Accent6	Акцент6
DocumentAPI.ThemeColorID_Hyperlink	Гиперссылка
DocumentAPI.ThemeColorID_FollowedHyperlink	Следующая гиперссылка

7.150 Таблица DocumentAPI.TimePatterns

Форматы времени представлены в таблице 89. Пример использования см. в главе [DocumentAPI.DateTimeCellFormatting](#).

Таблица 89 – Форматы времени

Наименование константы	Описание
DocumentAPI.TimePatterns_ShortTime	'hh:mm AM/PM' для языка en_US
DocumentAPI.TimePatterns_LongTime	'hh:mm:ss AM/PM' для языка en_US

7.151 Таблица DocumentAPI.TrackedChange

Таблица DocumentAPI.TrackedChange представляет отслеживаемое изменение в диапазоне текстового документа (см. Рисунок 51).

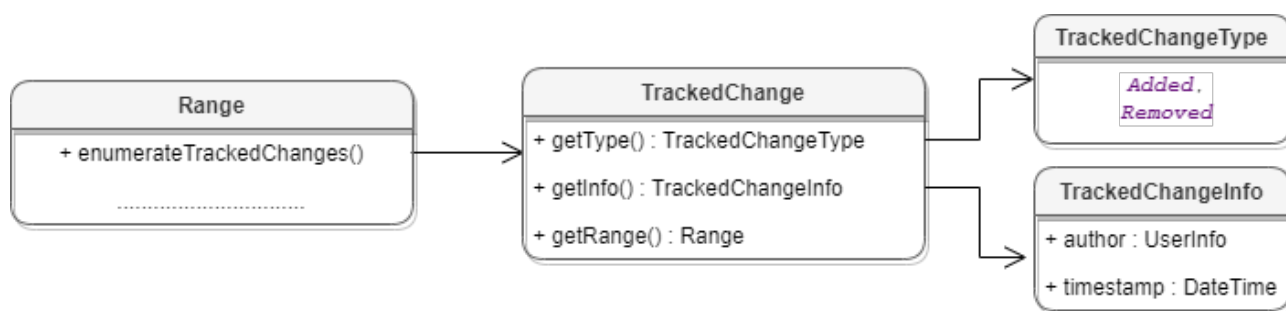


Рисунок 51 – Объектная модель таблиц для работы с отслеживаемыми изменениями

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

Пример

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
    print(change:getRange():extractText())
end
```

7.151.1 Метод TrackedChange:getInfo

Метод позволяет получить информацию об отслеживаемых изменениях ([DocumentAPI.TrackedChangeInfo](#)).

Пример

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
    print(tracked_change:getInfo().author.name)
end
```

7.151.2 Метод TrackedChange:getRange

Метод возвращает объект [DocumentAPI.Range](#), который соответствует измененному диапазону внутри абзаца.

Пример

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
    print(tracked_change:getRange():extractText())
end
```

7.151.3 Метод `TrackedChange:getType`

Метод позволяет получить информацию о типе отслеживаемого изменения [DocumentAPI.TrackedChangeType](#).

Пример

```
local tracked_changes = document:getRange():enumerateTrackedChanges()  
for tracked_change in tracked_changes do  
    print(tracked_change:getType())  
end
```

7.152 Таблица `DocumentAPI.TrackedChangeInfo`

Таблица `DocumentAPI.TrackedChangeInfo` содержит информацию об отслеживаемых изменениях. Описание полей таблицы представлено в таблице 90.

Таблица 90 – Описание полей таблицы `DocumentAPI.TrackedChangeInfo`

Поле	Тип	Описание
<code>DocumentAPI.TrackedChangeInfo.author</code>	<code>UserInfo</code>	Автор изменений
<code>DocumentAPI.TrackedChangeInfo.timeStamp</code>	DateTime	Дата и время изменений

Пример

```
local changesList = document:getRange():enumerateTrackedChanges()  
for change in changesList do  
    local trackedChangeInfo = change:getInfo()  
    local author = trackedChangeInfo.author  
    local ts = trackedChangeInfo.timeStamp  
    local ts_msg = string.format("%d/%d/%d - %d:%d:%d", ts.day, ts.month, ts.year,  
ts.hour, ts.minute, ts.second)  
    print(author.name, ts_msg)  
end
```

7.152.1 Метод `TrackedChangeInfo:__eq`

Метод используется для определения эквивалентности двух отслеживаемых изменений.

7.153 Таблица DocumentAPI.TrackedChangeType

Типы отслеживаемых изменений представлены в таблице 91.

Таблица 91 – Типы отслеживаемых изменений

Наименование константы	Описание
DocumentAPI.TrackedChangeType_Added	Добавленные изменения
DocumentAPI.TrackedChangeType_Removed	Удаленные изменения

Пример

```
local changesList = document:getRange():enumerateTrackedChanges()  
for change in changesList do  
    if DocumentAPI.TrackedChangeType_Added == change:getType() then action =  
"Добавлено: " else action = "Удалено: " end  
    print(action)  
end
```

7.154 Таблица DocumentAPI.ValueFieldsOrientation

Таблица DocumentAPI.ValueFieldsOrientation описывает варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений. Является полем таблицы [DocumentAPI.PivotTableLayoutSettings](#). Описание полей таблицы представлено в таблице 92.

Таблица 92 – Описание полей таблицы DocumentAPI.ValueFieldsOrientation

Поле	Описание
DocumentAPI.ValueFieldsOrientation_ByRows	По строкам
DocumentAPI.ValueFieldsOrientation_ByColumns	По столбцам

7.155 Таблица DocumentAPI.ValuesTableFilter

Таблица ValuesTableFilter реализует фильтр, содержащий значения, которые должны быть показаны в диапазоне фильтрации.

Конструктор по умолчанию:

```
local firstFilter = DocumentAPI.ValuesTableFilter()
```

Конструктор копирования:

```
local secondFilter = DocumentAPI.ValuesTableFilter(firstFilter)
```

Пример использования приведен в разделе [Работа с фильтрами](#).

7.155.1 Метод `ValuesTableFilter::add`

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

Пример

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

7.155.2 Метод `ValuesTableFilter::clear`

Метод `ValuesTableFilter::clear` удаляет все элементы фильтра.

Пример

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

7.156 Таблица `DocumentAPI.VectorString`

Таблица `DocumentAPI.VectorString` предназначена для реализации массива строк.

Пример

```
vector = DocumentAPI.VectorString(3)  
vector[0] = "1"  
vector[1] = "2"  
vector[2] = "3"  
print(vector:size()) -- 3
```

7.156.1 Метод `VectorString::back`

Метод возвращает последний элемент вектора.

Пример

```
local vector = DocumentAPI.VectorString()  
vector:push_back("12")
```

```
vector:push_back("13")  
print(vector:front()) -- 13
```

7.156.2 Метод VectorString:clear

Метод очищает содержимое вектора.

Пример

```
local vector = DocumentAPI.VectorString()  
vector:push_back("12")  
vector:clear()  
print(vector:empty()) -- true
```

7.156.3 Метод VectorString:empty

Метод возвращает true, если вектор не содержит элементов.

Пример

```
local vector = DocumentAPI.VectorString()  
vector:push_back("12")  
print(vector:empty()) -- false
```

7.156.4 Метод VectorString:front

Метод возвращает первый элемент вектора.

Пример

```
local vector = DocumentAPI.VectorString()  
vector:push_back("12")  
vector:push_back("13")  
print(vector:front()) -- 12
```

7.156.5 Метод VectorString:max_size

Метод возвращает максимальный размер вектора.

Пример

```
local vector = DocumentAPI.VectorString()  
print(vector:max_size())
```

7.156.6 Метод VectorString:pop_back

Метод удаляет последний элемент вектора.

Пример

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:pop_back()
print(vector:size()) -- 0
```

7.156.7 Метод VectorString:push_back

Метод добавляет элемент в конец вектора.

Пример

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
print(vector:size()) -- 1
```

7.156.8 Метод VectorString:size

Метод возвращает размер вектора.

Пример

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:push_back("13")
vector:push_back("14")
print(vector:size()) -- 3
```

7.156.9 Метод VectorString:__getitem

Метод возвращает элемент вектора по заданному индексу.

Пример

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:push_back("13")
print(vector:__getitem(0)) -- 12
print(vector:__getitem(1)) -- 13
```

7.156.10 Метод `VectorString::__setitem`

Метод устанавливает элемент вектора по заданному индексу.

Пример

```
local vector = DocumentAPI.VectorString(2)
vector::__setitem(0, "12")
vector::__setitem(1, "13")
print(vector::__getitem(0)) -- 12
print(vector::__getitem(1)) -- 13
```

7.157 Таблица `DocumentAPI.VectorUInt`

Таблица `DocumentAPI.VectorUInt` предназначена для реализации массива данных.

Пример

```
vector = DocumentAPI.VectorUInt(3)
vector[0] = 1
vector[1] = 13
vector[2] = 25
print(vector:size()) -- 3
```

7.157.1 Метод `VectorUInt:back`

Метод возвращает последний элемент вектора.

Пример

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
vector:push_back(13)
print(vector:front()) -- 13
```

7.157.2 Метод `VectorUInt:clear`

Метод очищает содержимое вектора.

Пример

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
vector:clear()
print(vector:empty()) -- true
```

7.157.3 Метод `VectorUInt:empty`

Метод возвращает `true`, если вектор не содержит элементов.

Пример

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
print(vector:empty()) -- false
```

7.157.4 Метод `VectorUInt:front`

Метод возвращает первый элемент вектора.

Пример

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
vector:push_back(13)  
print(vector:front()) -- 12
```

7.157.5 Метод `VectorUInt:max_size`

Метод возвращает максимальный размер вектора.

Пример

```
local vector = DocumentAPI.VectorUInt()  
print(vector:max_size())
```

7.157.6 Метод `VectorUInt:pop_back`

Метод удаляет последний элемент вектора.

Пример

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
vector:pop_back()  
print(vector:size()) -- 0
```

7.157.7 Метод `VectorUInt:push_back`

Метод добавляет элемент в конец вектора.

Пример

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
print(vector:size()) -- 1
```

7.157.8 Метод `VectorUInt.size`

Метод возвращает размер вектора.

Пример

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
vector:push_back(13)  
vector:push_back(14)  
print(vector:size()) -- 3
```

7.157.9 Метод `VectorUInt.__getitem`

Метод возвращает элемент вектора по заданному индексу.

Пример

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
vector:push_back(13)  
print(vector:__getitem(0)) -- 12  
print(vector:__getitem(1)) -- 13
```

7.157.10 Метод `VectorUInt.__setitem`

Метод устанавливает элемент вектора по заданному индексу.

Пример


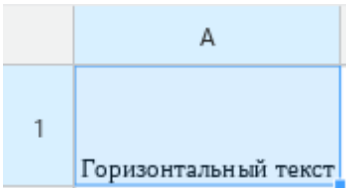

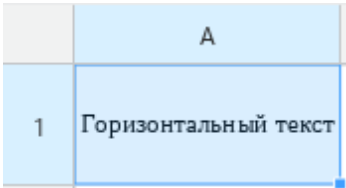

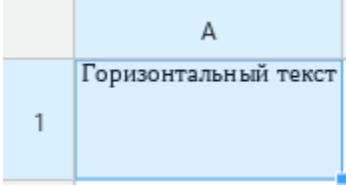
```
local vector = DocumentAPI.VectorUInt(2)  
vector:__setitem(0, 12)  
vector:__setitem(1, 13)  
print(vector:__getitem(0)) -- 12  
print(vector:__getitem(1)) -- 13
```

7.158 Таблица `DocumentAPI.VerticalAlignment`

В таблице 93 представлены константы видов выравнивания текста по вертикали.

Используется в [DocumentAPI.CellProperties](#), [DocumentAPI.ShapeProperties](#).

Таблица 93 – Виды выравнивания текста по вертикали

Наименование константы	Представление в интерфейсе	
DocumentAPI.VerticalAlignment_Bottom		
DocumentAPI.VerticalAlignment_Center		
DocumentAPI.VerticalAlignment_Top		

Пример

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A1")
local props = cell:getCellProperties()
props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
cell:setCellProperties(props)
```

7.159 Таблица DocumentAPI.VerticalAnchorAlignment

В таблице 94 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали. Используется в [DocumentAPI.VerticalTextAnchoredPosition](#).

Таблица 94 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
<code>DocumentAPI.VerticalAnchorAlignment_Top</code>	По верхнему краю
<code>DocumentAPI.VerticalAnchorAlignment_Bottom</code>	По нижнему краю
<code>DocumentAPI.VerticalAnchorAlignment_Center</code>	По центру
<code>DocumentAPI.VerticalAnchorAlignment_Inside</code> , <code>DocumentAPI.VerticalAnchorAlignment_Outside</code>	По границам

7.160 Таблица `DocumentAPI.VerticalRelativeTo`

В таблице 95 представлены типы размещения объекта относительно закрепленной позиции по вертикали. Используется в [DocumentAPI.VerticalTextAnchoredPosition](#).

Таблица 95 – Типы размещения объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
<code>DocumentAPI.VerticalRelativeTo_Character</code>	Символ
<code>DocumentAPI.VerticalRelativeTo_BaseLine</code>	Базовая линия
<code>DocumentAPI.VerticalRelativeTo_Paragraph</code>	Абзац
<code>DocumentAPI.VerticalRelativeTo_Page</code>	Страница
<code>DocumentAPI.VerticalRelativeTo_PageContent</code>	Содержимое страницы
<code>DocumentAPI.VerticalRelativeTo_PageTopMargin</code>	Верхнее поле страницы
<code>DocumentAPI.VerticalRelativeTo_PageBottomMargin</code>	Нижнее поле страницы
<code>DocumentAPI.VerticalRelativeTo_PageInsideMargin</code>	Внутреннее поле страницы
<code>DocumentAPI.VerticalRelativeTo_PageOutsideMargin</code>	Внешнее поле страницы

7.161 Таблица `DocumentAPI.VerticalTextAnchoredPosition`

Таблица `DocumentAPI.VerticalTextAnchoredPosition` предназначена для управления относительным положением объекта со смещением или выравниванием по вертикали. Пример использования см. в [InlineFrame.setPosition\(\)](#).

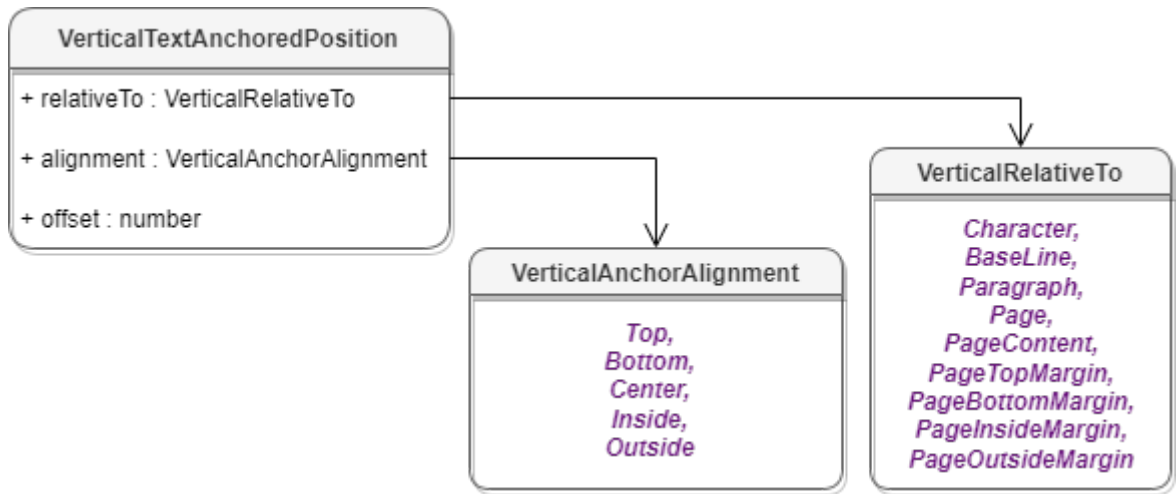


Рисунок 52 – Поля таблицы `DocumentAPI.VerticalTextAnchoredPosition`

Описание полей таблицы `DocumentAPI.VerticalTextAnchoredPosition` представлено в таблице 96.

Таблица 96 – Описание полей таблицы `DocumentAPI.VerticalTextAnchoredPosition`

Поле	Описание
<code>DocumentAPI.VerticalTextAnchoredPosition.alignment</code>	Тип выравнивания объекта относительно закрепленной позиции по вертикали VerticalAnchorAlignment .
<code>DocumentAPI.VerticalTextAnchoredPosition.relativeTo</code>	Тип размещения объекта относительно закрепленной позиции по вертикали VerticalRelativeTo .
<code>DocumentAPI.VerticalTextAnchoredPosition.offset</code>	Смещение объекта

7.161.1 Метод `VerticalTextAnchoredPosition.__eq`

Метод используется для определения эквивалентности двух положений объекта по вертикали.

Пример

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
pos1.vertical.offset = 1
```

```
local pos2 = DocumentAPI.TextAnchoredPosition()  
pos2.vertical =  
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)  
pos2.vertical.offset = 1  
  
print(pos1.vertical:__eq(pos2.vertical))
```

7.162 Таблица DocumentAPI.WorksheetPrinterFitType

В таблице 97 представлены варианты масштабирования при печати табличных документов. Используется в качестве поля worksheetPrinterFitType таблицы [DocumentAPI.PrintSettings](#).

Таблица 97 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
DocumentAPI.WorksheetPrinterFitType_ActualSize	Фактический размер
DocumentAPI.WorksheetPrinterFitType_ByPageScale	По масштабу страницы
DocumentAPI.WorksheetPrinterFitType_ByPageBreaksOnly	По разрыву страниц
DocumentAPI.WorksheetPrinterFitType_FitToPage	Вписать в страницу
DocumentAPI.WorksheetPrinterFitType_FitToWidth	Вписать по ширине
DocumentAPI.WorksheetPrinterFitType_FitToHeight	Вписать по высоте

8 Справочник функций DocumentAPI

8.1 Функция DocumentAPI.createSearch

Функция инициализирует механизм поиска для текущего документа. Возвращает ссылку на таблицу [DocumentAPI.Search](#), с помощью методов которой выполняются поисковые запросы.

Пример

```
search = DocumentAPI.createSearch(document)
ranges = search.findText("English")
```

8.2 Функция DocumentAPI.createScripting

Функция `DocumentAPI.createScripting` возвращает таблицу [DocumentAPI.Scripting](#). В качестве параметра используется текущий документ.

Пример

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

9 Справочник таблиц EditorAPI

9.1 Таблица EditorAPI.SelectionDirection

Таблица содержит параметры для управления направлением выделения. Используется в качестве аргумента метода [EditorAPI.changeSelection\(\)](#). Описание полей таблицы EditorAPI.SelectionDirection представлено в таблице 98.

Таблица 98 – Описание полей таблицы DocumentAPI.SelectionDirection

Поле	Описание
EditorAPI.SelectionDirection.Up	Изменить выделение вверх
EditorAPI.SelectionDirection.Down	Изменить выделение вниз
EditorAPI.SelectionDirection.Right	Изменить выделение направо
EditorAPI.SelectionDirection.Left	Изменить выделение налево
EditorAPI.SelectionDirection.DownRight	Изменить выделение вниз и направо
EditorAPI.SelectionDirection.UpLeft	Изменить выделение вверх и налево

Пример

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,  
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)
```

9.2 Таблица EditorAPI.SelectionMode

Таблица содержит варианты изменения текущего выделения. Используется в качестве аргумента метода [EditorAPI.changeSelection\(\)](#). Описание полей таблицы EditorAPI.SelectionMode представлено в таблице 99.

Таблица 99 – Описание полей таблицы EditorAPI.SelectionMode

Поле	Описание
EditorAPI.SelectionMode.Move	Переместить выделение
EditorAPI.SelectionMode.Resize	Изменить размер текущего выделения

Пример

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,  
EditorAPI.SelectionDirection.UpLeft, EditorAPI.TextSelectionUnit.Word)
```

9.3 Таблица EditorAPI.TableSelectionUnit

Таблица содержит параметры для управления шагом выделения в таблице документа. Используется в качестве аргумента метода [EditorAPI.changeSelection\(\)](#). Описание полей таблицы EditorAPI.TableSelectionUnit представлено в таблице 100.

Таблица 100 – Описание полей таблицы EditorAPI.TableSelectionUnit

Поле	Описание
EditorAPI.TableSelectionUnit.ToEdge	Направление смены выделения - угол таблицы
EditorAPI.TableSelectionUnit.ToClosestCell	Направление смены выделения - ближайшая ячейка

Пример

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,  
EditorAPI.SelectionDirection.UpLeft, EditorAPI.TableSelectionUnit.ToClosestCell)
```

9.4 Таблица EditorAPI.TextSelectionUnit

Таблица содержит параметры для управления шагом выделения в тексте документа. Используется в качестве аргумента метода [EditorAPI.changeSelection\(\)](#). Описание полей таблицы EditorAPI.TextSelectionUnit представлено в таблице 101.

Таблица 101 – Описание полей таблицы EditorAPI.TextSelectionUnit

Поле	Описание
EditorAPI.TextSelectionUnit.Character	Изменять выделение с шагом в один символ
EditorAPI.TextSelectionUnit.Word	Изменять выделение с шагом в одно слово
EditorAPI.TextSelectionUnit.Paragraph	Изменять выделение с шагом в один параграф
EditorAPI.TextSelectionUnit.Line	Изменять выделение с шагом в одну строку

Пример

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,  
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)
```

10 Справочник функций EditorAPI

Глобальная таблица EditorAPI содержит функции доступа к внешней функциональности редактора.

10.1 Функция EditorAPI.changeSelection

Функция EditorAPI.changeSelection позволяет изменить текущее выделение в текстовом или табличном документе.

Вызов функции

```
EditorAPI.changeSelection(mode, direction, unit, count = 1)
```

Где:

- mode – режим выделения, тип [DocumentAPI.SelectionMode](#);
- direction – направление выделения, тип [DocumentAPI.SelectionDirection](#);
- unit – шаг изменения выделения, тип [DocumentAPI.TableSelectionUnit](#) для таблиц, или [DocumentAPI.TextSelectionUnit](#) для текста;
- count – количество шагов, необязательный параметр, по умолчанию используется значение 1.

Функция возвращает true в случае, если выделение было изменено.

Пример для текстового документа

```
if (EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,  
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)) then  
    EditorAPI.messageBox("Selection changed")  
end
```

Пример для табличного документа

```
if (EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,  
EditorAPI.SelectionDirection.Right, EditorAPI.TableSelectionUnit.ToClosestCell,  
2)) then  
    EditorAPI.messageBox("Selection changed")  
end
```

10.2 Функция EditorAPI.getActiveWorksheet

Функция EditorAPI.getActiveWorksheet() возвращает активный лист в табличном документе (класс [Table](#)). Метод не предназначен для текстовых документов.

Пример

```
activeWorksheet = EditorAPI.getActiveWorksheet()  
print(activeWorksheet.getName()) -- Лист1
```

10.3 Функция EditorAPI.getSelection

Функция `EditorAPI.getSelection` предоставляет доступ к выделенному фрагменту документа.

В открытом текстовом документе может быть выделен только один фрагмент.

При использовании в редакторе текста функция `EditorAPI.getSelection` возвращает [Range](#), а при использовании в редакторе таблиц - [CellRange](#) или [CellRanges](#).

Пример для текстового редактора

Использование функции `EditorAPI.getSelection` в редакторе текста для печати выделенного фрагмента текста.

```
range = EditorAPI.getSelection()  
text = range.extractText()  
print(text)
```

Пример для табличного редактора

Использование функции `EditorAPI.getSelection` в редакторе таблиц для печати значений ячеек в выделенном фрагменте таблицы.

```
cellRange = EditorAPI.getSelection()  
for cell in cellRange.enumerate() do  
    print(cell:getFormattedValue())  
end
```

Использование функции `EditorAPI.getSelection` в редакторе таблиц для печати координат выделенных фрагментов.

```
local ranges = EditorAPI.getSelection()  
for range in ranges.enumerate() do  
    print(range:getTableRange():toString())  
end
```

10.4 Функция EditorAPI.isPrinterAvailable

Функция `EditorAPI.isPrinterAvailable` позволяет проверить доступность последнего использованного принтера. Возвращает `false`, если принтер недоступен.

Пример

```
if EditorAPI.isPrinterAvailable() then
    EditorAPI.messageBox("Printer is available")
else
    EditorAPI.messageBox("Printer is not available")
end
```

10.5 Функция EditorAPI.messageBox

Функция `EditorAPI.messageBox()` выводит на экран сообщение с заданным текстом и отображением кнопки **ОК**, при этом исполнение макрокоманды приостанавливается до нажатия кнопки **ОК**.

Вызов

```
messageBox(prompt : string)
messageBox(prompt : string)
messageBox(prompt : string, title : string)
```

Параметры

- `prompt` – текст сообщения;
- `title` – заголовок окна сообщения.

Пример

```
EditorAPI.messageBox(cell:getFormattedValue())
```

10.6 Функция EditorAPI.printDocument

Функция `EditorAPI.printDocument()` предоставляет возможность печати документа с заданными параметрами печати. Описание параметров печати представлено в разделе [DocumentAPI.PrintSettings](#).

Значения, возвращаемые функцией `EditorAPI.printDocument()`, перечислены в разделе [DocumentAPI.PrintDocumentResult](#).

Пример

```
local printSettings = {}
printSettings.printSelection = true
EditorAPI.printDocument(printSettings)
```

10.7 Функция `EditorAPI.setActiveWorksheet`

Функция `EditorAPI.setActiveWorksheet()` устанавливает активный лист в табличном документе. В качестве параметра используется имя листа. Возвращает `true`, если лист найден по имени и активирован. Метод не предназначен для текстовых документов.

Пример

```
print(EditorAPI.setActiveWorksheet("Лист1")) -- true or false
```

10.8 Функция `EditorAPI.setSelection`

Функция `EditorAPI.setSelection` позволяет выделить фрагмент документа. Если выделение успешно установлено, функция возвращает `true`.

В открытом текстовом документе может быть выделен только один фрагмент.

Вызов функции для текстового документа

```
bool EditorAPI.setSelection(range)
```

Где:

- `range` – выделяемый в текстовом документе фрагмент текста типа [DocumentAPI.Range](#).

Пример для текстового документа

```
EditorAPI.setSelection(document:getBlocks():getParagraph(0):getRange())
```

Вызов функции для табличного документа

```
bool EditorAPI.setSelection(cellRange)
```

Где:

- `cellRange` – выделяемый в табличном документе фрагмент таблицы типа [DocumentAPI.CellRange](#).

Пример для табличного документа

```
cellRange = document:getBlocks():getTable(0):getCellRange("A1:E5")
EditorAPI.setSelection(cellRange)
```

10.9 Функция `EditorAPI.showPrintDialog`

Функция `EditorAPI.showPrintDialog()` показывает стандартное окно печати редактора и распечатывает документ, если пользователь подтверждает необходимость

печати. Значения, возвращаемые функцией `EditorAPI.showPrintDialog()` перечислены в разделе [DocumentAPI.PrintDocumentResult](#).

Пример

```
printDocumentResult = EditorAPI.showPrintDialog()  
print(printDocumentResult)
```

11 Справочник методов EventsAPI

Глобальная таблица EventsAPI содержит методы, которые позволяют подписаться на события редактора табличных документов (см. [Обработка событий табличного документа](#)).

11.1 Метод EventsAPI.subscribeWorkbookBeforeClose

Метод EventsAPI.subscribeWorkbookBeforeClose позволяет задать функцию, которая выполняется перед закрытием документа.

Пример

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

Параметры

– toCancel: false, если документ будет закрыт после завершения события.

Если передаваемая функция возвращает true, то стандартный обработчик закрытия документа не будет вызван.

При одновременном закрытии нескольких окон с документами, выполнится только одно событие subscribeWorkbookBeforeClose, закрытие остальных окон будет отменено.

11.2 Метод EventsAPI.subscribeWorkbookBeforeSave

Метод EventsAPI.subscribeWorkbookBeforeSave позволяет задать функцию, которая выполняется перед сохранением документа.

Пример

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

Параметры

– isSaveAs: true, если событие было вызвано функцией "Сохранить как";

– toCancel: false, если документ будет сохранен после завершения события.

Если передаваемая функция возвращает true, то стандартный обработчик сохранения документа не будет вызван.

Событие `subscribeWorkbookBeforeSave` не выполняется при совместном редактировании в облаке.

11.3 Метод `EventsAPI.subscribeWorkbookOpen`

Метод `EventsAPI.subscribeWorkbookOpen` позволяет задать функцию, которая выполняется при открытии существующего документа.

Пример

```
EventsAPI.subscribeWorkbookOpen(function()  
    EditorAPI.messageBox("Macro 'Open' is working")  
end)
```

11.4 Метод `EventsAPI.subscribeWorksheetActivate`

Метод `EventsAPI.subscribeWorksheetActivate` позволяет задать функцию, которая выполняется при переключении между листами документа.

Пример

```
EventsAPI.subscribeWorksheetActivate(function(sheet)  
    EditorAPI.messageBox(string.format('Macro "Activate" is working! (%s)',  
sheet.getName()))  
end)
```

Параметры

– `sheet` (необязательный): лист, на который переключились, тип [DocumentAPI.Table](#).

11.5 Метод `EventsAPI.subscribeWorksheetChange`

Метод `EventsAPI.subscribeWorksheetChange` позволяет задать функцию, которая выполняется при изменении содержимого ячейки или нескольких ячеек.

Пример

```
EventsAPI.subscribeWorksheetChange(function(target)  
    EditorAPI.messageBox(  
        string.format('Macro "Change" is working! (BeginCell (%i, %i), LastCell  
(%i, %i)',  
            target.getBeginRow()+1,
```

```
target:getBeginColumn()+1,  
target:getLastRow()+1,  
target:getLastColumn()+1))  
end)
```

Параметры

- target: измененный диапазон, тип [DocumentAPI.CellRange](#).

Событие `subscribeWorksheetChange` не выполняется при:

- изменении стилей;
- изменении формата;
- изменении выравнивания;
- добавлении / удалении / изменении гиперссылок;
- добавлении / удалении комментариев;
- фильтрации / сортировке;
- скрытии строки / столбца;
- пересчете формул.

11.6 Метод `EventsAPI.subscribeWorksheetDeactivate`

Метод `EventsAPI.subscribeWorksheetDeactivate` позволяет задать функцию, которая выполняется при переключении между листами документа.

Пример

```
EventsAPI.subscribeWorksheetDeactivate(function(sheet)  
    EditorAPI.messageBox(string.format('Macro "Deactivate" is working! (%s)',  
sheet and sheet:getName() or 'nil'))  
end)
```

Параметры

- sheet (необязательный): лист, с которого переключились, тип [DocumentAPI.Table](#). Возвращает `nil`, если переключение вызвано удалением листа.

11.7 Метод `EventsAPI.subscribeWorksheetSelectionChange`

Метод `EventsAPI.subscribeWorksheetSelectionChange` позволяет задать функцию, которая выполняется при изменении выделения в документе.

Пример

```
EventsAPI.subscribeWorksheetSelectionChange(function(target)
    EditorAPI.messageBox(
        string.format('Macro "SelectionChange" is working! (BeginCell (%i, %i),
LastCell (%i, %i)',
            target:getBeginRow()+1,
            target:getBeginColumn()+1,
            target:getLastRow()+1,
            target:getLastColumn()+1))
end)
```

Параметры

— target: новая область выделения, тип [DocumentAPI.CellRange](#).

12 Функции для работы со строками в формате Юникод (UTF-8)

Для работы со строками, содержащими русские символы, можно использовать методы таблицы `utf8`. Предполагается, что аргументы методов являются допустимыми строками UTF-8.

12.1 Функция `utf8.char`

Функция `utf8.char` возвращает строку в формате UTF-8, соответствующую коду символа.

Вызов

```
utf8.char(code)
```

Параметры

– `code`: код символа UTF-8, тип `number`.

Возвращает

– `string`: символ UTF-8, полученный по коду.

Пример

```
print(utf8.char(244)) -- ô
```

12.2 Функция `utf8.charpattern`

Функция `utf8.charpattern` возвращает шаблон `"[\0-\x7F\xC2-\xF4][\x80-\xBF]+"` для определения последовательности символов формата UTF-8.

Пример

```
function len(s)
  local n = 0
  for match in s:gmatch(utf8.charpattern) do
    n = n + 1
  end
  return n
end

str = "МойОфис"
print(len(str))
```


12.3 Функция `utf8.codepoint`

Функция `utf8.codepoint` возвращает код заданного символа.

Вызов

```
utf8.codepoint(char)
```

Параметры

– `char`: символ UTF-8, тип `utf-char`.

Возвращает

– `number`: код символа UTF-8.

Примеры

```
print(utf8.codepoint("")) -- 29790
print(utf8.codepoint("A")) -- 1040
```

12.4 Функция `utf8.codes`

Функция `utf8.codes` возвращает последовательность кодов символов, из которых состоит строка UTF-8.

Вызов

```
utf8.codes(str)
```

Параметры

– `str`: строка в формате UTF-8

Возвращает

– итератор, с помощью которого можно получить коды символов исходной строки.

Пример

```
str = "МойОфис"
for p, c in utf8.codes(str) do
    print(c)
end

-- 1052, 1086, 1081, 1054, 1092, 1080, 1089
```

12.5 Функция `utf8.compare`

Функция `utf8.compare` возвращает результат сравнения двух строк согласно [алгоритму сортировки по Юникоду](#).

Вызов

```
utf8.compare(str1, str2, opt)
```

Параметры

- str1 – первая строка (string) в формате UTF-8;
- str2 – вторая строка (string) в формате UTF-8;
- opt – параметр (number) учета регистра при сравнении:
 - 0 – без учета регистра;
 - 1 – с учетом регистра.

Возвращает

- number: результат сравнения аргументов:
 - -1 – если str1 < str2;
 - 0 – если str1 = str2;
 - 1 – если str1 > str2.

Пример

```
print(utf8.compare("A", "a", 0)) -- 0, arg1 = arg2 с учетом регистра
print(utf8.compare("A", "a", 1)) -- -1, arg1 < arg2 без учета регистра
```

12.6 Функция utf8.isalpha

Функция utf8.isalpha проверяет, является ли буквенным символом переданный СИМВОЛ ИЛИ ЧИСЛО.

```
utf8.isalpha(char)
```

Параметр

- char: символ в кодировке UTF-8.

Возвращает

- boolean: true, если передан код буквенного символа.

Пример

```
print(utf8.isalpha('A')) -- true
print(utf8.isalpha('1')) -- false
```

12.7 Функция utf8.isdigit

Функция `utf8.isdigit` проверяет, является ли цифровым символом переданный символ или число.

```
utf8.isdigit(char)
```

Параметр

– `char`: UTF-8-character – символ в кодировке UTF-8.

Возвращает:

– `boolean`: значение `true`, если передан код цифрового символа.

Пример

```
print(utf8.isdigit("a")) -- false
print(utf8.isdigit("1")) -- true
```

12.8 Функция utf8.islower

Функция `utf8.islower` проверяет, находится ли в нижнем регистре переданный символ или строка.

```
utf8.islower(str)
```

Параметр

– `str`: строка, символ или число, представляющее код UTF-8.

Возвращает:

– `boolean`: `true`, если передан код символа в нижнем регистре.

Пример

```
print(utf8.islower("a")) -- false
print(utf8.islower("A")) -- true
```

12.9 Функция utf8.isupper

Функция `utf8.isupper` проверяет, находится ли в верхнем регистре переданный символ или строка.

```
utf8.isupper(str)
```

Параметр

– `str`: строка, символ или число, представляющее код UTF-8.

Возвращает:

– `boolean`: `true`, если передан код символа в верхнем регистре.

Пример

```
print(utf8.islower("a")) -- false
print(utf8.islower("A")) -- true
```

12.10 Функция `utf8.len`

Функция `utf8.len` позволяет определить длину заданной строки в символах.

```
utf8.len(str)
```

Параметр

– `str`: `string` – строка в формате UTF-8.

Возвращает:

– `number`: длина заданной строки в символах.

Пример

```
print(utf8.len("МойОфис")) -- 7
```

12.11 Функция `utf8.lower`

Функция `utf8.lower` возвращает строку в формате UTF-8, полученную из исходной строки путем преобразования в нижний регистр.

Вызов

```
utf8.lower(str)
```

Параметры

– `str`: строка в формате UTF-8

Возвращает

– `string`: строка в нижнем регистре.

Пример

```
print(utf8.lower("A")) -- a
print(utf8.lower("Abc")) -- abc
```

12.12 Функция `utf8.next`

Функция `utf8.next` позволяет получить байтовое смещение символа, следующего за указанным.

Вызов

```
utf8.next(str, offset)
```

Параметры

- `str` – строка (`string`) в формате UTF-8;
- `offset` – байтовое смещение внутри UTF-8 строки (по умолчанию равно 1).

Возвращает

- `number` – байтовое смещение следующего символа.

Пример

```
next_idx = utf8.next("АБВГДЕЖЗ", 5)
print(next_idx)
```

12.13 Функция `utf8.offset`

Функция `utf8.offset` возвращает позицию (в байтах), с которой начинается кодирование символа с заданной позицией.

```
utf8.offset(str, charPos)
```

Параметр

- `str: string` – строка в формате UTF-8;
- `charPos: number` – позиция символа.

Возвращает:

- `number`: позиция (в байтах), с которой начинается кодирование символа с заданной позицией.

Пример

```
print(utf8.offset("АБВГДЕЖЗ", 5)) -- 9
```

12.14 Функция `utf8.substr`

Функция `utf8.substr` возвращает подстроку в формате UTF-8, начиная с индекса `first` и заканчивая индексом `last`.

```
utf8.substr(str, first[, last])
```

Параметры

- `str`: `string` – исходная строка в формате UTF-8;
- `first`: `number` – позиция первого символа подстроки;
- `last`: `number` – позиция последнего символа подстроки (по умолчанию равна позиции последнего символа в строке).

Возвращает:

- `string`: подстрока в формате UTF-8
 - если позиция первого или последнего символа находится вне строки, то диапазон усекается до корректного;
 - если диапазон задан некорректно, то возвращается пустая строка.

Пример

```
print(utf8.substr("регистр", 5))    -- стр
print(utf8.substr("регистр", 0, 2)) -- ре
print(utf8.substr("регистр", 2, 1)) --
print(utf8.substr("регистр", 2, 100)) -- егистр
```

12.15 Функция `utf8.upper`

Функция `utf8.upper` возвращает строку в формате UTF-8, полученную из исходной строки путем преобразования в верхний регистр.

Вызов

```
utf8.upper(str)
```

Параметры

- `str`: строка в формате UTF-8

Возвращает

- `string`: строка в верхнем регистре.

Пример

```
print(utf8.upper("a")) -- A
print(utf8.upper("Abc")) -- ABC
```

13 Функции для работы с регулярными выражениями

13.1 Функция `Re.create`

Функция `Re.create` компилирует регулярное выражение и возвращает его в виде объекта. По умолчанию используется Perl - совместимый формат регулярных выражений.

Вызов

```
Re.create(pattern)
```

Параметры

- `pattern (string)` - строка шаблона.

Возвращает

- `regex (object)` - объект `Regex`, который содержит скомпилированное регулярное выражение для дальнейшего использования;
- `err (string)` - сообщение об ошибке или `nil`.

13.2 Функция `Re.match`

Сопоставляет скомпилированное регулярное выражение с заданной исходной строкой. Возвращает найденные подстроки.

Вызов

```
Re.match(subject, matchFlags, pattern)
```

Параметры

- `subject (string)` – исходная строка;
- `matchFlags (int)` – флаги, задающие правила применения регулярного выражения;
- `pattern (string, Regex)` – строка шаблона или скомпилированный шаблон.

Возвращает

- `matches (object)` – подстроки, найденные в соответствии с шаблоном;
- `err (string)` - сообщение об ошибке или `nil`.

13.2.1 Флаги, используемые в `Re.match`

Эти флаги определены в пространстве имен `Re.Match`. Они используются во всех алгоритмах. Когда регулярное выражение применяется к последовательности символов, применяются правила, описанные в таблице 102

Таблица 102 – Описание флагов `Re.Match`

Флаг	Описание
Default	Указывает, что работа с регулярными выражениями происходит в соответствии с обычными правилами: ECMA-262, спецификация языка ECMAScript, глава 15, часть 10, Регулярные Выражения.
NotBOB	Указывает, что выражения <code>"\A"</code> и <code>"\"</code> не должны совпадать с подмножеством <code>[first, first)</code> .
NotEOB	Указывает, что выражения <code>"\"</code> , <code>"\ z"</code> и <code>"\Z"</code> не должны совпадать с подмножеством <code>[last, last)</code> .
NotBOL	Указывает, что выражение <code>"^"</code> не должны совпадать с подмножеством <code>[first, first)</code> .
NotEOL	Указывает, что выражение <code>"\$"</code> не должно совпадать с подмножеством <code>[last, last)</code> .
NotBOW	Указывает, что выражения <code>"\<"</code> и <code>"\b"</code> не должны совпадать с подмножеством <code>[first, first)</code> .
NotEOW	Указывает, что выражения <code>"\>"</code> и <code>"\b"</code> не должны совпадать с подмножеством <code>[last, last)</code> .
Any	Указывает, что если существует более одного совпадения, то любое из совпадений является приемлемым результатом. Будет применено самое первое встретившееся совпадение, при этом, не всегда самое точное. Используйте этот флаг, если для вас важна скорость обработки, но не особо важно качество результата.
NotNull	Указывает, что выражение не может быть использовано для пустой последовательности.
Continuous	Указывает, что выражение должно применяться к подмножеству, которое начинается с начала.
Partial	Указывает, что если не найдено ни одного совпадения, то допустимо вернуть совпадение <code>[from, last)</code> , при этом <code>from! = last</code> , если может существовать более длинная последовательность символов <code>[from, to)</code> , из которых <code>[from, last)</code> - это префикс, который приведет к полному совпадению. Этот флаг используется при сопоставлении неполных или очень длинных текстов; дополнительную информацию см. документацию по частичным сравнениям.
Extra	Дает указание механизму поиска сохранять всю доступную информацию о наличии совпадений; если совпадение повторяется снова, то информация о каждом из них будет доступна через методы <code>match_results::captures()</code> или <code>sub_match_captures()</code> .
SingleLine	Эквивалентно обратному модификатору <code>"m/"</code> языка Perl; предотвращает поиск <code>^</code> после встроенного символа новой строки (для того, чтобы он совпадал только в начале исходного текста) и <code>\$</code> от поиска перед встроенным символом новой строки (чтобы он совпадал только в конце исходного текста).
PrevAvail	Указывает, что <code>--first</code> является допустимой позицией итератора, когда этот флаг установлен, тогда флаги <code>match_not_bol</code> и <code>match_not_bow</code>

Флаг	Описание
	игнорируются алгоритмами регулярных выражений (RE.7) и итераторов (RE.8).
DotNewLine	Указывает, что выражение "." не распознается как символ новой строки. Это инверсия модификатора "s/" языка Perl.
NotDotNull	Указывает, что выражение "." не распознается как символ null ("\0").
Posix	Указывает, что выражение должно быть обработано в соответствии с правилом POSIX <i>"leftmost-longest"</i> , независимо от того, какое выражение было скомпилировано. Эти правила плохо работают со многими специфичными для Perl моментами, например, такими как ленивые (<i>"non-greedy"</i>) повторы.
NoSubs	Заставляет выражение вести себя так, как будто оно не имеет найденных подмножеств, независимо от того, сколько их присутствует на самом деле. Класс <code>Matches</code> будет содержать только информацию об общем совпадении, а не о совпадении в подмножествах.

13.3 Функция `Re.replace`

Находит в заданной строке все фрагменты, удовлетворяющие регулярному выражению. Каждый найденный фрагмент форматируется в соответствии с форматтером и заменяет собой исходный текст.

Вызов

```
Re.replace(subject, formatter, matchFlags, pattern)
```

Параметры

- `subject (string)` – исходная строка для поиска;
- `formatter (string)` – строка, задающая форматирование найденных фрагментов;
- `matchFlags (int)` – флаги, задающие правила применения регулярного выражения, а также флаги, специфичные для замены;
- `pattern (string, Regex)` – строка шаблона или скомпилированный шаблон.

Возвращает

- `newString (string)` – новая строка с замененными подстроками;
- `err (string)` – сообщение об ошибке или `nil`.

13.4 Функция `Re.search`

Ищет скомпилированное регулярное выражение по заданной строке. Метод возвращает найденные подстроки.

Вызов

```
Re.search(subject, matchFlags, pattern)
```

Параметры

- `subject (string)` – исходная строка;
- `matchFlags (int)` – флаги, задающие правила применения регулярного выражения;
- `pattern (string, Regex)` – строка шаблона или скомпилированный шаблон.

Возвращает

- `matches (object)` – подстроки, найденные в соответствии с шаблоном;
- `err (string)` - сообщение об ошибке или `nil`.

13.5 Флаги, используемые для замены

Эти флаги определены в пространстве имен `Re.Replace`. Они используются в алгоритме, используемом методом `Re.replace()` и находятся в таблице 103

Таблица 103 – Описание флагов для функции `Re.replace()`

Флаг	Описание
<code>FormatDefault</code>	Когда к исходному тексту применяется регулярное выражение, и находится очередной фрагмент для замены, новая строка формируется с использованием функции замены <i>ECMAScript</i> , ECMA-262 , Спецификация языка ECMAScript, глава 15, часть 5.4.11 String.prototype.replace . Эта функциональность идентична описанной в руководстве Perl Format String Syntax . После того, как все неперекрывающиеся вхождения регулярного выражения найдены и заменены, фрагменты исходного текста, не соответствующие регулярному выражению, копируются в результирующую строку без изменений.
<code>FormatSed</code>	В случае, когда регулярное выражение применяется для замены в строке, новая строка формируется с использованием правил, описанных в стандарте IEEE Std 1003.1-2001, Portable Operating SystemInterface (POSIX), Shells and Utilities . См. также Sed Format String Syntax .
<code>FormatPerl</code>	В случае, когда регулярное выражение применяется для замены в строке, новая строка формируется по правилам Perl 5 .
<code>FormatLiteral</code>	В случае, когда регулярное выражение применяется для замены в строке, новая строка будет являться строковой копией заменяемого текста.
<code>FormatNoCopy</code>	В случае, когда регулярное выражение применяется для замены в строке, фрагменты, не соответствующие регулярному выражению, не будут копироваться в результирующую строку.
<code>FormatFirstOnly</code>	Когда данный флаг установлен при операции поиска или замены, то заменяется только первое вхождение регулярного выражения.

Флаг	Описание
FormatAll	Все синтаксические расширения включены, включая условные замены (<i>ddexpression1:expression2</i>). Для дополнительных деталей см. Руководство по форматированию строк Boost .

14 Функции для работы с датой и временем

В данном разделе описаны функции, предназначенные для работы с датой и временем:

[os.clock\(\)](#), [os.date\(\)](#), [os.difftime\(\)](#), [os.time\(\)](#).

Примеры задач, которые позволяют решать данные функции:

Преобразование DATETIME в POSIX

```
datetime = {year = 2013, month = 09, day = 13, hour = 21, min = 40, sec = 15}
seconds_since_epoch = os.time(datetime)
print(tostring(seconds_since_epoch))
```

Преобразование POSIX в DATETIME

```
seconds_since_epoch = 1379094015
datetime = os.date("!*t",seconds_since_epoch)
print( "year = "           .. tostring(datetime.year) .. " " ..
      "month = "          .. tostring(datetime.month) .. " " ..
      "day = "            .. tostring(datetime.day)  .. " " ..
      "hour = "           .. tostring(datetime.hour) .. " " ..
      "min = "            .. tostring(datetime.min)  .. " " ..
      "sec = "            .. tostring(datetime.sec)  .. " " ..
      "weekday = "        .. tostring(datetime.wday) .. " " ..
      "day of year = "    .. tostring(datetime.yday) .. " " ..
      "iddst = "          .. tostring(datetime.isdst))
```

Текущее время в формате POSIX

```
print(os.time())
```

Текущее время в формате DATETIME

```
datetime = os.date("!*t",os.time())
print(tostring(datetime.year) .. " " ..
      tostring(datetime.month) .. " " ..
      tostring(datetime.day)  .. " " ..
      tostring(datetime.hour) .. " " ..
      tostring(datetime.min)  .. " " ..
      tostring(datetime.sec)  .. " " ..
      tostring(datetime.wday) .. " " ..
      tostring(datetime.yday) .. " " ..
      tostring(datetime.isdst))
```

Текущий месяц

```
print(os.date("%B", os.time()))
```

День недели по дате

```
print(os.date("%W", os.time({ year=1997, month = 11, day = 10})))
```

Порядковый номер недели в году по дате

```
print(os.date("%W", os.time({ year=1997, month = 11, day = 10})))
```

Вывод даты / времени в различных форматах

```
print(os.date("%d.%m.%Y"))
print(os.date("%X", os.time()))
print(os.date("Сейчас %H часов %M минут %S секунд", os.time()))
print(os.date())
```

14.1 Функция os.clock

Функция `os.clock` возвращает время от начала запуска приложения / процесса. Время возвращается в секундах с точностью до миллисекунд.

Типичное применение - измерение времени выполнения фрагмента кода.

Пример измерения времени выполнения фрагмента кода

```
local x = os.clock()
local s = 0
for i=1,100000 do s = s + i end
print(string.format("elapsed time: %.2f\n", os.clock() - x))
```

14.2 Функция os.date

Функция `os.date(format, time)` возвращает форматированную дату / время. В качестве первого аргумента выступает формат, вторым аргументом является время в секундах. Оба аргумента не обязательны. При отсутствии второго аргумента будет использован заданный формат и текущая дата / время. Вызов без аргументов вернет текущую дату / время в формате 07.05.2024 13:33:10.

В строке формата могут быть использованы следующие опции:

%a - день недели, сокр. (англ.) (пример, Wed)

%A - день недели, полностью (англ.) (пример, Wednesday)

`%b` - месяц, сокр. (англ.) (пример, Sep)
`%B` - месяц, полностью (англ.) (пример, September)
`%c` - дата и время (по-умолчанию) (пример, 03/22/15 22:28:11)
`%d` - день месяца (пример, 22) [диапазон, 01-31]
`%H` - час, в 24-х часовом формате (пример, 23) [диапазон, 00-23]
`%I` - час, в 12-и часовом формате (пример, 11) [диапазон, 01-12]
`%M` - минута (пример, 48) [диапазон, 00-59]
`%m` - месяц (пример, 09) [диапазон, 01-12]
`%p` - время суток "am", или "pm"
`%S` - секунда (пример, 10) [диапазон, 00-59]
`%w` - день недели (пример, 3) [диапазон, 0-6, соответствует Sunday-Saturday]
`%x` - дата (пример, 09/16/98)
`%X` - время (пример, 23:48:10)
`%Y` - год, 4 цифры (пример, 2015)
`%y` - год, 2 цифры (пример, 15) [00-99]
`%%` - символ "%"
`*t` - вернет таблицу
`!t` - вернет таблицу (по Гринвичу)

Если параметр `format` начинается с `!`, то время форматируется в соответствии с универсальным глобальным временем (по Гринвичу). После этого опционального символа, если `format` равен `"*t"`, то `date` возвращает таблицу со следующими полями: `year` (год, четыре цифры), `month` (месяц, 1 – 12), `day` (день, 1 – 31), `hour` (час, 0 – 23), `min` (минуты, 0 – 59), `sec` (секунды, 0 – 61), `wday` (день недели, воскресенью соответствует 1), `yday` (день года), и `isdst` (флаг дневного времени суток, тип `boolean`).

Примеры

```
print (os.date( "%x " )) --> 07.05.2024
print (os.date( "%c " )) --> 25/04/07 10:10:05
```

14.3 Функция `os.difftime`

Функция `os.difftime(t1, t2)` возвращает число секунд, прошедших от времени `t1` до времени `t2`.

Пример сравнения двух дат в днях

```
reference = os.time{day=15, year=2024, month=2}
daysfrom = os.difftime(os.time(), reference) / (24 * 60 * 60) -- seconds in a
day
wholedays = math.floor(daysfrom)
print(wholedays)
```

14.4 Функция os.time

Функция `os.time()` возвращает время в формате posix (количество секунд, прошедших с 00:00:00 1 января 1970 года).

При вызове без аргументов возвращает текущее время.

Аргументом может являться таблица с обязательными ключами `year`, `month`, `day`, и необязательными `hour`, `min`, `sec`, `isdst`.

Пример

```
print (os.time()) -- текущее время в формате posix

local datetime = {year = 2017, month = 03, day = 1, hour = 14, min = 23, sec =
8}
print (os.time(datetime)) -- время, переданное в параметре
```

15 Класс Matches

Класс Matches содержит результат функций `Re.match()` и `Re.search()`.

15.1 Метод `getFirst`

Вызов

```
position, err = matches.getFirst(group)
```

Параметры

- `group (int, string)` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

Возвращает:

- `position (int)` – первая позиция (в байтах) исходной строки;
- `err (string)` - сообщение об ошибке или `nil`.

15.2 Метод `getLength`

Вызов

```
position, err = matches.getLength(group)
```

Параметры

- `group (int, string)` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

Возвращает:

- `length (int)` – длина исходной строки в байтах;
- `err (string)` - сообщение об ошибке или `nil`.

15.3 Метод `getSize`

Вызов

```
size, err = matches.getSize()
```

Возвращает:

- `size (int)` – количество найденных групп;
- `err (string)` - сообщение об ошибке или `nil`.

15.4 Метод `getString`

Вызов

```
substr, err = matches.getString(group, subject)
```

Параметры

- `group (int, string)` – позиция (или имя группы) найденных результатов, начинающаяся с 1;
- `subject (string)` – исходная строка. **Внимание:** объект `Matches` сохраняет только смещения и не хранит исходную строку. Таким образом, необходимо передать ту же строку, которая использовалась для поиска.

Возвращает:

- `substr (string)` – найденная подстрока;
- `err (string)` – сообщение об ошибке или `nil`.

15.5 Метод `_tostring`

Стандартная метафункция.

```
string = matches:__tostring()
```

Пример

```
local str = "-Номер:1234"
local regex = Re.create("-(\\w+):(\\d{4})")

local matches, err = Re.match(str, Re.Match.Default, regex)
print(tostring(regex), tostring(matches))

local number = matches:getString(3, str)
print(number)
```