



МойОфис Комплект Средств Разработки (SDK)

Руководство программиста

МОДУЛИ НАДСТРОЕК РЕДАКТОРОВ МОЙОФИС

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)»

МОДУЛИ НАДСТРОЕК РЕДАКТОРОВ МОЙОФИС

РУКОВОДСТВО ПРОГРАММИСТА

2.8

На 350 листах

Москва

2024

МойОфис

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

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

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

СОДЕРЖАНИЕ

1	Общие сведения	29
1.1	Назначение	29
1.2	Модули надстроек	30
1.3	Уровень подготовки пользователя	31
1.4	Системные требования	31
2	Настройки	32
2.1	Состав и структура надстройки	32
2.1.1	Файл регистрации надстройки	33
2.1.2	Файл сценария надстройки	37
2.1.2.1	Метод context.doWithApplication	39
2.1.2.2	Метод context.doWithDocument	40
2.1.2.3	Метод context.doWithSelection	41
2.1.3	Файл лицензионного соглашения	41
2.1.4	Локализация надстроек	43
2.1.4.1	Локализация полей таблицы регистрации надстройки и таблицы сценария надстройки	44
2.1.4.2	Словари локализации	45
2.2	Подпись надстройки	46
2.3	Подготовка сторонних модулей для использования в составе надстройки	49
2.3.1	Сборка внешних модулей для ОС Microsoft Windows	50
2.3.1.1	Установка MinGW	51
2.3.1.2	Установка и сборка LuaRocks	52
2.3.1.3	Установка модуля sqlite3complete	55
2.3.2	Сборка внешних модулей для ОС Linux	57
2.3.2.1	Установка LuaRocks	57
2.3.2.2	Установка модуля sqlite3complete	59
2.3.3	Использование сторонних модулей в составе надстройки	60
2.3.3.1	Использование сторонних модулей в ОС Microsoft Windows	60
2.3.3.2	Использование сторонних модулей в ОС Linux	61
2.4	Установка и управление надстройками в режиме диалога	63
2.4.1	Установка надстройки	63
2.4.2	Управление надстройками	69

МойОфис

2.4.3	Обновление надстройки	70
2.4.4	Удаление надстройки	71
2.4.5	Обновление редакторов	72
2.4.6	Редактирование кода установленной надстройки	73
2.5	Установка и обновление надстроек из командной строки	73
2.6	Вызов команд надстройки	74
2.7	Сообщения об ошибках	75
2.8	Функции для работы с файлами надстройки	76
2.8.1	Функция openBundledFile	77
2.8.2	Функция copyBundledDirectory	77
2.8.3	Функция getWorkingDirectory	77
2.8.4	Пример использования функций	77
2.9	Определение текущей локали	78
2.9.1	Функция getCurrentLocale	78
2.10	Пользовательский интерфейс в надстройках	78
2.11	Печать документа из надстройки	80
2.12	Пример создания надстройки	80
3	Объектная модель МойОфис SDK	83
4	Работа с документами	85
4.1	Работа с текстовым документом	85
4.1.1	Создание и открытие текстового документа	85
4.1.2	Сохранение и экспорт текстового документа	85
4.1.3	Разделы (секции) документа	86
4.1.3.1	Работа с колонтитулами раздела	88
4.1.3.2	Управление ориентацией и свойствами страниц раздела	88
4.1.4	Работа с таблицами текстового документа	89
4.1.5	Работа с закладками	90
4.1.6	Рецензирование документов	92
4.1.7	Поиск в текстовом документе	94
4.1.8	Работа с графическими объектами	94
4.2	Работа с табличным документом	97
4.2.1	Создание и открытие табличного документа	97
4.2.2	Сохранение и экспорт табличного документа	98

МойОфис

4.2.3	Диаграммы	99
4.2.4	Поиск в табличном документе	99
4.2.5	Работа с графическими объектами	100
4.2.6	Работа с листами табличного документа	102
4.2.7	Работа со сводными таблицами	103
4.2.7.1	Получение диапазона исходных данных сводной таблицы	104
4.2.7.2	Получение диапазона размещения сводной таблицы	104
4.2.7.3	Получение неподдерживаемых свойств сводной таблицы	104
4.2.7.4	Получение флагов отображения общих итогов для строк и колонок	105
4.2.7.5	Получение заголовков сводной таблицы	105
4.2.7.6	Получение и применение фильтра для сводной таблицы	105
4.2.7.7	Получение полей из области фильтров	106
4.2.7.8	Получение полей из области значений	106
4.2.7.9	Получение полей из области строк	106
4.2.7.10	Получение полей из области колонок	107
4.2.7.11	Получение настроек отображения сводной таблицы	107
4.2.7.12	Обновление сводной таблицы	108
4.3	Работа с макрокомандами	108
4.4	Работа с именованными диапазонами	109
4.4.1	Доступ к именованным диапазонам	109
4.4.2	Получение свойств именованного диапазона	110
4.4.3	Получение коллекции именованных диапазонов	110
4.4.4	Добавление именованного диапазона	110
4.4.5	Удаление именованного диапазона	111
4.4.6	Получение параметров именованного диапазона	111
4.5	Работа со строками и столбцами таблиц	111
4.5.1	Группировка строк и колонок таблицы	111
4.5.2	Управление видимостью строк / колонок	111
4.6	Работа с ячейками таблиц	112
4.6.1	Доступ к ячейкам	112
4.6.2	Форматирование ячеек	114
4.6.3	Форматирование границ ячеек	115
4.6.4	Объединение и разделение ячеек таблицы	116

МойОфис

5	Глобальные функции DocumentAPI	118
5.1	Функция DocumentAPI:createSearch	118
5.2	Функция DocumentAPI.createScripting	118
6	Справочник таблиц DocumentAPI	119
6.1	Таблица DocumentAPI.AbsoluteFrame	119
6.1.1	Метод AbsoluteFrame:moveTo	119
6.1.2	Метод AbsoluteFrame:getTopLeft	120
6.1.3	Метод AbsoluteFrame:scale	120
6.1.4	Метод AbsoluteFrame:setDimensions	120
6.1.5	Метод AbsoluteFrame:getDimensions	121
6.2	Таблица DocumentAPI.AccountingCellFormatting	121
6.3	Таблица DocumentAPI.Alignment	122
6.4	Таблица DocumentAPI.Application	122
6.4.1	Метод application:createDocument	123
6.4.2	Метод application:loadDocument	123
6.5	Таблица DocumentAPI.Blocks	124
6.5.1	Метод Blocks:getBlock	125
6.5.2	Метод Blocks:getParagraph	125
6.5.3	Метод Blocks:getTable	125
6.5.4	Метод Blocks:getShape	125
6.5.5	Метод Blocks:getField	125
6.5.6	Метод Blocks:enumerate	126
6.5.7	Метод Blocks:enumerateParagraphs	126
6.5.8	Метод Blocks:enumerateTables	126
6.5.9	Метод Blocks:enumerateShapes	126
6.5.10	Метод Blocks:enumerateFields	126
6.6	Таблица DocumentAPI.Block	127
6.6.1	Методы toParagraph, toTable, toShape, toField	127
6.6.2	Метод Block.getRange	127
6.6.3	Метод Block.remove	127
6.6.4	Метод Block.getSection	128
6.7	Таблица DocumentAPI.Borders	128
6.8	Таблица DocumentAPI.Bookmarks	129

МойОфис

6.8.1	Метод Bookmarks:getBookmarkRange	129
6.8.2	Метод Bookmarks:removeBookmark	129
6.9	Таблица DocumentAPI.Cell	130
6.9.1	Метод Cell:getRange	130
6.9.2	Метод Cell:setBorders	130
6.9.3	Метод Cell:setFormula	130
6.9.4	Метод Cell:getFormat	131
6.9.5	Метод Cell:setFormat	131
6.9.6	Метод Cell:getFormattedValue	133
6.9.7	Метод Cell:setFormattedValue	134
6.9.8	Метод Cell:unmerge	134
6.9.9	Метод Cell:getHyperlink	134
6.9.10	Метод Cell:setContent	134
6.9.11	Метод Cell:getBorders	134
6.9.12	Метод Cell:getRawValue	135
6.9.13	Метод Cell:getCustomFormat	135
6.9.14	Метод Cell:setCustomFormat	135
6.9.15	Метод Cell:setBool	135
6.9.16	Метод Cell:setNumber	135
6.9.17	Метод Cell:setText	135
6.9.18	Метод Cell:getFormulaAsString	136
6.9.19	Метод Cell:getCellProperties	136
6.9.20	Метод Cell:setCellProperties	136
6.9.21	Метод Cell:getParagraphProperties	136
6.9.22	Метод Cell:setParagraphProperties	137
6.9.23	Метод Cell:getPivotTable	137
6.10	Таблица DocumentAPI.CellFormat	137
6.11	Таблица DocumentAPI.CellProperties	139
6.12	Таблица DocumentAPI.CellPosition	141
6.12.1	Поле CellPosition.column	141
6.12.2	Поле CellPosition.row	141
6.12.3	Метод CellPosition:toString	142
6.13	Таблица DocumentAPI.CellRange	142

МойОфис

6.13.1	Метод CellRange:enumerate	142
6.13.2	Метод CellRange:getBeginRow	142
6.13.3	Метод CellRange:getBeginColumn	143
6.13.4	Метод CellRange:getLastRow	143
6.13.5	Метод CellRange:getLastColumn	143
6.13.6	Метод CellRange:setBorders	143
6.13.7	Метод CellRange:insertCurrentDateTime	144
6.13.8	Метод CellRange:getCellProperties	144
6.13.9	Метод CellRange:setCellProperties	144
6.13.10	Метод CellRange:merge	145
6.13.11	Метод CellRange:unmerge	145
6.14	Таблица DocumentAPI.CellRangePosition	145
6.14.1	Метод CellRangePosition:toString	146
6.15	Таблица DocumentAPI.Chart	146
6.15.1	Метод Chart:getType	147
6.15.2	Метод Chart:setType	147
6.15.3	Метод Chart:getRangesCount	148
6.15.4	Метод Chart:getRange	148
6.15.5	Метод Chart:setRange	148
6.15.6	Метод Chart:getTitle	148
6.15.7	Метод Chart:setRect	148
6.15.8	Метод Chart:isEmpty	149
6.15.9	Метод Chart:isSolidRange	149
6.15.10	Метод Chart:is3D	149
6.15.11	Метод Chart:getDirectionType	149
6.15.12	Метод Chart:getChartLabels	149
6.15.13	Метод Chart:getRangeAsString	150
6.15.14	Метод Chart:applySettings	150
6.16	Таблица DocumentAPI.ChartLabelsDetectionMode	151
6.17	Таблица DocumentAPI.ChartLabelsInfo	151
6.18	Таблица DocumentAPI.Charts	152
6.18.1	Метод Charts:getChartsCount	153
6.18.2	Метод Charts:getChart	153

6.18.3	Метод Charts:getChartIndexByDrawingIndex	154
6.19	Таблица DocumentAPI.ChartRangeInfo	154
6.20	Таблица DocumentAPI.ChartRangeType	154
6.21	Таблица DocumentAPI.ChartSeriesDirectionType	155
6.22	Таблица DocumentAPI.ChartType	155
6.23	Таблица DocumentAPI.Color	156
6.23.1	Метод Color:getRGBAColor	157
6.23.2	Метод Color:getThemeColorID	157
6.24	Таблица DocumentAPI.ColorRGBA	157
6.25	Таблица DocumentAPI.Comments	158
6.25.1	Метод Comments:enumerate	159
6.26	Таблица DocumentAPI.Comment	159
6.26.1	Метод Comment:getRange	159
6.26.2	Метод Comment:getText	159
6.26.3	Метод Comment:getInfo	160
6.26.4	Метод Comment:isResolved	160
6.26.5	Метод Comment:getReplies	160
6.27	Таблица DocumentAPI.CurrencyCellFormatting	160
6.28	Таблица DocumentAPI.CurrencySignPlacement	162
6.29	Таблица DocumentAPI.DatePatterns	162
6.30	Таблица DocumentAPI.DateTime	163
6.31	Таблица DocumentApi.DateTimeFormat	163
6.32	Таблица DocumentAPI.DateTimeCellFormatting	163
6.33	Таблица DocumentAPI.document	164
6.33.1	Метод document:saveAs	164
6.33.2	Метод document:exportAs	164
6.33.3	Метод document:merge	165
6.33.4	Метод document:getBlocks	166
6.33.5	Метод document:getBookmarks	166
6.33.6	Метод document:getScripts	166
6.33.7	Метод document:getRange	167
6.33.8	Метод document:isChangesTrackingEnabled	167
6.33.9	Метод document:setChangesTrackingEnabled	167

6.33.10	Метод document:getComments	167
6.33.11	Метод document:setPageProperties	168
6.33.12	Метод document:setFormulaType	168
6.33.13	Метод document:getFormulaType	168
6.33.14	Метод document:setPageOrientation	168
6.33.15	Метод document:enumerateSections	169
6.33.16	Метод document:getSections	169
6.33.17	Метод document:setMirroredMarginsEnabled	169
6.33.18	Метод document:areMirroredMarginsEnabled	169
6.33.19	Метод document:getPivotTablesManager	169
6.33.20	Метод document:getNamedExpressions	170
6.34	Таблица DocumentAPI.ExportFormat	170
6.35	Таблица DocumentAPI.Field	170
6.36	Таблица DocumentAPI.Fill	170
6.36.1	Метод Fill:getColor	170
6.36.2	Метод Fill:getUrl	170
6.36.3	Метод Fill:isNoFill	171
6.37	Таблица DocumentAPI.DocumentSettings	171
6.38	Таблица DocumentAPI.DocumentFormat	171
6.39	Таблица DocumentAPI.DocumentType	172
6.40	Таблица DocumentAPI.DSVSettings	172
6.41	Таблица DocumentAPI.Encoding	173
6.42	Таблица DocumentAPI.FormulaType	173
6.43	Таблица DocumentAPI.FractionCellFormatting	174
6.44	Таблица DocumentAPI.FrozenRangePosition	175
6.44.1	Конструкторы	175
6.44.2	Метод FrozenRangePosition:create	175
6.44.3	Метод FrozenRangePosition:createFrozenArea	175
6.44.4	Метод FrozenRangePosition:createFrozenRows	176
6.44.5	Метод FrozenRangePosition:createFrozenCols	176
6.44.6	Метод FrozenRangePosition:isRowsCols	176
6.44.7	Метод FrozenRangePosition:isArea	176
6.44.8	Метод FrozenRangePosition:isRows	177

6.44.9	Метод FrozenRangePosition:isCols	177
6.45	Таблица DocumentAPI.HeaderFooter	177
6.45.1	Метод HeaderFooter:getType	177
6.45.2	Метод HeaderFooter:getBlocks	177
6.45.3	Метод HeaderFooter:getRange	178
6.46	Таблица DocumentAPI.HeaderFooterType	178
6.47	Таблица DocumentAPI.HeadersFooters	178
6.47.1	Метод HeadersFooters:enumerate	178
6.48	Таблица DocumentAPI.HorizontalAnchorAlignment	179
6.49	Таблица DocumentAPI.HorizontalRelativeTo	179
6.50	Таблица DocumentAPI.HorizontalTextAnchoredPosition	180
6.51	Таблица DocumentAPI.Hyperlink	181
6.51.1	Метод Hyperlink: __eq	181
6.52	Таблица DocumentAPI.Image	181
6.52.1	Метод Image:getFrame	182
6.52.2	Метод Image:remove	182
6.53	Таблица DocumentAPI.Images	183
6.53.1	Метод Images:enumerate	183
6.54	Таблица DocumentAPI.InlineFrame	183
6.54.1	Метод InlineFrame:setPosition	184
6.54.2	Метод InlineFrame:getPosition	184
6.54.3	Метод InlineFrame:setDimensions	185
6.54.4	Метод InlineFrame:getDimensions	185
6.54.5	Метод InlineFrame:setWrapType	185
6.54.6	Метод InlineFrame:getWrapType	185
6.55	Таблица DocumentAPI.Insets	186
6.56	Таблица DocumentAPI.ListSchema	186
6.57	Таблица DocumentAPI.LineEndingProperties	187
6.58	Таблица DocumentAPI.LineEndingStyle	188
6.59	Таблица DocumentAPI.LineProperties	189
6.59.1	Поле LineProperties.style	190
6.59.2	Поле LineProperties.width	190
6.59.3	Поле LineProperties.color	190

6.59.4	Поле LineProperties.headLineEndingProperties	191
6.59.5	Поле LineProperties.tailLineEndingProperties	191
6.60	Таблица DocumentAPI.LineSpacing	191
6.61	Таблица DocumentAPI.LineSpacingRule	191
6.62	Таблица DocumentAPI.LineStyle	193
6.63	Таблица DocumentAPI.LoadDocumentSettings	194
6.64	Таблица DocumentAPI.LocaleInfo	195
6.65	Таблица DocumentAPI.MediaObjects	196
6.65.1	Метод MediaObjects:enumerate	196
6.66	Таблица DocumentAPI.MediaObject	196
6.66.1	Метод MediaObject:toImage	197
6.66.2	Метод MediaObject:getFrame	197
6.67	Таблица DocumentAPI.NamedExpressions	198
6.67.1	Метод NamedExpressions:get	198
6.67.2	Метод NamedExpressions:enumerate	199
6.67.3	Метод NamedExpression:addExpression	199
6.67.4	Метод NamedExpressions:removeExpression	199
6.68	Таблица DocumentAPI.NamedExpression	200
6.68.1	Метод NamedExpression:getName	200
6.68.2	Метод NamedExpression:getExpression	200
6.68.3	Метод NamedExpression:getCellRange	200
6.69	Таблица DocumentAPI.NamedExpressionsValidationResult	200
6.70	Таблица DocumentAPI.NumberCellFormatting	201
6.71	Таблица DocumentAPI.PageOrientation	201
6.72	Таблица DocumentAPI.PageProperties	202
6.73	Таблица DocumentAPI.Paragraph	203
6.73.1	Метод Paragraph:getParagraphProperties	203
6.73.2	Метод Paragraph:setParagraphProperties	204
6.73.3	Метод Paragraph:getListSchema	204
6.73.4	Метод Paragraph:setListSchema	205
6.73.5	Метод Paragraph:getListLevel	205
6.73.6	Метод Paragraph:setListLevel	205
6.73.7	Метод Paragraph:increaseListLevel	205

6.73.8	Метод Paragraph:decreaseListLevel	206
6.74	Таблица DocumentAPI.Paragraphs	206
6.74.1	Метод Paragraphs:setListSchema	206
6.74.2	Метод Paragraphs:setListLevel	207
6.74.3	Метод Paragraphs:increaseListLevel	207
6.74.4	Метод Paragraphs:decreaseListLevel	207
6.74.5	Метод Paragraphs:enumerate	207
6.75	Таблица DocumentAPI.ParagraphProperties	208
6.76	Таблица DocumentAPI.PercentageCellFormatting	211
6.77	Таблица DocumentAPI.PointU	211
6.77.1	Метод PointU:toString	212
6.78	Таблица DocumentAPI.Position	212
6.78.1	Метод Position:insertText	212
6.78.2	Метод Position:insertTable	212
6.78.3	Метод Position:insertPageBreak	213
6.78.4	Метод Position:insertLineBreak	213
6.78.5	Метод Position:insertSectionBreak	213
6.78.6	Метод Position:insertHyperlink	214
6.78.7	Метод Position:insertImage	214
6.78.8	Метод Position:removeBackward	214
6.78.9	Метод Position:removeForward	215
6.78.10	Метод Position:insertBookmark	215
6.79	Таблица DocumentAPI.PrintDocumentResult	215
6.80	Таблица DocumentAPI.PrintSettings	215
6.81	Таблица DocumentAPI.PivotTablesManager	217
6.81.1	Метод PivotTablesManager:create	217
6.82	Таблица DocumentAPI.PivotTable	217
6.82.1	Метод PivotTable:remove	218
6.82.2	Метод PivotTable:getSourceRangeAddress	218
6.82.3	Метод PivotTable:getSourceRange	218
6.82.4	Метод PivotTable:getPivotRange	218
6.82.5	Метод PivotTable:changeSourceRange	219
6.82.6	Метод PivotTable:isRowGrandTotalEnabled	219

6.82.7	Метод PivotTable.isColumnGrandTotalEnabled	219
6.82.8	Метод PivotTable.getPivotTableCaptions	219
6.82.9	Метод PivotTable.getPivotTableLayoutSettings	219
6.82.10	Метод PivotTable.getUnsupportedFeatures	220
6.82.11	Метод PivotTable.getFieldsList	220
6.82.12	Метод PivotTable.getRowFields	220
6.82.13	Метод PivotTable.getColumnFields	220
6.82.14	Метод PivotTable.getValueFields	221
6.82.15	Метод PivotTable.getPageFields	221
6.82.16	Метод PivotTable.getFieldCategories	221
6.82.17	Метод PivotTable.getFieldItems	221
6.82.18	Метод PivotTable.getFieldItemsByName	221
6.82.19	Метод PivotTable.getFilter	222
6.82.20	Метод PivotTable.getFilters	222
6.82.21	Метод PivotTable.update	222
6.82.22	Метод PivotTable.createPivotTableEditor	222
6.83	Таблица DocumentAPI.PivotTableCaptions	223
6.84	Таблица DocumentAPI.PivotTableLayoutSettings	223
6.85	Таблица DocumentAPI.PivotTableReportLayout	224
6.86	Таблица DocumentAPI.PageFieldOrder	224
6.87	Таблица DocumentAPI.PivotTableUnsupportedFeature	225
6.88	Таблица DocumentAPI.PivotTableFieldCategories	225
6.88.1	Метод PivotTableFieldCategories.enumerate	225
6.89	Таблица DocumentAPI.PivotTableFunction	225
6.90	Таблица DocumentAPI.PivotTableFilters	226
6.90.1	Метод PivotTableFilters.enumerate	226
6.91	Таблица DocumentAPI.PivotTableFilter	227
6.91.1	Метод PivotTableFilter.getFieldName	227
6.91.2	Метод PivotTableFilter.getCount	227
6.91.3	Метод PivotTableFilter.getName	228
6.91.4	Метод PivotTableFilter.isHidden	228
6.91.5	Метод PivotTableFilter.setHidden	228
6.92	Таблица DocumentAPI.PivotTableField	228

6.93	Таблица DocumentAPI.PivotTableFieldProperties	229
6.94	Таблица DocumentAPI.PivotTableCategoryField	229
6.95	Таблица DocumentAPI.PivotTableValueField	229
6.96	Таблица DocumentAPI.PivotTablePageField	230
6.97	Таблица DocumentAPI.PivotTableItems	230
6.97.1	Метод PivotTableItems:enumerate	230
6.98	Таблица DocumentAPI.PivotTableItem	231
6.98.1	Метод PivotTableItem:getName	231
6.98.2	Метод PivotTableItem:getAlias	231
6.98.3	Метод PivotTableItem:getItemType	231
6.98.4	Метод PivotTableItem:isCollapsed	231
6.99	Таблица DocumentAPI.PivotTableItemType	232
6.100	Таблица DocumentAPI.PivotTableEditor	232
6.100.1	Метод PivotTableEditor:addField	232
6.100.2	Метод PivotTableEditor:moveField	233
6.100.3	Метод PivotTableEditor:removeField	233
6.100.4	Метод PivotTableEditor:reorderField	233
6.100.5	Метод PivotTableEditor:enableField	233
6.100.6	Метод PivotTableEditor:disableField	234
6.100.7	Метод PivotTableEditor:setSummarizeFunction	234
6.100.8	Метод PivotTableEditor:setFilter	234
6.100.9	Метод PivotTableEditor:setFilters	235
6.100.10	Метод PivotTableEditor:setCaptions	235
6.100.11	Метод PivotTableEditor:setLayoutSettings	235
6.100.12	Метод PivotTableEditor:setGrandTotalSettings	236
6.100.13	Метод PivotTableEditor:apply	236
6.101	Таблица DocumentAPI.PivotTableUpdateResult	236
6.102	Таблица DocumentAPI.PivotTableFieldCategory	237
6.103	Таблица DocumentAPI.Range	237
6.103.1	Метод Range:getBegin	239
6.103.2	Метод Range:getEnd	239
6.103.3	Метод Range:extractText	240
6.103.4	Метод Range:removeContent	240

6.103.5	Метод Range:lockContent	240
6.103.6	Метод Range:unlockContent	241
6.103.7	Метод Range:isContentLocked	241
6.103.8	Метод Range:replaceText	242
6.103.9	Метод Range:setHyperlink	242
6.103.10	Метод Range:getTextProperties	243
6.103.11	Метод Range:setTextProperties	243
6.103.12	Метод Range:enumerateBlocks	244
6.103.13	Метод Range:enumerateTrackedChanges	244
6.103.14	Метод Range:getComments	244
6.103.15	Метод Range:getParagraphs	245
6.103.16	Метод Range:getImages	245
6.103.17	Метод Range:getInlineObjects	245
6.104	Таблица DocumentAPI.RangeBorders	246
6.105	Таблица DocumentAPI.RectU	246
6.105.1	Метод RectU:toString	246
6.106	Таблица DocumentAPI.SaveDocumentSettings	247
6.107	Таблица DocumentAPI.ScaleFrom	247
6.108	Таблица DocumentAPI.ScientificCellFormatting	248
6.109	Таблица DocumentAPI.Search	248
6.109.1	Метод Search:findText	248
6.110	Таблица DocumentAPI.ScriptPosition	249
6.111	Таблица DocumentAPI.Scripts	249
6.111.1	Метод Scripts:getScript	249
6.111.2	Метод Scripts:setScript	250
6.111.3	Метод Scripts:removeScript	250
6.111.4	Метод Scripts:enumerate	250
6.112	Таблица DocumentAPI.Script	250
6.112.1	Таблица DocumentAPI.Scripting	250
6.112.1.1	Метод Scripting:runScript	251
6.112.2	Метод Script:getName	251
6.112.3	Метод Script:setName	251
6.112.4	Метод Script:getBody	251

6.112.5	Метод Script:setBody	251
6.113	Таблица DocumentAPI.Sections	252
6.113.1	Метод Sections:enumerate	252
6.114	Таблица DocumentAPI.Section	252
6.114.1	Метод Section:setPageProperties	252
6.114.2	Метод Section:getPageProperties	252
6.114.3	Метод Section:setPageOrientation	253
6.114.4	Метод Section:getPageOrientation	253
6.114.5	Метод Section:getRange	253
6.114.6	Метод Section:getHeaders	253
6.114.7	Метод Section:getFooters	254
6.115	Таблица DocumentAPI.SizeU	254
6.115.1	Метод SizeU:toString	254
6.116	Таблица DocumentAPI.Shape	255
6.116.1	Метод Shape:getShapeProperties	255
6.116.2	Метод Shape:setShapeProperties	255
6.117	Таблица DocumentAPI.ShapeProperties	255
6.118	Таблица DocumentAPI.ShapeTextLayout	255
6.119	Таблица DocumentAPI.Table	256
6.119.1	Метод Table:setName	256
6.119.2	Метод Table:getName	257
6.119.3	Метод Table:getRowsCount	257
6.119.4	Метод Table:getColumnsCount	257
6.119.5	Метод Table:getCell	257
6.119.6	Метод Table:getCellRange	258
6.119.7	Метод Table:insertColumnAfter	258
6.119.8	Метод Table:insertColumnBefore	259
6.119.9	Метод Table:insertRowAfter	259
6.119.10	Метод Table:insertRowBefore	260
6.119.11	Метод Table:removeColumn	260
6.119.12	Метод Table:removeRow	261
6.119.13	Метод Table:groupRows	261
6.119.14	Метод Table:ungroupRows	261

6.119.15	Метод Table:clearRowGroups	262
6.119.16	Метод Table:groupColumns	262
6.119.17	Метод Table:ungroupColumns	262
6.119.18	Метод Table:clearColumnGroups	262
6.119.19	Метод Table:setColumnWidth	263
6.119.20	Метод Table:setRowHeight	263
6.119.21	Метод Table:duplicate	264
6.119.22	Метод Table:remove	264
6.119.23	Метод Table:moveTo	264
6.119.24	Метод Table:setShowZeroValue	264
6.119.25	Метод Table:getShowZeroValue	265
6.119.26	Метод Table:setVisible	265
6.119.27	Метод Table:isVisible	265
6.119.28	Метод Table:setColumnsVisible	265
6.119.29	Метод Table:setRowsVisible	266
6.119.30	Метод Table:getCharts	266
6.119.31	Метод Table:getFrozenRange	266
6.119.32	Метод Table:freeze	267
6.119.33	Метод Table:___eq	267
6.119.34	Метод Table:___ne	267
6.119.35	Метод Table:setPrintArea	268
6.119.36	Метод Table:setPrintAreas	268
6.119.37	Метод Table:getPrintAreas	268
6.119.38	Метод Table:getImages	268
6.119.39	Метод Table:getMediaObjects	269
6.119.40	Метод Table:getNamedExpressions	269
6.120	Таблица DocumentAPI.TableRangeInfo	269
6.121	Таблица DocumentAPI.TextAnchoredPosition	270
6.122	Таблица DocumentAPI.TextProperties	270
6.123	Таблица DocumentAPI.TextWrapType	272
6.124	Таблица DocumentAPI.TextLayout	273
6.125	Таблица DocumentAPI.TextOrientation	273
6.125.1	Метод TextOrientation:getAngle	274

6.126 Таблица DocumentAPI.TimePatterns	274
6.127 Таблица DocumentAPI.TimeZone	274
6.128 Таблица DocumentAPI.ThemeColorID	274
6.129 Таблица DocumentAPI.TrackedChange	275
6.129.1 Метод TrackedChange:getRange	276
6.129.2 Метод TrackedChange:getType	276
6.129.3 Метод TrackedChange:getInfo	276
6.130 Таблица DocumentAPI.TrackedChangeInfo	276
6.131 Таблица DocumentAPI.TrackedChangeType	277
6.132 Таблица DocumentAPI.UserInfo	277
6.133 Таблица DocumentAPI.WorksheetPrinterFitType	278
6.134 Таблица DocumentAPI.ValueFieldsOrientation	278
6.135 Таблица DocumentAPI.VectorString	278
6.135.1 Метод VectorString:size	279
6.135.2 Метод VectorString:max_size	279
6.135.3 Метод VectorString:empty	279
6.135.4 Метод VectorString:clear	279
6.135.5 Метод VectorString:push_back	280
6.135.6 Метод VectorString:pop_back	280
6.135.7 Метод VectorString:front	280
6.135.8 Метод VectorString:back	280
6.135.9 Метод VectorString:__getitem	280
6.135.10 Метод VectorString:__setitem	281
6.136 Таблица DocumentAPI.VectorUInt	281
6.136.1 Метод VectorUInt:size	281
6.136.2 Метод VectorUInt:max_size	282
6.136.3 Метод VectorUInt:empty	282
6.136.4 Метод VectorUInt:clear	282
6.136.5 Метод VectorUInt:push_back	282
6.136.6 Метод VectorUInt:pop_back	282
6.136.7 Метод VectorUInt:front	283
6.136.8 Метод VectorUInt:back	283
6.136.9 Метод VectorUInt:__getitem	283

6.136.10	Метод VectorUInt: __setitem	283
6.137	Таблица DocumentAPI.VerticalAlignment	284
6.138	Таблица DocumentAPI.VerticalTextAnchoredPosition	285
6.139	Таблица DocumentAPI.VerticalRelativeTo	285
6.140	Таблица DocumentAPI.VerticalAnchorAlignment	286
7	Справочник функций EditorAPI	287
7.1	Функция EditorAPI.messageBox	287
7.2	Функция EditorAPI.showPrintDialog	287
7.3	Функция EditorAPI.printDocument	288
7.4	Функция EditorAPI.isPrinterAvailable	288
8	Справочник функций пользовательского интерфейса надстроек	289
8.1	Таблица ui	289
8.1.1	Таблица ui.Dialog	289
8.1.1.1	Открытие диалогового окна	289
8.1.1.2	Конструктор ui:Dialog	290
8.1.1.3	Метод Dialog:setName	291
8.1.1.4	Метод Dialog:getName	291
8.1.1.5	Метод Dialog:setEnabled	291
8.1.1.6	Метод Dialog:isEnabled	291
8.1.1.7	Метод Dialog:setSize	292
8.1.1.8	Метод Dialog:getSize	292
8.1.1.9	Метод Dialog:setOnDone	292
8.1.1.10	Метод Dialog:setButtons	293
8.1.1.11	Метод Dialog:done	294
8.1.2	Таблица ui.DialogButtons	294
8.1.2.1	Конструктор ui:DialogButtons	294
8.1.2.2	Метод DialogButtons:addButton	295
8.1.3	Таблица ui.FolderDialog	296
8.1.3.1	Конструктор ui:FolderDialog	296
8.1.3.2	Метод setTitle	297
8.1.3.3	Метод getTitle	297
8.1.3.4	Метод setInitialDirectory	297
8.1.3.5	Метод getInitialDirectory	297

МойОфис

8.1.3.6	Метод setOnDone	297
8.1.4	Таблица ui.FileOpenDialog	297
8.1.4.1	Конструктор ui:FileOpenDialog	298
8.1.4.2	Метод setTitle	299
8.1.4.3	Метод getTitle	299
8.1.4.4	Метод setInitialDirectory	299
8.1.4.5	Метод getInitialDirectory	299
8.1.4.6	Метод setFilter	299
8.1.4.7	Метод getFilter	299
8.1.4.8	Метод setAllowMultiSelect	299
8.1.4.9	Метод isMultiSelectAllowed	299
8.1.4.10	Метод setOnDone	300
8.1.5	Таблица ui.FileSaveDialog	300
8.1.5.1	Конструктор ui:FileSaveDialog	300
8.1.5.2	Метод setTitle	301
8.1.5.3	Метод getTitle	301
8.1.5.4	Метод setInitialDirectory	301
8.1.5.5	Метод getInitialDirectory	301
8.1.5.6	Метод setFilter	301
8.1.5.7	Метод getFilter	301
8.1.5.8	Метод setOnDone	301
8.1.6	Таблица ui.Label	302
8.1.6.1	Конструктор ui:Label	302
8.1.6.2	Метод Label:setName	302
8.1.6.3	Метод Label:getName	303
8.1.6.4	Метод Label:setEnabled	303
8.1.6.5	Метод Label:isEnabled	303
8.1.6.6	Метод Label:setSize	303
8.1.6.7	Метод Label:getSize	303
8.1.6.8	Метод Label:setText	304
8.1.6.9	Метод Label:getText	304
8.1.6.10	Метод Label:setAligment	304
8.1.6.11	Метод Label:getAligment	304

8.1.6.12	Метод Label:setColor	304
8.1.6.13	Метод Label:getColor	305
8.1.7	Таблица ui.Button	305
8.1.7.1	Конструктор ui:Button	305
8.1.7.2	Метод Button:setName	305
8.1.7.3	Метод Button:getName	306
8.1.7.4	Метод Button:setEnabled	306
8.1.7.5	Метод Button:isEnabled	306
8.1.7.6	Метод Button:setSize	306
8.1.7.7	Метод Button:getSize	306
8.1.7.8	Метод Button:setTitle	307
8.1.7.9	Метод Button:getTitle	307
8.1.7.10	Метод Button:setOnClick	307
8.1.8	Таблица ui.CheckBox	307
8.1.8.1	Конструктор ui:CheckBox	307
8.1.8.2	Метод CheckBox:setName	308
8.1.8.3	Метод CheckBox:getName	308
8.1.8.4	Метод CheckBox:setEnabled	308
8.1.8.5	Метод CheckBox:isEnabled	308
8.1.8.6	Метод CheckBox:setSize	309
8.1.8.7	Метод CheckBox:getSize	309
8.1.8.8	Метод CheckBox:setState	309
8.1.8.9	Метод CheckBox:getState	309
8.1.8.10	Метод CheckBox:isChecked	309
8.1.8.11	Метод CheckBox:setOnStateChanged	309
8.1.9	Таблица ui.RadioButton	310
8.1.9.1	Конструктор ui:RadioButton	310
8.1.9.2	Метод RadioButton:setName	311
8.1.9.3	Метод RadioButton:getName	311
8.1.9.4	Метод RadioButton:setEnabled	311
8.1.9.5	Метод RadioButton:isEnabled	311
8.1.9.6	Метод RadioButton:setState	311
8.1.9.7	Метод RadioButton:getState	311

МойОфис

8.1.9.8	Метод RadioButton:setOnStateChanged	312
8.1.9.9	Метод RadioButton:getSize	312
8.1.9.10	Метод RadioButton:setSize	312
8.1.10	Таблица ui.GroupBox	312
8.1.10.1	Конструктор ui.GroupBox	312
8.1.10.2	Метод GroupBox:setName	313
8.1.10.3	Метод GroupBox:getName	313
8.1.10.4	Метод GroupBox:setEnabled	313
8.1.10.5	Метод GroupBox:isEnabled	313
8.1.10.6	Метод GroupBox:getSize	314
8.1.10.7	Метод GroupBox:setSize	314
8.1.11	Таблица ui.ListBox	314
8.1.11.1	Конструктор ui.ListBox	314
8.1.11.2	Метод ListBox:setName	315
8.1.11.3	Метод ListBox:getName	315
8.1.11.4	Метод ListBox:setEnabled	316
8.1.11.5	Метод ListBox:isEnabled	316
8.1.11.6	Метод ListBox:setSize	316
8.1.11.7	Метод ListBox:getSize	316
8.1.11.8	Метод ListBox:setItems	316
8.1.11.9	Метод ListBox:getItems	317
8.1.11.10	Метод ListBox:addItem	317
8.1.11.11	Метод ListBox:removeItem	317
8.1.11.12	Метод ListBox:removeRow	317
8.1.11.13	Метод ListBox:setCurrentItem	318
8.1.11.14	Метод ListBox:getCurrentItem	318
8.1.11.15	Метод ListBox:setOnCurrentItemChanged	318
8.1.11.16	Метод ListBox:setItemCheckState	318
8.1.11.17	Метод ListBox:setOnItemStateChanged	319
8.1.12	Таблица ui.ListItems	319
8.1.12.1	Метод addItem	319
8.1.13	Таблица ui.ComboBox	320
8.1.13.1	Конструктор ui.ComboBox	320

МойОфис

8.1.13.2	Метод ComboBox:setName	321
8.1.13.3	Метод ComboBox:getName	321
8.1.13.4	Метод ComboBox:setEnabled	322
8.1.13.5	Метод ComboBox:isEnabled	322
8.1.13.6	Метод ComboBox:setSize	322
8.1.13.7	Метод ComboBox:getSize	322
8.1.13.8	Метод ComboBox:setItems	322
8.1.13.9	Метод ComboBox:getItems	323
8.1.13.10	Метод ComboBox:addItem	323
8.1.13.11	Метод ComboBox:removeItem	323
8.1.13.12	Метод ComboBox:removeRow	324
8.1.13.13	Метод ComboBox:setCurrentItem	324
8.1.13.14	Метод ComboBox:getCurrentItem	324
8.1.13.15	Метод ComboBox:setOnCurrentItemChanged	325
8.1.14	Таблица ui.TextBox	325
8.1.14.1	Конструктор ui:TextBox	325
8.1.14.2	Метод TextBox:setName	326
8.1.14.3	Метод TextBox:getName	326
8.1.14.4	Метод TextBox:setEnabled	326
8.1.14.5	Метод TextBox:isEnabled	326
8.1.14.6	Метод TextBox:setSize	327
8.1.14.7	Метод TextBox:getSize	327
8.1.14.8	Метод TextBox:setText	327
8.1.14.9	Метод TextBox:getText	327
8.1.14.10	Метод TextBox:setOnTextChanged	327
8.1.14.11	Метод TextBox:setOnEditingFinished	327
8.1.15	Таблица ui.Row	328
8.1.15.1	Конструктор ui:Row	328
8.1.16	Таблица ui.Column	328
8.1.16.1	Конструктор ui:Column	328
8.1.17	Таблица ui.Spacer	329
8.1.17.1	Конструктор ui:Spacer	329
8.2	Таблица Forms	329

МойОфис

8.2.1	Тип Forms.ItemID	329
8.2.2	Тип Forms.CheckState	329
8.2.3	Тип Forms.Alignment	329
8.2.4	Тип Forms.DialogResult	331
8.2.5	Тип Forms.DialogResult	332
8.2.6	Тип Forms.DialogResultRole	332
8.2.7	Таблица Forms.Size	332
8.2.8	Таблица Forms.ListItem	333
8.2.9	Таблица Forms.ConstListItems	334
8.2.9.1	Метод getCount	334
8.2.9.2	Метод getItem	334
8.2.10	Таблица Forms.Color	334
9	Использование внешнего модуля для работы со строками UTF-8	335
10	Функции для работы со строками в формате Юникод (UTF-8)	336
10.1	Функция utf8.char	336
10.2	Функция utf8.codes	336
10.3	Функция utf8.codepoint	337
10.4	Функция utf8.charpattern	337
10.5	Функция utf8.upper	337
10.6	Функция utf8.lower	338
10.7	Функция utf8.substr	338
10.8	Функция utf8.compare	339
10.9	Функция utf8.islower	339
10.10	Функция utf8.isupper	340
10.11	Функция utf8.isdigit	340
10.12	Функция utf8.isalpha	341
10.13	Функция utf8.len	341
10.14	Функция utf8.offset	341
10.15	Функция utf8.next	342
11	Справочник методов расширения таблицы Regex	343
11.1	Метод create	343
11.2	Метод match	343
11.2.1	Флаги, используемые в Re.match	343

МойОфис

11.3	Метод search	345
11.4	Метод replace	345
11.5	Флаги, используемые при компиляции шаблона	346
11.6	Флаги, используемые для замены	346
12	Класс Regex	348
13	Класс Matches	349
13.1	Метод getFirst	349
13.2	Метод getLength	349
13.3	Метод getSize	349
13.4	Метод getString	349
13.5	Метод _tostring	350

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»
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)

МойОфис

1 Общие сведения

1.1 Назначение

«МойОфис Комплект Средств Разработки (SDK)» – комплект средств для разработчиков, обеспечивающий взаимодействие с приложениями МойОфис, для автоматизации бизнес-процессов компании и расширения возможностей ее прикладных систем.

В состав продукта входят следующие инструменты:

- **Document API** – комплект библиотек с интерфейсами на языках программирования C++, C# и Python для автоматизации создания и редактирования текстовых документов и электронных таблиц во внешних ИТ-системах;
- **Collaboration API** – программный интерфейс интеграции внешних ИТ-решений с системой редактирования и совместной работы МойОфис;
- **Автономный модуль редактирования** – встраиваемое веб-приложение для просмотра, редактирования текстовых документов, электронных таблиц и презентаций, а также просмотра PDF файлов в однопользовательском режиме работы;
- **Сервер совместного редактирования** — интегрируемая серверная система и клиентские веб-приложения для просмотра и совместного редактирования текстовых документов, электронных таблиц и презентаций в прикладных ИТ-системах.
- **Модули надстроек** - внешние устанавливаемые модули на языке Lua, которые позволяют автоматизировать типовые операции и расширить возможности настольных редакторов МойОфис.

Подробное описание возможностей продукта приведено в документе «МойОфис Комплект Средств Разработки (SDK). Функциональные возможности».

Настольные версии редакторов МойОфис Текст и МойОфис Таблица поддерживают механизм для расширения набора доступных пользователю операций за счет подключаемых внешних модулей надстроек. Данное преимущество редакторов МойОфис обеспечивает выполнение узкоспециальных операций, относящихся к определенному виду деятельности или должностным обязанностям, непосредственно в текстовом или табличном документе. Примерами использования Надстроек являются формирование входящего номера в документе, отправка документа по маршруту согласования в системе электронного документооборота, доступ к документам в облачном хранилище.

Разработчик надстроек получает доступ к содержимому открытого документа с помощью функций объектной модели документа. Перечень доступных возможностей включает, но не ограничивается следующими операциями:

- чтение или запись отдельных фрагментов текста, таблиц, ячеек, колонтитулов и т.д.;
- настройка свойств отображения документа;
- работа с текущим выделенным объектом;
- операции с файловой системой, открытие и сохранение документа и отдельных файлов;
- создание и отображение электронных форм ввода данных;
- печать документа;
- поиск и замена фрагмента документа.

1.2 Модули надстроек

Модуль надстройки представляет собой набор команд, с помощью которых производится управление содержимым документов в приложениях (редакторах) ПО МойОфис.

Поддержка модулей надстроек в приложениях ПО МойОфис предоставляет возможность запуска программного кода, разработанного сторонними разработчиками, из командного меню приложений, а также возможность формирования сторонними разработчиками наборов (библиотек) надстроек в различных предметных областях для их коммерческой реализации.

Надстройки позволяют расширить возможности управления содержимым документов в приложениях (редакторах) ПО МойОфис для решения задач высокого уровня сложности.

С помощью надстроек доступно редактирование содержимого документов, форматирование как документа в целом, так и отдельных частей (например, абзацы, таблицы, группы ячеек, отдельные ячейки и т.д.).

Для разработки надстроек для ПО МойОфис используется язык программирования Lua.

Справочное руководство по языку программирования Lua опубликовано по ссылке: <https://lua.org/manual/5.3> (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.

МойОфис

Для управления содержимым документа используется его объектная модель, которая представляет собой совокупность структур данных текстового или табличного документа ПО МойОфис.

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

Для создания надстройки необходимо сформировать файлы модуля надстройки с помощью текстового редактора (структура и состав файлов должны соответствовать описанию, приведенному в разделе [Состав и структура надстройки](#)).

Перед использованием надстройки ее необходимо установить в соответствии с описанием, приведенным в разделе [Установка надстройки](#) или разделе [Установка и обновление надстроек](#).

1.3 Уровень подготовки пользователя

Для разработки надстройки пользователь должен иметь опыт разработки приложений на языке программирования Lua под управлением ОС Microsoft Windows или ОС Linux.

Также необходим навык работы со стандартными офисными приложениями.

Полный список требований приведен в документе «МойОфис комплект средств разработки (SDK). Руководство программиста».

1.4 Системные требования

Для разработки и запуска надстроек должны учитываться условия, описанные в документе «МойОфис комплект средств разработки (SDK). Системные требования».

На его основании разработчик проверяет соответствие требований к программному и аппаратному обеспечению для обеспечения работоспособности модуля надстройки.

Для запуска программного кода модуля надстройки необходима установленная настольная версия приложений «МойОфис Текст» или «МойОфис Таблица».

2 Надстройки

2.1 Состав и структура надстройки

Надстройка представляет собой **zip**-архив с расширением **.mox**, в котором находятся LUA - скрипты и другие вспомогательные файлы.

Архивный файл надстройки содержит следующие папки и файлы:

- [файл регистрации надстройки](#) в корневом каталоге надстройки;
- [файл сценария надстройки](#) в каталоге **cmd**;
- [файл с текстом лицензионного соглашения](#) (End-user License Agreement, EULA) в каталоге **META-INF**.

В случае, если надстройка подписана, в каталоге **META-INF** находятся файлы электронной подписи:

- **.digest** - хэш-суммы файлов надстройки;
- **.signature** - файл ЭЦП.

Для неподписанных надстроек файлы электронной подписи отсутствуют.

В составе модуля надстройки могут содержаться следующие специальные каталоги:

- **bin**, в котором размещаются библиотеки и иные сторонние модули (см. раздел [Распространение сторонних модулей в составе надстройки](#));
- **i18n**, в котором размещаются файлы, необходимые для локализации надстройки (см. раздел [Словари локализации](#)).

Также архивный файл надстройки **MOX** может содержать любые другие файлы и каталоги, необходимые для обеспечения работы надстройки.

Пример структуры файла надстройки приведен на рисунке 1.

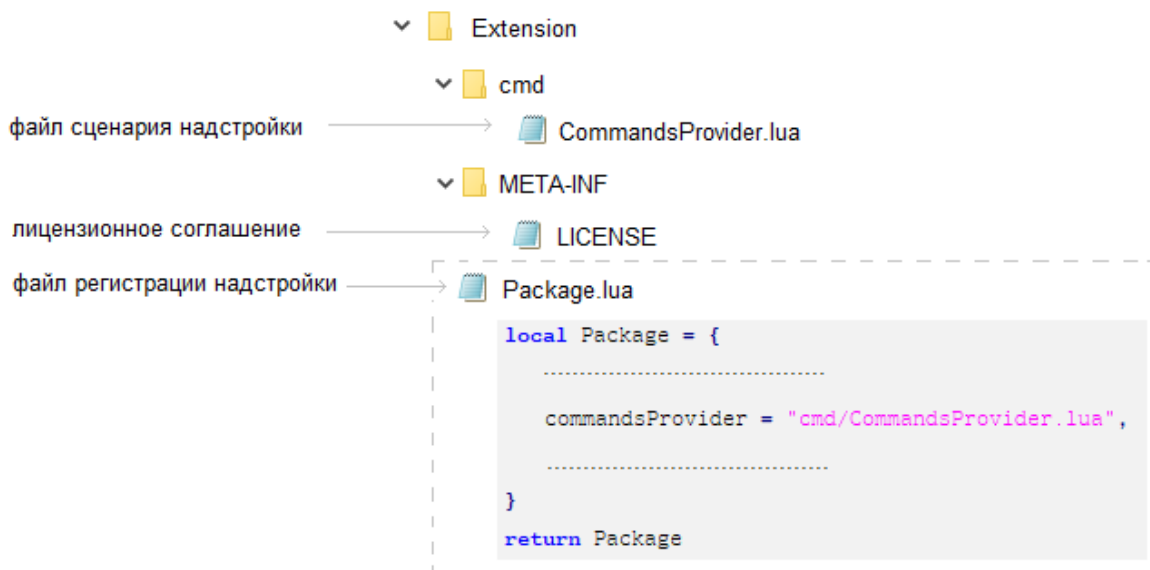


Рисунок 1 – Содержимое файла надстройки

При установке архив распаковывается в папку, которая зависит от используемой операционной системы:

```
Linux: ~/.local/share/New Cloud Technologies  
Ltd./MyOffice/<Application>/Extensions/<Extension ID>  
Windows: %LOCALAPPDATA%\New Cloud Technologies  
Ltd\MyOffice\<Application>\Extensions\<Extension ID>  
macOS: ~/Library/Application Support/New Cloud Technologies  
Ltd./MyOffice/<Application>/Extensions/<Extension ID>
```

Для доступа к файлам в **домашнем** каталоге надстройки используются функции, перечисленные в разделе [Функции для работы с файлами надстройки](#).

2.1.1 Файл регистрации надстройки

В составе надстройки должен присутствовать файл регистрации (манифест), содержащий пары ключ-значение для конфигурации надстройки в приложении редакторов МойОфис.

Файл регистрации имеет обязательное имя **Package.lua**. Часть ключей должна быть определена в обязательном порядке, в противном случае надстройка не будет зарегистрирована. Ключи могут быть перечислены в произвольном порядке. Поддерживаются ключи, приведенные в таблице 2.

Таблица 2 - Поддерживаемые ключи конфигурации надстройки

Ключ	Обязательный	Описание
extensionID	Да	<p>Уникальный идентификатор надстройки. Значение ключа указывается в нотации обратного доменного имени.</p> <p>Пример:</p> <pre>extensionID = "com.example.entryform".</pre>
extensionName	Да	<p>Название надстройки. Отображается в меню Надстройки командного меню приложения в том случае, если надстройка установлена и включена (см. Рисунок 2 и раздел Установка и управление надстройками в режиме диалога).</p> <p>Пример:</p> <pre>extensionName = "Учетная карточка сотрудника".</pre>
vendor	Да	<p>Автор надстройки. Значение ключа отображается в диалоговом окне информации о надстройке.</p> <p>Пример:</p> <pre>vendor = "\"Мой Офис\"".</pre>
extensionVersion	Да	<p>Версия надстройки в виде таблицы с полями:</p> <ul style="list-style-type: none"> – major (число); – minor (число); – patch (число); – build (строка). <p>Значение ключа отображается в диалоговом окне информации о надстройке.</p> <p>Пример:</p> <pre>extensionVersion = { major = 0, minor = 1, patch = 0, build = "" }.</pre>
description	Нет	<p>Краткое описание надстройки. Значение ключа отображается в диалоговом окне информации о надстройке.</p> <p>Пример:</p> <pre>description = "Учетная карточка научного сотрудника".</pre>

Ключ	Обязательный	Описание
applicationId	Да	<p>Идентификатор приложения, с которым может работать надстройка:</p> <ul style="list-style-type: none"> – только «МойОфисТаблица»: "MyOffice Spreadsheet"; – только «МойОфисТекст»: "MyOffice Text"; – одновременно в «МойОфисТаблица» и «МойОфисТекст»: {"MyOffice Spreadsheet", "MyOffice Text"}.
apiVersion	Да	<p>Актуальный номер версии API редактора для работы с надстройками в виде таблицы с полями:</p> <ul style="list-style-type: none"> – major (число); – minor (число). <p>Следует использовать значение 1.0</p> <p>Пример:</p> <pre>apiVersion = { major = 1, minor = 0 }</pre>
commandsProvider	Да	<p>Наименование файла сценария надстройки. Допустимо указание пути относительно домашнего каталога надстройки. Недопустимо указание полного пути к файлу сценария надстройки.</p> <p>Пример:</p> <pre>commandsProvider = "cmd/entryform.lua"</pre> <p>Файл сценария под названием entryform.lua расположен в каталоге cmd домашней папки надстройки.</p>
onLoad	Нет	<p>Наименование файла сценария, вызываемого при запуске надстройки. Допустимо указание пути относительно домашнего каталога надстройки.</p> <p>Пример:</p> <pre>onLoad = "cmd/entryform-start.lua"</pre> <p>При запуске надстройки сценарий, находящийся в файле entryform-start.lua в каталоге cmd домашней папки надстройки.</p>

Ключ	Обязательный	Описание
onUnLoad	Нет	<p>Наименование файла сценария, вызываемого перед завершением работы надстройки. Допустимо указание пути относительно домашнего каталога надстройки.</p> <p>Пример:</p> <pre>onUnload = "cmd/entryform-stop.lua"</pre> <p>При запуске надстройки выполнится сценарий, находящийся в файле entryform-stop.lua, в каталоге cmd домашней папки надстройки.</p>
fallbackLanguage	Нет	<p>Код языка (например, 'ru', 'en') который будет использован по умолчанию, если надстройка не содержит словаря для языка, установленного в настройках ОС. Подробнее о возможностях локализации см. раздел Локализация надстроек</p> <p>Пример:</p> <pre>fallbackLanguage = 'RU'.</pre>

Пример файла **Package.lua**:

```
local Package = {
    vendor = "ООО \"Новые Облачные Технологии\"",
    description = "Учетная карточка научного сотрудника",
    extensionID = "com.example.entryform",
    extensionName = "Учетная карточка сотрудника",
    extensionVersion = { major = 0, minor = 1, patch = 0, build = "" },
    applicationId = "MyOffice Spreadsheet",
    apiVersion = { major = 1, minor = 0 },
    commandsProvider = "cmd/entryform.lua",
    fallbackLanguage = 'RU',
    onLoad = "cmd/entryform-start.lua",
    onUnload = "cmd/entryform-stop.lua"
}
return Package
```

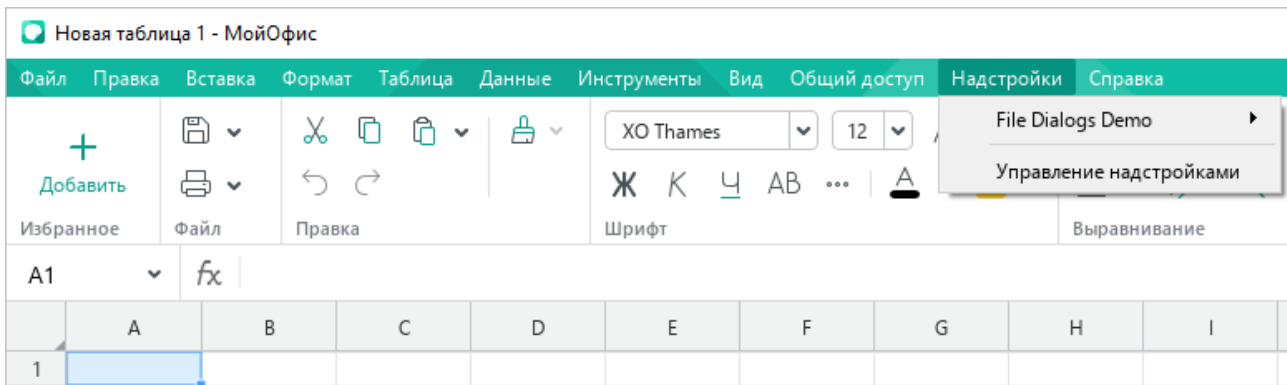


Рисунок 2 – Отображение меню **Надстройки** со списком надстроек в командном меню приложения «МойОфис Таблица»

Ключи `onLoad` и `onUnload` позволяют указать сценарии, выполняемые во время запуска или завершения работы надстройки. Эти сценарии могут быть использованы для подготовки ресурсов перед началом работы надстройки и освобождения ресурсов после выгрузки надстройки. Во время запуска и завершения работы надстройки не рекомендуется использование вызовов, взаимодействующих с пользовательским интерфейсом, использования метода `messageBox` или работы с формами ввода данных.

Сценарии для ключей `onLoad` и `onUnload` могут быть размещены в одном файле или в отдельных файлах. Сценарии должны содержать реализации методов `onLoadExtension()` и `onUnloadExtension()`, например:

```
local Actions = {}  
function Actions.onLoadExtension()  
-- Подготовка ресурсов для надстройки  
end  
function Actions.onUnloadExtension()  
-- Освобождение ресурсов надстройки  
end  
return Actions
```

2.1.2 Файл сценария надстройки

Наименование и расположение файла сценария надстройки задается с помощью ключа `commandsProvider` в файле **Package.lua** (см. таблицу 2).

С помощью функции `getCommands` разработчик определяет команды для управления надстройкой, доступные пользователю через командное меню **Надстройки**.

Пример:

```
function Actions.getCommands()  
    return {  
        {  
            -- Идентификатор команды меню  
            id = "EntryForm.addRecord",  
            -- Наименование команды в меню:  
            -- "Надстройки" - "Учетная карточка сотрудника" - "Новая запись"  
            menuItem = "Новая запись",  
            -- Наименование функции-обработчика  
            command = Actions.addRecord  
        }  
    }  
end
```

Описание полей таблицы регистрации команд приведено в таблице 3.

Таблица 3 - Описание полей таблицы регистрации команд надстройки

Параметр	Тип	Значение
id	Строка	Идентификатор команды.
menuItem	Строка	Наименование команды надстройки. Наименования команд надстройки отображаются в меню Надстройки командного меню приложений ПО МойОфис (см. Рисунок 2). При выборе команды надстройки происходит вызов обработчика, указанного в поле command.
command	Строка	Имя функции обработчика события.

Описание функции обработчика события имеет следующий вид:

```
local Actions = {}  
function Actions.getCommands()  
    return {  
        {  
            id = "EntryForm.addRecord",  
            menuItem = "Новая запись",  
            command = Actions.addRecord  
        }  
    }  
end  
  
function Actions.addRecord(context)
```

```
context.doWithDocument(function(document)
    -- TODO New employee
end)
end
return Actions
```

Параметр `context` функции обработчика события определяет уровень, на котором функция обработчика события взаимодействует с приложением редактора.

Возможны следующие уровни взаимодействия:

- на уровне приложения;
- на уровне открытого документа;
- на уровне выделенного фрагмента (`selection`) в открытом документе.

Разработчик должен вызвать один из методов контекста, передавая ему в качестве параметра анонимную функцию для фактической обработки события. Аргументом анонимной функции является объект, представляющий приложение, документ или выделенный фрагмент документа.

2.1.2.1 Метод `context.doWithApplication`

Метод `context.doWithApplication` позволяет выполнить действие над приложением редактора, объект которого передается как аргумент анонимной функции. Объект приложения (`application`) также позволяет получить доступ к открытому документу.

Пример:

```
local Actions = {}
function Actions.createDoc(context)
    context.doWithApplication(function(application)
        local d = application:createDocument(DocumentAPI.DocumentType_Workbook)
        d:saveAs( "c:\\NewDocument.xlsx" )
    end)
end
return Actions
```

Дополнительные методы, доступные на уровне приложения, описаны в таблице 4.

Таблица 4 - Методы `context.doWithApplication`

Метод	Описание
application.createDocument	Создает новый текстовый или табличный документ с типом, указанным в параметрах. Используются настройки по умолчанию или явно указанные в параметре DocumentSettings . Объект документа создается в памяти, дополнительный экземпляр приложения редактора не запускается.
application.loadDocument	Загружает существующий текстовый или табличный документ из файла, находящегося по указанному пути. Формат и тип документа определяются из расширения файла, если они не указаны явно с помощью параметра DocumentSettings . Объект документа создается в памяти, дополнительный экземпляр приложения редактора не запускается.

2.1.2.2 Метод `context.doWithDocument`

Метод `context.doWithDocument` позволяет выполнить действие над открытым документом. Объект документа передается как аргумент анонимной функции. Необходимо использовать методы `getBlocks`, `getRange` для доступа к отдельным объектам документа, таким как абзацы, таблицы, колонтитулы и т.д.

Пример:

```
local Actions = {}  
function Actions.copyDocument(context)  
  context.doWithDocument(function(document)  
    -- Копия открытого документа создается в файле c:\CopyDocument.xlsx  
    document:saveAs( "c:\\CopyDocument.xlsx" )  
  end)  
end  
return Actions
```


Дополнительные методы, доступные на уровне документа, описаны в таблице 5.

Таблица 5 - Методы `context.doWithDocument`

Метод	Описание
document.saveAs	Сохраняет документ в файл по указанному пути. Формат и тип документа определяются расширением файла, если не указаны в явном виде.
document.exportAs	Экспортирует документ в файл по указанному пути и с указанным форматом.
document.merge	Возвращает документ, в котором различия отмечены отслеживаемыми изменениями.

2.1.2.3 Метод `context.doWithSelection`

Метод позволяет выполнить действие над выделенным фрагментом документа. Фрагмент документа может быть передан как аргумент метода в виде:

- таблицы [DocumentAPI.Range](#) для текстового редактора;
- таблицы [DocumentAPI.CellRange](#) для табличного редактора.

Пример:

```
local Actions = {}
function Actions.printCell(context)
    context.doWithSelection(function(range)
        for cell in range:enumerate() do
            EditorAPI.messageBox(cell:getFormattedValue())
        end
    end)
end
return Actions
```

2.1.3 Файл лицензионного соглашения

Файл лицензионного соглашения содержит текст лицензионного соглашения (End User License Agreement, EULA) между конечным пользователем и владельцем компьютерной программы, в данном случае - надстройки. Лицензионное соглашение устанавливает условия использования надстройки.

Соглашение между конечным пользователем и владельцем надстройки является частью процесса регистрации надстройки для работы с ПО МойОфис. Во время регистрации

МойОфис

надстройки менеджер надстроек автоматически выполняет поиск файла с текстом лицензионного соглашения в составе надстройки. Текст лицензионного соглашения отображается в диалоговом окне, давая возможность конечному пользователю принять или отклонить условия соглашения. Для регистрации надстройки пользователь должен принять условия лицензионного соглашения. Если пользователь отклонил условия соглашения, то надстройка не будет зарегистрирована для работы с ПО МойОфис.

Файл лицензионного соглашения обязательно должен присутствовать в составе надстройки. Владелец надстройки имеет возможность представить несколько вариантов текста лицензионного соглашения на разных языках. Выбор варианта основывается на текущих установках локализации ОС и на значении ключа `fallbackLanguage` в файле **Package.lua** (см. раздел [Файл регистрации надстройки](#)).

Файлы с текстами лицензионных соглашений на разных языках размещаются в подкаталоге **META-INF** домашнего каталога надстройки. В составе надстройки может присутствовать одновременно несколько файлов лицензионных соглашений на разных языках.

Текст лицензионного соглашения хранится в текстовом файле в кодировке UTF-8. Файл лицензионного соглашения должен иметь наименование, соответствующее шаблону:

LICENSE_<Код языка>, где **<Код языка>** – двухбуквенный суффикс кодировки языка в соответствии с ISO 639-1, например:

- **LICENSE_RU** – файл содержит текст лицензионного соглашения на русском языке;
- **LICENSE_FR** – файл содержит текст лицензионного соглашения на французском языке.

Приложение редактора текста или таблиц запрашивает у операционной системы настройки локализации при старте приложения. Код языка передается в менеджер надстроек, который выполняет поиск текста лицензионного соглашения и текстов строк пользовательского соглашения на соответствующем языке.

Поддерживаемые коды языка перечислены в таблице 6.

Таблица 6 - Поддерживаемые коды языка лицензионного соглашения

Код языка	Расшифровка
RU	Русский язык
EN	Английский язык, включая варианты <code>en_us</code> и <code>en_gb</code>

Код языка	Расшифровка
FR	Французский язык
ES	Латиноамериканский испанский
PT	Бразильский португальский
DE	Немецкий язык
IT	Итальянский язык
TT	Татарский язык
BA	Башкирский язык
BE	Белорусский язык
KK	Казахский язык

Для хранения текста лицензионного соглашения допустимо использовать файл **LICENSE** без указания кода языка в том случае, если текст лицензионного соглашения представлен только на одном языке.

Поведение менеджера надстроек при поиске текста лицензионного соглашения на примере французского языка:

1. Использовать файл **LICENSE_FR**.
2. Если файл **LICENSE_FR** отсутствует, использовать значение ключа `fallbackLanguage` для получения кода языка.
3. Если значение ключа `fallbackLanguage` не задано, использовать **EN** в качестве кода языка.
4. Если файл **LICENSE_EN** отсутствует, использовать файл **LICENSE** (без указания кода языка).
5. Если файл **LICENSE** отсутствует, сообщить об ошибке.

2.1.4 Локализация надстроек

При создании надстройки разработчик может предусмотреть возможность отображения на экране текстовой информации о надстройке на языке, соответствующем используемому в приложении МойОфис.

Для обеспечения этой возможности в состав приложения включена библиотека Lua **i18n** (<https://github.com/kikito/i18n.lua>), которая содержит метод `currentLocale()`,

позволяющий из надстройки получить информацию о текущей локализации. Библиотека также настраивает среду для использования в надстройке подходящего словаря.

2.1.4.1 Локализация полей таблицы регистрации надстройки и таблицы сценария надстройки

Следующие ключи в файле регистрации надстройки **Package.lua**, могут быть локализованы (см. таблицу 2):

- `extensionName` – название надстройки;
- `vendor` – автор надстройки;
- `description` – описание надстройки.

Ключ `fallbackLanguage` в файле регистрации **Package.lua** (см. таблицу 2), содержит код языка, который используется, если в файле надстройки отсутствует словарь, соответствующий текущему языку системы.

В файле сценария надстройки наименование команды, отображаемой в меню (`menuItem`), также может быть локализовано (см. таблицу 3).

Значение поля `menuItem` считается идентификатором в словаре надстройки. Если в словаре надстройки такой идентификатор отсутствует, то значение `menuItem` отображается как текст.

Для перевода названий пунктов меню используются функции библиотеки **i18n.lua**, однако предпочтительнее использовать словарь для соответствующего языка, который должен быть размещен в папке **i18n** надстройки.

Пример файла **CommandsProvider**:

```
function CommandsProvider.getCommands()  
    return {  
        {  
            id = "goodBye1",  
            menuItem = 'good_bye', -- prefer this way  
            command = CommandsProvider.insertTable1x1  
        },  
        {  
            id = "goodBye3",  
            menuItem = i18n('good_bye'), -- allowed, but not recommended  
            command = CommandsProvider.insertTable1x1  
        }  
    }  
end
```

2.1.4.2 Словари локализации

Словари локализации предназначены для перевода пользовательского интерфейса надстройки на различные языки. Файл словаря расположен в каталоге **il8n** домашнего каталога надстройки, и представляет собой код на языке программирования Lua. Название файла соответствует коду языка, который он предоставляет, например, **ru.lua** – словарь для русского языка.

Пример словаря для немецкой локализации (имя файла – **de.lua**):

```
return {
  de = {
    description = "Testerweiterung für Lokalisierung",
    app_name = "Lokalisierungstest",
    good_bye = "Auf Wiedersehen!",
    age_msg = "Ihr Alter beträgt %{age}.",
    phone_msg = {
      one = "Sie haben eine neue Nachricht.",
      other = "Sie haben %{count} neue Nachrichten."
    }
  }
}
```

При запуске надстройки выполняется автоматический поиск подходящего словаря по следующей схеме:

- если каталог **il8n** содержит словарь, который соответствует текущему языку операционной системы, он загружается в качестве основного словаря;
- если каталог **il8n** не содержит словарь, соответствующий текущему языку операционной системы, то загружается словарь языка, код которого соответствует значению ключа **fallbackLanguage** в файле регистрации надстройки **Package.lua** (см. таблицу 2);
- если значение ключа **fallbackLanguage** не задано, либо отсутствует словарь для данного кода языка, то загружается словарь для кода языка «en»;
- если словарь для кода языка «en» не найден, то расширение не загружается.

Словари должны содержать все идентификаторы, используемые в файле регистрации надстройки или в файлах сценариев надстройки. При отсутствии идентификатора возникает ошибка выполнения сценария.

Пример:

```
local TestMessages = {}
function TestMessages.printMsg()
    print("Current locale: " .. getCurrentLocale())
    print("Chosen locale: " .. i18n.getLocale())
    print("Fallback locale: " .. i18n.getFallbackLocale())
    print(i18n('good_bye'))
    print(i18n('age_msg', {age = 81}))
    print(i18n('phone_msg', {count = 1}))
    print(i18n('phone_msg', {count = 9}))
end
return TestMessages
```

2.2 Подпись надстройки

Для полноценного использования надстройка должна быть подписана с использованием сертификата. Использование неподписанных надстроек не рекомендуется.

Для подписи надстройки используется **Сертификат разработчика**, который поставляется службой поддержки МойОфис.

Утилита подписи надстроек МОХ

Утилита командной строки для подписи надстроек входит в состав дистрибутива редакторов и называется **mox**.

Утилита **mox** вызывается с одной из следующих команд: **create**, **sign**, **deletesignature**.

Создание надстройки

```
mox create --source=${plugin directory} --package=${package name}
```

Параметры:

- s [--source] путь к директории, содержащему надстройку
- p [--package] имя надстройки
- d [--delete] удалить существующую надстройку

Создает надстройку из содержимого директории, заданного параметром **--source**. Если параметр **--package** отсутствует, в качестве имени надстройки будет использовано имя директории. Если файл надстройки уже существует команда выполнится с ошибкой. Для того чтобы этого избежать, используйте параметр **--delete** для предварительного удаления предыдущего файла. Файл надстройки всегда создается с расширением **'.mox'**. Если

МойОфис

надстройка подписана, будет произведена проверка подписи. Если проверка не пройдена, надстройка не будет создана.

Подпись надстройки

```
mox sign --package=${package} --certificate=${certificate path} --private-key=${private key path} --passphrase=${passphrase}
```

Параметры:

- p [--package] путь к папке, содержащей надстройку
- c [--certificate] путь к сертификату
- k [--private-key] путь к закрытому ключу
- s [--passphrase] секретный пароль для расшифровки закрытого ключа

Подписывает надстройку. Имя надстройки или папки задается в параметре --package. Закрытый ключ - необязательный параметр. Если он защищен паролем, пароль должен быть указан в параметре --passphrase.

Удаление подписи

```
mox deletesignature --package=${package}
```

Параметры:

- p [--package] путь к директории, содержащему надстройку

Удаляет подпись из надстройки. Имя надстройки задается в параметре --package.

Подпись надстроек для OS Windows

В OS Windows исполняемый файл утилиты `mox.exe` находится в директории установки МойОфис, например, `d:\Program Files\MyOffice\mox.exe`.

Пример создания надстройки:

```
mox.exe create --source=d:\Work\SampleExtension --package=SampleExtension
```

```
INFO: A digest of the package is successfully verified  
INFO: The package file 'SampleExtension.mox' is created
```

Пример подписи надстройки:

```
mox.exe sign --package=d:\Work\SampleExtension\SampleExtension.mox --private-key=d:\Work\SampleExtension\certificate.key --certificate=d:\Work\SampleExtension\certificate.crt
```

```
INFO: Certificate info:
```

```
S/N: XX:XX:XX:XX:XX:XX:XX:XX
Valid to: Fri Apr 22 12:00:00 2025
Common name: MyOffice Developer Cert
Email: extension@developer.com
Organization: MyOffice
Issuer common name: MyOffice Plugin Validation CA
INFO: The package 'd:\Work\SampleExtension\SampleExtension.mox' is
successfully signed
```

Пример удаления подписи надстройки:

```
mox.exe deletesignature --package=d:
\Work\SampleExtension\SampleExtension.mox

INFO: The file 'META-INF/.digest' is successfully removed.
INFO: The file 'META-INF/.signature' is successfully removed.
```

Подпись надстроек для OS Linux

В OS Linux исполняемый файл утилиты mox находится в директории установки МойОфис.

Пример создания надстройки:

```
$ build/mox create --source="./tmp" --package="SampleExtension.mox"

WARNING: the created package is not signed
INFO: The package file 'SampleExtension.mox' is created
```

Пример подписи надстройки:

```
$ build/mox sign --package='./tmp' --private-key='sign.key' --
certificate='sign.crt'

INFO: Certificate info:
S/N: XX:XX:XX:XX:XX:XX:XX:XX
Valid to: Fri Apr 22 12:00:00 2025
Common name: MyOffice Developer Cert
Email: extension@developer.com
Organization: MyOffice
Issuer common name: MyOffice Plugin Validation CA
INFO: The package './tmp' is successfully signed
```


Пример удаления подписи надстройки:

```
$ build/mox deletesignature --package='./tmp'
```

```
INFO: The file 'META-INF/.digest' is successfully removed.
```

```
INFO: The file 'META-INF/.signature' is successfully removed.
```

2.3 Подготовка сторонних модулей для использования в составе надстройки

Надстройка может использовать сторонние модули на языке программирования Lua, например, из репозитория LuaRocks (luarocks.org). Такие модули должны быть совместимы с версией Lua 5.3.2, используемой в ПО МойОфис для исполнения макрокоманд и надстроек. В данном разделе описывается процесс подготовки окружения, сборки и распространения сторонних модулей, совместимых с Lua 5.3.2, для использования в составе надстройки.

При переходе на ресурс LuaRocks на экране возникает основной экран, содержащий строку поиска (см. рисунок 3).

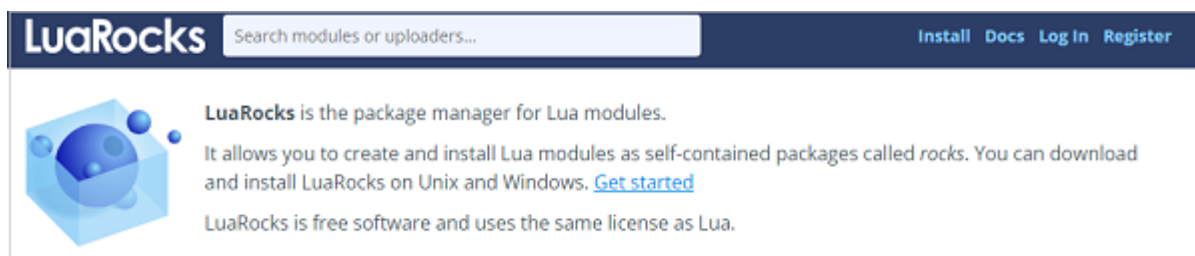


Рисунок 3 - Основной экран LuaRocks

Поиск выдает список библиотек, название которых удовлетворяет запросу (см. рисунок 4).

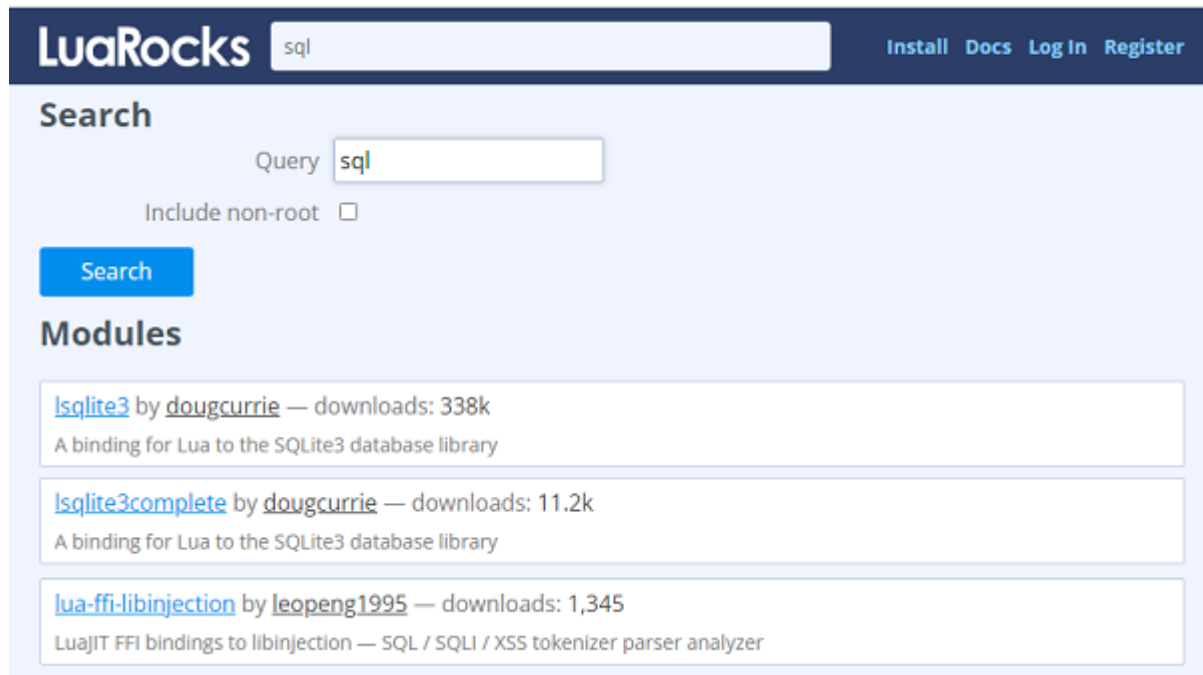


Рисунок 4 - Поиск библиотек LuaRocks

При переходе по ссылке на страницу библиотеки открывается экран, содержащий описание, исходный код, подробности установки и использования. В качестве примера далее будет описана работа с библиотекой `sqlite3complete`, поддерживающая взаимодействие с базами данных.



Разработчик надстройки МойОфис несет ответственность за соблюдение лицензионных соглашений и ограничений, возникающих при использовании и распространении сторонних модулей.

2.3.1 Сборка внешних модулей для ОС Microsoft Windows

Для сборки стороннего модуля под ОС Microsoft Windows требуется установить компилятор языка C++ **mingw**.

Для обеспечения совместимости с МойОфис Lua Extension API необходимо выполнить сборку стороннего модуля с использованием библиотек из комплекта поставки «МойОфис Стандартный» (начиная с релиза 2020.02). Комплект библиотек находится в инсталляционном каталоге МойОфис, по умолчанию в папке: `C:\Program Files\MyOffice\Lua532`.

Состав поставляемых файлов:

- `Lua532\bin` – содержит компилятор, интерпретатор и библиотеку для Lua версии 5.3.2;

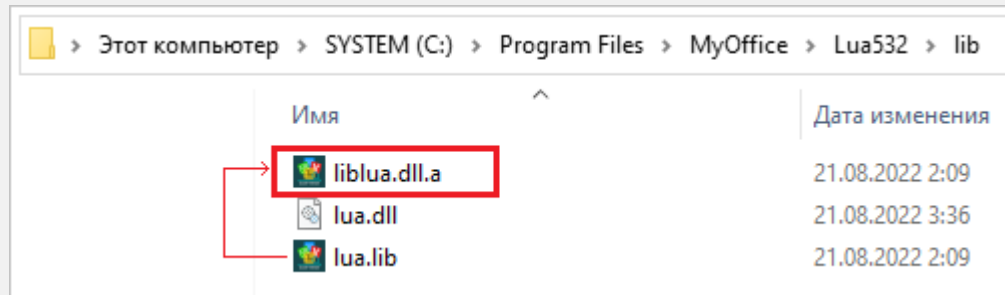
МойОфис

- Lua532\include – содержит комплект заголовочных файлов Lua для пересборки сторонних модулей;
- Lua532\lib – содержит комплект библиотек Lua для сборки сторонних модулей.

Фактически инсталляционный каталог МойОфис может отличаться от выбранного по умолчанию.

Путь к каталогам заголовочных файлов и библиотечных файлов должен быть включен в соответствующих ключах компилятора, либо в конфигурационных настройках среды разработки.

Для корректной сборки надстроек в версии 2.8 необходимо сделать копию файла `\MyOffice\Lua532\lib\lua.lib` и переименовать ее в `\MyOffice\Lua532\lib\liblua.dll.a`



2.3.1.1 Установка MinGW

Для установки **MinGW** необходимо скачать последнюю версию по ссылке: winlibs.com.

Необходимый вариант дистрибутива:

GCC X.X.X (with **POSIX** threads) + LLVM/Clang/LLD/LLDB X.X.X + MinGW-w64 XX.X.X (UCRT)

Например:

Release versions

UCRT runtime

- GCC 13.2.0 (with **POSIX** threads) + LLVM/Clang/LLD/LLDB 17.0.6 + MinGW-w64 11.0.1 (UCRT) - release 5 (**LATEST**)
 - Win32: [7-Zip archive*](#) | [Zip archive](#) - without LLVM/Clang/LLD/LLDB: [7-Zip archive*](#) | [Zip archive](#)
 - Win64: [7-Zip archive*](#) | [Zip archive](#) - without LLVM/Clang/LLD/LLDB: [7-Zip archive*](#) | [Zip archive](#)
- GCC 13.2.0 (with **MCF** threads) + MinGW-w64 11.0.1 (UCRT) - release 3 (**LATEST**)
 - Win32 (without LLVM/Clang/LLD/LLDB): [7-Zip archive*](#) | [Zip archive](#)
 - Win64 (without LLVM/Clang/LLD/LLDB): [7-Zip archive*](#) | [Zip archive](#)

МойОфис

Скачанный архив содержит единственную папку **mingw64**, ее нужно распаковать в рабочую, например, в C:\Work:

```
C:\
└─ Work
    └─ mingw64
        ├── bin
        ├── build-info.txt
        ├── etc
        ├── include
        ├── lib
        ├── libexec
        ├── licenses
        ├── opt
        ├── share
        └─ x86_64-w64-mingw32
```

В настройках переменных среды PATH необходимо добавить пути для каталогов mingw64\bin и mingw64\x86_64-w64-mingw32\lib.

- C:\Work\mingw64\bin
- C:\Work\mingw64\x86_64-w64-mingw32\lib

2.3.1.2 Установка и сборка LuaRocks

Далее необходимо скачать и собрать инструмент LuaRocks, предназначенный для поддержки репозитория библиотек Lua.

Дистрибутив для скачивания: <http://luarocks.github.io/luarocks/releases/luarocks-3.8.0-win32.zip>, следует развернуть в любую временную папку, например, c:\Work.

```
C:\
└─ Work
    └─ luarocks-3.8.0-win32
        ├── .....
        ├── .....
        ├── .....
        └─ .....
```



Для продолжения сборки LuaRocks необходимо убедиться в том, что в папке ...\\Lua532\\lib\ создана копия файла lua.lib с новым именем liblua.dll.a (как это было описано в разделе [Сборка внешних модулей для ОС Microsoft Windows](#))

Далее для сборки LuaRocks следует запустить консоль CMD и выполнить следующую команду в папке c:\Work\luarocks-3.8.0-win32 :

```
install /F /NOADMIN /P C:\Work\LuaRocks /TREE C:\Work\LuaRocks\5.3 /LUA "C:\Program Files\MyOffice\Lua532" /MW /Q
```

Где используются следующие ключи:

/F - очистить предыдущую установку;

/P C:\Work\LuaRocks - папка в которую установится LuaRocks;

/TREE C:\Work\LuaRocks\5.3 - папка в которую будут устанавливаться библиотеки;

/LUA "C:\Program Files\MyOffice\Lua532" - путь до интерпретатора LUA;

/MW - используем MinGW;

/Q - не запрашивать подтверждение установки.

Вывод в случае успешной установки:

```
C:\Work\luarocks-3.8.0-win32>install /F /NOADMIN /P C:\Work\LuaRocks /TREE
C:\Work\LuaRocks\5.3 /LUA "C:\Program Files\MyOffice\Lua532" /MW
C:\Work\luarocks-3.8.0-win32>rem=rem --[[--lua
LuaRocks 3.8.x installer.

=====
== Checking system... ==
=====

Attempting to install without admin privileges...
Looking for Lua interpreter
checking C:\Program Files\MyOffice\Lua532
Found lua.exe, testing it...
Interpreter found, now looking for link libraries...
checking for C:\Program Files\MyOffice\Lua532\lua5.3.lib
checking for C:\Program Files\MyOffice\Lua532\lua53.lib
checking for C:\Program Files\MyOffice\Lua532\lua5.3.dll
checking for C:\Program Files\MyOffice\Lua532\lua53.dll
checking for C:\Program Files\MyOffice\Lua532\liblua.dll.a
checking for C:\Program Files\MyOffice\Lua532\lib\lua5.3.lib
checking for C:\Program Files\MyOffice\Lua532\lib\lua53.lib
checking for C:\Program Files\MyOffice\Lua532\lib\lua5.3.dll
checking for C:\Program Files\MyOffice\Lua532\lib\lua53.dll
checking for C:\Program Files\MyOffice\Lua532\lib\liblua.dll.a
Found liblua.dll.a
Link library found, now looking for headers...
checking for C:\Program Files\MyOffice\Lua532\include\lua\5.3\lua.h
checking for C:\Program Files\MyOffice\Lua532\include\lua53\lua.h
checking for C:\Program Files\MyOffice\Lua532\include\lua5.3\lua.h
checking for C:\Program Files\MyOffice\Lua532\include\lua.h
Found lua.h
```

```
Headers found, checking runtime to use...
  C:\Program Files\MyOffice\Lua532\bin\lua.exe uses VCRUNTIME140.DLL as
runtime
Runtime check completed.
arch: 8664 -> IMAGE_FILE_MACHINE_AMD64

=====
== System check results ==
=====

Will configure LuaRocks with the following paths:
LuaRocks      : C:Work\LuaRocks
Config file   : C:Work\LuaRocks\config-5.3.lua
Rocktree     : C:Work\LuaRocks\5.3

Lua interpreter : C:\Program Files\MyOffice\Lua532\bin\lua.exe
  binaries      : C:\Program Files\MyOffice\Lua532\bin
  libraries     : C:\Program Files\MyOffice\Lua532\lib
  includes     : C:\Program Files\MyOffice\Lua532\include
  architecture: x86_64
  binary link  : liblua.dll.a with runtime VCRUNTIME140.dll

Compiler      : MinGW/gcc (make sure it is in your path before using LuaRocks)
               in: C:\mingw64\bin

=====
== Installing LuaRocks... ==
=====

Installing LuaRocks in C:Work\LuaRocks...
Created LuaRocks command: C:Work\LuaRocks\luarocks.bat
Created LuaRocks command: C:Work\LuaRocks\luarocks-admin.bat

Configuring LuaRocks...
Created LuaRocks hardcoded settings file:
C:Work\LuaRocks\lua\luarocks\core\hardcoded.lua
Created LuaRocks config file: C:Work\LuaRocks\config-5.3.lua

Creating rocktrees...
Created system rocktree      : "C:Work\LuaRocks\5.3"
Local user rocktree exists  : "C:\Users\Admin\AppData\Roaming\LuaRocks"
```

```
Loading registry information for ".rockspec" files

=====
== LuaRocks is installed! ==
=====

You may want to add the following elements to your paths;
Lua interpreter;
  PATH      :    C:\Program Files\MyOffice\Lua532\bin
  PATHEXT   :    .LUA
LuaRocks;
  PATH      :    C:\Work\LuaRocks
  LUA_PATH  :    C:\Work\LuaRocks\lua\?.lua;C:\Work\LuaRocks\lua\?\init.lua
Local user rocktree (Note: %APPDATA% is user dependent);
  PATH      :    %APPDATA%\LuaRocks\bin
  LUA_PATH  :    %APPDATA%\LuaRocks\share\lua\5.3\?.lua;%APPDATA%
\LuaRocks\share\lua\5.3\?\init.lua
  LUA_CPATH:    %APPDATA%\LuaRocks\lib\lua\5.3\?.dll
System rocktree
  PATH      :    C:\Work\LuaRocks\5.3\bin
  LUA_PATH  :    C:\Work\LuaRocks\5.3\share\lua\5.3\?
.lua;C:\Work\LuaRocks\5.3\share\lua\5.3\?\init.lua
  LUA_CPATH:    C:\Work\LuaRocks\5.3\lib\lua\5.3\?.dll

Note that the %APPDATA% element in the paths above is user specific and it MUST
be replaced by its actual value.
For the current user that value is: C:\Users\Admin\AppData\Roaming.
```

В результате приложение будет установлено в папке C:\Work\LuaRocks.

2.3.1.3 Установка модуля `lsqlite3complete`

Библиотека `lsqlite3complete` предназначена для создания баз данных SQLite3 и управления ими.

В файле `../LuaRocks/config-5.3.lua` необходимо внести следующие изменения в секции `variables`:

```
variables = {
  MSVCRT = 'm', -- make MinGW use MSVCRT.DLL as runtime
  ...
  MD5SUM = [[C:\Work\LuaRocks\tools\md5sum.exe]]
}
```

Для переменной MD5SUM следует поставить актуальный путь, по которому находится md5sum.exe.

Для сборки модуля `lsqlite3complete` следует запустить CMD и перейти в папку, в которой установлен LuaRocks, например, `C:\Work\LuaRocks`. Далее выполнить команду:

```
luarocks install lsqlite3complete
```

Вывод в случае успешного создания модуля:

```
C:\Work\LuaRocks>luarocks install lsqlite3complete
Installing https://luarocks.org/lsqlite3complete-0.9.5-1.src.rock

lsqlite3complete 0.9.5-1 depends on lua >= 5.1, < 5.5 (5.3-1 provided by
VM)
cl /nologo /MD /O2 -c -Folsqlite3.obj -Ic:\Program
Files\MyOffice\Lua532\include lsqlite3.c -DSQLITE_VERSION="0.9.5" -D
luaopen_lsqlite3=luaopen_lsqlite3complete
lsqlite3.c
lsqlite3.c(47): warning C4005: luaL_register: изменение макроопределения
c:\Program Files\MyOffice\Lua532\include\luauxlib.h(206): note: см.
предыдущее определение "luaL_register"
cl /nologo /MD /O2 -c -Fosqlite3.obj -Ic:\Program
Files\MyOffice\Lua532\include sqlite3.c -DSQLITE_VERSION="0.9.5" -Dlu
aopen_lsqlite3=luaopen_lsqlite3complete
sqlite3.c
link -dll -def:lsqlite3complete.def -out:lsqlite3complete.dll c:\Program
Files\MyOffice\Lua532\lib\liblua.dll.a lsqlite3
.obj sqlite3.obj
Microsoft (R) Incremental Linker Version 14.31.31104.0
Copyright (C) Microsoft Corporation. All rights reserved.

Создается библиотека lsqlite3complete.lib и объект lsqlite3complete.exp
lsqlite3complete 0.9.5-1 is now installed in D:\Work\LuaRocks\5.3
(license: MIT)
```

В случае успешной сборки будет создана библиотека

```
C:\Work\LuaRocks\5.3\lib\lua\5.3\lsqlite3complete.dll.
```



При возникновении ошибки **undefined reference to "....."** необходимо в файл конфигурации `..\LuaRocks\config-5.3.lua` добавить строку


```
external_deps_dirs = { "c:/Work/mingw64/x86_64-w64-mingw32/lib"
}, заменив расположение mingw64 на актуальное.
```

Дальнейшее использование собранной библиотеки `sqlite3complete` описано в разделе [Использование сторонних модулей в ОС Microsoft Windows](#).

2.3.2 Сборка внешних модулей для ОС Linux

Для сборки стороннего модуля под ОС Linux требуется предварительно установленный компилятор GNU GCC языка C++. В некоторых ОС он установлен по умолчанию и никаких дополнительных действий не требуется.

Для обеспечения совместимости с МойОфис Lua Extension API необходимо выполнить сборку стороннего модуля с использованием библиотек Lua из комплекта поставки «МойОфис Стандартный» (начиная с релиза 2020.02). Комплект библиотек находится в инсталляционном каталоге МойОфис, по умолчанию в папке `/opt/myoffice-standard/Lua532`.

Содержимое папок:

- `Lua532/bin` – содержит компилятор, интерпретатор и библиотеку для Lua версии 5.3.2;
- `Lua532/include` – содержит комплект заголовочных файлов Lua для сборки сторонних модулей;
- `Lua532/lib` – содержит комплект библиотек Lua для сборки сторонних модулей.

Фактически инсталляционный каталог МойОфис может отличаться от выбранного по умолчанию.

Путь к каталогам заголовочных файлов и библиотечных файлов должен быть включен в соответствующих ключах компилятора, либо в конфигурационных настройках среды разработки. Подробнее это будет указано далее.

2.3.2.1 Установка LuaRocks



Стоит обратить внимание на то, что в некоторых операционных системах (например, Astra Linux, Альт Линукс) пакеты LuaRocks уже доступны в системе

Для установки LuaRocks предварительно необходимо скачать исходные файлы LuaRocks:

```
curl -R -O https://luarocks.github.io/luarocks/releases/luarocks-3.11.0.tar.gz
```

Далее развернуть скачанный архив:

```
tar -zxf luarocks-3.11.0.tar.gz
```

И перейти в образовавшуюся папку:

```
cd luarocks-3.11.0
```



Для дальнейших шагов необходимо убедиться, что путь к исполняемому файлу Lua, который находится в составе МойОфис, прописан в системной переменной PATH:

```
export PATH="/opt/myoffice-standard/Lua532/bin:$PATH"
```

Далее необходимо запустить конфигурирование LuaRocks, в качестве параметра используется путь к папке `include` пакета Lua, находящегося в составе МойОфис:

```
./configure --with-lua-include=opt/myoffice-standard/Lua532/include/
```

В случае успешной настройки в логе отобразится текст «Done configuring», а также прописаны пути, по которым будет установлен LuaRocks. Пример вывода команды **configure** приведен на рисунке 5.

```
root@vm :/home/vm/luarocks-3.11.0 # ./configure --with-lua-include=opt/myoffice-standard/Lua532/include/
Configuring LuaRocks version 3.11.0...
Lua version detected: 5.3
Lua interpreter found: /opt/myoffice-standard/Lua532/bin/lua
lua.h found: /opt/myoffice-standard/Lua532/include/lua.h
unzip found in PATH: /usr/bin
Done configuring.
LuaRocks will be installed at.....: /usr/local
LuaRocks will install rocks at.....: /usr/local
LuaRocks configuration directory...: /usr/local/etc/luarocks
Using Lua from.....: /opt/myoffice-standard/Lua532
Lua include directory.....: /opt/myoffice-standard/Lua532/include
* Type make and make install:
  to install to /usr/local as usual.
* Type make bootstrap:
  to install LuaRocks into /usr/local as a rock.
root@vm :/home/vm/luarocks-3.11.0 #
```

Рисунок 5 - Установка LuaRocks

Далее необходимо запустить команды сборки и установки:

```
make
make install
```

МойОфис

В случае успешной установки приложение LuaRocks по умолчанию будет расположено в папке `/usr/local/bin/luarocks`.

Для проверки корректности установки следует запустить приложение, на экране отобразятся параметры использования, версия и текущая конфигурация:

```
root@va:/home/va# /usr/local/bin/luarocks

Usage: luarocks [-h] [--version] [--dev] [--server <server>]
      [--only-server <server>] [--only-sources <url>]
      [--namespace <namespace>] [--lua-dir <prefix>]
      [--lua-version <ver>] [--tree <tree>] [--local] [--global]
      [--no-project] [--force-lock] [--verbose] [--timeout <seconds>]
      [<command>] ...

LuaRocks 3.11.0, the Lua package manager
.....
```

2.3.2.2 Установка модуля `lsqlite3complete`

Библиотека `lsqlite3complete` предназначена для создания баз данных SQLite3 и управления ими.

Ссылка на страницу данной библиотеки:

<https://luarocks.org/modules/dougcurrie/lsqlite3complete>.

Для сборки библиотеки `lsqlite3complete` необходимо выполнить следующую команду:

```
/usr/local/bin/luarocks install lsqlite3complete
```

В зависимости от операционной системы и текущего набора установленных приложений может потребоваться установка дополнительных пакетов, например, `git`, `curl`.

Пример вывода в случае успешного выполнения команды:

```
root@va:/home/va# /usr/local/bin/luarocks install lsqlite3complete
Installing https://luarocks.org/lsqlite3complete-0.9.5-1.src.rock

lsqlite3complete 0.9.5-1 depends on lua >= 5.1, < 5.5 (5.3-1 provided by VM:
success)
gcc -O2 -fPIC -I/opt/myoffice-standard/Lua532/include -c lsqlite3.c -o
lsqlite3.o -DSQLite_VERSION="0.9.5" -Dluaopen_lsqliite3=luaopen_lsqliite3complete
-I/usr/include -I/usr/include -I/usr/include
gcc -shared -o /tmp/luarocks_build-lsqlite3complete-0.9.5-1-
9244593/lsqliite3complete.so lsqlite3.o sqlite3.o -L/usr/lib/x86_64-linux-gnu -
```

```
L/usr/lib/x86_64-linux-gnu -L/usr/lib/x86_64-linux-gnu -Wl,-  
rpath,/usr/lib/x86_64-linux-gnu -Wl,-rpath,/usr/lib/x86_64-linux-gnu -Wl,-  
rpath,/usr/lib/x86_64-linux-gnu -lpthread -lm -ldl
```

```
sqlite3complete 0.9.5-1 is now installed in /usr/local (license: MIT)
```

В случае успешной сборки в последней строке будет указан путь, по которому находится собранная библиотека.

В данном примере местоположение библиотеки будет следующим:

```
/usr/local/lib/lua/sqlite3complete.so
```

Список установленных библиотек LuaRocks можно получить, выполнив команду `luarocks list`:

```
root@va:/hime/va# /usr/local/bin/luarocks list  
  
Rocks installed for Lua 5.3  
-----  
sqlite3complete  
  0.9.5-1 (installed) - /usr/local/lib/luarocks/rocks-5.3
```

Пример использования библиотеки `sqlite3complete` в составе надстройки описан в разделе [Использование сторонних модулей в составе надстройки](#).

2.3.3 Использование сторонних модулей в составе надстройки

Для использования сторонних модулей в составе надстройки необходимо расположить их в подкаталоге **bin** рабочего каталога надстройки.

2.3.3.1 Использование сторонних модулей в ОС Microsoft Windows

Для использования сторонних модулей в надстройке (на примере `sqlite3complete`) необходимо выполнить следующие действия:

1. [Загрузить и собрать](#) модуль `sqlite3complete` при помощи LuaRocks.
2. Скопировать модуль библиотеки `sqlite3complete.dll` в папку надстройки **bin** (см. Рисунок 6).

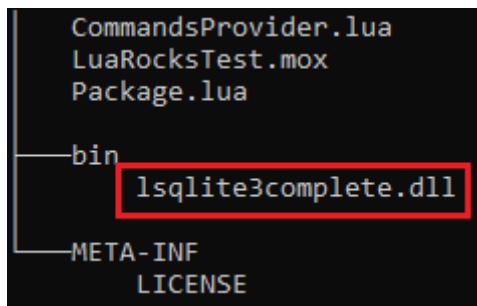


Рисунок 6 - Расположение модуля библиотеки в папке надстройки

3. Включить использование библиотеки в CommandsProvider:

```
local sqlite3 = require("lsqLite3complete")
```

4. Инициализировать модуль:

```
local db = assert(sqlite3:open_memory())
```

5. Объявить структуру баз данных:

```
assert(db:exec[[
CREATE TABLE customer (
  id      INTEGER PRIMARY KEY,
  name    VARCHAR(40)
);
INSERT INTO customer VALUES(1, "Michael");
INSERT INTO customer VALUES(2, "John");
]])
```

6. Использовать вызовы библиотеки для реализации обработки баз данных:

```
local CommandsProvider = {}
local displayCustomers = function(context)
  for row in db:nrows("SELECT * FROM customer") do
    EditorAPI.messageBox(row.name)
  end
end
return CommandsProvider
```

2.3.3.2 Использование сторонних модулей в ОС Linux

Для использования сторонних модулей в надстройке (на примере `lsqLite3complete`) необходимо выполнить следующие действия:

1. Загрузить и [собрать](#) модуль `lsqLite3complete` при помощи `LuaRocks` (см. Рисунок 7).

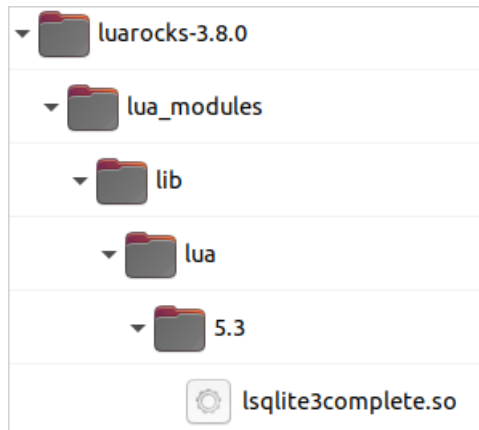


Рисунок 7 - Расположение модуля lsqLite3complete

2. Скопировать модуль библиотеки lsqLite3complete.so в папку **bin** (см. Рисунок 8).

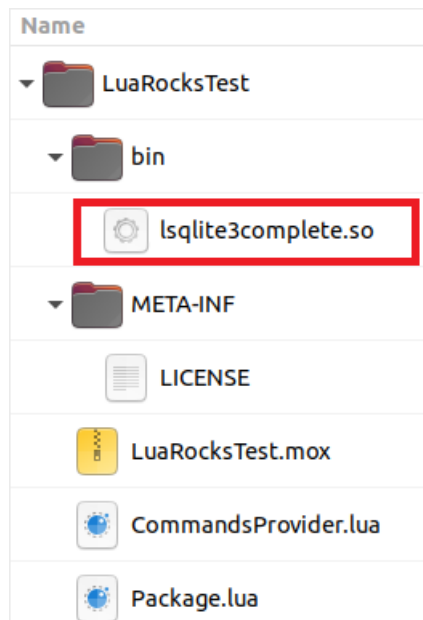


Рисунок 8 - Расположение модуля библиотеки в папке надстройки

3. Включить использование библиотеки в CommandsProvider:

```
local sqlite3 = require("lsqLite3complete")
```

4. Инициализировать модуль:

```
local db = assert(sqlite3:open_memory())
```

5. Объявить структуру баз данных:

```
assert(db:exec[[
CREATE TABLE customer (
  id      INTEGER PRIMARY KEY,
```

```
name    VARCHAR(40)
);
INSERT INTO customer VALUES(1, "Michael");
INSERT INTO customer VALUES(2, "John");
]])
```

6. Использовать вызовы библиотеки для реализации обработки баз данных:

```
local CommandsProvider = {}
local displayCustomers = function(context)
    for row in db:nrows("SELECT * FROM customer") do
        EditorAPI.messageBox(row.name)
    end
end
return CommandsProvider
```

2.4 Установка и управление надстройками в режиме диалога

Для установки архивный файл надстройки должен быть размещен в папке, к которой пользователю разрешен доступ.

2.4.1 Установка надстройки

Для установки надстройки необходимо выбрать пункт **Надстройки > Управление надстройками** в командном меню редактора.

Если надстройки ранее не устанавливались, то в окне **Управление надстройками** отображается сообщение об отсутствии надстроек и кнопка **Установить** (см. Рисунок 9).

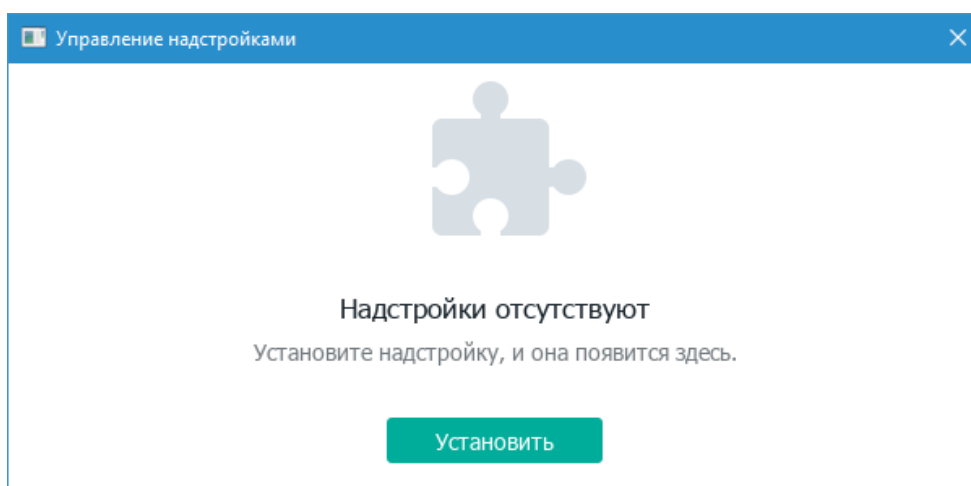


Рисунок 9 – Окно **Управление надстройками** редактора «МойОфис Таблица» при отсутствии установленных надстроек

МойОфис

Если в редакторе уже были установлены надстройки, то в окне **Управление надстройками** отобразится список ранее установленных надстроек (см. Рисунок 10).

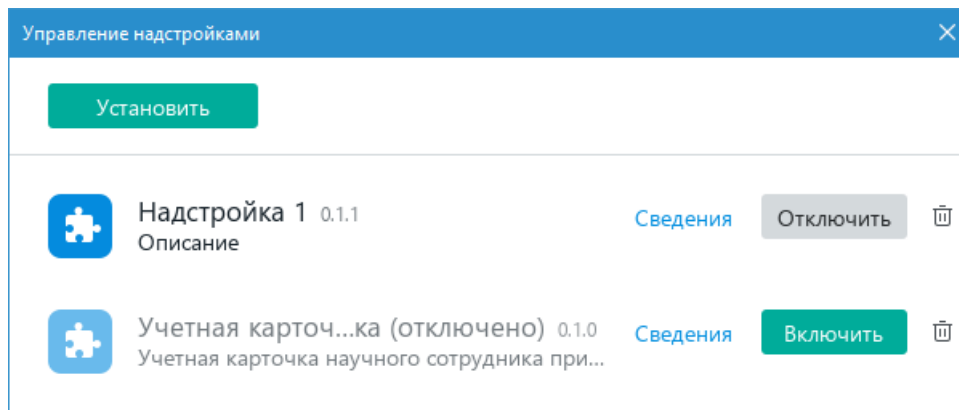



Рисунок 10 – Окно **Управление надстройками** редактора «МойОфис Таблица» со списком установленных надстроек

В строках списка надстроек отображаются: наименование, состояние, версия и описание надстройки.

Для каждой надстройки также отображаются кнопки **Отключить** или **Включить**, позволяющие управлять надстройкой (см. раздел [Управление надстройками](#)), и кнопка  (Удалить), служащая для удаления надстройки (см. раздел [Удаление надстройки](#)).

Для установки надстройки необходимо нажать кнопку **Установить**, на экране отобразится окно проводника для выбора архивного файла MOX устанавливаемой надстройки.

МойОфис

После выбора файла надстройки на экране откроется диалоговое окно **Установка надстройки**. Если в окне отображается сообщение «Надстройка готова к установке. Хотите продолжить?», то это значит, что автором надстройки является проверенный разработчик и надстройка подписана действующим сертификатом (см. Рисунок 11).

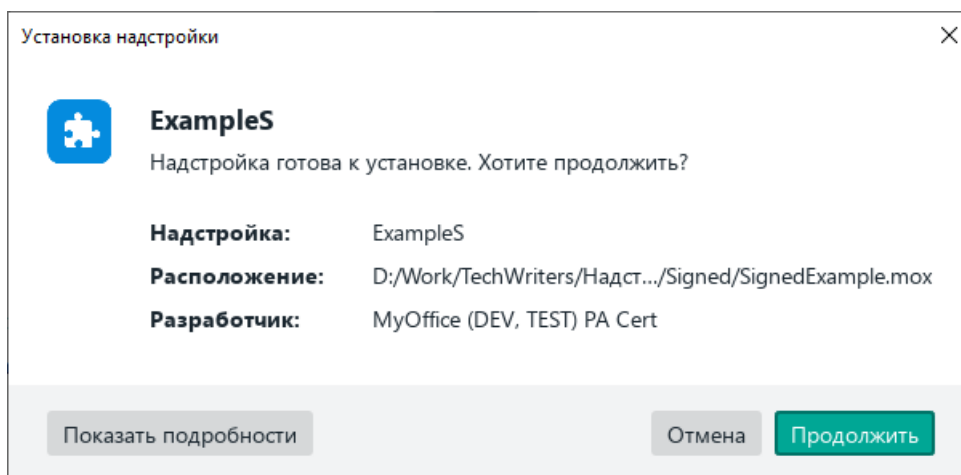


Рисунок 11 – Сообщение в случае подписанной надстройки

В случае, если возникают проблемы с подписью надстройки (см. раздел [Подпись надстройки](#)), в окне отображается соответствующее сообщение (возможные примеры см. в таблице 7).

Таблица 7 - Варианты ошибки подписи надстройки

Заголовок сообщения	Описание
Разработчик не может быть проверен	Надстройка не подписана сертификатом разработчика
Срок действия сертификата разработчика истек	Надстройка подписана, но сертификат недействительный



Настоятельно не рекомендуется устанавливать надстройки непроверенного разработчика или надстройки с недействительным сертификатом.
Установка таких надстроек может нанести вред вашему компьютеру.

Пример одного из возможных сообщений об ошибке подписи надстройки приведен на рисунке 12.

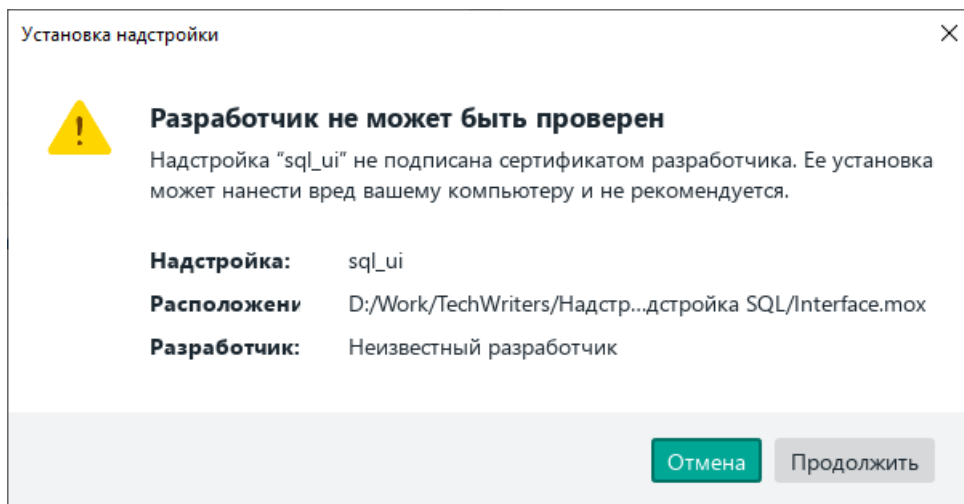


Рисунок 12 – Сообщение об ошибке подписи надстройки

Дальнейшая установка надстройки возможна, но не рекомендуется. Если вы уверены в своих действиях, нажмите кнопку **Продолжить**.

После выбора архива сертифицированной надстройки в открывшемся окне **Лицензионное соглашение** отобразится текст лицензионного соглашения (см. Рисунок 13).

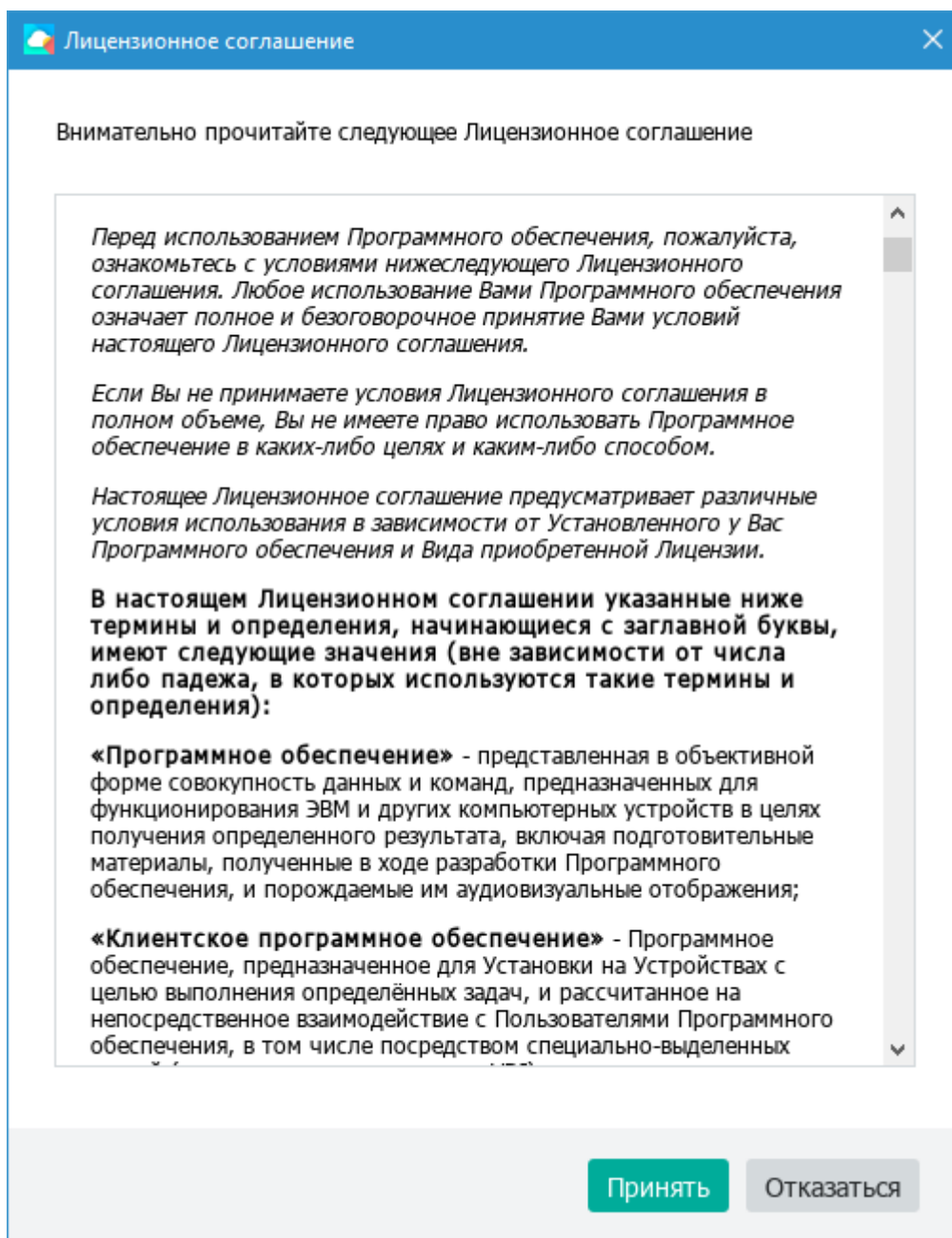


Рисунок 13 – Пример окна лицензионного соглашения

В случае нажатия кнопки **Отказаться** установка надстройки прервется, и на экране отобразится окно с сообщением о прерывании процесса установки (см. Рисунок 14).

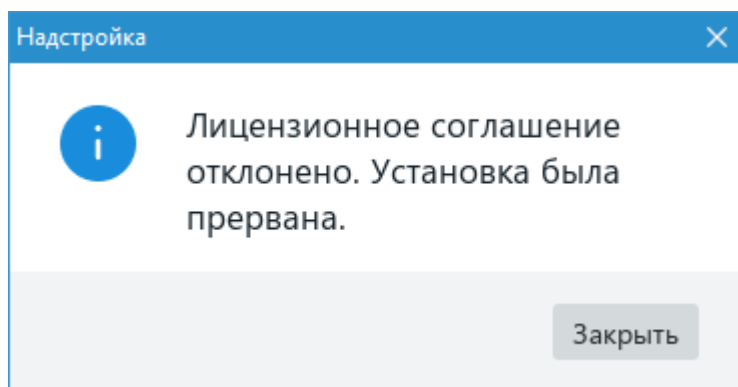


Рисунок 14 – Окно сообщения о прерывании установки

В случае согласия с условиями лицензионного соглашения необходимо нажать кнопку **Принять**, установка надстройки продолжится.

Если выбранная надстройка была установлена ранее, то инициируется процедура обновления надстройки, подробно описанная в разделе [Обновление надстройки](#).

После завершения установки ранее не установленной надстройки в окне **Управление надстройками** отобразится состояние надстройки «**Установлена**» (см. Рисунок 15).

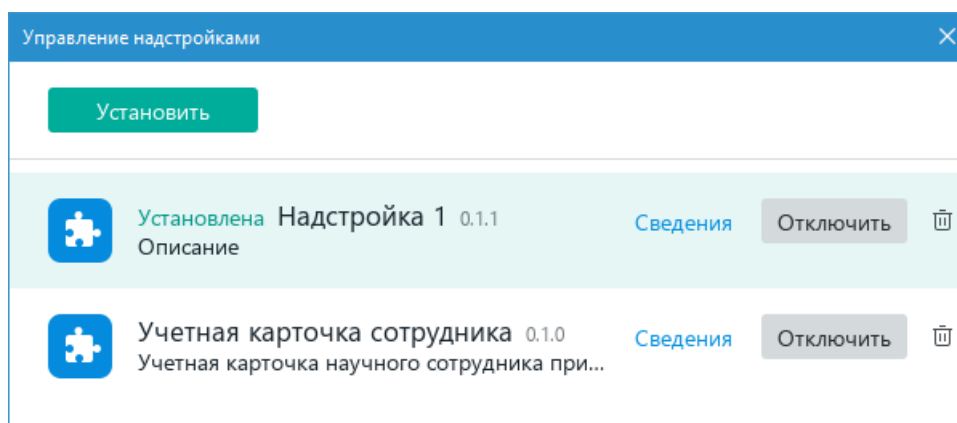


Рисунок 15 – Окно **Управление надстройками** редактора «МойОфис Таблица» после установки надстройки

В случае возникновения критической ошибки на экран выводится сообщение и установка настройки прерывается (см. раздел [Сообщения об ошибках](#)).

В случае, если настройка установлена, но является поврежденной, в строке надстройки вместо описания отобразится информация об ошибке, надстройка будет заблокирована и отключена (см. Рисунок 16), а при нажатии на строку надстройки отобразится окно сообщения об ошибке (см. раздел [Сообщения об ошибках](#)). Например, это произойдет если версия надстройки несовместима с версией редактора.

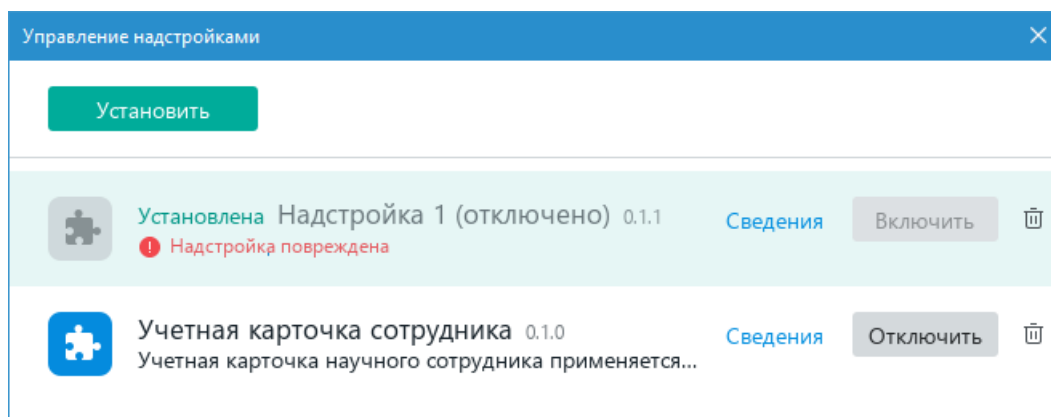


Рисунок 16 – Окно **Управление надстройками** с информацией об ошибке, возникшей при установке надстройки

2.4.2 Управление надстройками

После успешной установки надстройка по умолчанию находится в состоянии «включено».

Имя включенной надстройки отображается в командном меню **Надстройки**, и ее команды доступны для выполнения.

Надстройка может быть отключена нажатием кнопки **Отключить**, расположенной в строке надстройки окна **Управление надстройками** (см. Рисунок 2).

После отключения надстройки в окне управления надстройками отобразится ее состояние – «отключено» и кнопка включения надстройки **Включить**.

Наименование отключенной надстройки не отображается в командном меню **Надстройки** и ее команды не могут быть выполнены.

Для включения надстройки нажмите кнопку **Включить** в строке надстройки в окне **Управление надстройками** (см. Рисунок 3).

В строке надстройки также отображается кнопка **Сведения**, при нажатии на которую открывается окно с информацией о надстройке (см. Рисунок 17):

- **Описание** – описание надстройки;
- **Поставщик** – автор надстройки;
- **Версия** – текущая версия надстройки;
- **Размер** – размер файлов надстройки;
- **Юридическая информация** – в нижней части окна отображается кнопка **Лицензионное соглашение**, при нажатии на которую открывается окно с текстом лицензионного соглашения.

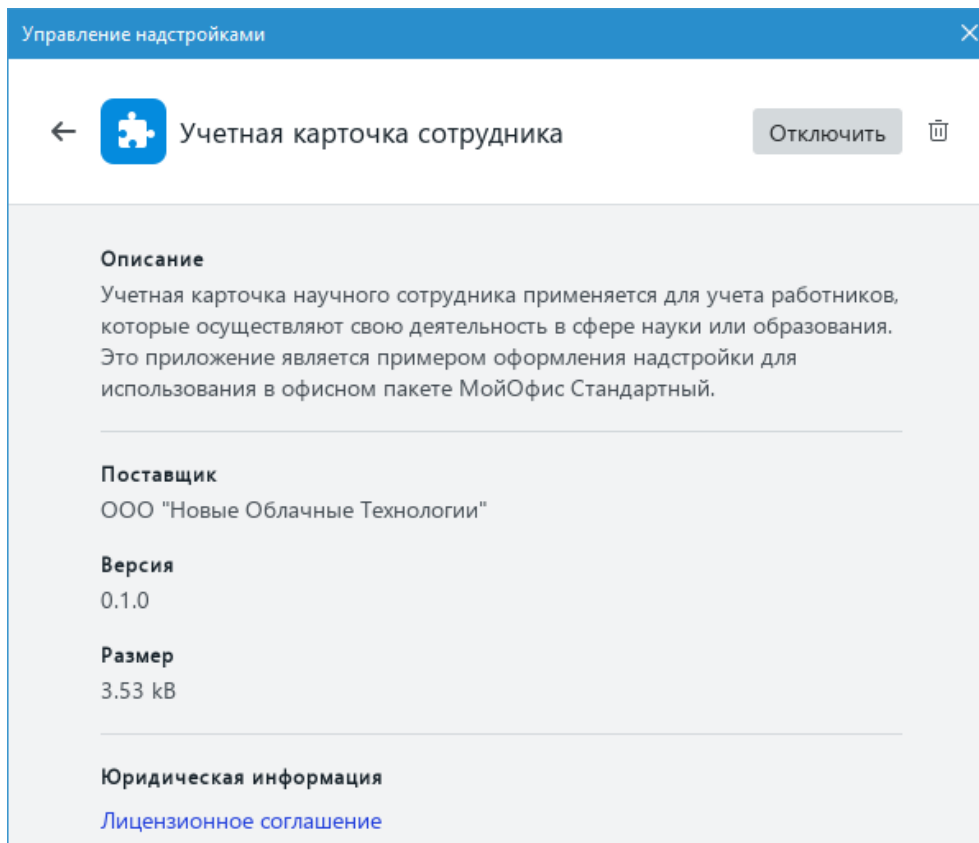




Рисунок 17 – Окно информации о надстройке

В окне информации можно включить или отключить надстройку, нажав соответствующую кнопку, или удалить, нажав кнопку  (Удалить).

Для выхода из окна информации о надстройке необходимо нажать кнопку  в верхней части окна.

2.4.3 Обновление надстройки

Для обновления надстройки необходимо выбрать пункт командного меню **Надстройки > Управление надстройками** и в окне **Управление надстройками** нажать кнопку **Установить**.

На экране отобразится окно проводника для выбора архивного файла обновляемой надстройки.

После выбора архивного файла надстройки в открывшемся окне **Лицензионное соглашение** отобразится текст лицензионного соглашения (см. Рисунок 13).

МойОфис

В случае отказа от принятия соглашения нажатием кнопки **Отказаться** (см. Рисунок 13) на экране отобразится окно с сообщением о прерывании обновления надстройки (см. Рисунок 14).

В случае подтверждения согласия с лицензионным соглашением нажатием кнопки **Принять** (см. Рисунок 13) на экране отобразится окно подтверждения необходимости обновления ранее установленной надстройки (см. Рисунок 18).

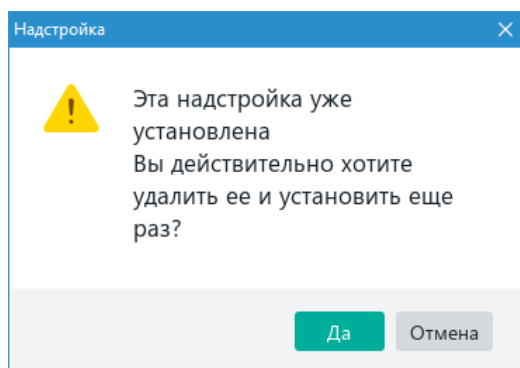


Рисунок 18 – Окно подтверждения необходимости обновления надстройки

После подтверждения необходимости обновления надстройки нажатием кнопки **Да**, произойдет обновление надстройки и в окне **Управление надстройками** отобразится ее состояние – **Обновлена** (см. Рисунок 19).

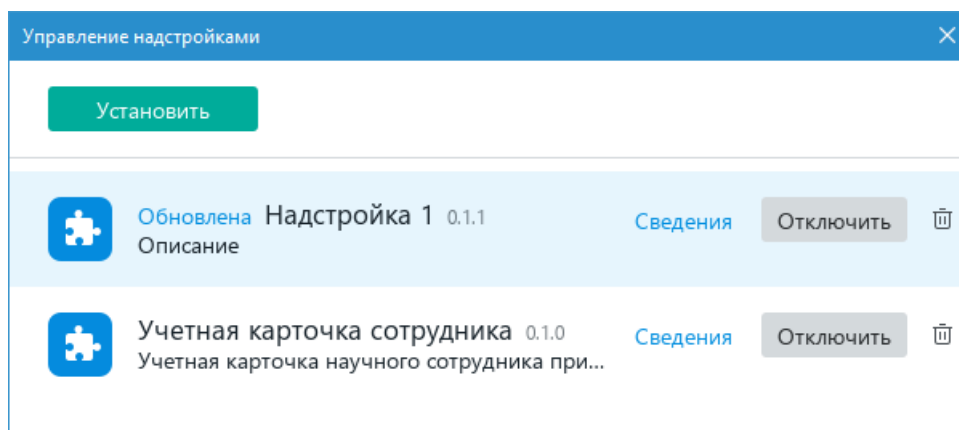



Рисунок 19 – Окно **Управление надстройками** редактора «МойОфис Таблица» после обновления надстройки

2.4.4 Удаление надстройки

Для удаления надстройки необходимо выбрать пункт командного меню **Надстройки > Управление надстройками** и в открывшемся окне **Управление надстройками** (см. Рисунок 20) в строке предназначенной для удаления надстройки нажать кнопку  (Удалить).

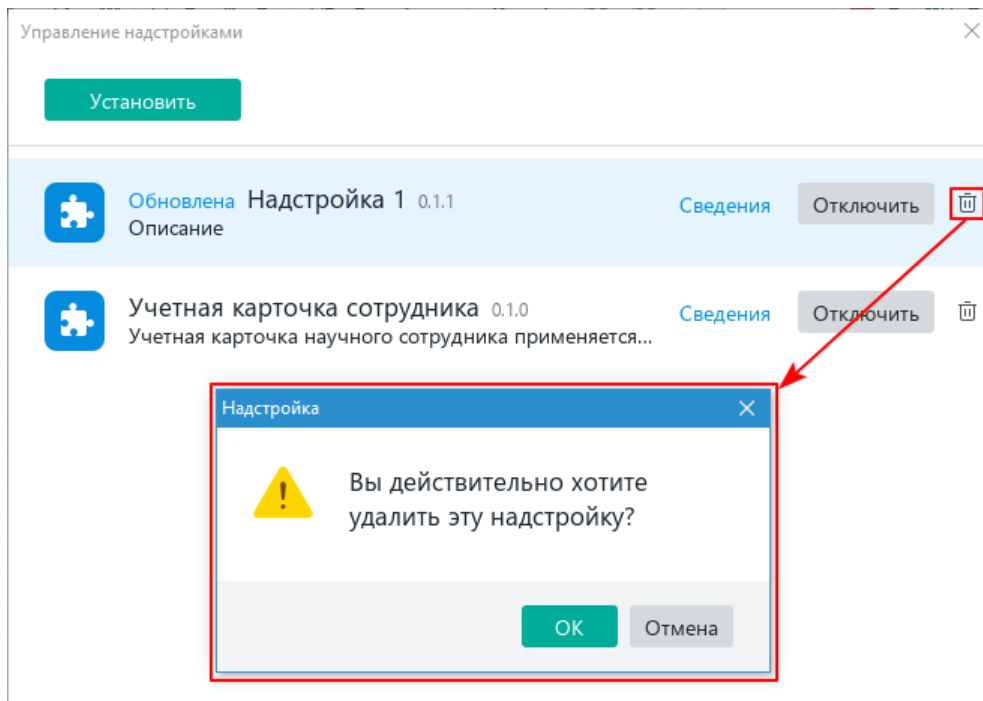


Рисунок 20 – Удаление надстройки

В открывшемся окне подтверждения для удаления надстройки нажмите кнопку **ОК** или кнопку **Отмена** для отмены удаления.

2.4.5 Обновление редакторов

При обновлении редакторов до более новой версии автоматически отключаются все надстройки непроверенных разработчиков и надстройки с недействительными сертификатами. Если вы доверяете источнику, из которого получены эти надстройки, включите их вручную (см. Рисунок 21).

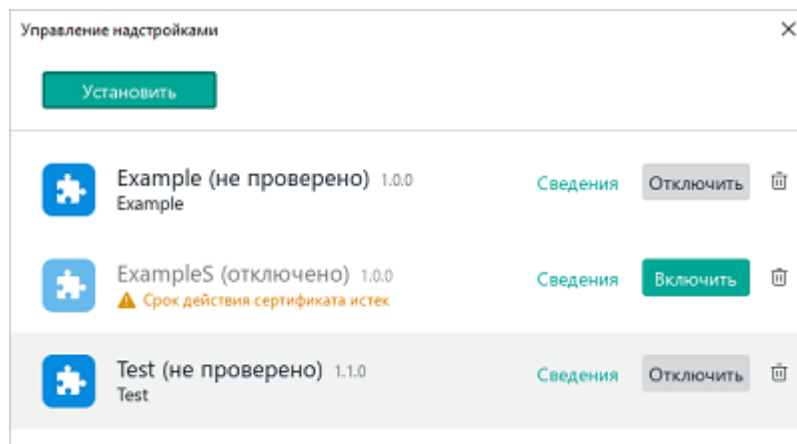


Рисунок 21 – Надстройки после обновления приложения

2.4.6 Редактирование кода установленной надстройки

После сборки, установки и запуска надстройки можно вносить изменения без необходимости каждый раз ее переустанавливать. Для этого нужно найти место на жестком диске, где располагается установленное расширение. Надстройки располагаются по следующему пути:

```
C:\Users\<<Your_User_Name>\AppData\Local\New Cloud Technologies  
Ltd\MyOffice\MyOffice Text\Extensions  
C:\Users\<<Your_User_Name>\AppData\Local\New Cloud Technologies  
Ltd\MyOffice\MyOffice Spreadsheet\Extensions
```

Для каждой надстройки характерен ее уникальный идентификатор. Если надстроек несколько, то соотношение идентификатора надстройки и названия можно узнать через реестр.

Для текстового редактора:

```
HKEY_CURRENT_USER\Software\MyOffice Standard\MyOffice Text\Plugins\Идентификатор  
Надстройки
```

Для табличного редактора:

```
HKEY_CURRENT_USER\Software\MyOffice  
Standard\MyOfficeSpreadsheet\Plugins\Идентификатор Надстройки
```

При использовании такого способа редактирования существующих надстроек важно периодически вручную создавать резервные копии файлов, т.к. при новой установке надстройки прежняя папка будет удалена со всем содержимым, и обновленное содержимое будет распаковано в папку под новым идентификатором, но фактически же это будет та же надстройка.

2.5 Установка и обновление надстроек из командной строки

Установка и обновление надстроек доступны из командной строки.

При установке надстройки из командной строки главное окно приложения не отображается, пользовательское соглашение надстройки (EULA) принимается автоматически, затем выполняется автоматическая регистрация надстройки.



Внимание ! Установка из командной строки может быть произведена только в случае, если надстройка подписана.

Для установки надстройки с помощью командной строки запустите редактор МойОфис, указав следующие параметры:

МойОфис

--installextension **fileName**, где **fileName** – путь к файлу **МОХ** надстройки;
--installextensionmode **installMode**, где **installMode** принимает следующие значения:

- **install** (опция по умолчанию) – выполняется установка надстройки, либо обновление с повышением версии, если такая надстройка уже установлена;
- **update** – выполняется обновление надстройки с повышением версии;
- **downgrade** – выполняется обновление надстройки с понижением версии.



После установки из командной строки надстройка находится в выключенном состоянии. Для включения нажмите кнопку **Включить** в строке настройки диалога **Управление надстройками**.

Примеры установки надстройки из командной строки:

Windows

```
Text Editor: "MyOffice Text.exe" --installextension C:\Users\[username]
\Downloads\for_TE.mox --installextensionmode install
Sheet Editor: "MyOffice Spreadsheet.exe" --installextension C:
\Users\[username]\Downloads\for_SE.mox --installextensionmode install
```

Linux

```
Text Editor: myoffice-text --installextension=/home/
[username]/Загрузки/for_TE.mox --installextensionmode=install
Sheet Editor: myoffice-spreadsheet --installextension=/home/
[username]/Загрузки/for_SE.mox --installextensionmode=install
```

MacOS

```
Text Editor: MyOffice\ Text.app/Contents/MacOs/MyOffice\ Text --
installextension /Users/[username]/Downloads/for_TE.mox --installextensionmode
install
Sheet Editor: MyOffice\ Spreadsheet.app/Contents/MacOs/MyOffice\
Spreadsheet --installextension /Users/[username]/Downloads/for_SE.mox --
installextensionmode install
```

2.6 Вызов команд надстройки

В меню **Надстройки** командного меню редакторов ПО МойОфис отображается список установленных и включенных надстроек (см. Рисунок 2).

При выборе надстройки открывается список команд выбранной надстройки (см. Рисунок 3).

МойОфис

После выбора команды надстройки (см. Рисунок 22) происходит передача соответствующего события в надстройку, в которой предусмотрена обработка этого события (см. раздел [Файл сценария надстройки](#)).

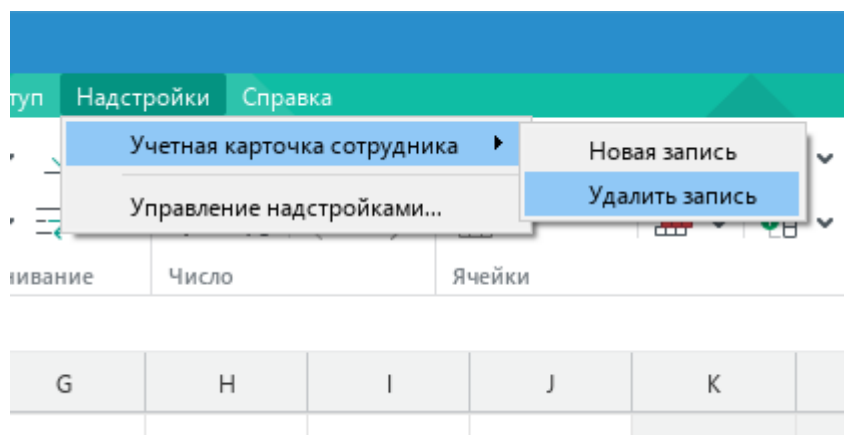


Рисунок 22 – Отображение меню **Надстройки** со списком надстроек и команд надстроек в приложении «МойОфис Таблица»

2.7 Сообщения об ошибках

Если в процессе установки надстройки возникает критическая ошибка, то установка прерывается и на экране возникает сообщение (см. Рисунок 23). При этом текст, описывающий подробности, более детально объясняет причину проблемы. Например, сообщение «Надстройка несовместима с данным приложением» может говорить о том, что значение [applicationId](#) в файле регистрации надстройки не соответствует текущему типу редактора.

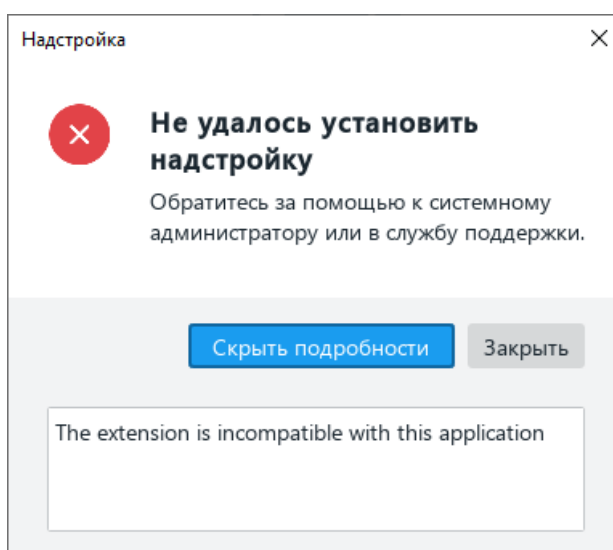


Рисунок 23 – Сообщение о критической ошибке

В случае поврежденной настройки на экран выводится сообщение (см. Рисунок 24).

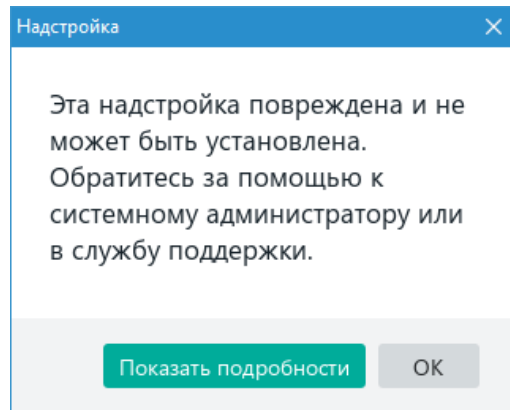


Рисунок 24 – Пример сообщения об ошибке, возникшей при установке надстройки

При нажатии кнопки **Показать подробности** в окне ошибки отображается информация о причине возникновения ошибки (см. Рисунок 25).

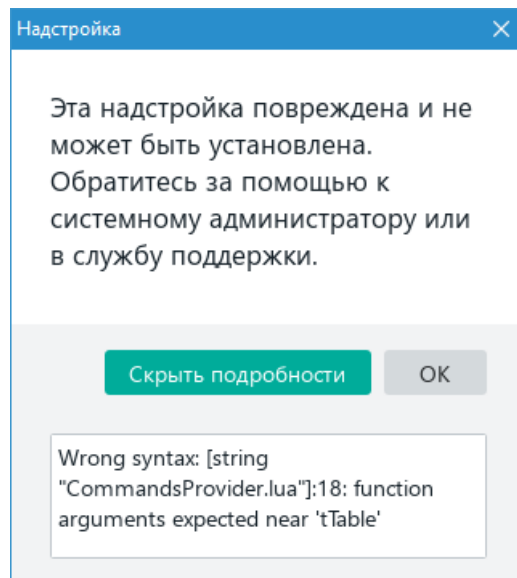


Рисунок 25 – Пример сообщения об ошибке с информацией о причине возникновения

2.8 Функции для работы с файлами надстройки

В состав архивного файла надстройки кроме обязательных файлов (см. раздел [Состав и структура надстройки](#)) могут входить и другие файлы.

При установке надстройки файлы, содержащиеся в архиве надстройки, автоматически копируются в **домашний** каталог надстройки. Файлы из **домашнего** каталога можно получить только в режиме просмотра. При обновлении надстройки содержимое **домашнего** каталога очищается и обновляется.

МойОфис

Для работы с файлами надстройки предназначен **рабочий** каталог, в котором надстройка может создавать, копировать, удалять любые файлы. При обновлении надстройки содержимое **рабочего** каталога надстройки не меняется.

2.8.1 Функция openBundledFile

Для открытия файла **домашнего** каталога используется функция openBundledFile.

```
file openBundledFile(path)
```

Где **path** – путь к файлу.

Файл открывается в режиме просмотра. Функция возвращает тот же объект, что и метод Lua io.open().

2.8.2 Функция copyBundledDirectory

Для копирования содержимого **домашнего** каталога в **рабочий** каталог используется функция copyBundledDirectory.

```
void copyBundledDirectory(from, to, overwrite)
```

Параметры функции:

– from – название файла;

– to – каталог размещения;

– overwrite – параметр может принимать следующие значения:

- "ignore" – файл не копировать, если он существует в каталоге размещения;
- "overwrite" – заменить файл в каталоге размещения;
- "fail" – вызвать исключение, если файл с указанным названием существует.

Если каталог размещения не существует, он будет создан автоматически.

2.8.3 Функция getWorkingDirectory

Функция getWorkingDirectory возвращает путь к **рабочему** каталогу.

```
string getWorkingDirectory()
```

Возвращаемый путь к **рабочему** каталогу не содержит завершающей косой черты.

2.8.4 Пример использования функций

Пример использования функций работы с файлами надстройки:

```
function CommandsProvider.readFileContent(context)
    context.doWithDocument(function(document)
```

```
local file = openBundledFile("hello.txt")
local content = file:read "*a"
file:close()

copyBundledFile("hello.txt", "data/hello.txt", "ignore")

local out = io.open(getWorkingDirectory(). "/data/hello.txt", "rb")
local outContent = out:read "*a"
out:close()

local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell(DocumentAPI.CellPosition(0, 0))
cell:setText(content .. outContent)
end)
end
```

2.9 Определение текущей локали

2.9.1 Функция getCurrentLocale

Для получения текстового представления текущей локали используется глобальная функция `getCurrentLocale`.

```
string getCurrentLocale()
```

Пример:

```
local PrintCurrentLocale = {}
function PrintCurrentLocale.printMsg()
    print("Current locale: " .. getCurrentLocale()) --- example: en_US
end
return PrintCurrentLocale
```

2.10 Пользовательский интерфейс в надстройках

Для описания пользовательского интерфейса в надстройках редакторов МойОфис разработчик использует виджеты (widgets) и компоновки (layouts) (см. раздел [Справочник таблиц и методов пользовательского интерфейса надстроек](#)).

Виджеты представляют собой элементы пользовательского интерфейса, такие как кнопки, выпадающие списки, флажки и т. д. Виджеты используются для отображения данных в диалоговом окне и для получения ввода от пользователя.

Компоновки предоставляют инструмент для автоматического размещения виджетов внутри диалогового окна.

МойОфис

Для создания пользовательского интерфейса надстроек доступны следующие виджеты:

- **Label (Надпись)** – элемент для отображения неизменяемого текста, например, пояснительная надпись для поля ввода (см. раздел [ui:Label](#));
- **Button (Кнопка)** – элемент для отображения кнопки, требующей нажатия (см. раздел [ui:Button](#));
- **CheckBox (Флажок)** – элемент для отображения флажка (см. раздел [ui:CheckBox](#));
- **RadioButton (Радио кнопка)** – элемент для отображения переключателя (см. раздел [ui:RadioButton](#));
- **GroupBox (Рамка группы)** – элемент для отображения группы элементов **Радио кнопка** (см. раздел [ui:GroupBox](#));
- **ListBox (Список элементов)** – элемент для отображения окна списка (см. раздел [ui:ListBox](#));
- **ComboBox (Поле с выпадающим списком)** – элемент для отображения поля с выпадающим списком (см. раздел [ui:ComboBox](#));
- **TextBox (Текстовое поле)** – элемент для ввода небольшого объема текста (см. раздел [ui:TextBox](#)).

Для размещения виджетов в пространстве диалогового окна используются следующие виды компоновок:

- **Row** – располагает виджеты по горизонтали слева направо по ширине диалогового окна (см. раздел [ui:Row](#));
- **Column** – располагает виджеты по вертикали сверху вниз по высоте диалогового окна (см. раздел [ui:Column](#));
- **Spacer** – используется для выравнивания позиции виджета относительно ширины или высоты диалогового окна (см. раздел [ui:Spacer](#)).

Пример:

```
local dlg = ui:Dialog {
  Size = Forms.Size(600, 300),
  ui:Column {
    ui:Row {
      ui:Spacer {},
      ui:Column {lblHello},      -- Надпись
      ui:Column {txtEntryField}, -- Текстовое поле
    }
  }
}
```

```
        ui:Spacer {}
    }
}
context.showDialog(dlg)
```

В данном примере **Надпись** и **Текстовое поле** располагаются в виде двух столбцов (метка-поле), выровненных по центру относительно ширины диалогового окна.

2.11 Печать документа из надстройки

Вывод документа на печать производится с помощью метода глобальной таблицы [EditorAPI.showPrintDialog](#).

Пример:

```
local Actions = {}
function Actions.printDocument(context)
    context.doWithDocument(function(document)
        EditorAPI.showPrintDialog()
    end)
end
return Actions
```

2.12 Пример создания надстройки

Для создания надстройки необходимо в текстовом редакторе создать файл регистрации надстройки – **Package.lua**, файл сценария надстройки и файл лицензионного соглашения.

Структура файла **Package.lua** должна соответствовать описанию, приведенному в разделе [Файл регистрации надстройки](#).

Пример файла регистрации надстройки **Package.lua**:

```
local Package = {
    vendor = "ООО \"Новые Облачные Технологии\"",
    description = "Учетная карточка научного сотрудника",
    extensionID = "com.example.entryform",
    extensionName = "Учетная карточка сотрудника",
    extensionVersion = { major = 0, minor = 1, patch = 0, build = "" },
    applicationId = "MyOffice Spreadsheet",
    apiVersion = { major = 1, minor = 0 },
    commandsProvider = "cmd/entryform.lua",
    fallbackLanguage = 'RU'
```



```
}  
return Package
```

Для приведенного выше примера файл сценария надстройки **Учетная карточка сотрудника** называется **entryform.lua**, который находится в папке **/cmd** домашнего каталога надстройки.

Файл сценария надстройки должен соответствовать описанию, приведенному в разделе [Файл сценария надстройки](#).

Пример файла сценария надстройки **entryform.lua**:

```
local Actions = {}  
  
function Actions.getCommands()  
    return {  
        {  
            -- Идентификатор команды меню  
            id = "EntryForm.addRecord",  
            -- Наименование команды в меню:  
            -- "Надстройки" - "Учетная карточка сотрудника" - "Новая запись"  
            menuItem = "Новая запись",  
            -- Наименование функции-обработчика  
            command = Actions.addRecord  
        },  
        {  
            id = "EntryForm.deleteRecord",  
            menuItem = "Удалить запись",  
            command = Actions.deleteRecord  
        }  
    }  
end  
  
-- TODO New employee  
function Actions.addRecord(context)  
    context.doWithDocument(function(document)  
        EditorAPI.messageBox("Новая запись!")  
    end)  
end  
  
-- TODO Delete employee  
function Actions.deleteRecord(context)  
    context.doWithDocument(function(document)
```

МойОфис

```
EditorAPI.messageBox("Удаление записи!")  
end)  
end  
return Actions
```

Требования к файлу лицензионного соглашения приведены в разделе [Файл лицензионного соглашения](#).

После завершения создания файлов надстройки, которые обязательно должны присутствовать в архиве надстройки, необходимо поместить их и другие необходимые файлы в архивный файл формата ZIP с расширением MOX.

Установка надстройки должна происходить в соответствии с описанием, приведенном в разделе [Установка надстройки](#) или в разделе [Установка и обновление надстроек из командной строки](#).

Вызов и выполнение команд надстройки должны происходить в соответствии с описанием, приведенном в разделе [Вызов команд надстройки](#).

Результат выполнения команды **Новая запись** надстройки **Учетная карточка сотрудника** отобразится в главном окне редактора МойОфис (см. Рисунок 26).

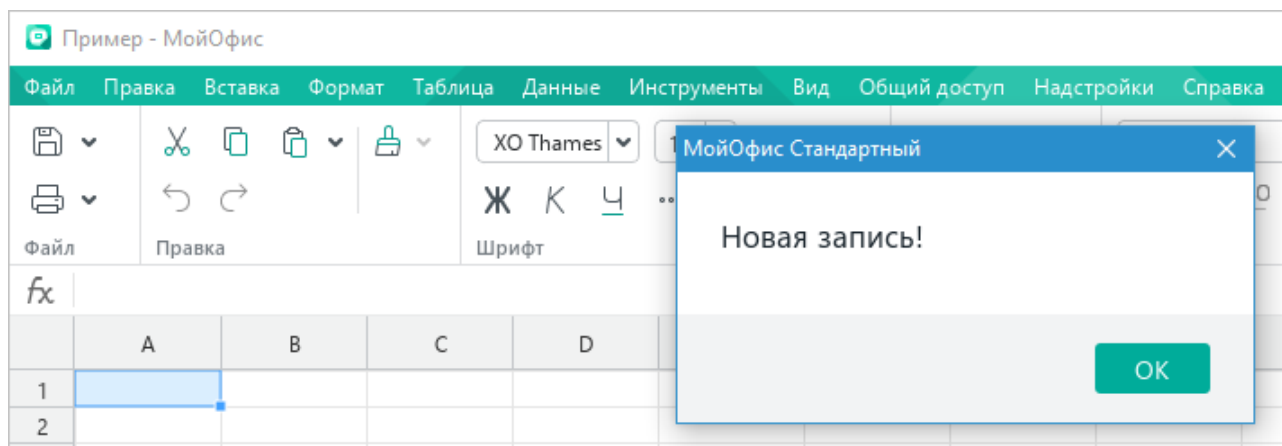


Рисунок 26 – Главное окно приложения «МойОфис Таблица» с отображением результата выполнения команды **Новая запись** надстройки **Учетная карточка сотрудника**

3 Объектная модель МойОфис SDK

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

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

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

Можно выделить следующие группы таблиц:

- [DocumentAPI](#) – содержит таблицы и функции для представления тех 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;
- [Forms](#) и [ui](#) – используются при создании интерактивных форм ввода данных в надстройках.

Вышеописанные таблицы составляют объектную модель МойОфис SDK (см. Рисунок 27).

МойОфис

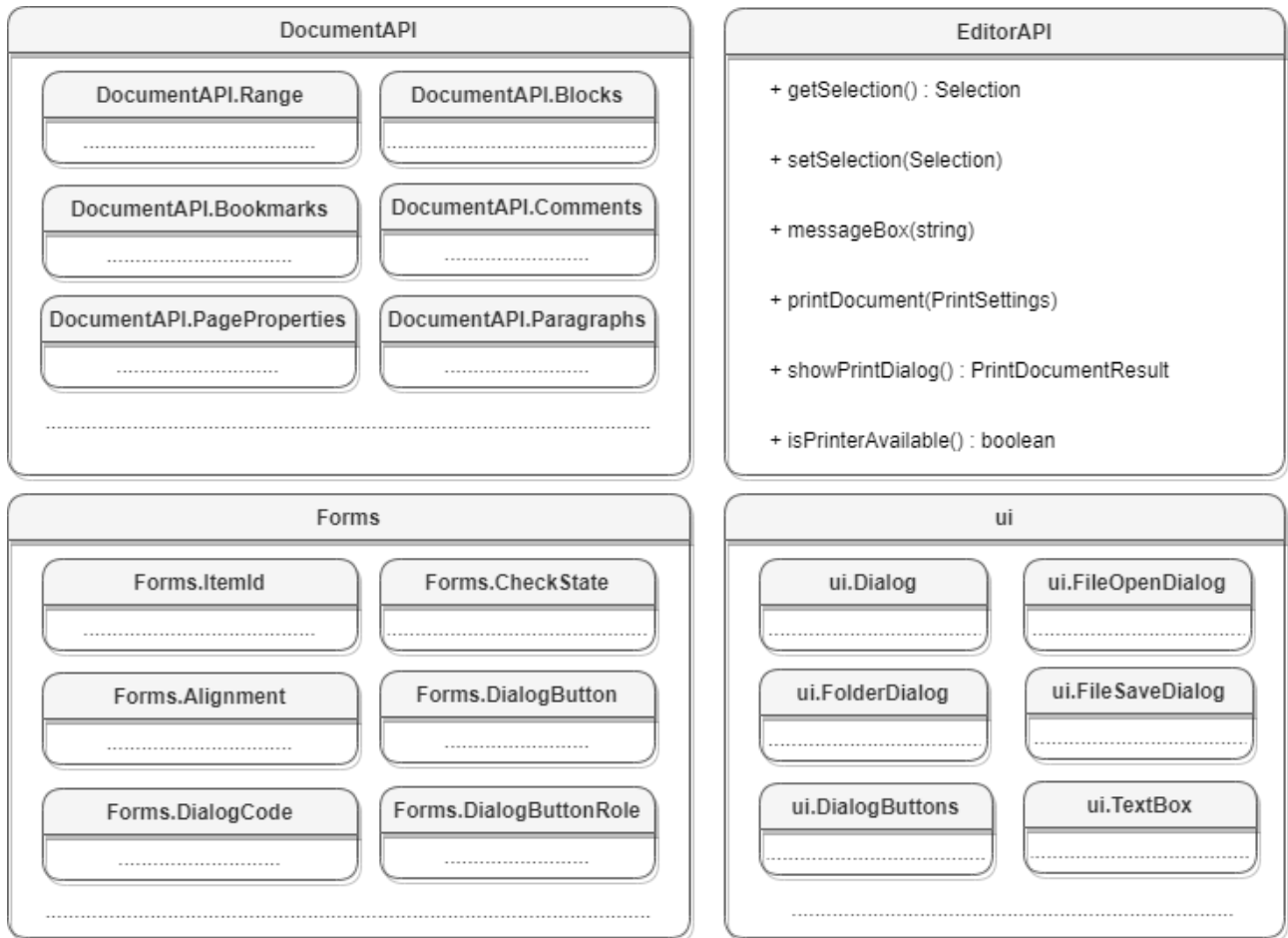


Рисунок 27 – Объектная модель МойОфис SDK.

4 Работа с документами

4.1 Работа с текстовым документом

4.1.1 Создание и открытие текстового документа

Метод [Application::createDocument](#) создает документ. В качестве параметра используются [DocumentType](#) или [DocumentSettings](#).

Примеры создания текстового документа:

```
local document = application:createDocument(DocumentAPI.DocumentType_Text)
```

```
local settings = DocumentAPI.DocumentSettings()  
settings.documentType = DocumentAPI.DocumentType_Text  
local document = application:createDocument(settings)
```

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

Примеры загрузки текстового документа:

```
local doc = application:loadDocument("sample.docx")
```

```
local docSettings = DocumentAPI.DocumentSettings()  
docSettings.documentType = DocumentAPI.DocumentType_Text  
local loadSettings = DocumentAPI.LoadDocumentSettings()  
loadSettings.commonDocumentSettings = docSettings  
local doc = application:loadDocument("sample.docx", loadSettings)
```

4.1.2 Сохранение и экспорт текстового документа

Метод [Document::saveAs](#) сохраняет документ по указанному пути.

Примеры сохранения текстового документа:

```
function CheckCtx.saveAs(context)  
    context.doWithDocument(function(document)  
        document:saveAs(filePath)  
    end)  
end
```

```
function CheckCtx.saveAs(context)  
    context.doWithDocument(function(document)  
        saveDocumentSettings = DocumentAPI.SaveDocumentSettings()  
    end)  
end
```

```
saveDocumentSettings.documentFormat = DocumentAPI.DocumentFormat_OXML
saveDocumentSettings.documentType = DocumentAPI.DocumentType_Text
saveDocumentSettings.documentPassword = "password"
saveDocumentSettings.isTemplate = false

saveDocumentSettings.dsvSettings = DocumentAPI.DSVSettings()
saveDocumentSettings.dsvSettings.autofit = true
saveDocumentSettings.dsvSettings.startBlockIndex = 0
saveDocumentSettings.dsvSettings.lastBlockIndex = 10

document:saveAs(filePath, saveDocumentSettings)
end)
end
```

Метод [Document::exportAs](#) экспортирует документ в файл по указанному пути с заданным форматом типа [ExportFormat](#).

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

Примеры экспорта табличного документа:

```
function CheckCtx.exportAs(context)
context.doWithDocument(function(document)
document:exportAs(filePath, DocumentAPI.ExportFormat_PDFA1)
end)
end
```

4.1.3 Разделы (секции) документа

На рисунке 28 изображена объектная модель таблиц, относящихся к работе с секциями текстового документа.

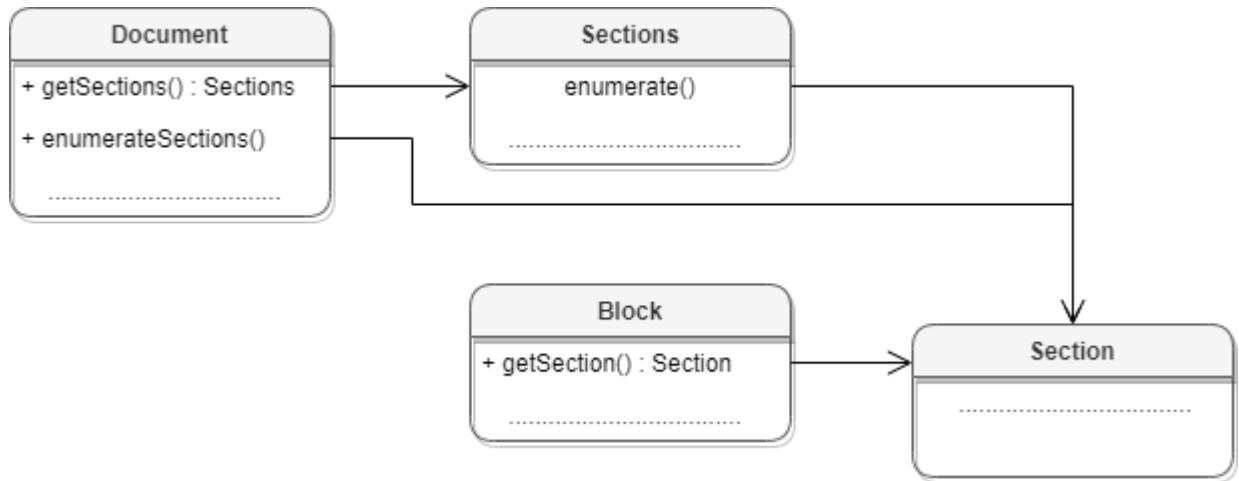


Рисунок 28 – Объектная модель таблиц для работы с секциями

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 [Sections](#) с помощью вызова [document:getSections\(\)](#);
- перечисление всех доступных секций [Section](#) с помощью вызова [document:enumerateSections\(\)](#);
- получение секции [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()
```

4.1.3.1 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
end
```

4.1.3.2 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)
```

4.1.4 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены являются листы документа. Доступ к объектам [Table](#) осуществляется из [Blocks](#) (см. Рисунок 29).

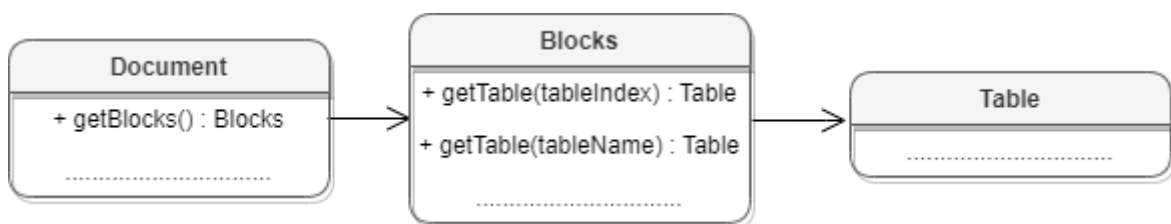


Рисунок 29 – Объектная модель для работы с таблицами

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")
```

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()
```

4.1.5 Работа с закладками

Основной таблицей для работы с закладками является [DocumentAPI.Bookmarks](#) (см. Рисунок 30). 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()`.

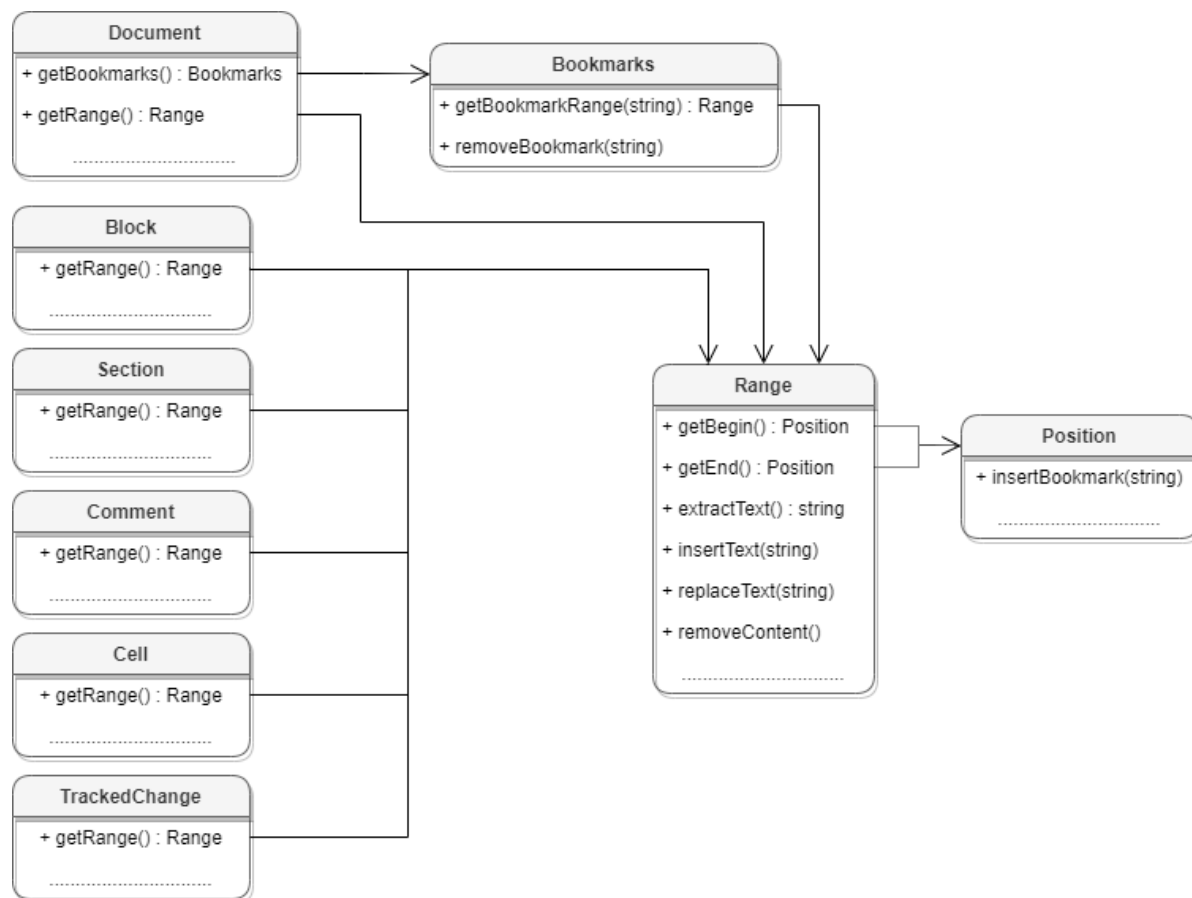


Рисунок 30 – Объектная модель для работы с закладками

Доступны следующие операции с закладками:

Вставка закладки в указанное местоположение

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

Удаление закладки с заданным именем

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

Поиск диапазона документа по имени

```
-- Поиск диапазона для закладки с именем "Signers"
local bookmarkRange = 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

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

Удаление содержимого закладки

```
bookmarkRange:removeContent()
```

Получение текстового содержимого закладки

```
local bookmarkContent = bookmarkRange:extractText()  
print(bookmarkContent)
```

Вставка таблицы в закладку

```
-- Вставка таблицы в закладку "Signers"  
local table = bookmarkRange:getEnd():insertTable(3, 3, "Signers")
```

4.1.6 Рецензирование документов

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

- помечать изменения, вносимые пользователем в текстовый документ [TrackedChange](#);
- добавлять текстовые комментарии для фрагментов текстового документа [Comments](#).

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

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

Пример:

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

МойОфис

Инструменты рецензирования применяются к диапазону документа, по этой причине методы доступа к ним находятся в таблице [Range](#) (см. Рисунок 31).

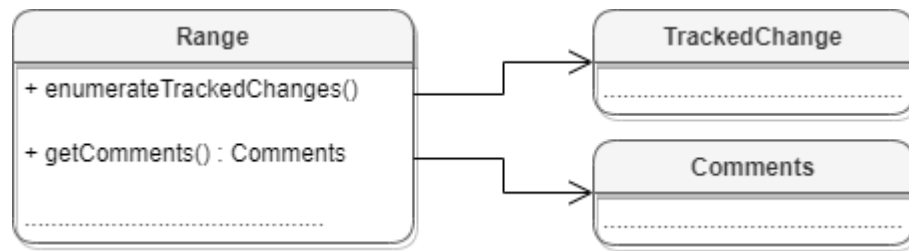


Рисунок 31 – Инструменты рецензирования документа

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.

Пример работы с изменениями:

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
    print(change:getRange():extractText())
end
```

Пример работы с комментариями:

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
    print(change:getRange():extractText())
    print(change:getType())

    local trackedChangeInfo = change:getInfo()
    local author = trackedChangeInfo.author
    local ts = trackedChangeInfo.timeStamp
    local time = string.format("%d/%d/%d - %d:%d:%d", ts.day, ts.month, ts.year,
ts.hour, ts.minute, ts.second)
    print(author.name, time)
end
```

4.1.7 Поиск в текстовом документе

Для поиска в документе необходимо создать экземпляр таблицы [DocumentAPI.Search](#) посредством вызова [DocumentAPI.createSearch](#), затем использовать метод [Search.findText](#) (см. Рисунок 32).

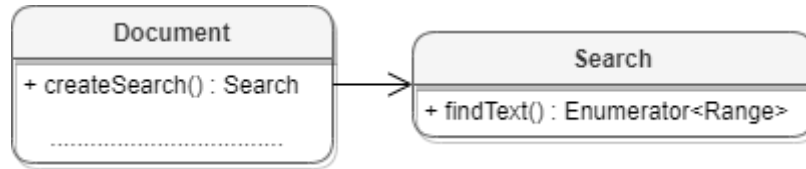


Рисунок 32 – Объектная модель для поиска в документе

Примеры:

```
// Поиск по всему документу
search = DocumentAPI.createSearch(document)
searchResult = search.findText("English")
for occurrence in searchResult do
    print(occurrence.extractText())
end

// Поиск только в диапазоне первого блока
search = DocumentAPI.createSearch(document)
firstBlock = document:getBlocks():getBlock(0)
searchResult = search.findText("ук", firstBlock:getRange())
for occurrence in searchResult do
    print(occurrence.extractText())
end
```

4.1.8 Работа с графическими объектами

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.

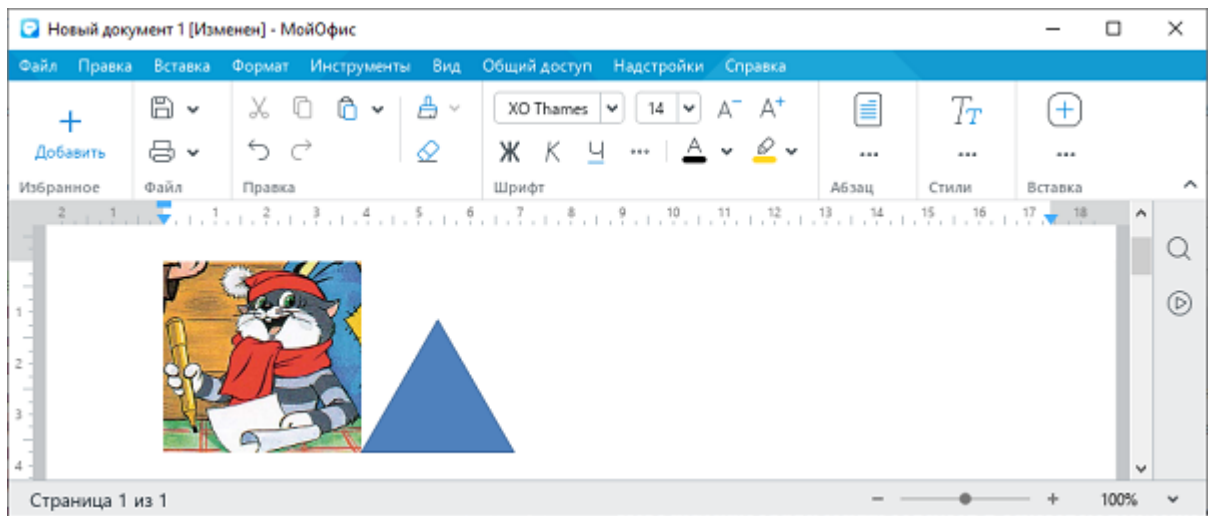


Рисунок 33 – Графические объекты в текстовом документе

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
        -- Image
    else
        -- Shape
    end
end
end
```

Вариант 2: перечисление изображений в текстовом документе

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

Перечисление графических объектов в таблицах текстового документа

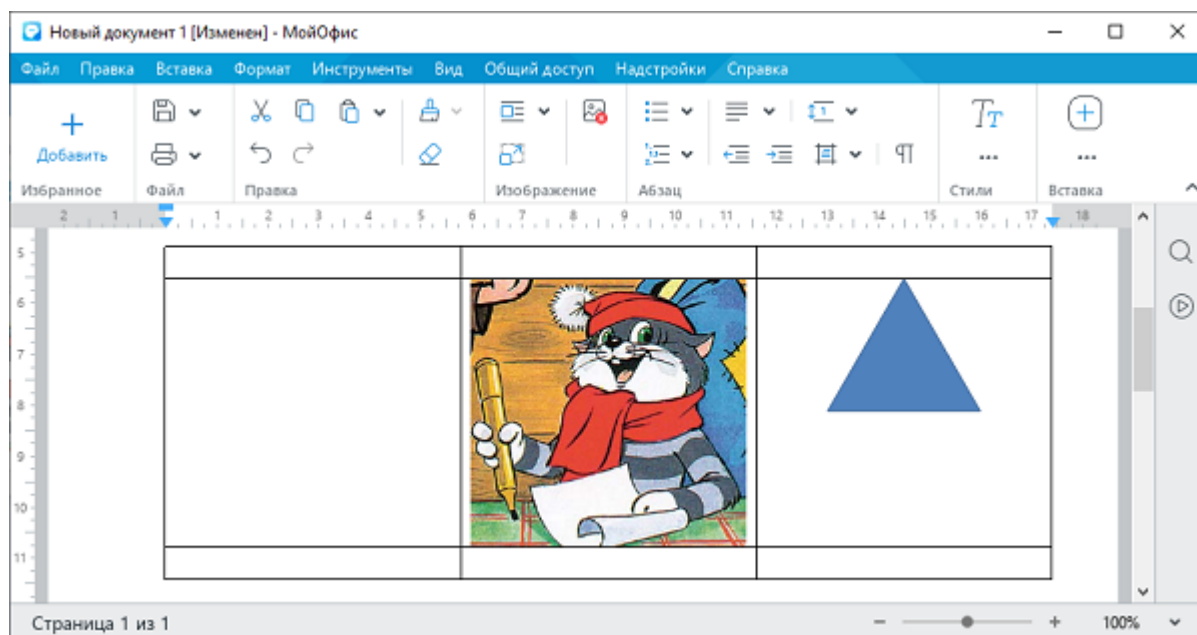


Рисунок 34 – Графические объекты в таблице текстового документа

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
        -- Image
    else
        -- Shape
    end
end
```

Вариант 2: перечисление изображений в таблице текстового документа

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

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)  
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)  
range:getBegin():insertImage("C://Tmp/123.jpg", imageSize)
```

4.2 Работа с табличным документом

4.2.1 Создание и открытие табличного документа

Метод [Application::createDocument](#) создает документ. В качестве параметра используются [DocumentType](#) или [DocumentSettings](#).

Примеры создания табличного документа:

```
local document = application:createDocument(DocumentAPI.DocumentType_Workbook)  
  
local settings = DocumentAPI.DocumentSettings()  
settings.documentType = DocumentAPI.DocumentType_Workbook  
local document = application:createDocument(settings)
```

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

Примеры загрузки табличного документа:

```
local doc = application:loadDocument("sample.xlsx")  
  
local docSettings = DocumentAPI.DocumentSettings()  
docSettings.documentType = DocumentAPI.DocumentType_Workbook  
local loadSettings = DocumentAPI.LoadDocumentSettings()  
loadSettings.commonDocumentSettings = docSettings  
local doc = application:loadDocument("sample.xlsx", loadSettings)
```

4.2.2 Сохранение и экспорт табличного документа

Метод [Document::saveAs](#) сохраняет документ по указанному пути.

Примеры сохранения табличного документа:

```
function CheckCtx.saveAs(context)
    context.doWithDocument(function(document)
        document.saveAs(filePath)
    end)
end
```

```
function CheckCtx.saveAs(context)
    context.doWithDocument(function(document)
        saveDocumentSettings = DocumentAPI.SaveDocumentSettings()

        saveDocumentSettings.documentFormat = DocumentAPI.DocumentFormat_OXML
        saveDocumentSettings.documentType = DocumentAPI.DocumentType_Workbook
        saveDocumentSettings.documentPassword = "password"
        saveDocumentSettings.isTemplate = false

        saveDocumentSettings.dsvSettings = DocumentAPI.DSVSettings()
        saveDocumentSettings.dsvSettings.autofit = true
        saveDocumentSettings.dsvSettings.startBlockIndex = 0
        saveDocumentSettings.dsvSettings.lastBlockIndex = 10

        document.saveAs(filePath, saveDocumentSettings)
    end)
end
```

Метод [Document::exportAs](#) экспортирует документ в файл по указанному пути с заданным форматом типа [ExportFormat](#).

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

Примеры экспорта табличного документа:

```
function CheckCtx.exportAs(context)
    context.doWithDocument(function(document)
        document.exportAs(filePath, DocumentAPI.ExportFormat_PDFA1)
    end)
end
```

МойОфис

4.2.3 Диаграммы

Работа с диаграммами реализована только в табличных документах. На рисунке 35 изображена объектная модель таблиц, относящихся к работе с диаграммами.

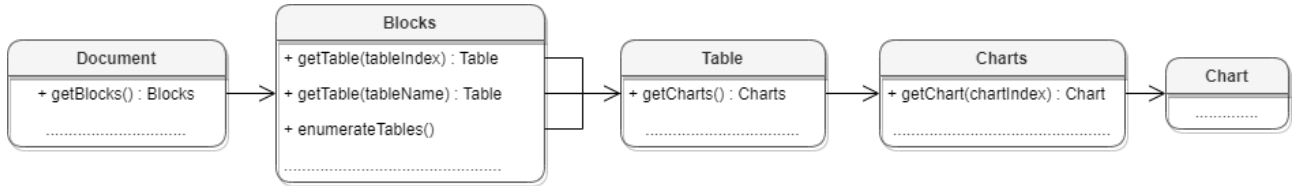


Рисунок 35 – Объектная модель таблиц для работы с диаграммами

Для доступа к списку диаграмм используется метод таблицы (листа документа) [Table::getCharts\(\)](#).

Для получения диаграммы [Chart](#) используется метод [Charts::getChart\(\)](#).

Пример

```
for sheetList in document:getBlocks():enumerateTables() do
    charts = sheetList:getCharts()
    chart = charts:getChart(0)
    print(chart:getTitle())
end
```



Создание и удаление диаграмм в текущей версии не поддерживается.

4.2.4 Поиск в табличном документе

Для поиска в табличном документе необходимо создать экземпляр класса [Search](#) посредством вызова [DocumentAPI.createSearch\(document\)](#), затем использовать метод [Search::findText](#) (см. Рисунок 36).



Рисунок 36 – Объектная модель для поиска в документе

Пример для табличного документа:

```
search = DocumentAPI.createSearch(document)
-- Поиск по всему документу
ranges = search.findText("English")
for occurrence in ranges do
    print(occurrence.extractText())
end
-- Поиск только в диапазоне первого блока
local firstBlockRange = document:getBlocks():getBlock(0):getRange()
ranges = search.findText("English", firstBlockRange)
for occurrence in ranges do
    print(occurrence.extractText())
end
```

4.2.5 Работа с графическими объектами

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

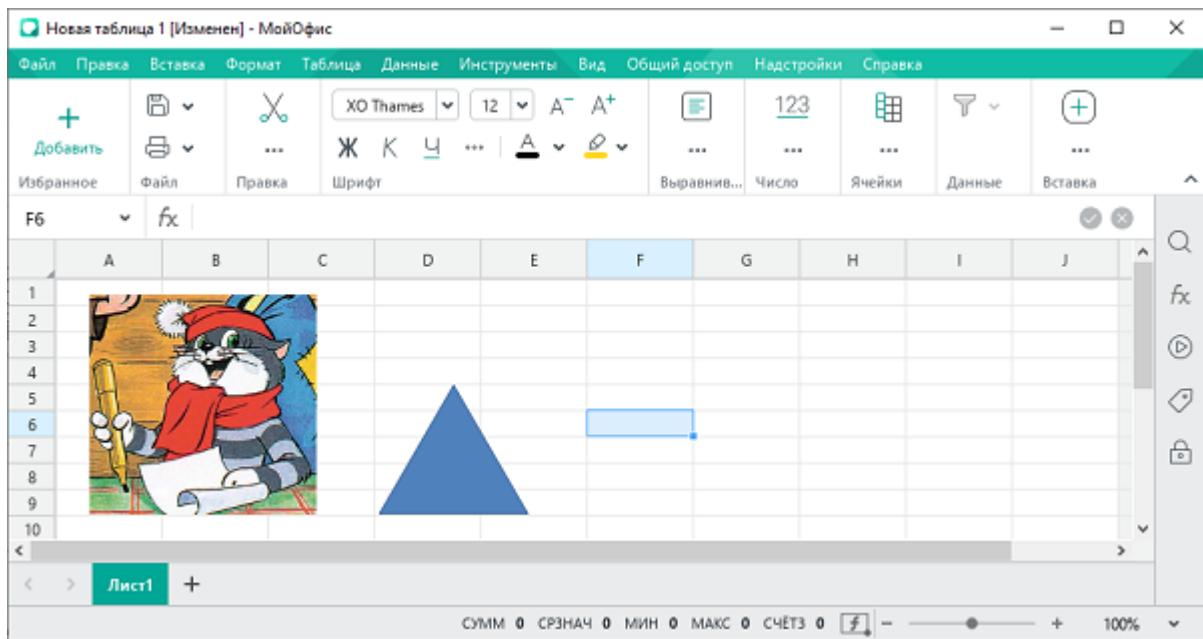


Рисунок 37 – Графические объекты в табличном документе

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
        -- Image
    else
        -- Shape
    end
end
```

Вариант 2: перечисление изображений в табличном документе

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

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 табличный документ

В текущей версии не поддерживается.

4.2.6 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам [Table](#) осуществляется из [Blocks](#) (см. Рисунок 38).

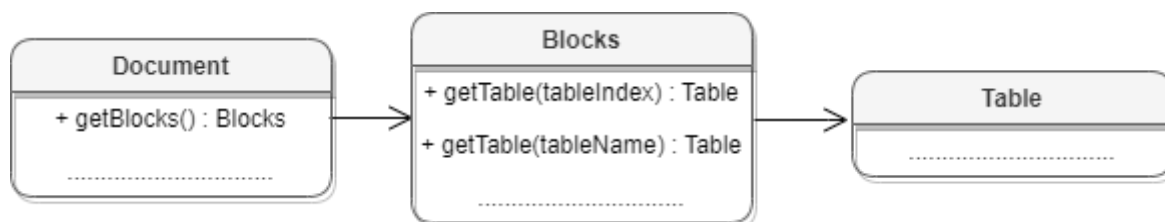


Рисунок 38 – Объектная модель для работы с таблицами

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")
```

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()
```

4.2.7 Работа со сводными таблицами

Сводная таблица - инструмент обработки данных, служащий для их обобщения и удобства обработки. Схема взаимодействия объектов, связанных со сводными таблицами, приведена на рисунке 39.

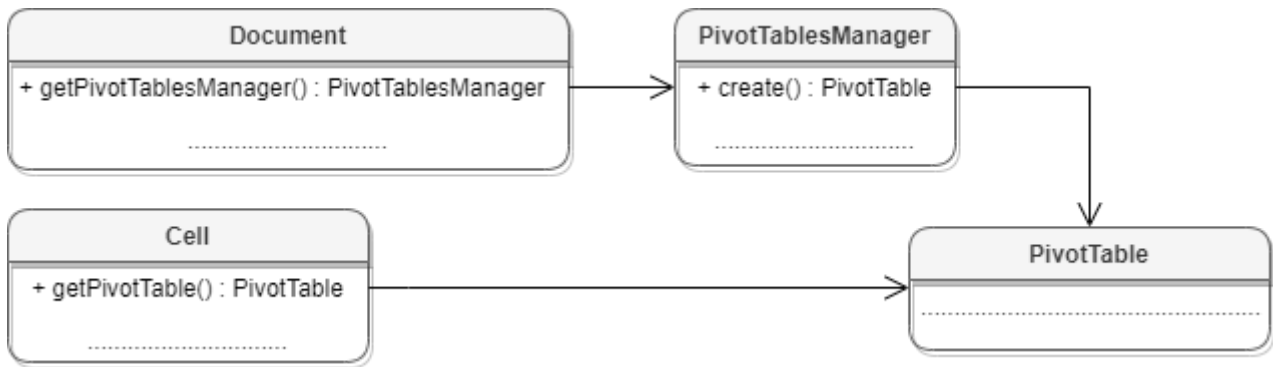


Рисунок 39 – Сводные таблицы

4.2.7.1 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())
```

4.2.7.2 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()
```

4.2.7.3 Получение неподдерживаемых свойств сводной таблицы

Для получения неподдерживаемых свойств сводной таблицы используется метод [PivotTable::getUnsupportedFeatures\(\)](#).

Пример:

```
-- Получаем неподдерживаемые свойства сводной таблицы
local unsupportedFeatures = pivotTable:getUnsupportedFeatures()
for featureIndex = 0, unsupportedFeatures:size() - 1 do
    print(unsupportedFeatures[featureIndex])
end
```

4.2.7.4 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()
```

4.2.7.5 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())
```

4.2.7.6 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()
```

4.2.7.7 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
    print(pageFields[fieldIdx].fieldName)
    print(pageFields[fieldIdx].fieldAlias)
    print(pageFields[fieldIdx].subtotalAlias)
end
```

4.2.7.8 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].fieldName)
    print(valueFields[fieldIdx].fieldAlias)
    print(valueFields[fieldIdx].subtotalAlias)
end
```

4.2.7.9 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
    print(rowFields[fieldIdx].fieldName)
    print(rowFields[fieldIdx].fieldAlias)
    print(rowFields[fieldIdx].subtotalAlias)
end
```

4.2.7.10 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
```

4.2.7.11 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)
```

4.2.7.12 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()
```

4.3 Работа с макросами

Таблица `DocumentAPI.Scripts` предоставляет доступ к списку макросов документа. На рисунке 40 изображена объектная модель таблиц, относящихся к работе с макросами.

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

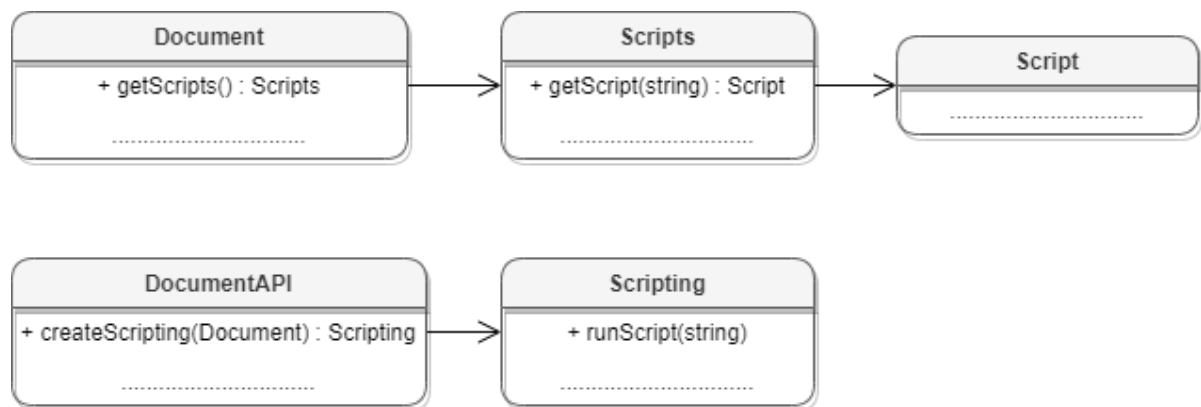


Рисунок 40 – Объектная модель таблиц для работы с макросами

Доступны следующие операции:

- [получение списка макросов](#);
- [добавление макроса](#);
- [получение макроса по имени](#);
- [удаление макроса](#);
- [запуск макроса](#).

4.4 Работа с именованными диапазонами

Именованный диапазон – это диапазон ячеек или формула, которым присвоено имя. Преимуществом именованного диапазона является его информативность. Именованные диапазоны упрощают работу с ячейками, также их удобно использовать при работе с формулами. На данный момент доступна возможность работы с именованными диапазонами, представляющими собой ссылки на диапазоны ячеек. Доступ к именованным диапазонам осуществляется посредством методов [Document::getNamedExpressions\(\)](#) и [Table::getNamedExpressions\(\)](#) (см. Рисунок 41).

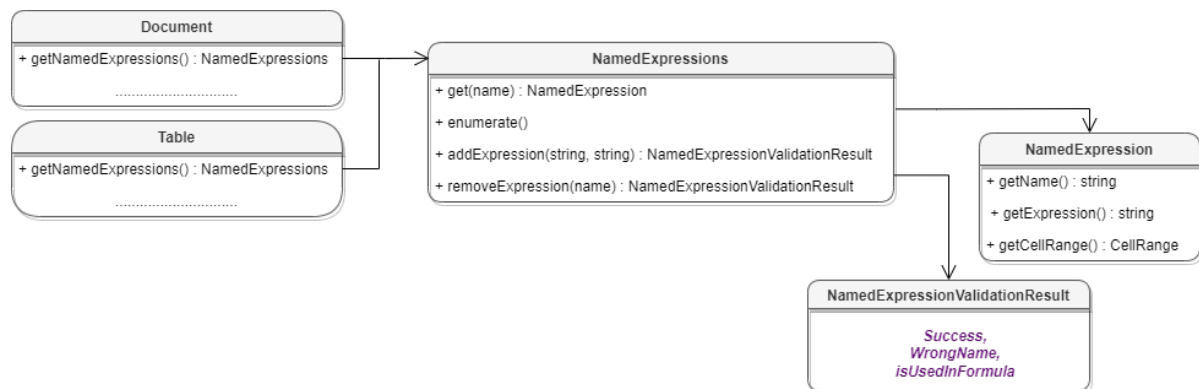


Рисунок 41 – Таблицы для работы с именованными диапазонами

Именованные диапазоны могут быть использованы в странице табличного документа или таблице текстового документа.

4.4.1 Доступ к именованным диапазонам

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

Пример для текстового документа:

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

Пример для табличного документа:

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

4.4.2 Получение свойств именованного диапазона

Пример:

```
local sheet = document:getBlocks():getTable(0)
local namedExpressions = sheet:getNamedExpressions()
// Получить именованное выражение с именем "Alice_Age"
local namedExpression = namedExpressions:get("Alice_Age")
// Далее используются поля структуры NamedExpression:
local name = namedExpression:getName()
local formula = namedExpression:getExpression()
local range = namedExpression:getCellRange()
```

4.4.3 Получение коллекции именованных диапазонов

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

Пример:

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

4.4.4 Добавление именованного диапазона

Для добавления именованного диапазона используется метод [NamedExpressions::addExpression\(\)](#). В качестве результата операции метод возвращает значение типа [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
```

4.4.5 Удаление именованного диапазона

Для удаления именованного диапазона используется метод [NamedExpressions::removeExpression\(\)](#). В качестве результата операции метод возвращает значение типа [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
```

4.4.6 Получение параметров именованного диапазона

Для получения детальной информации об именованном диапазоне используются методы [NamedExpression:getName](#), [NamedExpression:getExpression](#), [NamedExpression:getCellRange](#).

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

4.5 Работа со строками и столбцами таблиц

4.5.1 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::clearColumnnGroups\(\)](#).

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

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

4.5.2 Управление видимостью строк / колонок

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

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

4.6 Работа с ячейками таблиц

4.6.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 42):

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

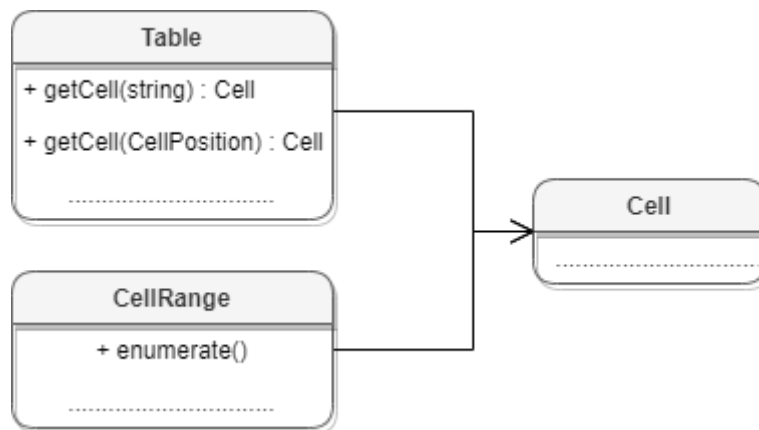


Рисунок 42 – Объектная модель для работы с ячейками таблиц

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
```


МойОфис

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()
```

4.6.2 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)
```

4.6.3 Форматирование границ ячеек

Для оформления границ ячеек используется таблица [DocumentAPI.Borders](#) (см. Рисунок 43). 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.

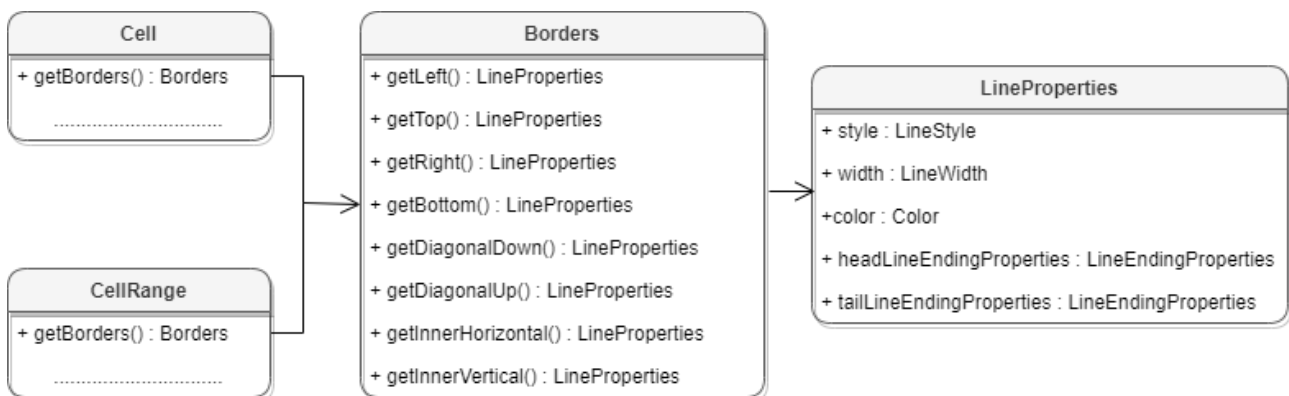


Рисунок 43 – Таблицы для работы с границами ячеек

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

- получить ячейку [DocumentAPI.Cell](#) или область ячеек [DocumentAPI.CellRange](#);

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

Пример настройки границ ячеек:

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

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

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

--установка внешних границ
borders:setOuter(lineProp)

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

4.6.4 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()
```

МойОфис

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 [CellRange.unmerge\(\)](#).

Пример:

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

5 Глобальные функции DocumentAPI

5.1 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")
```

5.2 Функция DocumentAPI.createScripting

Функция `DocumentAPI.createScripting` возвращает таблицу [Scripting](#). В качестве параметра используется текущий документ.

Пример:

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

6 Справочник таблиц DocumentAPI

6.1 Таблица DocumentAPI.AbsoluteFrame

Таблица `DocumentAPI.AbsoluteFrame` описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 44). Предназначена для получения и изменения свойств позиции медиаобъектов (см. [Image:getFrame\(\)](#), [MediaObject:getFrame\(\)](#)). Используется в табличном документе.

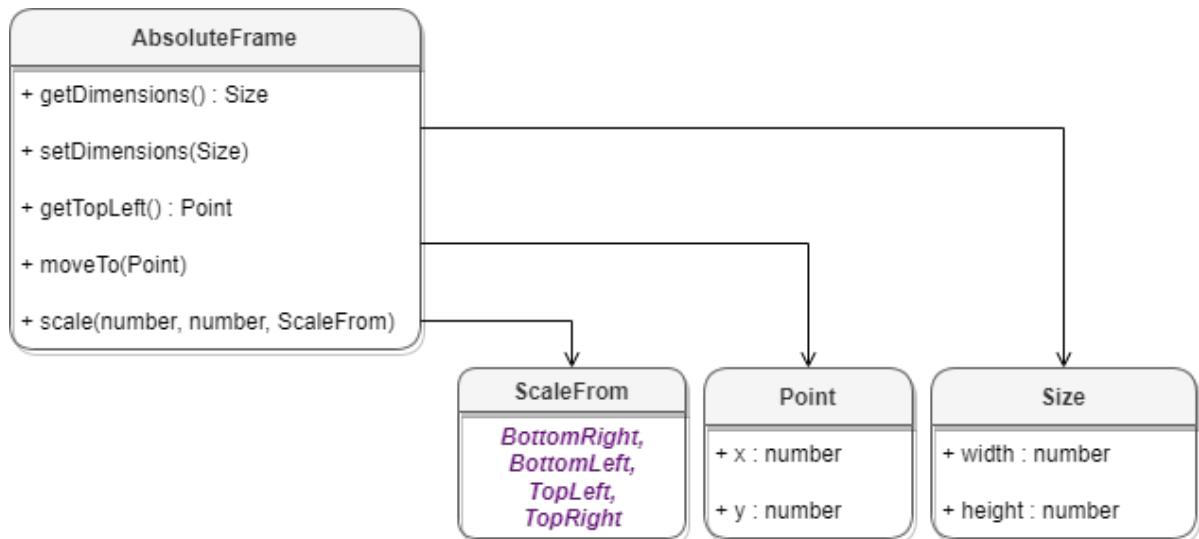


Рисунок 44 – Объектная модель таблицы `DocumentAPI.AbsoluteFrame`

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
```

6.1.1 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
```

6.1.2 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
```

6.1.3 Метод `AbsoluteFrame:scale`

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

Вызов:

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

Параметры:

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

6.1.4 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(100, 100)
end
```

6.1.5 Метод `AbsoluteFrame:getDimensions`

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

6.2 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 8.

Таблица 8 – 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()
```


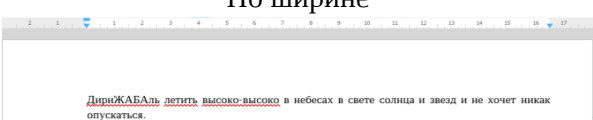
```
accountingCellFormatting.decimalPlaces = 3
accountingCellFormatting.symbol = 'Руб'

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

6.3 Таблица DocumentAPI.Alignment

В таблице 9 представлены варианты выравнивания текста по горизонтали в текстовом редакторе или содержимого ячеек в табличном редакторе.

Таблица 9 – 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:

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

6.4 Таблица DocumentAPI.Application

Таблица DocumentAPI.Application предоставляет интерфейс для создания и загрузки документов. Допустимо использование только одного объекта Application для всего сеанса обработки документа.

На рисунке 45 изображена схема взаимодействия таблиц `DocumentAPI.Application` и [DocumentAPI.Document](#).

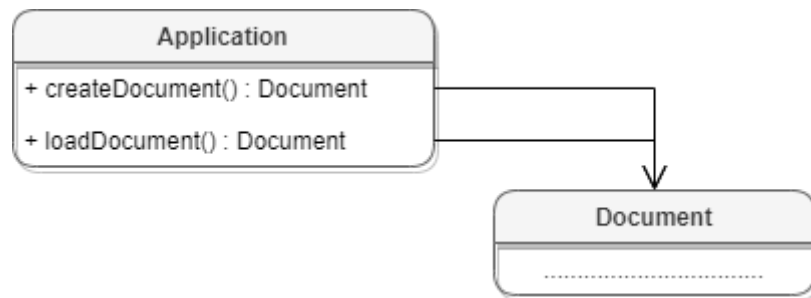


Рисунок 45 – Объектная модель таблиц для работы с Application

6.4.1 Метод `application:createDocument`

Метод `application.createDocument` создает новый документ с типом [DocumentAPI.DocumentType](#), либо [DocumentAPI.DocumentSettings](#).

Примеры:

```
local doc = application:createDocument(DocumentAPI.DocumentType_Text)
doc:saveAs("NewTextDocument.xodt")
```

```
local settings = DocumentAPI.DocumentSettings()
settings.documentType = DocumentAPI.DocumentType_Workbook
local doc = application:createDocument(settings)
doc:saveAs("NewSheetDocument.xlsx")
```

6.4.2 Метод `application:loadDocument`

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

Существуют следующие варианты реализации метода:

```
application:loadDocument(path: string)
application:loadDocument(path: string, loadSettings: LoadDocumentSettings)
```

Примеры использования метода `loadDocument` приведены в разделах [Создание и открытие текстового документа](#) и [Создание и открытие табличного документа](#).

Примеры загрузки текстового документа:

```
local doc = application:loadDocument("sample.docx")
```

```
local docSettings = DocumentAPI.DocumentSettings()  
docSettings.documentType = DocumentAPI.DocumentType_Text  
local loadSettings = DocumentAPI.LoadDocumentSettings()  
loadSettings.commonDocumentSettings = docSettings  
local doc = application:loadDocument("sample.docx", loadSettings)
```

Примеры загрузки табличного документа:

```
local doc = application:loadDocument("sample.xlsx")
```

```
local docSettings = DocumentAPI.DocumentSettings()  
docSettings.documentType = DocumentAPI.DocumentType_Workbook  
local loadSettings = DocumentAPI.LoadDocumentSettings()  
loadSettings.commonDocumentSettings = docSettings  
local doc = application:loadDocument("sample.xlsx", loadSettings)
```

6.5 Таблица DocumentAPI.Blocks

Таблица `DocumentAPI.Blocks` обеспечивает доступ к блокам [DocumentAPI.Block](#) документа или диапазона документа (см. Рисунок 46). Таблица `DocumentAPI.Blocks` может быть получена вызовом метода [Document:getBlocks](#) или [HeaderFooter:getBlocks](#).

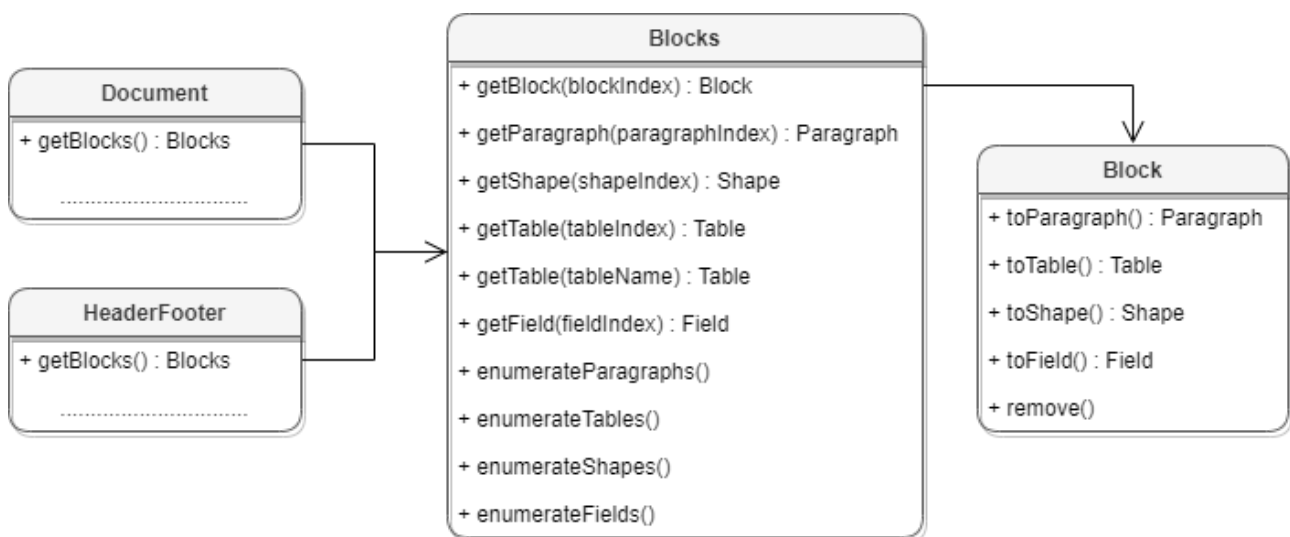


Рисунок 46 – Объектная модель таблицы `DocumentAPI.Blocks`

6.5.1 Метод `Blocks:getBlock`

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

Пример:

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

6.5.2 Метод `Blocks:getParagraph`

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

Пример:

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

6.5.3 Метод `Blocks:getTable`

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

Пример:

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

В качестве параметра метода также можно указать имя таблицы.

Пример:

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

6.5.4 Метод `Blocks:getShape`

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

Пример:

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

6.5.5 Метод `Blocks:getField`

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

Пример:

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

6.5.6 Метод `Blocks:enumerate`

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

Пример:

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

6.5.7 Метод `Blocks:enumerateParagraphs`

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

Пример:

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

6.5.8 Метод `Blocks:enumerateTables`

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

Пример:

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

6.5.9 Метод `Blocks:enumerateShapes`

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

Пример:

```
for shape in document:getBlocks():enumerateShapes() do
    print(shape:getShapeProperties())
end
```

6.5.10 Метод `Blocks:enumerateFields`

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

Пример:

```
for field in document:getBlocks():enumerateFields() do
    print(field:getRange():extractText())
end
```

6.6 Таблица DocumentAPI.Block

Таблица `DocumentAPI.Block` является базовой для всех блоков документа. От нее наследуются таблицы [Paragraph](#), [Table](#), [Shape](#), [Field](#) (см. Рисунок 47).

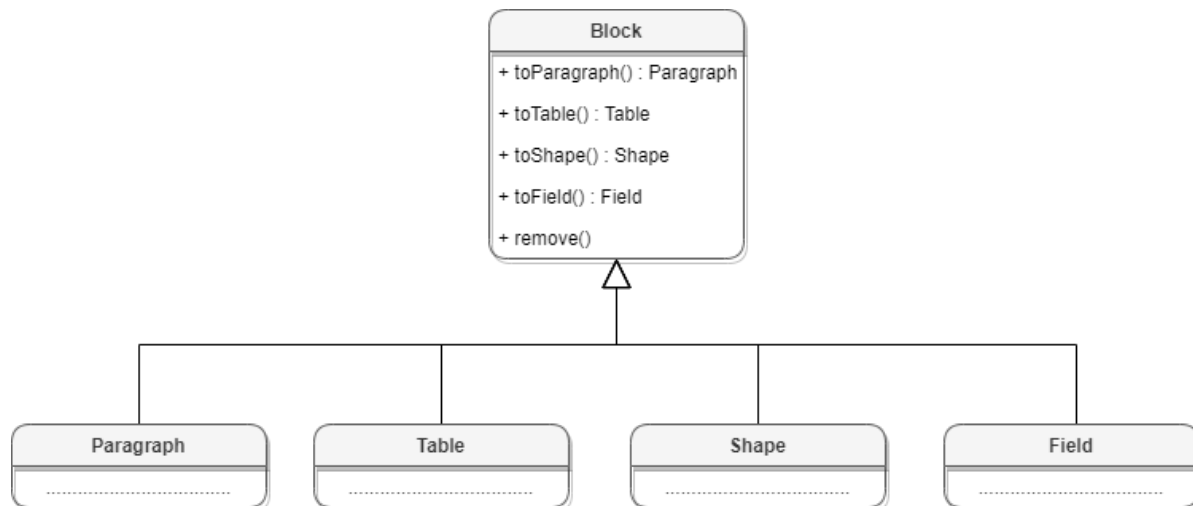


Рисунок 47 – Объектная модель таблицы `DocumentAPI.Block`

6.6.1 Методы `toParagraph`, `toTable`, `toShape`, `toField`

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

Пример:

```
local paragraph = document:getBlocks():getBlock(0):toParagraph()
local para_props = paragraph:getParagraphProperties()
```

6.6.2 Метод `Block.getRange`

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

Пример:

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

6.6.3 Метод `Block.remove`

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

Пример:

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

6.6.4 Метод `Block.getSection`

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



Пример:

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

6.7 Таблица `DocumentAPI.Borders`

Таблица `DocumentAPI.Borders` предназначена для оформления границ отдельной ячейки таблицы (см. таблицу 10). Параметры линии, такие как тип линии, ее ширина и цвет, задаются с помощью таблицы [DocumentAPI.LineProperties](#).

Таблица 10 – Описание методов таблицы `DocumentAPI.Borders`

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

Метод	Описание
<code>Borders:getInnerVertical</code>	Получение внутренних вертикальных границ 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 = borders:setLeft(LineProperties)
borders = borders:setRight(LineProperties)
borders = borders:setTop(LineProperties)
borders = borders:setBottom(LineProperties)

borders = cell:setBorders(borders)
```

6.8 Таблица `DocumentAPI.Bookmarks`

Предоставляет доступ к операциям с закладками в документе.

6.8.1 Метод `Bookmarks:getBookmarkRange`

Возвращает объект [DocumentAPI.Range](#) для дальнейшей работы с содержимым закладки (bookmark).

Пример:

```
local bookmarks = document:getBookmarks()
local bookmark = bookmarks:getBookmarkRange("Bookmark")
bookmark:replaceText("Lua")
```

6.8.2 Метод `Bookmarks:removeBookmark`

Удаляет закладку по ее названию.

Пример:

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

6.9 Таблица DocumentAPI.Cell

Таблица `DocumentAPI.Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 48).

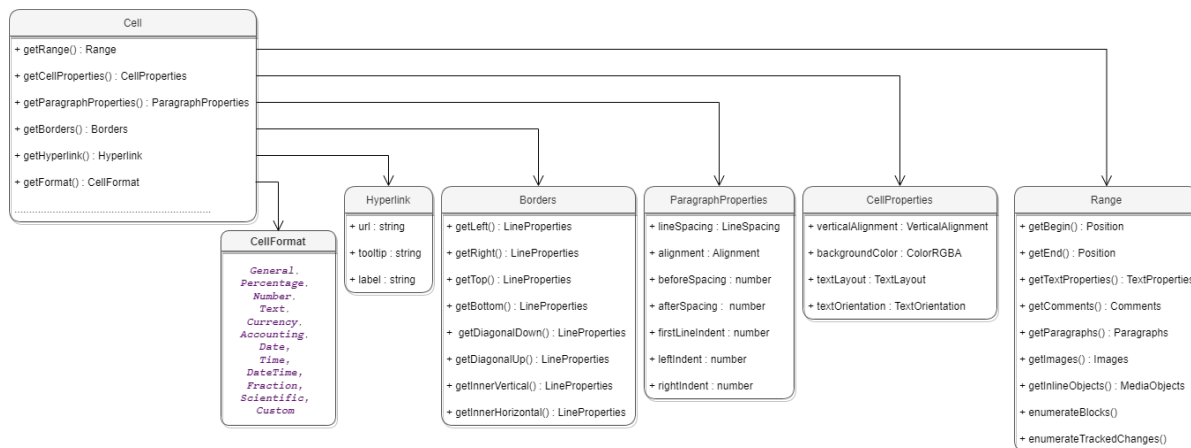


Рисунок 48 – Объектная модель ячейки таблиц

6.9.1 Метод `Cell:getRange`

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

Пример:

```
table = document:getBlocks():getTable(0)
cell = table:getCell("B2")
cellRange = cell:getRange()
```

6.9.2 Метод `Cell:setBorders`

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

6.9.3 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
```

6.9.4 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
```

6.9.5 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: Экспоненциальный
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
```

6.9.6 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
```

6.9.7 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](#).

6.9.8 Метод `Cell:unmerge`

Разъединяет несколько ячеек, которые были объединены ранее.

Пример:

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

6.9.9 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()
```

6.9.10 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")
```

6.9.11 Метод `Cell:getBorders`

Позволяет получить границы ячейки.

Пример:

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

6.9.12 Метод Cell:getRawValue

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

Пример:

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

6.9.13 Метод Cell:getCustomFormat

Возвращает строку формата ячейки.

Пример:

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

6.9.14 Метод Cell:setCustomFormat

Устанавливает формат ячейки.

Пример:

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

6.9.15 Метод Cell:setBool

Устанавливает для ячейки значение логического типа.

Пример:

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

6.9.16 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)
```

6.9.17 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)
```

6.9.18 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()
```

6.9.19 Метод Cell:getCellProperties

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

Пример:

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

6.9.20 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)
```

6.9.21 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)
```


6.9.22 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)
```

6.9.23 Метод Cell:getPivotTable

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

Пример:

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

6.10 Таблица DocumentAPI.CellFormat

По умолчанию при создании документа всем ячейкам присваивается формат «Общий». Полный список форматов представлен в таблице 11.

Таблица 11 – Поддерживаемые форматы ячеек таблицы

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

Наименование константы	Описание
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 в ячейке, а в строке формул остаются в том виде, в котором они были введены.</p>
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_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</p>

Наименование константы	Описание
	<p>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:

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

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

6.11 Таблица DocumentAPI.CellProperties

Таблица DocumentAPI.CellProperties предназначена для форматирования содержимого в ячейках таблицы. Описание полей DocumentAPI.CellProperties представлено в таблице 12.

МойОфис

Для задания свойств ячейки используется метод [Cell.setCellProperties\(CellProperties\)](#). Для получения свойств ячейки используется метод [Cell.getCellProperties\(\)](#). Иерархия таблиц и полей DocumentAPI.CellProperties отображена на рисунке 49.

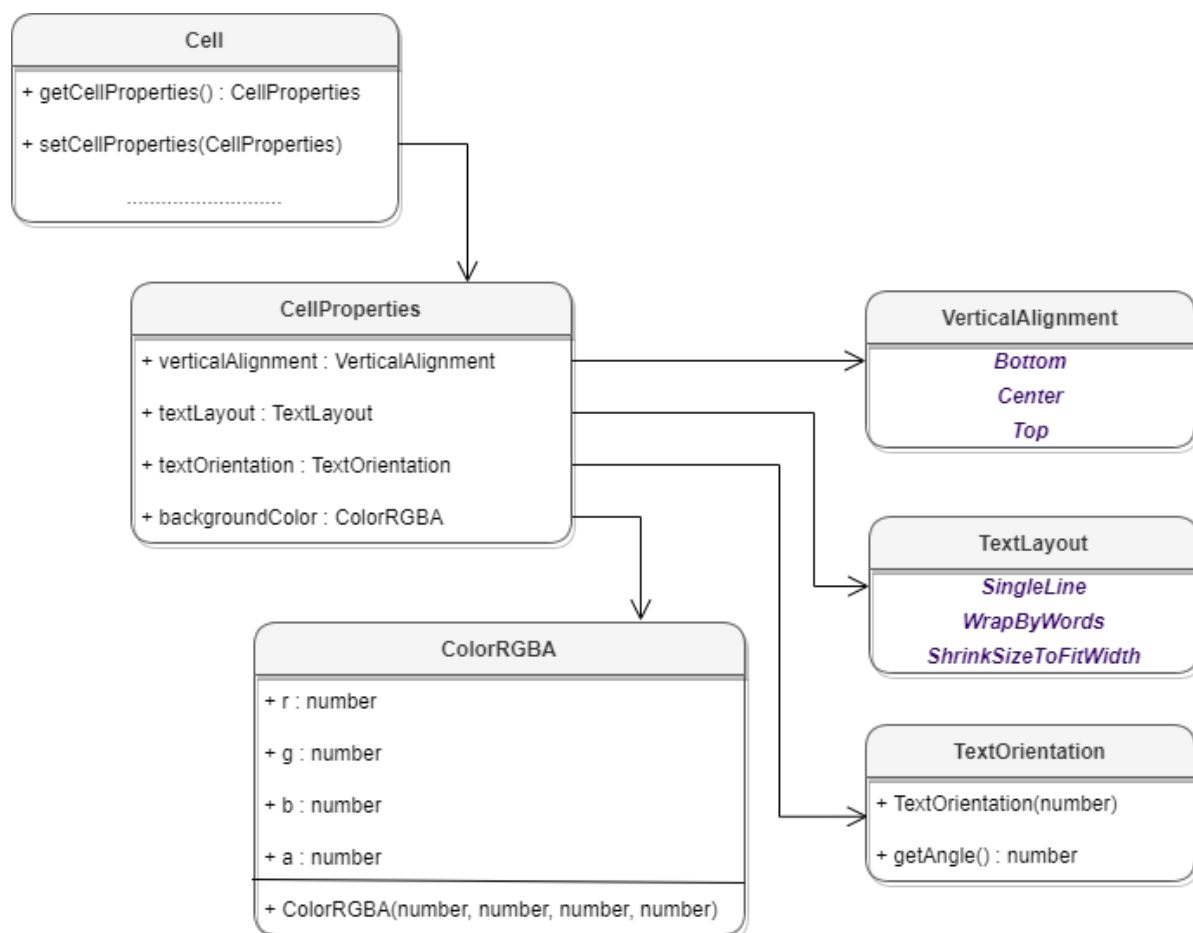


Рисунок 49 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 12 – Описание полей таблицы DocumentAPI.CellProperties

Поле	Тип	Значение
CellProperties.verticalAlignment	VerticalAlignment	Вертикальное выравнивание в ячейке
CellProperties.textLayout	TextLayout	Способ отображения значения ячейки
CellProperties.backgroundColor	ColorRGBA	Цвет 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 cellProperties = cell:getCellProperties()

cellProperties.verticalAlignment = DocumentAPI.VerticalAlignment_Center
cellProperties.textLayout = DocumentAPI.TextLayout_ShrinkSizeToFitWidth
cellProperties.backgroundColor = DocumentAPI.ColorRGBA(255, 255, 0, 255)
cellProperties.textOrientation = DocumentAPI.TextOrientation(45)

cell:setCellProperties(cellProperties)
```

6.12 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
```

6.12.1 Поле `CellPosition.column`

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

Пример:

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

6.12.2 Поле `CellPosition.row`

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

Пример:

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

6.12.3 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)
```

6.13 Таблица `DocumentAPI.CellRange`

Таблица `DocumentAPI.CellRange` описывает диапазон ячеек таблицы.

Примеры:

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

```
local cellRangePosition = DocumentAPI.CellRangePosition(0, 0, 5, 5)
```

```
local cellRange = table:getCellRange(cellRangePosition)
```

6.13.1 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
```

6.13.2 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
```

6.13.3 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
```

6.13.4 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
```

6.13.5 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
```

6.13.6 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)
```

6.13.7 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)
```

6.13.8 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()  
print(cellProperties.backgroundColor.r)
```

6.13.9 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)
```


6.13.10 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.

Пример:

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

6.13.11 Метод `CellRange:unmerge`

Метод разъединяет ранее объединенные ячейки.

Пример:

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

6.14 Таблица `DocumentAPI.CellRangePosition`

Таблица `DocumentAPI.CellRangePosition` представляет положение диапазона ячеек в таблице. 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
<code>bottomRight</code>	CellPosition	Содержит позицию правой нижней ячейки таблицы прямоугольного диапазона

Примеры:

```
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)
```

6.14.1 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)]
```

6.15 Таблица `DocumentAPI.Chart`

Таблица `DocumentAPI.Chart` представляет диаграмму в табличном документе и описывает все ее элементы (заголовок, легенда, тип, данные, диапазон и т.д.). Объектная модель `DocumentAPI.Chart` приведена на Рис. 50.

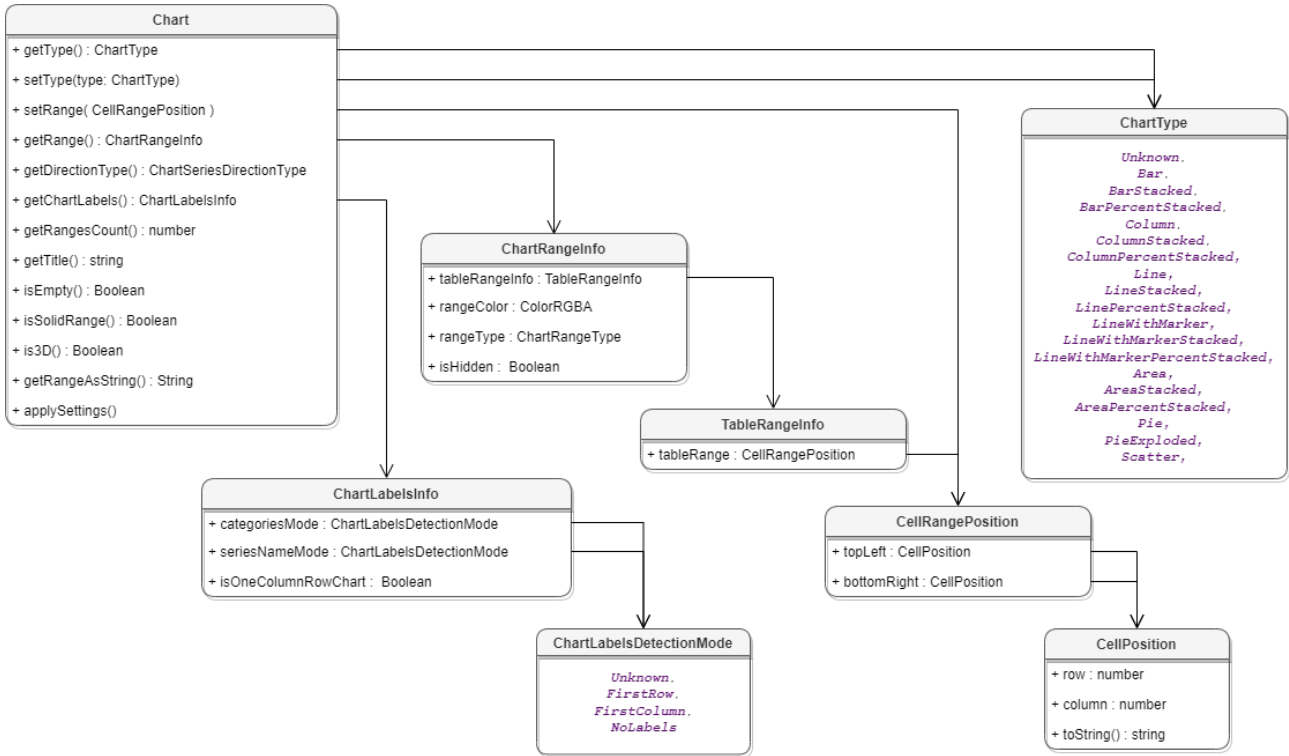


Рисунок 50 – Объектная модель таблицы DocumentAPI . Chart

6.15.1 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())
  
```

6.15.2 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())
  
```

6.15.3 Метод `Chart:getRangesCount`

Метод возвращает количество серий диаграммы.

Пример:

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

6.15.4 Метод `Chart:getRange`

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

Пример:

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

6.15.5 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)
```

6.15.6 Метод `Chart:getTitle`

Метод возвращает заголовок диаграммы.

Пример:

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

6.15.7 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.

6.15.8 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())
```

6.15.9 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())
```

6.15.10 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())
```

6.15.11 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())
```

6.15.12 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)
```

6.15.13 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())
```

6.15.14 Метод Chart:applySettings

Метод позволяет обновить параметры 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)
```

6.16 Таблица `DocumentAPI.ChartLabelsDetectionMode`

Таблица `DocumentAPI.ChartLabelsDetectionMode` описывает режимы автоматического определения меток диаграмм. Описание полей таблицы представлено в таблице 14.

Таблица 14 – 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)
```

6.17 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 15.

Таблица 15 – Описание полей таблицы `DocumentAPI.ChartLabelsInfo`

Поле	Описание	Тип
<code>categoriesMode</code>	Режим автоматического определения меток для категорий.	DocumentAPI.ChartLabelsDetectionMode

Поле	Описание	Тип
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)
```

6.18 Таблица DocumentAPI.Charts

Таблица `DocumentAPI.Charts` обеспечивает доступ к списку диаграмм (см. Рисунок 51) табличного документа. Доступ к списку диаграмм осуществляется с помощью метода [Table.getCharts\(\)](#).

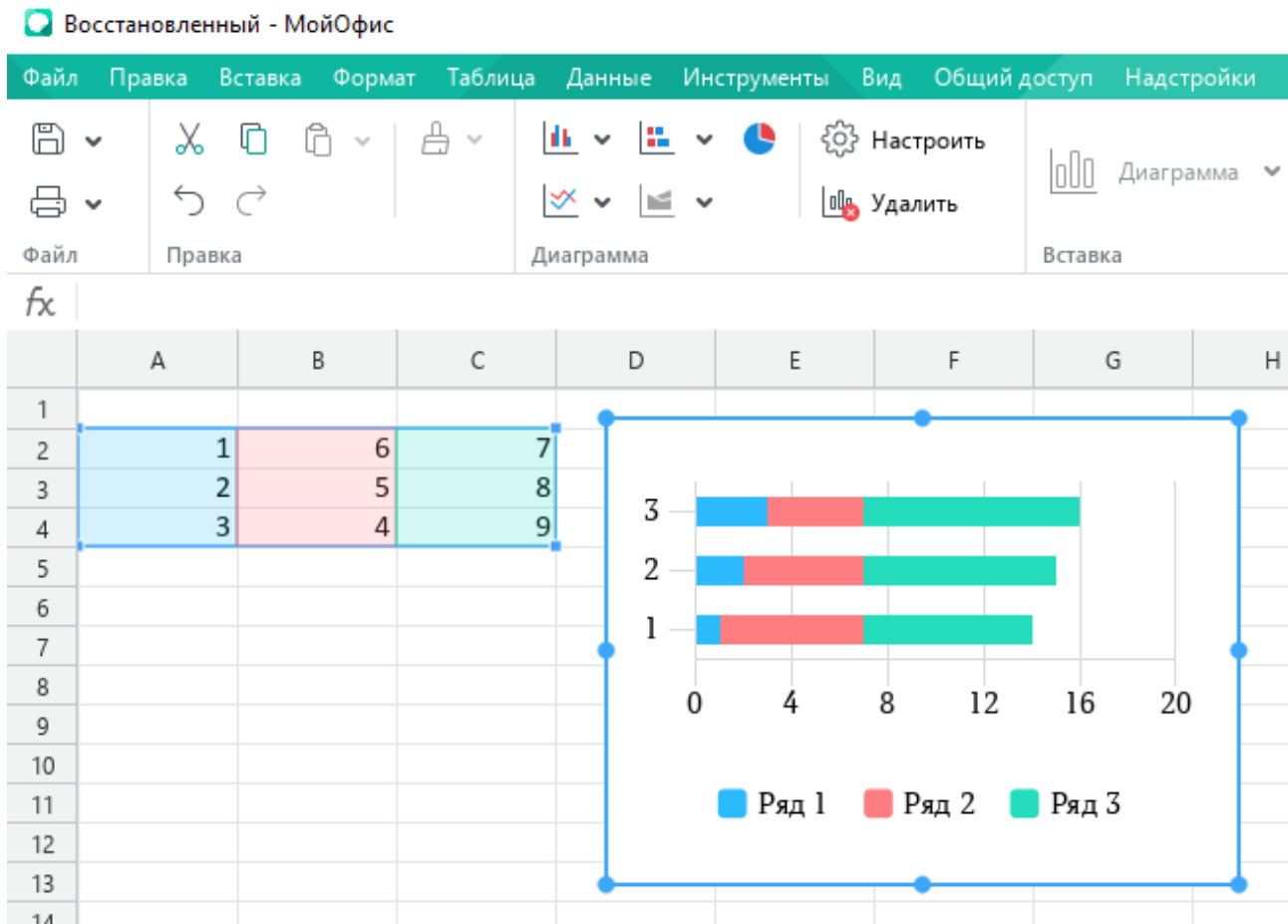


Рисунок 51 – Пример отображения диаграммы в МойОфис Таблица.

6.18.1 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())
```

6.18.2 Метод Charts:getChart

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

Пример:

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

6.18.3 Метод `Charts:getChartIndexByDrawingIndex`

Метод возвращает индекс диаграммы по индексу отрисовки `drawingIndex`.

Пример:

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

6.19 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 16.

Таблица 16 – Описание полей таблицы `DocumentAPI.ChartRangeInfo`

Поле	Описание	Тип
<code>tableRangeInfo</code>	Исходный диапазон ячеек для серии	DocumentAPI.TableRangeInfo
<code>rangeColor</code>	Цвет для отрисовки серии	DocumentAPI.ColorRGBA
<code>isHidden</code>	Задаёт видимость серии диаграммы	<code>Boolean</code>
<code>rangeType</code>	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)
```

6.20 Таблица `DocumentAPI.ChartRangeType`

Таблица `DocumentAPI.ChartRangeType` описывает тип диапазона исходных данных диаграммы. Описание полей таблицы представлено в таблице 17.

Таблица 17 – 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])
```

6.21 Таблица DocumentAPI.ChartSeriesDirectionType

Таблица DocumentAPI.ChartSeriesDirectionType описывает направление серий диаграмм. Описание полей таблицы представлено в таблице 18.

Таблица 18 – 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())
```

6.22 Таблица DocumentAPI.ChartType

Таблица DocumentAPI.ChartType описывает все поддерживаемые типы диаграмм. Описание полей таблицы представлено в таблице 19.

Таблица 19 – Описание полей таблицы DocumentAPI.ChartType

Поле	Описание
DocumentAPI.ChartType_Unknown	Неопределенный тип

Поле	Описание
DocumentAPI.ChartType_Bar	Линейчатая диаграмма с группировкой
DocumentAPI.ChartType_BarStacked	Линейчатая диаграмма с накоплением
DocumentAPI.ChartType_BarPercentStacked	Линейчатая нормированная диаграмма с накоплением
DocumentAPI.ChartType_Column	Гистограмма с группировкой
DocumentAPI.ChartType_ColumnStacked	Гистограмма с накоплением
DocumentAPI.ChartType_ColumnPercentStacked	Нормированная гистограмма с накоплением
DocumentAPI.ChartType_Line	Стандартный график
DocumentAPI.ChartType_LineStacked	График с накоплением
DocumentAPI.ChartType_LinePercentStacked	Нормированный график с накоплением
DocumentAPI.ChartType_LineWithMarker	Стандартный график с маркерами
DocumentAPI.ChartType_LineWithMarkerStacked	График с накоплением и маркерами
DocumentAPI.ChartType_LineWithMarkerPercentStacked	Нормированный график с накоплением и маркерами
DocumentAPI.ChartType_Area	Стандартная диаграмма с областями
DocumentAPI.ChartType_AreaStacked	Диаграмма с областями с накоплением
DocumentAPI.ChartType_AreaPercentStacked	Нормированная диаграмма с областями с накоплением
DocumentAPI.ChartType_PieAreaPercentStacked	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())
```

6.23 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)
```

6.23.1 Метод Color:getRGBAColor

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

Пример:

```
local rgbaColor = color:getRGBAColor()
if rgbaColor then
    print(rgbaColor.r, rgbaColor.g, rgbaColor.b, rgbaColor.a)
end
```

6.23.2 Метод Color:getThemeColorID

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

Пример:

```
local themeColorID = color:getThemeColorID()
if themeColorID then
    print(themeColorID)
end
```

6.24 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 20.

Таблица 20 – Описание полей 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.

Поле	Тип	Описание
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)
```

6.25 Таблица DocumentAPI.Comments

Таблица `DocumentAPI.Comments` содержит коллекцию комментариев диапазона (см. Рисунок 52).

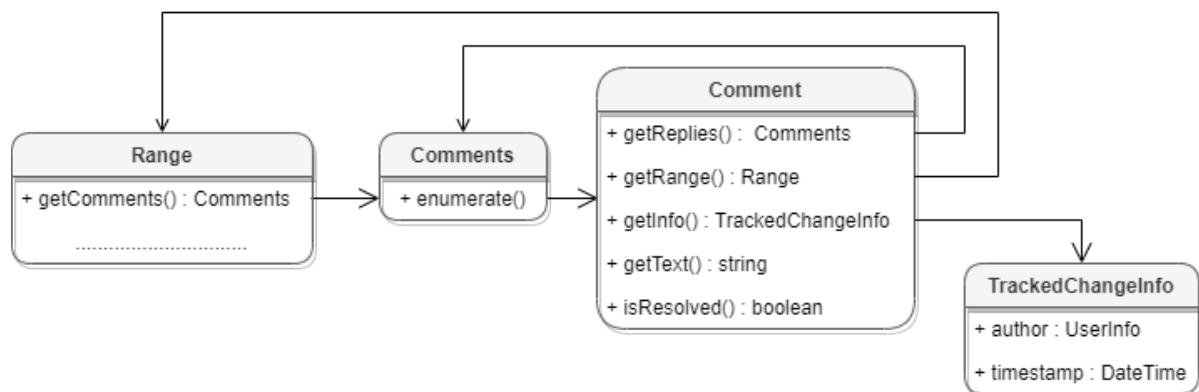


Рисунок 52 – Объектная модель таблиц для работы с комментариями

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
```

6.25.1 Метод `Comments:enumerate`

Метод возвращает коллекцию комментариев всего документа.

Пример:

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

6.26 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](#).

6.26.1 Метод `Comment:getRange`

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

Пример:

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

6.26.2 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
```

6.26.3 Метод `Comment:getInfo`

Метод предоставляет доступ к информации о комментарии [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
```

6.26.4 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
```

6.26.5 Метод `Comment:getReplies`

Метод предоставляет доступ к ответам на комментарии. Ответы находятся в такой же таблице [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
```

6.27 Таблица `DocumentAPI.CurrencyCellFormatting`

Таблица содержит параметры для денежного формата ячеек таблицы. Описание полей таблицы `DocumentAPI.CurrencyCellFormatting` представлено в таблице 21.

Таблица 21 – 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
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())
```

6.28 Таблица DocumentAPI.CurrencySignPlacement

Варианты размещения знака валюты представлены в таблице 22. Данный тип используется в поле `currencyFormat` таблицы [DocumentAPI.LocaleInfo](#), а также в поле `currencySignPlacement` таблицы [DocumentAPI.CurrencyCellFormatting](#) (см. пример в ее описании).

Таблица 22 – Описание полей таблицы DocumentAPI.CurrencySignPlacement

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

6.29 Таблица DocumentAPI.DatePatterns

Форматы даты представлены в таблице 23. Пример использования см. в главе [DocumentAPI.DateTimeCellFormatting](#).

Таблица 23 – Форматы даты

Наименование константы	Описание
<code>DocumentAPI.DatePatterns_DayMonthTextLongYearLong</code>	'mmmm dd, yyyy' для языка en_US
<code>DocumentAPI.DatePatterns_FullDate</code>	'день недели, mmmm dd, yyyy' для языка en_US
<code>DocumentAPI.DatePatterns_DayMonthNumberLongYearLong</code>	'mm/dd/yyyy' для языка en_US
<code>DocumentAPI.DatePatterns_DayMonthNumberLongYearShort</code>	'm/dd/yy' для языка en_US
<code>DocumentAPI.DatePatterns_DayMonthNumberShortYearShort</code>	'dd-mmm' для языка en_US
<code>DocumentAPI.DatePatterns_DayMonthTextShort</code>	'mmm-yy' для языка en_US
<code>DocumentAPI.DatePatterns_DayMonthTextShortYearShort</code>	'mmm dd, yy' для языка en_US
<code>DocumentAPI.DatePatterns_DayMonthYearLongNonLocalizableHyphen</code>	Нелокализуемый шаблон 'dd-mm-yyyy'
<code>DocumentAPI.DatePatterns_DayMonthYearLongNonLocalizableSlash</code>	Нелокализуемый шаблон 'dd/mm/yyyy'

6.30 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 24.

Таблица 24 – Описание полей таблицы `DocumentAPI.DateTime`

Поле	Тип	Описание
<code>DocumentAPI.DateTime.year</code>	Дата	Год
<code>DocumentAPI.DateTime.month</code>	Дата	Месяц
<code>DocumentAPI.DateTime.day</code>	Дата	День
<code>DocumentAPI.DateTime.hour</code>	Время	Часы
<code>DocumentAPI.DateTime.minute</code>	Время	Минуты
<code>DocumentAPI.DateTime.second</code>	Время	Секунды

6.31 Таблица DocumentApi.DateTimeFormat

В таблице 25 представлены варианты масштабирования при печати табличных документов. Используется в качестве параметра метода [CellRange.insertCurrentDateTime\(\)](#).

Таблица 25 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
<code>DocumentAPI.DateTimeFormat_DateTime</code>	Дата/время
<code>DocumentAPI.DateTimeFormat_Date</code>	Дата
<code>DocumentAPI.DateTimeFormat_Time</code>	Время

6.32 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 26.

Таблица 26 – Описание полей таблицы `DocumentAPI.DateTimeCellFormatting`

Поле	Описание
<code>DocumentAPI.DateTimeCellFormatting.dateListID</code>	Формат даты DatePatterns
<code>DocumentAPI.DateTimeCellFormatting.timeListID</code>	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())
```

6.33 Таблица DocumentAPI.document

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

Пример:

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

6.33.1 Метод `document:saveAs`

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

При необходимости, в качестве второго аргумента можно использовать таблицу [DocumentAPI.SaveDocumentSettings](#), которая содержит формат документа [DocumentAPI.DocumentFormat](#), тип документа [DocumentAPI.DocumentType](#) и пароль для защиты документа от несанкционированного доступа.

Примеры использования метода `saveAs` приведены в разделах [Сохранение и экспорт текстового документа](#) и [Сохранение и экспорт табличного документа](#).

6.33.2 Метод `document:exportAs`

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

```
exportAs(String filePath, ExportFormat exportFormat);
```

В настоящее время поддерживается только операция экспорта документа в формат [PDF/A-1b](#).

Примеры использования метода `exportAs` приведены в разделах [Сохранение и экспорт текстового документа](#) и [Сохранение и экспорт табличного документа](#).

6.33.3 Метод `document:merge`

Метод `Document:merge` сравнивает текущий документ с другим документом, который передается в параметре типа [DocumentAPI.Document](#).

Метод возвращает объект [DocumentAPI.Document](#), содержащий результат сравнения в виде отслеживаемых изменений.

Пример:

```
function MergeSample.mergeDocuments(context)
    context.doWithApplication(function(application)
        local docSettings = DocumentAPI.DocumentSettings()
        docSettings.documentType = DocumentAPI.DocumentType_Text
        local loadSettings = DocumentAPI.LoadDocumentSettings()
        loadSettings.commonDocumentSettings = docSettings

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

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

Результат выполнения данного примера (сравнение двух документов, содержащих "1111" и "2222") приведен на рисунке 53.

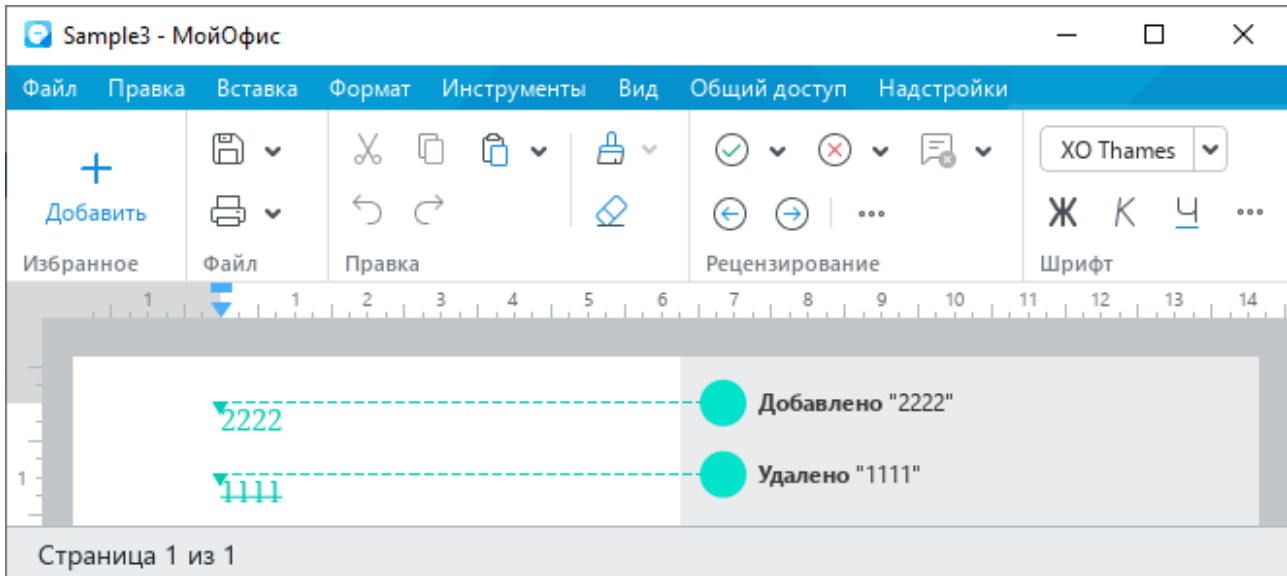


Рисунок 53 – Результат выполнения метода merge

6.33.4 Метод `document:getBlocks`

Метод предоставляет доступ к метатаблице [DocumentAPI.Blocks](#) и далее к отдельным фрагментам (абзацам, таблицам и т. д.), из которых состоит документ.

Пример:

```
local blocks = document:getBlocks()
```

6.33.5 Метод `document:getBookmarks`

Метод предоставляет доступ к таблице закладок [DocumentAPI.Bookmarks](#).

Пример:

```
local bookmarks = document:getBookmarks()
```

6.33.6 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
```

6.33.7 Метод `document:getRange`

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

Пример:

```
local range = document:getRange()
print(range:extractText())
```

6.33.8 Метод `document:isChangesTrackingEnabled`

Метод возвращает текущее состояние отслеживания изменений в текстовом документе (`true` - включены). Подробнее см. в разделе [Рецензирование документов](#).

Пример:

```
local trackingChanges = "Disabled"
if document:isChangesTrackingEnabled() then
    trackingChanges = "Enabled"
end
```

6.33.9 Метод `document:setChangesTrackingEnabled`

Метод управляет состоянием отслеживания изменений в текстовом документе (включены или выключены). Подробнее см. в разделе [Рецензирование документов](#).

Пример:

```
if trackingChanges == "Disabled" then
    document:setChangesTrackingEnabled(true)
    if document:isChangesTrackingEnabled() then
        trackingChanges = "Enabled"
    end
end
```

6.33.10 Метод `document:getComments`

Метод обеспечивает доступ к комментариям текстового документа, возвращает таблицу [DocumentAPI.Comments](#).

Пример:

```
local comments = document:getComments()
for comment in comments:enumerate() do
    print(comment:getRange())
    print(comment:getText())
    print(comment:getInfo().author)
    print(comment:getInfo().timeStamp)
```

```
print(comment:isResolved())
print(comment:getReplies())
end
```

6.33.11 Метод `document:setPageProperties`

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

Пример:

```
local properties = DocumentAPI.PageProperties()
properties.width = 100
properties.height = 200
document:setPageProperties(properties)
```

6.33.12 Метод `document:setFormulaType`

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

Пример:

```
document:setFormulaType(DocumentAPI.FormulaType_A1)
```

6.33.13 Метод `document:getFormulaType`

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

Пример:

```
document:setFormulaType(DocumentAPI.FormulaType_R1C1)
print(document:getFormulaType())
```

6.33.14 Метод `document:setPageOrientation`

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

Пример:

```
document:setPageOrientation(DocumentAPI.PageOrientation_Landscape)
```


6.33.15 Метод `document:enumerateSections`

Возвращает таблицу объектов типа [DocumentAPI.Section](#).

Пример:

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

6.33.16 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
```

6.33.17 Метод `document:setMirroredMarginsEnabled`

Метод позволяет включать и выключать зеркальные поля в документе.

Пример:

```
document:setMirroredMarginsEnabled(true)  
print(document:areMirroredMarginsEnabled())
```

6.33.18 Метод `document:areMirroredMarginsEnabled`

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

Пример:

```
document:setMirroredMarginsEnabled(true)  
print(document:areMirroredMarginsEnabled())
```

6.33.19 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)
```

6.33.20 Метод `document:getNameExpressions`

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

Пример:

```
local namedExpressions = document:getNameExpressions()
```

6.34 Таблица `DocumentAPI.ExportFormat`

В таблице 27 приведены поддерживаемые форматы экспорта документов (см. [document:exportAs\(\)](#)).

Таблица 27 - Форматы экспорта документов

Наименование константы	Описание
<code>DocumentAPI.ExportFormat_PDFA1</code>	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

6.35 Таблица `DocumentAPI.Field`

Таблица `Field` предназначена для реализации некоторых полей, например, содержания.

6.36 Таблица `DocumentAPI.Fill`

Таблица описывает свойства заполнения фигуры: цвет заполнения, путь к изображению фона. Используется в [DocumentAPI.ShapeProperties](#).

6.36.1 Метод `Fill:getColor`

Метод возвращает цвет заполнения [DocumentAPI.Color](#).

6.36.2 Метод `Fill:getUrl`

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

6.36.3 Метод `Fill:isNoFill`

Метод возвращает `true`, если заполнения нет.

6.37 Таблица `DocumentAPI.DocumentSettings`

Таблица `DocumentAPI.DocumentSettings` предоставляет общие настройки документа и используется в [document:createDocument\(\)](#).

Описание полей таблицы `DocumentAPI.DocumentSettings` представлено в таблице 28.

Таблица 28 – Описание полей таблицы `DocumentAPI.DocumentSettings`

Поле	Тип	Описание
<code>DocumentAPI.DocumentSettings.documentType</code>	DocumentType	Тип документа
<code>DocumentAPI.DocumentSettings.userInfo</code>	UserInfo	Информация о пользователе
<code>DocumentAPI.DocumentSettings.localeInfo</code>	LocaleInfo	Информация о локализации
<code>DocumentAPI.DocumentSettings.timeZone</code>	TimeZone	Информация о временной зоне
<code>DocumentAPI.DocumentSettings.formulaType</code>	FormulaType	Система адресации ячеек

6.38 Таблица `DocumentAPI.DocumentFormat`

В таблице 29 приведены поддерживаемые форматы документов, структура используется в поле `documentFormat` таблицы [DocumentAPI.SaveDocumentSettings](#).

Таблица 29 – Форматы документов

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

Наименование константы	Описание
DocumentAPI.DocumentFormat_PDF	Используется для работы с документами в формате Portable Document Format (PDF) версии 1.4. Средствами Document API поддерживается только операция экспорта документа в формат PDF/A-1b
DocumentAPI.DocumentFormat_PDFA	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b). Средствами Document API поддерживается только операция экспорта документа в формат PDF/A-1b

6.39 Таблица DocumentAPI.DocumentType

В таблице 30 приведены поддерживаемые типы документов, используется при создании документа [application:createDocument\(\)](#), [DocumentAPI.DocumentSettings](#).

Таблица 30 - Типы документов

Наименование константы	Описание
DocumentAPI.DocumentType_Text	Используется для работы с текстовыми документами в форматах DOCX, ODT, XODT, TXT
DocumentAPI.DocumentType_Workbook	Используется для работы с табличными документами в форматах XLSX, ODS, XODS
DocumentAPI.DocumentType_Presentation	Используется для работы с презентационными документами в форматах PPTX, ODP. Работа с презентационными документами средствами Document API в настоящий момент не поддерживается

6.40 Таблица DocumentAPI.DSVSettings

Таблица DocumentAPI.DSVSettings предоставляет настройки, необходимые для работы с файлами CSV (comma-separated value) и DSV (delimiter-separated value). Используется в [DocumentAPI.SaveDocumentSettings](#), [DocumentAPI.LoadDocumentSettings](#).

Описание полей таблицы DocumentAPI.DSVSettings представлено в таблице 31.

Таблица 31 – Описание полей таблицы DocumentAPI.DSVSettings

Поле	Описание
DocumentAPI.DSVSettings.autofit	Признак необходимости автоматического подстраивания ширины столбца под размер данных в ячейке

Поле	Описание
DocumentAPI.DSVSettings.startBlockIndex	Индекс блока документа сохранения
DocumentAPI.DSVSettings.lastBlockIndex	Индекс блока документа для окончания сохранения

Пример:

```
saveDocumentSettings = DocumentAPI.SaveDocumentSettings()
saveDocumentSettings.dsvSettings = DocumentAPI.DSVSettings()
saveDocumentSettings.dsvSettings.autofit = true
saveDocumentSettings.dsvSettings.startBlockIndex = 0
saveDocumentSettings.dsvSettings.lastBlockIndex = 10
```

6.41 Таблица DocumentAPI.Encoding

В таблице 32 приведены поддерживаемые кодировки документов. Используется в [DocumentAPI.LoadDocumentSettings](#).

Таблица 32 - Кодировки документов

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

6.42 Таблица DocumentAPI.FormulaType

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 33. Используется в [document:getFormulaType\(\)](#), [document:setFormulaType\(\)](#), [DocumentAPI.DocumentSettings](#).

Таблица 33 – Системы адресации ячеек в табличном документе

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

6.43 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 34.

Таблица 34 – Описание полей таблицы DocumentAPI.FractionCellFormatting

Поле	Описание
DocumentAPI.FractionCellFormatting.minNumeratorDigits	Количество позиций числителя
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())
```

6.44 Таблица DocumentAPI.FrozenRangePosition

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

6.44.1 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())
```

6.44.2 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())
```

6.44.3 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())
```

6.44.4 Метод FrozenRangePosition:createFrozenRows

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

Вызов:

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

Пример:

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

6.44.5 Метод FrozenRangePosition:createFrozenCols

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

Вызов:

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

Пример:

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

6.44.6 Метод FrozenRangePosition:isRowsCols

Возвращает true если диапазон содержит строки и колонки.

Пример:

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

6.44.7 Метод FrozenRangePosition:isArea

Возвращает true если диапазон является непрерывной областью.

Пример:

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


6.44.8 Метод `FrozenRangePosition:isRows`

Возвращает `true` если диапазон состоит из строк.

Пример:

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

6.44.9 Метод `FrozenRangePosition:isCols`

Возвращает `true` если диапазон состоит из колонок.

Пример:

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

6.45 Таблица `DocumentAPI.HeaderFooter`

Таблица `DocumentAPI.HeaderFooter` определяет колонтитул текстового документа.

6.45.1 Метод `HeaderFooter:getType`

Метод предоставляет информацию о типе колонтитула [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
```

6.45.2 Метод `HeaderFooter:getBlocks`

Метод предоставляет доступ к блокам [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
```

6.45.3 Метод `HeaderFooter:getRange`

Метод предоставляет диапазон [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
```

6.46 Таблица `DocumentAPI.HeaderFooterType`

Типы колонтитулов представлены в таблице 35.

Таблица 35 – Типы колонтитулов

Наименование константы	Описание
<code>DocumentAPI.HeaderFooterType_Header</code>	Верхний колонтитул
<code>DocumentAPI.HeaderFooterType_Footer</code>	Нижний колонтитул

6.47 Таблица `DocumentAPI.HeadersFooters`

Таблица `DocumentAPI.HeadersFooters` представляет коллекцию верхних и нижних колонтитулов раздела (см. Рисунок 54). Доступ к колонтитулам осуществляется посредством методов [Section.getHeaders\(\)](#), [Section.getFooters\(\)](#).

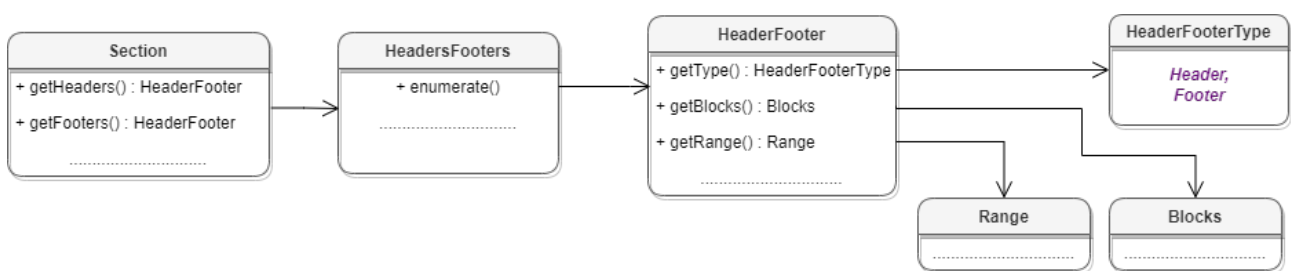


Рисунок 54 – Таблицы колонтитулов

6.47.1 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
```

6.48 Таблица DocumentAPI.HorizontalAnchorAlignment

В таблице 36 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали.

Таблица 36 – 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

6.49 Таблица DocumentAPI.HorizontalRelativeTo

В таблице 37 представлены типы размещения объекта относительно закрепленной позиции по горизонтали.

Таблица 37 – 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
DocumentAPI.HorizontalRelativeTo_PageRightMargin	Правое поле страницы

Наименование константы	Описание
DocumentAPI.HorizontalRelativeTo_PageInsideMargin	Внутреннее поле страницы
DocumentAPI.HorizontalRelativeTo_PageOutsideMargin	Внешнее поле страницы

6.50 Таблица DocumentAPI.HorizontalTextAnchoredPosition

Таблица `DocumentAPI.HorizontalTextAnchoredPosition` (см. Рисунок 55) предназначена для управления относительным положением объекта со смещением или выравниванием по горизонтали. Пример использования см. в [InlineFrame.setPosition\(\)](#).

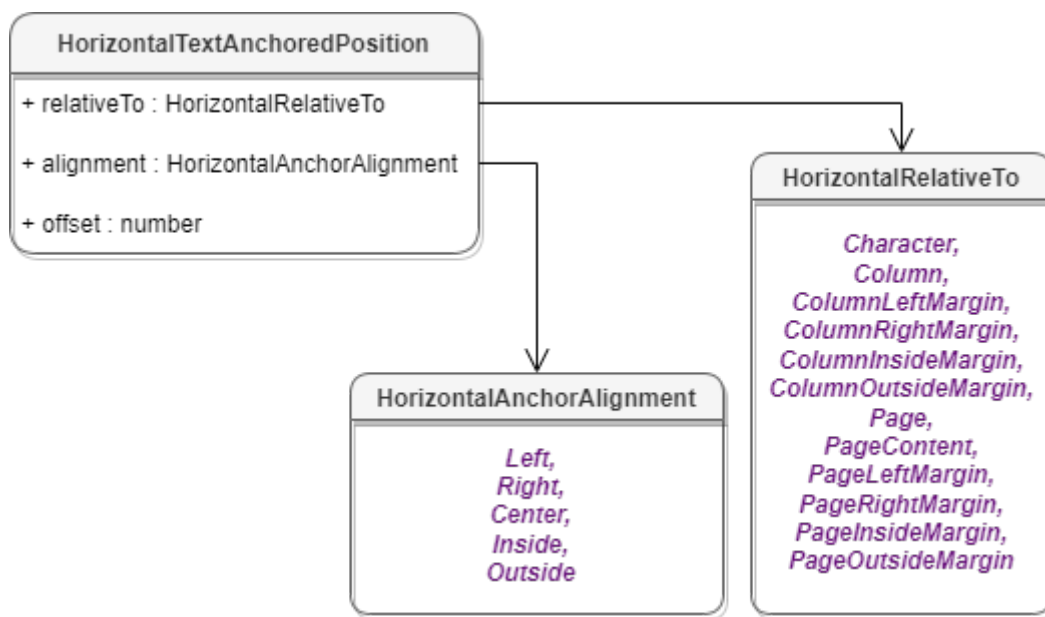


Рисунок 55 – Поля таблицы `DocumentAPI.HorizontalTextAnchoredPosition`

Описание полей таблицы `DocumentAPI.HorizontalTextAnchoredPosition` представлено в таблице 38.

Таблица 38 – Описание полей таблицы `DocumentAPI.HorizontalTextAnchoredPosition`

Поле	Описание
<code>DocumentAPI.HorizontalTextAnchoredPosition.alignment</code>	Тип выравнивания объекта относительно закрепленной позиции по горизонтали HorizontalAnchorAlignment
<code>DocumentAPI.HorizontalTextAnchoredPosition.relativeTo</code>	Тип размещения объекта относительно закрепленной позиции по горизонтали HorizontalRelativeTo

Поле	Описание
DocumentAPI.HorizontalTextAnchoredPosition.offset	Смещение объекта

6.51 Таблица DocumentAPI.Hyperlink

Таблица DocumentAPI.Hyperlink описывает свойства ссылки. Может быть получена посредством вызова метода [Cell.getHyperlink\(\)](#).

Таблица 39 – Описание полей таблицы DocumentAPI.Hyperlink

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

Пример:

```
local cell = document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0, 0))
local hyperlink = cell:getHyperlink()
print(hyperlink.url, hyperlink.tooltip, hyperlink.label)
```

6.51.1 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
```

6.52 Таблица DocumentAPI.Image

Таблица DocumentAPI.Image представляет собой изображение, находящееся в текстовом или табличном документе.

6.52.1 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
```

6.52.2 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
```

6.53 Таблица DocumentAPI.Images

Таблица `DocumentAPI.Images` используется для доступа к коллекции изображений. Может быть получена вызовом методов [Table.getImages\(\)](#), [Range.getImages\(\)](#).

6.53.1 Метод Images:enumerate

Метод позволяет перечислить коллекцию изображений.

Пример для текстового документа:

```
local images = document:getRange():getImages()
for image in images: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
```

6.54 Таблица DocumentAPI.InlineFrame

Таблица `DocumentAPI.InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 56). Предназначена для получения и изменения свойств позиции графических объектов (см. [Image:getFrame\(\)](#), [MediaObject:getFrame\(\)](#)). Используется в текстовом документе.

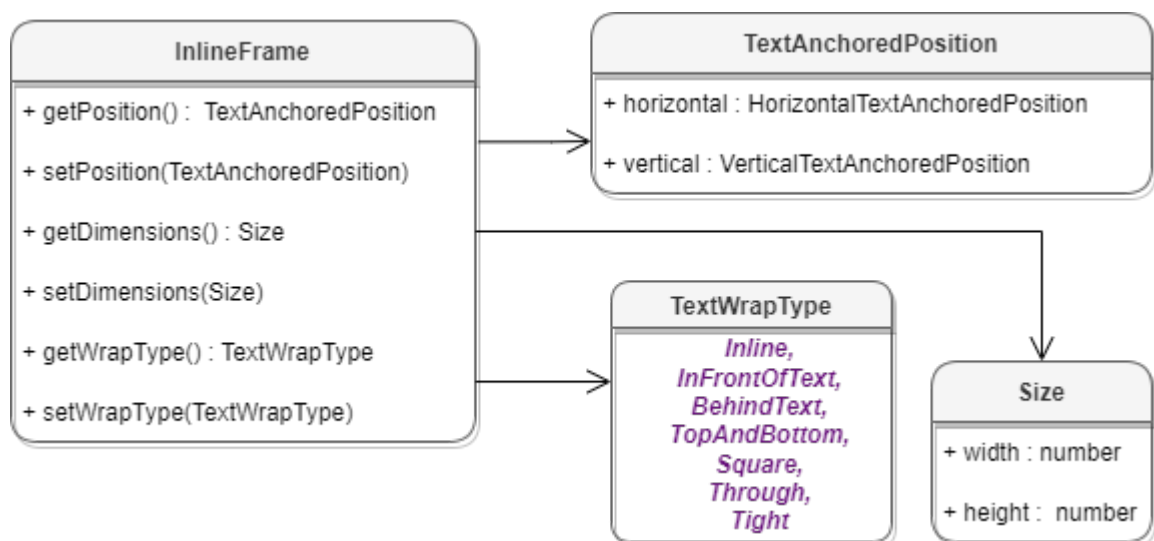


Рисунок 56 – Объектная модель таблицы `DocumentAPI.InlineFrame`

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
```

6.54.1 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 pos = DocumentAPI.TextAnchoredPosition()

-- Установка смещения по горизонтали относительно края колонки
pos.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos.horizontal.offset = x

-- Установка смещения по вертикали относительно края страницы
pos.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
pos.vertical.offset = y

-- Установка позиции рамки графического объекта
inlineFrame:setPosition(pos)
```

6.54.2 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
```

6.54.3 Метод `InlineFrame:setDimensions`

Метод задает размер [DocumentAPI.SizeU](#) встроенного объекта.

Пример:

```
inlineFrame:setDimensions(DocumentAPI.SizeU(100, 100))
```

6.54.4 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
```

6.54.5 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
```

6.54.6 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
```

6.55 Таблица DocumentAPI.Insets

Таблица `DocumentAPI.Insets` предназначена для задания полей, например, страницы. `DocumentAPI.Insets` представлено в таблице 40. Используется в поле `margins` таблицы [DocumentAPI.PageProperties](#).

Таблица 40 – 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)
```

6.56 Таблица DocumentAPI.ListSchema

Типы схем форматирования списков, которые могут быть применены к абзацам текста представлены в таблице 41. Данные константы используются в методах [Paragraph.getListSchema\(\)](#), [Paragraph.setListSchema\(\)](#).

Таблица 41 – Типы схем форматирования списков

Наименование константы	Описание
<code>DocumentAPI.ListSchema_Unknown</code>	Схема не определена
<code>DocumentAPI.ListSchema_UnknownBullet</code>	Список без маркера
<code>DocumentAPI.ListSchema_UnknownNumbering</code>	Нумерация без номера
<code>DocumentAPI.ListSchema_BulletCircleSolid</code>	Список с маркерами в виде заполненного круга
<code>DocumentAPI.ListSchema_BulletCircleContour</code>	Список с маркерами в виде окружности

Наименование константы	Описание
<code>DocumentAPI.ListSchema_BulletSquareSolid</code>	Список с маркерами в виде квадрата
<code>DocumentAPI.ListSchema_BulletDiamondDots</code>	Список с маркерами в виде четырех ромбов
<code>DocumentAPI.ListSchema_BulletHyphen</code>	Список с маркерами в виде дефиса
<code>DocumentAPI.ListSchema_BulletConcaveArrowSolid</code>	Список с маркерами в виде вогнутой стрелки
<code>DocumentAPI.ListSchema_BulletCheckmark</code>	Список с маркерами в виде галочки
<code>DocumentAPI.ListSchema_EnumeratorDecimalDot</code>	Десятичная нумерация с точкой
<code>DocumentAPI.ListSchema_EnumeratorDecimalDotMultiLevel</code>	Многоуровневая десятичная нумерация с точкой
<code>DocumentAPI.ListSchema_EnumeratorDecimalBracket</code>	Десятичная нумерация со скобкой
<code>DocumentAPI.ListSchema_EnumeratorLatinUpperCaseDot</code>	Нумерация латинскими прописными буквами с точкой
<code>DocumentAPI.ListSchema_EnumeratorLatinLowerCaseDot</code>	Нумерация латинскими строчными буквами с точкой
<code>DocumentAPI.ListSchema_EnumeratorLatinLowerCaseBracket</code>	Нумерация латинскими строчными буквами со скобкой
<code>DocumentAPI.ListSchema_EnumeratorRomanUpperCaseDot</code>	Нумерация римскими прописными цифрами с точкой
<code>DocumentAPI.ListSchema_EnumeratorRomanLowerCaseDot</code>	Нумерация римскими строчными цифрами с точкой
<code>DocumentAPI.ListSchema_EnumeratorDecimalRussianBracket</code>	Десятичная нумерация через запятую со скобкой
<code>DocumentAPI.ListSchema_EnumeratorRussianLowerCaseBracket</code>	Нумерация с русскими строчными буквами со скобкой

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

6.57 Таблица `DocumentAPI.LineEndingProperties`

Таблица `DocumentAPI.LineEndingProperties` содержит варианты оформления окончаний линий. Описание полей таблицы `DocumentAPI.LineEndingProperties` представлено в таблице 42. Используется в полях `headLineEndingProperties` и `tailLineEndingProperties` таблицы [DocumentAPI.LineProperties](#).

Таблица 42 – Описание полей таблицы DocumentAPI.LineEndingProperties

Поле	Тип	Описание
DocumentAPI.LineEndingProperties.style	LineEndingStyle	Стиль окончания линии
DocumentAPI.LineEndingProperties.relativeExtent	Size	Размер окончания линии относительно ее ширины

Пример:

```

local table = 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)

```

6.58 Таблица DocumentAPI.LineEndingStyle

В таблице 43 приведены типы окончания линии. Используется в поле style таблицы [DocumentAPI.LineEndingProperties](#).

Таблица 43 – 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 table = 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)
```

6.59 Таблица DocumentAPI.LineProperties

Таблица DocumentAPI.LineProperties предназначена для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 57).

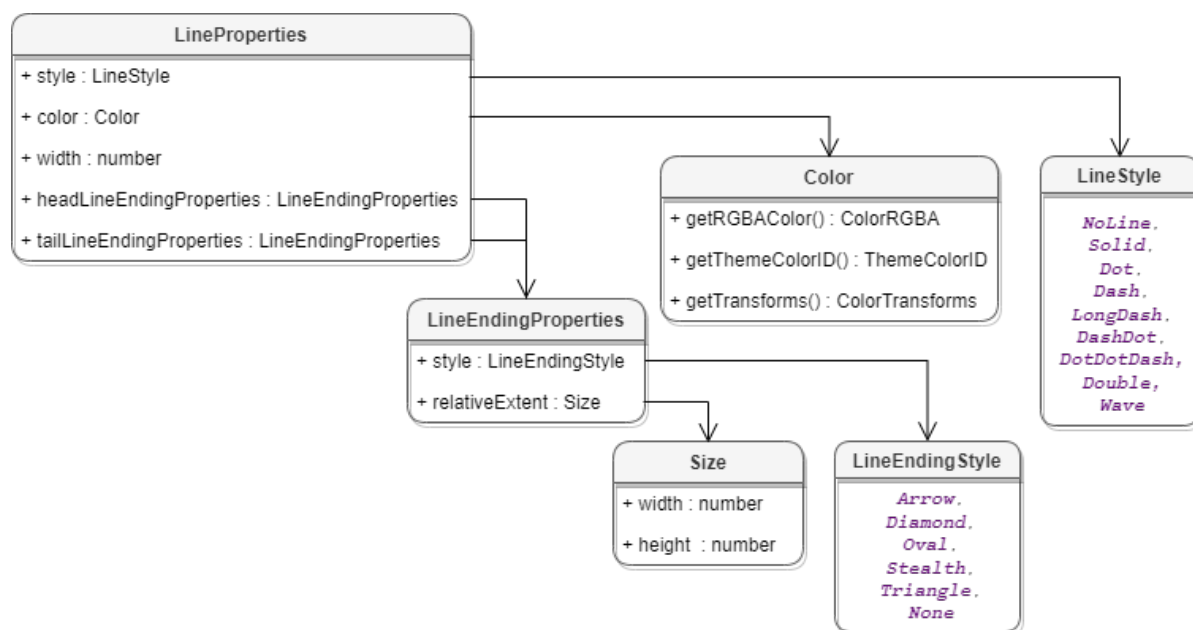


Рисунок 57 – Свойства границ ячеек

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)
    
```

6.59.1 Поле `LineProperties.style`

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

6.59.2 Поле `LineProperties.width`

Поле предназначено для установки ширины линии. Тип - числовой.

6.59.3 Поле `LineProperties.color`

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

6.59.4 Поле `LineProperties.headLineEndingProperties`

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

6.59.5 Поле `LineProperties.tailLineEndingProperties`

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

6.60 Таблица `DocumentAPI.LineSpacing`

Таблица `DocumentAPI.LineSpacing` задает межстрочный интервал абзаца. Поля таблицы приведены в таблице 44. Для управления значением межстрочного интервала используются значения, представленные в разделе [DocumentAPI.LineSpacingRule](#).

Таблица 44 – Параметры межстрочного интервала

Поле	Описание
<code>DocumentAPI.LineSpacing.value</code>	Значение межстрочного интервала, тип <code>number</code>
<code>DocumentAPI.LineSpacing.rule</code>	Правило формирования межстрочного интервала DocumentAPI.LineSpacingRule

Пример:

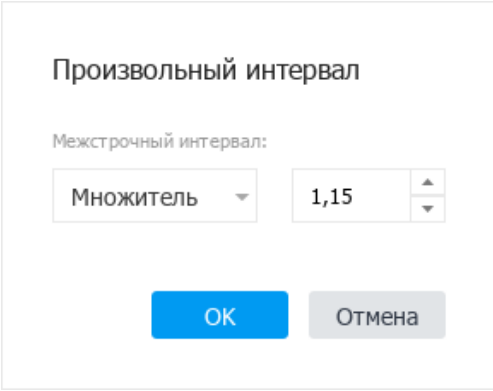
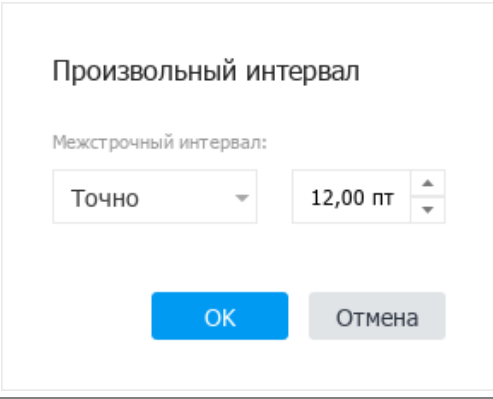
```
-- Конструктор
local lineSpacing = DocumentAPI.LineSpacing(1.5,
DocumentAPI.LineSpacingRule_Multiple)
-- Обращение к полям
lineSpacing.value = 1
lineSpacing.rule = DocumentAPI.LineSpacingRule_Exact
```

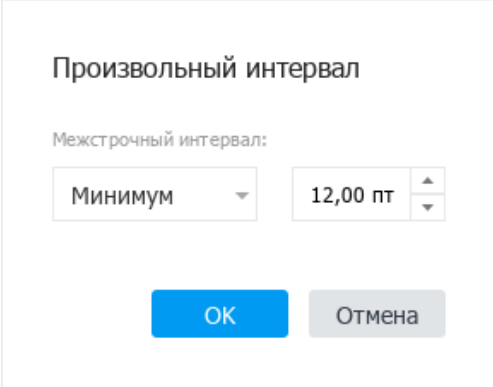
6.61 Таблица `DocumentAPI.LineSpacingRule`

В таблице 45 представлены варианты правил формирования межстрочного интервала текстового абзаца.

Таблица 45 – Виды межстрочного интервала

Наименование константы	Описание
<code>DocumentAPI.LineSpacingRule_Multiple</code>	Установка произвольного межстрочного интервала с использованием множителя. При вызове необходимо указать значение множителя, например: <pre>pPr.lineSpacing = DocumentAPI.LineSpacing(1.15, DocumentAPI.LineSpacingRule_Multiple)</pre>

Наименование константы	Описание
	<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> 
<p>DocumentAPI.LineSpacingRule_Exact</p>	<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> 
<p>DocumentAPI.LineSpacingRule_AtLeast</p>	<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>

Наименование константы	Описание
	<p>Действие команды аналогично ручной настройке межстрочного интервала в диалоге «Произвольный интервал» (см. документ «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).</p> 





Пример:





```
paragraph = document:getBlocks():getParagraph(0)
props = paragraph:getParagraphProperties()
props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
paragraph:setParagraphProperties(props)
```

6.62 Таблица DocumentAPI.LineStyle

В таблице 46 приведены типы линий. Используется в поле `style` таблицы [DocumentAPI.LineProperties](#).

Таблица 46 – Типы линий

Наименование константы	Описание
DocumentAPI.LineStyle_NoLine	Нет линии
DocumentAPI.LineStyle_Solid	
DocumentAPI.LineStyle_Dot	
DocumentAPI.LineStyle_Dash	
DocumentAPI.LineStyle_LongDash	

Наименование константы	Описание
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)

```

6.63 Таблица DocumentAPI.LoadDocumentSettings

Таблица `DocumentAPI.LoadDocumentSettings` предоставляет дополнительные настройки, необходимые для загрузки документов из файла (см. [application:loadDocument\(\)](#)).

Описание полей таблицы `DocumentAPI.LoadDocumentSettings` представлено в таблице 47.

Таблица 47 – Описание полей таблицы `DocumentAPI.LoadDocumentSettings`

Поле	Описание
DocumentAPI .LoadDocumentSettings .commonDocumentSettings	Экземпляр таблицы, общие настройки документа DocumentSettings
DocumentAPI .LoadDocumentSettings.encoding	Кодировка документа Encoding
DocumentAPI .LoadDocumentSettings.dsvSettings	Экземпляр класса DSVSettings , настройки, необходимые для работы с файлами CSV и DSV

Поле	Описание
DocumentAPI .LoadDocumentSettings .documentPassword	Пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows

6.64 Таблица DocumentAPI.LocaleInfo

Таблица DocumentAPI.LocaleInfo предоставляет информацию о локализации. Используется в поле localeInfo таблицы [DocumentAPI.DocumentSettings](#).

Описание полей таблицы DocumentAPI.LocaleInfo представлено в таблице 48.

Таблица 48 – Описание полей таблицы DocumentAPI.LocaleInfo

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

6.65 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 58).

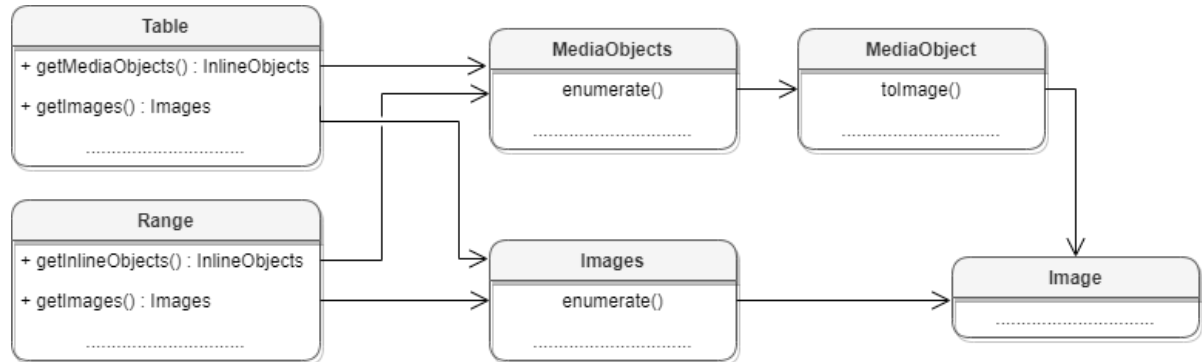


Рисунок 58 – Графические объекты

6.65.1 Метод MediaObjects:enumerate

Метод позволяет перечислить коллекцию встроенных объектов.

Пример для текстового документа:

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    print(mediaObject:getFrame():getWrapType())
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
```

6.66 Таблица DocumentAPI.MediaObject

Таблица `DocumentAPI.MediaObject` представляет собой встроенный объект документа.

6.66.1 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
```

6.66.2 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
```

6.67 Таблица DocumentAPI.NamedExpressions

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

Пример для текстового документа:

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

Пример для табличного документа:

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

6.67.1 Метод NamedExpressions:get

Возвращает именованный диапазон [NamedExpression](#) по имени name, если он существует.

Пример:

```
local namedExpression = namedExpressions:get("Продажи")
if (namedExpression) then
    print(namedExpression:getName()) -- Продажи
else
    print("No named expression was found")
end
```

6.67.2 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
```

6.67.3 Метод `NamedExpression:addExpression`

Добавляет новый именованный 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
```

6.67.4 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
```

6.68 Таблица `DocumentAPI.NamedExpression`

Класс описывает структуру именованного диапазона.

Пример:

```
local namedExpressions = sheet:getNamedExpressions()  
for namedExpression in namedExpressions:enumerate() do  
    print(namedExpression:getName())  
    print(namedExpression:getExpression())  
    cellRange = namedExpression:getCellRange()  
    print(cellRange:getBeginRow(), cellRange:getLastRow())  
end
```

6.68.1 Метод `NamedExpression.getName`

Возвращает имя именованного диапазона. См. пример в главе [DocumentAPI.NamedExpression](#).

6.68.2 Метод `NamedExpression.getExpression`

Возвращает текст диапазона (формулы). См. пример в главе [DocumentAPI.NamedExpression](#).

6.68.3 Метод `NamedExpression.getCellRange`

Возвращает диапазон ячеек [CellRange](#), если диапазон является ссылкой на именованный диапазон. См. пример в главе [DocumentAPI.NamedExpression](#).

6.69 Таблица `DocumentAPI.NamedExpressionsValidationResult`

Таблица `DocumentAPI.NamedExpressionsValidationResult` описывает результат операций [NamedExpressions.addExpression\(\)](#), [NamedExpressions.removeExpression\(\)](#). Описание полей таблицы представлено в таблице 49.

Таблица 49 – Описание полей таблицы `DocumentAPI.NamedExpressionsValidationResult`

Поле	Описание
<code>DocumentAPI.NamedExpressionsValidationResult_Success</code>	Операция выполнена успешно
<code>DocumentAPI.NamedExpressionsValidationResult_WrongName</code>	Неправильный формат имени

Поле	Описание
DocumentAPI.NamedExpressionsValidationResult_isUsedInFormula	Имя уже используется в формуле

6.70 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 50.

Таблица 50 – Описание полей таблицы DocumentAPI.NumberCellFormatting

Поле	Описание
DocumentAPI.NumberCellFormatting.decimalPlaces	Количество десятичных позиций
DocumentAPI.NumberCellFormatting.useThousandsSeparator	Использовать разделитель для тысячных
DocumentAPI.NumberCellFormatting.useRedForNegative	Использовать красный цвет для отрицательных значений
DocumentAPI.NumberCellFormatting.useBracketsForNegative	Использовать скобки для отрицательных значений
DocumentAPI.NumberCellFormatting.hideSign	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())
```

6.71 Таблица DocumentAPI.PageOrientation

Типы ориентации страницы представлены в таблице 51. Данная константа может быть использована для получения / установки ориентации страниц для секции или документа.

Таблица 51 – Типы ориентации страницы

Наименование константы	Описание
DocumentAPI.PageOrientation_Landscape	Альбомная ориентация страницы
DocumentAPI.PageOrientation_Portrait	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())
```

6.72 Таблица DocumentAPI.PageProperties

Таблица DocumentAPI.PageProperties предоставляет такие свойства страницы как высота, ширина, размеры полей. Описание полей приведено в таблице 52. Используется в [Document.setPageProperties\(\)](#), [Section.getPageProperties\(\)](#), [Section.setPageProperties\(\)](#).

Таблица 52 – Описание полей таблицы DocumentAPI.PageProperties

Поле	Описание
DocumentAPI.PageProperties.height	Высота страницы
DocumentAPI.PageProperties.width	Ширина страницы
DocumentAPI.PageProperties.margins	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)
```

6.73 Таблица DocumentAPI.Paragraph

Таблица `DocumentAPI.Paragraph` (см. Рисунок 59) предоставляет доступ к свойствам абзаца.

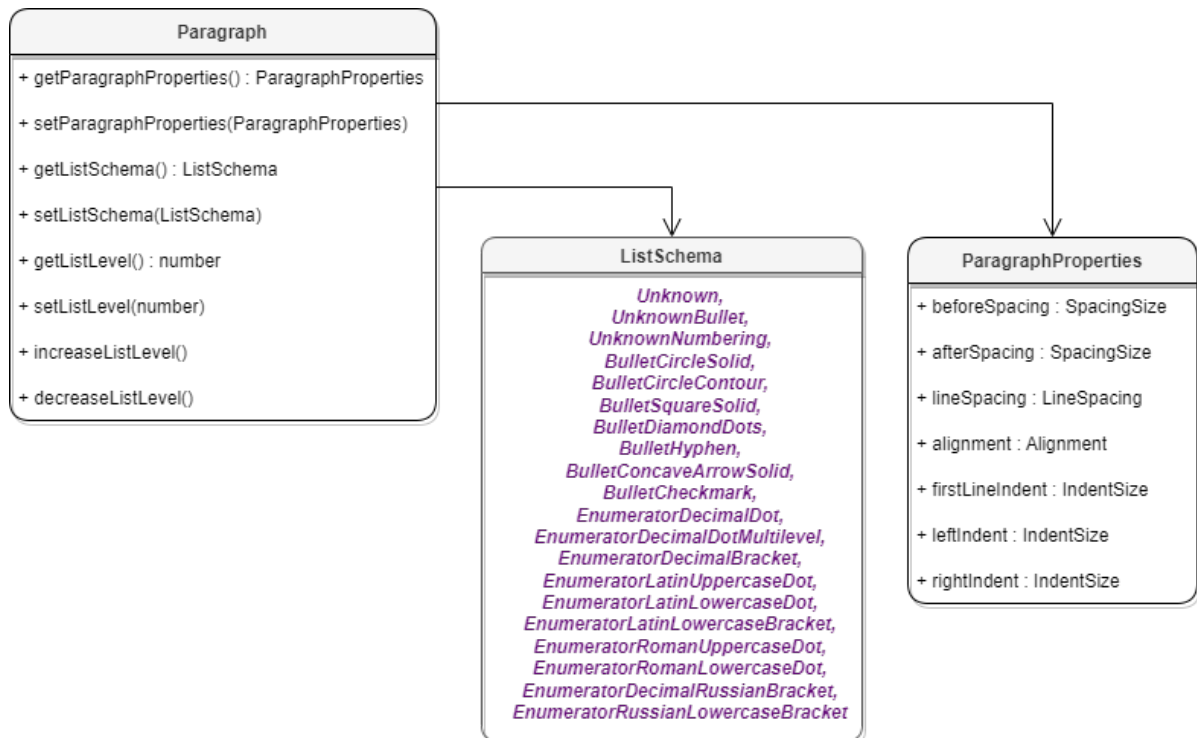


Рисунок 59 – Объектная модель таблиц для работы со свойствами параграфа

6.73.1 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
```

6.73.2 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
```

6.73.3 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()
```

6.73.4 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)
```

6.73.5 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()
```

6.73.6 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)
```

6.73.7 Метод Paragraph:increaseListLevel

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

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:increaseListLevel()
```

6.73.8 Метод Paragraph:decreaseListLevel

Метод позволяет уменьшить на единицу глубину вложенности элемента списка. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе.

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:decreaseListLevel()
```

6.74 Таблица DocumentAPI.Paragraphs

Таблица `DocumentAPI.Paragraphs` предоставляет доступ к коллекции абзацев типа [DocumentAPI.Paragraph](#) (см. Рисунок 60). Коллекция абзацев может быть получена из таблицы [DocumentAPI.Range](#) посредством использования вызова [Range:getParagraphs\(\)](#).

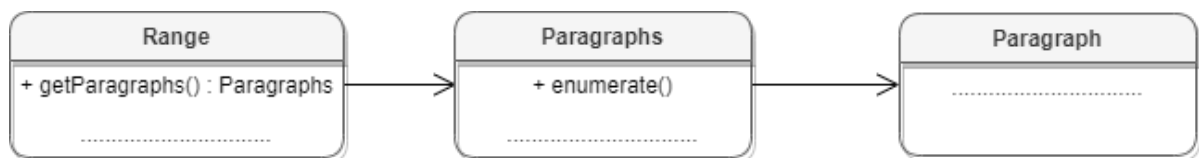


Рисунок 60 – Объектная модель для работы со списком абзацев

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()
```

6.74.1 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)
```

6.74.2 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)
```

6.74.3 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()
```

6.74.4 Метод Paragraphs:decreaseListLevel

Метод уменьшает уровень списка на единицу. В случае, если минимальный уровень уже установлен, уменьшения не происходит. Данный метод используется только в текстовом документе.

Пример:

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:decreaseListLevel()
```

6.74.5 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
  
```

6.75 Таблица DocumentAPI.ParagraphProperties

Таблица `DocumentAPI.ParagraphProperties` предназначена для управления свойствами форматирования (см. Рисунок 61). Таблица `DocumentAPI.ParagraphProperties` используется в методах [Paragraph:getParagraphProperties](#) и [Paragraph:setParagraphProperties](#).

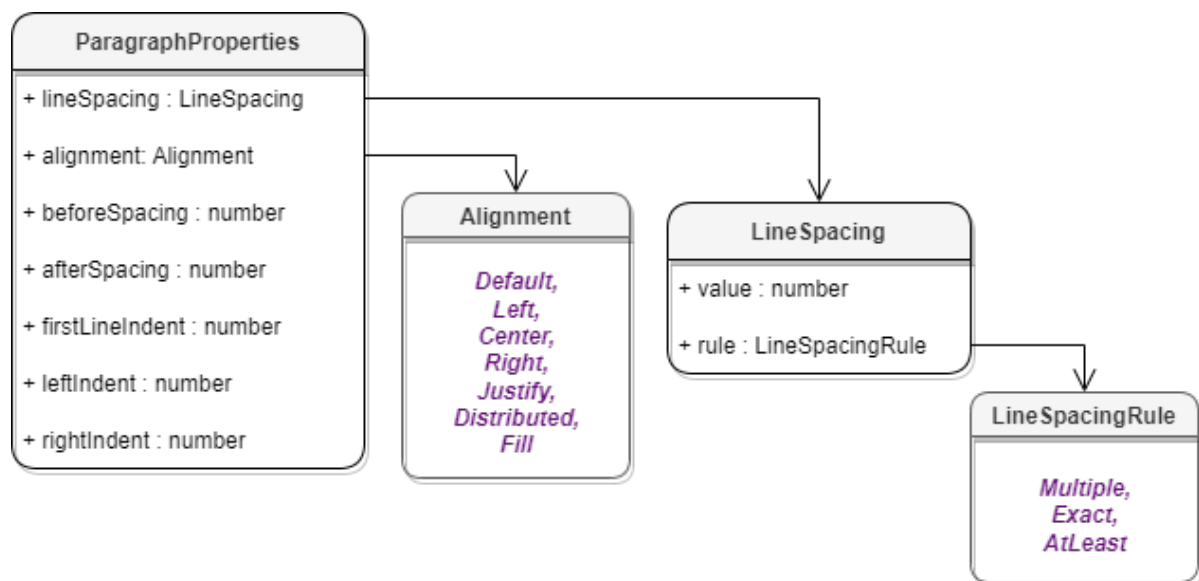
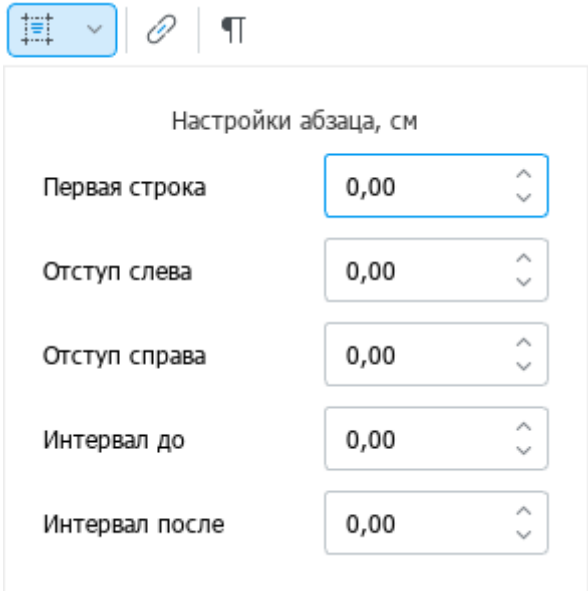


Рисунок 61 – Объектная модель таблиц для работы со свойствами параграфа

Описание полей таблицы [DocumentAPI.ParagraphProperties](#) представлено в таблице 53.

Таблица 53 – Описание полей таблицы `DocumentAPI.ParagraphProperties`

Поле	Описание
<code>ParagraphProperties.beforeSpacing</code>	Установка величины расстояния до абзаца.

Поле	Описание
	<p>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»).</p>
ParagraphProperties.afterSpacing	<p>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»).</p>  <p>The screenshot shows a dialog box titled 'Настройки абзаца, см' (Paragraph Properties, cm). It contains five input fields, each with a value of '0,00' and up/down arrows: 'Первая строка' (First line), 'Отступ слева' (Left indent), 'Отступ справа' (Right indent), 'Интервал до' (Interval before), and 'Интервал после' (Interval after). The 'Интервал после' field is highlighted with a blue border.</p>
ParagraphProperties.lineSpacing	<p>Расстояние между строк одного абзаца (межстрочный интервал), LineSpacing.</p>
ParagraphProperties.alignment	<p>Выравнивание текстового фрагмента по горизонтали. Список допустимых значений находится в разделе Alignment.</p>
ParagraphProperties.firstLineIndent	<p>Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне Настройки абзаца, (см. рисунок выше), в поле Первая строка (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).</p>
ParagraphProperties.leftIndent	<p>Расстояние от левого поля документа до абзаца (отступ слева).</p>

Поле	Описание
	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
```

6.76 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 54.

Таблица 54 – 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())
```

6.77 Таблица DocumentAPI.PointU

Таблица DocumentAPI.PointU представляет позицию объекта в двухмерном пространстве. Описание полей таблицы DocumentAPI.PointU представлено в таблице 55.

Таблица 55 – Описание полей таблицы DocumentAPI.PointU

Поле	Описание
DocumentAPI.PointU.x	Позиция x
DocumentAPI.PointU.y	Позиция y

Пример:

```
local point = DocumentAPI.PointU(2, 3)
print("x=", point.x, ", y=", point.y) --(x = 2.0, y = 3.0)
```

6.77.1 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)
```

6.78 Таблица DocumentAPI.Position

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

6.78.1 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")
```

6.78.2 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 текстовый документ:

```
local rng = document:getRange()
local begin_pos = rng:getBegin()
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")
```

6.78.3 Метод Position:insertPageBreak

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



Внимание ! Данный метод может быть использован только в текстовом документе.

Пример:

```
local rng = document:getRange()  
local end_pos = rng:getEnd()  
end_pos:insertPageBreak()
```

6.78.4 Метод Position:insertLineBreak

Метод предназначен для вставки перевода строки.



Внимание ! Данный метод может быть использован только в текстовом документе.

Пример:

```
local rng = document:getRange()  
local end_pos = rng:getEnd()  
end_pos:insertLineBreak()
```

6.78.5 Метод Position:insertSectionBreak

Вставляет разрыв раздела в текущую позицию.



Внимание ! Данный метод может быть использован только в текстовом документе.

Пример:

```
document:getRange():getEnd():insertSectionBreak()
```

6.78.6 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")
```

6.78.7 Метод Position:insertImage

Вставляет изображение из файла в текущую позицию.



Внимание ! Данный метод может быть использован только в текстовом документе.

Вызов:

```
insertImage( url, size )
```

Параметры:

- `url` – полный путь к файлу;
- `size` – геометрические размеры изображения для вставки.

Пример:

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

6.78.8 Метод Position:removeBackward

Метод удаляет `count` объектов (символов, картинок и т.д.) до текущей позиции.

Пример:

```
document:getRange():getEnd():removeBackward(3)
```

6.78.9 Метод `Position:removeForward`

Метод удаляет `count` объектов (символов, картинок и т.д.) после текущей позиции.

Пример:

```
document:getRange():getBegin():removeForward(3)
```

6.78.10 Метод `Position:insertBookmark`

Вставляет закладку с наименованием в текущую позицию.



Внимание ! Данный метод может быть использован только в текстовом документе.

Пример:

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

6.79 Таблица `DocumentAPI.PrintDocumentResult`

В таблице 56 представлены коды, возвращаемые после печати (см. [EditorAPI.showPrintDialog\(\)](#)).

Таблица 56 – 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

6.80 Таблица `DocumentAPI.PrintSettings`

Таблица `DocumentAPI.PrintSettings` представляет установки, используемые при печати документов. Описание полей таблицы `DocumentAPI.PrintSettings` представлено в таблице 57. Используется в [EditorAPI.printDocument](#).

Таблица 57 – 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
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

Поле	Тип	Описание
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

6.81 Таблица DocumentAPI.PivotTablesManager

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

Пример:

```
local pivotTablesManager = document:getPivotTablesManager()
```

6.81.1 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"))
```

6.82 Таблица DocumentAPI.PivotTable

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

6.82.1 Метод `PivotTable:remove`

Метод удаляет сводную таблицу.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
pivotTable:remove()
```

6.82.2 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()
print(pivotTable:getSourceRangeAddress()) -- 'Sheet1'!I3:K7
```

6.82.3 Метод `PivotTable:getSourceRange`

Метод возвращает диапазон [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
```

6.82.4 Метод `PivotTable:getPivotRange`

Метод возвращает диапазон ячеек [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
```

6.82.5 Метод `PivotTable:changeSourceRange`

Метод позволяет задать новый диапазон исходных данных сводной таблицы без обновления самой таблицы. Параметр `sourceRange` – строка, представляющая новый диапазон таблицы.

Пример:

```
pivotTable:changeSourceRange("I3:K5")
local cellRange = pivotTable:getSourceRange()
print(cellRange:getBeginRow(), cellRange:getLastRow())
```

6.82.6 Метод `PivotTable:isRowGrandTotalEnabled`

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

6.82.7 Метод `PivotTable:isColumnGrandTotalEnabled`

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

6.82.8 Метод `PivotTable:getPivotTableCaptions`

Метод возвращает информацию [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)
```

6.82.9 Метод `PivotTable:getPivotTableLayoutSettings`

Метод возвращает настройки отображения [PivotTableLayoutSettings](#) сводной таблицы.

Пример:

```
local settings = pivotTable:getPivotTableLayoutSettings()
print(settings.reportLayout)
print(settings.valueFieldsOrientation)
print(settings.pageFieldOrder)
print(settings.indentForCompactLayout)
print(settings.pageFieldWrapCount)
```

6.82.10 Метод `PivotTable:getUnsupportedFeatures`

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

Пример:

```
local unsupportedFeatures = pivotTable:getUnsupportedFeatures()
for featureIndex = 0, unsupportedFeatures:size() - 1 do
    print(unsupportedFeatures[featureIndex])
end
```

6.82.11 Метод `PivotTable:getFieldsList`

Метод возвращает список [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
```

6.82.12 Метод `PivotTable:getRowFields`

Метод возвращает список полей [PivotTableCategoryField](#) из области строк.

Пример:

```
local rowFields = pivotTable:getRowFields()
for fieldIdx = 0, rowFields:size() - 1 do
    print(rowFields[fieldIdx].fieldProperties.fieldName)
end
```

6.82.13 Метод `PivotTable:getColumnFields`

Метод возвращает список полей [PivotTableCategoryField](#) из области колонок.

Пример:

```
local columnFields = pivotTable:getColumnFields()
for fieldIdx = 0, columnFields:size() - 1 do
    print(columnFields[fieldIdx].fieldProperties.fieldName)
end
```

6.82.14 Метод `PivotTable:getValueFields`

Метод возвращает список полей [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
```

6.82.15 Метод `PivotTable:getPageFields`

Метод возвращает список полей [PivotTablePageField](#) из области фильтров.

Пример:

```
local pageFields = pivotTable:getPageFields()
print(pageFields:size())
```

6.82.16 Метод `PivotTable:getFieldCategories`

Метод возвращает список категорий [PivotTableFieldCategories](#), содержащих заданное поле `fieldName`.

Пример:

```
local fieldCategories = pivotTable:getFieldCategories("Age")
```

6.82.17 Метод `PivotTable:getFieldItems`

Метод возвращает все элементы [PivotTableItems](#) сводной таблицы по заданному имени поля `fieldName`.

Пример:

```
local pivotTableItems = pivotTable:getFieldItems("Age")
print(pivotTableItems)
```

6.82.18 Метод `PivotTable:getFieldItemsByName`

Метод возвращает все элементы [PivotTableItems](#) из заданного поля `fieldName` по имени `itemName`.

Пример:

```
local pivotTableItemsByName = pivotTable:getFieldItemsByName("Ultimate Question  
of Life", "42")  
print(pivotTableItemsByName)
```

6.82.19 Метод `PivotTable:getFilter`

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

Пример:

```
local filter = pivotTable:getFilter("Age")  
print(filter:getFieldName())
```

6.82.20 Метод `PivotTable:getFilters`

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

Пример:

```
local filters = pivotTable:getFilters()  
for filter in filters:enumerate() do  
    -- use filter  
end
```

6.82.21 Метод `PivotTable:update`

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

Пример:

```
local updateResult = pivotTable:update()  
if (updateResult ~= DocumentAPI.PivotTableUpdateResult_Success) then  
    print(updateResult)  
end
```

6.82.22 Метод `PivotTable:createPivotTableEditor`

Метод возвращает объект [PivotTableEditor](#), который служит для обновления свойств и редактирования сводной таблицы.

Пример:

```
local cell = tbl:getCell("L8")  
local pivotTable = cell:getPivotTable()  
local pivotTableEditor = pivotTable:createPivotTableEditor()
```

6.83 Таблица DocumentAPI.PivotTableCaptions

Таблица `DocumentAPI.PivotTableCaptions` хранит все пользовательские заголовки сводной таблицы. Описание полей таблицы представлено в таблице 58.

Таблица 58 – 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).

6.84 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 59.

Таблица 59 – 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 .

Поле	Описание
<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.

6.85 Таблица `DocumentAPI.PivotTableReportLayout`

Таблица `DocumentAPI.PivotTableReportLayout` описывает внешний вид отчетов сводной таблицы. Является полем таблицы [DocumentAPI.PivotTableLayoutSettings](#). Описание полей таблицы представлено в таблице 60.

Таблица 60 – 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

6.86 Таблица `DocumentAPI.PageFieldOrder`

Таблица `DocumentAPI.PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем таблицы [DocumentAPI.PivotTableLayoutSettings](#). Описание полей таблицы представлено в таблице 61.

Таблица 61 – Описание полей таблицы `DocumentAPI.PageFieldOrder`

Поле	Описание
<code>DocumentAPI.PageFieldOrder_DownThenOver</code>	Вниз, затем поперек

Поле	Описание
DocumentAPI.PageFieldOrder_OverThenDown	Поперек, затем вниз

6.87 Таблица DocumentAPI.PivotTableUnsupportedFeature

Таблица `DocumentAPI.PivotTableUnsupportedFeature` описывает неподдерживаемую функциональность сводных таблиц. Получение неподдерживаемой функциональности сводных таблиц описано в [PivotTable:getUnsupportedFeatures\(\)](#). Описание полей таблицы представлено в таблице 62.

Таблица 62 – Описание полей таблицы `DocumentAPI.PivotTableUnsupportedFeature`

Поле	Описание
DocumentAPI.PivotTableUnsupportedFeature_CalculatedField	Вычисляемые поля
DocumentAPI.PivotTableUnsupportedFeature_CalculatedItem	Вычисляемые элементы
DocumentAPI.PivotTableUnsupportedFeature_CollapsedValues	Свернутые поля
DocumentAPI.PivotTableUnsupportedFeature_ShowDataAs	Вычисления (Show data как в MS Excel)

6.88 Таблица DocumentAPI.PivotTableFieldCategories

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

6.88.1 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
```

6.89 Таблица DocumentAPI.PivotTableFunction

Таблица `DocumentAPI.PivotTableFunction` описывает функции, которые могут быть использованы в сводных таблицах. Описание полей таблицы представлено в таблице 63.

Таблица используется в качестве поля `subtotalFunctions` таблицы [DocumentAPI.PivotTableCategoryField](#).

Таблица 63 – 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

6.90 Таблица DocumentAPI.PivotTableFilters

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

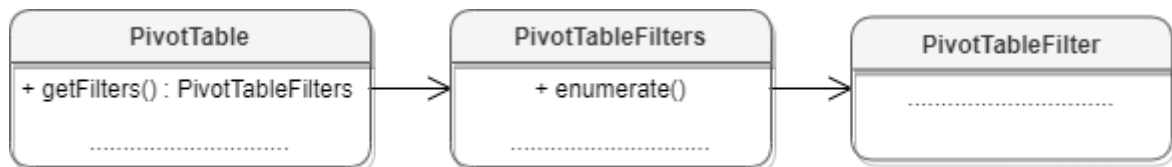


Рисунок 62 – Объектная модель таблиц для работы с фильтрами

6.90.1 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
  
```

6.91 Таблица DocumentAPI.PivotTableFilter

Позволяет осуществить доступ к списку фильтров таблицы, каждый из которых обладает свойством видимости. При любом изменении фильтров они должны быть применены к сводной таблице посредством использования методов [PivotTableEditor.setFilter\(\)](#), [PivotTableEditor.setFilters\(\)](#).

Пример:

```
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()
```

6.91.1 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
```

6.91.2 Метод PivotTableFilter:getCount

Возвращает количество фильтруемых полей.

Пример:

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
    print(filter:getCount())
end
```

6.91.3 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
```

6.91.4 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
```

6.91.5 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
```

6.92 Таблица DocumentAPI.PivotTableField

Таблица `DocumentAPI.PivotTableField` содержит свойства полей сводной таблицы (см. таблицу 64). Таблица может быть получена посредством вызова [PivotTable:getFieldsList\(\)](#).

Таблица 64 – Описание полей таблицы `DocumentAPI.PivotTableField`

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

6.93 Таблица `DocumentAPI.PivotTableFieldProperties`

`DocumentAPI.PivotTableFieldProperties` содержит свойства поля [DocumentAPI.PivotTableField](#) сводной таблицы (см. таблицу 65).

Таблица 65 – 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

6.94 Таблица `DocumentAPI.PivotTableCategoryField`

`DocumentAPI.PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. таблицу 66). Таблица может быть получена посредством вызовов [PivotTable.getRowFields\(\)](#), [PivotTable.getColumnFields\(\)](#).

Таблица 66 – Описание полей таблицы `DocumentAPI.PivotTableCategoryField`

Поле	Описание
<code>PivotTableCategoryField.fieldProperties</code>	Свойства поля PivotTableFieldProperties
<code>PivotTableCategoryField.subtotalFunctions</code>	Список функций PivotTableFunction для вычисления подытога

6.95 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

6.96 Таблица `DocumentAPI.PivotTablePageField`

Содержит свойства поля из области фильтров (см. таблицу 68). Таблица может быть получена посредством вызова [PivotTable:getPageFields\(\)](#).

Таблица 68 – Описание полей таблицы `DocumentAPI.PivotTablePageField`

Поле	Описание
<code>PivotTablePageField.fieldProperties</code>	Свойства поля PivotTableFieldProperties

6.97 Таблица `DocumentAPI.PivotTableItems`

Таблица обеспечивает доступ к списку элементов сводной таблицы (см. Рисунок 63).

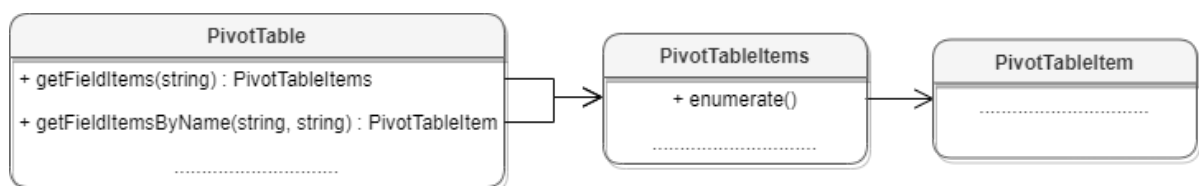


Рисунок 63 – Объектная модель таблиц для работы с элементами сводных таблиц

6.97.1 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
```

6.98 Таблица DocumentAPI.PivotTableItem

DocumentAPI.PivotTableItem описывает элемент сводной таблицы (см. Рисунок 64). См. пример в главе [PivotTableItems:enumerate](#).

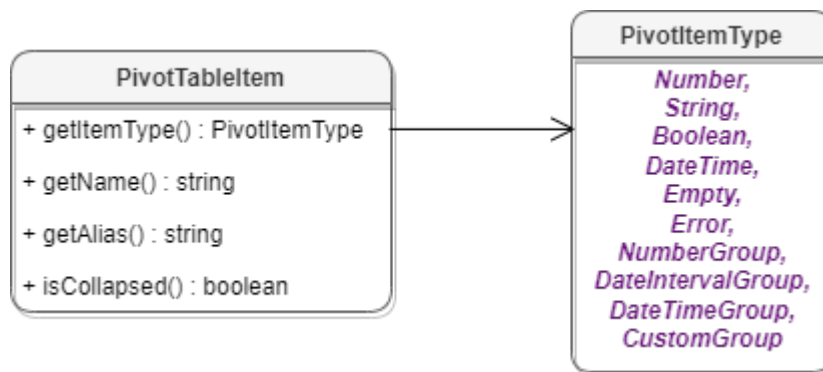


Рисунок 64 – Таблица DocumentAPI.PivotTableItem

6.98.1 Метод PivotTableItem:getName

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

6.98.2 Метод PivotTableItem:getAlias

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

6.98.3 Метод PivotTableItem:getItemType

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

6.98.4 Метод PivotTableItem:isCollapsed

Метод возвращает true, если элемент сводной таблицы свернут. См. пример в главе [PivotTableItems:enumerate](#).

6.99 Таблица `DocumentAPI.PivotTableItemType`

Таблица `DocumentAPI.PivotTableItemType` содержит возможные типы элементов сводной таблицы. Описание полей таблицы представлено в таблице 69.

Таблица 69 – Описание полей таблицы `DocumentAPI.PivotTableItemType`

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

6.100 Таблица `DocumentAPI.PivotTableEditor`

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

6.100.1 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()
```

6.100.2 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()
```

6.100.3 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()
```

6.100.4 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()
```

6.100.5 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()
```

6.100.6 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()
```

6.100.7 Метод `PivotTableEditor:setSummarizeFunction`

Метод задает суммирующую функцию для поля из области значений. Параметр `valueFieldName` - имя поля (тип - строка), `summarizeFunction` - суммирующая функция, тип - [DocumentAPI.PivotTableFunction](#). Метод возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример:

```
pivotTableEditor = pivotTableEditor:addField("Age",  
DocumentAPI.PivotTableFieldCategory_Values)  
pivotTableEditor:apply()
```

6.100.8 Метод `PivotTableEditor:setFilter`

Метод задает фильтр [DocumentAPI.PivotTableFilter](#) сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`. 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)  
        pivotTableEditor:setFilter(filter)  
    end  
end  
pivotTableEditor:apply()
```

6.100.9 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:setFilter(filters)
pivotTableEditor:apply()
```

6.100.10 Метод `PivotTableEditor:setCaptions`

Метод задает заголовки сводной таблицы [DocumentAPI.PivotTableCaptions](#), возвращает объект [DocumentAPI.PivotTableEditor](#).

Пример:

```
local pivotTableCaptions = pivotTable:getPivotTableCaptions()
pivotTableCaptions.grandTotalCaption = "Общий итог за год"

local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor = pivotTableEditor:setCaptions(pivotTableCaptions)
pivotTableEditor:apply()
```

6.100.11 Метод `PivotTableEditor:setLayoutSettings`

Метод [DocumentAPI.PivotTableLayoutSettings](#) устанавливает настройки отображения 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()
```

6.100.12 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)
```

6.100.13 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
```

6.101 Таблица `DocumentAPI.PivotTableUpdateResult`

В таблице 70 приведены константы, которые соответствуют возможным результатам обновления сводной таблицы (см. методы [PivotTable.update\(\)](#), [PivotTableEditor:apply\(\)](#)).

Таблица 70 – Результаты обновления сводной таблицы

Наименование константы	Описание
<code>DocumentAPI.PivotTableUpdateResult_Success</code>	Успешное обновление таблицы
<code>DocumentAPI.PivotTableUpdateResult_NoPivotTable</code>	Сводная таблица не найдена
<code>DocumentAPI.PivotTableUpdateResult_NoSuchFieldInCategory</code>	Не найдено поле в категории
<code>DocumentAPI.PivotTableUpdateResult_NoSuchFieldInPivotTable</code>	Не найдено поле в сводной таблице
<code>DocumentAPI.PivotTableUpdateResult_InvalidIndex</code>	Ошибка в индексе
<code>DocumentAPI.PivotTableUpdateResult_FieldAlreadyEnabled</code>	Поле уже существует
<code>DocumentAPI.PivotTableUpdateResult_MovingFieldToTheSameCategoryForbi</code>	Попытка перемещения поля в рамках текущей категории

Наименование константы	Описание
dden	
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
DocumentAPI.PivotTableUpdateResult_AnotherPivotInsideDataSource	Найдена другая сводная таблица в этом же диапазоне
DocumentAPI.PivotTableUpdateResult_Canceled	Обновление сводной таблицы отменено

6.102 Таблица DocumentAPI.PivotTableFieldCategory

Таблица DocumentAPI.PivotTableFieldCategory описывает флаги, которые задают категорию области полей. Описание полей таблицы представлено в таблице 71.

Таблица 71 – Описание полей таблицы DocumentAPI.PivotTableFieldCategory

Поле	Описание
DocumentAPI.PivotTableFieldCategory_Pages	Область фильтров
DocumentAPI.PivotTableFieldCategory_Rows	Область строк
DocumentAPI.PivotTableFieldCategory_Columns	Область колонок
DocumentAPI.PivotTableFieldCategory_Values	Область значений

6.103 Таблица DocumentAPI.Range

Таблица DocumentAPI.Range предоставляет доступ к диапазону документа. На рисунке 65 изображена объектная модель таблиц, относящихся к работе с диапазонами.

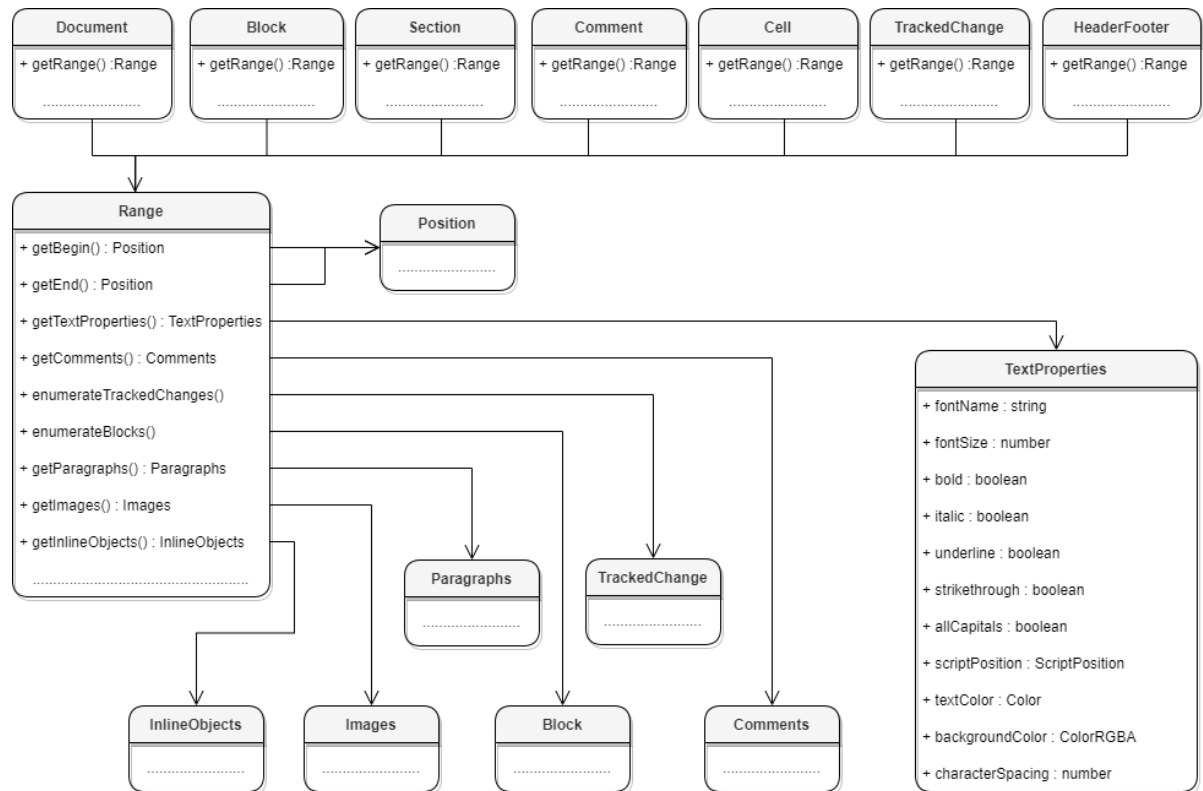


Рисунок 65 – Объектная модель для работы с таблицей DocumentAPI .Range

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
```

6.103.1 Метод Range:getBegin

Метод возвращает позицию [DocumentAPI.Position](#) в начале диапазона.

Пример для текстового документа:

```
local range = document:getRange() -- содержимое всего документа
local pos = range:getBegin() -- в начало документа
pos:insertText("Hello")
```

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("Hello")
```

6.103.2 Метод Range:getEnd

Метод возвращает позицию в конце диапазона, не включая последний символ paragraph mark.

Пример для текстового документа:

```
local range = document:getRange() -- содержимое всего документа
local pos = range:getEnd() -- в конец документа
pos:insertText("Hello")
```

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:getEnd() -- в конец ячейки
pos:insertText("Hello")
```

6.103.3 Метод Range:extractText

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и другие объекты игнорируются.

Пример для текстового документа:

```
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())
```

6.103.4 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())
```

6.103.5 Метод Range:lockContent

Метод запрещает изменения содержимого диапазона.



Внимание ! Данный метод может быть использован только в текстовом документе.

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()
```

6.103.6 Метод Range:unlockContent

Метод разрешает изменения содержимого диапазона.



Внимание ! Данный метод может быть использован только в текстовом документе.

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()
```

6.103.7 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
```

6.103.8 Метод Range:replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример для текстового документа:

```
local range = document:getRange() -- содержимое всего документа
range:replaceText("New text")
```

Пример для табличного документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки таблицы
range:replaceText("New text")
```

6.103.9 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())
```

6.103.10 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)
```

6.103.11 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)
```

6.103.12 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
```

6.103.13 Метод Range:enumerateTrackedChanges

Предоставляет возможность итерации по отслеживаемым изменениям [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
```

6.103.14 Метод Range:getComments

Обеспечивает доступ к комментариям в диапазоне.

Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам. Если дат нет, то порядок комментариев не определен.

Пример:

```
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())  
end
```

```
print(comment:getReplies())
end
```

6.103.15 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
```

6.103.16 Метод Range:getImages

Обеспечивает доступ к изображениям [DocumentAPI.Image](#) в диапазоне.



Внимание ! Данный метод может быть использован только в текстовом документе.

Пример:

```
local images = document:getRange():getImages()
for image in images:enumerate() do
    print(image:getFrame():getWrapType())
end
```

6.103.17 Метод Range:getInlineObjects

Обеспечивает доступ к перечислению [DocumentAPI.MediaObjects](#) графических объектов диапазона.



Внимание ! Данный метод может быть использован только в текстовом документе.

Пример для текстового документа:

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
    print(mediaObject)
end
```

6.104 Таблица DocumentAPI.RangeBorders

Таблица DocumentAPI.RangeBorders оставлена для совместимости. Вместо нее необходимо использовать таблицу [DocumentAPI.Borders](#).

6.105 Таблица DocumentAPI.RectU

Таблица DocumentAPI.RectU представляет описание прямоугольной области. Описание полей таблицы DocumentAPI.RectU представлено в таблице 72.

Таблица 72 – 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)
```

6.105.1 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)]
```

6.106 Таблица DocumentAPI.SaveDocumentSettings

Таблица `DocumentAPI.SaveDocumentSettings` предоставляет настройки, используемые для сохранения документа в файл (см. [document.saveAs\(\)](#)). Описание полей таблицы `DocumentAPI.SaveDocumentSettings` представлено в таблице 73.

Таблица 73 – Описание полей таблицы `DocumentAPI.SaveDocumentSettings`

Поле	Описание
<code>DocumentAPI.SaveDocumentSettings.documentFormat</code>	Формат документа DocumentFormat
<code>DocumentAPI.SaveDocumentSettings.documentType</code>	Тип документа DocumentType
<code>DocumentAPI.SaveDocumentSettings.documentPassword</code>	Пароль для защиты электронного документа от несанкционированного доступа
<code>DocumentAPI.SaveDocumentSettings.isTemplate</code>	Флаг, обозначающий, что документ должен быть сохранен как шаблон
<code>DocumentAPI.SaveDocumentSettings.dsvSettings</code>	Структура DSVSettings , необходимая для сохранения в формате DSV

6.107 Таблица DocumentAPI.ScaleFrom

В таблице 74 представлены позиции объекта, остающиеся неизменными при масштабировании объекта. Используется в [AbsoluteFrame.scale\(\)](#).

Таблица 74 – Неизменные позиции объекта при масштабировании

Наименование константы	Позиция
<code>DocumentAPI.ScaleFrom_BottomRight</code>	Правый нижний угол
<code>DocumentAPI.ScaleFrom_BottomLeft</code>	Левый нижний угол
<code>DocumentAPI.ScaleFrom_TopLeft</code>	Левый верхний угол
<code>DocumentAPI.ScaleFrom_TopRight</code>	Правый верхний угол

6.108 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 75.

Таблица 75 – 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())
```

6.109 Таблица DocumentAPI.Search

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

6.109.1 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.

Примеры:

```
search = DocumentAPI.createSearch(document)
-- Поиск по всему документу
ranges = search:findText("English")
for occurrence in ranges do
    print(occurrence:extractText())
end
```



```
end
-- Поиск только в диапазоне первого блока
local firstBlockRange = document:getBlocks():getBlock(0):getRange()
ranges = search:findText("English", firstBlockRange)
for occurrence in ranges do
    print(occurrence:extractText())
end
```

6.110 Таблица DocumentAPI.ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 76. Используется в качестве поля scriptPosition таблицы [DocumentAPI.TextProperties](#).

Таблица 76 – Типы надстрочного и подстрочного форматирования

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

Пример:

```
local props = DocumentAPI.TextProperties()
props.scriptPosition = DocumentAPI.ScriptPosition_SuperScript
range:setTextProperties(props)
```

6.111 Таблица DocumentAPI.Scripts

Таблица DocumentAPI.Scripts предоставляет доступ к списку макрокоманд документа. Коллекцию макрокоманд 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
```

6.111.1 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())
```

6.111.2 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)
```

6.111.3 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")
```

6.111.4 Метод `Scripts:enumerate`

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

Пример:

```
for script in document:getScripts():enumerate() do
    print(script:getName())
end
```

6.112 Таблица `DocumentAPI.Script`

Таблица `DocumentAPI.Script` предназначена для управления отдельной макрокомандой. Таблица содержит поля `Name` и `Body`.

6.112.1 Таблица `DocumentAPI.Scripting`

Таблица `DocumentAPI.Scripting` может быть получена путем вызова [DocumentAPI.createScripting\(\)](#) и содержит метод [runScript](#), который используется

для запуска макрокоманды.

6.112.1.1 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")
```

6.112.2 Метод `Script:getName`

Метод возвращает имя макрокоманды.

Пример:

```
local scripts = document:getScripts()
local sc = scripts:getScript("Enumerate scripts for document")
print(sc.getName())
```

6.112.3 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")
```

6.112.4 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)
```

6.112.5 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")
```

6.113 Таблица DocumentAPI.Sections

Таблица DocumentAPI.Sections представляет интерфейс для доступа к коллекции секций документа. Описание секции см. в разделе [Section](#).

6.113.1 Метод Sections:enumerate

Метод возвращает коллекцию секций документа.

Пример:

```
local sections = document:getSections()\nfor section in sections:enumerate() do\n    local properties = section:getPageProperties()\n    print(properties.width)\n    print(properties.height)\nend
```

6.114 Таблица DocumentAPI.Section

Таблица DocumentAPI.Section представляет собой раздел в документе.

6.114.1 Метод Section:setPageProperties

Метод устанавливает параметры [PageProperties](#) страниц, находящихся в разделе.

Пример:

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

6.114.2 Метод Section:getPageProperties

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

Пример:

```
local section = document:getBlocks():getBlock(0):getSection()\nlocal properties = section:getPageProperties()
```

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

6.114.3 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)
```

6.114.4 Метод `Section:getPageOrientation`

Метод возвращает ориентацию страниц раздела.

Пример:

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

6.114.5 Метод `Section:getRange`

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

Пример:

```
local sections = document:enumerateSections()
for section in sections do
    print(section:getRange():extractText())
end
```

6.114.6 Метод `Section:getHeaders`

Метод возвращает коллекцию [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
```

```
end  
end
```

6.114.7 Метод Section:getFooters

Метод возвращает коллекцию [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  
end
```

6.115 Таблица DocumentAPI.SizeU

Таблица `DocumentAPI.SizeU` представляет размер объекта в двухмерном пространстве. Описание полей таблицы `DocumentAPI.SizeU` представлено в таблице 77.

Таблица 77 – 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)
```

6.115.1 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)
```

6.116 Таблица DocumentAPI.Shape

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

6.116.1 Метод Shape:getShapeProperties

Метод возвращает свойства фигуры [DocumentAPI.ShapeProperties](#).

Пример:

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

6.116.2 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)
```

6.117 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](#).

6.118 Таблица DocumentAPI.ShapeTextLayout

Таблица DocumentAPI.ShapeTextLayout описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 78. Используется в таблице [DocumentAPI.ShapeProperties](#).

Таблица 78 – 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

6.119 Таблица DocumentAPI.Table

Таблица DocumentAPI.Table предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 66).

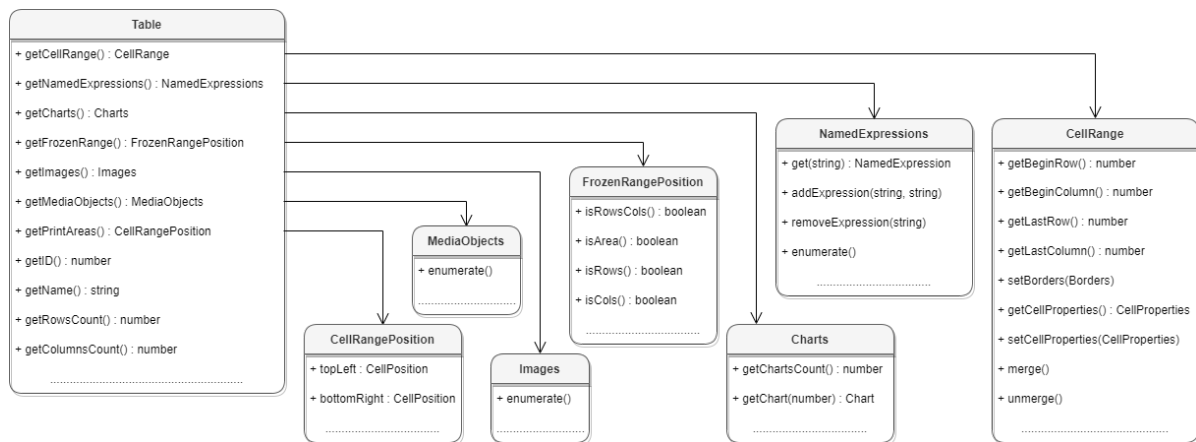


Рисунок 66 – Структура полей таблицы DocumentAPI.Table

6.119.1 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")
```

6.119.2 Метод Table:getName

Метод позволяет получить наименование листа табличного документа.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getName())
```

6.119.3 Метод Table:getRowCount

Метод позволяет получить количество строк таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getRowCount())
```

6.119.4 Метод Table:getColumnsCount

Метод позволяет получить количество столбцов таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getColumnsCount())
```

6.119.5 Метод Table:getCell

Метод позволяет получить доступ к отдельной ячейке таблицы [Cell](#). В качестве аргумента может выступать текстовое представление адреса ячейки, либо экземпляр таблицы [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())
```

6.119.6 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
```

6.119.7 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)
```

6.119.8 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)
```

6.119.9 Метод Table:insertRowAfter

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

Вызов:

```
insertRowAfter( rowIndex, copyRowStyle, rowsCount )
```

Параметры:

- rowIndex – индекс строки в таблице, после которой производится вставка. Индексация строк начинается с нуля.
- copyRowStyle – флаг наследования стиля. Если этот параметр установлен в значение true, то новая строка наследует настройки форматирования строки с индексом rowIndex. Если параметр copyRowStyle установлен в значение false, то настройки форматирования не копируются. Значение по умолчанию true.

- `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)
```

6.119.10 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)
```

6.119.11 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.

6.119.12 Метод `Table:removeRow`

Метод предназначен для удаления строки таблицы, начиная с заданного индекса.

Вызов:

```
removeRow(rowIndex, rowsCount)
```

Параметры:

- `rowIndex` – индекс строки, начиная с которого будет удалено `rowsCount` строк. Индексация строк начинается с нуля.
- `rowsCount` – количество строк для удаления. Значение по умолчанию 1.

6.119.13 Метод `Table:groupRows`

Метод предназначен для группировки строк таблицы, начиная с заданного индекса.

Индексация строк начинается с нуля.

Вызов:

```
groupRows(rowIndex, rowsCount)
```

Параметры:

- `rowIndex` – индекс строки, начиная с которого будет начата группировка строк;
- `rowsCount` – количество строк для группировки.

6.119.14 Метод `Table:ungroupRows`

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

Индексация строк начинается с нуля.

Вызов:

```
ungroupRows(rowIndex, rowsCount)
```

Параметры:

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

6.119.15 Метод `Table:clearRowGroups`

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

Вызов:

```
clearRowGroups(rowIndex, rowCount)
```

Параметры:

- `rowIndex` – индекс строки, начиная с которой будет начата очистка групп;
- `rowCount` – количество строк для очистки групп.

6.119.16 Метод `Table:groupColumns`

Метод предназначен для группировки столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
groupColumns(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет начата группировка столбцов;
- `columnsCount` – количество столбцов для группировки.

6.119.17 Метод `Table:ungroupColumns`

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

Вызов:

```
ungroupColumns(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет начата разгруппировка столбцов;
- `columnsCount` – количество столбцов для разгруппировки.

6.119.18 Метод `Table:clearColumnGroups`

Метод предназначен для очистки группированных столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
clearColumnGroups(columnIndex, columnsCount)
```

Параметры:

- columnIndex – индекс столбца, начиная с которого будет начата очистка групп;
- columnsCount – количество столбцов для очистки групп.

6.119.19 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)
```

6.119.20 Метод Table:setRowHeight

Метод устанавливает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов:

```
setRowHeight(rowIndex, height)
```

Параметры:

- rowIndex – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- height – высота строки в пунктах (1/72 дюйма).
- rowHeightRule – точность значения (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)
```

6.119.21 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()
```

6.119.22 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()
```

6.119.23 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)
```

6.119.24 Метод Table:setShowZeroValue

Для упрощения чтения таблицы нулевые значения ячеек могут быть скрыты. Для управления скрытием/показом ячеек используется метод `setShowZeroValue`.

Пример:

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(true)
```


6.119.25 Метод Table:showZeroValue

Для проверки режима отображения нулевых значений ячеек используется метод `showZeroValue`.

Пример:

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(false)
print(tbl:showZeroValue())
```

6.119.26 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)
```

6.119.27 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
```

6.119.28 Метод Table:setColumnsVisible

Метод `Table::setColumnsVisible` позволяет задавать видимость столбцов, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
setColumnsVisible(first, columnsCount, visible)
```

Параметры:

`first` – начальный индекс;
`columnsCount` – количество столбцов;
`visible` – видимость.

6.119.29 Метод `Table:setRowsVisible`

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

Вызов:

```
setRowsVisible(first, rowsCount, visible)
```

Параметры:

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

6.119.30 Метод `Table:getCharts`

Для получения списка диаграмм [Charts](#) таблицы используется метод `Table:getCharts`.

Пример:

```
for tbl in document:getBlocks():enumerateTables() do
  print(tbl:getCharts():getChartsCount())
end
```

6.119.31 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())
```

6.119.32 Метод Table:freeze

Метод freeze закрепляет заданную область [DocumentAPI.FrozenRangePosition](#) таблицы.

Пример:

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local tbl = document:getBlocks():getTable(0)
tbl:freeze(frozenRangePosition)
print(tbl:getFrozenRange():isCols())
```

6.119.33 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 ~= nil and tbl2 ~= nil and tbl1 == tbl2 then
    print("tbl1 и tbl2 ссылаются на общую таблицу в документе")
end
```

6.119.34 Метод Table:__ne

Метод используется для определения неэквивалентности двух таблиц.

Пример:

```
local tbl1 = document:getBlocks():getTable(0)
local tbl2 = document:getBlocks():getTable("Table Name")
if tbl1 ~= nil and tbl2 ~= nil and tbl1 ~= tbl2 then
    print("tbl1 и tbl2 ссылаются на разные таблицы в документе")
end
```

6.119.35 Метод Table:setPrintArea

Метод служит для установки и сброса области печати [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
```

6.119.36 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())
```

6.119.37 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())
```

6.119.38 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
```

6.119.39 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
```

6.119.40 Метод Table:getNamedExpressions

Для получения списка именованных диапазонов [DocumentAPI:NamedExpressions](#) используется метод `Table:getNamedExpressions()`.

6.120 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)
```

6.121 Таблица `DocumentAPI.TextAnchoredPosition`

Таблица `DocumentAPI.TextAnchoredPosition` (см. Рисунок 67) представляет позицию объекта на странице текстового документа. Используется в качестве позиции в таблице [DocumentAPI.InlineFrame](#). Пример использования см. в разделе [InlineFrame.setPosition\(\)](#).

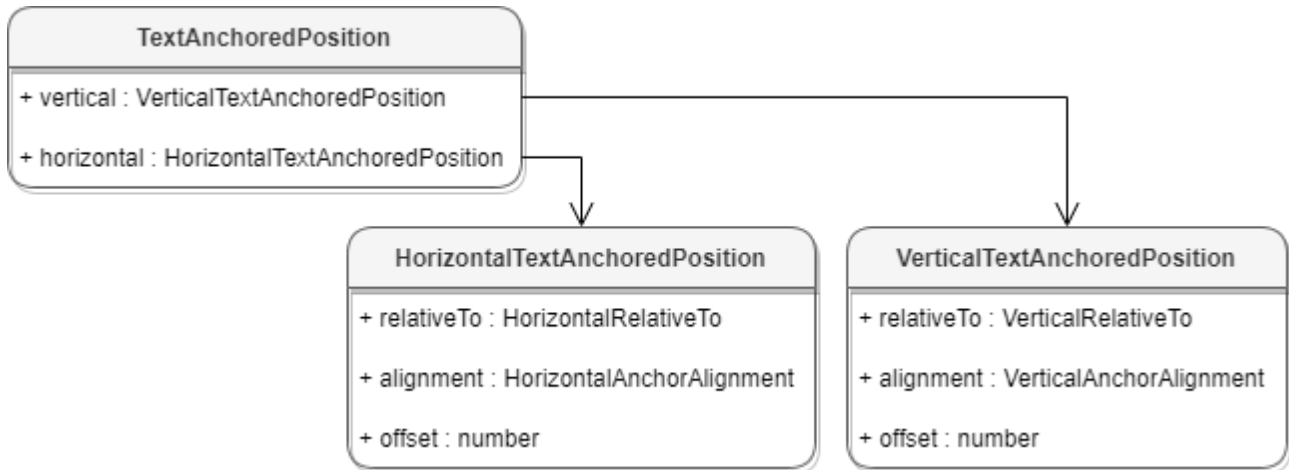


Рисунок 67 – Поля таблицы `DocumentAPI.TextAnchoredPosition`

Описание полей таблицы `DocumentAPI.TextAnchoredPosition` представлено в таблице 80.

Таблица 80 – Описание полей таблицы `DocumentAPI.TextAnchoredPosition`

Поле	Описание
<code>DocumentAPI.TextAnchoredPosition.vertical</code>	Позиция по вертикали VerticalTextAnchoredPosition
<code>DocumentAPI.TextAnchoredPosition.horizontal</code>	Позиция по горизонтали HorizontalTextAnchoredPosition

6.122 Таблица `DocumentAPI.TextProperties`

Таблица `DocumentAPI.TextProperties` содержит поля, задающие параметры текста. На рисунке 68 изображена объектная модель таблицы `DocumentAPI.TextProperties`.

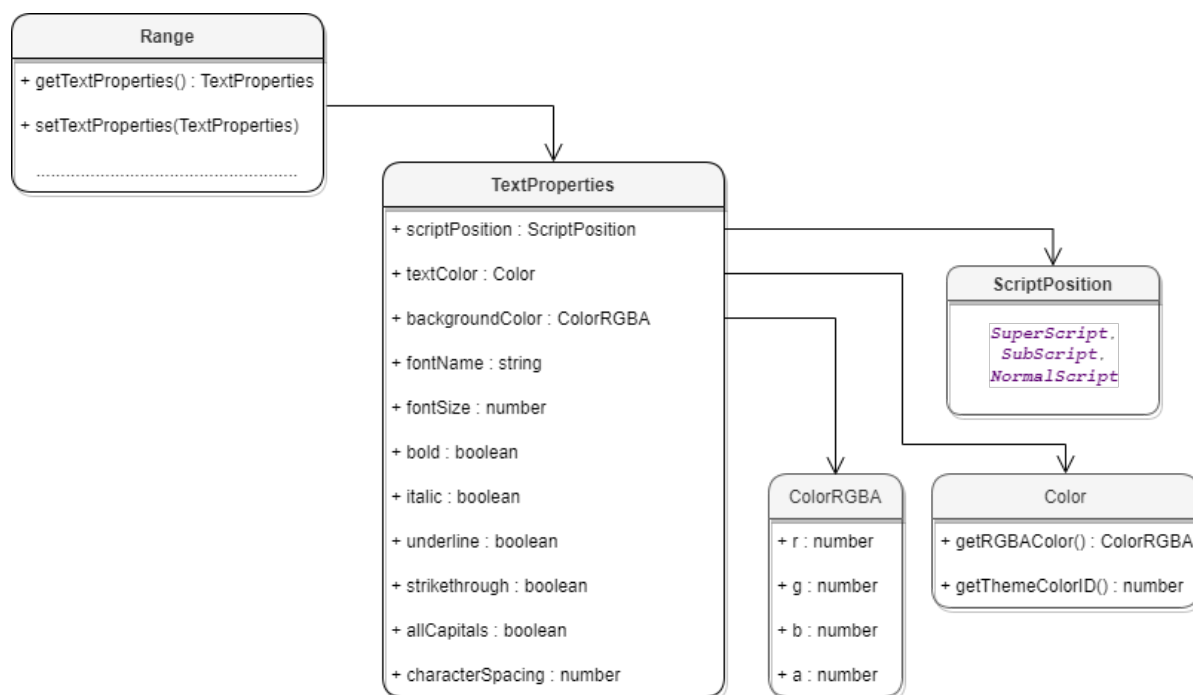


Рисунок 68 – Объектная модель для работы с таблицей

DocumentAPI.TextProperties

Описание полей таблицы DocumentAPI.TextProperties представлено в таблице 81.

Свойства DocumentAPI.TextProperties применяются к диапазону текста DocumentAPI.Range (методы [Range.getTextProperties\(\)](#), [Range.setTextProperties\(\)](#)).

Таблица 81 – 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.

Поле	Тип	Описание
		Значение <code>false</code> устанавливает все буквы указанного фрагмента текста как строчные
<code>TextProperties.scriptPosition</code>	ScriptPosition	Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме
<code>TextProperties.textColor</code>	Color	Цвет указанного фрагмента документа
<code>TextProperties.backgroundColor</code>	ColorRGBA	Цвет фона указанного фрагмента документа
<code>TextProperties.characterSpacing</code>	Числовое	Размер межсимвольного интервала

Пример:

```
local props = DocumentAPI.TextProperties()
props.fontName = "XO Oriel"
props.fontSize = 20

-- доступ к тексту третьего абзаца
local range = document:getBlocks():getParagraph(2):getRange()

-- установить свойства фрагмента текста
range:setTextProperties(props)
```

6.123 Таблица DocumentAPI.TextWrapType

В таблице 82 представлены варианты обтекания текстом встроенного объекта. Используется в [InlineFrame.setWrapType\(\)](#).

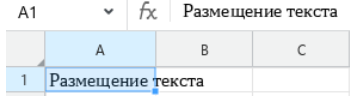
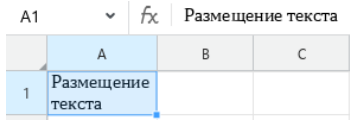
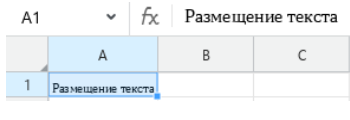
Таблица 82 – Варианты обтекания текстом встроенного объекта

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

6.124 Таблица DocumentAPI.TextLayout

В таблице 83 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле `textLayout` таблицы [CellProperties](#).

Таблица 83 – 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)
```

6.125 Таблица DocumentAPI.TextOrientation

Таблица `DocumentAPI.TextOrientation` предоставляет доступ к свойству ориентации текста (тип `number`) в ячейке, фигуре и т. д. Используется как тип поля `textOrientation` в [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(45.5)
cell:setCellProperties(props)
print(props.textOrientation:getAngle())
```

6.125.1 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())
```

6.126 Таблица `DocumentAPI.TimePatterns`

Форматы времени представлены в таблице 84. Пример использования см. в главе [DocumentAPI.DateTimeCellFormatting](#).

Таблица 84 – Форматы времени

Наименование константы	Описание
<code>DocumentAPI.TimePatterns_ShortTime</code>	'hh:mm AM/PM' для языка en_US
<code>DocumentAPI.TimePatterns_LongTime</code>	'hh:mm:ss AM/PM' для языка en_US

6.127 Таблица `DocumentAPI.TimeZone`

Таблица `DocumentAPI.TimeZone` предоставляет настройки, необходимые для экспорта текстовых документов. Используется в поле `timeZone` таблицы [DocumentAPI.DocumentSettings](#).

Поле таблицы `DocumentAPI.TimeZone.offsetInSecondsToUTC` (числовой тип) содержит значение, с помощью которого задается смещение или разность между временем в данном часовом поясе и временем в формате UTC (Всемирное координированное время).

6.128 Таблица `DocumentAPI.ThemeColorID`

В таблице 85 представлены типы идентификаторов цветов тем. Используется в [DocumentAPI.Color](#).

Таблица 85 – Типы идентификаторов цветов тем

Наименование константы	Описание
<code>DocumentAPI.ThemeColorID_Background1</code>	Фон1
<code>DocumentAPI.ThemeColorID_Text1</code>	Текст1
<code>DocumentAPI.ThemeColorID_Background2</code>	Фон2

Наименование константы	Описание
DocumentAPI.ThemeColorID_Text2	Текст2
DocumentAPI.ThemeColorID_Dark1	Темная1
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

6.129 Таблица DocumentAPI.TrackedChange

Таблица DocumentAPI.TrackedChange представляет отслеживаемое изменение в диапазоне документа (см. Рисунок 69).

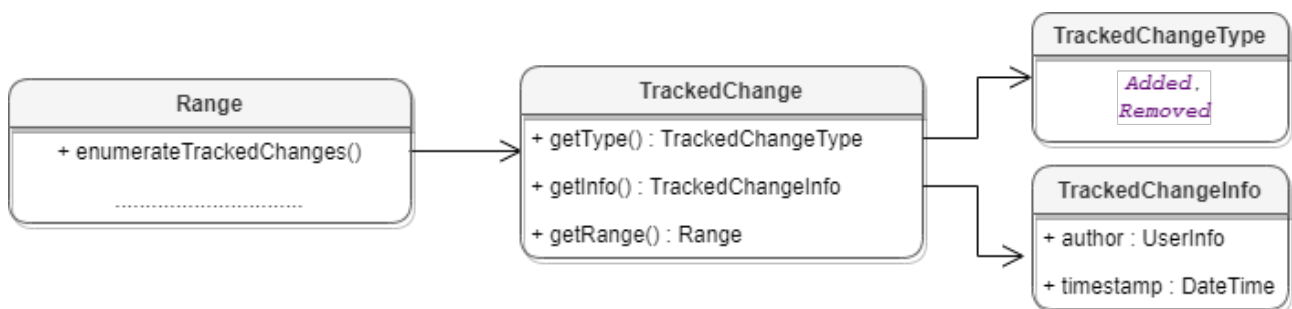


Рисунок 69 – Объектная модель таблиц для работы с отслеживаемыми изменениями

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
```

6.129.1 Метод `TrackedChange:getRange`

Метод возвращает объект [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
```

6.129.2 Метод `TrackedChange:getType`

Метод позволяет получить информацию о типе отслеживаемого изменения [TrackedChangeType](#).

Пример:

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
    print(tracked_change:getType())
end
```

6.129.3 Метод `TrackedChange:getInfo`

Метод позволяет получить информацию об отслеживаемых изменениях [TrackedChangeInfo](#).

Пример:

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
    print(tracked_change:getInfo().author.name)
end
```

6.130 Таблица `DocumentAPI.TrackedChangeInfo`

Таблица `DocumentAPI.TrackedChangeInfo` содержит информацию об отслеживаемых изменениях. Описание полей таблицы представлено в таблице 86.

Таблица 86 – Описание полей таблицы DocumentAPI.TrackedChangeInfo

Поле	Тип	Описание
DocumentAPI.TrackedChangeInfo.author	UserInfo	Автор изменений
DocumentAPI.TrackedChangeInfo.timeStamp	DateTime	Дата и время изменений

Пример:

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
    local trackedChangeInfo = change:getInfo()
    local author = trackedChangeInfo.author
    local ts = trackedChangeInfo.timeStamp
    local time = string.format("%d/%d/%d - %d:%d:%d", ts.day, ts.month, ts.year,
ts.hour, ts.minute, ts.second)
    print(author.name, time)
end
```

6.131 Таблица DocumentAPI.TrackedChangeType

Типы отслеживаемых изменений представлены в таблице 87.

Таблица 87 – 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
```

6.132 Таблица DocumentAPI.UserInfo

Таблица DocumentAPI.UserInfo предоставляет информацию о пользователе. Используется в поле author таблицы [DocumentAPI.TrackedChangeInfo](#), а также в поле userInfo таблицы [DocumentAPI.DocumentSettings](#).

Описание полей таблицы DocumentAPI.UserInfo представлено в таблице 88.

Таблица 88 – Описание полей таблицы `DocumentAPI.UserInfo`

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

6.133 Таблица `DocumentAPI.WorksheetPrinterFitType`

В таблице 89 представлены варианты масштабирования при печати табличных документов. Используется в качестве поля `worksheetPrinterFitType` таблицы [DocumentAPI.PrintSettings](#).

Таблица 89 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
<code>DocumentAPI.WorksheetPrinterFitType_ActualSize</code>	Фактический размер
<code>DocumentAPI.WorksheetPrinterFitType_ByPageScale</code>	По масштабу страницы
<code>DocumentAPI.WorksheetPrinterFitType_ByPageBreaksOnly</code>	По разрыву страниц
<code>DocumentAPI.WorksheetPrinterFitType_FitToPage</code>	Вписать в страницу
<code>DocumentAPI.WorksheetPrinterFitType_FitToWidth</code>	Вписать по ширине
<code>DocumentAPI.WorksheetPrinterFitType_FitToHeight</code>	Вписать по высоте

6.134 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 90.

Таблица 90 – Описание полей таблицы `DocumentAPI.ValueFieldsOrientation`

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

6.135 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
```

6.135.1 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
```

6.135.2 Метод VectorString:max_size

Метод возвращает максимальный размер вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
print(vector:max_size())
```

6.135.3 Метод VectorString:empty

Метод возвращает true, если вектор не содержит элементов.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
print(vector:empty()) -- false
```

6.135.4 Метод VectorString:clear

Метод очищает содержимое вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:clear()
print(vector:empty()) -- true
```

6.135.5 Метод `VectorString:push_back`

Метод добавляет элемент в конец вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
print(vector:size()) -- 1
```

6.135.6 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
```

6.135.7 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
```

6.135.8 Метод `VectorString:back`

Метод возвращает последний элемент вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:push_back("13")
print(vector:back()) -- 13
```

6.135.9 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
```

6.135.10 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
```

6.136 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
```

6.136.1 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
```

6.136.2 Метод `VectorUInt:max_size`

Метод возвращает максимальный размер вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()  
print(vector:max_size())
```

6.136.3 Метод `VectorUInt:empty`

Метод возвращает `true`, если вектор не содержит элементов.

Пример:

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
print(vector:empty()) -- false
```

6.136.4 Метод `VectorUInt:clear`

Метод очищает содержимое вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
vector:clear()  
print(vector:empty()) -- true
```

6.136.5 Метод `VectorUInt:push_back`

Метод добавляет элемент в конец вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
print(vector:size()) -- 1
```

6.136.6 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
```

6.136.7 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
```

6.136.8 Метод VectorUInt:back

Метод возвращает последний элемент вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()  
vector:push_back(12)  
vector:push_back(13)  
print(vector:back()) -- 13
```

6.136.9 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
```

6.136.10 Метод VectorUInt:__setitem

Метод устанавливает элемент вектора по заданному индексу.

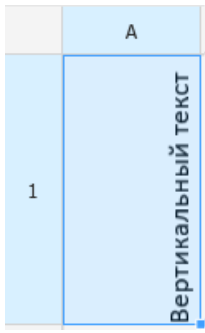


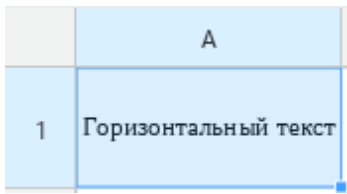

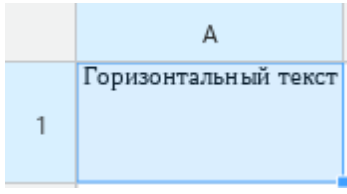
Пример:

```
local vector = DocumentAPI.VectorUInt(2)  
vector:__setitem(0, 12)  
vector:__setitem(1, 13)  
print(vector:__getitem(0)) -- 12  
print(vector:__getitem(1)) -- 13
```

6.137 Таблица DocumentAPI.VerticalAlignment

В таблице 91 представлены константы видов выравнивания текста по вертикали. Используется в [DocumentAPI.CellProperties](#), [DocumentAPI.ShapeProperties](#).

Таблица 91 – 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)
    
```

6.138 Таблица DocumentAPI.VerticalTextAnchoredPosition

Таблица `DocumentAPI.VerticalTextAnchoredPosition` (см. Рисунок 70) предназначена для управления относительным положением объекта со смещением или выравниванием по вертикали. Пример использования см. в [InlineFrame.setPosition\(\)](#).

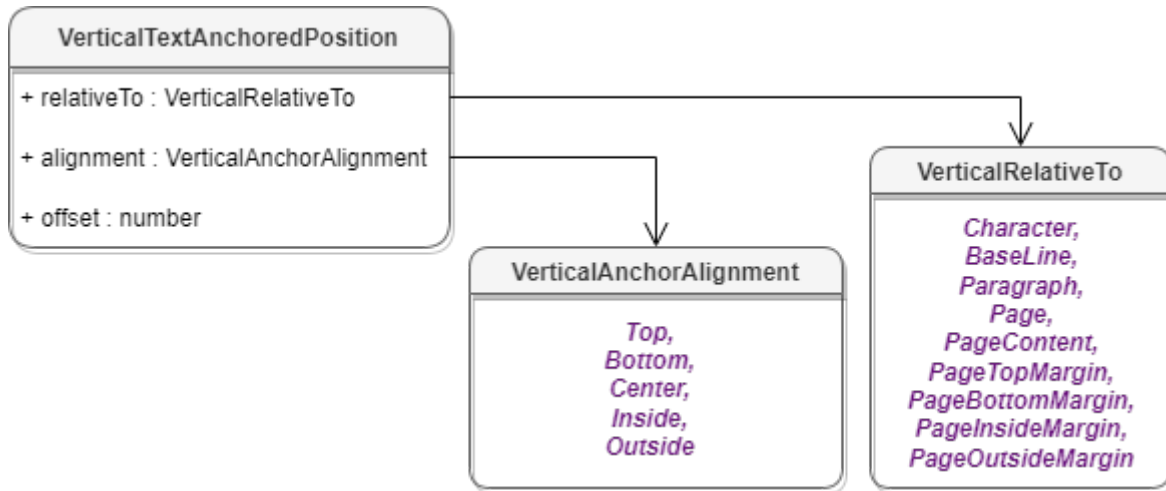


Рисунок 70 – Поля таблицы `DocumentAPI.VerticalTextAnchoredPosition`

Описание полей таблицы `DocumentAPI.VerticalTextAnchoredPosition` представлено в таблице 92.

Таблица 92 – 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

6.139 Таблица DocumentAPI.VerticalRelativeTo

В таблице 93 представлены типы размещения объекта относительно закрепленной позиции по вертикали.

Таблица 93 – Типы размещения объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
<code>DocumentAPI.VerticalRelativeTo_Character</code>	Символ

Наименование константы	Описание
DocumentAPI.VerticalRelativeTo_BaseLine	Базовая линия
DocumentAPI.VerticalRelativeTo_Paragraph	Абзац
DocumentAPI.VerticalRelativeTo_Page	Страница
DocumentAPI.VerticalRelativeTo_PageContent	Содержимое страницы
DocumentAPI.VerticalRelativeTo_PageTopMargin	Верхнее поле страницы
DocumentAPI.VerticalRelativeTo_PageBottomMargin	Нижнее поле страницы
DocumentAPI.VerticalRelativeTo_PageInsideMargin	Внутреннее поле страницы
DocumentAPI.VerticalRelativeTo_PageOutsideMargin	Внешнее поле страницы

6.140 Таблица DocumentAPI.VerticalAnchorAlignment

В таблице 94 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали.

Таблица 94 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
DocumentAPI.VerticalAnchorAlignment_Top	По верхнему краю
DocumentAPI.VerticalAnchorAlignment_Bottom	По нижнему краю
DocumentAPI.VerticalAnchorAlignment_Center	По центру
DocumentAPI.VerticalAnchorAlignment_Inside, DocumentAPI.VerticalAnchorAlignment_Outside	По границам

7 Справочник функций EditorAPI

Глобальная таблица EditorAPI содержит функции доступа к внешней функциональности редактора.

7.1 Функция EditorAPI.messageBox

Функция EditorAPI.messageBox() выводит на экран сообщение с заданным текстом и отображением кнопки **ОК**, при этом исполнение макроккоманды 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:

```
local Actions = {}
function Actions.printCell(context)
    context.doWithSelection(function(range)
        for cell in range:enumerate() do
            EditorAPI.messageBox(cell:getFormattedValue())
        end
    end)
end
return Actions
```

7.2 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:

```
local Actions = {}
function Actions.printDocument(context)
    context.doWithDocument(function(document)
```

```
        local printDocumentResult = EditorAPI.showPrintDialog()  
        print(printDocumentResult)  
    end)  
end  
return Actions
```

7.3 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 Actions = {}  
function Actions.printDocument(context)  
    local printSettings = {}  
    printSettings.printSelection = true  
    EditorAPI.printDocument(printSettings)  
end  
return Actions
```

7.4 Функция EditorAPI.isPrinterAvailable

Функция `EditorAPI.isPrinterAvailable()` позволяет проверить доступность последнего использованного принтера. Возвращает `false`, если принтер недоступен.

Пример:

```
local Actions = {}  
function Actions.checkPrinter(context)  
    if EditorAPI.isPrinterAvailable() then  
        EditorAPI.messageBox("Printer is available")  
    else  
        EditorAPI.messageBox("Printer is not available")  
    end  
end  
return Actions
```


8 Справочник функций пользовательского интерфейса надстроек

8.1 Таблица ui

Таблица предоставляет доступ к методам для определения пользовательского интерфейса в надстройках редакторов МойОфис с помощью виджетов (widgets) и компоновок (layouts), представленных в таблице [Forms](#).

Таблица ui была введена специально для того, чтобы облегчить описание разработчиками пользовательского интерфейса в виде наглядной древовидной структуры.

8.1.1 Таблица ui.Dialog

Реализует виджет **Диалоговое окно** (см. Рисунок 71).

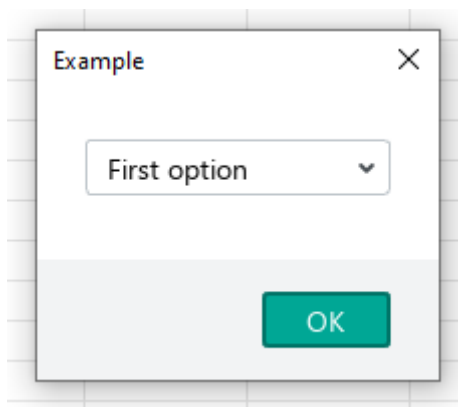


Рисунок 71 – Внешний вид виджета **ui:Dialog**

8.1.1.1 Открытие диалогового окна

Пример открытия модального диалогового окна:

```
local dialog = ui:Dialog {
  Size = Forms.Size(600, 300),
  ui:Column {
    ui:Row {
      ui:Spacer {},
      ui:Column {lblHello},      -- Надпись
      ui:Column {txtEntryField}, -- Текстовое поле
      ui:Spacer {}
    }
  }
}
context.showDialog(dialog)
```

8.1.1.2 Конструктор `ui:Dialog`

Конструктор элемента **Диалоговое окно**, позволяет установить свойства виджета в момент создания.

Разработчик в момент создания элемента может задать его свойства:

- `Name : string` – внутреннее имя (идентификатор) виджета **Диалоговое окно** в диалоговом окне;
- `Enabled : boolean` – статус виджета (активный/неактивный);
- `Size : Forms.Size` – размеры виджета;
- `OnDone : function(integer)` – функция, вызываемая при закрытии диалогового окна пользователем.

Пример:

```
uiDialogButtons = ui:DialogButtons{ Forms.DialogButton_OK,  
Forms.DialogButton_Cancel }  
  
label = ui:Label {  
    Name = "lblHello",  
    Text = "Здравствуй, мир!" --,  
}  
textBox = ui:TextBox {  
    Name = "TextBox",  
    Text = "Напишите что-нибудь" --,  
}  
onDone = function(ret)  
    if ret == Forms.DialogCode_Accepted then  
        -- OK Pressed  
    end  
end  
  
dlg = ui:Dialog {  
    Name = "Форма приветствия",  
    Enabled = true,  
    Size = Forms.Size(600, 300),  
    Buttons = uiDialogButtons,  
    OnDone = onDone,  
    ui:Column {  
        ui:Row {  
            ui:Spacer {},
```

```
        ui:Column {label},
        ui:Spacer {}
    }
    ui:Row {textBox}
}
}
context.showDialog(dlg)
```

8.1.1.3 Метод Dialog:setName

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
dialog = ui:Dialog {}
dialog.setName("uiDialog")
```

8.1.1.4 Метод Dialog:getName

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
dialog = ui:Dialog {}
local name = dialog.getName()
```

8.1.1.5 Метод Dialog:setEnabled

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
dialog = ui:Dialog {}
dialog.setEnabled(true)
```

8.1.1.6 Метод Dialog:isEnabled

Используется для получения статуса виджета (активный/неактивный).

Пример:

```
dialog = ui:Dialog {}
local state = dialog.isEnabled()
```

8.1.1.7 Метод Dialog:setSize

Используется для установки значений размеров [Forms.Size](#) виджета.

Пример:

```
dialog = ui:Dialog {}  
dialog:setSize(Forms.Size(600, 300))
```

8.1.1.8 Метод Dialog:getSize

Используется для получения значений размеров [Forms.Size](#) виджета.

Пример:

```
dialog = ui:Dialog {}  
local size = dialog:getSize()
```

8.1.1.9 Метод Dialog:setOnDone

Используется для установки функции обработчика диалогового окна (тип: `function`). Функция вызывается, когда диалог был закрыт пользователем, и возвращает значение в зависимости от нажатой пользователем кнопки (см. раздел [Forms.DialogCode](#)).

Если пользователь нажал кнопку **Отмена**, то возвращается значение `DialogCode_Rejected`, если пользователь нажал кнопку **ОК**, то возвращается значение `DialogCode_Accepted`.

Пример:

```
local contextWrapper = { context = nil}  
local dialogButtons = ui:DialogButtons{}  
  
dialogButtons:addButton("OK", Forms.DialogButtonRole_Accept)  
dialogButtons:addButton("Cancel", Forms.DialogButtonRole_Reject)  
  
dlg = ui:Dialog {  
  Name = "Hello, world!"  
  Size = Forms.Size(600, 300),  
  Buttons = dialogButtons,  
  ui:Column {  
    ui:Row {lblHello},  
    ui:Row {txtEntryField}  
  }  
}  
dlg:setOnDone(function(ret)  
  if ret == Forms.DialogButtonRole_Accept then
```

```
        -- OK pressed
    else
        -- Cancel pressed
    end
end)

contextWrapper.handler = function(context)
    contextWrapper.context = context
    context.doWithDocument(function(document)
        context.showDialog(dlg)
    end)
end
```

Стоит обратить особое внимание на то, что если в обработке результата необходимо производить действия с документом, стоит использовать `context.doWithDocument`.

```
dlg:setOnDone(function(ret)
    if ret == Forms.DialogButtonRole_Accept then
        contextWrapper.context.doWithDocument(function(document)
            local firstSheet = document:getBlocks():getTable(0)
            local cell = firstSheet:getCell("A1")
            cell:setText("Cell content")
            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)
        end)
    else
        -- Cancel pressed
    end
end)
```

8.1.1.10 Метод `Dialog:setButtons`

Используется для установки диалоговых кнопок [ui:DialogButtons](#). По умолчанию в диалоговом окне есть только кнопка **ОК**.

Пример:

```
local dialog = ui:Dialog{}
local buttons = ui:DialogButtons{ Forms.DialogButton_OK,
```

```
Forms.DialogButton_Cancel }  
dialog:setButtons(buttons)
```

8.1.1.11 Метод Dialog:done

Метод используется для принудительного закрытия диалогового окна. В качестве параметра передается код закрытия диалога.

Пример:

```
local dialog = ui:Dialog{}  
local buttons = ui:DialogButtons {  
  ui:Button {  
    Title = "Done",  
    OnClick = function()  
      onDialogDone()  
      dialog:done(0)  
    end  
  }  
}  
dialog:setButtons(buttons)
```

8.1.2 Таблица ui.DialogButtons

Используется для создания стандартных и пользовательских кнопок в нижней части диалогового окна.

8.1.2.1 Конструктор ui:DialogButtons

Конструктор элемента **DialogButtons** позволяет создать набор кнопок в одном операторе.

Пример:

```
dialogButtons = ui:DialogButtons  
{  
  -- Стандартная кнопка  
  Forms.DialogButton_OK,  
  Forms.DialogButton_Cancel,  
  
  -- Пользовательская кнопка со стандартным действием  
  {  
    title = "Button1",  
    role = Forms.DialogButtonRole_Accept  
  },  
}
```

```
-- Полностью пользовательская кнопка
ui:Button {}
}

dialog = ui:Dialog {
    .....
    Buttons = dialogButtons
    .....
}
```

8.1.2.2 Метод DialogButtons:addButton

Создает кнопку заданного типа. Существует несколько вариантов реализации.

```
addButton(button: ui:Button, role : Forms.DialogButtonRole)
addButton(title: string, role : Forms.DialogButtonRole)
addButton(button : Forms.DialogButton)
```

Пример:

```
local btnStart = ui:Button {
    Name = "btnStart",
    Title = "Начать обработку"
}

local dialogButtons = ui:DialogButtons{}
dialogButtons:addButton(btnStart)
dialogButtons:addButton("Закреть", Forms.DialogButtonRole_Reject)
dialogButtons:addButton(Forms.DialogButton_Done)

dlg = ui:Dialog {
    .....
    Buttons = dialogButtons
    .....
}
```

В результате выполнения данного примера на экране отобразится диалог (см. Рисунок 72).

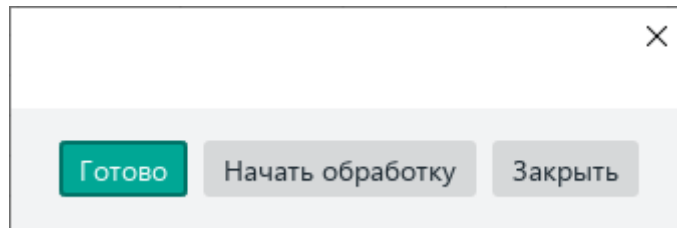


Рисунок 72 – Пример создания кнопок диалога

8.1.3 Таблица `ui:FolderDialog`

Используется для реализации диалога выбора папки (см. Рисунок 73).

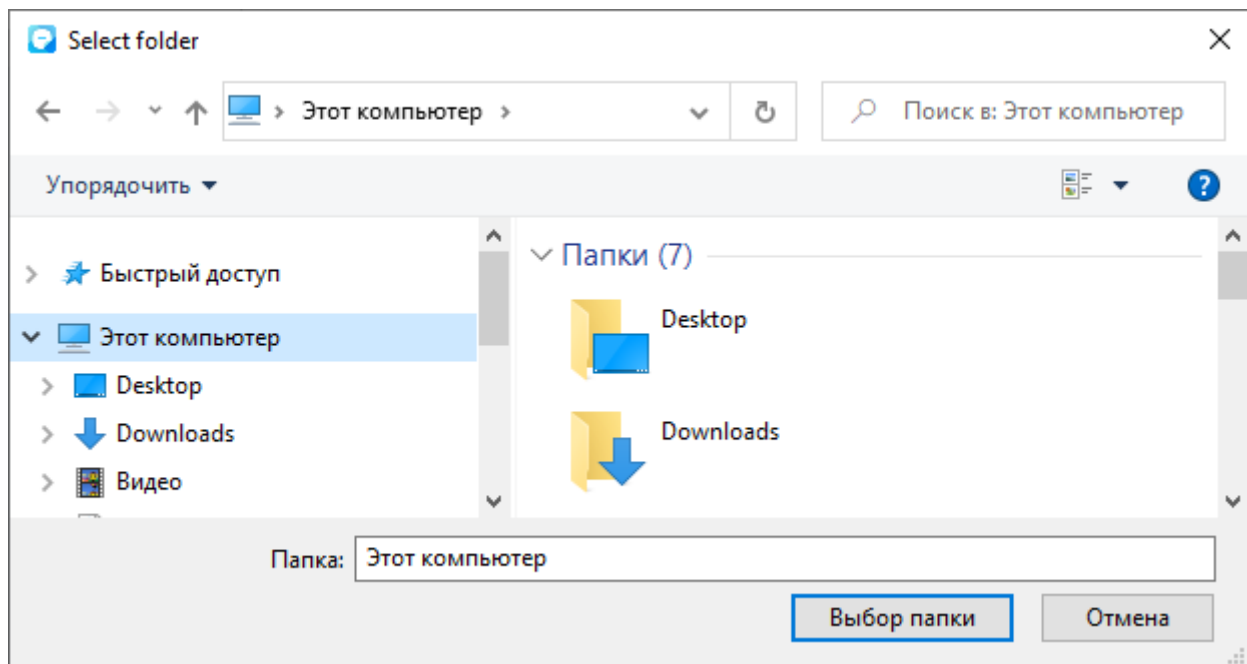


Рисунок 73 – Внешний вид виджета `ui:FolderDialog`

8.1.3.1 Конструктор `ui:FolderDialog`

Конструктор элемента `FolderDialog` позволяет установить свойства виджета в момент создания.

Пример:

```
local showFolderDialog = function(context)
    local dialog = ui:FolderDialog {
        Title = "Select folder",
        InitialDirectory = SomeDefaultDirectory,
        OnDone = function(path)
            if #path > 0 then
                context.doWithDocument(function(document)
                    document:getBlocks():getTable
```



```
("Sheet1"):getCell("A2"):setText(path)
        end)
    end
end
}
context.showDialog(dialog)
end
```

8.1.3.2 Метод setTitle

Задаёт заголовок диалогового окна.

```
setTitle(title : string)
```

8.1.3.3 Метод getTitle

Возвращает заголовок диалогового окна.

```
getTitle() : string
```

8.1.3.4 Метод setInitialDirectory

Устанавливает стартовый каталог при открытии диалога.

```
setInitialDirectory(path : string)
```

8.1.3.5 Метод getInitialDirectory

Возвращает стартовый каталог диалога.

```
getInitialDirectory() : string
```

8.1.3.6 Метод setOnDone

Задаёт обработчик, который вызывается при выборе папки и закрытии диалога. Возвращается параметр path – выбранный путь к папке.

```
setOnDone(handler : function(path : string))
```

8.1.4 Таблица ui.FileOpenDialog

Используется для реализации диалога выбора файлов. Поддерживается режим множественного выбора (см. Рисунок 74).

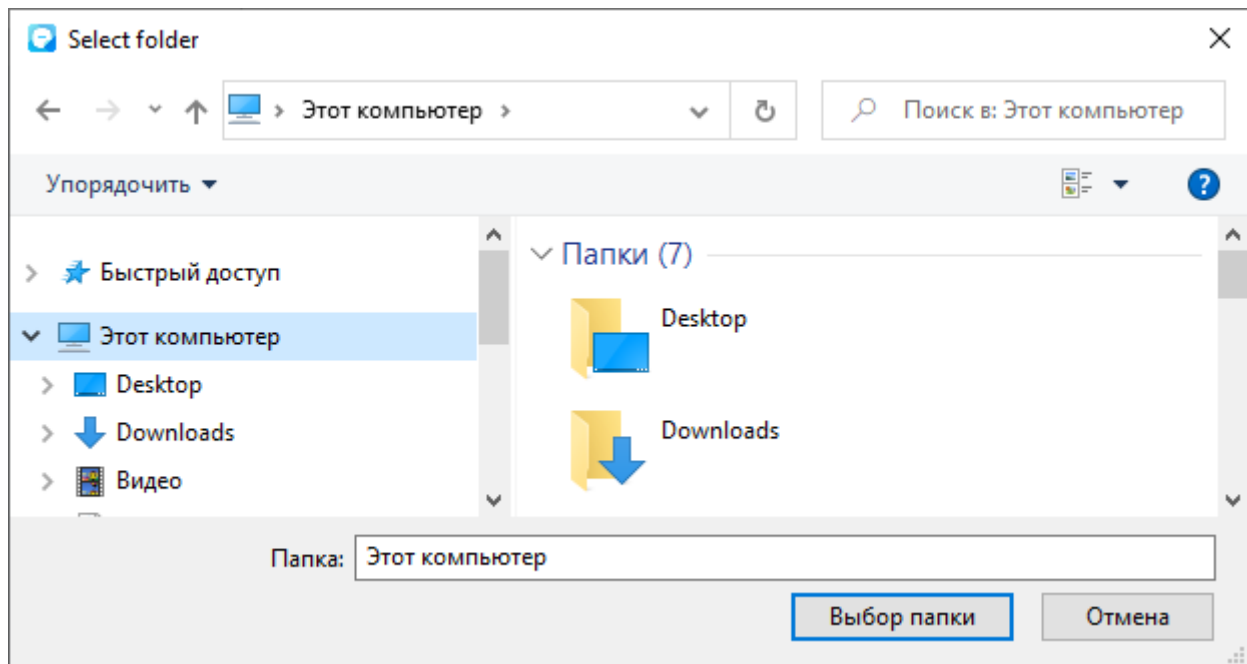


Рисунок 74 – Внешний вид виджета **ui:FileOpenDialog**

8.1.4.1 Конструктор **ui:FileOpenDialog**

Конструктор элемента **FileOpenDialog** позволяет установить свойства виджета в момент создания.

Пример:

```
local showFileOpenDialog = function(context)
    local dialog = ui:FileOpenDialog {
        Title = "Select configuration file",
        InitialDirectory = SomeDefaultDirectory,
        Filter = "Text Files (*.txt);;XML Files (*.xml)",
        AllowMultiSelect = false,
        OnDone = function(paths, filter)
            if paths:size() > 0 then
                context.doWithDocument(function(document)
                    document:getBlocks():getTable
("Sheet1"):getCell("A2"):setText(paths[0])
                    document:getBlocks():getTable
("Sheet1"):getCell("A4"):setText(filter)
                end)
            end
        end
    }
    context.showDialog(dialog)
end
```

8.1.4.2 Метод setTitle

Задаёт заголовок диалогового окна.

```
setTitle(title : string)
```

8.1.4.3 Метод getTitle

Возвращает заголовок диалогового окна.

```
getTitle() : string
```

8.1.4.4 Метод setInitialDirectory

Устанавливает стартовый каталог при открытии диалога.

```
setInitialDirectory(path : string)
```

8.1.4.5 Метод getInitialDirectory

Возвращает стартовый каталог диалога.

```
getInitialDirectory() : string
```

8.1.4.6 Метод setFilter

Используется для задания типов файлов, доступных для выбора в диалоге.

```
setFilter(filter : string)
```

8.1.4.7 Метод getFilter

Возвращает текущий фильтр, использующийся для отображения списка файлов диалогового окна.

```
getFilter() : string
```

8.1.4.8 Метод setAllowMultiSelect

Метод включает или выключает возможность множественного выбора в списке файлов. Значение параметра по умолчанию — false.

```
setAllowMultiSelect(allow : bool)
```

8.1.4.9 Метод isMultiSelectAllowed

Возвращает текущее значение установки множественного выбора.

```
isMultiSelectAllowed() : bool
```

8.1.4.10 Метод setOnDone

Задаёт обработчик, который вызывается при выборе файлов и закрытии диалога. Возвращаются следующие параметры: `paths` – список путей выбранных файлов, `filter` – выбранный фильтр.

```
setOnDone(handler : function(paths : StringVector, filter: string))
```

8.1.5 Таблица ui.FileSaveDialog

Используется для реализации диалога сохранения файла (см. Рисунок 75).

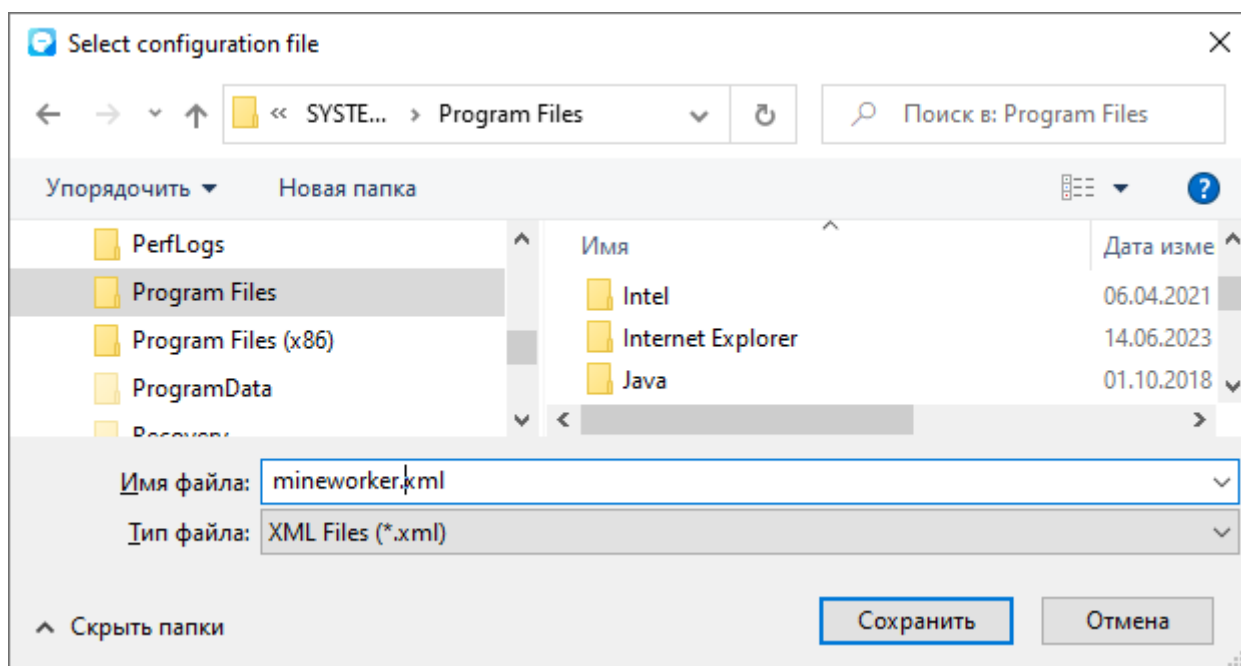


Рисунок 75 – Внешний вид виджета `ui:FileSaveDialog`

8.1.5.1 Конструктор ui:FileSaveDialog

Конструктор элемента `FileSaveDialog` позволяет установить свойства виджета в момент создания.

Пример:

```
local showFileSaveDialog = function(context)
    local dialog = ui:FileSaveDialog {
        Title = "Select configuration file",
        InitialDirectory = SomeDefaultDirectory,
        Filter = "Text Files (*.txt);;XML Files (*.xml)",
        OnDone = function(path, filter)
            if #path > 0 then
                context.doWithDocument(function(document)
                    document:getBlocks():getTable
```

```
("Sheet1"):getCell("A2"):setText(path)
    document:getBlocks():getTable
("Sheet1"):getCell("A4"):setText(filter)
    end)
    end
end
}
context.showDialog(dialog);
end
```

8.1.5.2 Метод setTitle

Задаёт заголовок диалогового окна.

```
setTitle(title : string)
```

8.1.5.3 Метод getTitle

Возвращает заголовок диалогового окна.

```
getTitle() : string
```

8.1.5.4 Метод setInitialDirectory

Устанавливает стартовый каталог при открытии диалогового окна.

```
setInitialDirectory(path : string)
```

8.1.5.5 Метод getInitialDirectory

Возвращает стартовый каталог диалогового окна.

```
getInitialDirectory() : string
```

8.1.5.6 Метод setFilter

Используется для задания типов файлов, доступных для выбора в диалоговом окне.

```
setFilter(filter : string)
```

8.1.5.7 Метод getFilter

Возвращает текущий фильтр, используемый для отображения списка файлов диалогового окна.

```
getFilter() : string
```

8.1.5.8 Метод setOnDone

Задаёт обработчик, который вызывается при выборе файла и закрытии диалогового окна. Возвращаются следующие параметры: path – путь к выбранному файлу, filter –

выбранный фильтр.

```
setOnDone(handler : function(path : string, filter: string))
```

8.1.6 Таблица `ui.Label`

Элемент **Метка** представляет собой не редактируемое текстовое поле на диалоге.

8.1.6.1 Конструктор `ui.Label`

Конструктор элемента **Надпись** позволяет установить свойства виджета в момент создания.

Пример:

```
local label = ui.Label {}
```

В момент создания элемента можно задать его свойства:

- `Name : string` – внутреннее имя (идентификатор) виджета **Надпись** в диалоговом окне;
- `Enabled : boolean` – статус виджета (активный/неактивный);
- `Size : Forms.Size` – размеры виджета (см. [Forms.Size](#));
- `Text : string` – отображаемый текст;
- `Alignment : Forms.Alignment` – тип выравнивания положения виджета.

Пример:

```
local label = ui.Label {  
    Name = "lblHello",  
    Text = "Hello, world!",  
    Size = Forms.Size(600, 300),  
    Alignment = Alignment_TopLeft  
}
```

8.1.6.2 Метод `Label:setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
local label = ui.Label {}  
label:setName("lblHello")
```

8.1.6.3 Метод `Label:getName`

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
local label = ui:Label {  
    Name = "lblHello",  
}  
local name = label:getName()
```

8.1.6.4 Метод `Label:setEnabled`

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
local label = ui:Label {}  
label:setEnabled(true)
```

8.1.6.5 Метод `Label:isEnabled`

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
local label = ui:Label {}  
local isEnabled = label:isEnabled()
```

8.1.6.6 Метод `Label:setSize`

Используется для установки значений размеров [Forms.Size](#) виджета.

Пример:

```
local label = ui:Label {}  
label:setSize(Forms.Size(50,50))
```

8.1.6.7 Метод `Label:getSize`

Используется для получения значений размеров [Forms.Size](#) виджета.

Пример:

```
local label = ui:Label {}  
local size = label:getSize()
```

8.1.6.8 Метод Label:setText

Используется для установки отображаемого текста виджета.

Пример:

```
local label = ui:Label {
    Text = "Hello, world!"
}
label:setText("New text.")
```

8.1.6.9 Метод Label:getText

Используется для получения отображаемого текста виджета.

Пример:

```
local label = ui:Label {
    Text = "Hello, world!"
}
local text = label:getText()
```

8.1.6.10 Метод Label:setAlignment

Используется для установки значения (тип - [Forms.Alignment](#)) выравнивания положения виджета.

Пример:

```
local label = ui:Label {}
label:setAlignment(Forms.Alignment_MiddleCenter)
```

8.1.6.11 Метод Label:getAlignment

Используется для получения значения (тип - [Forms.Alignment](#)) выравнивания положения виджета.

Пример:

```
local label = ui:Label {}
local alignment = label:getAlignment()
```

8.1.6.12 Метод Label:setColor

Используется для установки цвета [Forms.Color](#) текста виджета.

Пример:

```
local label = ui:Label {}
label:setColor(Forms.Color(255, 0, 0))
```


8.1.6.13 Метод `Label:getColor`

Используется для получения цвета [Forms.Color](#) текста виджета.

Пример:

```
local label = ui:Label {}  
local color = label:getColor()
```

8.1.7 Таблица `ui.Button`

Элемент реализует кнопку на диалоге.

8.1.7.1 Конструктор `ui.Button`

Конструктор элемента **Button** позволяет установить свойства виджета в момент создания.

Пример:

```
local button = ui.Button {  
    Name = "btnStart",  
    Title = "Начать обработку",  
    OnClick = function()  
        self.settingsId = 1  
    end  
}
```

Разработчик в момент создания элемента может задать его свойства:

- `Name : string` – внутреннее имя (идентификатор) виджета **Кнопка** в диалоговом окне;
- `Enabled : boolean` – статус виджета (активный/неактивный);
- `Size : Forms.Size` – размеры виджета;
- `Title : string` – отображаемый текст заголовка виджета;
- `OnClick : function()` – функция обработчик нажатия кнопки виджета.

8.1.7.2 Метод `Button:setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
local button = ui.Button {}  
button:setName("Start")
```

8.1.7.3 Метод Button:getName

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
local button = ui:Button {  
    Name = "Start",  
}  
local name = button:getName()
```

8.1.7.4 Метод Button:setEnabled

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
local button = ui:Button {}  
button:setEnabled(true)
```

8.1.7.5 Метод Button:isEnabled

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
local button = ui:Button {}  
local isEnabled = button:isEnabled()
```

8.1.7.6 Метод Button:setSize

Используется для установки значений размеров [Forms.Size](#) виджета.

Пример:

```
local button = ui:Button {}  
button:setSize(Forms.Size(50, 50))
```

8.1.7.7 Метод Button:getSize

Используется для получения значений размеров [Forms.Size](#) виджета.

Пример:

```
local button = ui:Button {}  
local size = button:getSize()
```

8.1.7.8 Метод `Button:setTitle`

Используется для установки отображаемого заголовка виджета.

Пример:

```
local button = ui:Button {}  
button:setTitle("Начать обработку")
```

8.1.7.9 Метод `Button:getTitle`

Используется для получения значения отображаемого заголовка виджета.

Пример:

```
local button = ui:Button {}  
local title = button:getTitle()
```

8.1.7.10 Метод `Button:setOnClick`

Используется для задания функции обработчика нажатия на кнопку виджета. Вызывается при нажатии пользователя на кнопку.

Пример:

```
local button = ui:Button {  
    Name = "btnStart",  
    Title = "Начать обработку"  
}  
button:setOnClick(function()  
    self.settingsId = 1  
end)
```

8.1.8 Таблица `ui.CheckBox`

Элемент реализует флаговую кнопку на диалоге.

8.1.8.1 Конструктор `ui.CheckBox`

Конструктор элемента **CheckBox** позволяет установить свойства виджета в момент создания.

Пример:

```
local checkBox = ui:CheckBox {  
    Name = "cbPrintBothSides",  
    Title = "Печать на обеих сторонах листа",  
    OnStateChanged = function(state)  
        self.printBothSides = state == Forms.CheckState_Checked
```

```
end  
}
```

Разработчик в момент создания элемента может задать его свойства:

- Name : string – внутреннее имя (идентификатор) виджета **Флажок** в диалоговом окне;
- Enabled : boolean – статус виджета (активный/неактивный);
- Size : [Forms.Size](#) – размеры ;
- Title : string – отображаемый текст заголовка виджета;
- State : [Forms.CheckState](#) – состояние флажка ;
- OnStateChanged : function([Forms.CheckState](#)) – функция обработчика изменения пользователем состояния (установлен/не установлен) флажка виджета.

8.1.8.2 Метод `CheckBox:setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
local checkBox = ui:CheckBox {}  
checkBox:setName("cbPrintBothSides")
```

8.1.8.3 Метод `CheckBox:getName`

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
local checkBox = ui:CheckBox {}  
local name = checkBox:getName()
```

8.1.8.4 Метод `CheckBox:setEnabled`

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
local checkBox = ui:CheckBox {}  
checkBox:setEnabled(true)
```

8.1.8.5 Метод `CheckBox:isEnabled`

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
local checkBox = ui:CheckBox {}  
local isEnabled = checkBox:isEnabled()
```

8.1.8.6 Метод `CheckBox:setSize`

Используется для установки значений размеров [Forms.Size](#) виджета.

Пример:

```
local checkBox = ui:CheckBox {}
checkBox:setSize(Forms.Size(30,60))
```

8.1.8.7 Метод `CheckBox:getSize`

Используется для получения значений размеров [Forms.Size](#) виджета.

Пример:

```
local checkBox = ui:CheckBox {}
local size = checkBox:getSize()
```

8.1.8.8 Метод `CheckBox:setState`

Используется для установки состояния флажка (установлен/не установлен) виджета. Устанавливаемый статус задается параметром, тип: [Forms.CheckState](#).

Пример:

```
local checkBox = ui:CheckBox {}
checkBox:setState(Forms.CheckState_Unchecked)
```

8.1.8.9 Метод `CheckBox:getState`

Используется для получения состояния флажка виджета (см. раздел [Forms.CheckState](#)).

Пример:

```
local checkBox = ui:CheckBox {}
local checkState = checkBox:getState()
```

8.1.8.10 Метод `CheckBox:isChecked`

Используется для проверки флажка виджета, возвращает значение `true`, если флажок установлен, и значение `false`, если не установлен.

Пример:

```
local checkBox = ui:CheckBox {}
local isChecked = checkBox:isChecked()
```

8.1.8.11 Метод `CheckBox:setOnStateChanged`

Используется для задания функции обработчика изменения флажка виджета, которое

возникает в случае установки/отмены установки флажка пользователем.

Пример:

```
local checkBox = ui:CheckBox {}
checkBox:setOnStateChanged(function(state)
    self.printBothSides = state == Forms.CheckState_Checked
end)
```

8.1.9 Таблица ui.RadioButton

Элемент реализует кнопку - переключатель на диалоге.

8.1.9.1 Конструктор ui.RadioButton

Конструктор элемента **RadioButton**, позволяет установить свойства виджета в момент создания.

Пример:

```
radioButton = ui:RadioButton {
    Title = "Use print dialog",
    OnStateChanged = function()
        self.printContext = "PrintFromPrintDialog"
        self.setActive(false, self.printSettings)
        self.setActive(false, self.printSettingsSetup)
    end
}
```

Разработчик в момент создания элемента может задать его свойства:

- Name : string – внутреннее имя (идентификатор) виджета **Радио кнопка** в диалоговом окне;
- Enabled : boolean – статус виджета (активный/неактивный);
- Size : [Forms.Size](#) – размеры виджета;
- Title : string – отображаемый текст заголовка виджета;
- State : [Forms.CheckState](#) – состояние флажка;
- OnStateChanged : function(Forms.CheckState) – функция обработчика нажатия кнопки виджета.

8.1.9.2 Метод `RadioButton:setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
radioButton = ui:RadioButton {}  
radioButton.setName("rbPrintDialog")
```

8.1.9.3 Метод `RadioButton:getName`

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
radioButton = ui:RadioButton {}  
local name = radioButton.getName()
```

8.1.9.4 Метод `RadioButton:setEnabled`

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
radioButton = ui:RadioButton {}  
radioButton.setEnabled(true)
```

8.1.9.5 Метод `RadioButton:isEnabled`

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
radioButton = ui:RadioButton {}  
local isEnabled = radioButton.isEnabled()
```

8.1.9.6 Метод `RadioButton:setState`

Используется для установки состояния флажка (установлен/не установлен) виджета.

Устанавливаемый статус задается параметром, тип: [Forms.CheckState](#).

Пример:

```
local radioButton = ui:RadioButton {}  
radioButton.setState(Forms.CheckState_Unchecked)
```

8.1.9.7 Метод `RadioButton:getState`

Используется для получения состояния флажка виджета

(см. раздел [Forms.CheckState](#)).

Пример:

```
local radioButton = ui:RadioButton {}
local checkState = radioButton:getState()
```

8.1.9.8 Метод `RadioButton:setOnStateChanged`

Используется для задания функции обработчика изменения флажка виджета, которое возникает в случае установки/отмены установки флажка пользователем.

Пример:

```
local radioButton = ui:RadioButton{}
radioButton:setOnStateChanged(function(state)
    self.printUserScale = state == Forms.CheckState_Checked
end)
```

8.1.9.9 Метод `RadioButton:getSize`

Используется для установки значений [Forms.Size](#) размеров виджета.

Пример:

```
local radioButton = ui:RadioButton {}
local size = radioButton:getSize()
```

8.1.9.10 Метод `RadioButton:setSize`

Используется для установки значений [Forms.Size](#) размеров виджета.

Пример:

```
local radioButton = ui:RadioButton {}
radioButton:setSize(Forms.Size(30,60))
```

8.1.10 Таблица `ui.GroupBox`

Элемент реализует рамку группы на диалоге.

8.1.10.1 Конструктор `ui.GroupBox`

Конструктор элемента **GroupBox**, позволяет установить свойства виджета в момент создания.

Пример:

```
groupBox = ui.GroupBox {
    ui:Column {
        self.printFromPrintDialogRadioButton,
        self.printFromPrintDocumentRadioButton
    }
}
```



```
}  
}
```

Разработчик в момент создания элемента может задать его свойства:

- `Name : string` – внутреннее имя (идентификатор) виджета **Рамка группы** в диалоговом окне;
- `Enabled : boolean` – статус виджета (активный/неактивный);
- `Size : Forms.Size` – размеры виджета.

Каждый элемент, входящий в виджет **Рамка группы**, должен быть представлен как элемент компоновки [ui:Row](#) или [ui:Column](#).

8.1.10.2 Метод `GroupBox:setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
groupBox = ui:GroupBox {}  
printMethodGroupGroupBox:setName("printMethodGroupBox")
```

8.1.10.3 Метод `GroupBox:getName`

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
groupBox = ui:GroupBox {}  
local name = groupBox:getName()
```

8.1.10.4 Метод `GroupBox:setEnabled`

Используется для установки статуса виджета (активный/неактивный).

Пример:

```
groupBox = ui:GroupBox {}  
groupBox:setEnabled(true)
```

8.1.10.5 Метод `GroupBox:isEnabled`

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
groupBox = ui:GroupBox {}  
local isEnabled = groupBox:isEnabled()
```

8.1.10.6 Метод `GroupBox:getSize`

Используется для получения значений [Forms.Size](#) размеров виджета.

Пример:

```
local groupBox = ui:GroupBox {}
local size = groupBox:getSize()
```

8.1.10.7 Метод `GroupBox:setSize`

Используется для установки значений [Forms.Size](#) размеров виджета.

Пример:

```
local groupBox = ui:GroupBox {}
groupBox:setSize(Forms.Size(30,60))
```

8.1.11 Таблица `ui.ListBox`

Элемент реализует список на диалоге.

8.1.11.1 Конструктор `ui.ListBox`

Конструктор элемента **Listbox** позволяет установить свойства виджета в момент создания.

Пример:

```
local Actions = {}

Actions.listItems = ui:ListItems {
  {
    text = "Яблоко",
    id = 0,
    checkState = Forms.CheckState_Checked
  },
  {
    text = "Груша",
    id = 1,
    checkState = Forms.CheckState_Unchecked
  },
}

Actions.listBox = ui.ListBox {
  Name = "lbFruits",
  Items = Actions.listItems,
  OnCurrentItemChanged = function(itemId)
```

```
        -- TODO
    end,
    OnItemStateChanged = function(itemId, itemState)
        -- TODO
    end
}
return Actions
```

Разработчик в момент создания элемента может задать его свойства:

- Name : string – внутреннее имя (идентификатор) виджета **Список элементов** в диалоговом окне;
- Enabled : boolean – статус виджета (активный/неактивный);
- Size : [Forms.Size](#) – размеры виджета;
- Items : [Forms.ListItems](#) – элементы списка;
- CurrentItem : [Forms.ItemID](#) – текущий элемент;
- OnCurrentItemChanged : function([Forms.ItemID](#)) – функция, выполняемая при изменении пользователем текущего элемента списка элементов;
- OnItemStateChanged : function([Forms.ItemID](#), [Forms.CheckState](#)) – функция обработчика состояния флажка элемента списка.

8.1.11.2 Метод `ListBox:setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
listBox = ui:ListBox {}
listBox:setName("lbFruits")
```

8.1.11.3 Метод `ListBox:getName`

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
listBox = ui:ListBox {}
local name = listBox:getName()
```

8.1.11.4 Метод `ListBox:setEnabled`

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
listBox = ui:ListBox {}  
listBox:setEnabled(true)
```

8.1.11.5 Метод `ListBox:isEnabled`

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
listBox = ui:ListBox {}  
local state = listBox.isEnabled()
```

8.1.11.6 Метод `ListBox:setSize`

Используется для установки значений размеров виджета.

Пример:

```
listBox = ui:ListBox {}  
listBox:setSize(Forms.Size(50, 50))
```

8.1.11.7 Метод `ListBox:getSize`

Используется для получения значений размеров виджета.

Пример:

```
listBox = ui:ListBox {}  
local size = listBox.getSize()
```

8.1.11.8 Метод `ListBox:setItems`

Используется для замещения всех элементов списка виджета новыми элементами.

Пример:

```
listItems = ui:ListItems {  
  {  
    text = "Яблоко",  
    id = 0,  
    checkState = Forms.CheckState_Checked  
  },  
  {  
    text = "Груша",  
    id = 1,  
  }  
}
```

```
        checkState = Forms.CheckState_Unchecked
    },
}
listBox = ui:ListBox {}
listBox.setItems(listItems)
```

8.1.11.9 Метод `ListBox:getItems`

Используется для получения списка (тип [Forms.ConstListItems](#)) всех элементов виджета.

Пример:

```
listBox = ui:ListBox {}
local items = listBox.getItems()
```

8.1.11.10 Метод `ListBox:addItem`

Добавляет новый элемент с указанным отображаемым текстом (`text: string`), идентификатором и состоянием в конец списка. Идентификатор [Forms.ItemID](#), наличие флажка [Forms.CheckState](#) – необязательные аргументы. Если наличие флажка не было указано, то добавляется элемент без флажка.

Пример:

```
local text = "Яблоко"
local id = 0

listBox = ui:ListBox {}
listBox.addItem(text, id)
```

8.1.11.11 Метод `ListBox:removeItem`

Удаляет элемент с указанным идентификатором. Для недействительного идентификатора не осуществляется никаких действий.

Пример:

```
listBox.removeItem(10)
```

8.1.11.12 Метод `ListBox:removeRow`

Удаляет элемент по индексу строки. Выдает ошибку для недействительного индекса строки.

Пример:

```
listBox.addItem("Яблоко", 0) -- id элемента = 0, индекс строки = 1
listBox.addItem("Груша", 1) -- id элемента = 1, индекс строки = 2
listBox.addItem("Слива", 2) -- id элемента = 2, индекс строки = 3

listBox.removeRow(2) -- Удаляет строку "Груша"
```

8.1.11.13 Метод `ListBox:setCurrentItem`

Изменяет текущий выбранный элемент на элемент с указанным идентификатором. Сбрасывает выделение для недействительного идентификатора элемента.

Пример:

```
listBox:setCurrentItem(1)
```

8.1.11.14 Метод `ListBox:getCurrentItem`

Возвращает идентификатор элемента (см. раздел [Forms.ItemID](#)) для текущего элемента.

Пример:

```
listBox = ui:ListBox {}
local currentItem = listBox:getCurrentItem()
```

8.1.11.15 Метод `ListBox:setOnCurrentItemChanged`

Используется для задания функции обработчика изменений текущего элемента. Вызывается, когда новый элемент был выбран пользователем.

Пример:

```
listBox = ui:ListBox {
  Name = "lbFruits",
  Items = listItems
}
listBox:setOnCurrentItemChanged(function(itemId)
  -- TODO
end)
```

8.1.11.16 Метод `ListBox:setItemCheckState`

Используется для изменения состояния флажка элемента с указанным идентификатором и включения флажка, если элемент изначально был создан как элемент без флажка.

Пример:

```
listBox = ui:ListBox {}  
listBox:addItem("Яблоко", 0, Forms.CheckState_Unchecked)  
listBox:addItem("Груша", 1, Forms.CheckState_Checked)  
  
listBox:setItemCheckState(1, Forms.CheckState_Unchecked)
```

8.1.11.17 Метод `ListBox:setOnItemStateChanged`

Используется для задания функции обработчика изменения флажка элемента. Вызывается при изменении флажка пользователем.

Пример:

```
listBox = ui:ListBox {  
  Name = "lbFruits",  
  Items = listItems  
}  
listBox:setOnItemStateChanged(function(itemId, itemState)  
  -- TODO  
end)
```

8.1.12 Таблица `ui.ListItems`

Используется для создания составных элементов виджетов [GroupBox](#) и [ComboBox](#).

8.1.12.1 Метод `addItem`

Добавляет элемент списка с заданным текстом, идентификатором и состоянием в конец списка.

Варианты реализации:

```
addItem(listItem : Forms.ListItem)  
addItem(text : string)  
addItem(text : string, id : Forms.ItemID)  
addItem(text : string, id : Forms.ItemID, checkState : Forms.Check\_State)
```

Пример:

```
local listItems = ui:ListItems{}  
  
local firstItem = Forms.ListItem("First item", 0, Forms.CheckState_Unchecked)  
listItems:addItem(firstItem)  
listItems:addItem("Second item")  
listItems:addItem("Third item", 2)
```

```
listItems.addItem("Fourth item", 3, Forms.CheckState_Unchecked)

local listBox = ui:ListBox {
    Name = "ListBox",
    Items = listItems
}
```

8.1.13 Таблица ui.ComboBox

Элемент реализует элемент выпадающий список.

8.1.13.1 Конструктор ui.ComboBox

Конструктор элемента **ComboBox** позволяет установить свойства виджета в момент создания.

Пример:

```
local Actions = {}

Actions.listItems = ui:ListItems {
    {
        text = "Яблоко",
        id = 0,
        checkState = Forms.CheckState_Unchecked
    },
    {
        text = "Груша",
        id = 1,
        checkState = Forms.CheckState_Unchecked
    },
    {
        text = "Слива",
        id = 2,
        checkState = Forms.CheckState_Unchecked
    }
}

Actions.comboBox = ui:ComboBox {
    Name = "ComboFruits",
    Items = Actions.listItems,
    OnCurrentItemChanged = function(id)
        msg = "Выбран элемент " .. id
        EditorAPI.messageBox(msg)
    end
}
```



```
end
}

Actions.dlg = ui:Dialog {
  Size = Forms.Size(600,300),
  Title = "Демонстрация",
  ui:Column {
    ui:Row {Actions.comboBox}
  }
}

function Actions.ShowControls(context)
  context.showDialog(Actions.dlg)
end

return Actions
```

Разработчик в момент создания элемента может задать его свойства:

- Name : string – внутреннее имя (идентификатор) виджета **Поле с выпадающим списком** в диалоговом окне;
- Enabled : boolean – статус виджета (активный/неактивный);
- Size : [Forms.Size](#) – размеры виджета;
- Items : [Forms.ListItems](#) – элементы списка;
- CurrentItem : [Forms.ItemID](#) – текущий элемент;
- OnCurrentItemChanged : function(Forms.ItemID) – функция, выполняемая при изменении текущего элемента в выпадающем списке элементов виджета.

8.1.13.2 Метод `ComboBox:setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
comboBox = ui:ComboBox {}
comboBox:setName("ComboFruits")
```

8.1.13.3 Метод `ComboBox:getName`

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
comboBox = ui:ComboBox {}  
local name = comboBox.getName()
```

8.1.13.4 Метод `ComboBox:setEnabled`

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
comboBox = ui:ComboBox {}  
comboBox.setEnabled(true)
```

8.1.13.5 Метод `ComboBox:isEnabled`

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
comboBox = ui:ComboBox {}  
local state = comboBox.isEnabled()
```

8.1.13.6 Метод `ComboBox:setSize`

Используется для установки значений размеров [Forms.Size](#) виджета.

Пример:

```
comboBox = ui:ComboBox {}  
comboBox.setSize(Forms.Size(50, 30))
```

8.1.13.7 Метод `ComboBox:getSize`

Используется для получения значений размеров [Forms.Size](#) виджета.

Пример:

```
comboBox = ui:ComboBox {}  
local sz = comboBox.getSize()
```

8.1.13.8 Метод `ComboBox:setItems`

Используется для замещения всех элементов списка виджета новыми элементами.

Пример:

```
listItems = ui:ListItems {  
  {  
    text = "Яблоко",  
    id = 0,  
  }  
}
```

```
        checkState = Forms.CheckState_Unchecked
    },
    {
        text = "Груша",
        id = 1,
        checkState = Forms.CheckState_Unchecked
    },
    {
        text = "Слива",
        id = 2,
        checkState = Forms.CheckState_Unchecked
    }
}
comboBox = ui:ComboBox {}
comboBox.setItems(listItems)
```

8.1.13.9 Метод `ComboBox:getItems`

Используется для получения списка (тип [Forms.ConstListItems](#)) всех элементов виджета.

Пример:

```
comboBox = ui:ComboBox {}
local listItems = comboBox.getItems()
```

8.1.13.10 Метод `ComboBox:addItem`

Добавляет новый элемент с указанным отображаемым текстом (`text: string`), идентификатором (`id: Forms.ItemID`) и наличием флажка (`state: Forms.CheckState`) в конец списка. Идентификатор, состояние – необязательные аргументы. Если наличие флажка не было указано, то добавляется элемент без флажка.

Пример:

```
local text = "Яблоко"
local id = 0

comboBox = ui:ComboBox {}
comboBox.addItem(text, id, Forms.CheckState_Checked)
```

8.1.13.11 Метод `ComboBox:removeItem`

Удаляет элемент с указанным идентификатором. Для недействительного идентификатора не осуществляется никаких действий.

Пример:

```
local text = "Яблоко"
local id = 0

comboBox = ui:ComboBox {}
comboBox.addItem(text, id, Forms.CheckState_Checked)
comboBox.removeItem(0)
```

8.1.13.12 Метод `ComboBox:removeRow`

Удаляет элемент по индексу строки. Выдает ошибку для недействительного индекса строки.

Пример:

```
comboBox = ui:ComboBox {}

comboBox.addItem("Яблоко", 0) -- id элемента = 0, индекс строки = 1
comboBox.addItem("Груша", 1) -- id элемента = 1, индекс строки = 2
comboBox.addItem("Слива", 2) -- id элемента = 2, индекс строки = 3

comboBox.removeRow(2) -- Удаляет строку «Груша»
```

8.1.13.13 Метод `ComboBox:setCurrentItem`

Изменяет текущий выбранный элемент на элемент с указанным идентификатором. Сбрасывает выделение для недействительного идентификатора элемента.

Пример:

```
comboBox = ui:ComboBox {}

comboBox.addItem("Яблоко", 0) -- id элемента = 0, индекс строки = 1
comboBox.addItem("Груша", 1) -- id элемента = 1, индекс строки = 2
comboBox.addItem("Слива", 2) -- id элемента = 2, индекс строки = 3

comboBox.removeRow(2) -- Удаляет строку «Груша»
comboBox.setCurrentItem(2)
```

8.1.13.14 Метод `ComboBox:getCurrentItem`

Метод возвращает идентификатор (см. раздел [Forms.ItemID](#)) текущего выбранного элемента.

Пример:

```
comboBox = ui:ComboBox {}

comboBox.addItem("Яблоко", 0) -- id элемента = 0, индекс строки = 1
comboBox.addItem("Груша", 1) -- id элемента = 1, индекс строки = 2
comboBox.addItem("Слива", 2) -- id элемента = 2, индекс строки = 3

comboBox.setCurrentItem(2)
comboBox.getCurrentItem() -- 2
```

8.1.13.15 Метод `ComboBox:setOnCurrentItemChanged`

Используется для задания функции обработчика изменений текущего элемента. Вызывается, когда новый элемент был изменен пользователем.

Пример:

```
comboBox = ui:ComboBox {
    Name = "ComboFruits"
}
comboBox.setOnCurrentItemChanged(function(id)
    msg = "Выбран элемент " .. id
    EditorAPI.messageBox(msg)
end)
```

8.1.14 Таблица `ui.TextBox`

Элемент реализует область для ввода текста.

8.1.14.1 Конструктор `ui:TextBox`

Конструктор элемента `TextBox` позволяет установить свойства виджета в момент создания.

Пример:

```
local textBox = ui:TextBox {
    Name = "txtEntryField",
    Text = "Type text here"
}
textBox.setOnEditingFinished(function()
    EditorAPI.messageBox(textBox.getText())
end)
```

Разработчик в момент создания элемента может задать его свойства:

- Name : string – внутреннее имя (идентификатор) виджета **Текстовое поле** в диалоговом окне;
- Enabled : boolean – статус виджета (активный/неактивный);
- Size : [Forms.Size](#) – размеры виджета;
- Text : string – отображаемый текст;
- OnTextChanged : function(text) – функция, вызываемая при изменении текста;
- OnEditingFinished : function() – функция, вызываемая при завершении ввода текста.

8.1.14.2 Метод `TextBox.setName`

Используется для установки значения внутреннего имени (идентификатора) виджета.

Пример:

```
textBox = ui:TextBox {}  
textBox.setName("EntryField")
```

8.1.14.3 Метод `TextBox.getName`

Используется для получения значения внутреннего имени (идентификатора) виджета.

Пример:

```
textBox = ui:TextBox {}  
local name = textBox.getName()
```

8.1.14.4 Метод `TextBox.setEnabled`

Используется для изменения статуса виджета (активный/неактивный).

Пример:

```
textBox = ui:TextBox {}  
textBox.setEnabled(true)
```

8.1.14.5 Метод `TextBox.isEnabled`

Используется для проверки статуса виджета (активный/неактивный).

Пример:

```
textBox = ui:TextBox {}  
textBox.isEnabled()
```

8.1.14.6 Метод `TextBox:setSize`

Используется для установки значений размеров `Forms.Size` виджета.

Пример:

```
textBox = ui:TextBox {}  
textBox:setSize(Forms.Size(30, 30))
```

8.1.14.7 Метод `TextBox:getSize`

Используется для получения значений размеров виджета.

Пример:

```
textBox = ui:TextBox {}  
local size = textBox:getSize()
```

8.1.14.8 Метод `TextBox:setText`

Используется для установки текста виджета.

Пример:

```
textBox = ui:TextBox {}  
textBox:setText("Type text here")
```

8.1.14.9 Метод `TextBox:getText`

Используется для получения текста виджета.

Пример:

```
textBox = ui:TextBox {}  
local size = textBox:getText()
```

8.1.14.10 Метод `TextBox:setOnTextChanged`

Используется для задания функции обработчика изменения текста виджета (тип: `function`), которое возникает в случае редактирования текста пользователем.

Пример:

```
textBox:setOnTextChanged(function(text)  
    EditorAPI.messageBox(textBox:getText())  
end)
```

8.1.14.11 Метод `TextBox:setOnEditingFinished`

Используется для задания функции обработчика завершения изменения текста виджета, которое возникает в случае нажатия клавиш **Enter** или **Return** или потери фокуса

виджетом.

Пример:

```
textBox:setOnEditingFinished(function()  
    EditorAPI.messageBox(textBox.getText())  
end)
```

8.1.15 Таблица `ui.Row`

Реализует контейнер для горизонтального выравнивания элементов в диалоговом окне.

8.1.15.1 Конструктор `ui:Row`

Конструктор элемента **Row** позволяет установить свойства виджета в момент создания.

Пример:

```
dlg = ui:Dialog {  
    Size = Forms.Size(600, 300),  
    ui:Column {  
        ui:Row {lblHello},  
        ui:Row {txtEntryField}  
    }  
}
```

8.1.16 Таблица `ui.Column`

Элемент обеспечивает вертикальное выравнивание для своих дочерних элементов в диалоговом окне.

8.1.16.1 Конструктор `ui:Column`

Конструктор элемента **Column** позволяет установить свойства виджета в момент создания.

Пример:

```
dlg = ui:Dialog {  
    Size = Forms.Size(600, 300),  
    ui:Column {  
        ui:Row {lblHello},  
        ui:Row {txtEntryField}  
    }  
}
```


8.1.17 Таблица `ui.Spacer`

Используется для реализации выравнивания позиции виджета относительно ширины или высоты диалогового окна и относительно других виджетов.

8.1.17.1 Конструктор `ui.Spacer`

Конструктор таблицы **Spacer** пустого пространства используется для выравнивания позиции виджета относительно ширины или высоты диалогового окна и относительно других виджетов.

Пример:

```
dlg = ui:Dialog {
    Size = Forms.Size(600,300),
    -- Центрировать listBox относительно границ окна
    ui:Column {
        ui:Spacer {},
        ui:Row {listBox},
        ui:Spacer {}
    }
}
```

8.2 Таблица `Forms`

Таблица предоставляет доступ к общим структурам данных, используемым для установки свойств виджетов.

8.2.1 Тип `Forms.ItemID`

Идентификатор элемента виджета, тип - числовой. Используется в [ListBox:addItem](#), [ListBox:getCurrentItem](#), [ListItems:addItem](#), в качестве идентификатора элемента списка.

8.2.2 Тип `Forms.CheckState`

Содержит состояние выбора элемента [ui:CheckBox](#).

Если элемент выбран, то поле `State` содержит значение `Forms.CheckState_Checked`, в противном случае - `Forms.CheckState_Unchecked`.

8.2.3 Тип `Forms.Alignment`

Поле содержит тип выравнивания элемента виджета по горизонтали и вертикали. Используется для выравнивания метки [ui:Label](#). Типы выравнивания элемента виджета

представлены в таблице 95.

Таблица 95 – Типы выравнивания элементов виджета Forms.Alignment

Значение	Описание
Alignment_TopLeft	Выравнивание по верхнему левому углу
Alignment_TopCenter	Выравнивание сверху по центру
Alignment_TopRight	Выравнивание по верхнему правому углу
Alignment_MiddleLeft	Выравнивание по левой средней части
Alignment_MiddleCenter	Выравнивание по центру средней части
Alignment_MiddleRight	Выравнивание по правой средней части
Alignment_BottomLeft	Выравнивание по нижнему левому углу
Alignment_BottomCenter	Выравнивание снизу по центру
Alignment_BottomRight	Выравнивание по нижнему правому углу

8.2.4 Тип Forms.DialogButton

Поле содержит константы диалоговых кнопок виджетов. Используется при добавлении кнопок на форму (см. метод [ui:DialogButtons:addButton](#)). В зависимости от выбранного типа для кнопки задается локализованный заголовок кнопки, а также код ответа (тип [Forms.DialogCode](#)). Возможные виды констант представлены в таблице 96.

Таблица 96 – Константы диалоговых кнопок Forms.DialogButton

Константы	Заголовок (Ru)	Заголовок (En)	Код ответа
Forms.DialogButton_OK	ОК	OK	Forms.DialogCode_Accepted
Forms.DialogButton_Done	Готово	Done	
Forms.DialogButton_Yes	Да	Yes	
Forms.DialogButton_Retry	Повторить	Retry	
Forms.DialogButton_Ignore	Пропустить	Ignore	Forms.DialogCode_Rejected
Forms.DialogButton_Cancel	Отмена	Cancel	
Forms.DialogButton_No	Нет	No	
Forms.DialogButton_Close	Закрыть	Close	

Пример:

```
local dialogButtons = ui:DialogButtons{}
dialogButtons:addButton(Forms.DialogButton_OK)
dialogButtons:addButton(Forms.DialogButton_Cancel)

dialog = ui:Dialog{
  Buttons = dialogButtons
}
dialog:setOnDone(function(ret)
  if ret == Forms.DialogCode_Accepted then
    -- OK pressed
  elseif ret == Forms.DialogCode_Rejected then
    -- Cancel pressed
  end
end)
```

```
end)  
context.showDialog(dialog)
```

8.2.5 Тип Forms.DialogCode

Содержит код, возвращаемый после закрытия диалогового окна пользователем.

Если пользователь нажал кнопку **Отмена**, то поле содержит значение Forms.DialogCode_Rejected, если пользователь нажал кнопку **ОК** то поле содержит значение Forms.DialogCode_Accepted.

Пример:

```
dialog.setOnDone(function(ret)  
    if ret == Forms.DialogCode_Accepted then  
        -- OK pressed  
    elseif ret == Forms.DialogCode_Rejected then  
        -- Cancel pressed  
    end  
end)
```

8.2.6 Тип Forms.DialogButtonRole

Поле содержит константы ролей диалоговых кнопок виджетов. Используется в [ui:DialogButtons:addButton\(\)](#). Возможные значения констант представлены в таблице 97.

Таблица 97 – Константы ролей диалоговых кнопок элементов виджетов Forms.DialogButtonRole

Константы	Описание роли
Forms.DialogButtonRole_Accept	Подтверждение при нажатии на кнопку в диалоговом окне, например, кнопка ОК
Forms.DialogButtonRole_Reject	Отмена при нажатии на кнопку в диалоговом окне, например, кнопка Cancel

8.2.7 Таблица Forms.Size

Используется для указания величин ширины и высоты виджета. Описание полей таблицы представлены в таблице 98.

Таблица 98 – Описание полей таблицы Forms.Size

Поле	Тип	Описание
Size.width	number	Ширина виджета
Size.height	number	Высота виджета

8.2.8 Таблица Forms.ListItem

Элемент списка виджетов. Описание полей таблицы представлены в таблице 99.

Варианты инициализации:

```
Forms.ListItem()  
Forms.ListItem(itemId: Forms.ItemID)  
Forms.ListItem(text: string)  
Forms.ListItem(text: string, itemId: Forms.ItemID)  
Forms.ListItem(text: string, itemId: Forms.ItemID, checkState: Forms.CheckState)
```

Таблица 99 – Описание полей таблицы Forms.ListItem

Поле	Тип	Описание
ListItem.text	Строка	Наименование элемента
ListItem.id	Forms.ItemID	Идентификатор элемента
ListItem.checkState	Forms.CheckState или nil	Состояние (checked / unchecked)

Пример:

```
local listItems = ui.ListItems{  
  {  
    text = "First option",  
    id = 0,  
    checkState = Forms.CheckState_Unchecked  
  }  
}  
  
local secondItem = Forms.ListItem()  
secondItem.id = 1  
secondItem.text = "Second option"  
secondItem.checkState = Forms.CheckState_Checked  
  
local thirdItem = Forms.ListItem(2)  
thirdItem.text = "Third option"  
thirdItem.checkState = Forms.CheckState_Unchecked  
  
local fourthItem = Forms.ListItem("Fourth option", 3)  
fourthItem.checkState = Forms.CheckState_Unchecked  
  
local fifthItem = Forms.ListItem("Fifth option", 4, Forms.CheckState_Unchecked)  
  
listItems.addItem(secondItem)
```

```
listItems.addItem(thirdItem)
listItems.addItem(fourthItem)
listItems.addItem(fifthItem)
```

8.2.9 Таблица Forms.ConstListItems

Используется при получении списка элементов виджетов при вызовах [ListBox:getItems\(\)](#) и [ComboBox:getItems\(\)](#).

8.2.9.1 Метод getCount

Возвращает количество элементов таблицы.

```
getCount() : integer
```

8.2.9.2 Метод getItem

Возвращает элемент списка [Forms.ListItem](#) по индексу.

```
getItem(index: number) : Forms.ListItem
```

8.2.10 Таблица Forms.Color

Предназначена для настройки цвета отображения элементов виджетов. Используется в [Label:setColor\(\)](#). Описание полей таблицы представлены в таблице 100.

Таблица 100 – Описание полей таблицы Forms.Color

Поле	Тип	Описание
Color.red	number	Значение для установки интенсивности красного цвета
Color.green	number	Значение для установки интенсивности зеленого цвета
Color.blue	number	Значение для установки интенсивности синего цвета

9 Использование внешнего модуля для работы со строками UTF-8

Стандартная таблица `string` не позволяет работать со строками в формате UTF-8. В качестве решения можно использовать одну из библиотек репозитория **LuaRocks**. Ниже приведен пример установки и использования библиотеки [luautf8](#), которая реализует все аналогичные методы таблицы `string` для работы со строками в формате UTF-8.

Для загрузки и сборки библиотеки `luautf8` посредством **LuaRocks** необходимо выполнить команду `luarocks install luautf8`. Ниже приведен пример для OS Windows:

```
D:\Work\LuaRocks>luarocks install luautf8
Installing https://luarocks.org/luautf8-0.1.5-2.src.rock

luautf8 0.1.5-2 depends on lua >= 5.1 (5.3-1 provided by VM)
cl /nologo /MD /O2 -c -Folutf8lib.obj -Ic:\Program Files\MyOffice\Lua532\include
lutf8lib.c lutf8lib.c
link -dll -def:lua-utf8.def -out:lua-utf8.dll c:\Program
Files\MyOffice\Lua532\lib\liblua.dll.a lutf8lib.obj
Microsoft (R) Incremental Linker Version 14.31.31104.0

Copyright (C) Microsoft Corporation. All rights reserved.
Создается библиотека lua-utf8.lib и объект lua-utf8.exp luautf8 0.1.5-2 is now
installed in D:\Work\LuaRocks\5.3 (license: MIT)
```

Далее необходимо скопировать полученную библиотеку **lua-utf8.dll** в папку **bin** надстройки, а затем использовать вызовы библиотеки в теле надстройки:

```
local utf_8 = require 'lua-utf8'
.....
text = "Здравствуй, мир"
position = utf_8.find(text, "мир") -- 13
```

10 Функции для работы со строками в формате Юникод (UTF-8)

Для работы со строками, содержащими русские символы, можно использовать методы таблицы `utf8`. Предполагается, что аргументы методов являются допустимыми строками UTF-8.

10.1 Функция `utf8.char`

Функция `utf8.char` возвращает строку в формате UTF-8, соответствующую коду символа.

Вызов:

```
utf8.char(code)
```

Параметры:

– `code`: код символа UTF-8, тип `number`.

Возвращает:

– `string`: символ UTF-8, полученный по коду.

Пример:

```
print(utf8.char(244)) -- ô
```

10.2 Функция `utf8.codes`

Функция `utf8.codes` возвращает последовательность кодов символов, из которых состоит строка UTF-8.

Вызов:

```
utf8.codes(str)
```

Параметры:

– `str`: строка в формате UTF-8

Возвращает:

– итератор, с помощью которого можно получить коды символов исходной строки.

Пример:

```
str = "МойОфис"  
for p, c in utf8.codes(str) do  
  print(c)  
end  
-- 1052, 1086, 1081, 1054, 1092, 1080, 1089
```


10.3 Функция `utf8.codepoint`

Функция `utf8.codepoint` возвращает код заданного символа.

Вызов:

```
utf8.codepoint(char)
```

Параметры:

– `char`: символ UTF-8, тип `utf-char`.

Возвращает:

– `number`: код символа UTF-8.

Примеры:

```
print(utf8.codepoint(" ")) -- 29790
print(utf8.codepoint("A")) -- 1040
```

10.4 Функция `utf8.charpattern`

Функция `utf8.charpattern` возвращает шаблон `"[\0-\x7F\xC2-\xF4][\x80-\xBF]*"` для определения последовательности символов формата UTF-8.

Пример:

```
function len(s)
  local n = 0
  for match in s:gmatch(utf8.charpattern) do
    n = n + 1
  end
  return n
end

str = "МойОфис"
print(len(str))
```

10.5 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
```

10.6 Функция utf8.lower

Функция `utf8.lower` возвращает строку в формате UTF-8, полученную из исходной строки путем преобразования в нижний регистр.

Вызов:

```
utf8.lower(str)
```

Параметры:

– str: строка в формате UTF-8

Возвращает:

– string: строка в нижнем регистре.

Пример:

```
print(utf8.lower("A")) -- a
print(utf8.lower("Abc")) -- abc
```

10.7 Функция utf8.substr

Функция `utf8.substr` возвращает подстроку в формате UTF-8, начиная с индекса `first` и заканчивая индексом `last`.

```
utf8.substr(str, first[, last])
```

Параметры:

– str: string – исходная строка в формате UTF-8;

– first: number – позиция первого символа подстроки;

– last: number – позиция последнего символа подстроки (по умолчанию равна позиции последнего символа в строке).

Возвращает:

– string: подстрока в формате UTF-8

- если позиция первого или последнего символа находится вне строки, то диапазон усекается до корректного;
- если диапазон задан некорректно, то возвращается пустая строка.

Пример:

```
print(utf8.substr("регистр", 5))    -- стр
print(utf8.substr("регистр", 0, 2)) -- ре
print(utf8.substr("регистр", 2, 1)) --
print(utf8.substr("регистр", 2, 100)) -- егистр
```

10.8 Функция utf8.compare

Функция `utf8.compare` возвращает результат сравнения двух строк согласно [алгоритму сортировки по Юникоду](#).

Вызов:

```
utf8.compare(str1, str2, opt)
```

Параметры:

- `str1` – первая строка (`string`) в формате UTF-8;
- `str2` – вторая строка (`string`) в формате UTF-8;
- `opt` – параметр (`number`) учета регистра при сравнении:
 - 0 – без учета регистра;
 - 1 – с учетом регистра.

Возвращает:

- `number`: результат сравнения аргументов:
 - -1 – если `str1 < str2`;
 - 0 – если `str1 = str2`;
 - 1 – если `str1 > str2`.

Пример:

```
print(utf8.compare("A", "a", 0)) -- 0, arg1 = arg2 с учетом регистра
print(utf8.compare("A", "a", 1)) -- -1, arg1 < arg2 без учета регистра
```

10.9 Функция utf8.islower

Функция `utf8.islower` проверяет, находится ли в нижнем регистре переданный символ или строка.

```
utf8.islower(str)
```

Параметр:

– str: строка, символ или число, представляющее код UTF-8.

Возвращает:

– boolean: true, если передан код символа в нижнем регистре.

Пример:

```
print(utf8.islower("a")) -- false  
print(utf8.islower("A")) -- true
```

10.10 Функция utf8.isupper

Функция `utf8.isupper` проверяет, находится ли в верхнем регистре переданный символ или строка.

```
utf8.isupper(str)
```

Параметр:

– str: строка, символ или число, представляющее код UTF-8.

Возвращает:

– boolean: true, если передан код символа в верхнем регистре.

Пример:

```
print(utf8.islower("a")) -- false  
print(utf8.islower("A")) -- true
```

10.11 Функция utf8.isdigit

Функция `utf8.isdigit` проверяет, является ли цифровым символом переданный символ или число.

```
utf8.isdigit(char)
```

Параметр:

– char: UTF-8-character – символ в кодировке UTF-8.

Возвращает:

– boolean: значение true, если передан код цифрового символа.

Пример:

```
print(utf8.isdigit("a")) -- false
print(utf8.isdigit("1")) -- true
```

10.12 Функция utf8.isalpha

Функция `utf8.isalpha` проверяет, является ли буквенным символом переданный СИМВОЛ или ЧИСЛО.

```
utf8.isalpha(char)
```

Параметр:

– `char`: символ в кодировке UTF-8.

Возвращает:

– `boolean`: `true`, если передан код буквенного символа.

Пример:

```
print(utf8.isalpha('A')) -- true
print(utf8.isalpha('1')) -- false
```

10.13 Функция utf8.len

Функция `utf8.len` позволяет определить длину заданной строки в символах.

```
utf8.len(str)
```

Параметр:

– `str`: `string` – строка в формате UTF-8.

Возвращает:

– `number`: длина заданной строки в символах.

Пример:

```
print(utf8.len("МойОфис")) -- 7
```

10.14 Функция utf8.offset

Функция `utf8.offset` возвращает позицию (в байтах), с которой начинается кодирование символа с заданной позицией.

```
utf8.offset(str, charPos)
```

Параметр:

- `str`: `string` – строка в формате UTF-8;
- `charPos`: `number` – позиция символа.

Возвращает:

- `number`: позиция (в байтах), с которой начинается кодирование символа с заданной позицией.

Пример:

```
print(utf8.offset("АБВГДЕЖЗ", 5)) -- 9
```

10.15 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)
```

11 Справочник методов расширения таблицы Regex

11.1 Метод create

Компилирует регулярное выражение и возвращает его в виде объекта. По умолчанию используется Perl - совместимый формат регулярных выражений.

```
Re.create(pattern)
```

Параметр:

– pattern:string – строка шаблона.

Возвращает:

- regex:object - объект Regex, который содержит скомпилированное регулярное выражение для дальнейшего использования;
- err:string - сообщение об ошибке или nil.

11.2 Метод match

Сопоставляет скомпилированное регулярное выражение с заданной исходной строкой. Возвращает найденные подстроки.

```
Re.match(subject, matchFlags, pattern)
```

Параметры:

- subject:string – исходная строка;
- matchFlags:int – флаги, задающие правила применения регулярного выражения;
- pattern:string, Regex – строка шаблона или скомпилированный шаблон.

Возвращает:

- matches:object – подстроки, найденные в соответствии с шаблоном;
- err:string - сообщение об ошибке или nil.

11.2.1 Флаги, используемые в Re.match

Эти флаги определены в пространстве имен Re.Match. Они используются во всех алгоритмах. Когда регулярное выражение применяется к последовательности символов, применяются правила, описанные в таблице 101

Таблица 101 - 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

Флаг	Описание
	15, часть 10, Регулярные Выражения.
NotBOB	Указывает, что выражения "\A" и "\\" не должны совпадать с подмножеством <i>[first, first)</i> .
NotEOB	Указывает, что выражения "\z", "\Z" и "\Z" не должны совпадать с подмножеством <i>[last, last)</i> .
NotBOL	Указывает, что выражение "^" не должны совпадать с подмножеством <i>[first, first)</i> .
NotEOL	Указывает, что выражение "\$" не должно совпадать с подмножеством <i>[last, last)</i> .
NotBOW	Указывает, что выражения "<" и "\b" не должны совпадать с подмножеством <i>[first, first)</i> .
NotEOW	Указывает, что выражения ">" и "\b" не должны совпадать с подмножеством <i>[last, last)</i> .
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.
Continious	Указывает, что выражение должно применяться к подмножеству, которое начинается с начала.
Partial	Указывает, что если не найдено ни одного совпадения, то допустимо вернуть совпадение <i>[from, last)</i> , при этом <i>from! = last</i> , если может существовать более длинная последовательность символов <i>[from, to)</i> , из которых <i>[from, last)</i> - это префикс, который приведет к полному совпадению. Этот флаг используется при сопоставлении неполных или очень длинных текстов; дополнительную информацию см. документацию по частичным сравнениям.
Extra	Дает указание механизму поиска сохранять всю доступную информацию о наличии совпадений; если совпадение повторяется снова, то информация о каждом из них будет доступна через методы match_results :: captures() или sub_match_captures() .
SingleLine	Эквивалентно обратному модификатору "m/" языка Perl; предотвращает поиск ^ после встроенного символа новой строки (для того, чтобы он совпадал только в начале исходного текста) и \$ от поиска перед встроенным символом новой строки (чтобы он совпадал только в конце исходного текста).
PrevAvail	Указывает, что --first является допустимой позицией итератора, когда этот флаг установлен, тогда флаги match_not_bol и match_not_bow игнорируются алгоритмами регулярных выражений (RE.7) и итераторов (RE.8).
DotNewLine	Указывает, что выражение "." не распознается как символ новой строки. Это инверсия модификатора "s/" языка Perl.

Флаг	Описание
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 Matches будет содержать только информацию об общем совпадении, а не о совпадении в подмножествах.

11.3 Метод search

Ищет скомпилированное регулярное выражение по заданной строке. Метод возвращает найденные подстроки.

```
Re.search(subject, matchFlags, pattern)
```

Параметры:

- `subject:string` – исходная строка;
- `matchFlags:int` – флаги, задающие правила применения регулярного выражения;
- `pattern:string, Regex` – шаблон поиска в виде строки или скомпилированного шаблона.

Возвращает:

- `matches:object` – подстроки, найденные в соответствии с шаблоном;
- `err:string` – сообщение об ошибке или `nil`.

11.4 Метод replace

Находит в заданной строке все фрагменты, удовлетворяющие регулярному выражению. Каждый найденный фрагмент форматируется в соответствии с форматтером и заменяет собой исходный текст.

```
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` – шаблон поиска в виде строки или скомпилированного шаблона.

Возвращает:

- `newString:string` – новая строка с замененными подстроками;
- `err:string` – сообщение об ошибке или `nil`.

11.5 Флаги, используемые при компиляции шаблона

Эти флаги определены в пространстве имен `Re`, они передаются в качестве последнего, но необязательного аргумента. По умолчанию используется синтаксис `Perl`. С детальным описанием флагов можно ознакомиться [по ссылке](#).

11.6 Флаги, используемые для замены

Эти флаги определены в пространстве имен `Re.Replace`. Они используются в алгоритме, используемом методом `Re.replace()` и находятся в таблице 102.

Таблица 102 – 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.

Флаг	Описание
FormatNoCopy	В случае, когда регулярное выражение применяется для замены в строке, фрагменты, не соответствующие регулярному выражению, не будут копироваться в результирующую строку.
FormatFirstOnly	Когда данный флаг установлен при операции поиска или замены, то заменяется только первое вхождение регулярного выражения.
FormatAll	Все синтаксические расширения включены, включая условные замены (? <i>ddexpression1:expression2</i>). Для дополнительных деталей см. Руководство по форматированию строк Boost .

12 Класс `Regex`

Класс `Regex` совмещает разбор регулярных выражений и компиляцию. Экземпляр этого класса может быть получен с помощью функции `Re.create()`.

Методы:

```
pattern, err = regex.toString()  
int, err = regex.getStatus()  
string = regex.__toString()
```

13 Класс Matches

Класс `Matches` содержит результат функций `Re.match()` и `Re.search()`.

13.1 Метод `getFirst`

```
position, err = matches.getFirst(group)
```

Параметры:

– `group: int, string` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

Возвращает:

– `position: int` – первая позиция (в байтах) исходной строки;
– `err: string` – сообщение об ошибке или `nil`.

13.2 Метод `getLength`

```
position, err = matches.getLength(group)
```

Параметры:

– `group: int, string` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

Возвращает:

– `length: int` – длина исходной строки в байтах;
– `err: string` – сообщение об ошибке или `nil`.

13.3 Метод `getSize`

```
size, err = matches.getSize()
```

Возвращает:

– `size: int` – количество найденных групп;
– `err: string` – сообщение об ошибке или `nil`.

13.4 Метод `getString`

```
substr, err = matches.getString(group, subject)
```

Параметры:

- `group:int, string` – позиция (или имя группы) найденных результатов, начинающаяся с 1;
- `subject:string` – исходная строка. **Внимание:** объект `Matches` сохраняет только смещения и не хранит исходную строку. Таким образом, необходимо передать ту же строку, которая использовалась для поиска.

Возвращает:

- `substr:string` – найденная подстрока;
- `err:string` – сообщение об ошибке или `nil`.

13.5 Метод `_toString`

Стандартная метафункция.

```
string = matches: __toString()
```

Пример:

```
local str = "-Номер:1234"
local regex = Re.create("-(\\w+):(\\d{4})")

local matches, err = Re.match(str, Re.Match.Default, regex)
print(tostring(regex), tostring(matches))

local number = matches:getString(3, str)
print(number)
```