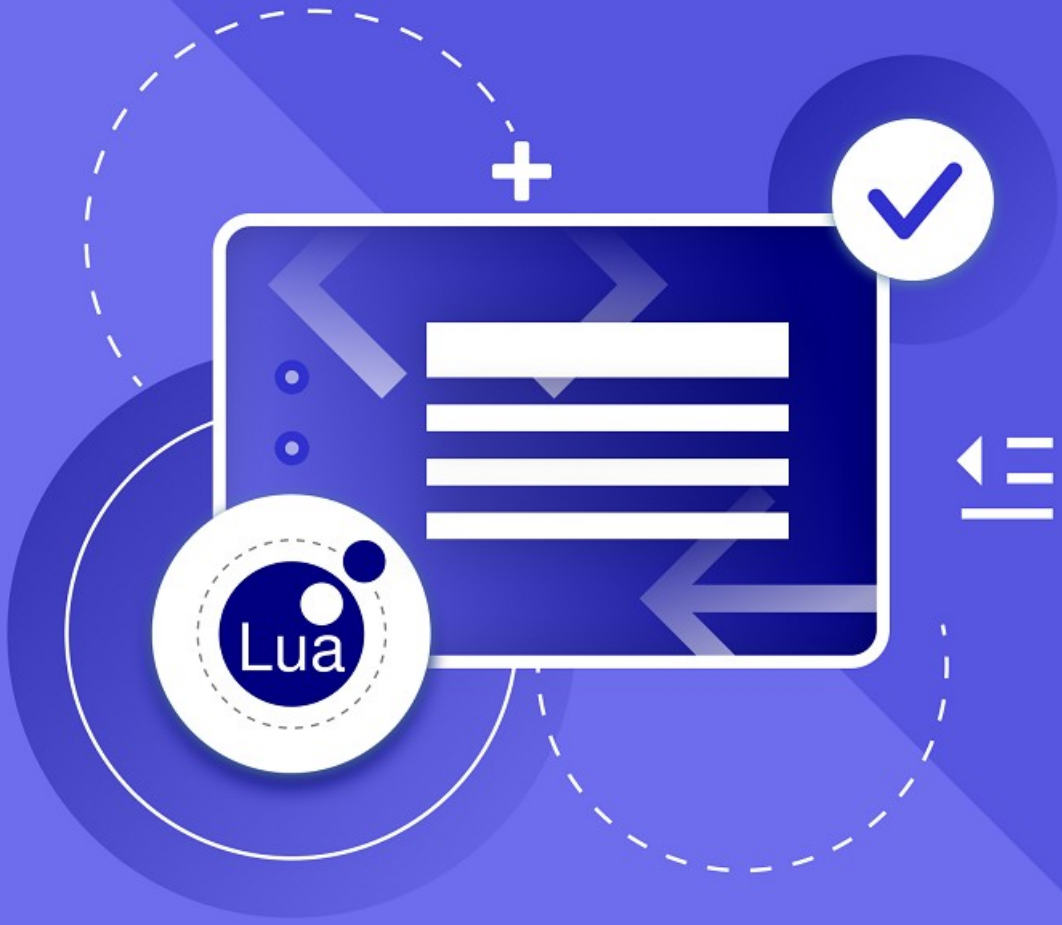


МойОфис

Справочник макрокоманд на языке программирования Lua

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «МОЙОФИС СТАНДАРТНЫЙ 3»

СПРАВОЧНИК МАКРОКОМАНД НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ LUA

3.1

Версия 1

На 270 листах

Дата публикации: 29.07.2024

Москва
2024

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

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

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

СОДЕРЖАНИЕ

1 Общие сведения ... 23
1.1 Назначение ... 23
1.2 Макрокоманды ПО МойОфис ... 23
1.3 Перечень эксплуатационной документации ... 24
2 Работа с макрокомандами ... 25
2.1 Редактор макрокоманд ... 25
2.1.1 Окно редактора макрокоманд ... 25
2.1.2 Создание макрокоманд ... 26
2.1.3 Выполнение макрокоманд ... 26
2.1.4 Редактирование макрокоманд ... 26
2.1.5 Удаление макрокоманд ... 27
2.1.6 Отладка макрокоманд ... 27
2.2 Пример подготовки и запуска макрокоманды ... 30
2.2.1 Редактирование и запуск макрокоманды в текстовом документе ... 30
2.2.2 Описание примера работы макрокоманды ... 30
2.3 Преобразование макрокоманд на языке программирования VBA ... 32
3 Объектная модель МойОфис ... 33
4 Работа с документами ... 34
4.1 Работа с текстовым документом ... 34
4.1.1 Разделы (секции) документа ... 34
4.1.1.1 Работа с колонтитулами раздела ... 35
4.1.1.2 Управление ориентацией и свойствами страниц раздела ... 35
4.1.2 Работа с таблицами текстового документа ... 36
4.1.3 Работа с закладками ... 38
4.1.4 Рецензирование документов ... 39
4.1.5 Работа с графическими объектами в текстовом документе ... 40
4.2 Работа с табличным документом ... 43
4.2.1 Работа с текстом в табличном документе ... 43
4.2.2 Копирование ячеек в табличном документе ... 43
4.2.3 Диаграммы ... 44
4.2.4 Работа с графическими объектами в табличном документе ... 45
4.2.5 Работа с листами табличного документа ... 46

4.2.6 Работа со сводными таблицами ... 48
4.2.7 Работа с фильтрами ... 49
4.3 Поиск в документе ... 50
4.4 Работа с макрокомандами ... 52
4.5 Работа с именованными диапазонами ... 52
5 Работа со строками и столбцами таблиц ... 54
5.1 Группировка строк и колонок таблицы ... 54
5.2 Управление видимостью строк / колонок ... 54
6 Работа с ячейками таблиц ... 56
6.1 Доступ к ячейкам ... 56
6.2 Форматирование ячеек ... 58
6.3 Форматирование границ ячеек ... 60
6.4 Объединение и разделение ячеек таблицы ... 61
7 Справочник таблиц DocumentAPI ... 62
7.1 Таблица DocumentAPI.AbsoluteFrame ... 62
7.1.1 Метод AbsoluteFrame:moveTo ... 62
7.1.2 Метод AbsoluteFrame:getTopLeft ... 63
7.1.3 Метод AbsoluteFrame:scale ... 63
7.1.4 Метод AbsoluteFrame:setDimensions ... 63
7.1.5 Метод AbsoluteFrame:getDimensions ... 64
7.2 Таблица DocumentAPI.AccountingCellFormatting ... 64
7.3 Таблица DocumentAPI.Alignment ... 65
7.4 Таблица DocumentAPI.Block ... 66
7.4.1 Методы toParagraph, toTable, toShape, toField ... 66
7.4.2 Метод Block.getRange ... 67
7.4.3 Метод Block.remove ... 67
7.4.4 Метод Block.getSection ... 67
7.5 Таблица DocumentAPI.Blocks ... 67
7.5.1 Метод Blocks:getBlock ... 68
7.5.2 Метод Blocks:getParagraph ... 68
7.5.3 Метод Blocks:getTable ... 68
7.5.4 Метод Blocks:getShape ... 69
7.5.5 Метод Blocks:getField ... 69
7.5.6 Метод Blocks:enumerate ... 69

7.5.7 Метод Blocks:enumerateParagraphs ... 69
7.5.8 Метод Blocks:enumerateTables ... 69
7.6 Таблица DocumentAPI.Bookmarks ... 70
7.6.1 Метод Bookmarks:getBookmarkRange ... 70
7.6.2 Метод Bookmarks:removeBookmark ... 70
7.7 Таблица DocumentAPI.Borders ... 70
7.8 Таблица DocumentAPI.CaseSensitive ... 72
7.9 Таблица DocumentAPI.Cell ... 72
7.9.1 Метод Cell:getBorders ... 72
7.9.2 Метод Cell:getCellProperties ... 73
7.9.3 Метод Cell:getCustomFormat ... 73
7.9.4 Метод Cell:getFormat ... 73
7.9.5 Метод Cell:getFormattedValue ... 73
7.9.6 Метод Cell:getFormulaAsString ... 74
7.9.7 Метод Cell:getHyperlink ... 74
7.9.8 Метод Cell:getParagraphProperties ... 74
7.9.9 Метод Cell:getPivotTable ... 74
7.9.10 Метод Cell:getRange ... 75
7.9.11 Метод Cell:getRawValue ... 75
7.9.12 Метод Cell:setContent ... 75
7.9.13 Метод Cell:setBool ... 75
7.9.14 Метод Cell:setBorders ... 75
7.9.15 Метод Cell:setCellProperties ... 76
7.9.16 Метод Cell:setCustomFormat ... 76
7.9.17 Метод Cell:setFormula ... 76
7.9.18 Метод Cell:setFormat ... 76
7.9.19 Метод Cell:setFormattedValue ... 79
7.9.20 Метод Cell:setNumber ... 79
7.9.21 Метод Cell:setParagraphProperties ... 79
7.9.22 Метод Cell:setText ... 79
7.9.23 Метод Cell::isPivotTableRoot ... 80
7.9.24 Метод Cell:unmerge ... 80
7.10 Таблица DocumentAPI.CellFormat ... 80
7.11 Таблица DocumentAPI.CellPosition ... 83

7.11.1 Поле CellPosition.column ... 83
7.11.2 Поле CellPosition.row ... 83
7.11.3 Метод CellPosition:toString ... 83
7.11.4 Метод CellPosition:__eq ... 84
7.12 Таблица DocumentAPI.CellProperties ... 84
7.12.1 Метод CellProperties:__eq ... 85
7.13 Таблица DocumentAPI.CellRange ... 85
7.13.1 Метод CellRange::autoFill ... 86
7.13.2 Метод CellRange:containsCell ... 86
7.13.3 Метод CellRange:copyInto ... 87
7.13.4 Метод CellRange:enumerate ... 87
7.13.5 Метод CellRange:getBeginRow ... 88
7.13.6 Метод CellRange:getBeginColumn ... 88
7.13.7 Метод CellRange:getCellProperties ... 88
7.13.8 Метод CellRange:getLastRow ... 89
7.13.9 Метод CellRange:getLastColumn ... 89
7.13.10 Метод CellRange:getTable ... 89
7.13.11 Метод CellRange:insertCurrentDateTime ... 89
7.13.12 Метод CellRange:merge ... 89
7.13.13 Метод CellRange:moveInto ... 90
7.13.14 Метод CellRange:setBorders ... 90
7.13.15 Метод CellRange:setCellProperties ... 91
7.14 Таблица DocumentAPI.CellRangePosition ... 91
7.14.1 Метод CellRangePosition:toString ... 93
7.14.2 Метод CellRangePosition:__eq ... 93
7.15 Таблица DocumentAPI.Charts ... 93
7.15.1 Метод Charts:getChartsCount ... 94
7.15.2 Метод Charts:getChart ... 94
7.15.3 Метод Charts:getChartIndexByDrawingIndex ... 95
7.16 Таблица DocumentAPI.Chart ... 95
7.16.1 Метод Chart:getType ... 95
7.16.2 Метод Chart:setType ... 96
7.16.3 Метод Chart:getRangesCount ... 96
7.16.4 Метод Chart:getRange ... 96

7.16.5 Метод Chart:getTitle ... 96
7.16.6 Метод Chart:setRange ... 96
7.16.7 Метод Chart:setRect ... 97
7.16.8 Метод Chart:isEmpty ... 97
7.16.9 Метод Chart:isSolidRange ... 97
7.16.10 Метод Chart:is3D ... 97
7.16.11 Метод Chart:getDirectionType ... 97
7.16.12 Метод Chart:getChartLabels ... 98
7.16.13 Метод Chart:getRangeAsString ... 98
7.16.14 Метод Chart:applySettings ... 98
7.17 Таблица DocumentAPI.ChartLabelsDetectionMode ... 99
7.18 Таблица DocumentAPI.ChartLabelsInfo ... 99
7.19 Таблица DocumentAPI.ChartRangeInfo ... 100
7.20 Таблица DocumentAPI.ChartRangeType ... 101
7.21 Таблица DocumentAPI.ChartSeriesDirectionType ... 101
7.22 Таблица DocumentAPI.ChartType ... 102
7.23 Таблица DocumentAPI.Color ... 103
7.23.1 Метод Color:getRGBAColor ... 103
7.23.2 Метод Color:getThemeColorID ... 104
7.23.3 Метод Color:setTransforms ... 104
7.23.4 Метод Color:getTransforms ... 104
7.23.5 Метод Color:__eq ... 104
7.24 Таблица DocumentAPI.ColorRGBA ... 104
7.24.1 Метод ColorRGBA:__eq ... 105
7.25 Таблица DocumentAPI.ColorTransforms ... 106
7.25.1 Метод ColorTransforms:apply ... 106
7.26 Таблица DocumentAPI.Comment ... 106
7.26.1 Метод Comment:getRange ... 106
7.26.2 Метод Comment:getText ... 107
7.26.3 Метод Comment:getInfo ... 107
7.26.4 Метод Comment:isResolved ... 107
7.26.5 Метод Comment:getReplies ... 107
7.27 Таблица DocumentAPI.Comments ... 108
7.27.1 Метод Comments:enumerate ... 108

7.28 Таблица DocumentAPI.ConditionalTableFilter ... 109
7.28.1 Метод ConditionalTableFilter:setAndOperation ... 109
7.28.2 Методы добавления условий ... 110
7.29 Таблица DocumentAPI.CurrencyCellFormatting ... 110
7.30 Таблица DocumentAPI.CurrencySignPlacement ... 111
7.31 Таблица DocumentAPI.document ... 112
7.31.1 Метод document:getAbsoluteFilePath ... 112
7.31.2 Метод document:getBlocks ... 113
7.31.3 Метод document:getBookmarks ... 113
7.31.4 Метод document:getScripts ... 113
7.31.5 Метод document:getRange ... 113
7.31.6 Метод document:isChangesTrackingEnabled ... 113
7.31.7 Метод document:setChangesTrackingEnabled ... 114
7.31.8 Метод document:getComments ... 114
7.31.9 Метод document:setPageProperties ... 114
7.31.10 Метод document:setFormulaType ... 115
7.31.11 Метод document:getFormulaType ... 115
7.31.12 Метод document:setPageOrientation ... 115
7.31.13 Метод document:enumerateSections ... 115
7.31.14 Метод document:getSections ... 115
7.31.15 Метод document:setMirroredMarginsEnabled ... 116
7.31.16 Метод document:areMirroredMarginsEnabled ... 116
7.31.17 Метод document:getPivotTablesManager ... 116
7.31.18 Метод document:getNamedExpressions ... 116
7.32 Таблица DocumentAPI.DatePatterns ... 116
7.33 Таблица DocumentAPI.DateTime ... 117
7.33.1 Метод DateTime:__eq ... 118
7.34 Таблица DocumentApi.DateTimeFormat ... 118
7.35 Таблица DocumentAPI.DateTimeCellFormatting ... 118
7.36 Таблица DocumentAPI.Field ... 119
7.37 Таблица DocumentAPI.Fill ... 119
7.37.1 Метод Fill:getColor ... 119
7.37.2 Метод Fill:getUrl ... 119
7.37.3 Метод Fill:isNoFill ... 119

7.38 Таблица DocumentAPI.FiltersRange 119
7.38.1 Метод FIltersRange:clear 120
7.38.2 Метод FIltersRange:eraseFilters 120
7.38.3 Метод FiltersRange:getCellRange 120
7.38.4 Метод FiltersRange:setFilters 120
7.39 Таблица DocumentAPI.FractionCellFormatting 121
7.40 Таблица DocumentAPI.FrozenRangePosition 122
7.40.1 Конструкторы 122
7.40.2 Метод FrozenRangePosition:create 122
7.40.3 Метод FrozenRangePosition:createFrozenArea 122
7.40.4 Метод FrozenRangePosition:createFrozenRows 123
7.40.5 Метод FrozenRangePosition:createFrozenCols 123
7.40.6 Метод FrozenRangePosition:isRowsCols 123
7.40.7 Метод FrozenRangePosition:isArea 124
7.40.8 Метод FrozenRangePosition:isRows 124
7.40.9 Метод FrozenRangePosition:isCols 124
7.40.10 Метод FrozenRangePosition:__eq 124
7.41 Таблица DocumentAPI.HeadersFooters 124
7.41.1 Метод HeadersFooters:enumerate 125
7.42 Таблица DocumentAPI.HeaderFooter 125
7.42.1 Метод HeaderFooter:getType 125
7.42.2 Метод HeaderFooter:getBlocks 125
7.42.3 Метод HeaderFooter:getRange 126
7.43 Таблица DocumentAPI.HeaderFooterType 126
7.44 Таблица DocumentAPI.HorizontalAnchorAlignment 126
7.45 Таблица DocumentAPI.HorizontalRelativeTo 127
7.46 Таблица DocumentAPI.HorizontalTextAnchoredPosition 127
7.46.1 Метод HorizontalTextAnchoredPosition:__eq 128
7.47 Таблица DocumentAPI.Hyperlink 129
7.47.1 Метод Hyperlink:__eq 129
7.48 Таблица DocumentAPI.Image 130
7.48.1 Метод Image:getFrame 130
7.48.2 Метод Image:remove 131
7.49 Таблица DocumentAPI.Images 131

7.49.1 Метод Images:enumerate ... 131
7.50 Таблица DocumentAPI.InlineFrame ... 132
7.50.1 Метод InlineFrame:setPosition ... 132
7.50.2 Метод InlineFrame:getPosition ... 133
7.50.3 Метод InlineFrame:setDimensions ... 133
7.50.4 Метод InlineFrame:getDimensions ... 134
7.50.5 Метод InlineFrame:setWrapType ... 134
7.50.6 Метод InlineFrame:getWrapType ... 134
7.51 Таблица DocumentAPI.Insets ... 134
7.52 Таблица DocumentAPI.ListSchema ... 135
7.53 Таблица DocumentAPI.LineEndingProperties ... 136
7.53.1 Метод LineEndingProperties:__eq ... 137
7.54 Таблица DocumentAPI.LineEndingStyle ... 137
7.55 Таблица DocumentAPI.LineProperties ... 138
7.55.1 Поле LineProperties.style ... 139
7.55.2 Поле LineProperties.width ... 139
7.55.3 Поле LineProperties.color ... 139
7.55.4 Поле LineProperties.headLineEndingProperties ... 140
7.55.5 Поле LineProperties.tailLineEndingProperties ... 140
7.55.6 Метод LineProperties:__eq ... 140
7.56 Таблица DocumentAPI.LineSpacing ... 140
7.57 Таблица DocumentAPI.LineSpacingRule ... 141
7.58 Таблица DocumentAPI.LineStyle ... 143
7.59 Таблица DocumentAPI.MediaObject ... 144
7.59.1 Метод MediaObject:toImage ... 144
7.59.2 Метод MediaObject:toChart ... 144
7.59.3 Метод MediaObject:getFrame ... 145
7.60 Таблица DocumentAPI.MediaObjects ... 145
7.60.1 Метод MediaObjects:enumerate ... 146
7.61 Таблица DocumentAPI.NamedExpressions ... 147
7.61.1 Метод NamedExpressions:get ... 147
7.61.2 Метод NamedExpressions:enumerate ... 147
7.61.3 Метод NamedExpression:addExpression ... 147
7.61.4 Метод NamedExpressions:removeExpression ... 148

7.62 Таблица DocumentAPI.NamedExpression 148
7.62.1 Метод NamedExpression:getName 148
7.62.2 Метод NamedExpression:getExpression 148
7.62.3 Метод NamedExpression:getCellRange 149
7.63 Таблица DocumentAPI.NamedExpressionsValidationResult 149
7.64 Таблица DocumentAPI.NumberCellFormatting 149
7.65 Таблица DocumentAPI.PageFieldOrder 150
7.66 Таблица DocumentAPI.PageProperties 150
7.66.1 Метод PageProperties:__eq 151
7.67 Таблица DocumentAPI.Paragraph 151
7.67.1 Метод Paragraph:getParagraphProperties 152
7.67.2 Метод Paragraph:setParagraphProperties 153
7.67.3 Метод Paragraph:getListSchema 153
7.67.4 Метод Paragraph:setListSchema 153
7.67.5 Метод Paragraph:getListLevel 154
7.67.6 Метод Paragraph:setListLevel 154
7.67.7 Метод Paragraph:increaseListLevel 154
7.67.8 Метод Paragraph:decreaseListLevel 154
7.68 Таблица DocumentAPI.ParagraphProperties 155
7.69 Таблица DocumentAPI.Paragraphs 157
7.69.1 Метод Paragraphs:setListSchema 158
7.69.2 Метод Paragraphs:setListLevel 158
7.69.3 Метод Paragraphs:increaseListLevel 158
7.69.4 Метод Paragraphs:decreaseListLevel 159
7.69.5 Метод Paragraphs:enumerate 159
7.70 Таблица DocumentAPI.PageOrientation 160
7.71 Таблица DocumentAPI.PercentageCellFormatting 160
7.72 Таблица DocumentAPI.PivotTable 161
7.72.1 Метод PivotTable:remove 161
7.72.2 Метод PivotTable:getSourceRangeAddress 161
7.72.3 Метод PivotTable:getSourceRange 161
7.72.4 Метод PivotTable:getPivotRange 162
7.72.5 Метод PivotTable:changeSourceRange 162
7.72.6 Метод PivotTable:isRowGrandTotalEnabled 162

7.72.7 Метод PivotTable:isColumnGrandTotalEnabled ... 162
7.72.8 Метод PivotTable:getPivotTableCaptions ... 163
7.72.9 Метод PivotTable:getPivotTableLayoutSettings ... 163
7.72.10 Метод PivotTable:getUnsupportedFeatures ... 163
7.72.11 Метод PivotTable:getFieldsList ... 163
7.72.12 Метод PivotTable:getRowFields ... 164
7.72.13 Метод PivotTable:getColumnFields ... 164
7.72.14 Метод PivotTable:getValueFields ... 164
7.72.15 Метод PivotTable:getPageFields ... 165
7.72.16 Метод PivotTable:getFieldCategories ... 165
7.72.17 Метод PivotTable:getFieldItems ... 165
7.72.18 Метод PivotTable:getFieldItemsByName ... 165
7.72.19 Метод PivotTable:getFilter ... 165
7.72.20 Метод PivotTable:getFilters ... 166
7.72.21 Метод PivotTable:update ... 166
7.72.22 Метод PivotTable:createPivotTableEditor ... 166
7.73 Таблица DocumentAPI.PivotTableCaptions ... 166
7.74 Таблица DocumentAPI.PivotTableCategoryField ... 167
7.75 Таблица DocumentAPI.PivotTableEditor ... 167
7.75.1 Метод PivotTableEditor:addField ... 167
7.75.2 Метод PivotTableEditor:moveField ... 168
7.75.3 Метод PivotTableEditor:removeField ... 168
7.75.4 Метод PivotTableEditor:reorderField ... 169
7.75.5 Метод PivotTableEditor:enableField ... 169
7.75.6 Метод PivotTableEditor:disableField ... 169
7.75.7 Метод PivotTableEditor:setSummarizeFunction ... 169
7.75.8 Метод PivotTableEditor:setFilter ... 170
7.75.9 Метод PivotTableEditor:setFilters ... 170
7.75.10 Метод PivotTableEditor:setCaptions ... 171
7.75.11 Метод PivotTableEditor:setLayoutSettings ... 171
7.75.12 Метод PivotTableEditor:setGrandTotalSettings ... 171
7.75.13 Метод PivotTableEditor:apply ... 171
7.76 Таблица DocumentAPI.PivotTableField ... 172
7.77 Таблица DocumentAPI.PivotTableFieldCategory ... 172

7.78 Таблица DocumentAPI.PivotTableFieldCategories ... 172
7.78.1 Метод PivotTableFieldCategories:enumerate ... 173
7.79 Таблица DocumentAPI.PivotTableFieldProperties ... 173
7.80 Таблица DocumentAPI.PivotTableFilter ... 173
7.80.1 Метод PivotTableFilter:getFieldName ... 174
7.80.2 Метод PivotTableFilter:getCount ... 174
7.80.3 Метод PivotTableFilter:getName ... 174
7.80.4 Метод PivotTableFilter:isHidden ... 175
7.80.5 Метод PivotTableFilter:setHidden ... 175
7.81 Таблица DocumentAPI.PivotTableFilters ... 175
7.81.1 Метод PivotTableFilters:enumerate ... 176
7.82 Таблица DocumentAPI.PivotTableFunction ... 176
7.83 Таблица DocumentAPI.PivotTableItem ... 177
7.83.1 Метод PivotTableItem:getName ... 177
7.83.2 Метод PivotTableItem:getAlias ... 177
7.83.3 Метод PivotTableItem:getItemType ... 177
7.83.4 Метод PivotTableItem:isCollapsed ... 177
7.84 Таблица DocumentAPI.PivotTableItems ... 178
7.84.1 Метод PivotTableItems:enumerate ... 178
7.85 Таблица DocumentAPI.PivotTableItemType ... 178
7.86 Таблица DocumentAPI.PivotTableLayoutSettings ... 179
7.87 Таблица DocumentAPI.PivotTablePageField ... 180
7.88 Таблица DocumentAPI.PivotTableReportLayout ... 180
7.89 Таблица DocumentAPI.PivotTablesManager ... 180
7.89.1 Метод PivotTablesManager:create ... 180
7.90 Таблица DocumentAPI.PivotTableUnsupportedFeature ... 181
7.91 Таблица DocumentAPI.PivotTableValueField ... 181
7.92 Таблица DocumentAPI.PivotTableUpdateResult ... 182
7.93 Таблица DocumentAPI.PointU ... 183
7.93.1 Метод PointU:toString ... 183
7.94 Таблица DocumentAPI.Position ... 183
7.94.1 Метод Position:getCell ... 184
7.94.2 Метод Position:insertText ... 184
7.94.3 Метод Position:insertTable ... 184

7.94.4 Метод Position:insertPageBreak ... 185
7.94.5 Метод Position:insertLineBreak ... 185
7.94.6 Метод Position:insertBookmark ... 186
7.94.7 Метод Position:insertSectionBreak ... 186
7.94.8 Метод Position:insertHyperlink ... 186
7.94.9 Метод Position:insertImage ... 187
7.94.10 Метод Position:removeBackward ... 187
7.94.11 Метод Position:removeForward ... 187
7.94.12 Метод Position:__eq ... 187
7.95 Таблица DocumentAPI.PrintSettings ... 188
7.96 Таблица DocumentAPI.PrintDocumentResult ... 189
7.97 Таблица DocumentAPI.Range ... 189
7.97.1 Метод Range:getBegin ... 191
7.97.2 Метод Range:getEnd ... 191
7.97.3 Метод Range:extractText ... 192
7.97.4 Метод Range:removeContent ... 192
7.97.5 Метод Range:lockContent ... 192
7.97.6 Метод Range:unlockContent ... 193
7.97.7 Метод Range:isContentLocked ... 193
7.97.8 Метод Range:replaceText ... 194
7.97.9 Метод Range:setHyperlink ... 194
7.97.10 Метод Range:getTextProperties ... 195
7.97.11 Метод Range:setTextProperties ... 195
7.97.12 Метод Range:enumerateBlocks ... 196
7.97.13 Метод Range:enumerateTrackedChanges ... 196
7.97.14 Метод Range:getComments ... 196
7.97.15 Метод Range:getParagraphs ... 197
7.97.16 Метод Range:getImages ... 197
7.97.17 Метод Range:getInlineObjects ... 198
7.98 Таблица DocumentAPI.RangeBorders ... 198
7.99 Таблица DocumentAPI.RectU ... 198
7.99.1 Метод RectU:toString ... 198
7.100 Таблица DocumentAPI.ScaleFrom ... 199
7.101 Таблица DocumentAPI.ScientificCellFormatting ... 199

7.102 Таблица DocumentAPI.Script ... 200
7.102.1 Таблица DocumentAPI.Scripting ... 200
7.102.1.1 Метод Scripting:runScript ... 200
7.102.2 Метод Script:getName ... 200
7.102.3 Метод Script:setName ... 200
7.102.4 Метод Script:getBody ... 201
7.102.5 Метод Script:setBody ... 201
7.103 Таблица DocumentAPI.ScriptPosition ... 201
7.104 Таблица DocumentAPI.Scripts ... 201
7.104.1 Метод Scripts:getScript ... 202
7.104.2 Метод Scripts:setScript ... 202
7.104.3 Метод Scripts:removeScript ... 202
7.104.4 Метод Scripts:enumerate ... 202
7.105 Таблица DocumentAPI.Search ... 203
7.105.1 Метод Search:findText ... 203
7.106 Таблица DocumentAPI.Section ... 204
7.106.1 Метод Section:setPageProperties ... 204
7.106.2 Метод Section:getPageProperties ... 204
7.106.3 Метод Section:setPageOrientation ... 204
7.106.4 Метод Section:getPageOrientation ... 205
7.106.5 Метод Section:getRange ... 205
7.106.6 Метод Section:getHeaders ... 205
7.106.7 Метод Section:getFooters ... 205
7.107 Таблица DocumentAPI.Sections ... 206
7.107.1 Метод Sections:enumerate ... 206
7.108 Таблица DocumentAPI.Shape ... 206
7.108.1 Метод Shape:getShapeProperties ... 206
7.108.2 Метод Shape:setShapeProperties ... 207
7.109 Таблица DocumentAPI.ShapeProperties ... 207
7.109.1 Поле ShapeProperties:borderProperties ... 207
7.109.2 Поле ShapeProperties:verticalAlignment ... 207
7.109.3 Поле ShapeProperties:fill ... 207
7.109.4 Поле ShapeProperties:shapeTextLayout ... 207
7.110 Таблица DocumentAPI.ShapeTextLayout ... 208

7.111 Таблица DocumentAPI.SizeU 208
7.111.1 Метод SizeU:toString 208
7.112 Таблица DocumentAPI.Table 208
7.112.1 Метод Table:createFiltersRange 209
7.112.2 Метод Table:setName 209
7.112.3 Метод Table:getName 210
7.112.4 Метод Table:getFiltersRange 210
7.112.5 Метод Table:getRowsCount 210
7.112.6 Метод Table:getColumnsCount 211
7.112.7 Метод Table:getCell 211
7.112.8 Метод Table:getCellRange 211
7.112.9 Метод Table:insertColumnAfter 212
7.112.10 Метод Table:insertColumnBefore 212
7.112.11 Метод Table:insertRowAfter 213
7.112.12 Метод Table:insertRowBefore 213
7.112.13 Метод Table:isColumnVisible 214
7.112.14 Метод Table:isRowVisible 214
7.112.15 Метод Table:removeColumn 215
7.112.16 Метод Table:removeRow 215
7.112.17 Метод Table:groupRows 215
7.112.18 Метод Table:ungroupRows 216
7.112.19 Метод Table:clearRowGroups 216
7.112.20 Метод Table:groupColumns 216
7.112.21 Метод Table:ungroupColumns 217
7.112.22 Метод Table:clearColumnGroups 217
7.112.23 Метод Table:setColumnsVisible 217
7.112.24 Метод Table:setRowsVisible 218
7.112.25 Метод Table:setColumnWidth 218
7.112.26 Метод Table:setRowHeight 219
7.112.27 Метод Table:duplicate 219
7.112.28 Метод Table:remove 219
7.112.29 Метод Table:moveTo 220
7.112.30 Метод Table:setShowZeroValue 220
7.112.31 Метод Table:getShowZeroValue 220

7.112.32 Метод Table:setVisible ... 220
7.112.33 Метод Table:isVisible ... 221
7.112.34 Метод Table:getFrozenRange ... 221
7.112.35 Метод Table:freeze ... 221
7.112.36 Метод Table:__eq ... 222
7.112.37 Метод Table:setPrintArea ... 222
7.112.38 Метод Table:setPrintAreas ... 222
7.112.39 Метод Table:getPrintAreas ... 223
7.112.40 Метод Table:getCharts ... 223
7.112.41 Метод Table:getImages ... 223
7.112.42 Метод Table:getMediaObjects ... 223
7.112.43 Метод Table:getNamedExpressions ... 224
7.113 Таблица DocumentAPI.TableFilters ... 224
7.113.1 Метод TableFilters:clear ... 224
7.113.2 Метод TableFilters:erase ... 224
7.113.3 Метод TableFilters:setFilter ... 225
7.114 Таблица DocumentAPI.TableRangeInfo ... 225
7.115 Таблица DocumentAPI.TextAnchoredPosition ... 226
7.115.1 Метод TextAnchoredPosition:__eq ... 227
7.116 Таблица DocumentAPI.TextLayout ... 227
7.117 Таблица DocumentAPI.TextOrientation ... 228
7.117.1 Метод TextOrientation:getAngle ... 228
7.117.2 Метод TextOrientation:isStackedChars ... 228
7.117.3 Метод TextOrientation:__eq ... 228
7.118 Таблица DocumentAPI.TextProperties ... 229
7.119 Таблица DocumentAPI.TextWrapType ... 231
7.120 Таблица DocumentAPI.ThemeColorID ... 231
7.121 Таблица DocumentAPI.TimePatterns ... 232
7.122 Таблица DocumentAPI.TrackedChange ... 232
7.122.1 Метод TrackedChange:getRange ... 233
7.122.2 Метод TrackedChange:getType ... 233
7.122.3 Метод TrackedChange:getInfo ... 233
7.123 Таблица DocumentAPI.TrackedChangeInfo ... 234
7.123.1 Метод TrackedChangeInfo:__eq ... 234

7.124 Таблица DocumentAPI.TrackedChangeType 234
7.125 Таблица DocumentAPI.ValueFieldsOrientation 235
7.126 Таблица DocumentAPI.ValuesTableFilter 235
7.126.1 Метод ValuesTableFilter:add 235
7.126.2 Метод ValuesTableFilter:clear 236
7.127 Таблица DocumentAPI.VectorString 236
7.127.1 Метод VectorString:size 236
7.127.2 Метод VectorString:max_size 236
7.127.3 Метод VectorString:empty 237
7.127.4 Метод VectorString:clear 237
7.127.5 Метод VectorString:push_back 237
7.127.6 Метод VectorString:pop_back 237
7.127.7 Метод VectorString:front 237
7.127.8 Метод VectorString:back 238
7.127.9 Метод VectorString:__getitem 238
7.127.10 Метод VectorString:__setitem 238
7.128 Таблица DocumentAPI.VectorUInt 238
7.128.1 Метод VectorUInt:size 239
7.128.2 Метод VectorUInt:max_size 239
7.128.3 Метод VectorUInt:empty 239
7.128.4 Метод VectorUInt:clear 239
7.128.5 Метод VectorUInt:push_back 240
7.128.6 Метод VectorUInt:pop_back 240
7.128.7 Метод VectorUInt:front 240
7.128.8 Метод VectorUInt:back 240
7.128.9 Метод VectorUInt:__getitem 241
7.128.10 Метод VectorUInt:__setitem 241
7.129 Таблица DocumentAPI.VerticalAlignment 241
7.130 Таблица DocumentAPI.VerticalAnchorAlignment 242
7.131 Таблица DocumentAPI.VerticalRelativeTo 243
7.132 Таблица DocumentAPI.VerticalTextAnchoredPosition 243
7.132.1 Метод VerticalTextAnchoredPosition:__eq 244
7.133 Таблица DocumentAPI.WorksheetPrinterFitType 244
8 Справочник функций DocumentAPI 246

8.1 Функция DocumentAPI:createSearch 246
8.2 Функция DocumentAPI.createScripting 246
9 Справочник таблиц EditorAPI 247
9.1 Таблица EditorAPI.SelectionMode 247
9.2 Таблица EditorAPI.SelectionDirection 247
9.3 Таблица EditorAPI.TableSelectionUnit 248
9.4 Таблица EditorAPI.TextSelectionUnit 248
10 Справочник функций EditorAPI 249
10.1 Функция EditorAPI.changeSelection 249
10.2 Функция EditorAPI.getSelection 249
10.3 Функция EditorAPI.setSelection 250
10.4 Функция EditorAPI.messageBox 251
10.5 Функция EditorAPI.showPrintDialog 251
10.6 Функция EditorAPI.printDocument 251
10.7 Функция EditorAPI.isPrinterAvailable 252
10.8 Функция EditorAPI.getActiveWorksheet 252
10.9 Функция EditorAPI.setActiveWorksheet 252
11 Функции для работы со строками в формате Юникод (UTF-8) 253
11.1 Функция utf8.char 253
11.2 Функция utf8.codes 253
11.3 Функция utf8.codepoint 254
11.4 Функция utf8.charpattern 254
11.5 Функция utf8.upper 254
11.6 Функция utf8.lower 255
11.7 Функция utf8.substr 255
11.8 Функция utf8.compare 256
11.9 Функция utf8.islower 257
11.10 Функция utf8.isupper 257
11.11 Функция utf8.isdigit 257
11.12 Функция utf8.isalpha 258
11.13 Функция utf8.len 258
11.14 Функция utf8.offset 259
11.15 Функция utf8.next 259
12 Функции для работы с регулярными выражениями 260

12.1 Функция Re.create 260
12.2 Функция Re.match 260
12.2.1 Флаги, используемые в Re.match 260
12.3 Функция Re.search 262
12.4 Функция Re.replace 262
12.5 Флаги, используемые для замены 263
13 Функции для работы с датой и временем 265
13.1 Функция os.clock 266
13.2 Функция os.date 266
13.3 Функция os.difftime 267
13.4 Функция os.time 268
14 Класс Matches 269
14.1 Метод getFirst 269
14.2 Метод getLength 269
14.3 Метод getSize 269
14.4 Метод getString 269
14.5 Метод _tostring 270

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

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

Таблица 1 - Сокращения и расшифровки

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

1 Общие сведения

1.1 Назначение

«МойОфис Стандартный 3» – настольный офисный пакет для работы с текстовыми документами и электронными таблицами, презентациями, почтой, календарями и адресными книгами. Приложения устанавливаются на компьютер пользователя и не требуют подключения к интернету.

В состав продукта входят следующие приложения для работы на компьютерах с операционными системами Linux, Windows и macOS:

«МойОфис Текст» – редактор для быстрого и удобного создания и форматирования текстовых документов любой сложности;

«МойОфис Таблица» – редактор для создания электронных таблиц, ведения расчетов, анализа данных, формирования сводных отчетов и автоматизации обработки данных с использованием макрокоманд;

«МойОфис Презентация» – приложение для просмотра и демонстрации презентаций;

«Редактор презентаций» – редактор на основе компонентов с открытым исходным кодом для быстрого и удобного создания и оформления презентаций;

«МойОфис Почта» – почтовый клиент для работы с электронными сообщениями, календарями, задачами и адресными книгами.

Подробное описание возможностей продукта приведено в документе «МойОфис Стандартный. Функциональные возможности».

В операционной системе macOS доступны приложения «МойОфис Текст», «МойОфис Таблица», «Редактор Презентаций» и «МойОфис Почта».

1.2 Макрокоманды ПО МойОфис

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

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

Справочное руководство по языку программирования Lua опубликовано по ссылкам: https://lua.org.ru (ru), https://devdocs.io (en).

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

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

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

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

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

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

1.3 Перечень эксплуатационной документации

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

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

2 Работа с макрокомандами

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

2.1 Редактор макрокоманд

2.1.1 Окно редактора макрокоманд

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

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

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

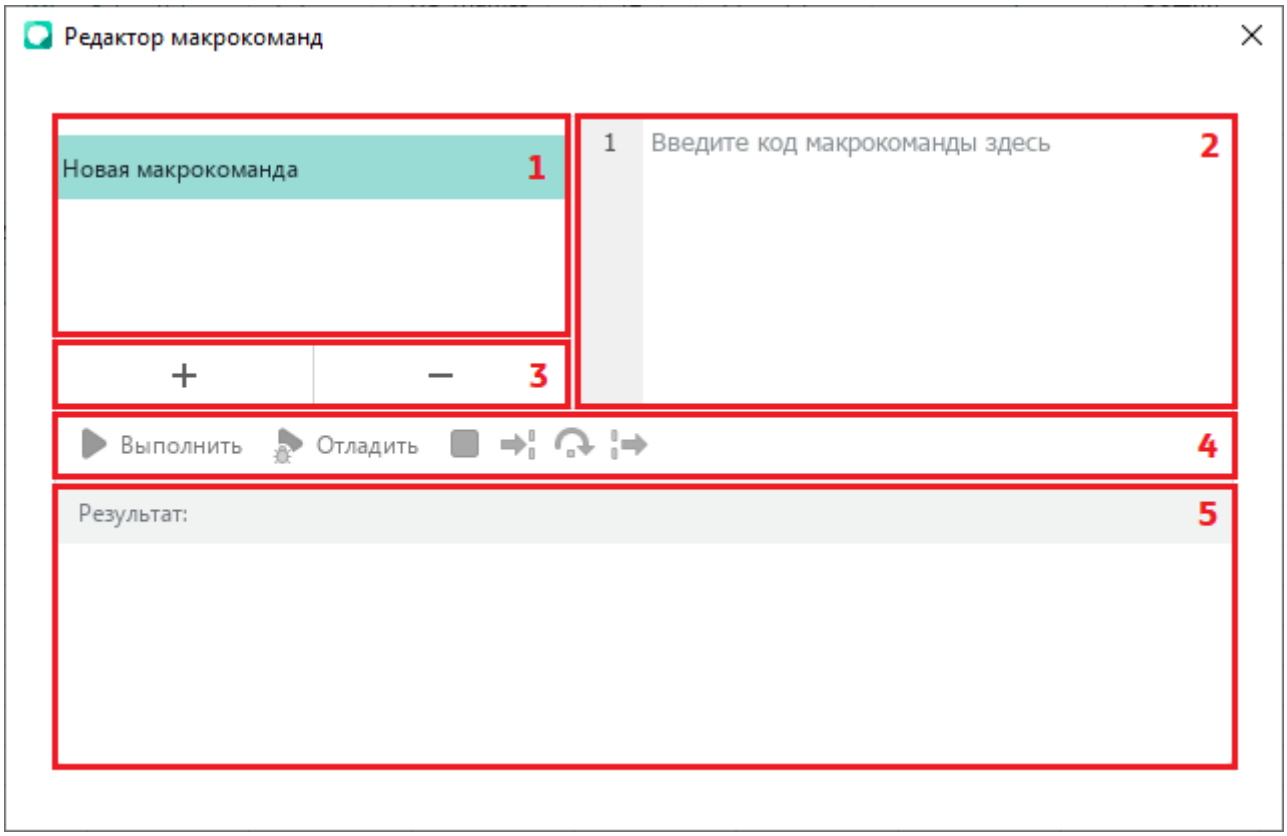


Рисунок 1 – Окно редактора макрокоманд

2.1.2 Создание макрокоманд

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

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

2.1.3 Выполнение макрокоманд

Чтобы выполнить макрокоманду, выберите ее в перечне макрокоманд и нажмите кнопку **Выполнить** (см. Рисунок 1).

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

2.1.4 Редактирование макрокоманд

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

2.1.5 Удаление макрокоманд

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

2.1.6 Отладка макрокоманд

Для отладки макрокоманды выполните следующие действия:

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

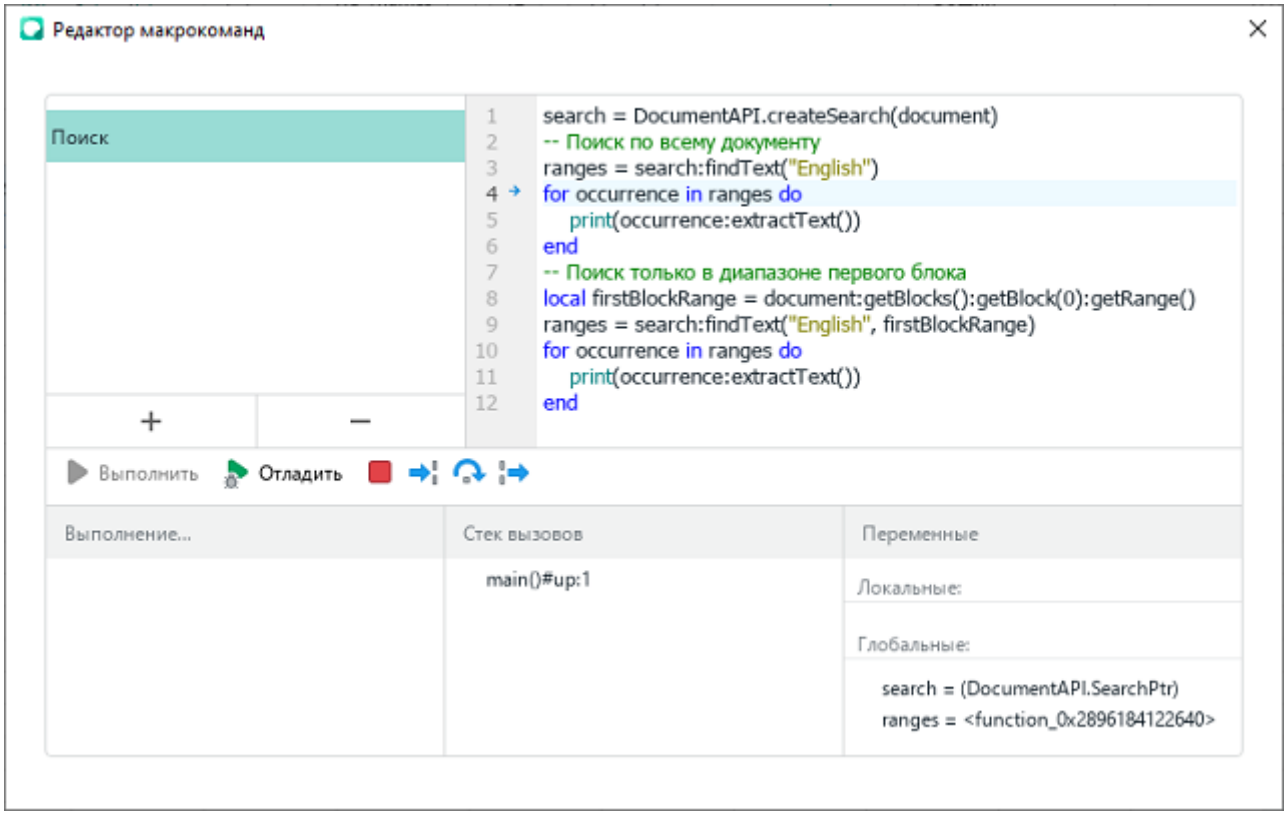


Рисунок 2 – Окно редактора макрокоманд в начале отладки

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

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

–  – для выполнения одного шага отладки или захода в тело функции, если таковая есть в текущей позиции отладки;

–  – для выполнения одного шага отладки без захода в тело функции;

–  – для продолжения выполнения макрокоманды до момента выхода из функции, в которой отладчик находится в текущей позиции.

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

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

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

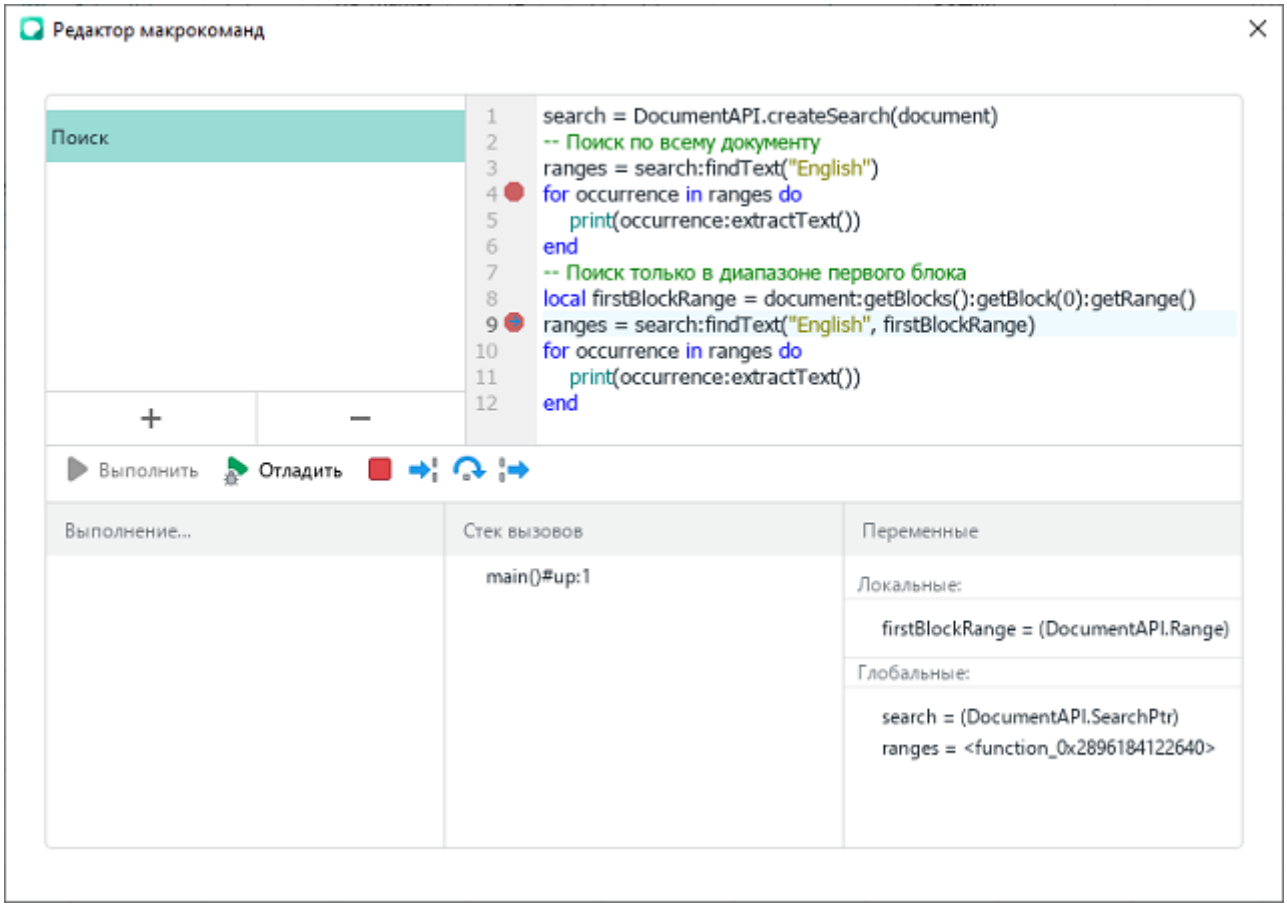


Рисунок 3 – Окно редактора макрокоманд в процессе отладки макрокоманд

Отладка завершается при достижении конца макрокоманды.

2.2 Пример подготовки и запуска макрокоманды

2.2.1 Редактирование и запуск макрокоманды в текстовом документе

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

Чтобы создать и запустить макрокоманду, необходимо выполнить следующие действия:

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

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

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

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

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

2.2.2 Описание примера работы макрокоманды

Ниже приведено описание макрокоманды, текст которой приведен в разделе Редактирование и запуск макрокоманды в текстовом документе.

С помощью последовательности вызовов устанавливается курсор в начало документа:

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

Таблица `DocumentAPI.Document` представляет текущий открытый текстовый документ.

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

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

Следующая последовательность вызовов настраивает форматирование для документа:

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

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

Таблица `DocumentAPI.TextProperties` позволяет управлять такими характеристиками как наименование и размер шрифта, цвет, начертание и т.п.

В данном примере устанавливаются две настройки форматирования:

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

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

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

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

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

2.3 Преобразование макрокоманд на языке программирования VBA

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

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

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

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

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

3 Объектная модель МойОфис

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

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

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

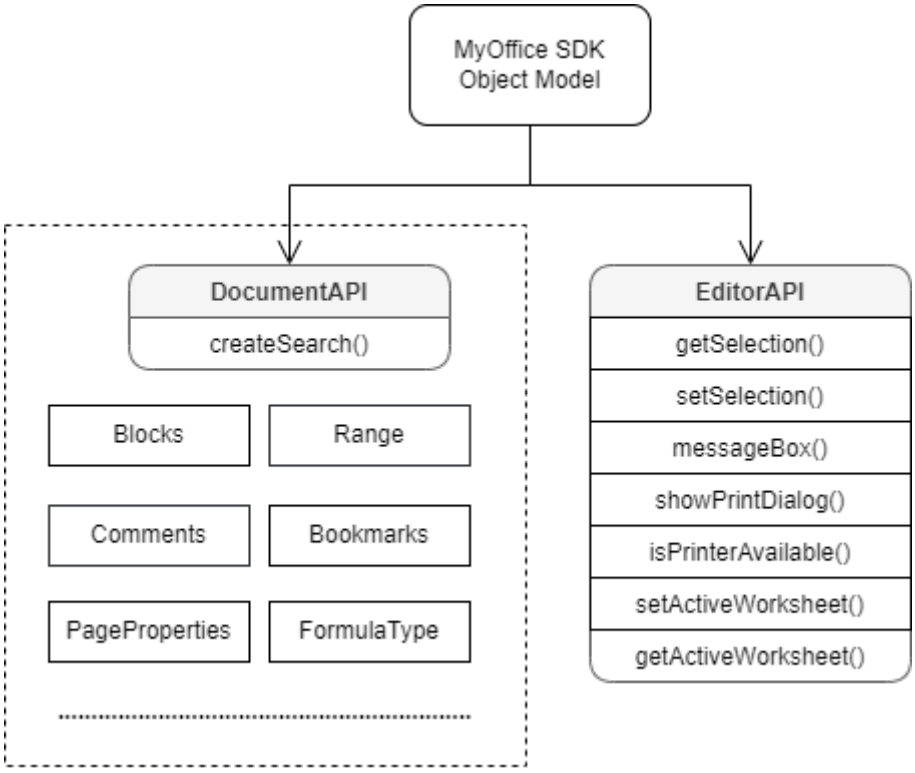


Рисунок 4 – Объектная модель МойОфис SDK.

4 Работа с документами

4.1 Работа с текстовым документом

4.1.1 Разделы (секции) документа

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

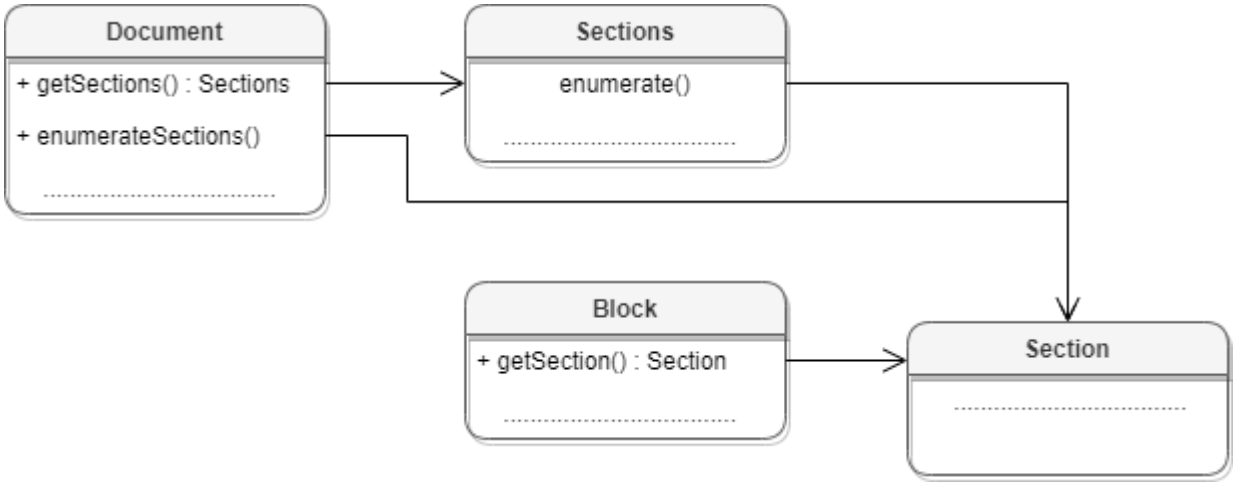


Рисунок 5 – Объектная модель таблиц для работы с секциями

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

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

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

Примеры:

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

```
    print(properties.height)
end
```

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

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

4.1.1.1 Работа с колонтитулами раздела

Для получения колонтитулов раздела следует использовать методы Section::getHeaders() или Section::getFooters().

Пример:

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

4.1.1.2 Управление ориентацией и свойствами страниц раздела

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

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

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

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

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

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

Ориентация страниц объекта Section может быть получена с использованием метода Section::getPageOrientation().

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

Свойства страниц объекта Section могут быть получены с использованием метода Section::getPageProperties().

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

4.1.2 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены являются листы документа. Доступ к объектам Table осуществляется из Blocks (см. Рисунок 6).

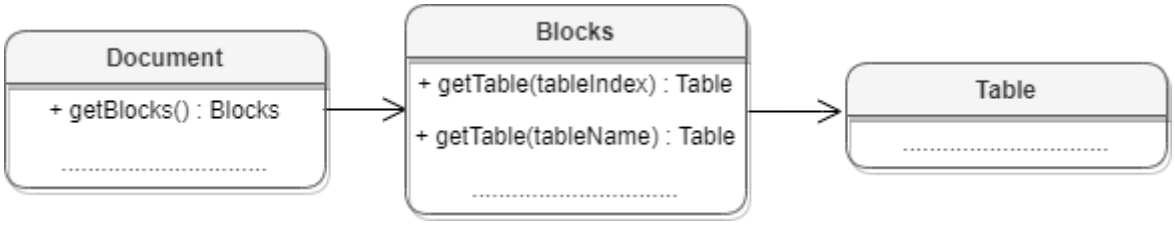


Рисунок 6 – Объектная модель для работы с таблицами

Для работы с таблицами доступны следующие операции:

– перечисление таблиц документа;

– получение таблицы документа;

– вставка таблицы в позицию документа;

– переименование таблицы;

– удаление таблицы.

Ниже приведены примеры работы с таблицами в текстовых документах:

Перечисление таблиц документа:

Для перечисления таблиц текстового документа используется метод `Blocks::getTablesEnumerator()`.

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

Получение таблицы текстового документа:

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

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

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

Вставка таблицы в текстовый документ:

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

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

Вставка текста в таблицу текстового документа:

Для вставки текста в таблицу текстового документа используется метод Position::insertText().

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

Переименование таблицы:

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

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

Удаление таблицы:

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

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

4.1.3 Работа с закладками

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

Доступны следующие операции с закладками:

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

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

Вставка закладки в указанное местоположение

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

Удаление закладки с заданным именем

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

Поиск закладки по имени

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

Замена текстового содержимого закладки

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

Вставка текста в закладку

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

Удаление содержимого закладки

```
sig_rng:removeContent()
```

Получение текстового содержимого закладки

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

Вставка таблицы в закладку

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

4.1.4 Рецензирование документов

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

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

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

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

Пример:

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

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

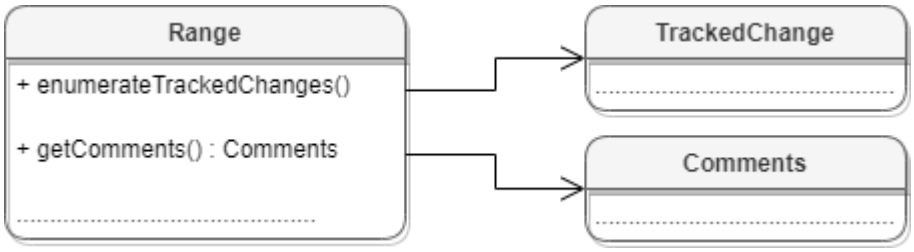


Рисунок 7 – Инструменты рецензирования документа

4.1.5 Работа с графическими объектами в текстовом документе

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

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

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

– Вставка изображений в текстовый документ. Место вставки определяется типом `Position`.

– Перемещение графических объектов, изменение их размеров и масштаба.

Перечисление графических объектов в текстовом документе.

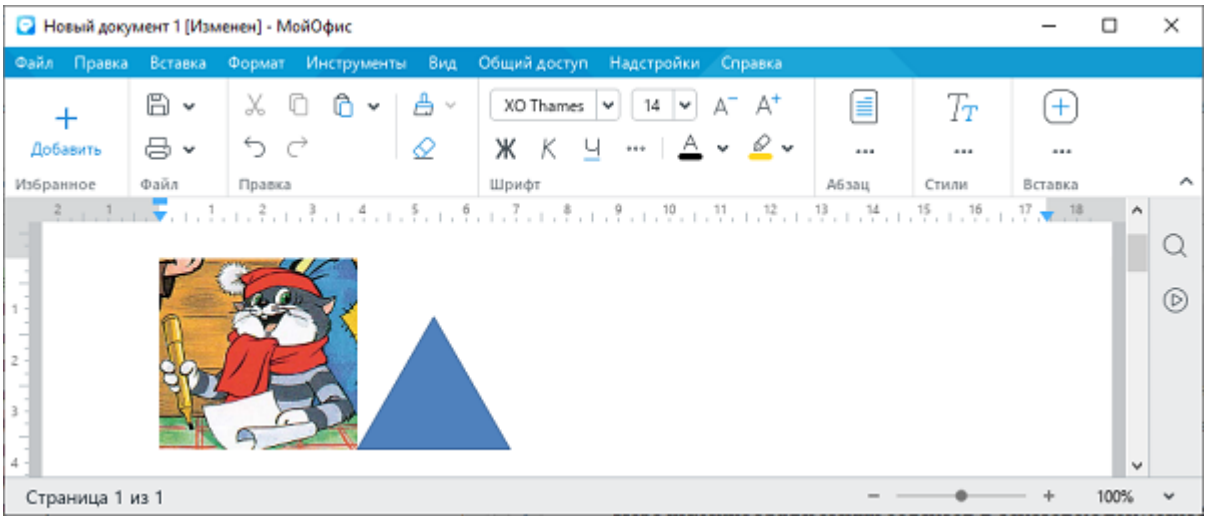


Рисунок 8 – Графические объекты в текстовом документе

Вариант 1: перечисление графических объектов в текстовом документе

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

Вывод:

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

Вариант 2: перечисление изображений в текстовом документе

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

Вывод:

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

Перечисление графических объектов в таблицах текстового документа

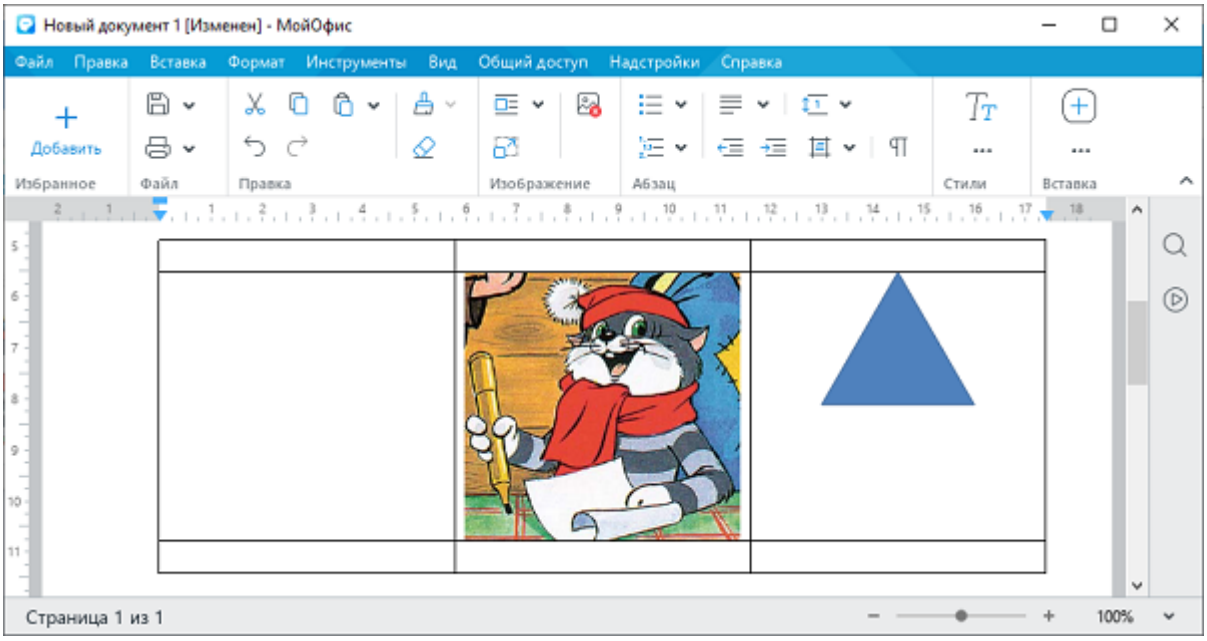


Рисунок 9 – Графические объекты в таблице текстового документа

Вариант 1: перечисление графических объектов в таблице текстового документа

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

Вывод:

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

Вариант 2: перечисление изображений в таблице текстового документа

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

Вывод:

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

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

Вставка изображения в текстовый документ

Вариант 1: вставка изображения в позицию диапазона текстового документа

```
local range = document:getRange()
local imageSize = DocumentAPI.SizeU(50, 50)
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("https://www.images.ru/images/fish.jpg", imageSize)
```

4.2 Работа с табличным документом

4.2.1 Работа с текстом в табличном документе

Вставка текста в табличный документ:

Для вставки текста в ячейку табличного документа используется метод Position::insertText().

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

4.2.2 Копирование ячеек в табличном документе

Для копирования / переноса группы ячеек вместе с их содержимым и свойствами

используются методы CellRange.copyInto() и CellRange.moveInto().

Следующий пример копирует ячейки диапазона "A1:B2" в позицию диапазона "E6:F7":

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

local strTargetRange = "E6:F7"

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

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

```
sourceRange:moveInto(destRange)
```

4.2.3 Диаграммы

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

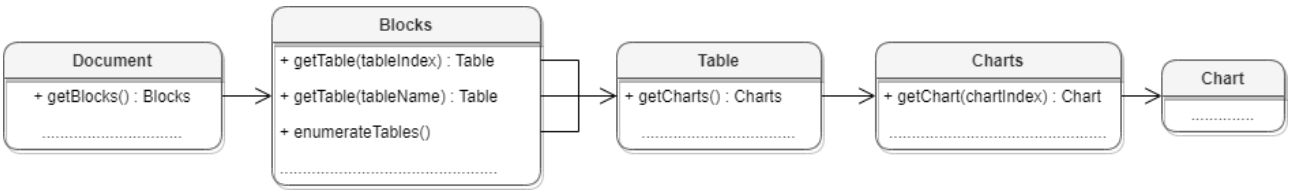


Рисунок 10 – Объектная модель таблиц для работы с диаграммами

Доступ к списку диаграмм производится через таблицу DocumentAPI.Table, соответствующую листу табличного документа.

Пример:

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

Создание и удаление диаграмм в текущей версии не поддерживается.

4.2.4 Работа с графическими объектами в табличном документе

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

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

– Перечисление графических объектов, находящихся в документе, определение их типа и геометрических размеров.
– Вставка изображений в текстовый документ. Место вставки определяется типом `Position`.
– Перемещение графических объектов, изменение их размеров и масштаба.

Перечисление графических объектов в табличном документе

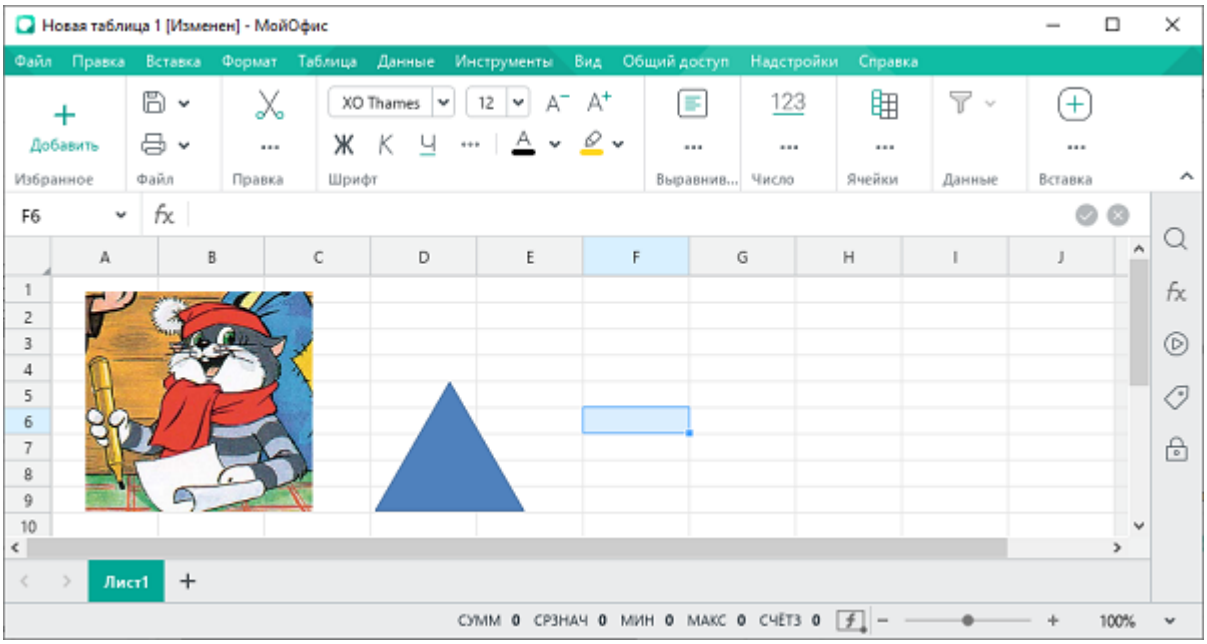


Рисунок 11 – Графические объекты в табличном документе

Вариант 1: перечисление графических объектов в табличном документе

```
local tbl = document:getBlocks():getTable(0)
local mediaObjects = tbl:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
```

```
   image = mediaObject:toImage()
   if image then
      print("Image:", image)
   else
     chart = mediaObject:toChart()
     if chart then
       print("Chart:", chart)
     else
       print("Shape", mediaObject)
     end
   end
end
```

Вывод:

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

Вариант 2: перечисление изображений в табличном документе

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

Вывод:

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

Вставка изображения в табличный документ

В текущей версии не поддерживается.

4.2.5 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 12).

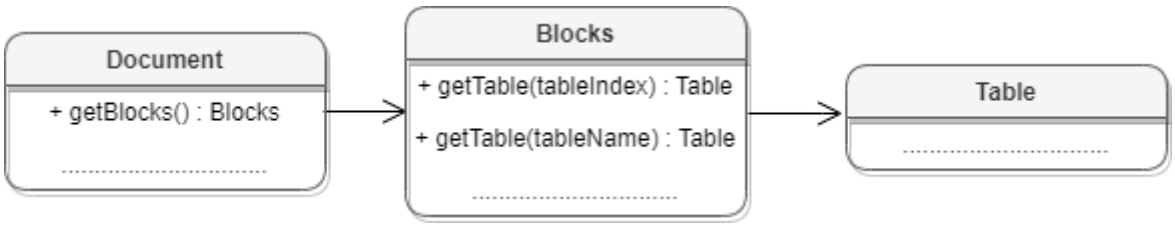


Рисунок 12 – Объектная модель для работы с таблицами

Получение листа табличного документа:

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

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

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

Перечисление страниц табличного документа:

Для перечисления листов табличного документа используется метод Blocks::getTablesEnumerator().

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

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

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

Вставка страницы в табличный документ:

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

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

Активация страницы, получение активной страницы:

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

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

Переименование страницы:

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

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

Скрытие и отображение страниц табличного документа:

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

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

Копирование страницы:

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

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

Удаление страницы:

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

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

4.2.6 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 13.

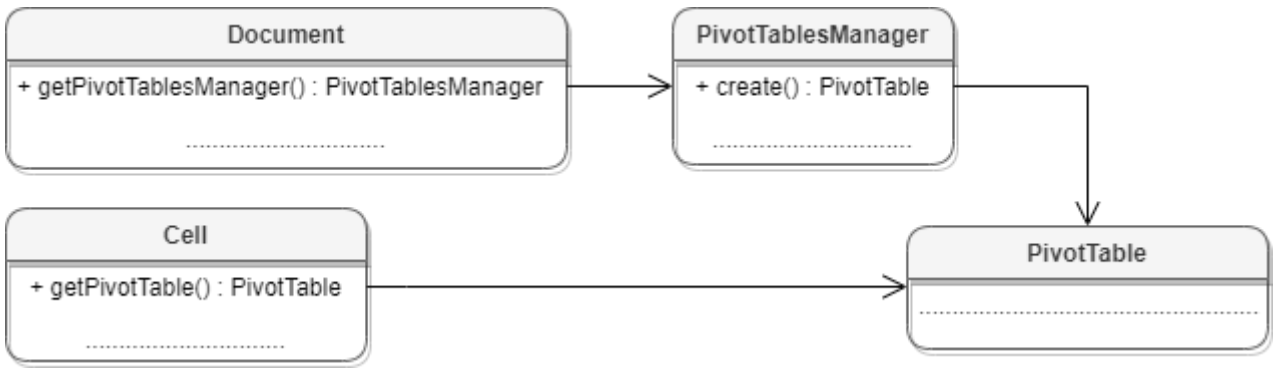


Рисунок 13 – Сводные таблицы

4.2.7 Работа с фильтрами

Работа с фильтрами возможна только в табличном документе. Диаграмма взаимодействия объектов приведена на рисунке 14.

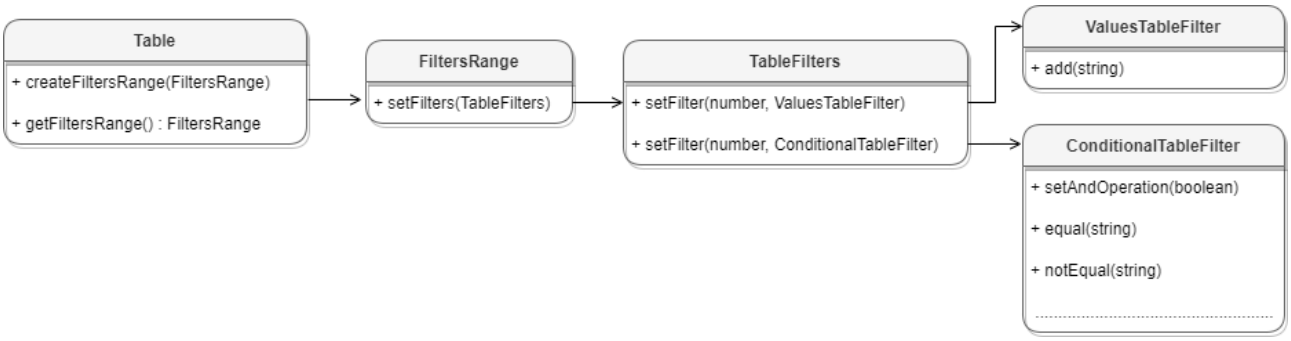


Рисунок 14 – Объектная модель таблиц для работы с фильтрами

Диапазон ячеек для фильтров `FiltersRange` формируется посредством метода `Table:createFiltersRange()`.

Далее создаются фильтры (возможные варианты: `ValuesTableFilter`, `ConditionalTableFilter`).

Фильтры помещаются в структуру `TableFilters`.

Далее фильтры `TableFilters` помещаются в диапазон `FiltersRange` посредством использования метода `FiltersRange:setFilters()`.

Пример работы с фильтрами в табличном документе:

```
local tbl = EditorAPI.getActiveWorksheet()
local range = DocumentAPI.CellRangePosition(1, 1, 8, 2)
local filtersRange = tbl:createFiltersRange(range)

local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")

local songFilter = DocumentAPI.ConditionalTableFilter()
songFilter:setAndOperation(true)
songFilter:notEqual("")
songFilter:notBegins("TODO")

local tableFilters = DocumentAPI.TableFilters()
```

```
tableFilters:setFilter(0, johnPaulFilter)
tableFilters:setFilter(1, songFilter)

filtersRange:setFilters(tableFilters)
```

Пример фильтров, сформированных в результате работы данного примера, приведен на рисунке 15.

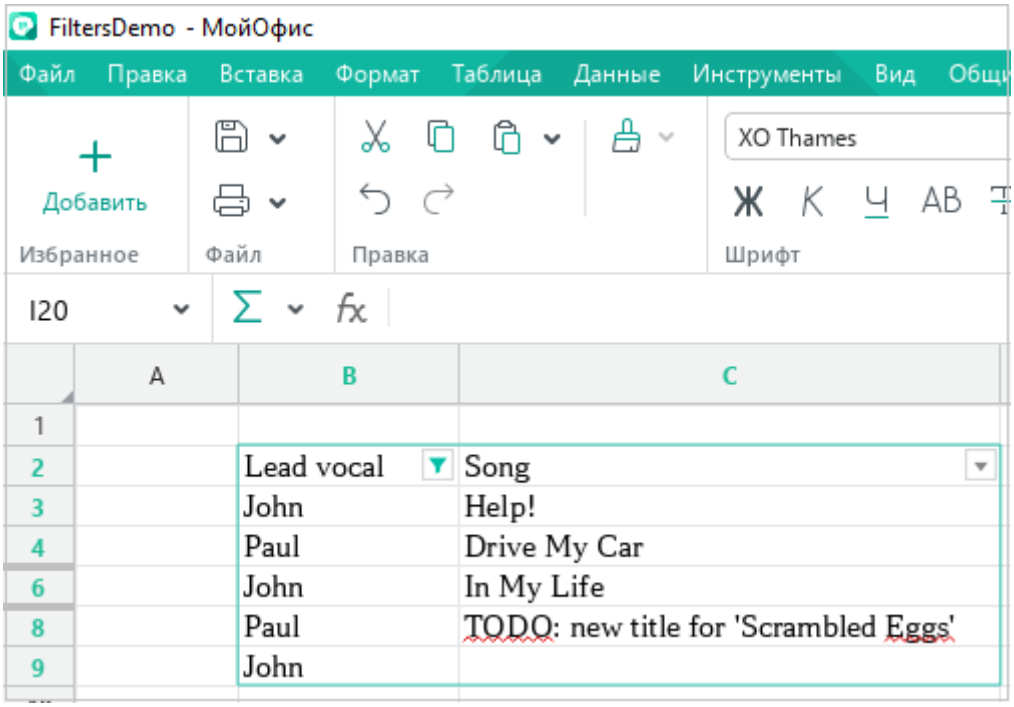


Рисунок 15 – Применение фильтров

4.3 Поиск в документе

Для поиска в текстовом или табличном документе необходимо создать экземпляр класса `Search` посредством вызова `DocumentAPI.createSearch(document)`, затем использовать метод `Search:findText` (см. Рисунок 16).

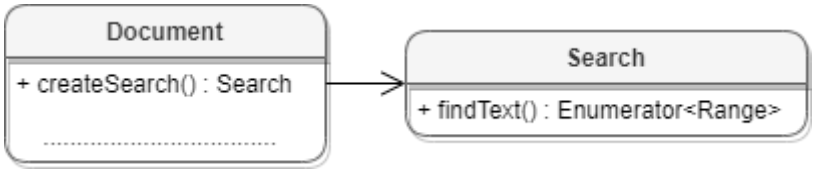


Рисунок 16 – Объектная модель для поиска в документе

Примеры поиска в документе:

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

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

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

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

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

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

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

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

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

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

4.4 Работа с макрокомандами

Таблица `DocumentAPI.Scripts` предоставляет доступ к списку макрокоманд документа. На рисунке 17 изображена объектная модель таблиц, относящихся к работе с макрокомандами.

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

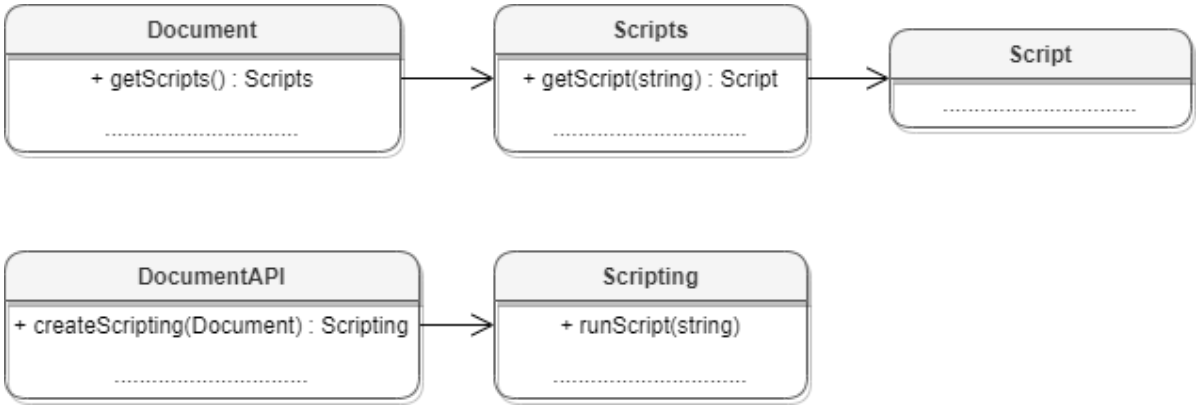


Рисунок 17 – Объектная модель таблиц для работы с макрокомандами

4.5 Работа с именованными диапазонами

Именованный диапазон – это диапазон ячеек или формула, которым присвоено имя. Преимуществом именованного диапазона является его информативность. Именованные диапазоны упрощают работу с ячейками, также их удобно использовать при работе с формулами. На данный момент доступна возможность работы с именованными диапазонами, представляющими собой ссылки на диапазоны ячеек. Доступ к именованным диапазонам осуществляется посредством методов `Document:getNamedExpressions()` и `Table:getNamedExpressions()` (см. Рисунок 18).

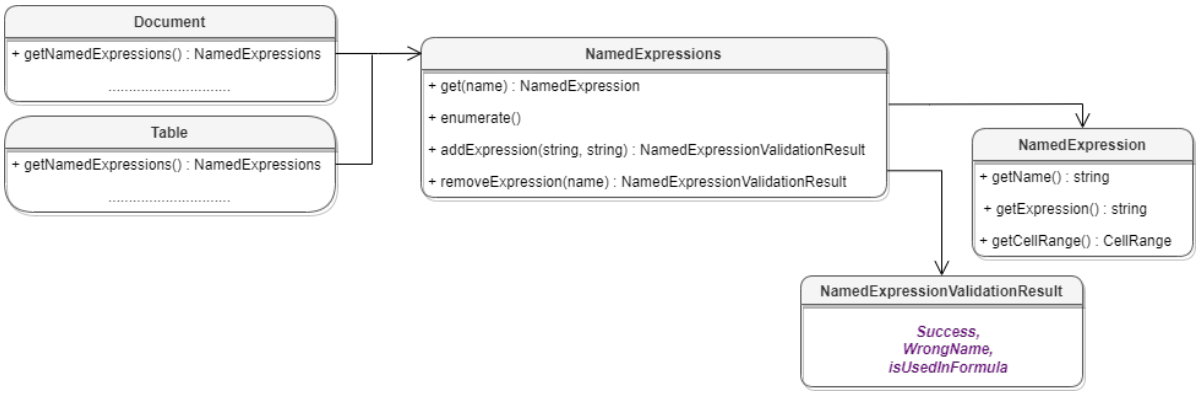


Рисунок 18 – Таблицы для работы с именованными диапазонами

5 Работа со строками и столбцами таблиц

5.1 Группировка строк и колонок таблицы

Следующий набор методов позволяет группировать строки и колонки таблицы: `Table::groupRows()`, `Table::ungroupRows()`, `Table::clearRowGroups()`, `Table::groupColumns()`, `Table::ungroupColumns()`, `Table::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.

5.2 Управление видимостью строк / колонок

Метод `Table:isRowVisible` позволяет определять видимость строки с заданным индексом.

Метод `Table:isColumnVisible` позволяет определять видимость столбца с заданным индексом.

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

Пример для текстового и табличного редактора:

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

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

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

Пример для табличного редактора:

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

```
    local tbl = selection:getTable()

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

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

setSelectionVisible(false)
```

6 Работа с ячейками таблиц

6.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 19):

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

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

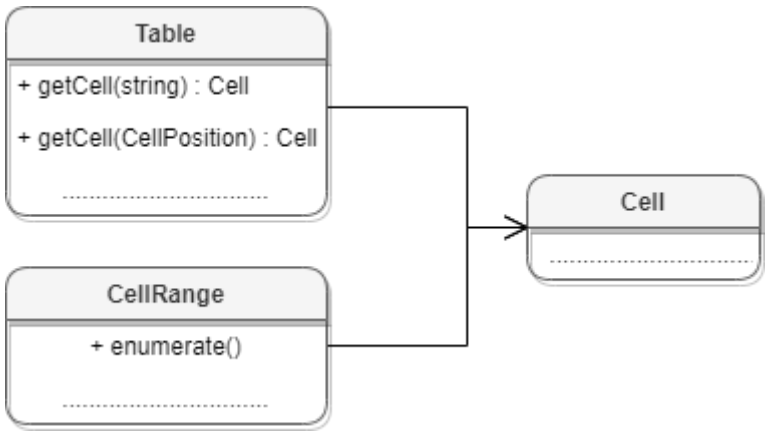


Рисунок 19 – Объектная модель для работы с ячейками таблиц

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

Пример:

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

Второй вариант доступа к ячейке - перечисление диапазона ячеек методом `CellRange.enumerate()`.

Пример:

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

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

Примеры:

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

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

local cell1 = table1:getCell("A1")
local cell2 = table1:getCell("C4")
local cell3 = table1:getCell("E4")

print(cellRange1:containsCell(cell1))
print(cellRange1:containsCell(cell2))
print(cellRange1:containsCell(cell3))

print(cellRange2:containsCell(cell1))
print(cellRange2:containsCell(cell2))
print(cellRange2:containsCell(cell3))
```

Для установки значений ячеек используются методы Cell:setText, Cell:setNumber, Cell:setFormula, Cell:setBool.

Примеры:

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

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

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

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

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

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

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

Пример:

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

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

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

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

Пример:

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

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

Пример:

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

6.2 Форматирование ячеек

При работе с ячейками таблиц можно использовать следующие варианты форматирования:

- форматирование параметров ячейки DocumentAPI.CellProperties, например, цвет фона, угол поворота текста;
- форматирование абзаца ячейки, например, отступы абзаца, межстрочный интервал текста;

– форматирование текста, например, цвет текста, начертание;

– задание параметров границ ячеек.

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

Пример установки и получения свойств параграфа ячейки:

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

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

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

Пример настроек текста ячейки:

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

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

6.3 Форматирование границ ячеек

Для оформления границ ячеек используется таблица DocumentAPI.Borders (см. Рисунок 20). 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.

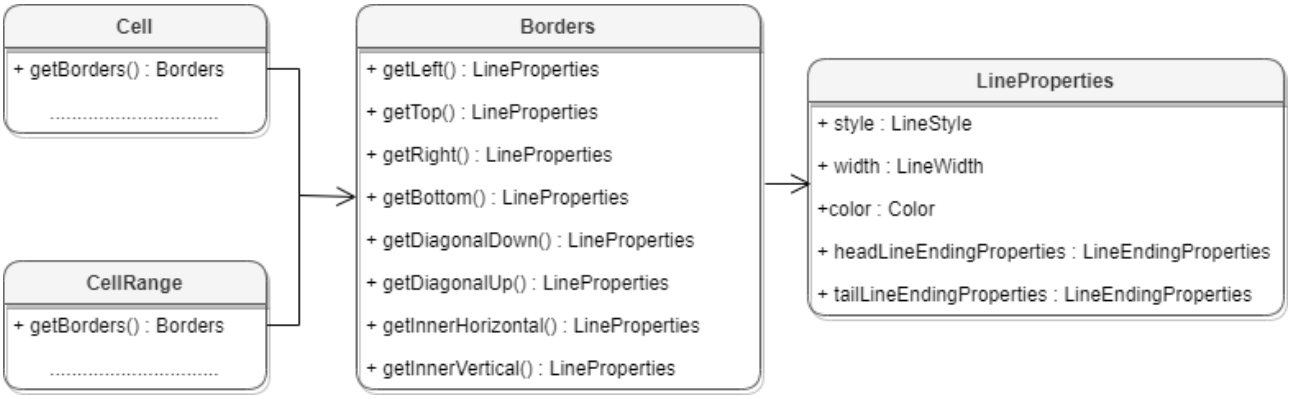


Рисунок 20 – Таблицы для работы с границами ячеек

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

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

Пример настройки границ ячеек:

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

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

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

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

--установка внешних границ
borders:setOuter(lineProp)

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

6.4 Объединение и разделение ячеек таблицы

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

Для объединения нескольких ячеек используйте метод `CellRange.merge()`.

Пример:

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

Допустимо разъединение только тех ячеек, которые были объединены ранее. После завершения операции данные, содержавшиеся 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()
```

7 Справочник таблиц DocumentAPI

7.1 Таблица DocumentAPI.AbsoluteFrame

Таблица `DocumentAPI.AbsoluteFrame` описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 21). Предназначена для получения и изменения свойств позиции медиаобъектов. Используется в табличном документе.

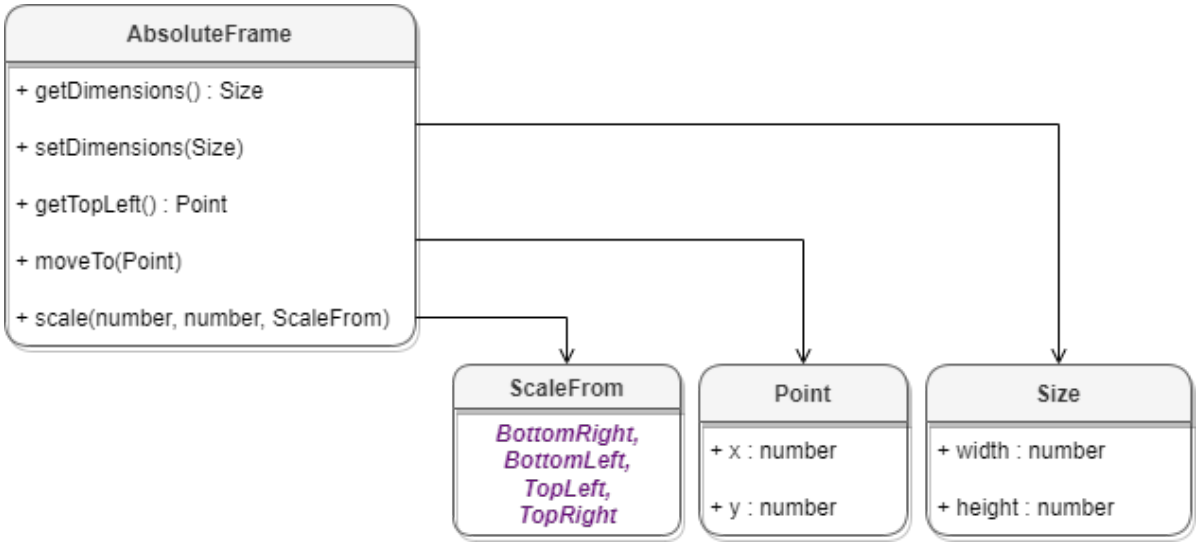


Рисунок 21 – Объектная модель таблицы `DocumentAPI.AbsoluteFrame`

Пример для табличного документа:

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

7.1.1 Метод AbsoluteFrame:moveTo

Метод перемещает объект в заданную позицию, тип аргумента - `DocumentAPI.PointU`.

Пример:

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

```
   absoluteFrame:moveTo(newFramePosition)
end
```

7.1.2 Метод AbsoluteFrame:getTopLeft

Метод возвращает позицию верхней левой точки медиаобъекта, тип - DocumentAPI.PointU.

Пример:

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

7.1.3 Метод AbsoluteFrame:scale

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

Вызов:

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

Параметры:

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

Пример:

```
-- Уменьшение масштаба всех медиаобъектов на 50%
sheet = document:getBlocks():getTable(0)
for mediaObject in sheet:getMediaObjects():enumerate() do
   mediaObject:getFrame():scale(0.5, 0.5, DocumentAPI.ScaleFrom_TopLeft)
end
```

7.1.4 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(DocumentAPI.SizeU(100, 100))
end
```

7.1.5 Метод AbsoluteFrame:getDimensions

Возвращает размеры медиаобъекта, тип - `DocumentAPI.SizeU`.

Пример:

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

7.2 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 2.

Таблица 2 – Описание полей таблицы `DocumentAPI.AccountingCellFormatting`

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

Поле	Описание
DocumentAPI.AccountingCellFormatting.currencySignPlacement	Тип размещения знака валюты CurrencySignPlacement

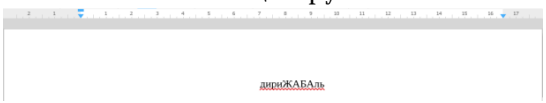
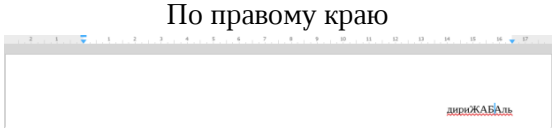
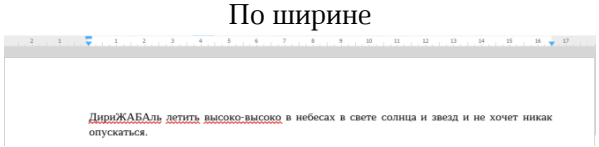
Пример:

```
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())
```

7.3 Таблица DocumentAPI.Alignment

В таблице 3 представлены варианты выравнивания текста по горизонтали в текстовом редакторе или содержимого ячеек в табличном редакторе.

Таблица 3 – Варианты выравнивания по горизонтали

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

Пример для текстового документа:

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

Пример для табличного документа:

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

7.4 Таблица DocumentAPI.Block

Таблица `DocumentAPI.Block` является базовой для всех блоков документа. От нее наследуются таблицы `Paragraph`, `Table`, `Shape`, `Field` (см. Рисунок 22).

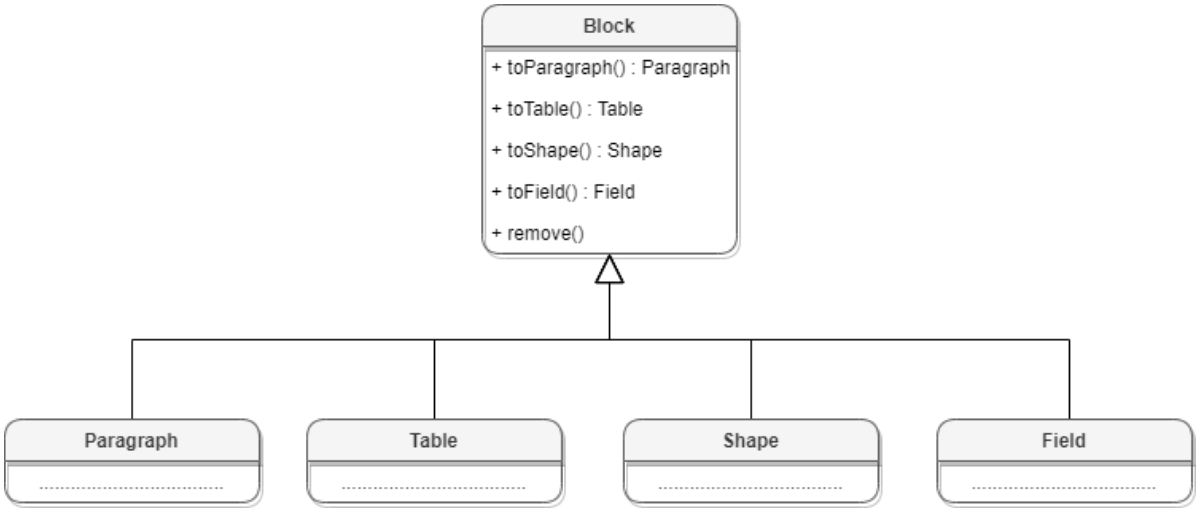


Рисунок 22 – Объектная модель таблицы `DocumentAPI.Block`

7.4.1 Методы toParagraph, toTable, toShape, toField

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

Пример:

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

```
  local para_props = paragraph:getParagraphProperties()
end
```

7.4.2 Метод Block.getRange

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

Пример:

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

7.4.3 Метод Block.remove

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

Пример:

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

7.4.4 Метод Block.getSection

Метод возвращает раздел DocumentAPI.Section, содержащий блок.

Пример:

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

7.5 Таблица DocumentAPI.Blocks

Таблица DocumentAPI.Blocks обеспечивает доступ к блокам DocumentAPI.Block документа или диапазона документа (см. Рисунок 23). Таблица DocumentAPI.Blocks может быть получена вызовом метода Document:getBlocks или HeaderFooter:getBlocks.

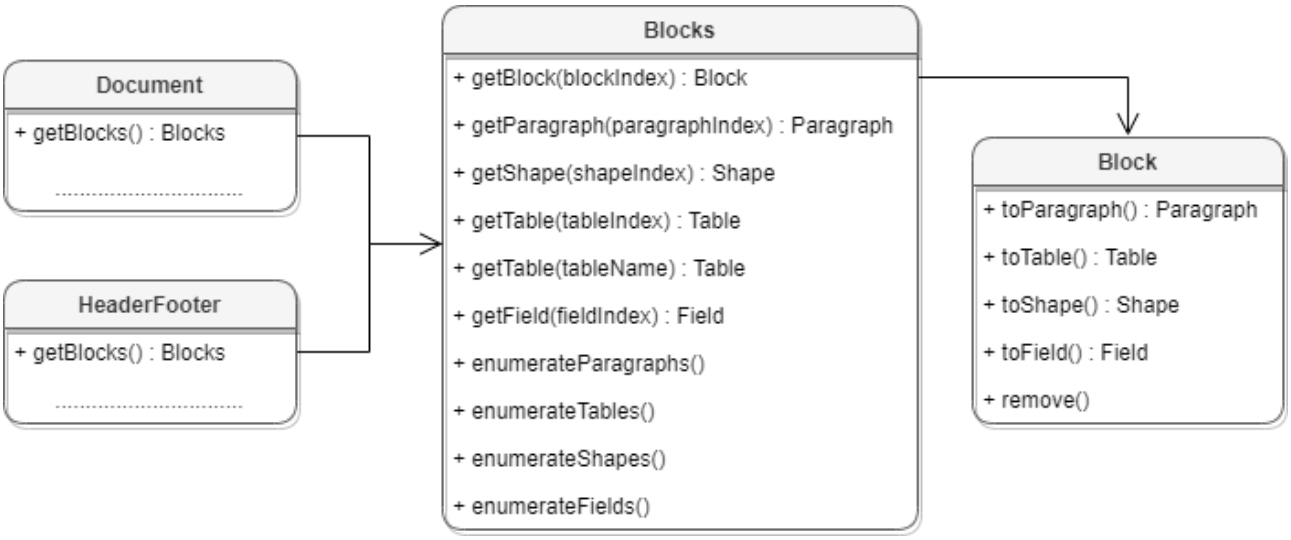


Рисунок 23 – Объектная модель таблицы `DocumentAPI.Blocks`

7.5.1 Метод Blocks:getBlock

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

Пример:

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

7.5.2 Метод Blocks:getParagraph

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

Пример:

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

7.5.3 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")
```

7.5.4 Метод Blocks:getShape

Возвращает фигуру DocumentAPI.Shape по заданному индексу.

Пример:

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

7.5.5 Метод Blocks:getField

Возвращает объект типа DocumentAPI.Field по заданному индексу.

Пример:

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

7.5.6 Метод Blocks:enumerate

Позволяет перечислить объекты типа DocumentAPI.Block.

Пример:

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

7.5.7 Метод Blocks:enumerateParagraphs

Позволяет реализовать перечисление абзацев DocumentAPI.Paragraph.

Пример:

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

7.5.8 Метод Blocks:enumerateTables

Позволяет перечислить объекты типа DocumentAPI.Table.

Пример:

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

7.6 Таблица DocumentAPI.Bookmarks

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

7.6.1 Метод Bookmarks:getBookmarkRange

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

Пример:

```
local bookmarks = document:getBookmarks()
local bookmarkRange = bookmarks:getBookmarkRange("Bookmark")
if (bookmarkRange ~= nil) then
  bookmarkRange:replaceText("Lua")
end
```

7.6.2 Метод Bookmarks:removeBookmark

Удаляет закладку по ее названию.

Пример:

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

7.7 Таблица DocumentAPI.Borders

Таблица `DocumentAPI.Borders` предназначена для оформления границ отдельной ячейки таблицы (см. таблицу 4). Параметры линии, такие как тип линии, ее ширина и цвет, задаются с помощью таблицы `DocumentAPI.LineProperties`.

Таблица 4 – Описание методов таблицы `DocumentAPI.Borders`

Метод	Описание
`Borders:setLeft`	Установка левой границы ячейки
`Borders:setRight`	Установка правой границы ячейки
`Borders:setTop`	Установка верхней границы ячейки
`Borders:setBottom`	Установка нижней границы ячейки
`Borders:setDiagonalDown`	Установка диагональной линии 
`Borders:setDiagonalUp`	Установка диагональной линии

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

Пример:

```
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)
```

7.8 Таблица DocumentAPI.CaseSensitive

Таблица `DocumentAPI.CaseSensitive` используется для настройки параметров поиска в текстовом или табличном документе (см. метод `Search:findText`). Описание полей таблицы представлено в таблице 5.

Таблица 5 – Описание полей таблицы `DocumentAPI.CaseSensitive`

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

Пример:

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

7.9 Таблица DocumentAPI.Cell

Таблица `DocumentAPI.Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 24).

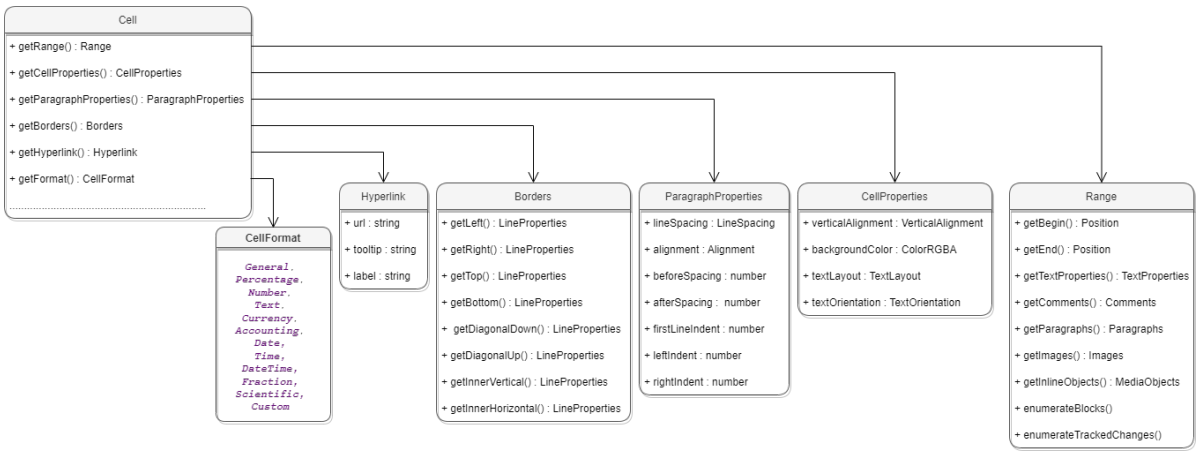


Рисунок 24 – Объектная модель ячейки таблиц

7.9.1 Метод Cell:getBorders

Позволяет получить границы ячейки (тип `DocumentAPI.Borders`). Примеры использования приведены в разделе `DocumentAPI.Borders`.

Пример:

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

7.9.2 Метод Cell:getCellProperties

Позволяет получить свойства DocumentAPI.CellProperties ячейки.

Пример:

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

7.9.3 Метод Cell:getCustomFormat

Возвращает строку формата ячейки.

Пример:

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

7.9.4 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
```

7.9.5 Метод Cell:getFormattedValue

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

Пример:

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

7.9.6 Метод Cell:getFormulaAsString

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

Пример:

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

7.9.7 Метод Cell:getHyperlink

Возвращает первый объект в ячейке типа DocumentAPI.Hyperlink.

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

7.9.8 Метод Cell:getParagraphProperties

Возвращает свойства абзаца DocumentAPI.ParagraphProperties, находящегося в ячейке.

Пример:

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

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

7.9.9 Метод Cell:getPivotTable

Возвращает сводную таблицу DocumentAPI.PivotTable, относящуюся к ячейке.

Пример:

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

7.9.10 Метод Cell:getRange

Метод возвращает объект DocumentAPI.Range для управления содержимым ячейки.

Пример:

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

7.9.11 Метод Cell:getRawValue

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

Пример:

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

7.9.12 Метод Cell:setContent

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

Пример:

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

7.9.13 Метод Cell:setBool

Устанавливает для ячейки значение логического типа.

Пример:

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

7.9.14 Метод Cell:setBorders

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

7.9.15 Метод Cell:setCellProperties

Позволяет установить свойства ячейки DocumentAPI.CellProperties.

Пример:

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

7.9.16 Метод Cell:setCustomFormat

Устанавливает формат ячейки.

Пример:

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

7.9.17 Метод Cell:setFormula

Метод позволяет вставить формулу в ячейку табличного документа.

Пример:

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

7.9.18 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 a1CellFormatting = DocumentAPI.PercentageCellFormatting()
a1CellFormatting.decimalPlaces = 1
cellA1:setFormat(a1CellFormatting)
print("Формат Процентный: ", cellA1:getFormat()) -- 1
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 230,0%
```

```
-- Формат: Числовой
a1CellFormatting = DocumentAPI.NumberCellFormatting()
a1CellFormatting.decimalPlaces = 2
cellA1:setFormat(a1CellFormatting)
print("Формат Числовой: ", cellA1:getFormat()) -- 2
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30

-- Формат: Денежный
a1CellFormatting = DocumentAPI.CurrencyCellFormatting()
a1CellFormatting.symbol = '$'
cellA1:setFormat(a1CellFormatting)
print("Формат Денежный: ", cellA1:getFormat()) -- 4
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30$

-- Формат: Финансовый
a1CellFormatting = DocumentAPI.AccountingCellFormatting()
a1CellFormatting.symbol = '₽'
cellA1:setFormat(a1CellFormatting)
print("Формат Финансовый: ", cellA1:getFormat()) -- 5
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30₽

-- Формат: Дата / Время
a1CellFormatting = DocumentAPI.DateTimeCellFormatting()
a1CellFormatting.dateListID = DocumentAPI.DatePatterns_FullDate
a1CellFormatting.timeListID = DocumentAPI.TimePatterns_ShortTime
cellA1:setFormat(a1CellFormatting)
print("Формат Дата / Время: ", cellA1:getFormat()) -- 8
print("Значение ячейки: ", cellA1:getRange():extractText()) -- понедельник, 1
января 1900 г. 7:12

-- Формат: Экспоненциальный
a1CellFormatting = DocumentAPI.FractionCellFormatting()
a1CellFormatting.minNumeratorDigits = 2
cellA1:setFormat(a1CellFormatting)
print("Формат Экспоненциальный: ", cellA1:getFormat()) -- 9
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2 2/7

-- Формат: Научный
a1CellFormatting = DocumentAPI.ScientificCellFormatting()
a1CellFormatting.decimalPlaces = 5
```

```
cellA1:setFormat(a1CellFormatting)
print("Формат Научный: ", cellA1:getFormat()) -- 10
print("Значение ячейки: ", cellA1:getRange():extractText()) -- 2,30000E+00
```

7.9.19 Метод Cell:setFormattedValue

Анализирует переданное значение и автоматически устанавливает формат ячейки и ее значение. В случае, если распознать тип переданного значения не удается, то для ячейки устанавливается формат `DocumentAPI.CellFormat_Text`.

Список поддерживаемых форматов см. в разделе `DocumentAPI.CellFormat`. Пример использования метода см. в разделе Доступ к ячейкам.

7.9.20 Метод Cell:setNumber

Устанавливает для ячейки значение числового типа.

Пример:

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

7.9.21 Метод Cell:setParagraphProperties

Устанавливает свойства абзаца `DocumentAPI.ParagraphProperties`, находящегося в ячейке.

Пример:

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

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

7.9.22 Метод Cell:setText

Устанавливает для ячейки значение строкового типа.

Пример:

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

```
   trackingChanges = "Enabled"
end
cell:setText(trackingChanges)
```

7.9.23 Метод Cell::isPivotTableRoot

Метод позволяет определить является ли ячейка основанием сводной таблицы.

Пример:

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

7.9.24 Метод Cell:unmerge

Разъединяет несколько ячеек, которые были объединены ранее. Примеры объединения и разъединения ячеек см. в разделе Объединение и разделение ячеек таблицы.

Пример:

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

7.10 Таблица DocumentAPI.CellFormat

По умолчанию при создании документа всем ячейкам присваивается формат «Общий». Полный список форматов представлен в таблице 6.

Таблица 6 – 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

Наименование константы	Описание
	формата «Процентный» введенное число умножается на 100 и обозначается знаком «%».
`DocumentAPI.CellFormat_Number`	Формат ячейки «Числовой». Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя.
`DocumentAPI.CellFormat_Text`	Формат ячейки «Текстовый».
`DocumentAPI.CellFormat_Currency`	Формат ячейки «Денежный». Этот формат используется для представления чисел со знаком или кодом валюты.
`DocumentAPI.CellFormat_Accounting`	Формат ячейки «Финансовый». Этот формат применяется для чисел, используемых в бухгалтерских документах. В формате «Финансовый» введенное число автоматически дополняется названием валюты, которая соответствует настройкам системы компьютера. Отрицательные числа в формате «Финансовый» заключаются в круглые скобки в ячейке, а в строке формул остаются в том виде, в котором они были введены.
`DocumentAPI.CellFormat_Date`	Формат ячейки «Дата». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ.
`DocumentAPI.CellFormat_Time`	Формат ячейки «Время». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.
`DocumentAPI.CellFormat_DateTime`	Формат ячейки «Дата + Время»
`DocumentAPI.CellFormat_Fraction`	Формат ячейки «Дробный». Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель).

Наименование константы	Описание
`DocumentAPI.CellFormat_Scientific`	Формат ячейки «Экспоненциальный». Экспоненциальный (или научный) формат используется для представления больших чисел в короткой форме. Все введенные числа длиной более 12 символов автоматически переводятся в этот формат. В ячейке число в формате «Экспоненциальный» представлено следующим образом: – целая часть, всегда состоящая из одной цифры; – разделитель целой и дробной части; – дробная часть, по умолчанию состоящая из двух цифр; – показатель степени числа 10 в виде **E<знак показателя степени> <показатель степени>**.
`DocumentAPI.CellFormat_Custom`	Пользовательский формат

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

Примеры использования:

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

Результат выполнения данного примера приведен на рисунке 25.

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

Рисунок 25 – Результат установки формата

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

7.11 Таблица DocumentAPI.CellPosition

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

Позиция ячейки A1 имеет координаты (0, 0).

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

Примеры:

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

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

7.11.1 Поле CellPosition.column

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

Пример:

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

7.11.2 Поле CellPosition.row

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

Пример:

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

7.11.3 Метод CellPosition:toString

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

Пример:

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

7.11.4 Метод CellPosition:__eq

Метод используется для определения эквивалентности двух объектов `CellPosition`.

Пример:

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

7.12 Таблица DocumentAPI.CellProperties

Таблица `DocumentAPI.CellProperties` предназначена для форматирования содержимого в ячейках таблицы. Описание полей таблицы `DocumentAPI.CellProperties` представлено в таблице 7.

Для задания свойств ячейки используется метод `Cell.setCellProperties()`. Для получения свойств ячейки используется метод `Cell.getCellProperties()`. Иерархия таблиц и полей `DocumentAPI.CellProperties` отображена на рисунке 26.

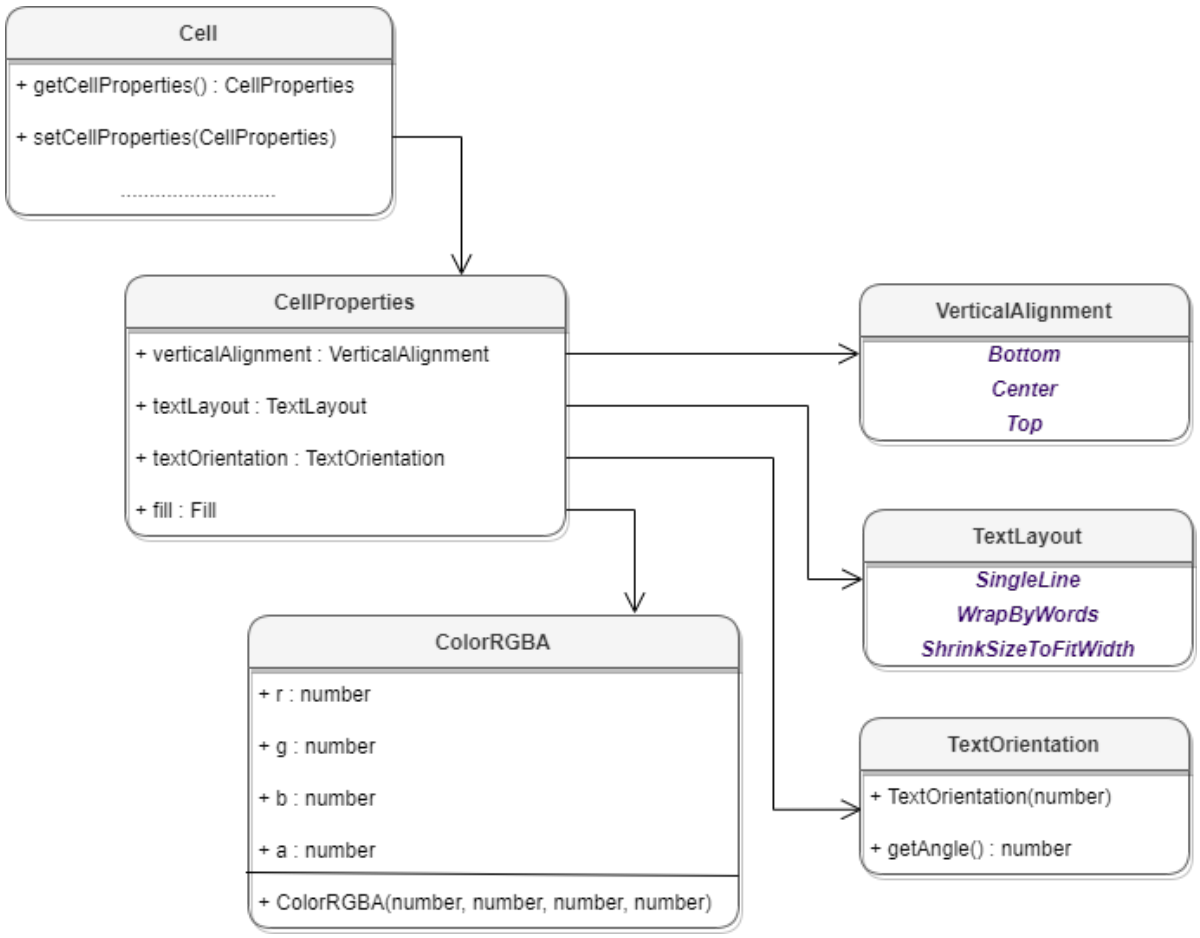


Рисунок 26 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 7 – Описание полей таблицы `DocumentAPI.CellProperties`

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

Пример:

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

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

cell:setCellProperties(props)
```

7.12.1 Метод CellProperties:__eq

Метод используется для определения эквивалентности значений двух объектов `CellProperties`.

Пример:

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

7.13 Таблица DocumentAPI.CellRange

Таблица `DocumentAPI.CellRange` описывает диапазон ячеек таблицы.

Пример:

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

7.13.1 Метод CellRange::autoFill

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

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

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

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

Пример:

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

7.13.2 Метод CellRange:containsCell

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

Примеры:

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

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

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

7.13.3 Метод CellRange:copyInto

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

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

Пример (только для табличного документа):

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

При копировании ячеек в качестве новой позиции достаточно указать верхнюю левую ячейку нового диапазона, однако, если необходимо продублировать исходный блок ячеек, в качестве параметра следует использовать диапазон, превышающий размеры исходного диапазона, но кратный его размерам. Например, при копировании диапазона `"A1:B2"` (размер 2x2) в диапазон `"B5:E6"` (размер 2x4) блок исходных ячеек продублируется два раза (см. Рисунок 27).

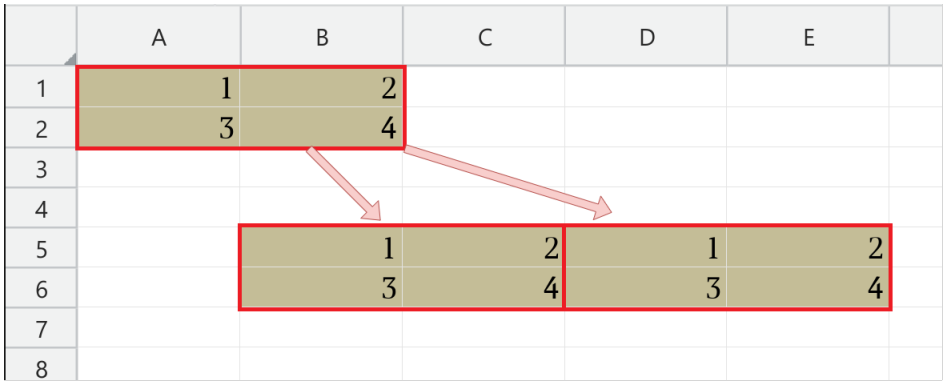


Рисунок 27 – Копирование ячеек табличного документа

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

7.13.4 Метод CellRange:enumerate

Метод возвращает коллекцию ячеек в диапазоне.

Пример:

```
-- Печать значений ячеек в диапазоне B3:C4
local tbl = document:getBlocks():getTable(0)
```

```
local rng = tbl:getCellRange("B3:C4")
for cell in rng:enumerate() do
    print(cell:getFormattedValue())
end
```

7.13.5 Метод CellRange:getBeginRow

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

Пример:

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

7.13.6 Метод CellRange:getBeginColumn

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

Пример:

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

7.13.7 Метод CellRange:getCellProperties

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

Пример:

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

7.13.8 Метод CellRange:getLastRow

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

Пример:

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

7.13.9 Метод CellRange:getLastColumn

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

Пример:

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

7.13.10 Метод CellRange:getTable

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

Пример:

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

7.13.11 Метод CellRange:insertCurrentDateTime

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

Пример:

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

7.13.12 Метод CellRange:merge

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

помещается в объединенной ячейке. Примеры объединения и разъединения ячеек см. в разделе Объединение и разделение ячеек таблицы.

Пример:

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

7.13.13 Метод CellRange:moveInto

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

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

Пример (только для табличного документа):

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

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

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

7.13.14 Метод CellRange:setBorders

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

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("C3")
--
line_prop = DocumentAPI.LineProperties()
line_prop.style = DocumentAPI.LineStyle_Dash
line_prop.width = 1.5
line_prop.color = DocumentAPI.Color(DocumentAPI.ColorRGBA(255, 0, 0, 255))
--
```

```
newBorders = DocumentAPI.Borders()
newBorders = newBorders:setLeft(line_prop)
newBorders = newBorders:setRight(line_prop)
newBorders = newBorders:setTop(line_prop)
newBorders = newBorders:setBottom(line_prop)
--
local borders = cell:setBorders(newBorders)
```

7.13.15 Метод CellRange:setCellProperties

Метод предназначен для установки свойств DocumentAPI.CellProperties для всех ячеек диапазона.

Пример:

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

7.14 Таблица DocumentAPI.CellRangePosition

Таблица DocumentAPI.CellRangePosition представляет положение диапазона ячеек в таблице.

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

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

Где:

– leftTopPosition – позиция левой верхней ячейки диапазона;

– rightBottomPosition – позиция правой нижней ячейки диапазона.

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

Где:

– leftColumn – индекс левого столбца диапазона, индексирование происходит с нуля;

– topRow – индекс верхней строки диапазона (индексирование производится с нуля);

– `rightColumn` – индекс правого столбца диапазона (индексирование производится с нуля);
– `bottomRow` – индекс нижней строки диапазона (индексирование производится с нуля).

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

Описание полей таблицы `DocumentAPI.CellRangePosition` представлено в таблице 8.

Таблица 8 – Поля таблицы `DocumentAPI.CellRangePosition`

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

Примеры:

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

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

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

```
print("bottomRight=", cellRangePosition.bottomRight.row,
cellRangePosition.bottomRight.column)
```

7.14.1 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)]
```

7.14.2 Метод CellRangePosition:__eq

Метод используется для определения эквивалентности двух объектов `CellRangePosition`.

Пример:

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

7.15 Таблица DocumentAPI.Charts

Таблица `DocumentAPI.Charts` обеспечивает доступ к списку диаграмм (см. Рисунок 28) табличного документа. Доступ к списку диаграмм осуществляется с помощью метода `Table.getCharts()`.

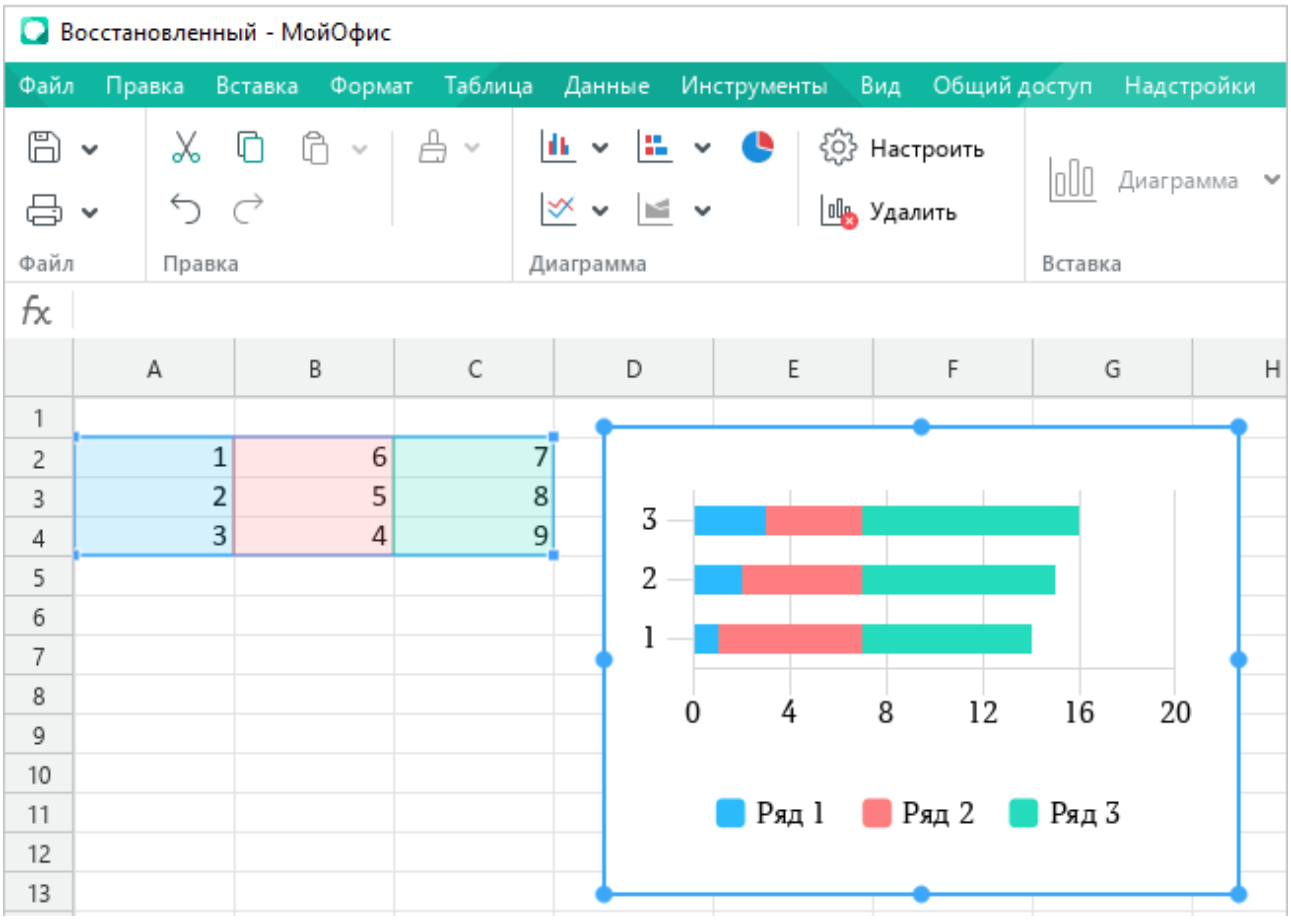


Рисунок 28 – Пример отображения диаграммы в МойОфис Таблица.

7.15.1 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())
```

7.15.2 Метод Charts:getChart

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

Пример:

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

7.15.3 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))
```

7.16 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 рисунке 29.

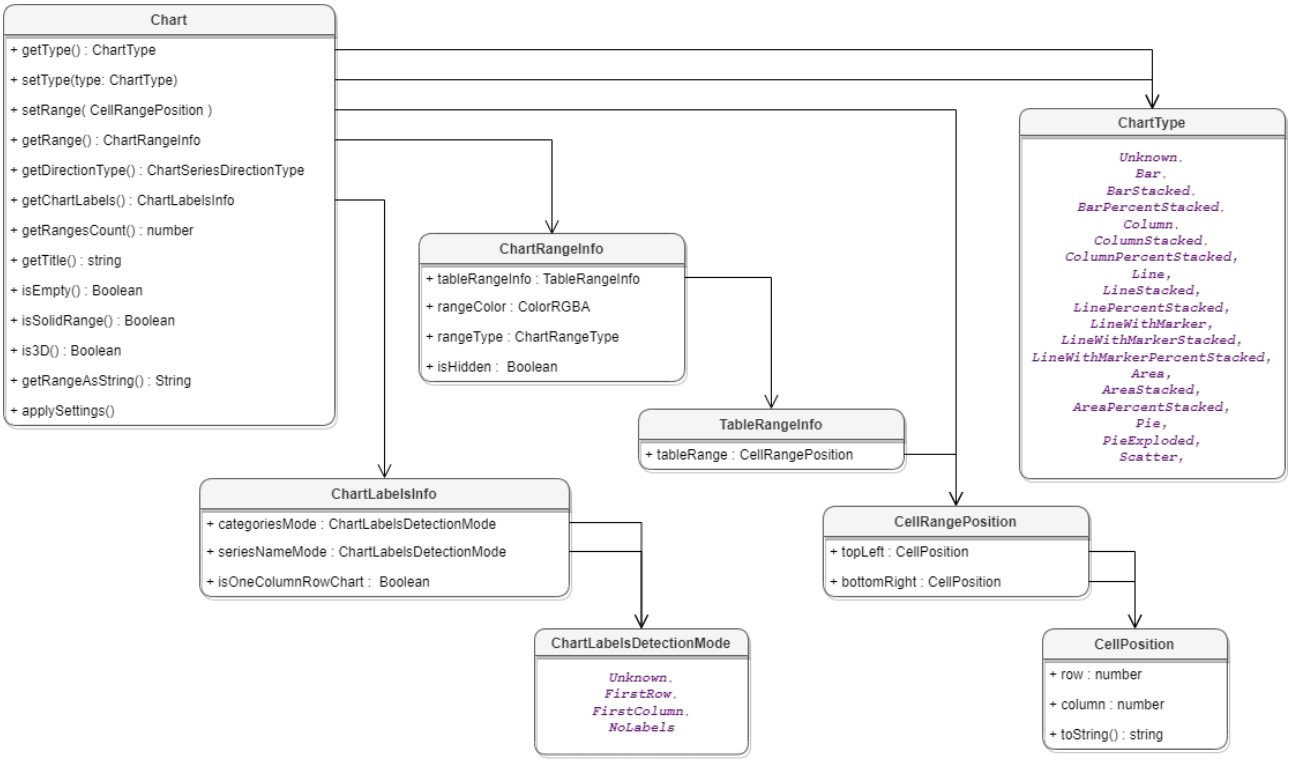


Рисунок 29 – Объектная модель таблицы `DocumentAPI.Chart`

7.16.1 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())
```

7.16.2 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())
```

7.16.3 Метод Chart:getRangesCount

Метод возвращает количество серий диаграммы.

Пример:

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

7.16.4 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)
```

7.16.5 Метод Chart:getTitle

Метод возвращает заголовок диаграммы.

Пример:

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

7.16.6 Метод Chart:setRange

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

Пример:

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

7.16.7 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.

7.16.8 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())
```

7.16.9 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())
```

7.16.10 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())
```

7.16.11 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())
```

7.16.12 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)
```

7.16.13 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())
```

7.16.14 Метод Chart:applySettings

Метод позволяет обновить параметры текущей выбранной диаграммы.

Вызов:

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

Параметры:

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

– `directionType` – направление серий DocumentAPI.ChartSeriesDirectionType;

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

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

Пример:

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

local cellRange = tbl:getCellRange("B3:C4")
local directionType = DocumentAPI.ChartSeriesDirectionType_ByRow
local title = 'Title'
local chartLabelsInfo =
DocumentAPI.ChartLabelsInfo(DocumentAPI.ChartLabelsDetectionMode_FirstRow,
DocumentAPI.ChartLabelsDetectionMode_FirstRow, false)

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

7.17 Таблица DocumentAPI.ChartLabelsDetectionMode

Таблица `DocumentAPI.ChartLabelsDetectionMode` описывает режимы автоматического определения меток диаграмм. Описание полей таблицы представлено в таблице 9.

Таблица 9 – Описание полей таблицы `DocumentAPI.ChartLabelsDetectionMode`

Поле	Описание
`DocumentAPI.ChartLabelsDetectionMode_Unknown`	Неопределенный тип
`DocumentAPI.ChartLabelsDetectionMode_FirstRow`	Метка на первой строке
`DocumentAPI.ChartLabelsDetectionMode_FirstColumn`	Метка на первой колонке
`DocumentAPI.ChartLabelsDetectionMode_NoLabels`	Не отрисовывать метки

Пример:

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

7.18 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 10.

Таблица 10 – Описание полей таблицы `DocumentAPI.ChartLabelsInfo`

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

Пример:

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

7.19 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 11.

Таблица 11 – Описание полей таблицы `DocumentAPI.ChartRangeInfo`

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

Пример:

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

7.20 Таблица DocumentAPI.ChartRangeType

Таблица `DocumentAPI.ChartRangeType` описывает тип диапазона исходных данных диаграммы. Описание полей таблицы представлено в таблице 12.

Таблица 12 – Описание полей таблицы `DocumentAPI.ChartRangeType`

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

Пример:

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

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

7.21 Таблица DocumentAPI.ChartSeriesDirectionType

Таблица `DocumentAPI.ChartSeriesDirectionType` описывает направление серий

диаграмм. Описание полей таблицы представлено в таблице 13.

Таблица 13 – Описание полей таблицы `DocumentAPI.ChartSeriesDirectionType`

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

Пример:

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

7.22 Таблица DocumentAPI.ChartType

Таблица `DocumentAPI.ChartType` описывает все поддерживаемые типы диаграмм. Описание полей таблицы представлено в таблице 14.

Таблица 14 – Описание полей таблицы `DocumentAPI.ChartType`

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

Поле	Описание
DocumentAPI.ChartType_LineWithMarkerPercentStacked	Нормированный график с накоплением и маркерами
DocumentAPI.ChartType_Area	Стандартная диаграмма с областями
DocumentAPI.ChartType_AreaStacked	Диаграмма с областями с накоплением
DocumentAPI.ChartType_AreaPercentStacked	Нормированная диаграмма с областями с накоплением
DocumentAPI.ChartType_Pie	Круговая диаграмма
DocumentAPI.ChartType_PieExploded	Круговая диаграмма с отделенными секторами
DocumentAPI.ChartType_Scatter	Диаграмма рассеяния

Пример:

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

7.23 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)
```

7.23.1 Метод Color:getRGBAColor

Метод возвращает цвет `DocumentAPI.ColorRGBA`.

Пример:

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

7.23.2 Метод Color:getThemeColorID

Метод возвращает цвет идентификатора темы DocumentAPI.ThemeColorID.

Пример:

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

7.23.3 Метод Color:setTransforms

Метод устанавливает трансформацию цвета ColorTransforms.

Пример:

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

7.23.4 Метод Color:getTransforms

Метод возвращает текущую трансформацию цвета ColorTransforms.

Пример:

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

7.23.5 Метод Color:__eq

Метод используется для определения эквивалентности двух значений цвета.

Пример:

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

7.24 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 15.

Таблица 15 – Описание таблицы `DocumentAPI.ColorRGBA`

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

Примеры использования:

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

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

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

7.24.1 Метод ColorRGBA:__eq

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

Пример:

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

7.25 Таблица DocumentAPI.ColorTransforms

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

Пример:

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

7.25.1 Метод ColorTransforms:apply

Метод устанавливает цвет трансформации `ColorRGBA`.

Пример:

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

7.26 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`.

7.26.1 Метод Comment:getRange

Метод возвращает диапазон документа `DocumentAPI.Range`, которому соотвествует комментарий.

Пример:

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

7.26.2 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
```

7.26.3 Метод Comment:getInfo

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

Пример:

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

7.26.4 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
```

7.26.5 Метод Comment:getReplies

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

Пример:

```
local commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    local commentReplies = comment:getReplies()
    for reply in commentReplies:enumerate() do
```

```
        local name = reply.author.name
        print("Ответ на комментарий " .. name .. ": ", reply:getText())
    end
end
```

7.27 Таблица DocumentAPI.Comments

Таблица `DocumentAPI.Comments` содержит коллекцию комментариев диапазона (см. Рисунок 30).

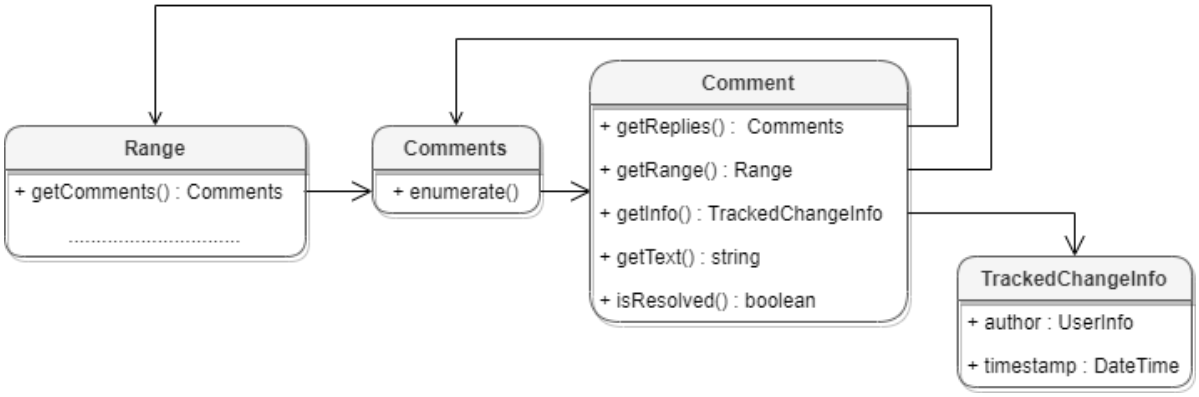


Рисунок 30 – Объектная модель таблиц для работы с комментариями

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

Пример:

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

7.27.1 Метод Comments:enumerate

Метод возвращает коллекцию комментариев всего документа.

Пример:

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

7.28 Таблица DocumentAPI.ConditionalTableFilter

Таблица `ConditionalTableFilter` реализует фильтр, содержащий предикат(ы) для фильтрации строк. Согласно схеме XML, можно использовать одно или два условия, которые объединяются с помощью логической операции AND или OR. На самом деле поддерживается больше критериев, но рекомендуется использовать только один или два. Если не было добавлено ни одного унарного условия, этот фильтр очищает любой другой фильтр, примененный к определенному столбцу. Этот фильтр сохраняется в документе, но редактор не полностью его поддерживает. Фильтр может быть загружен и применен редактором, но если документ изменяется, фильтр будет изменен на тип `ValuesTableFilter`. Если этот фильтр применяется через API, и документ не изменяется после применения фильтра, он будет сохранен как `ConditionalTableFilter`.

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

```
ConditionalTableFilter()
```

Конструктор с параметром, задающим логическую операцию фильтра

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

Параметр:

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

Конструктор копирования:

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

Пример использования приведен в разделе Работа с фильтрами.

7.28.1 Метод ConditionalTableFilter:setAndOperation

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

Пример:

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

7.28.2 Методы добавления условий

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

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

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

Пример:

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

Пример использования приведен в разделе Работа с фильтрами.

7.29 Таблица DocumentAPI.CurrencyCellFormatting

Таблица содержит параметры для денежного формата ячеек таблицы. Описание полей таблицы `DocumentAPI.CurrencyCellFormatting` представлено в таблице 16.

Таблица 16 – Описание полей таблицы `DocumentAPI.CurrencyCellFormatting`

Поле	Описание
`DocumentAPI.CurrencyCellFormatting.decimalPlaces`	Количество десятичных позиций

Поле	Описание
`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())
```

7.30 Таблица DocumentAPI.CurrencySignPlacement

Варианты размещения знака валюты представлены в таблице 17. Данный тип используется в поле `currencyFormat` таблицы `DocumentAPI.LocaleInfo`, а также в поле

`currencySignPlacement` таблицы `DocumentAPI.CurrencyCellFormatting` (см.пример в ее описании).

Таблица 17 – Описание полей таблицы `DocumentAPI.CurrencySignPlacement`

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

7.31 Таблица DocumentAPI.document

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

Пример:

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

7.31.1 Метод document:getAbsoluteFilePath

Метод возвращает строку, содержащую абсолютный путь к текущему документу. Получаемый путь имеет ОС - зависимый формат (например, содержит символы "/" для Unix и "\" для Windows).

Пример:

```
local documentPath = document:getAbsoluteFilePath()
print(documentPath)
```

Ограничения:

- Если документ был создан, но не сохранен, данный метод вернет пустую строку;
- Абсолютный путь может быть получен только для локальных файлов и не будет доступен для получения пути хранения облачного документа;
- В текущей реализации отсутствует возможность полноценного использования метода при совместном редактировании.

7.31.2 Метод document:getBlocks

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

Пример:

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

7.31.3 Метод document:getBookmarks

Метод предоставляет доступ к таблице закладок `DocumentAPI.Bookmarks`. Используется только в текстовых документах.

Пример:

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

7.31.4 Метод document:getScripts

Метод предоставляет доступ к таблице макрокоманд `DocumentAPI.Scripts`, содержащихся в документе.

Пример:

```
local scripts = document:getScripts()
for script in document:getScripts():enumerate() do
    print(script:getName())
end
```

7.31.5 Метод document:getRange

Метод предоставляет доступ ко всему диапазону `DocumentAPI.Range` документа.

Пример:

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

7.31.6 Метод document:isChangesTrackingEnabled

Метод возвращает текущее состояние отслеживания изменений в документе (`true` - включены). Используется только в текстовых документах.

Пример:

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

```
end
print(trackingChanges)
```

7.31.7 Метод document:setChangesTrackingEnabled

Метод управляет состоянием отслеживания изменений в документе (включены или выключены). Используется только в текстовых документах.

Пример:

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

7.31.8 Метод document:getComments

Метод обеспечивает доступ к комментариям документа, возвращает таблицу DocumentAPI.Comments. Используется только в текстовых документах.

Пример:

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

7.31.9 Метод document:setPageProperties

Метод устанавливает свойство DocumentAPI.PageProperties в документе.

Пример:

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

7.31.10 Метод document:setFormulaType

Метод устанавливает поддерживаемую адресацию ячеек `DocumentAPI.FormulaType` документа.

Пример:

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

7.31.11 Метод document:getFormulaType

Метод возвращает поддерживаемую адресацию ячеек `DocumentAPI.FormulaType` документа.

Пример:

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

7.31.12 Метод document:setPageOrientation

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

Пример:

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

7.31.13 Метод document:enumerateSections

Возвращает таблицу объектов типа `DocumentAPI.Section`.

Пример:

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

7.31.14 Метод document:getSections

Возвращает таблицу объектов типа `DocumentAPI.Sections`.

Пример:

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

```
    print(properties.height)
end
```

7.31.15 Метод document:setMirroredMarginsEnabled

Метод позволяет включать и выключать зеркальные поля в документе.

Пример:

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

7.31.16 Метод document:areMirroredMarginsEnabled

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

Пример:

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

7.31.17 Метод document:getPivotTablesManager

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

Пример:

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

7.31.18 Метод document:getNamedExpressions

Используется для получения списка именованных диапазонов DocumentAPI.NamedExpressions.

Пример:

```
namedExpressions = document:getNamedExpressions()
print(namedExpressions)
```

7.32 Таблица DocumentAPI.DatePatterns

Форматы даты представлены в таблице 18. Пример использования см. в главе DocumentAPI.DateTimeCellFormatting.

Таблица 18 – Форматы даты

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

7.33 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 19.

Таблица 19 – Описание полей таблицы DocumentAPI.DateTime

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

7.33.1 Метод DateTime:__eq

Метод используется для определения эквивалентности двух значений времени.

7.34 Таблица DocumentApi.DateTimeFormat

В таблице 20 представлены варианты масштабирования при печати табличных документов. Используется в качестве параметра метода `CellRange.insertCurrentDateTime()`.

Таблица 20 – Варианты масштабирования при печати табличных документов

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

7.35 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 21.

Таблица 21 – Описание полей таблицы `DocumentAPI.DateTimeCellFormatting`

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

Пример:

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

local dateTimeCellFormatting = DocumentAPI.DateTimeCellFormatting()
dateTimeCellFormatting.dateListID = DocumentAPI.DatePatterns_FullDate
dateTimeCellFormatting.timeListID = DocumentAPI.TimePatterns_LongTime

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

7.36 Таблица DocumentAPI.Field

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

7.37 Таблица DocumentAPI.Fill

Таблица описывает свойства заполнения для `ShapeProperties`, `CellProperties`.

Варианты заполнения:

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

Примеры:

```
-- Без заполнения
shapeProperties.fill = DocumentAPI.Fill()
```

```
-- Заполнение цветом
shapeProperties.fill =
DocumentAPI.Fill(DocumentAPI.Color(DocumentAPI.ColorRGBA(55, 146, 179, 200)))
```

```
-- Заполнение шаблоном из url
shapeProperties.fill = DocumentAPI.Fill("https://fillpattern.url")
```

7.37.1 Метод Fill:getColor

Метод возвращает цвет заполнения `DocumentAPI.Color`.

7.37.2 Метод Fill:getUrl

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

7.37.3 Метод Fill:isNoFill

Метод возвращает `true`, если заполнения нет.

7.38 Таблица DocumentAPI.FiltersRange

Таблица `FiltersRange` реализует диапазон таблицы, позволяющий манипулировать

фильтрами столбцов. Пример использования приведен в разделе Работа с фильтрами.

7.38.1 Метод FIltersRange:clear

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

Пример:

```
local filtersRange = table:createFiltersRange(range)
.....................................
tableFilters:clear()
```

7.38.2 Метод FIltersRange:eraseFilters

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

Пример:

```
local filtersRange = table:createFiltersRange(range)
.....................................
tableFilters:eraseFilters()
```

7.38.3 Метод FiltersRange:getCellRange

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

Пример:

```
local cellRange = filtersRange:getCellRange()
print(cellRange:getBeginRow() .. ", " .. cellRange:getLastRow())
```

Более подробный пример приведен в разделе Работа с фильтрами.

7.38.4 Метод FiltersRange:setFilters

Метод `FiltersRange:setFilters` устанавливает фильтры `TableFilters` для столбцов диапазона. Фильтрация выполняется с использованием логической операции AND по столбцам. Нумерация столбцов начинается относительно левой позиции диапазона фильтрации. Если номер столбца в фильтрах превышает диапазон фильтрации, то фильтр будет успешно добавлен, но фильтрация для этого столбца будет пропущена. Такое поведение полезно, если необходимо изменить размер диапазона фильтрации при этом

сохранить ранее определенные фильтры. Диапазон фильтрации должен существовать до вызова этого метода. В настоящее время DocumentAPI позволяет создавать диапазон фильтрации только для всего листа табличного документа. Для создания или изменения размера существующего диапазона используется метод `Table:createFiltersRange()`.

Пример:

```
local filtersRange = table:createFiltersRange(range)
.....................................
local tableFilters = DocumentAPI.TableFilters()
tableFilters:setFilter(0, johnPaulFilter)
tableFilters:setFilter(1, songFilter)
.....................................
filtersRange:setFilters(tableFilters)
```

Более подробный пример приведен в разделе Работа с фильтрами.

7.39 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 22.

Таблица 22 – 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())
```

7.40 Таблица DocumentAPI.FrozenRangePosition

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

7.40.1 Конструкторы

Конструктор с параметрами по умолчанию.

```
FrozenRangePosition()
```

Конструктор, создающий диапазон ячеек. В качестве параметров используются координаты левой верхней и правой нижней точек области.

```
FrozenRangePosition(top, left, bottom, right)
```

Примеры:

```
frozenRangePosition = DocumentAPI.FrozenRangePosition()
print(frozenRangePosition:isRowsCols())

frozenRangePosition = DocumentAPI.FrozenRangePosition(0, 2, 5, 5)
print(frozenRangePosition:isRowsCols())
```

7.40.2 Метод FrozenRangePosition:create

Создает объект заблокированной области таблицы `FrozenRangePosition`. В качестве параметров используются координаты левой верхней и правой нижней точек области.

Вызов:

```
FrozenRangePosition create(top, left, bottom, right)
```

Пример:

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.create(0, 2, 5, 5)
print(frozenRangePosition:isRowsCols())
```

7.40.3 Метод FrozenRangePosition:createFrozenArea

Создает объект заблокированной области таблицы `FrozenRangePosition`. Область содержит все ячейки прямоугольника `{0, 0, bottom, right}`.

Вызов:

```
FrozenRangePosition createFrozenArea(bottom, right)
```

Пример:

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenArea(0, 2)
print(frozenRangePosition:isArea())
```

7.40.4 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())
```

7.40.5 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())
```

7.40.6 Метод FrozenRangePosition:isRowsCols

Возвращает `true` если диапазон содержит строки и колонки.

Пример:

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

7.40.7 Метод FrozenRangePosition:isArea

Возвращает `true` если диапазон является непрерывной областью.

Пример:

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

7.40.8 Метод FrozenRangePosition:isRows

Возвращает `true` если диапазон состоит из строк.

Пример:

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

7.40.9 Метод FrozenRangePosition:isCols

Возвращает `true` если диапазон состоит из колонок.

Пример:

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

7.40.10 Метод FrozenRangePosition:__eq

Метод используется для определения эквивалентности двух объектов `FrozenRangePosition`.

Пример:

```
local pos1 = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local pos2 = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
print(pos1:__eq(pos2)) -- true
```

7.41 Таблица DocumentAPI.HeadersFooters

Таблица `DocumentAPI.HeadersFooters` представляет коллекцию верхних и нижних колонтитулов раздела текстового документа (см. Рисунок 31). Доступ к колонтитулам осуществляется посредством методов `Section.getHeaders()`, `Section.getFooters()`.

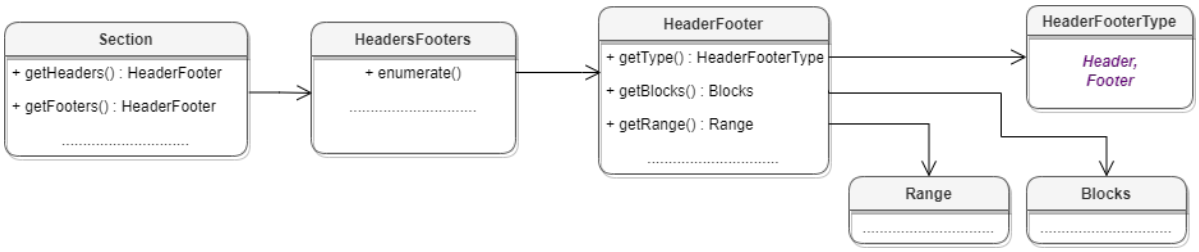


Рисунок 31 – Таблицы колонтитулов

7.41.1 Метод HeadersFooters:enumerate

Метод возвращает коллекцию колонтитулов.

Пример:

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

7.42 Таблица DocumentAPI.HeaderFooter

Таблица `DocumentAPI.HeaderFooter` определяет колонтитул текстового документа.

7.42.1 Метод HeaderFooter:getType

Метод предоставляет информацию о типе колонтитула (`DocumentAPI.HeaderFooterType`).

Пример:

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

7.42.2 Метод HeaderFooter:getBlocks

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

колонтитуле.

Пример:

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    for block in header:getBlocks():enumerate() do
        print(block:getRange():extractText())
    end
end
```

7.42.3 Метод HeaderFooter:getRange

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

Пример:

```
local section = document:getBlocks():getBlock(0):getSection()
local headers = section:getHeaders()
for header in headers:enumerate() do
    print(header:getRange():extractText())
end
```

7.43 Таблица DocumentAPI.HeaderFooterType

Типы колонтитулов представлены в таблице 23.

Таблица 23 – Типы колонтитулов

Наименование константы	Описание
DocumentAPI.HeaderFooterType_Header	Верхний колонтитул
DocumentAPI.HeaderFooterType_Footer	Нижний колонтитул

7.44 Таблица DocumentAPI.HorizontalAnchorAlignment

В таблице 24 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали. Используется в DocumentAPI.HorozontalTextAnchoredPosition.

Таблица 24 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

Наименование константы	Описание
DocumentAPI.HorizontalAnchorAlignment_Left	По верхнему краю

Наименование константы	Описание
DocumentAPI.HorizontalAnchorAlignment_Right	По нижнему краю
DocumentAPI.HorizontalAnchorAlignment_Center	По центру
DocumentAPI.HorizontalAnchorAlignment_Inside, DocumentAPI.HorizontalAnchorAlignment_Outside	По границам

7.45 Таблица DocumentAPI.HorizontalRelativeTo

В таблице 25 представлены типы размещения объекта относительно закрепленной позиции по горизонтали. Используется в DocumentAPI.HorozontalTextAnchoredPosition.

Таблица 25 – 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
DocumentAPI.HorizontalRelativeTo_PageInsideMargin	Внутреннее поле страницы
DocumentAPI.HorizontalRelativeTo_PageOutsideMargin	Внешнее поле страницы

7.46 Таблица DocumentAPI.HorizontalTextAnchoredPosition

Таблица DocumentAPI.HorizontalTextAnchoredPosition (см. Рисунок 32) предназначена для управления относительным положением объекта со смещением или выравниванием по горизонтали. Пример использования см. в InlineFrame:setPosition().

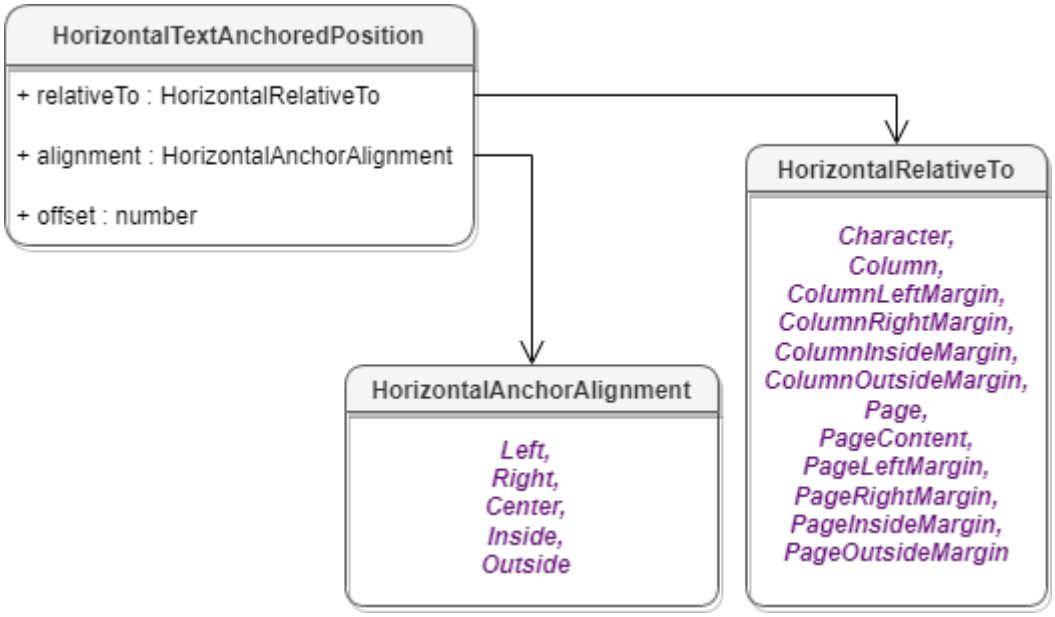


Рисунок 32 – Поля таблицы `DocumentAPI.HorizontalTextAnchoredPosition`

Описание полей таблицы `DocumentAPI.HorizontalTextAnchoredPosition` представлено в таблице 26.

Таблица 26 – Описание полей таблицы `DocumentAPI.HorizontalTextAnchoredPosition`

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

7.46.1 Метод HorizontalTextAnchoredPosition:__eq

Метод используется для определения эквивалентности двух положений объекта по горизонтали.

Пример:

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
```

```
(DocumentAPI.HorizontalRelativeTo_Column)
pos1.horizontal.offset = 1

local pos2 = DocumentAPI.TextAnchoredPosition()
pos2.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos2.horizontal.offset = 1

print(pos1.horizontal:__eq(pos2.horizontal))
```

7.47 Таблица DocumentAPI.Hyperlink

Таблица `DocumentAPI.Hyperlink` описывает свойства ссылки. Может быть получена посредством вызова метода `Cell.getHyperlink()`.

Таблица 27 – 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()
if (hyperlink ~= nil) then
  print(hyperlink.url, hyperlink.tooltip, hyperlink.label)
end
```

7.47.1 Метод Hyperlink:__eq

Метод используется для определения эквивалентности двух объектов `Hyperlink`.

Пример:

```
table_0 = document:getBlocks():getTable(0)
cell_00 = table_0:getCell(DocumentAPI.CellPosition(0, 0))
cell_01 = table_0:getCell(DocumentAPI.CellPosition(0, 1))
local hyperlink_00 = cell_00:getHyperlink()
local hyperlink_01 = cell_01:getHyperlink()
if (hyperlink_00 and hyperlink_01) then
```

```
   print(hyperlink_00:__eq(hyperlink_01))
end
```

7.48 Таблица DocumentAPI.Image

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

7.48.1 Метод Image:getFrame

Метод аналогичен методу `MediaObject:getFrame()`, он возвращает свойства позиции изображения. В зависимости от текущего редактора метод возвращает разные типы рамок. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют тип `DocumentAPI.InlineFrame`, табличные документы работают с абсолютной позицией и используют тип `DocumentAPI.AbsoluteFrame`.

Пример для текстового документа:

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if (image) then
    print(image:getFrame()) -- <userdata of type
'CO::API::Document::InlineFrame'>
  end
end
```

Пример для табличного документа:

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if (image) then
    print(image:getFrame()) -- <userdata of type
'CO::API::Document::AbsoluteFrame'>
  end
end
```

7.48.2 Метод Image:remove

Метод удаляет изображение из документа.

Пример для текстового документа:

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if image then
    image:remove()
    break
  end
end
```

7.49 Таблица DocumentAPI.Images

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

7.49.1 Метод Images:enumerate

Метод позволяет перечислить коллекцию изображений.

Пример для текстового документа:

```
for image in EditorAPI.getSelection():getImages():enumerate() do
  print(image:getFrame():getWrapType())
end
```

Пример для табличного документа:

```
local sheet = document:getBlocks():getTable(0)
local images = sheet:getImages()
for image in images:enumerate() do
  print(image:getFrame():getTopLeft().x)
end
```

7.50 Таблица DocumentAPI.InlineFrame

Таблица `DocumentAPI.InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 33). Предназначена для получения и изменения свойств позиции графических объектов. Используется в текстовом документе.

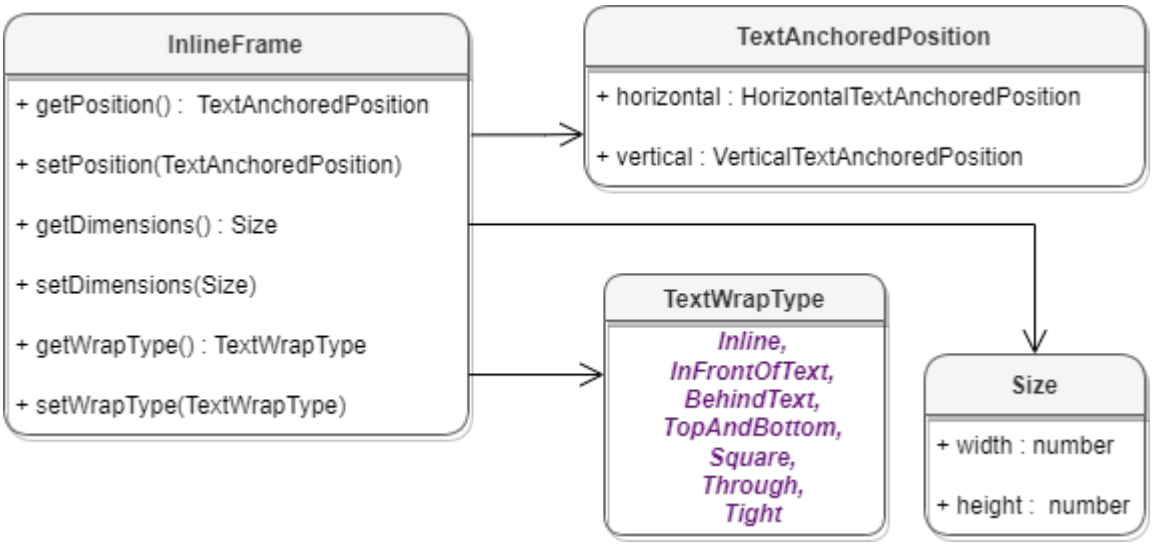


Рисунок 33 – Объектная модель таблицы `DocumentAPI.InlineFrame`

Пример для текстового документа:

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

7.50.1 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 mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
```

```
   local inlineFrame = mediaObject:getFrame()
   if (inlineFrame:getWrapType() ~= DocumentAPI.TextWrapType_Inline) then
     local pos = DocumentAPI.TextAnchoredPosition()

     -- Установка смещения по горизонтали относительно края колонки
     pos.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
     pos.horizontal.offset = 10
     -- Установка смещения по вертикали относительно края страницы
     pos.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
     pos.vertical.offset = 10

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

7.50.2 Метод InlineFrame:getPosition

Метод возвращает позицию встроенного объекта на странице в виде таблицы DocumentAPI.TextAnchoredPosition.

Пример:

```
local mediaObjects = document:getRange():getImages()
for mediaObject in mediaObjects:enumerate() do
  local inlineFrame = mediaObject:getFrame()
  local textAnchoredPosition = inlineFrame:getPosition()
  if (textAnchoredPosition) then
    print(textAnchoredPosition.horizontal, textAnchoredPosition.vertical)
  end
end
```

7.50.3 Метод InlineFrame:setDimensions

Метод задает размер DocumentAPI.SizeU встроенного объекта.

Пример:

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

7.50.4 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
```

7.50.5 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
```

7.50.6 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
```

7.51 Таблица DocumentAPI.Insets

Таблица DocumentAPI.Insets предназаначина для задания полей, например, страницы. Данный тип используется в поле margins таблицы DocumentAPI.PageProperties. Поля DocumentAPI.Insets представлены в таблице 28.

Таблица 28 – Описание полей таблицы `DocumentAPI.Insets`

Поле	Тип	Описание
`DocumentAPI.Insets.left`	number	Левая граница поля
`DocumentAPI.Insets.top`	number	Верхняя граница поля
`DocumentAPI.Insets.right`	number	Правая граница поля
`DocumentAPI.Insets.bottom`	number	Нижняя граница поля

Пример:

```
local insets = DocumentAPI.Insets()
insets.left = 10.0
print(insets.left)
```

7.52 Таблица DocumentAPI.ListSchema

Типы схем форматирования списков, которые могут быть применены к абзацам текста, представлены в таблице 29. Данные константы используются в методах `Paragraph:getListSchema()`, `Paragraph:setListSchema()`.

Таблица 29 – Типы схем форматирования списков

Наименование константы	Описание
`DocumentAPI.ListSchema_Unknown`	Схема не определена
`DocumentAPI.ListSchema_UnknownBullet`	Список без маркера
`DocumentAPI.ListSchema_UnknownNumbering`	Нумерация без номера
`DocumentAPI.ListSchema_BulletCircleSolid`	Список с маркерами в виде заполненного круга
`DocumentAPI.ListSchema_BulletCircleContour`	Список с маркерами в виде окружности
`DocumentAPI.ListSchema_BulletSquareSolid`	Список с маркерами в виде квадрата
`DocumentAPI.ListSchema_BulletDiamondDots`	Список с маркерами в виде четырех ромбов
`DocumentAPI.ListSchema_BulletHyphen`	Список с маркерами в виде дефиса
`DocumentAPI.ListSchema_BulletConcaveArrowSolid`	Список с маркерами в виде вогнутой стрелки.
`DocumentAPI.ListSchema_BulletCheckmark`	Список с маркерами в виде галочки.
`DocumentAPI.ListSchema_EnumeratorDecimalDot`	Десятичная нумерация с точкой.
`DocumentAPI.ListSchema_EnumeratorDecimalDotMultiLevel`	Многоуровневая десятичная нумерация с точкой

Наименование константы	Описание
DocumentAPI.ListSchema_EnumeratorDecimalBracket	Десятичная нумерация со скобкой
DocumentAPI.ListSchema_EnumeratorLatinUppercaseDot	Нумерация латинскими прописными буквами с точкой
DocumentAPI.ListSchema_EnumeratorLatinLowercaseDot	Нумерация латинскими строчными буквами с точкой
DocumentAPI.ListSchema_EnumeratorLatinLowercaseBracket	Нумерация латинскими строчными буквами со скобкой
DocumentAPI.ListSchema_EnumeratorRomanUppercaseDot	Нумерация римскими прописными цифрами с точкой
DocumentAPI.ListSchema_EnumeratorRomanLowercaseDot	Нумерация римскими строчными цифрами с точкой
DocumentAPI.ListSchema_EnumeratorDecimalRussianBracket	Десятичная нумерация через запятую со скобкой
DocumentAPI.ListSchema_EnumeratorRussianLowercaseBracket	Нумерация с русскими строчными буквами со скобкой

Пример:

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

7.53 Таблица DocumentAPI.LineEndingProperties

Таблица DocumentAPI.LineEndingProperties содержит варианты оформления окончаний линий. Описание полей таблицы DocumentAPI.LineEndingProperties представлено в таблице 30. Используется в полях headLineEndingProperties и tailLineEndingProperties таблицы DocumentAPI.LineProperties.

Таблица 30 – 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 tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("C3")

lineProperties = DocumentAPI.LineProperties()
lineProperties.headLineEndingProperties = DocumentAPI.LineEndingProperties()
```

```
lineProperties.headLineEndingProperties.style =
DocumentAPI.LineEndingStyle_Arrow

lineProperties.headLineEndingProperties.relativeExtent = DocumentAPI.SizeU()
lineProperties.headLineEndingProperties.relativeExtent.width = 2
lineProperties.headLineEndingProperties.relativeExtent.height = 2

lineProperties.tailLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.tailLineEndingProperties.style =
DocumentAPI.LineEndingStyle_Arrow
lineProperties.tailLineEndingProperties.relativeExtent = DocumentAPI.SizeU()
lineProperties.tailLineEndingProperties.relativeExtent.width = 2
lineProperties.tailLineEndingProperties.relativeExtent.height = 2

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

7.53.1 Метод LineEndingProperties:__eq

Метод используется для определения эквивалентности значений двух объектов `LineEndingProperties`.

Пример:

```
lineEnding1 = DocumentAPI.LineEndingProperties()
lineEnding1.style = DocumentAPI.LineEndingStyle_Arrow
lineEnding2 = DocumentAPI.LineEndingProperties()
lineEnding2.style = DocumentAPI.LineEndingStyle_Diamond
print("Eq: " .. tostring(lineEnding1:__eq(lineEnding2)))
```

7.54 Таблица DocumentAPI.LineEndingStyle

В таблице 31 приведены типы окончания линии. Используется в поле `style` таблицы `DocumentAPI.LineEndingProperties`.

Таблица 31 – Типы окончания линии

Наименование константы	Описание
`DocumentAPI.LineEndingStyle_Arrow`	
`DocumentAPI.LineEndingStyle_Diamond`	

Наименование константы	Описание
DocumentAPI.LineEndingStyle_Oval	
DocumentAPI.LineEndingStyle_Stealth	
DocumentAPI.LineEndingStyle_Triangle	
DocumentAPI.LineEndingStyle_None	

Пример:

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

lineProperties = DocumentAPI.LineProperties()
lineProperties.headLineEndingProperties = DocumentAPI.LineEndingProperties()
lineProperties.headLineEndingProperties.style = DocumentAPI.LineEndingStyle_Oval

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

7.55 Таблица DocumentAPI.LineProperties

Таблица `DocumentAPI.LineProperties` предназначена для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 34).

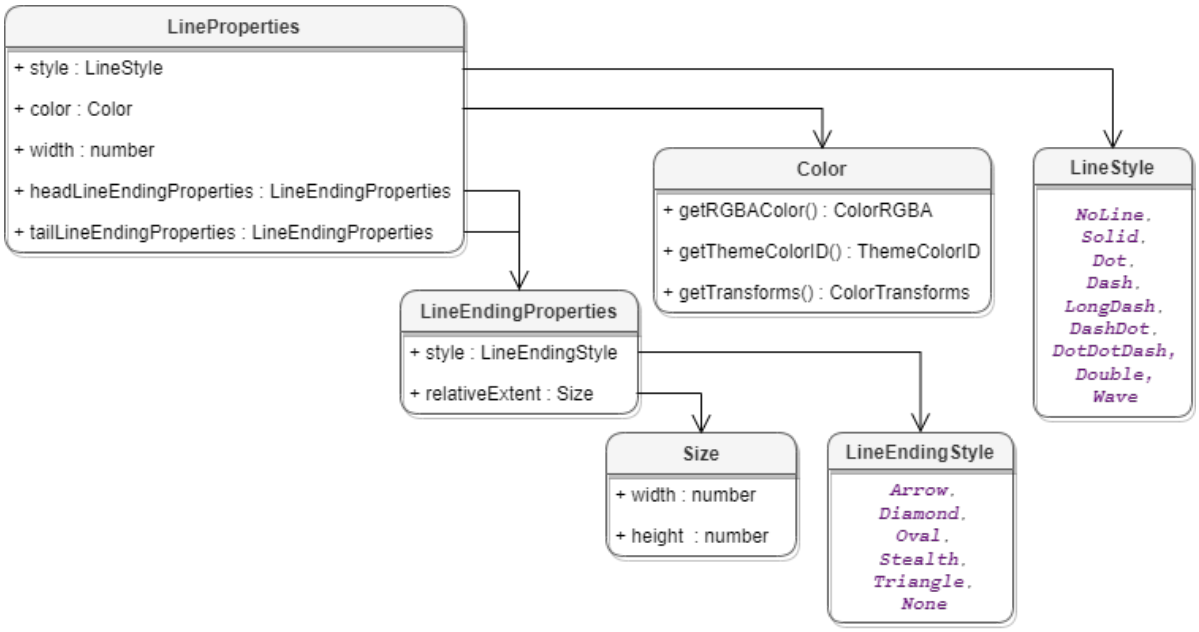


Рисунок 34 – Свойства границ ячеек

Пример:

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

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

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
local brds = cell:setBorders(borders)
```

7.55.1 Поле LineProperties.style

Поле предназначено для установки типа линии. Допустимые значения представлены в разделе `DocumentAPI.LineStyle`.

7.55.2 Поле LineProperties.width

Поле предназначено для установки ширины линии. Тип - числовой.

7.55.3 Поле LineProperties.color

Поле предназначено для установки цвета линии. Тип - `DocumentAPI.Color`.

7.55.4 Поле LineProperties.headLineEndingProperties

Поле предназначено для оформления начала линии DocumentAPI.LineEndingProperties.

7.55.5 Поле LineProperties.tailLineEndingProperties

Поле предназначено для оформления конца линии DocumentAPI.LineEndingProperties.

7.55.6 Метод LineProperties:__eq

Метод используется для определения эквивалентности значений двух объектов LineProperties.

Пример:

```
lineProperties1 = DocumentAPI.LineProperties()
lineProperties1.style = DocumentAPI.LineStyle_Solid

lineProperties2 = DocumentAPI.LineProperties()
lineProperties2.style = DocumentAPI.LineStyle_Dot

print("Eq: " .. tostring(lineProperties1:__eq(lineProperties2)))
```

7.56 Таблица DocumentAPI.LineSpacing

Таблица DocumentAPI.LineSpacing задает межстрочный интервал абзаца. Поля таблицы приведены в таблице 32. Для управления значением межстрочного интервала используются значения, представленные в разделе DocumentAPI.LineSpacingRule.

Таблица 32 – Параметры межстрочного интервала

Поле	Описание
DocumentAPI.LineSpacing.value	Значение межстрочного интервала.
DocumentAPI.LineSpacing.rule	Правило формирования межстрочного интервала DocumentAPI.LineSpacingRule.

Пример:

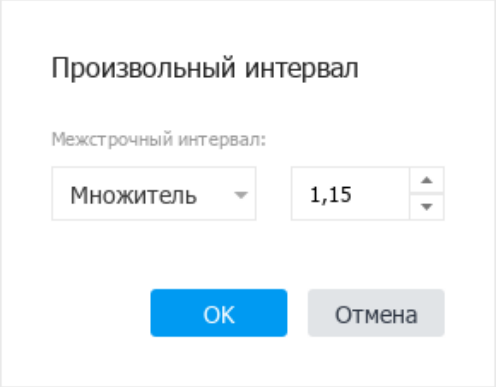
```
-- Конструктор
local lineSpacing = DocumentAPI.LineSpacing(1.5,
DocumentAPI.LineSpacingRule_Multiple)
-- Обращение к полям
```

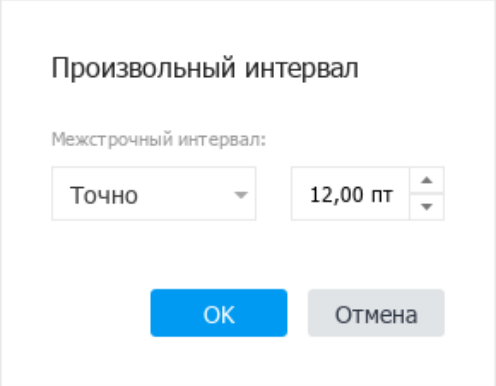
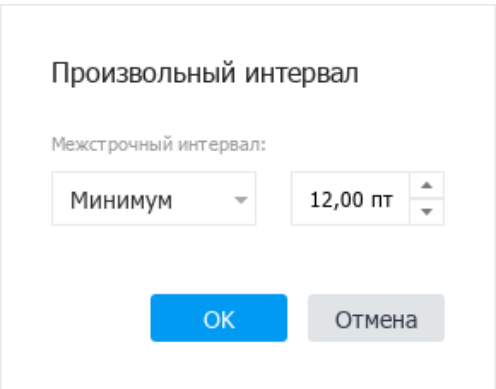
```
lineSpacing.value = 1
lineSpacing.rule = DocumentAPI.LineSpacingRule_Exact
```

7.57 Таблица DocumentAPI.LineSpacingRule

В таблице 33 представлены варианты правил формирования межстрочного интервала текстового абзаца.

Таблица 33 – Виды межстрочного интервала

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

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

Пример:

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

7.58 Таблица DocumentAPI.LineStyle

В таблице 34 приведены типы линий. Используется в поле `style` таблицы `DocumentAPI.LineProperties`.

Таблица 34 – Типы линий

Наименование константы	Описание
`DocumentAPI.LineStyle_NoLine`	Нет линии
`DocumentAPI.LineStyle_Solid`	
`DocumentAPI.LineStyle_Dot`	
`DocumentAPI.LineStyle_Dash`	
`DocumentAPI.LineStyle_LongDash`	
`DocumentAPI.LineStyle_DashDot`	
`DocumentAPI.LineStyle_DotDotDash`	
`DocumentAPI.LineStyle_Double`	
`DocumentAPI.LineStyle_Wave`	

Пример:

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

lineProperties = DocumentAPI.LineProperties()
lineProperties.style = DocumentAPI.LineStyle_Wave

borders = DocumentAPI.Borders()
borders = borders:setTop(lineProperties)
cell:setBorders(borders)
```

7.59 Таблица DocumentAPI.MediaObject

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

7.59.1 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
```

7.59.2 Метод MediaObject:toChart

Метод возвращает диаграмму `DocumentAPI.Chart`, связанную со встроенным объектом. Если объект не является диаграммой, метод возвращает `nil`.

Диаграммы реализованы только в табличных документах.

Пример для табличного документа:

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

```
for mediaObject in mediaObjects:enumerate() do
  local chart = mediaObject:toChart()
  if chart ~= nil then
    print("Текущий объект является диаграммой")
  else
    print("Текущий объект является фигурой")
  end
end
```

7.59.3 Метод MediaObject:getFrame

Метод возвращает свойства позиции встроенного объекта. В зависимости от текущего редактора метод возвращает разные типы таблиц. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют таблицу `DocumentAPI.InlineFrame`, табличные документы работают с абсолютной позицией и используют таблицу `DocumentAPI.AbsoluteFrame`.

Пример для текстового документа:

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  print(mediaObject:getFrame()) -- <userdata of type
'CO::API::Document::InlineFrame'>
end
```

Пример для табличного документа:

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  print(mediaObject:getFrame()) -- <userdata of type
'CO::API::Document::AbsoluteFrame'>
end
```

7.60 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 35).

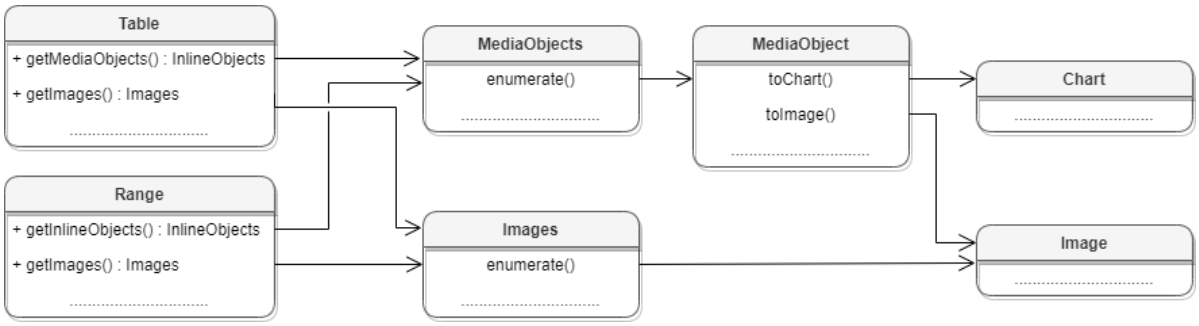


Рисунок 35 – Графические объекты

7.60.1 Метод MediaObjects:enumerate

Метод позволяет перечислить коллекцию встроенных объектов.

Примеры для текстового документа:

```
local mediaObjects = document:getRange():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  print("Встроенный объект:", mediaObject)
end
```

```
local mediaObjects = EditorAPI.getSelection():getInlineObjects()
for mediaObject in mediaObjects:enumerate() do
  print("Встроенный объект:", mediaObject)
end
```

Пример для табличного документа:

```
local table = document:getBlocks():getTable(0)
local mediaObjects = table:getMediaObjects()
for mediaObject in mediaObjects:enumerate() do
  local image = mediaObject:toImage()
  if image ~= nil then
    print("Объект является изображением")
  else
    local chart = mediaObject:toChart()
    if chart ~= nil then
      print("Объект является диаграммой")
    else
      print("Объект является фигурой")
    end
  end
end
```

7.61 Таблица DocumentAPI.NamedExpressions

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

7.61.1 Метод NamedExpressions:get

Возвращает именованный диапазон NamedExpression по имени name, если он существует.

Пример:

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

7.61.2 Метод NamedExpressions:enumerate

Позволяет получить доступ ко всему списку именованных диапазонов.

Пример:

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

7.61.3 Метод NamedExpression:addExpression

Добавляет новый диапазон в список именованных диапазонов, возвращает результат операции NamedExpressionsValidationResult.

Пример:

```
local expressionName = "Покупки"
local expressionValue = "=Формула покупки!$E$6:$E$14"
local validationResult = namedExpressions:addExpression(expressionName,
expressionValue)
if (validationResult == DocumentAPI.NamedExpressionsValidationResult_Success)
then
```

```
   print("Named expression was added")
end
```

7.61.4 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
```

7.62 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
```

7.62.1 Метод NamedExpression:getName

Возвращает имя именованного диапазона. Пример см. в DocumentAPI.NamedExpression.

7.62.2 Метод NamedExpression:getExpression

Возвращает текст диапазона (формулы). Пример см. в DocumentAPI.NamedExpression.

7.62.3 Метод NamedExpression:getCellRange

Возвращает диапазон ячеек DocumentAPI.CellRange именованного диапазона. Пример см. в DocumentAPI.NamedExpression.

7.63 Таблица DocumentAPI.NamedExpressionsValidationResult

Таблица DocumentAPI.NamedExpressionsValidationResult описывает результат операций NamedExpressions:addExpression(), NamedExpressions:removeExpression(). Описание полей таблицы представлено в таблице 35.

Таблица 35 – Описание полей таблицы DocumentAPI.NamedExpressionsValidationResult

Поле	Описание
DocumentAPI.NamedExpressionsValidationResult_Success	Операция выполнена успешно
DocumentAPI.NamedExpressionsValidationResult_WrongName	Неправильный формат имени
DocumentAPI.NamedExpressionsValidationResult_isUsedInFormula	Имя уже используется в формуле

7.64 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 36.

Таблица 36 – 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.useBracketsForNagative	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())
```

7.65 Таблица DocumentAPI.PageFieldOrder

Таблица `DocumentAPI.PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем таблицы `DocumentAPI.PivotTableLayoutSettings`. Описание полей таблицы представлено в таблице 37.

Таблица 37 – Описание полей таблицы `DocumentAPI.PageFieldOrder`

Поле	Описание
`DocumentAPI.PageFieldOrder_DownThenOver`	Вниз, затем поперек
`DocumentAPI.PageFieldOrder_OverThenDown`	Поперек, затем вниз

7.66 Таблица DocumentAPI.PageProperties

Таблица `DocumentAPI.PageProperties` предоставляет такие свойства страницы как высота, ширина, размеры полей. Описание полей приведено в таблице 38. Используется в `Document.setPageProperties()`, `Section.getPageProperties()`, `Section.setPageProperties()`. Применяется только в текстовом документе.

Таблица 38 – 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)
```

7.66.1 Метод PageProperties:__eq

Метод позволяет использовать оператор сравнения `__eq` для определения эквивалентности содержимого двух структур `DocumentAPI.PageProperies`.

Пример:

```
local pageProperties1 = DocumentAPI.PageProperties(100, 200)
document:setPageProperties(pageProperties1)

local pageProperties2 = DocumentAPI.PageProperties(100, 200)
document:setPageProperties(pageProperties2)

print(pageProperties1:__eq(pageProperties2)) -- true
```

7.67 Таблица DocumentAPI.Paragraph

Таблица `DocumentAPI.Paragraph` предоставляет доступ к свойствам абзаца (см. Рисунок 36).

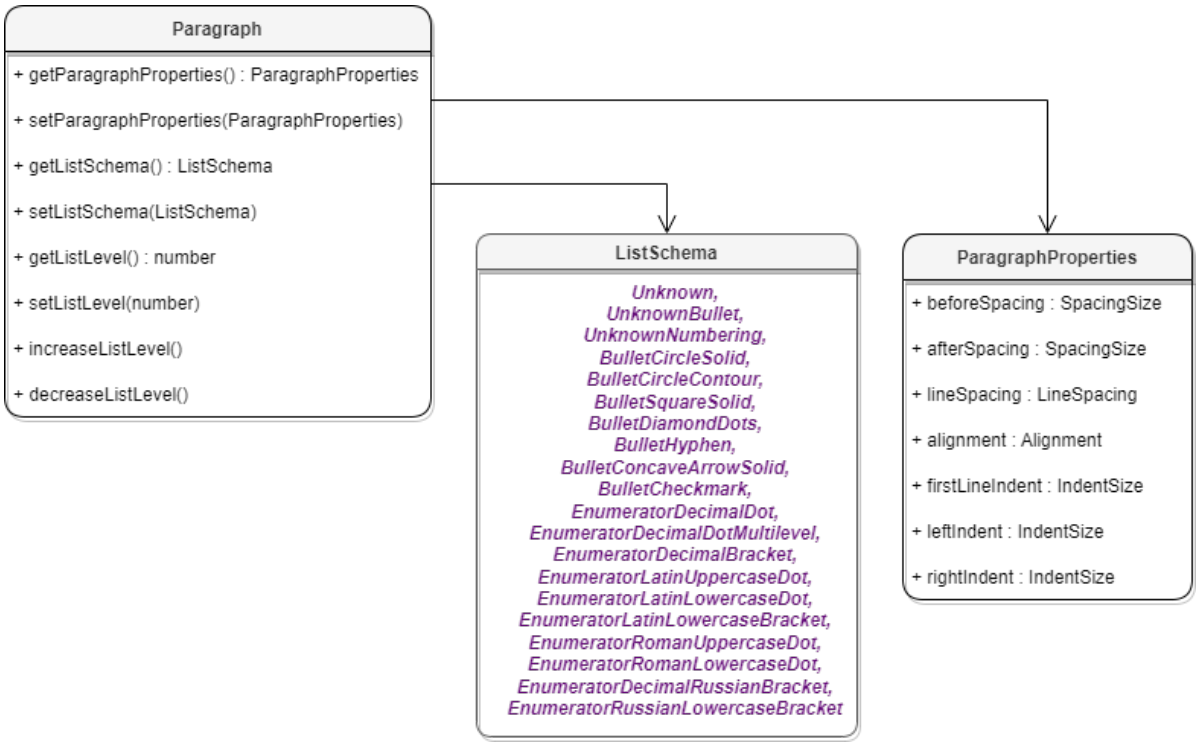


Рисунок 36 – Объектная модель таблиц для работы со свойствами параграфа

7.67.1 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
```

7.67.2 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
```

7.67.3 Метод Paragraph:getListSchema

Метод возвращает схему форматирования абзаца DocumentAPI.ListSchema либо значение `nil`, если схема нумерации не установлена для абзаца. Данный метод используется только в текстовом документе.

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
local schema = paragraph:getListSchema()
```

7.67.4 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)
```

7.67.5 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()
```

7.67.6 Метод Paragraph:setListLevel

Метод позволяет установить глубину вложенности элемента списка.

Значение может быть равным `nil`, если схема нумерации не установлена для абзаца. В этом случае будет установлено минимальное значение. Данный метод используется только в текстовом документе.

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
local level = paragraph:setListLevel(1)
```

7.67.7 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()
```

7.67.8 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()
```

7.68 Таблица DocumentAPI.ParagraphProperties

Таблица `DocumentAPI.ParagraphProperties` предназначена для управления свойствами форматирования (см. Рисунок 37). Таблица `DocumentAPI.ParagraphProperties` используется в методах `Paragraph:getParagraphProperties` и `Paragraph:setParagraphProperties`.

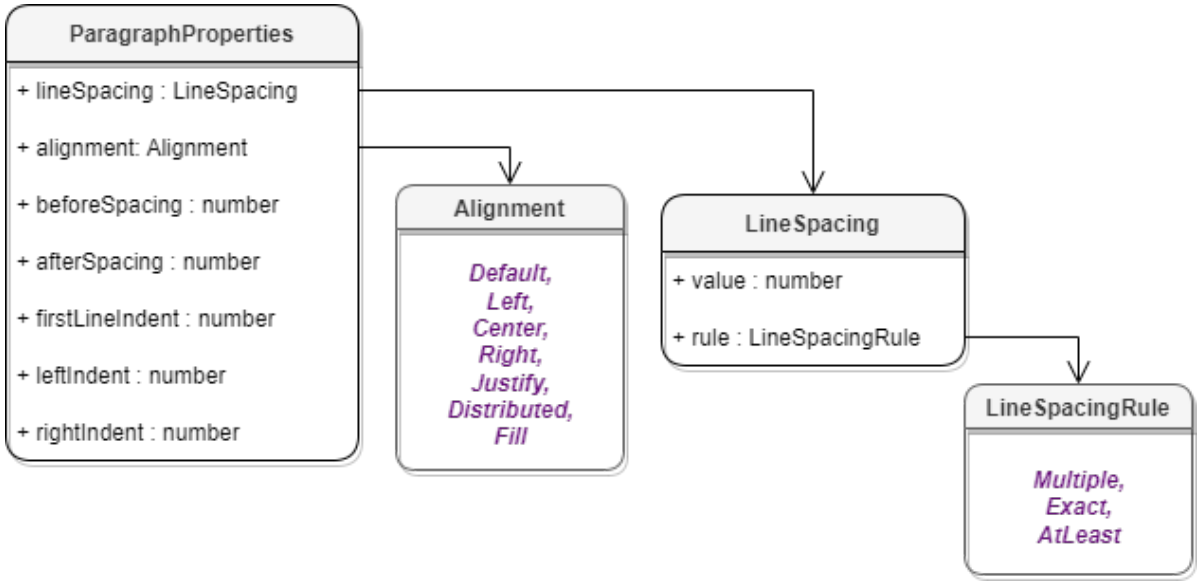
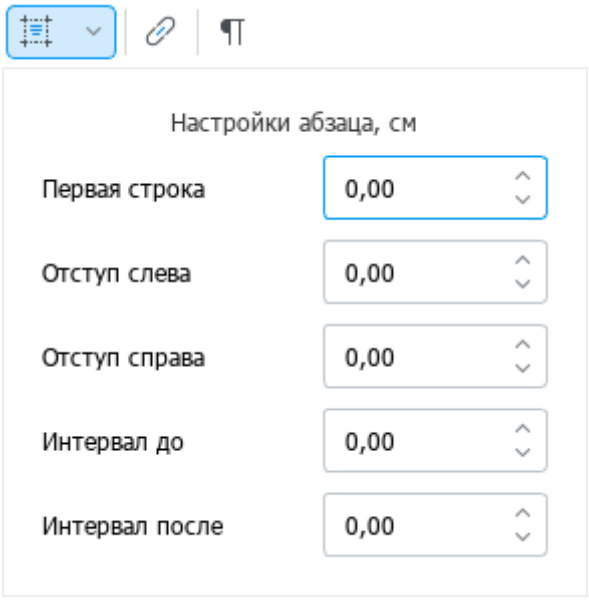


Рисунок 37 – Объектная модель таблиц для работы со свойствами параграфа

Описание полей таблицы `DocumentAPI.ParagraphProperties` представлено в таблице 39.

Таблица 39 – Описание полей таблицы `DocumentAPI.ParagraphProperties`

Поле	Описание
`ParagraphProperties.beforeSpacing`	Установка величины расстояния до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца,** (см. рисунок выше), в поле **Интервал до** (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).
`ParagraphProperties.afterSpacing`	Установка величины расстояния после абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, в поле **Интервал после** (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).

Поле	Описание
	
`ParagraphProperties.lineSpacing`	Расстояние между строк одного абзаца (межстрочный интервал), `LineSpacing`.
`ParagraphProperties.alignment`	Выравнивание текстового фрагмента по горизонтали. Список допустимых значений находится в разделе `Alignment`.
`ParagraphProperties.firstLineIndent`	Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Первая строка** (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).
`ParagraphProperties.leftIndent`	Расстояние от левого поля документа до абзаца (отступ слева). При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ слева** (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).
`ParagraphProperties.rightIndent`	Расстояние от правого поля документа до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ справа** (подробнее см. в документе «МойОфис Стандартный. МойОфис Текст. Руководство пользователя»).

Пример для текстового документа:

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
--
para_props.afterSpacing = 28.3 -- значение соответствует 1 см
para_props.beforeSpacing = 28.3 -- значение соответствует 1 см
para_props.alignment = DocumentAPI.Alignment_Center
para_props.firstLineIndent = 28.3 -- значение соответствует 1 см
para_props.leftIndent = 28.3 -- значение соответствует 1см
para_props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
para_props.rightIndent = 28.3 -- значение соответствует 1см
--
para:setParagraphProperties(para_props)
```

Пример для табличного документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange()
local paragraphs = range:getParagraphs()

for para in paragraphs:enumerate() do
   local para_props = para:getParagraphProperties()
   para_props.afterSpacing = 28.3 -- значение соответствует 1 см
   para_props.beforeSpacing = 28.3 -- значение соответствует 1 см
   para_props.alignment = DocumentAPI.Alignment_Center
   para_props.firstLineIndent = 28.3 -- значение соответствует 1 см
   para_props.leftIndent = 28.3 -- значение соответствует 1см
   para_props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
   para_props.rightIndent = 28.3 -- значение соответствует 1см
   para:setParagraphProperties(para_props)
end
```

7.69 Таблица DocumentAPI.Paragraphs

Таблица `DocumentAPI.Paragraphs` предоставляет доступ к коллекции абзацев типа `DocumentAPI.Paragraph` (см. Рисунок 38). Коллекция абзацев может быть получена из таблицы `DocumentAPI.Range` посредством использования вызова `Range:getParagraphs()`.

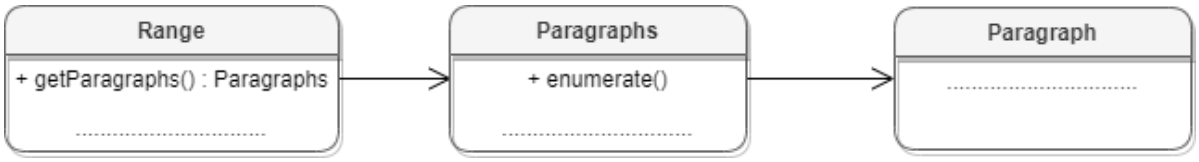


Рисунок 38 – Объектная модель для работы со списком абзацев

Пример для текстового документа:

```
local paragraphs = document:getRange():getParagraphs()
```

Пример для табличного документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local paragraphs = cell:getRange():getParagraphs()
```

7.69.1 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)
```

7.69.2 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)
```

7.69.3 Метод Paragraphs:increaseListLevel

Метод увеличивает уровень списка на единицу. В случае, если максимальный уровень уже установлен, увеличения не происходит. Данный метод используется только в текстовом документе.

Пример:

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

7.69.4 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()
```

7.69.5 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
```

7.70 Таблица DocumentAPI.PageOrientation

Типы ориентации страницы представлены в таблице 40. Данная константа может быть использована для получения / установки ориентации страниц для секции или документа.

Таблица 40 – 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())
```

7.71 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 41.

Таблица 41 – Описание полей таблицы `DocumentAPI.PercentageCellFormatting`

Поле	Описание
`DocumentAPI.PercentageCellFormatting.decimalPlaces`	Количество десятичных позиций

Пример:

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

local percentageCellFormatting = DocumentAPI.PercentageCellFormatting()
percentageCellFormatting.decimalPlaces = 2

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

7.72 Таблица DocumentAPI.PivotTable

Таблица для представления сводной таблицы. Может быть получена из ячейки `Cell.getPivotTable()`, или при создании новой сводной таблицы `PivotTablesManager.create()`.

7.72.1 Метод PivotTable:remove

Метод удаляет сводную таблицу.

Пример:

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

7.72.2 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()
if (pivotTable) then
  print(pivotTable:getSourceRangeAddress()) -- 'Sheet1'!I3:K7
end
```

7.72.3 Метод PivotTable:getSourceRange

Метод возвращает диапазон `DocumentAPI.CellRange` исходных данных сводной таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local cellRange = pivotTable:getSourceRange()
print(cellRange:getBeginRow(), cellRange:getLastRow()) -- 2  6
```

7.72.4 Метод PivotTable:getPivotRange

Метод возвращает диапазон ячеек DocumentAPI.CellRange, в котором размещена сводная таблица.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local cellRange = pivotTable:getPivotRange()
print(cellRange:getBeginRow(), cellRange:getLastRow()) -- 7  10
```

7.72.5 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())
```

7.72.6 Метод PivotTable:isRowGrandTotalEnabled

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

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A3")
local pivotTable = cell:getPivotTable()
print(pivotTable:isRowGrandTotalEnabled())
```

7.72.7 Метод PivotTable:isColumnGrandTotalEnabled

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

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A3")
local pivotTable = cell:getPivotTable()
print(pivotTable:isColumnGrandTotalEnabled())
```

7.72.8 Метод PivotTable:getPivotTableCaptions

Метод возвращает информацию DocumentAPI.PivotTableCaptions о всех заголовках сводной таблицы.

Пример:

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

7.72.9 Метод PivotTable:getPivotTableLayoutSettings

Метод возвращает настройки отображения DocumentAPI.PivotTableLayoutSettings сводной таблицы.

Пример:

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

7.72.10 Метод PivotTable:getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства DocumentAPI.PivotTableUnsupportedFeature сводной таблицы.

Пример:

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

7.72.11 Метод PivotTable:getFieldsList

Метод возвращает список DocumentAPI.PivotTableField всех полей сводной таблицы.

Пример:

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

7.72.12 Метод PivotTable:getRowFields

Метод возвращает список полей DocumentAPI.PivotTableCategoryField из области строк.

Пример:

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

7.72.13 Метод PivotTable:getColumnFields

Метод возвращает список полей DocumentAPI.PivotTableCategoryField из области колонок.

Пример:

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

7.72.14 Метод PivotTable:getValueFields

Метод возвращает список полей DocumentAPI.PivotTableValueField из области значений.

Пример:

```
local valueFields = pivotTable:getValueFields()
for fieldIdx = 0, valueFields:size() - 1 do
   print(valueFields[fieldIdx].baseFieldName)
   print(valueFields[fieldIdx].valueFieldName)
   print(valueFields[fieldIdx].cellNumberFormat)
   print(valueFields[fieldIdx].totalFunction)
end
```

7.72.15 Метод PivotTable:getPageFields

Метод возвращает список полей DocumentAPI.PivotTablePageField из области фильтров.

Пример:

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

7.72.16 Метод PivotTable:getFieldCategories

Метод возвращает список категорий DocumentAPI.PivotTableFieldCategories, содержащих заданное поле `fieldName`.

Пример:

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

7.72.17 Метод PivotTable:getFieldItems

Метод возвращает все элементы DocumentAPI.PivotTableItems сводной таблицы по заданному имени поля `fieldName`.

Пример:

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

7.72.18 Метод PivotTable:getFieldItemsByName

Метод возвращает все элементы DocumentAPI.PivotTableItems из заданного поля `fieldName` по имени `itemName`.

Пример:

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

7.72.19 Метод PivotTable:getFilter

Метод возвращает фильтр DocumentAPI.PivotTableFilter по заданному имени поля `fieldName`.

Пример:

```
local filter = pivotTable:getFilter("Age")
print(filter:getFieldName())
```

7.72.20 Метод PivotTable:getFilters

Метод возвращает список фильтров DocumentAPI.PivotTableFilter сводной таблицы.

Пример:

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   -- use filter
end
```

7.72.21 Метод PivotTable:update

Метод обновляет и полностью пересчитывает сводную таблицу, возвращает DocumentAPI.PivotTableUpdateResult.

Пример:

```
local updateResult = pivotTable:update()
if (updateResult ~= DocumentAPI.PivotTableUpdateResult_Success) then
   print(updateResult)
end
```

7.72.22 Метод PivotTable:createPivotTableEditor

Метод возвращает объект DocumentAPI.PivotTableEditor, который служит для обновления свойств и редактирования сводной таблицы.

Пример:

```
local cell = tbl:getCell("L8")
local pivotTable = cell:getPivotTable()
local pivotTableEditor = pivotTable:createPivotTableEditor()
```

7.73 Таблица DocumentAPI.PivotTableCaptions

Таблица DocumentAPI.PivotTableCaptions хранит все пользовательские заголовки сводной таблицы. Описание полей таблицы представлено в таблице 42.

Таблица 42 – Описание полей таблицы DocumentAPI.PivotTableCaptions

Поле	Описание
PivotTableCaptions.errorCaption	Алиас для значений, которые возвращают ошибку.
PivotTableCaptions.emptyCaption	Алиас для значений, которые возвращают пустое значение.

Поле	Описание
PivotTableCaptions.grandTotalCaption	Алиас общих итогов.
PivotTableCaptions.valuesHeaderCaption	Алиас поля из области значений; это поле отображается в отчете в случае, если в сводной таблице наличие более двух полей из области значений, и макет имеют тип ‘outline’ или ‘tabular’.
PivotTableCaptions.rowHeaderCaption	Алиас заголовка строк (виден только при включенном компактном макете, это алиас по умолчанию).
PivotTableCaptions.columnHeaderCaption	Алиас заголовка колонок (виден только при включенном компактном макете, это алиас по умолчанию).

7.74 Таблица DocumentAPI.PivotTableCategoryField

DocumentAPI.PivotTableCategoryField содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. таблицу 43). Таблица может быть получена посредством вызовов PivotTable:getRowFields(), PivotTable:getColumnFields().

Таблица 43 – Описание полей таблицы DocumentAPI.PivotTableCategoryField

Поле	Описание
PivotTableCategoryField .fieldProperties	Свойства поля PivotTableFieldProperties
PivotTableCategoryField .subtotalFunctions	Список функций PivotTableFunction для вычисления подытога

7.75 Таблица DocumentAPI.PivotTableEditor

Предназначена для редактирования сводных таблиц. Возвращается посредством метода PivotTable:createPivotTableEditor().

7.75.1 Метод PivotTableEditor:addField

Метод добавляет новое поле в сводную таблицу, используя параметры:

– fieldName - имя поля;

– toCategory - категория поля (тип - DocumentAPI.PivotTableFieldCategory);

– index - позиция в категории.

Метод возвращает объект DocumentAPI.PivotTableEditor.

Пример:

```
pivotTableEditor = pivotTableEditor:addField("CC",
DocumentAPI.PivotTableFieldCategory_Values)
pivotTableEditor:apply()
```

7.75.2 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 категориии.

Метод возвращает объект `DocumentAPI.PivotTableEditor`.

Пример:

```
pivotTableEditor = pivotTableEditor:moveField("BB",
DocumentAPI.PivotTableFieldCategory_Values, 0)
pivotTableEditor:apply()
```

7.75.3 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()
```

7.75.4 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()
```

7.75.5 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()
```

7.75.6 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()
```

7.75.7 Метод PivotTableEditor:setSummarizeFunction

Метод задает суммирующую функцию для поля из области значений.

Параметры:

– `valueFieldName` - имя поля (тип - строка);

– summarizeFunction – суммирующая функция, тип - DocumentAPI.PivotTableFunction.

Метод возвращает объект DocumentAPI.PivotTableEditor.

Пример:

```
pivotTableEditor = pivotTableEditor:setSummarizeFunction("Age",
DocumentAPI.PivotTableFunction_Sum)
pivotTableEditor:apply()
```

7.75.8 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()
```

7.75.9 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:setFilters(filters)
pivotTableEditor:apply()
```

7.75.10 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()
```

7.75.11 Метод PivotTableEditor:setLayoutSettings

Метод устанавливает настройки отображения DocumentAPI.PivotTableLayoutSettings сводной таблицы, возвращает объект DocumentAPI.PivotTableEditor.

Пример:

```
local layoutSettings = pivotTable:getPivotTableLayoutSettings()
layoutSettings.reportLayout = DocumentAPI.PivotTableReportLayout_Tabular

local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor = pivotTableEditor:setLayoutSettings(layoutSettings)
pivotTableEditor:apply()
```

7.75.12 Метод PivotTableEditor:setGrandTotalSettings

Метод задает настройки отображения общего итога.

Параметры:

– `isRowGrandTotalEnabled` – показывать общие итоги для строк;

– `isColGrandTotalEnabled` – показывать общие итоги для столбцов.

Пример:

```
local pivotTableEditor = pivotTable:createPivotTableEditor()
pivotTableEditor:setGrandTotalSettings(true, true)
```

7.75.13 Метод PivotTableEditor:apply

Метод обновляет сводную таблицу с заданными свойствами и возвращает результат DocumentAPI.PivotTableUpdateResult.

Пример:

```
local pivotTab8leEditor = pivotTable:createPivotTableEditor()
if DocumentAPI.PivotTableUpdateResult_Success == pivotTableEditor:apply() then
    print("Successfully applied");
end
```

7.76 Таблица DocumentAPI.PivotTableField

Таблица `DocumentAPI.PivotTableField` содержит свойства полей сводной таблицы (см. таблицу 44). Таблица может быть получена посредством вызова `PivotTable:getFieldsList()`.

Таблица 44 – Описание полей таблицы `DocumentAPI.PivotTableField`

Поле	Описание
`PivotTableField.fieldProperties`	Свойства полей сводной таблицы `PivotTablFieldProperties`
`PivotTableField.fieldCategories`	Категории полей сводной таблицы `PivotTablFieldCategories`
`PivotTableField.customFormula`	Вычисляемая формула (строка)

7.77 Таблица DocumentAPI.PivotTableFieldCategory

Таблица `DocumentAPI.PivotTableFieldCategory` описывает флаги, которые задают категорию области полей. Описание полей таблицы представлено в таблице 45.

Таблица 45 – 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

7.78 Таблица DocumentAPI.PivotTableFieldCategories

Класс обеспечивает доступ к списку категорий поля сводной таблицы. Может быть получена посредством использования метода `PivotTable.getFieldCategories()`.

7.78.1 Метод PivotTableFieldCategories:enumerate

Метод для перечисления категорий поля DocumentAPI.PivotTableFieldCategory.

Пример:

```
local fieldCategories = pivotTable:getFieldCategories("Age")
for fieldCategory in fieldCategories:enumerate() do
   print(fieldCategory)
end
```

7.79 Таблица DocumentAPI.PivotTableFieldProperties

DocumentAPI.PivotTableFieldProperties содержит свойства поля DocumentAPI.PivotTableField сводной таблицы (см. таблицу 46).

Таблица 46 – Описание полей таблицы DocumentAPI.PivotTableFieldProperties

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

7.80 Таблица DocumentAPI.PivotTableFilter

Позволяет осуществить доступ к списку фильтров таблицы, каждый из которых обладает свойством видимости (Рис. 39).

. При любом изменении фильтров они должны быть применены к сводной таблице посредством использования методов PivotTableEditor.setFilter(), PivotTableEditor.setFilters().

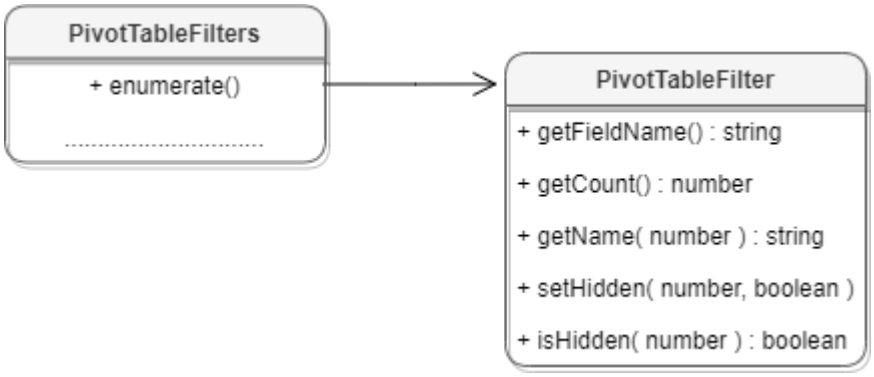


Рисунок 39 – Таблица DocumentAPI.PivotTableFilter

Пример:

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      filter:setHidden(filterIdx, false)
   end
end
pivotTableEditor:setFilters(filters)
pivotTableEditor:apply()
```

7.80.1 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
```

7.80.2 Метод PivotTableFilter:getCount

Возвращает количество фильтруемых полей.

Пример:

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   print(filter:getCount())
end
```

7.80.3 Метод PivotTableFilter:getName

Возвращает имя поля для заданного индекса.

Пример:

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      print(filter:getName(filterIdx))
```

```
   end
end
```

7.80.4 Метод PivotTableFilter:isHidden

Возвращает видимость поля для заданного индекса `itemIndex`. Если `true`, то поле скрыто.

Пример:

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   for filterIdx = 0, filter:getCount() - 1 do
      print(filter:isHidden(filterIdx))
   end
end
```

7.80.5 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
```

7.81 Таблица DocumentAPI.PivotTableFilters

Таблица обеспечивает доступ к списку фильтров. Для получения `DocumentAPI.PivotTableFilters` используется метод `PivotTable.getFilters()`.

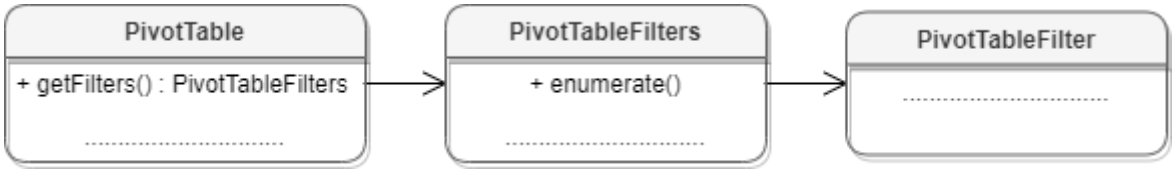


Рисунок 40 – Объектная модель таблиц для работы с фильтрами

7.81.1 Метод PivotTableFilters:enumerate

Метод используется для доступа к коллекции фильтров (см. DocumentAPI.PivotTableFilter).

Пример:

```
local filters = pivotTable:getFilters()
for filter in filters:enumerate() do
   print(filter:getName(0))
   print(filter:getFieldName())
end
```

7.82 Таблица DocumentAPI.PivotTableFunction

Таблица DocumentAPI.PivotTableFunction описывает функции, которые могут быть использованы в сводных таблицах. Описание полей таблицы представлено в таблице 47. Таблица используется в качестве поля subtotalFunctions таблицы DocumentAPI.PivotTableCategoryField.

Таблица 47 – Описание полей таблицы DocumentAPI.PivotTableFunction

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

7.83 Таблица DocumentAPI.PivotTableItem

`DocumentAPI.PivotTableItem` описывает элемент сводной таблицы (см. Рисунок 41). См. пример в главе `PivotTableItems:enumerate`.

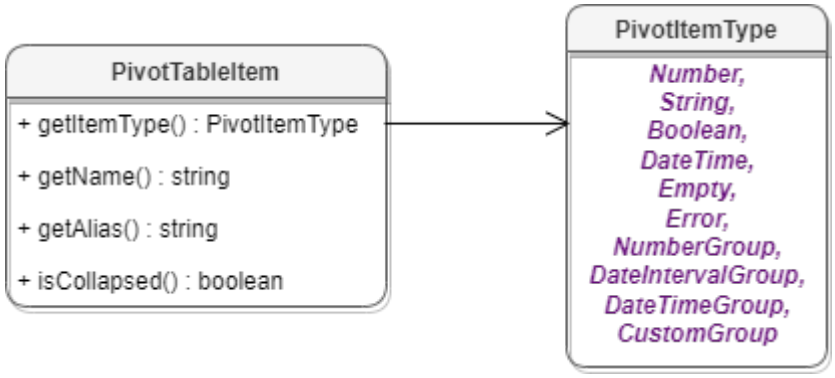


Рисунок 41 – Таблица `DocumentAPI.PivotTableItem`

7.83.1 Метод PivotTableItem:getName

Метод возвращает имя элемента сводной таблицы, тип - строка. См. пример в главе `PivotTableItems:enumerate`.

7.83.2 Метод PivotTableItem:getAlias

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем), тип - строка. См. пример в главе `PivotTableItems:enumerate`.

7.83.3 Метод PivotTableItem:getItemType

Метод возвращает тип `DocumentAPI.PivotTableItemType` элемента сводной таблицы. См. пример в главе `PivotTableItems:enumerate`.

7.83.4 Метод PivotTableItem:isCollapsed

Метод возвращает `true`, если элемент сводной таблицы свернут. См. пример в главе `PivotTableItems:enumerate`.

7.84 Таблица DocumentAPI.PivotTableItems

Таблица обеспечивает доступ к списку элементов сводной таблицы (см. Рисунок 42).

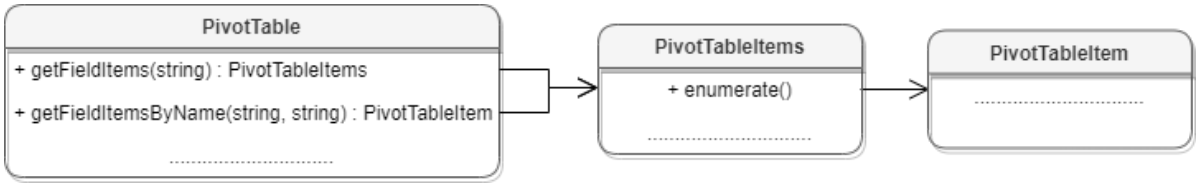


Рисунок 42 – Объектная модель таблиц для работы с элементами сводных таблиц

7.84.1 Метод PivotTableItems:enumerate

Используется для перечисления элементов сводной таблицы.

Пример:

```
local fieldItems = pivotTable:getFieldItems("Age")
for fieldItem in fieldItems:enumerate() do
   print(fieldItem:getName())
   print(fieldItem:getAlias())
   print(fieldItem:getItemType())
   print(fieldItem:isCollapsed())
end
```

7.85 Таблица DocumentAPI.PivotTableItemType

Таблица `DocumentAPI.PivotTableItemType` содержит возможные типы элементов сводной таблицы. Описание полей таблицы представлено в таблице 48.

Таблица 48 – Описание полей таблицы `DocumentAPI.PivotTableItemType`

Поле	Описание
`DocumentAPI.PivotTableItemType_Number`	Числовой
`DocumentAPI.PivotTableItemType_String`	Строковый
`DocumentAPI.PivotTableItemType_Boolean`	Логический
`DocumentAPI.PivotTableItemType_DateTime`	Дата / время
`DocumentAPI.PivotTableItemType_Empty`	Пустой тип
`DocumentAPI.PivotTableItemType_Error`	Ошибка
`DocumentAPI.PivotTableItemType_NumberGroup`	Интервальная группировка
`DocumentAPI.PivotTableItemType_DateIntervalGroup`	Интервальная группировка по датам
`DocumentAPI.PivotTableItemType_DateTimeGroup`	Группировка по дате / времени

Поле	Описание
DocumentAPI.PivotTableItemType_CustomGroup	Пользовательская (произвольная) группировка

Пример:

```
local fieldItems = pivotTable:getFieldItems("Age")
for fieldItem in fieldItems:enumerate() do
   if (fieldItem:getItemType() == DocumentAPI.PivotTableItemType_Number) then
      print("Numeric type")
   end
end
```

7.86 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 49.

Таблица 49 – Описание полей таблицы DocumentAPI.PivotTableLayoutSettings

Поле	Описание
PivotTableLayoutSettings.reportLayout	Настройка вида макета сводной таблицы (PivotTableReportLayout: компактный, табличный, структурный).
PivotTableLayoutSettings.valueFieldsOrientation	Настраивает положение значений в случае, если в сводной таблице более двух полей значений. Тип - ValueFieldsOrientation.
PivotTableLayoutSettings.pageFieldOrder	Настройка порядка полей фильтров (PageFieldOrder: вниз, затем поперек или сначала поперек, потом вниз).
PivotTableLayoutSettings.indentForCompactLayout	Размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей).
PivotTableLayoutSettings.pageFieldWrapCount	Настройка связана с pageFieldOrder, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д).
PivotTableLayoutSettings.isMergeAndCenterLabelsEnabled	Настройка позволяет объединить ячейки заголовков.

Поле	Описание
PivotTableLayoutSettings.useGridDropZones	Флаг, отвечающий за отображение классического вида (как в Excel 2003). Влияет только на расположение полей в отчете.
PivotTableLayoutSettings.displayFieldCaptions	Флаг, отвечающий за отображение заголовков полей.

7.87 Таблица DocumentAPI.PivotTablePageField

Содержит свойства поля из области фильтров (см. таблицу 50). Таблица может быть получена посредством вызова PivotTable:getPageFields().

Таблица 50 – Описание полей таблицы DocumentAPI.PivotTablePageField

Поле	Описание
PivotTablePageField.fieldProperties	Свойства поля PivotTableFieldProperties

7.88 Таблица DocumentAPI.PivotTableReportLayout

Таблица DocumentAPI.PivotTableReportLayout описывает внешний вид отчетов сводной таблицы. Является полем таблицы DocumentAPI.PivotTableLayoutSettings. Описание полей таблицы представлено в таблице 51.

Таблица 51 – Описание полей таблицы DocumentAPI.PivotTableReportLayout

Поле	Описание
DocumentAPI.PivotTableReportLayout_Compact	Компактный вид
DocumentAPI.PivotTableReportLayout_Tabular	Табличный вид
DocumentAPI.PivotTableReportLayout_Outline	Структурный вид

7.89 Таблица DocumentAPI.PivotTablesManager

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

Пример:

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

7.89.1 Метод PivotTablesManager:create

Метод создает сводную таблицу PivotTable на основе диапазона исходных данных CellRange.

Если местоположение не задано, создается новый лист (таблица), и сводная таблица будет расположена по умолчанию.

Пример:

```
local pivotTablesManager = document:getPivotTablesManager()
local tbl = document:getBlocks():getTable(0)
local cellRange = tbl:getCellRange("I3:K7")
local pivotTable = pivotTablesManager:create(cellRange, tbl:getCell("L8"))
```

7.90 Таблица DocumentAPI.PivotTableUnsupportedFeature

Таблица `DocumentAPI.PivotTableUnsupportedFeature` описывает неподдерживаемую функциональность сводных таблиц. Их получение описано в `PivotTable:getUnsupportedFeatures()`. Описание полей таблицы представлено в таблице 52.

Таблица 52 – 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)

7.91 Таблица DocumentAPI.PivotTableValueField

`DocumentAPI.PivotTableValueField` содержит свойства поля сводной таблицы, использующегося как значение столбец (см. таблицу 53). Таблица может быть получена посредством вызова `PivotTable:getValueFields()`.

Таблица 53 – Описание полей таблицы `DocumentAPI.PivotTableValueField`

Поле	Описание
`PivotTableValueField baseFieldName`	Оригинальное поле на основе которого было создано данное поле, тип - строка.
`PivotTableValueField valueFieldName`	Автоматический уникальный псевдоним такой как “Sum of %имя поля%“, тип – строка.

Поле	Описание
`PivotTableValueField cellNumberFormat`	Числовой формат типа `CellFormat` для конкретного поля значений.
`PivotTableValueField totalFunction`	Агрегирующая функция `PivotTableFunction` поля значений (SUM, COUNT, MAX и т.д).
`PivotTableValueField customFormula`	Вычисляемая формула для поля значений, тип - строка.

7.92 Таблица DocumentAPI.PivotTableUpdateResult

В таблице 54 приведены константы, которые соответствуют возможным результатам обновления сводной таблицы (см. методы `PivotTable.update()`, `PivotTableEditor:apply()`).

Таблица 54 – Результаты обновления сводной таблицы

Наименование константы	Описание
`DocumentAPI.PivotTableUpdateResult_Success`	Успешное обновление таблицы
`DocumentAPI.PivotTableUpdateResult_NoPivotTable`	Сводная таблица не найдена
`DocumentAPI.PivotTableUpdateResult_NoSuchFieldInCategory`	Не найдено поле в категории
`DocumentAPI.PivotTableUpdateResult_NoSuchFieldInPivotTable`	Не найдено поле в сводной таблице
`DocumentAPI.PivotTableUpdateResult_InvalidIndex`	Ошибка в индексе
`DocumentAPI.PivotTableUpdateResult_FieldAlreadyEnabled`	Поле уже существует
`DocumentAPI.PivotTableUpdateResult_MovingFieldToTheSameCategoryForbidden`	Попытка перемещения поля в рамках текущей категории
`DocumentAPI.PivotTableUpdateResult_InvalidFunction`	Неправильная функция
`DocumentAPI.PivotTableUpdateResult_InvalidCategory`	Неправильная область
`DocumentAPI.PivotTableUpdateResult_InvalidDataSourceRange`	Ошибка диапазона исходных данных
`DocumentAPI.PivotTableUpdateResult_NoDataRowsInDataSource`	В исходных данных нет строк с данными

Наименование константы	Описание
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 удается раскрыть свернутый элемент
DocumentAPI.PivotTableUpdateResult_AnotherPivotInsideDataSource	Найдена другая сводная таблица в этом же диапазоне
DocumentAPI.PivotTableUpdateResult_Canceled	Обновление сводной таблицы отменено

7.93 Таблица DocumentAPI.PointU

Таблица `DocumentAPI.PointU` представляет позицию объекта (x, y). 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)
```

7.93.1 Метод PointU:toString

Возвращает информацию о позиции в виде строкового значения формата `(width: <value>, height: <value>)`.

Пример:

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

7.94 Таблица DocumentAPI.Position

Таблица `DocumentAPI.Position` представляет местоположение в текстовом

документе. Используется для обозначения начала и конца диапазона `DocumentAPI.Range`.

7.94.1 Метод Position:getCell

Метод возвращает ячейку, в которой находится позиция, либо `nil` если позиция не находится в ячейке.

Пример для табличного документа:

```
local table = document:getBlocks():getTable(0)
local cell = table:getCell("A1")
local range = cell:getRange()
local pos = range:getBegin()
print(pos:getCell())
```

Пример для текстового документа:

```
local range = document:getRange()
local pos = rng:getBegin()
print(pos:getCell())
```

7.94.2 Метод Position:insertText

Метод предназначен для вставки текстовой строки в заданное местоположение в документе.

Пример:

```
local rng = document:getRange()
local begin_pos = rng:getBegin()
begin_pos:insertText("Текст в начале строки")
```

7.94.3 Метод Position:insertTable

Метод предназначен для вставки таблицы с заданным числом строк и столбцов в заданное местоположение в документе. Возвращает объект таблицы.

Следует учитывать, что при вставке таблицы к ее имени автоматически добавляется порядковый номер, начинающийся с единицы. Таким образом, вызов

```
t = position:insertTable(3, 3, "Table")
```

приведет к созданию в текстовом документе таблицы с именем «Table1».

Пример вставки таблицы в начало текстового документа:

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

Пример вставки таблицы в конец текстового документа:

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

В табличном документе данный метод используется для вставки нового рабочего листа.

Пример вставки нового листа в табличный документ:

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

7.94.4 Метод Position:insertPageBreak

Метод предназначен для вставки разрыва страницы в указанную позицию текстового документа.

Внимание ! Метод может быть использован только в текстовом редакторе.

Пример:

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

7.94.5 Метод Position:insertLineBreak

Метод предназначен для вставки перевода строки в указанную позицию текстового документа.

Внимание ! Метод может быть использован только в текстовом редакторе.

Пример:

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

7.94.6 Метод Position:insertBookmark

Вставляет закладку с наименованием в заданную позицию текстового документа.

Внимание ! Метод может быть использован только в текстовом редакторе.

Пример:

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

7.94.7 Метод Position:insertSectionBreak

Вставляет разрыв раздела в текущую позицию текстового документа.

Внимание ! Метод может быть использован только в текстовом редакторе.

Пример:

```
document:getRange():getBegin():insertSectionBreak()
```

7.94.8 Метод Position:insertHyperlink

Метод `insertHyperlink` вставляет ссылку в текущую позицию. В качестве параметров передаются адрес ссылки и текст ссылки.

Вызов:

```
insertHyperlink( url, size )
```

Параметры:

– `url` – адрес ссылки;

– `label` – текст ссылки.

Пример:

```
document:getRange():getBegin():insertHyperlink("https://testhyperlink.com",
"Hyperlink")
```

7.94.9 Метод Position:insertImage

Вставляет изображение в позицию текстового документа.

Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

Вызов:

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

Параметры:

– `url` – полный путь к локальному файлу, либо ссылка на сетевой ресурс;

– `size` – геометрические размеры изображения для вставки.

Примеры:

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

```
document:getRange():getBegin():insertImage
("https://www.images.ru/images/fish.jpg", DocumentAPI.SizeU(50, 50))
```

7.94.10 Метод Position:removeBackward

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

Пример:

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

7.94.11 Метод Position:removeForward

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

Пример:

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

7.94.12 Метод Position:__eq

Метод используется для определения эквивалентности значений двух местоположений в документе.

Пример:

```
print(document:getRange():getBegin():__eq(document:getRange():getEnd()))
```

7.95 Таблица DocumentAPI.PrintSettings

Таблица `DocumentAPI.PrintSettings` представляет установки, используемые при печати документов. Описание полей таблицы `DocumentAPI.PrintSettings` представлено в таблице 56. Используется в `EditorAPI.printDocument`.

Таблица 56 – Описание полей таблицы `DocumentAPI.PrintSettings`

Поле	Тип	Описание
`DocumentAPI.PrintSettings.printerName`	string	Имя используемого принтера. Если не указано, то используется принтер по умолчанию. Если принтер с указанным именем недоступен, то возникает ошибка.
`DocumentAPI.PrintSettings.landscapeOrientation`	bool	Если значение равно `true`, то размер страницы поворачивается на 90 градусов. В настоящее время используется только для рабочих таблиц.
`DocumentAPI.PrintSettings.leftMargin`	number	Ширина левого поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
`DocumentAPI.PrintSettings.topMargin`	number	Ширина верхнего поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
`DocumentAPI.PrintSettings.rightMargin`	number	Ширина правого поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
`DocumentAPI.PrintSettings.bottomMargin`	number	Ширина нижнего поля. Размер указывается в типографских точках. В настоящее время используется только для рабочих таблиц.
`DocumentAPI.PrintSettings.parity`	`PageParity`	Выбор страниц для печати .
`DocumentAPI.PrintSettings.firstPage`	number	Номер первой страницы для печати.

Поле	Тип	Описание
DocumentAPI.PrintSettings.lastPage	number	Номер последней страницы для печати.
DocumentAPI.PrintSettings.printSelection	bool	Область печати. Значение по умолчанию false.
DocumentAPI.PrintSettings.worksheetPrinterFitType	WorksheetPrinterFitType	Вариант масштабирования при печати табличных документов.
DocumentAPI.PrintSettings.copies	number	Количество копий при печати.
DocumentAPI.PrintSettings.collateCopies	bool	Если параметр имеет значение false, то печать каждой отдельной страницы будет повторена заданное количество копий раз до начала печати следующей страницы. Если параметр имеет значение true, то все страницы печатаются до запуска печати следующей копии этих страниц. Значение по умолчанию false.

7.96 Таблица DocumentAPI.PrintDocumentResult

В таблице 57 представлены коды, возвращаемые после печати (см. EditorAPI.showPrintDialog()).

Таблица 57 – Коды, возвращаемые после печати

Наименование константы	Описание
DocumentAPI.PrintDocumentResult_Success	Печать прошла успешно
DocumentAPI.PrintDocumentResult_OneCopyPrinted	Напечатана только одна копия из заданных
DocumentAPI.PrintDocumentResult_CancelPrinting	Печать была отменена
DocumentAPI.PrintDocumentResult_NoPrinter	Принтер не найден
DocumentAPI.PrintDocumentResult_BlankDocument	На печать отправлен пустой документ

7.97 Таблица DocumentAPI.Range

Таблица DocumentAPI.Range предоставляет доступ к диапазону документа. На рисунке 43 изображена объектная модель таблиц, относящихся к работе с диапазонами.

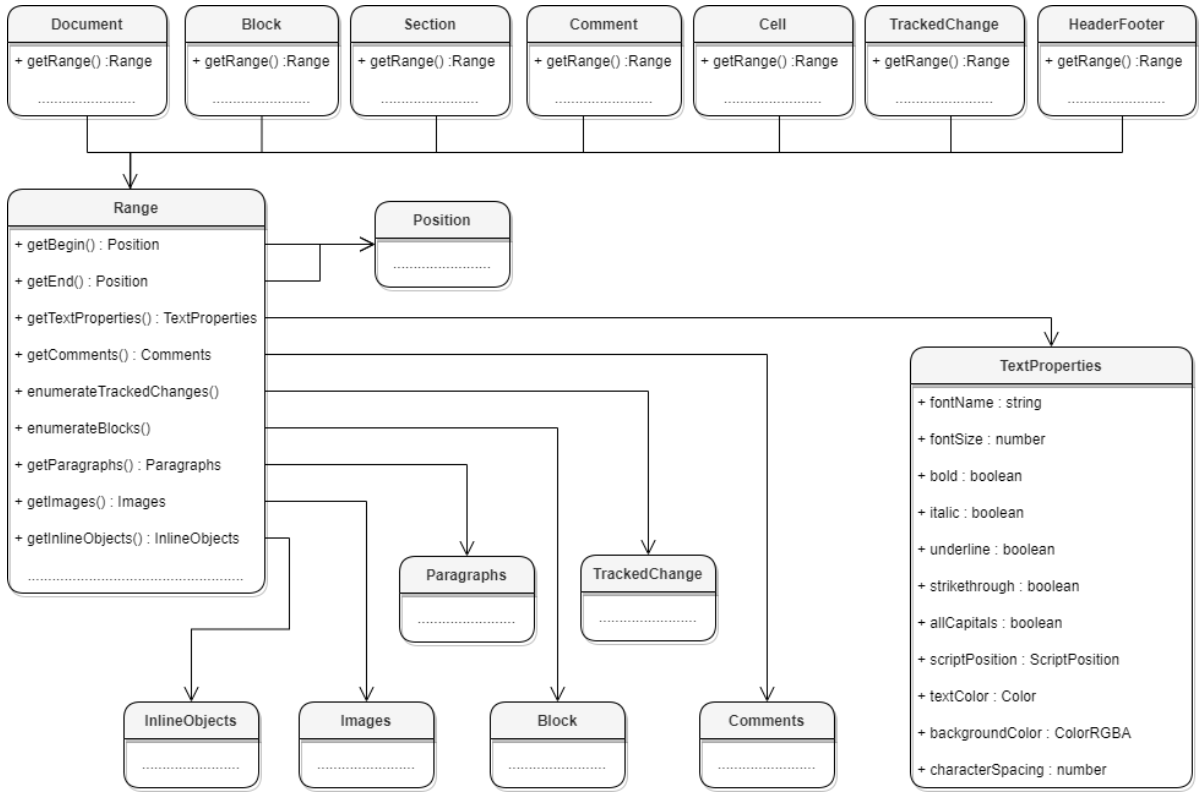


Рисунок 43 – Объектная модель для работы с таблицей `DocumentAPI.Range`

Варианты получения диапазона для текстового документа:

```
-- диапазон всего документа
documentRange = document:getRange()
-- диапазон блока
block = document:getBlocks():getBlock(0)
blockRange = block:getRange()
-- диапазон секций
sections = document:getSections()
for section in sections:enumerate() do
    sectionRange = section:getRange()
end
-- диапазон комментариев
commentsList = document:getRange():getComments()
for comment in commentsList:enumerate() do
    commentRange = comment:getRange()
end
-- диапазон ячейки
table = document:getBlocks():getTable(0)
cell = table:getCell("B2")
cellRange = cell:getRange()
```

```
-- диапазон верхних колонтитулов
section = document:getBlocks():getBlock(0):getSection()
headers = section.getHeaders()
for header in headers:enumerate() do
    headerRange = header:getRange()
end
-- диапазон отслеживаемых изменений
local trackedChangesList = document:getRange():enumerateTrackedChanges()
for trackedChange in trackedChangesList do
    trackedChangeRange = trackedChange:getRange()
end
```

7.97.1 Метод Range:getBegin

Метод возвращает позицию в начале диапазона.

Пример для текстового документа:

```
local range = document:getRange() -- содержимое всего документа
local pos = range:getBegin() -- в начало документа
pos:insertText("Привет")
```

Пример для табличного документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки B2
local pos = range:getBegin() -- в начало ячейки
pos:insertText("Привет")
```

7.97.2 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("Привет")
```

Пример для табличного документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки B2
```

```
local pos = range:getEnd() -- в конец ячейки
pos:insertText("Привет")
```

7.97.3 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())
```

7.97.4 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())
```

7.97.5 Метод Range:lockContent

Метод запрещает изменения содержимого диапазона.

Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа:

```
local range = document:getRange() -- содержимое всего документа
range:lockContent()
```

Пример для таблицы внутри текстового документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:lockContent()
```

7.97.6 Метод Range:unlockContent

Метод разрешает изменения содержимого диапазона.

Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа:

```
local range = document:getRange() -- содержимое всего документа
range:unlockContent()
```

Пример для таблицы внутри текстового документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки
range:unlockContent()
```

7.97.7 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
```

7.97.8 Метод Range:replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример для текстового документа:

```
local range = document:getRange() -- содержимое всего документа
range:replaceText("Новый текст")
```

Пример для табличного документа:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
local range = cell:getRange() -- содержимое ячейки таблицы
range:replaceText("Новый текст")
```

7.97.9 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())
```

7.97.10 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)
```

7.97.11 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)
```

7.97.12 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
```

7.97.13 Метод Range:enumerateTrackedChanges

Предоставляет возможность итерации по отслеживаемым изменениям DocumentAPI.TrackedChange. Метод может быть использован только в текстовых документах.

Пример:

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

7.97.14 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())
  print(comment:getReplies())
end
```

7.97.15 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
```

7.97.16 Метод Range:getImages

Обеспечивает доступ к изображениям (DocumentAPI.Image) в диапазоне.

Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

Примеры:

```
local images = document:getRange():getImages()
for image in images:enumerate() do
  print(image:getFrame():getWrapType())
end
```

```
for image in EditorAPI.getSelection():getImages():enumerate() do
  print(image:getFrame():getWrapType())
end
```

7.97.17 Метод Range:getInlineObjects

Обеспечивает доступ к перечислению DocumentAPI.MediaObjects графических объектов диапазона.

Внимание ! В текущей версии метод может быть использован только в текстовом редакторе.

Пример:

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

7.98 Таблица DocumentAPI.RangeBorders

Таблица DocumentAPI.RangeBorders оставлена для совместимости. Вместо нее необходимо использовать таблицу DocumentAPI.Borders.

7.99 Таблица DocumentAPI.RectU

Таблица DocumentAPI.RectU представляет описание прямоугольной области. Описание полей таблицы DocumentAPI.RectU представлено в таблице 58.

Таблица 58 – Описание полей таблицы DocumentAPI.RectU

Поле	Описание
DocumentAPI.RectU.topLeft	Координаты левого верхнего угла области, тип CellPosition
DocumentAPI.RectU.rightBottom	Координаты правого нижнего угла области, тип CellPosition

Пример:

```
local rect = DocumentAPI.RectU(2, 3, 4, 5)
print("tlx=", rect.topLeft.x, ", tly=", rect.topLeft.y, ", rbx=",
rect.bottomRight.x, ", rby=", rect.bottomRight.y)  --(tlx = 2.0, tly = 3.0, brx
= 4.0, bry = 5.0)
```

7.99.1 Метод RectU:toString

Возвращает информацию о прямоугольной области в виде строкового описания координат [topLeft: (x: <value>, y: <value>), bottomRight: (x: <value>, y:

<value>)].

Пример:

```
local point = DocumentAPI.RectU(2, 3, 4, 5)
print(point:toString())  --[topLeft: (x: 2.0, y: 3.0), bottomRight: (x: 4.0, y: 5.0)]
```

7.100 Таблица DocumentAPI.ScaleFrom

В таблице 59 представлены позиции объекта, остающиеся неизменными при масштабировании объекта. Используется в `AbsoluteFrame.scale()`.

Таблица 59 – Неизменные позиции объекта при масштабировании

Наименование константы	Позиция
`DocumentAPI.ScaleFrom_BottomRight`	Правый нижний угол
`DocumentAPI.ScaleFrom_BottomLeft`	Левый нижний угол
`DocumentAPI.ScaleFrom_TopLeft`	Левый верхний угол
`DocumentAPI.ScaleFrom_TopRight`	Правый верхний угол

7.101 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 60.

Таблица 60 – Описание полей таблицы `DocumentAPI.ScientificCellFormatting`

Поле	Описание
`DocumentAPI.ScientificCellFormatting.decimalPlaces`	Количество десятичных позиций
`DocumentAPI.ScientificCellFormatting.minExponentDigits`	Минимальное количество позиций экспоненты

Пример:

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

local scientificCellFormatting = DocumentAPI.ScientificCellFormatting()
scientificCellFormatting.decimalPlaces = 2
scientificCellFormatting.minExponentDigits = 3
```

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

7.102 Таблица DocumentAPI.Script

Таблица `DocumentAPI.Script` предназначена для управления отдельной макрокомандой. Таблица содержит поля `Name` и `Body`.

7.102.1 Таблица DocumentAPI.Scripting

Таблица `DocumentAPI.Scripting` может быть получена путем вызова `DocumentAPI.createScripting()` и содержит метод `runScript`, который используется для запуска макрокоманды.

7.102.1.1 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")
```

7.102.2 Метод Script:getName

Метод возвращает имя макрокоманды.

Пример:

```
local scripts = document:getScripts()
local sc = scripts:getScript("Enumerate scripts for document")
print(sc:getName())
```

7.102.3 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")
```

7.102.4 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)
```

7.102.5 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
scripts:enumerate() do\nprint(script:getName())\nend")
```

7.103 Таблица DocumentAPI.ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 61. Используется в качестве поля `scriptPosition` таблицы `DocumentAPI.TextProperties`.

Таблица 61 – Типы надстрочного и подстрочного форматирования

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

Пример:

```
local props = DocumentAPI.TextProperties()
props.scriptPosition = DocumentAPI.ScriptPosition_SuperScript
range:setTextProperties(props)
```

7.104 Таблица DocumentAPI.Scripts

Таблица `DocumentAPI.Scripts` предоставляет доступ к списку макрокоманд документа. Коллекцию макрокоманд `DocumentAPI.Scripts` можно получить из документа посредством вызова метода `document:getScripts()`.

Пример:

```
local scripts = document:getScripts()
for script in scripts:enumerate() do
   print(script:getName())
   print(script:getBody())
end
```

7.104.1 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())
```

7.104.2 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
scripts:enumerate() do\nprint(script:getName())\nend"
scripts:setScript(script_name, script_code)
```

7.104.3 Метод Scripts:removeScript

Метод удаляет макрокоманду из текущего документа. В качестве аргумента используется имя макрокоманды.

Пример:

```
local scripts = document:getScripts()
scripts:removeScript("Enumerate scripts for document")
```

7.104.4 Метод Scripts:enumerate

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

Пример:

```
for script in document:getScripts():enumerate() do
    print(script:getName())
end
```

7.105 Таблица DocumentAPI.Search

Таблица `DocumentAPI.Search` предоставляет доступ к механизму поиска фрагментов документа, открытого в редакторе текста или таблиц.

7.105.1 Метод Search:findText

Метод выполняет поиск строки без учета регистра во всем документе или выбранном диапазоне документа. Результат возвращается в виде диапазона `DocumentAPI.Range`, содержащего искомый фрагмент.

Если строка не обнаружена, возвращается пустая таблица.

Возможно использование следующих вариантов метода:

```
Range findText(String text)
Range findText(String text, CaseSensitive caseSensitive)
Range findText(String text, Range range)
Range findText(String text, Range range, CaseSensitive caseSensitive)
Range findText(String text, CellRange cellRange)
Range findText(String text, CellRange cellRange, CaseSensitive caseSensitive)
Range findText(String text, Table tbl)
Range findText(String text, Table tbl, CaseSensitive caseSensitive)
```

Параметры:

- `text` – строка для поиска;
- `caseSensitive` – поиск с учетом или без учета регистра, тип `DocumentAPI.CaseSensitive`;
- `range` – диапазон, в котором будет производиться поиск, тип `DocumentAPI.Range`;
- `cellRange` – диапазон ячеек, в котором будет производиться поиск, тип `DocumentAPI.CellRange`;
- `table` – таблица, в которой будет производиться поиск, тип `DocumentAPI.Table`.

Пример:

```
search = DocumentAPI.createSearch(document)
-- Поиск по всему документу
```

```
ranges = search:findText("English")
for occurrence in ranges do
    print(occurrence:extractText())
end
```

Дополнительные примеры использования метода `Search:findText` приведены в разделе Поиск в документе.

7.106 Таблица DocumentAPI.Section

Таблица `DocumentAPI.Section` представляет собой раздел в документе.

7.106.1 Метод Section:setPageProperties

Метод устанавливает параметры `DocumentAPI.PageProperties` страниц, находящихся в разделе.

Пример:

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

7.106.2 Метод Section:getPageProperties

Метод возвращает параметры страниц раздела `DocumentAPI.PageProperties`.

Пример:

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

7.106.3 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)
```

7.106.4 Метод Section:getPageOrientation

Метод возвращает ориентацию страниц раздела.

Пример:

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

7.106.5 Метод Section:getRange

Метод возвращает диапазон DocumentAPI.Range в документе, соответствующий данному разделу.

Пример:

```
local sections = document:enumerateSections()
for section in sections do
  print(section:getRange():extractText())
end
```

7.106.6 Метод Section:getHeaders

Метод возвращает коллекцию DocumentAPI.HeadersFooters верхних колонтитулов данного раздела.

Пример:

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

7.106.7 Метод Section:getFooters

Метод возвращает коллекцию DocumentAPI.HeadersFooters нижних колонтитулов данного раздела.

Пример:

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

7.107 Таблица DocumentAPI.Sections

Таблица `DocumentAPI.Sections` представляет интерфейс для доступа к коллекции секций документа. Может быть получена посредством вызова метода `document::getSections()`. Описание секции см. в разделе `DocumentAPI.Section`.

7.107.1 Метод Sections:enumerate

Метод возвращает коллекцию секций документа.

Пример:

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

7.108 Таблица DocumentAPI.Shape

Таблица `Shape` представляет собой фигуру, содержит методы для установки и получения ее свойств `DocumentAPI.ShapeProperties`.

7.108.1 Метод Shape:getShapeProperties

Метод возвращает свойства фигуры `DocumentAPI.ShapeProperties`.

Пример:

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

7.108.2 Метод Shape:setShapeProperties

Метод устанавливает свойства фигуры DocumentAPI.ShapeProperties.

Пример:

```
local shape = document:getBlocks():getShape(0)
local shape_properties = shape:getShapeProperties()
shape_properties.verticalAlignment = DocumentAPI.VerticalAlignment_Center
shape:setShapeProperties(shape_properties)
```

7.109 Таблица DocumentAPI.ShapeProperties

Таблица описывает свойства фигуры и содержит следующие поля:

- verticalAlignment - вертикальное выравнивание, тип DocumentAPI.VerticalAlignment;
- borderProperties - свойства границ фигуры, тип DocumentAPI.LineProperties;
- fill - свойства заполнения фигуры, тип DocumentAPI.Fill;
- shapeTextLayout - свойства текста внутри фигуры, тип DocumentAPI.ShapeTextLayout.

7.109.1 Поле ShapeProperties:borderProperties

Поле предназначено для установки свойств границ фигуры DocumentAPI.LineProperties.

7.109.2 Поле ShapeProperties:verticalAlignment

Поле предназначено для установки типа вертикального выравнивания DocumentAPI.VerticalAlignment.

7.109.3 Поле ShapeProperties:fill

Поле предназначено для установки свойств заполнения фигуры DocumentAPI.Fill.

7.109.4 Поле ShapeProperties:shapeTextLayout

Поле предназначено для установки свойств текста внутри фигуры DocumentAPI.ShapeTextLayout.

7.110 Таблица DocumentAPI.ShapeTextLayout

Таблица `DocumentAPI.ShapeTextLayout` описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 62. Используется в таблице `DocumentAPI.ShapeProperties`.

Таблица 62 – Описание полей таблицы `DocumentAPI.ShapeTextLayout`

Поле	Описание
`ShapeTextLayout_DoNotAutoFit`	Размещение текста в фигуре по умолчанию
`ShapeTextLayout_FitShapeExtentToText`	Расширение фигуры под текст
`ShapeTextLayout_FitTextToShape`	Заполнение фигуры текстом

7.111 Таблица DocumentAPI.SizeU

Таблица `DocumentAPI.SizeU` представляет размер объекта в двухмерном пространстве. Описание полей таблицы `DocumentAPI.SizeU` представлено в таблице 63.

Таблица 63 – Описание полей таблицы `DocumentAPI.SizeU`

Поле	Тип	Описание
`DocumentAPI.SizeU.width`	number	Ширина
`DocumentAPI.SizeU.height`	number	Высота

Пример:

```
local size = DocumentAPI.SizeU(2, 3)
print("width=", size.width, ", height=", size.height)  --(width = 2,0, height = 3,0)
```

7.111.1 Метод SizeU:toString

Возвращает информацию о размерах в виде строкового значения формата `(width: <value>, height: <value>)`.

Пример:

```
local size = DocumentAPI.SizeU(2, 3)
print(size:toString())  --(width: 2.0, height: 3.0)
```

7.112 Таблица DocumentAPI.Table

Таблица `DocumentAPI.Table` предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 44).

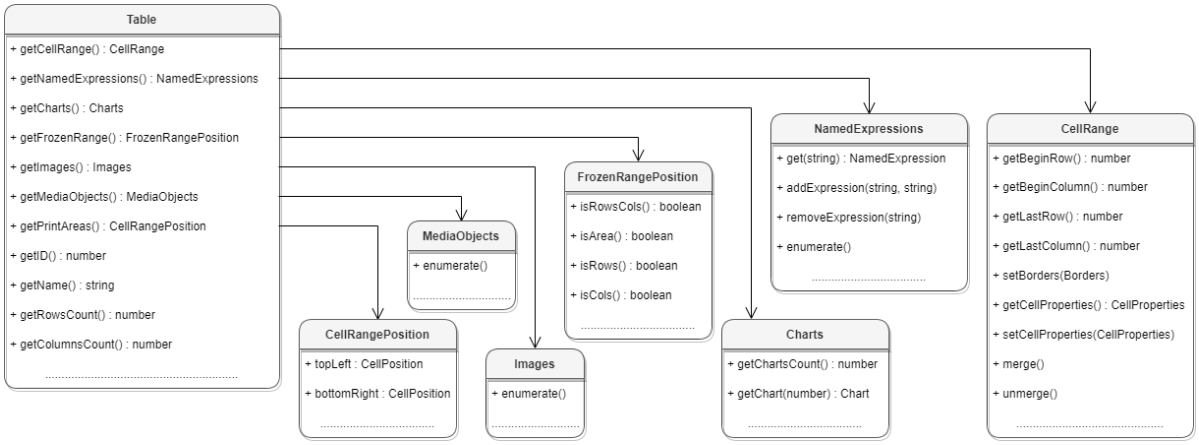


Рисунок 44 – Структура полей таблицы `DocumentAPI.Table`

7.112.1 Метод Table:createFiltersRange

Метод `Table:createFiltersRange` задает диапазон, который используется как диапазон фильтрации.

В качестве параметра используется диапазон ячеек типа `CellRangePosition`. Метод возвращает `FiltersRange`.

Разрешен только один диапазон фильтрации на таблицу. Это означает, что данный метод удаляет ранее определенный диапазон фильтрации. При этом есть исключение: если новый диапазон начинается с той же позиции, предыдущие фильтры будут сохранены.

Диапазон фильтрации должен включать дополнительную строку, которая используется как заголовок таблицы. Эта строка никогда не фильтруется.

Метод может быть использован только в табличных документах.

Пример:

```
local sheet = EditorAPI.getActiveWorksheet()
local cellRange = DocumentAPI.CellRangePosition(1, 1, 8, 2)
local filtersRange = sheet:createFiltersRange(cellRange)
```

Более подробный пример приведен в разделе Работа с фильтрами.

7.112.2 Метод Table:setName

Метод задает имя таблицы. В случае с табличным документом это имя будет являться заголовком листа документа. Данное значение должно быть уникальным, т.к. может использоваться для ссылки на таблицу, например, из формул.

Пример:

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

Для текстовых документов использование данного метода также допустимо, наименование таблицы нигде не отображается, но в дальнейшем его можно использовать для доступа к таблице по имени.

Пример:

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

7.112.3 Метод Table:getName

Метод позволяет получить наименование листа табличного документа.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getName())
```

7.112.4 Метод Table:getFiltersRange

Метод `Table:getFiltersRange` возвращает текущий диапазон фильтрации, принадлежащий таблице. Рабочий лист табличного документа может содержать только один диапазон фильтрации.

Метод возвращает `FiltersRange`, если диапазон фильтрации существует.

Метод может быть использован только в табличных документах.

Пример:

```
local filtersRange = sheet:getFiltersRange()
local cellRange = filtersRange:getCellRange()
print(cellRange:getBeginRow() .. ", " .. cellRange:getLastRow())
```

Более подробный пример приведен в разделе Работа с фильтрами.

7.112.5 Метод Table:getRowsCount

Метод позволяет получить количество строк таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getRowsCount())
```

7.112.6 Метод Table:getColumnsCount

Метод позволяет получить количество столбцов таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getColumnsCount())
```

7.112.7 Метод Table:getCell

Метод позволяет получить доступ к отдельной ячейке таблицы. В качестве аргумента может выступать текстовое представление адреса ячейки, либо экземпляр таблицы DocumentApi.CellPosition.

Примеры:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
print(cell:getFormattedValue())
```

```
local cellPosition = DocumentAPI.CellPosition(2, 1)
local cell = tbl:getCell(cellPosition)
print(cell:getFormattedValue())
```

7.112.8 Метод Table:getCellRange

Метод позволяет получить доступ к диапазону ячеек таблицы DocumentAPI.CellRange. В качестве аргумента может использоваться строка, описывающая диапазон ("A1:C4"), либо объект типа DocumentAPI.CellRangePosition.

Примеры:

```
local table = document:getBlocks():getTable(0)
local range = table:getCellRange("A1:C4")
for cell in range:enumerate() do
   print(cell:getFormattedValue())
end
```

```
local table = document:getBlocks():getTable(0)
local range = table:getCellRange(DocumentAPI.CellRangePosition(0, 0, 2, 2))
for cell in range:enumerate() do
   print(cell:getFormattedValue())
end
```

7.112.9 Метод Table:insertColumnAfter

Метод предназначен для вставки нового столбца после указанной позиции в таблице.

Вызов:

```
insertColumnAfter( columnIndex, copyColumnStyle, columnsCount )
```

Параметры:

- `columnIndex` – индекс столбца в таблице, после которого производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример:

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")
-- Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
tbl:insertColumnAfter(0, false, 2)
```

7.112.10 Метод Table:insertColumnBefore

Метод предназначен для вставки нового столбца до указанной позиции в таблице.

Вызов:

```
insertColumnBefore( columnIndex, copyColumnStyle, columnsCount )
```

Параметры:

- `columnIndex` – индекс столбца в таблице, перед которым производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример:

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух столбцов в середину таблицы, без наследования настроек форматирования
tbl:insertColumnBefore(1, false, 2)
```

7.112.11 Метод Table:insertRowAfter

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

Вызов:

```
insertRowAfter( rowIndex, copyRowStyle, rowsCount )
```

Параметры:

- `rowIndex` – индекс строки в таблице, после которой производится вставка. Индексация строк начинается с нуля.
- `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `rowsCount` – количество вставляемых строк. Значение по умолчанию 1.

Пример:

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl:insertRowAfter(0, false, 2)
```

7.112.12 Метод Table:insertRowBefore

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

Вызов:

```
insertRowBefore( rowIndex, copyRowStyle, rowsCount )
```

Параметры:

- `rowIndex` – индекс строки в таблице, перед которой производится вставка. Индексация строк начинается с нуля.

- `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `rowsCount` – количество вставляемых строк. Значение по умолчанию 1.

Пример:

```
-- Создать в документе новую таблицу 2x2
tbl = document:getRange():getBegin():insertTable(2, 2, "SomeTable")

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl:insertRowBefore(1, false, 2)
```

7.112.13 Метод Table:isColumnVisible

Метод `Table::isColumnVisible` позволяет определять видимость столбца по заданному индексу. Индексация столбцов начинается с нуля. Метод возвращает `true` если столбец отображается.

Для задания видимости столбцов таблицы применяется метод `Table::setColumnsVisible`.

Вызов:

```
isColumnVisible(columnIndex)
```

Параметр:

`columnIndex` – индекс столбца.

Пример:

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

Дополнительный пример использования метода `Table::isColumnVisible` приведен в разделе Управление видимостью строк / колонок.

7.112.14 Метод Table:isRowVisible

Метод `Table::isRowVisible` позволяет определять видимость строки по заданному индексу. Индексация строк начинается с нуля. Метод возвращает `true` если строка отображается.

Для задания видимости строк таблицы применяется метод `Table::setRowsVisible`.

Вызов:

```
isRowVisible(rowIndex)
```

Параметр:

`rowIndex` – индекс строки.

Пример:

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

Дополнительный пример использования метода `Table::isRowVisible` приведен в разделе Управление видимостью строк / колонок.

7.112.15 Метод Table:removeColumn

Метод предназначен для удаления столбца таблицы, начиная с заданного индекса.

Вызов:

```
removeColumn(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет удалено заданное количество столбцов. Индексация столбцов начинается с нуля.
- `columnsCount` – количество столбцов для удаления. Значение по умолчанию 1.

7.112.16 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.

7.112.17 Метод Table:groupRows

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

Вызов:

```
groupRows(rowIndex, rowsCount)
```

Параметры:

- `rowIndex` – индекс строки, начиная с которого будет начата группировка строк;
- `rowsCount` – количество строк для группировки.

7.112.18 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.

7.112.19 Метод Table:clearRowGroups

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

Вызов:

```
clearRowGroups(rowIndex, rowCount)
```

Параметры:

- `rowIndex` – индекс строки, начиная с которой будет начата очистка групп;
- `rowCount` – количество строк для очистки групп.

7.112.20 Метод Table:groupColumns

Метод предназначен для группировки столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
groupColumns(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет начата группировка столбцов;
- `columnsCount` – количество столбцов для группировки.

7.112.21 Метод Table:ungroupColumns

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

Вызов:

```
ungroupColumns(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет начата разгруппировка столбцов;
- `columnsCount` – количество столбцов для разгруппировки.

7.112.22 Метод Table:clearColumnGroups

Метод предназначен для очистки группированных столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
clearColumnGroups(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет начата очистка групп;
- `columnsCount` – количество столбцов для очистки групп.

7.112.23 Метод Table:setColumnsVisible

Метод `Table::setColumnsVisible` позволяет задавать видимость столбцов, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Метод предназначен для использования только в табличном редакторе.

Вызов:

```
setColumnsVisible(first, columnsCount, visible)
```

Параметры:

`first` – начальный индекс;

`columnsCount` – количество столбцов;

`visible` – видимость.

Пример использования в табличном редакторе:

```
local tbl = document:getBlocks():getTable(0)
tbl:setColumnsVisible(0, 2, false)
```

Дополнительный пример использования метода `Table::setColumnsVisible` приведен в разделе Управление видимостью строк / колонок.

7.112.24 Метод Table:setRowsVisible

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

Метод предназначен для использования только в табличном редакторе.

Вызов:

```
setRowsVisible(first, rowsCount, visible)
```

Параметры:

`first` – начальный индекс;

`columnsCount` – количество строк;

`visible` – видимость.

Пример использования в табличном редакторе:

```
local tbl = document:getBlocks():getTable(0)
tbl:setRowsVisible(0, 2, false)
```

Дополнительный пример использования метода `Table::setRowsVisible` приведен в разделе Управление видимостью строк / колонок.

7.112.25 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)
```

7.112.26 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)
```

7.112.27 Метод Table:duplicate

Для создания копии листа в табличном документе используется метод `duplicate`. Созданная копия листа размещается после копируемого листа. Метод может быть использован только в табличном документе.

Пример:

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

7.112.28 Метод Table:remove

Для удаления таблицы в текстовом документе или листа в табличном документе используется метод `remove()`.

Пример:

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

7.112.29 Метод Table:moveTo

Для перемещения листа таблицы по указанному индексу в табличном документе используется метод `moveTo`. Указанный индекс должен быть меньше или равен количеству листов в документе. Индексация листов начинается с нуля. Метод может быть использован только в табличном документе.

Пример:

```
-- В табличном документе два листа с индексами 0 и 1.
-- Поменяем их местами.
local tbl = document:getBlocks():getTable(0)
tbl:moveTo(1)
```

7.112.30 Метод Table:setShowZeroValue

Для упрощения чтения таблицы нулевые значения ячеек могут быть скрыты. Для управления скрытием/показом ячеек используется метод `setShowZeroValue`. Метод может быть использован только в табличном документе.

Пример:

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(true)
```

7.112.31 Метод Table:getShowZeroValue

Для проверки режима отображения нулевых значений ячеек используется метод `getShowZeroValue`.

Пример:

```
tbl = document:getBlocks():getTable(0)
tbl:setShowZeroValue(false)
print(tbl:getShowZeroValue())
```

7.112.32 Метод Table:setVisible

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

Вызов:

```
setVisible( visible )
```

Параметр:

`visible` – параметр, задающий видимость листа. Если значение параметра `visible` равно `true`, то лист таблицы отображается в редакторе таблиц.

Пример:

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

7.112.33 Метод Table:isVisible

Метод возвращает значение true, если лист таблицы в табличном документе отображается в редакторе таблиц.

Пример:

```
local tbl = document:getBlocks():getTable(0)
if not tbl:isVisible() then
  tbl:setVisible(true)
end
```

7.112.34 Метод Table:getFrozenRange

Существует возможность закрепления диапазона строк и столбцов. Такие диапазоны всегда остаются видимыми на экране в случае, когда пользователь осуществляет навигацию по таблице.

Метод `getFrozenRange` возвращает закрепленный диапазон `DocumentAPI.FrozenRangePosition`.

Пример:

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local tbl = document:getBlocks():getTable(0)
tbl:freeze(frozenRangePosition)
print(tbl:getFrozenRange():isCols())
```

7.112.35 Метод Table:freeze

Метод `freeze` закрепляет заданную область `DocumentAPI.FrozenRangePosition` таблицы. Может быть использован только в табличном документе.

Пример:

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenCols(0, 2)
local tbl = document:getBlocks():getTable(0)
tbl:freeze(frozenRangePosition)
print(tbl:getFrozenRange():isCols())
```

7.112.36 Метод Table:__eq

Метод используется для определения эквивалентности двух таблиц.

Пример:

```
local tbl1 = document:getBlocks():getTable(0)
local tbl2 = document:getBlocks():getTable("Table Name")
if tbl1:__eq(tbl2) then
    print("tbl1 и tbl2 ссылаются на общую таблицу в документе")
end
```

7.112.37 Метод Table:setPrintArea

Метод служит для установки и сброса области печати DocumentAPI.CellRangePosition.

Пример:

```
local tbl = document:getBlocks():getTable(0)
tbl:setPrintArea(DocumentAPI.CellRangePosition(0, 0, 5, 5)) -- уставить область печати размером в пять строк и пять колонок, начиная с левого верхнего угла таблицы
```

7.112.38 Метод Table:setPrintAreas

Метод `Table:setPrintAreas` задает множественные области печати или экспорта `CellRangePositions`, где `CellRangePositions` - вектор из элементов `CellRangePosition` (см. описание вектора).

Пример:

```
tbl = document:getBlocks():getTable(0)
ranges = DocumentAPI.CellRangePositions()
ranges:push_back(DocumentAPI.CellRangePosition(0, 0, 5, 5))
ranges:push_back(DocumentAPI.CellRangePosition(1, 2, 5, 5))
tbl:setPrintAreas(ranges)

printAreas = tbl:getPrintAreas()
print(printAreas[0]:toString(), printAreas[1]:toString())
```

7.112.39 Метод Table:getPrintAreas

Метод `Table:getPrintAreas` возвращает текущие области печати - вектор элементов `DocumentAPI.CellRangePosition`. См. `описание` методов вектора.

Пример:

```
tbl = document:getBlocks():getTable(0)
tbl:setPrintArea(DocumentAPI.CellRangePosition(0, 0, 5, 5))

printAreas = tbl:getPrintAreas()
print(printAreas[0]:toString())
```

7.112.40 Метод Table:getCharts

Для получения списка диаграмм (`DocumentAPI.Charts`) таблицы используется метод `Table:getCharts`.

Пример:

```
for tbl in document:getBlocks():enumerateTables() do
  print(tbl:getCharts():getChartsCount())
end
```

7.112.41 Метод Table:getImages

Для получения списка изображений (`DocumentAPI.Images`) таблицы используется метод `Table:getImages`.

Пример:

```
tbl = document:getBlocks():getTable(0)
images = tbl:getImages()

for image in images:enumerate() do
  print(image)
end
```

7.112.42 Метод Table:getMediaObjects

Для получения списка медиаобъектов (`DocumentAPI.MediaObjects`) таблицы используется метод `Table:getMediaObjects`.

Пример:

```
tbl = document:getBlocks():getTable(0)
mediaObjects = tbl:getMediaObjects()
```

```
for mediaObject in mediaObjects:enumerate() do
  print(mediaObject)
end
```

7.112.43 Метод Table:getNamedExpressions

Метод используется для получения списка именованных диапазонов DocumentAPI:NamedExpressions.

7.113 Таблица DocumentAPI.TableFilters

TableFilters - это таблица, которая хранит фильтры столбцов. Фильтры можно применять к диапазону фильтрации FiltersRange. При применении фильтров, соответствующие строки рабочего листа будут скрыты.

Конструктор по умолчанию:

```
local firstTableFilters = DocumentAPI.TableFilters()
```

Конструктор копирования:

```
local secondTableFilters = DocumentAPI.TableFilters(firstTableFilters)
```

Пример использования приведен в разделе Работа с фильтрами.

7.113.1 Метод TableFilters:clear

Метод TableFilters:clear удаляет все фильтры столбцов, которые были сохранены ранее.

Пример:

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

7.113.2 Метод TableFilters:erase

Метод TableFilters:erase удаляет фильтр из заданного столбца. В качестве параметра используется индекс столбца.

Пример:

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

7.113.3 Метод TableFilters:setFilter

Метод `TableFilters:setFilter` устанавливает фильтр для конкретного столбца. В качестве параметров используются индекс столбца, а также используемый фильтр: `ValuesTableFilter` или `ConditionalTableFilter`.

Пример:

```
local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")
johnPaulFilter:clear()

local songFilter = DocumentAPI.ConditionalTableFilter()
songFilter:setAndOperation(true)
songFilter:notEqual("")
songFilter:notBegins("TODO")

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

Более подробный пример приведен в разделе Работа с фильтрами.

7.114 Таблица DocumentAPI.TableRangeInfo

Таблица `DocumentAPI.TableRangeInfo` описывает диапазон ячеек таблицы.

Описание полей таблицы `DocumentAPI.TableRangeInfo` представлено в таблице 64.

Таблица 64 – Поля таблицы `DocumentAPI.TableRangeInfo`

Поле	Тип	Описание
`tableRange`	`DocumentAPI.CellRangePosition`	Диапазон ячеек

Пример:

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

7.115 Таблица DocumentAPI.TextAnchoredPosition

Таблица `DocumentAPI.TextAnchoredPosition` (см. Рисунок 45) представляет позицию объекта на странице текстового документа. Пример использования см. в разделе `InlineFrame:setPosition()`.

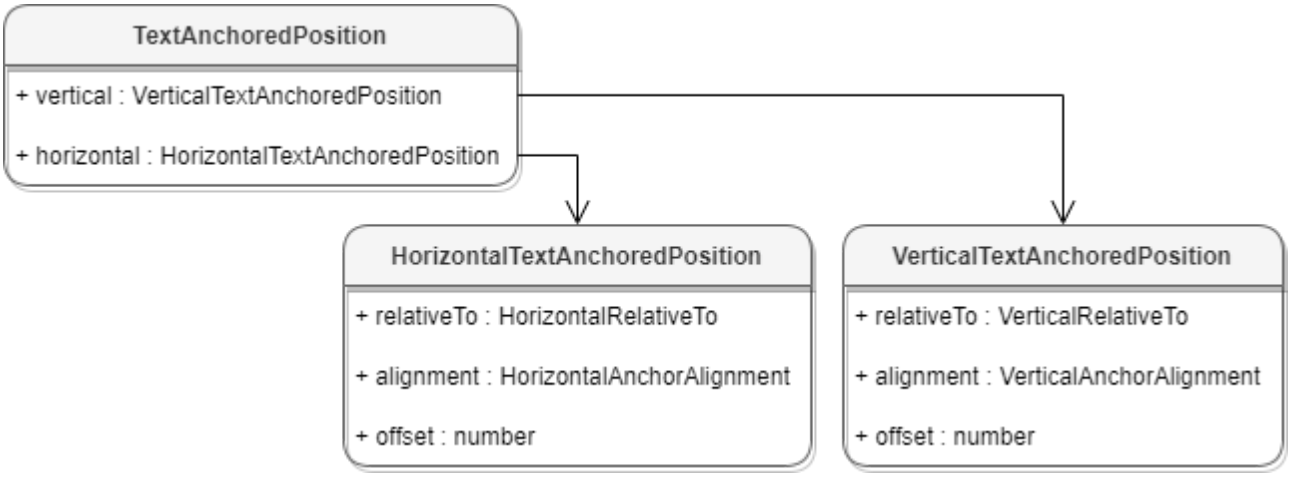


Рисунок 45 – Поля таблицы `DocumentAPI.TextAnchoredPosition`

Описание полей таблицы представлено в таблице 65.

Таблица 65 – Описание полей таблицы `DocumentAPI.TextAnchoredPosition`

Поле	Описание
`DocumentAPI.TextAnchoredPosition.horizontal`	Позиция по горизонтали `HorizontalTexAnchoredPosition`
`DocumentAPI.TextAnchoredPosition.vertical`	Позиция по вертикали `VerticalTextAnchoredPosition`

7.115.1 Метод TextAnchoredPosition:__eq

Метод используется для определения эквивалентности значений двух позиций объектов.

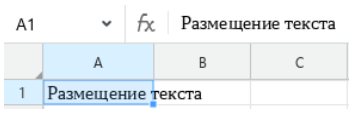
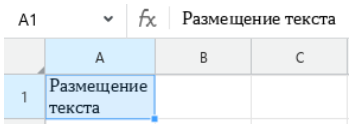
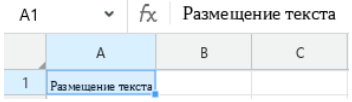
Пример:

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos1.horizontal.offset = 1

local pos2 = DocumentAPI.TextAnchoredPosition()
pos2.horizontal =
DocumentAPI.HorizontalTextAnchoredPosition
(DocumentAPI.HorizontalRelativeTo_Column)
pos2.horizontal.offset = 1

print(pos1:__eq(pos2))
```

7.116 Таблица DocumentAPI.TextLayout

В таблице 66 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле `textLayout` таблицы `CellProperties`.

Таблица 66 – Варианты размещения текста в ячейках таблицы

Наименование константы	Описание	Отображение
`DocumentAPI.TextLayout_SingleLine`	Текст располагается в одну строку с наложением на соседние ячейки.	
`DocumentAPI.TextLayout_WrapByWords`	Текст внутри ячейки переносится по словам. Высота ряда увеличивается чтобы разместить текст полностью.	
`DocumentAPI.TextLayout_ShrinkSizeToFitWidth`	Текст располагается в одну линию, отображение масштабируется таким образом, чтобы полностью разместиться в ячейке без изменения ее размера. Размер шрифта не изменяется, данная настройка влияет только на отображение содержимого ячейки таблицы.	

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell_A1 = tbl:getCell("A1")
local props = cell_A1:getCellProperties()
props.textLayout = DocumentAPI.TextLayout_ShrinkSizeToFitWidth
cell_A1:setCellProperties(props)
```

7.117 Таблица DocumentAPI.TextOrientation

Таблица `DocumentAPI.TextOrientation` предоставляет доступ к свойствам ориентации текста в ячейке, фигуре и т. д (см. `DocumentAPI.CellProperties`).

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("D2") --(DocumentAPI.CellPosition(3,1))
local props = cell:getCellProperties()
props.textOrientation = DocumentAPI.TextOrientation(45)
cell:setCellProperties(props)
print(props.textOrientation:getAngle())
```

7.117.1 Метод TextOrientation:getAngle

Возвращает угол ориентации текста в ячейке. Значение угла указывается в градусах.

Пример:

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

7.117.2 Метод TextOrientation:isStackedChars

Возвращает `True` в случае, если ориентация текста представляет собой вертикальный столбец.

Пример:

```
local cellProperties = cell:getCellProperties()
print(cellProperties.textOrientation:isStackedChars())
```

7.117.3 Метод TextOrientation:__eq

Метод используется для определения эквивалентности значений двух объектов

TextOrientation.

Пример:

```
print(DocumentAPI.TextOrientation(45):__eq(DocumentAPI.TextOrientation(45)))
```

7.118 Таблица DocumentAPI.TextProperties

Таблица DocumentAPI.TextProperties содержит поля, задающие параметры текста. На рисунке 46 изображена объектная модель таблицы DocumentAPI.TextProperties.

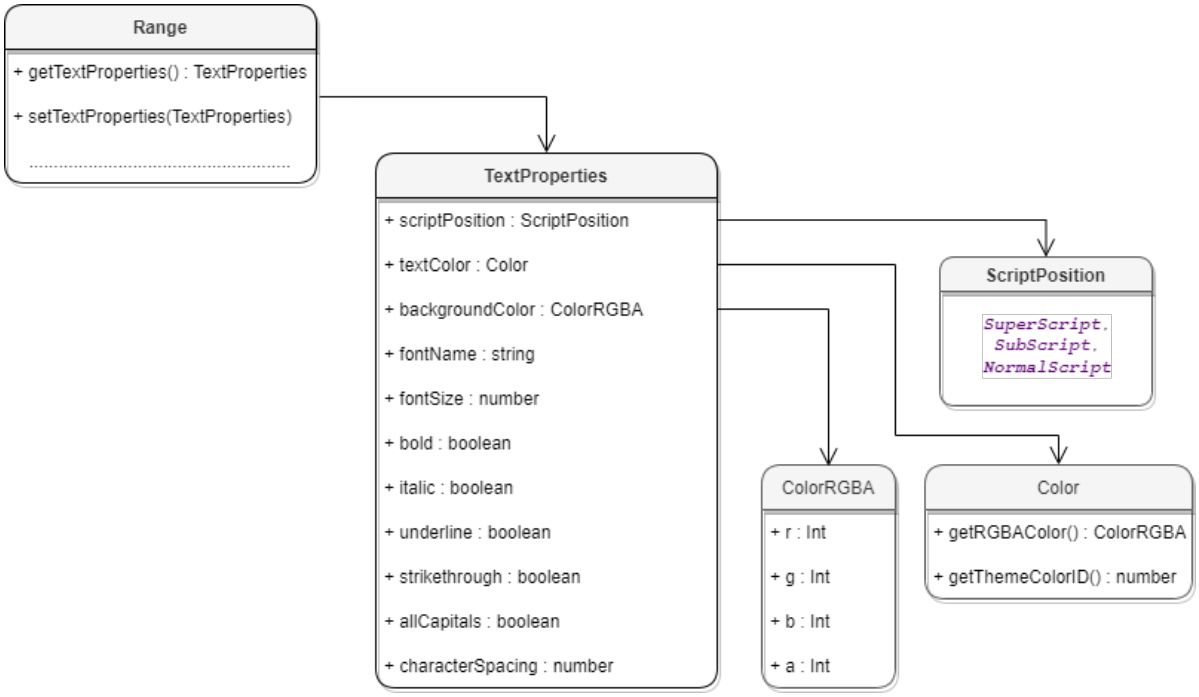


Рисунок 46 – Объектная модель для работы с таблицей DocumentAPI.TextProperties

Описание полей таблицы DocumentAPI.TextProperties представлено в таблице 67. Свойства DocumentAPI.TextProperties применяются к диапазону текста DocumentAPI.Range (методы Range.getTextProperties(), Range.setTextProperties()).

Таблица 67 – Описание полей таблицы DocumentAPI.TextProperties

Поле	Тип	Описание
TextProperties.fontName	Строковое	Наименование шрифта, использованного для оформления фрагмента документа.

Поле	Тип	Описание
`TextProperties.fontSize`	Числовое	Размер шрифта, использованного для оформления фрагмента документа.
`TextProperties.bold`	Логическое	Значение `true` устанавливает жирное начертание для указанного фрагмента текста.
`TextProperties.italic`	Логическое	Значение `true` устанавливает начертание курсивом для указанного фрагмента текста.
`TextProperties.underline`	Логическое	Значение `true` устанавливает подчеркивание для указанного фрагмента текста.
`TextProperties.strikethrough`	Логическое	Значение `true` устанавливает начертание «зачеркнутый» для указанного фрагмента текста.
`TextProperties.allCapitals`	Логическое	Значение `true` устанавливает все буквы указанного фрагмента текста как прописные. Значение `false` устанавливает все буквы указанного фрагмента текста как строчные.
`TextProperties.scriptPosition`	`ScriptPosition`	Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме.
`TextProperties.textColor`	`Color`	Цвет указанного фрагмента документа.
`TextProperties.backgroundColor`	`ColorRGBA`	Цвет фона указанного фрагмента документа.
`TextProperties.characterSpacing`	Числовое	Размер межсимвольного интервала.

Пример:

```
local props = DocumentAPI.TextProperties()
props.fontName = "XO Oriel"
props.fontSize = 20

-- текст третьего абзаца
local range = document:getBlocks():getParagraph(2):getRange()
```

```
-- установить свойства фрагмента текста
range:setTextProperties(props)
```

7.119 Таблица DocumentAPI.TextWrapType

В таблице 68 представлены варианты обтекания текстом встроенного объекта. Используется в `InlineFrame.setWrapType()`.

Таблица 68 – Варианты обтекания текстом встроенного объекта

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

7.120 Таблица DocumentAPI.ThemeColorID

В таблице 69 представлены типы идентификаторов цветов тем. Используется в `DocumentAPI.Color`.

Таблица 69 – Типы идентификаторов цветов тем

Наименование константы	Описание
`DocumentAPI.ThemeColorID_Background1`	Фон1
`DocumentAPI.ThemeColorID_Text1`	Текст1
`DocumentAPI.ThemeColorID_Background2`	Фон2
`DocumentAPI.ThemeColorID_Text2`	Текст2
`DocumentAPI.ThemeColorID_Dark1`	Темная1
`DocumentAPI.ThemeColorID_Dark2`	Темная2
`DocumentAPI.ThemeColorID_Light1`	Светлая1
`DocumentAPI.ThemeColorID_Light2`	Светлая2
`DocumentAPI.ThemeColorID_Accent1`	Акцент1

Наименование константы	Описание
DocumentAPI.ThemeColorID_Accent2	Акцент2
DocumentAPI.ThemeColorID_Accent3	Акцент3
DocumentAPI.ThemeColorID_Accent4	Акцент4
DocumentAPI.ThemeColorID_Accent5	Акцент5
DocumentAPI.ThemeColorID_Accent6	Акцент6
DocumentAPI.ThemeColorID_Hyperlink	Гиперссылка
DocumentAPI.ThemeColorID_FollowedHyperlink	Следующая гиперссылка

7.121 Таблица DocumentAPI.TimePatterns

Форматы времени представлены в таблице 70. Пример использования см. в главе DocumentAPI.DateTimeCellFormatting.

Таблица 70 – Форматы времени

Наименование константы	Описание
DocumentAPI.TimePatterns_ShortTime	'hh:mm AM/PM' для языка en_US
DocumentAPI.TimePatterns_LongTime	'hh:mm:ss AM/PM' для языка en_US

7.122 Таблица DocumentAPI.TrackedChange

Таблица DocumentAPI.TrackedChange представляет отслеживаемое изменение в диапазоне текстового документа (см. Рисунок 47).

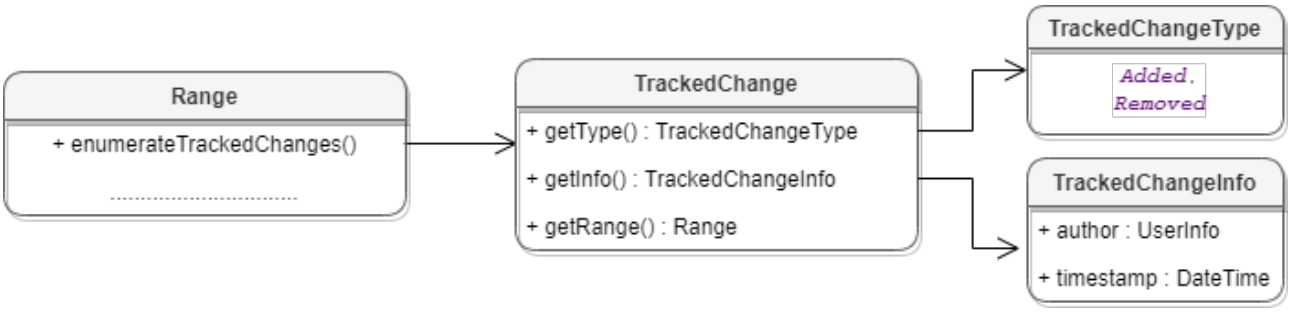


Рисунок 47 – Объектная модель таблиц для работы с отслеживаемыми изменениями

Для получения списка отслеживаемых изменений используется метод Range.enumerateTrackedChanges().

Пример:

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

7.122.1 Метод TrackedChange:getRange

Метод возвращает объект DocumentAPI.Range, который соответствует измененному диапазону внутри абзаца.

Пример:

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
  print(tracked_change:getRange():extractText())
end
```

7.122.2 Метод TrackedChange:getType

Метод позволяет получить информацию о типе отслеживаемого изменения DocumentAPI.TrackedChangeType.

Пример:

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
  print(tracked_change:getType())
end
```

7.122.3 Метод TrackedChange:getInfo

Метод позволяет получить информацию об отслеживаемых изменениях (DocumentAPI.TrackedChangeInfo).

Пример:

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
for tracked_change in tracked_changes do
  print(tracked_change:getInfo().author.name)
end
```

7.123 Таблица DocumentAPI.TrackedChangeInfo

Таблица `DocumentAPI.TrackedChangeInfo` содержит информацию об отслеживаемых изменениях. Описание полей таблицы представлено в таблице 71.

Таблица 71 – 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 ts_msg = string.format("%d/%d/%d - %d:%d:%d", ts.day, ts.month, ts.year,
ts.hour, ts.minute, ts.second)
  print(author.name, ts_msg)
end
```

7.123.1 Метод TrackedChangeInfo:__eq

Метод используется для определения эквивалентности двух отслеживаемых изменений.

7.124 Таблица DocumentAPI.TrackedChangeType

Типы отслеживаемых изменений представлены в таблице 72.

Таблица 72 – Типы отслеживаемых изменений

Наименование константы	Описание
`DocumentAPI.TrackedChangeType_Added`	Добавленные изменения
`DocumentAPI.TrackedChangeType_Removed`	Удаленные изменения

Пример:

```
local changesList = document:getRange():enumerateTrackedChanges()
for change in changesList do
  if DocumentAPI.TrackedChangeType_Added == change:getType() then action =
```

```
"Добавлено: " else action = "Удалено: " end
  print(action)
end
```

7.125 Таблица DocumentAPI.ValueFieldsOrientation

Таблица `DocumentAPI.ValueFueldsOrientation` 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 73.

Таблица 73 – Описание полей таблицы `DocumentAPI.ValueFueldsOrientation`

Поле	Описание
`DocumentAPI.ValueFueldsOrientation_ByRows`	По строкам
`DocumentAPI.ValueFueldsOrientation_ByColumns`	По столбцам

7.126 Таблица DocumentAPI.ValuesTableFilter

Таблица `ValuesTableFilter` реализует фильтр, содержащий значения, которые должны быть показаны в диапазоне фильтрации.

Конструктор по умолчанию:

```
local firstFilter = DocumentAPI.ValuesTableFilter()
```

Конструктор копирования:

```
local secondFilter = DocumentAPI.ValuesTableFilter(firstFilter)
```

Пример использования приведен в разделе Работа с фильтрами.

7.126.1 Метод ValuesTableFilter:add

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

Пример:

```
local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")
```

7.126.2 Метод ValuesTableFilter:clear

Метод `ValuesTableFilter::clear` удаляет все элементы фильтра.

Пример:

```
local johnPaulFilter = DocumentAPI.ValuesTableFilter()
johnPaulFilter:add("John")
johnPaulFilter:add("Paul")
.........................
johnPaulFilter:clear()
```

7.127 Таблица DocumentAPI.VectorString

Таблица `DocumentAPI.VectorString` предназначена для реализации массива строк.

Пример:

```
vector = DocumentAPI.VectorString(3)
vector[0] = "1"
vector[1] = "2"
vector[2] = "3"
print(vector:size()) -- 3
```

7.127.1 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
```

7.127.2 Метод VectorString:max_size

Метод возвращает максимальный размер вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
print(vector:max_size())
```

7.127.3 Метод VectorString:empty

Метод возвращает `true`, если вектор не содержит элементов.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
print(vector:empty()) -- false
```

7.127.4 Метод VectorString:clear

Метод очищает содержимое вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:clear()
print(vector:empty()) -- true
```

7.127.5 Метод VectorString:push_back

Метод добавляет элемент в конец вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
print(vector:size()) -- 1
```

7.127.6 Метод VectorString:pop_back

Метод удаляет последний элемент вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:pop_back()
print(vector:size()) -- 0
```

7.127.7 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
```

7.127.8 Метод VectorString:back

Метод возвращает последний элемент вектора.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:push_back("13")
print(vector:front()) -- 13
```

7.127.9 Метод VectorString:__getitem

Метод возвращает элемент вектора по заданному индексу.

Пример:

```
local vector = DocumentAPI.VectorString()
vector:push_back("12")
vector:push_back("13")
print(vector:__getitem(0)) -- 12
print(vector:__getitem(1)) -- 13
```

7.127.10 Метод VectorString:__setitem

Метод устанавливает элемент вектора по заданному индексу.

Пример:

```
local vector = DocumentAPI.VectorString(2)
vector:__setitem(0, "12")
vector:__setitem(1, "13")
print(vector:__getitem(0)) -- 12
print(vector:__getitem(1)) -- 13
```

7.128 Таблица DocumentAPI.VectorUInt

Таблица `DocumentAPI.VectorUInt` предназначена для реализации массива данных.

Пример:

```
vector = DocumentAPI.VectorUInt(3)
vector[0] = 1
vector[1] = 13
```

```
vector[2] = 25
print(vector:size()) -- 3
```

7.128.1 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
```

7.128.2 Метод VectorUInt:max_size

Метод возвращает максимальный размер вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()
print(vector:max_size())
```

7.128.3 Метод VectorUInt:empty

Метод возвращает `true`, если вектор не содержит элементов.

Пример:

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
print(vector:empty()) -- false
```

7.128.4 Метод VectorUInt:clear

Метод очищает содержимое вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
vector:clear()
print(vector:empty()) -- true
```

7.128.5 Метод VectorUInt:push_back

Метод добавляет элемент в конец вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
print(vector:size()) -- 1
```

7.128.6 Метод VectorUInt:pop_back

Метод удаляет последний элемент вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
vector:pop_back()
print(vector:size()) -- 0
```

7.128.7 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
```

7.128.8 Метод VectorUInt:back

Метод возвращает последний элемент вектора.

Пример:

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
vector:push_back(13)
print(vector:front()) -- 13
```

7.128.9 Метод VectorUInt:__getitem

Метод возвращает элемент вектора по заданному индексу.

Пример:

```
local vector = DocumentAPI.VectorUInt()
vector:push_back(12)
vector:push_back(13)
print(vector:__getitem(0)) -- 12
print(vector:__getitem(1)) -- 13
```

7.128.10 Метод VectorUInt:__setitem

Метод устанавливает элемент вектора по заданному индексу.

Пример:

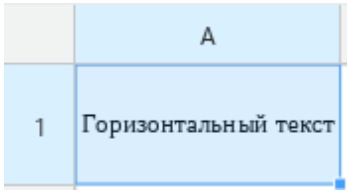
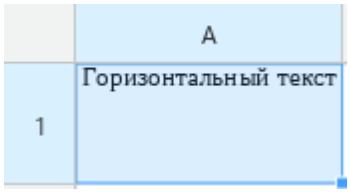
```
local vector = DocumentAPI.VectorUInt(2)
vector:__setitem(0, 12)
vector:__setitem(1, 13)
print(vector:__getitem(0)) -- 12
print(vector:__getitem(1)) -- 13
```

7.129 Таблица DocumentAPI.VerticalAlignment

В таблице 74 представлены константы видов выравнивания текста по вертикали. Используется в DocumentAPI.CellProperties, DocumentAPI.ShapeProperties.

Таблица 74 – Виды выравнивания текста по вертикали

Наименование константы	Представление в интерфейсе	
DocumentAPI.VerticalAlignment_Bottom		

Наименование константы	Представление в интерфейсе	
DocumentAPI.VerticalAlignment_Center		
DocumentAPI.VerticalAlignment_Top		

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A1")
local props = cell:getCellProperties()
props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
cell:setCellProperties(props)
```

7.130 Таблица DocumentAPI.VerticalAnchorAlignment

В таблице 75 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали. Используется в DocumentAPI.VerticalTextAnchoredPosition.

Таблица 75 – 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

7.131 Таблица DocumentAPI.VerticalRelativeTo

В таблице 76 представлены типы размещения объекта относительно закрепленной позиции по вертикали. Используется в `DocumentAPI.VerticalTextAnchoredPosition`.

Таблица 76 – Типы размещения объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
`DocumentAPI.VerticalRelativeTo_Character`	Символ
`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

7.132 Таблица DocumentAPI.VerticalTextAnchoredPosition

Таблица `DocumentAPI.VerticalTextAnchoredPosition` предназначена для управления относительным положением объекта со смещением или выравниванием по вертикали. Пример использования см. в `InlineFrame:setPosition()`.

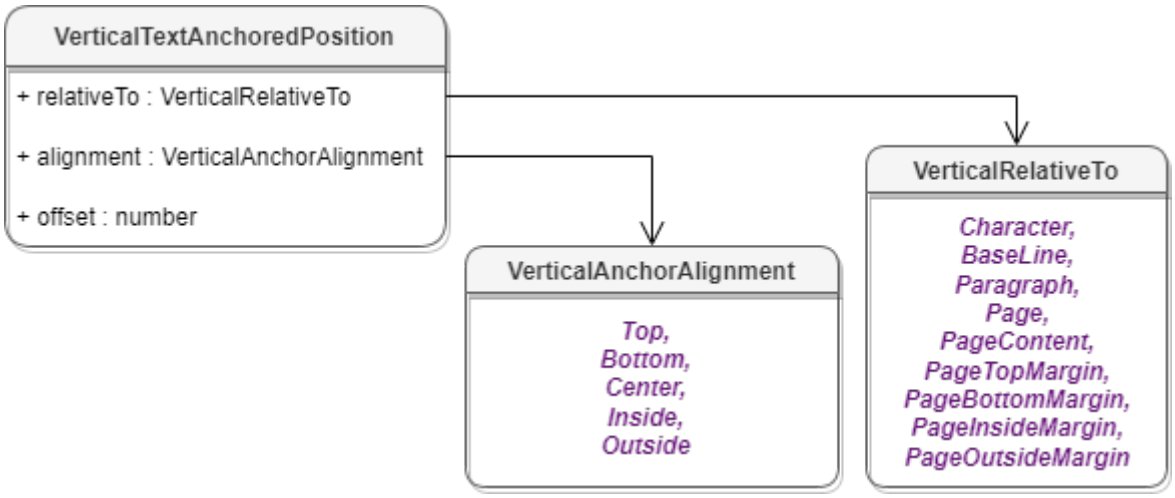


Рисунок 48 – Поля таблицы `DocumentAPI.VerticalTextAnchoredPosition`

Описание полей таблицы `DocumentAPI.VerticalTextAnchoredPosition` представлено в таблице 77.

Таблица 77 – Описание полей таблицы DocumentAPI.VerticalTextAnchoredPosition

Поле	Описание
DocumentAPI.VerticalTextAnchoredPosition.alignment	Тип выравнивания объекта относительно закрепленной позиции по вертикали VerticalAnchorAlignment.
DocumentAPI.VerticalTextAnchoredPosition.relativeTo	Тип размещения объекта относительно закрепленной позиции по вертикали VerticalRelativeTo.
DocumentAPI.VerticalTextAnchoredPosition.offset	Смещение объекта

7.132.1 Метод VerticalTextAnchoredPosition:__eq

Метод используется для определения эквивалентности двух положений объекта по вертикали.

Пример:

```
local pos1 = DocumentAPI.TextAnchoredPosition()
pos1.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
pos1.vertical.offset = 1

local pos2 = DocumentAPI.TextAnchoredPosition()
pos2.vertical =
DocumentAPI.VerticalTextAnchoredPosition(DocumentAPI.VerticalRelativeTo_Page)
pos2.vertical.offset = 1

print(pos1.vertical:__eq(pos2.vertical))
```

7.133 Таблица DocumentAPI.WorksheetPrinterFitType

В таблице 78 представлены варианты масштабирования при печати табличных документов. Используется в качестве поля worksheetPrinterFitType таблицы DocumentAPI.PrintSettings.

Таблица 78 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
DocumentAPI.WorksheetPrinterFitType_ActualSize	Фактический размер

Наименование константы	Описание
`DocumentAPI.WorksheetPrinterFitType_ByPageScale`	По масштабу страницы
`DocumentAPI.WorksheetPrinterFitType_ByPageBreaksOnly`	По разрыву страниц
`DocumentAPI.WorksheetPrinterFitType_FitToPage`	Вписать в страницу
`DocumentAPI.WorksheetPrinterFitType_FitToWidth`	Вписать по ширине
`DocumentAPI.WorksheetPrinterFitType_FitToHeight`	Вписать по высоте

8 Справочник функций DocumentAPI

8.1 Функция DocumentAPI:createSearch

Функция инициализирует механизм поиска для текущего документа. Возвращает ссылку на таблицу `DocumentAPI.Search`, с помощью методов которой выполняются поисковые запросы.

Пример:

```
search = DocumentAPI.createSearch(document)
ranges = search:findText("English")
```

8.2 Функция DocumentAPI.createScripting

Функция `DocumentAPI.createScripting` возвращает таблицу `DocumentAPI.Scripting`. В качестве параметра используется текущий документ.

Пример:

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

9 Справочник таблиц EditorAPI

9.1 Таблица EditorAPI.SelectionMode

Таблица содержит варианты изменения текущего выделения. Используется в качестве аргумента метода EditorAPI.changeSelection(). Описание полей таблицы EditorAPI.SelectionMode представлено в таблице 79.

Таблица 79 – Описание полей таблицы EditorAPI.SelectionMode

Поле	Описание
EditorAPI.SelectionMode.Move	Переместить выделение
EditorAPI.SelectionMode.Resize	Изменить размер текущего выделения

Пример:

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.UpLeft, EditorAPI.TextSelectionUnit.Word)
```

9.2 Таблица EditorAPI.SelectionDirection

Таблица содержит параметры для управления направлением выделения. Используется в качестве аргумента метода EditorAPI.changeSelection(). Описание полей таблицы EditorAPI.SelectionDirection представлено в таблице 80.

Таблица 80 – Описание полей таблицы DocumentAPI.SelectionDirection

Поле	Описание
EditorAPI.SelectionDirection.Up	Изменить выделение вверх
EditorAPI.SelectionDirection.Down	Изменить выделение вниз
EditorAPI.SelectionDirection.Right	Изменить выделение направо
EditorAPI.SelectionDirection.Left	Изменить выделение налево
EditorAPI.SelectionDirection.DownRight	Изменить выделение вниз и направо
EditorAPI.SelectionDirection.UpLeft	Изменить выделение вверх и налево

Пример:

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)
```

9.3 Таблица EditorAPI.TableSelectionUnit

Таблица содержит параметры для управления шагом выделения в таблице документа. Используется в качестве аргумента метода `EditorAPI.changeSelection()`. Описание полей таблицы `EditorAPI.TableSelectioUnit` представлено в таблице 81.

Таблица 81 – Описание полей таблицы `EditorAPI.TableSelectioUnit`

Поле	Описание
`EditorAPI.TableSelectionUnit.ToEdge`	Направление смены выделения - угол таблицы
`EditorAPI.TableSelectionUnit.ToClosestCell`	Направление смены выделения - ближайшая ячейка

Пример:

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.UpLeft, EditorAPI.TableSelectionUnit.ToClosestCell)
```

9.4 Таблица EditorAPI.TextSelectionUnit

Таблица содержит параметры для управления шагом выделения в тексте документа. Используется в качестве аргумента метода `EditorAPI.changeSelection()`. Описание полей таблицы `EditorAPI.TextSelectionUnit` представлено в таблице 82.

Таблица 82 – Описание полей таблицы `EditorAPI.TextSelectionUnit`

Поле	Описание
`EditorAPI.TextSelectionUnit.Character`	Изменять выделение с шагом в один символ
`EditorAPI.TextSelectionUnit.Word`	Изменять выделение с шагом в одно слово
`EditorAPI.TextSelectionUnit.Paragraph`	Изменять выделение с шагом в один параграф
`EditorAPI.TextSelectionUnit.Line`	Изменять выделение с шагом в одну строку

Пример:

```
EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)
```

10 Справочник функций EditorAPI

Глобальная таблица `EditorAPI` содержит функции доступа к внешней функциональности редактора.

10.1 Функция EditorAPI.changeSelection

Функция `EditorAPI.changeSelection` позволяет изменить текущее выделение в текстовом или табличном документе.

Вызов функции:

```
EditorAPI.changeSelection(mode, direction, unit, count = 1)
```

Где:

- `mode` – режим выделения, тип `DocumentAPI.SelectionMode`;
- `direction` – направление выделения, тип `DocumentAPI.SelectionDirection`;
- `unit` – шаг изменения выделения, тип `DocumentAPI.TableSelectionUnit` для таблиц, или `DocumentAPI.TextSelectionUnit` для текста;
- `count` – количество шагов, необязательный параметр, по умолчанию используется значение 1.

Функция возвращает `true` в случае, если выделение было изменено.

Пример для текстового документа:

```
if (EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Left, EditorAPI.TextSelectionUnit.Character)) then
   EditorAPI.messageBox("Selection changed")
end
```

Пример для табличного документа:

```
if (EditorAPI.changeSelection(EditorAPI.SelectionMode.Resize,
EditorAPI.SelectionDirection.Right, EditorAPI.TableSelectionUnit.ToClosestCell,
2)) then
   EditorAPI.messageBox("Selection changed")
end
```

10.2 Функция EditorAPI.getSelection

Функция `EditorAPI.getSelection` предоставляет доступ к выделенному фрагменту документа.

В открытом документе может быть выделен только один фрагмент.

При использовании в редакторе текста функция `EditorAPI.getSelection` возвращает `Range`, а при использовании в редакторе таблиц - `CellRange`.

Пример для текстового редактора:

Использование функции `EditorAPI.getSelection` в редакторе текста для печати выделенного фрагмента текста.

```
range = EditorAPI.getSelection()
text = range:extractText()
print(text)
```

Пример для табличного редактора:

Использование функции `EditorAPI.getSelection` в редакторе таблиц для печати значений ячеек в выделенном фрагменте таблицы.

```
cellRange = EditorAPI.getSelection()
for cell in cellRange:enumerate() do
    print(cell:getFormattedValue())
end
```

10.3 Функция EditorAPI.setSelection

Функция `EditorAPI.setSelection` позволяет выделить фрагмент документа.

В открытом документе может быть выделен только один фрагмент.

Вызов функции для текстового документа:

```
EditorAPI.setSelection(range)
```

Где:

- `range` – выделяемый в текстовом документе фрагмент текста типа `DocumentAPI.Range`.

Пример для текстового документа:

```
EditorAPI.setSelection(document:getBlocks():getParagraph(0):getRange())
```

Вызов функции для табличного документа:

```
EditorAPI.setSelection(cellRange)
```

Где:

– `cellRange` – выделяемый в табличном документе фрагмент таблицы типа `DocumentAPI.CellRange`.

Пример для табличного документа:

```
cellRange = document:getBlocks():getTable(0):getCellRange("A1:E5")
EditorAPI.setSelection(cellRange)
```

10.4 Функция EditorAPI.messageBox

Функция `EditorAPI.messageBox()` выводит на экран сообщение с заданным текстом и отображением кнопки **ОК**, при этом исполнение макрокоманды приостанавливается до нажатия кнопки **ОК**.

Вызов:

```
messageBox(prompt : string)
messageBox(prompt : string)
messageBox(prompt : string, title : string)
```

Параметры:

– `prompt` – текст сообщения;

– `title` – заголовок окна сообщения.

Пример:

```
EditorAPI.messageBox(cell:getFormattedValue())
```

10.5 Функция EditorAPI.showPrintDialog

Функция `EditorAPI.showPrintDialog()` показывает стандартное окно печати редактора и распечатывает документ, если пользователь подтверждает необходимость печати. Значения, возвращаемые функцией `EditorAPI.showPrintDialog()` перечислены в разделе `DocumentAPI.PrintDocumentResult`.

Пример:

```
printDocumentResult = EditorAPI.showPrintDialog()
print(printDocumentResult)
```

10.6 Функция EditorAPI.printDocument

Функция `EditorAPI.printDocument()` предоставляет возможность печати документа с заданными параметрами печати. Описание параметров печати представлено в разделе `DocumentAPI.PrintSettings`.

Значения, возвращаемые функцией `EditorAPI.printDocument()`, перечислены в разделе `DocumentAPI.PrintDocumentResult`.

Пример:

```
local printSettings = {}
printSettings.printSelection = true
EditorAPI.printDocument(printSettings)
```

10.7 Функция EditorAPI.isPrinterAvailable

Функция `EditorAPI.isPrinterAvailable` позволяет проверить доступность последнего использованного принтера. Возвращает `false`, если принтер недоступен.

Пример:

```
if EditorAPI.isPrinterAvailable() then
    EditorAPI.messageBox("Printer is available")
else
    EditorAPI.messageBox("Printer is not available")
end
```

10.8 Функция EditorAPI.getActiveWorksheet

Функция `EditorAPI.getActiveWorksheet()` возвращает активный лист в табличном документе (класс `Table`). Метод не предназначен для текстовых документов.

Пример:

```
activeWorksheet = EditorAPI.getActiveWorksheet()
print(activeWorksheet:getName()) -- Лист1
```

10.9 Функция EditorAPI.setActiveWorksheet

Функция `EditorAPI.setActiveWorksheet()` устанавливает активный лист в табличном документе. В качестве параметра используется имя листа. Возвращает `true`, если лист найден по имени и активирован. Метод не предназначен для текстовых документов.

Пример:

```
print(EditorAPI.setActiveWorksheet("Лист1")) -- true or false
```

11 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.

11.1 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
```

11.2 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
```

11.3 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
```

11.4 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))
```

11.5 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
```

11.6 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
```

11.7 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
```

11.8 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
```

11.9 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
```

11.10 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
```

11.11 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
```

11.12 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
```

11.13 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
```

11.14 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
```

11.15 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)
```

12 Функции для работы с регулярными выражениями

12.1 Функция Re.create

Функция `Re.create` компилирует регулярное выражение и возвращает его в виде объекта. По умолчанию используется Perl - совместимый формат регулярных выражений.

Вызов:

```
Re.create(pattern)
```

Параметры:

- `pattern` (`string`) - строка шаблона.

Возвращает:

- `regex` (`object`) - объект Regex, который содержит скомпилированное регулярное выражение для дальнейшего использования;
- `err` (`string`) - сообщение об ошибке или `nil`.

12.2 Функция Re.match

Сопоставляет скомпилированное регулярное выражение с заданной исходной строкой. Возвращает найденные подстроки.

Вызов:

```
Re.match(subject, matchFlags, pattern)
```

Параметры:

- `subject` (`string`) – исходная строка;
- `matchFlags` (`int`) – флаги, задающие правила применения регулярного выражения;
- `pattern` (`string`, `Regex`) – строка шаблона или скомпилированный шаблон.

Возвращает:

- `matches` (`object`) – подстроки, найденные в соответствии с шаблоном;
- `err` (`string`) - сообщение об ошибке или nil.

12.2.1 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 83

Таблица 83 – Описание флагов `Re.Match`

Флаг	Описание
`Default`	Указывает, что работа с регулярными выражениями происходит в соответствии с обычными правилами: ECMA-262, спецификация языка ECMAScript, глава 15, часть 10, Регулярные Выражения.
`NotBOB`	Указывает, что выражения ”\A” и "\`" не должны совпадать с подмножеством *[first, first)*.
`NotEOB`	Указывает, что выражения "\`", "\ z" и "\Z" не должны совпадать с подмножеством *[last, last)*.
`NotBOL`	Указывает, что выражение "^" не должны совпадать с подмножеством *[first, first)*.
`NotEOL`	Указывает, что выражение "$" не должно совпадать с подмножеством *[last, last)*.
`NotBOW`	Указывает, что выражения "\<" и "\b" не должны совпадать с подмножеством *[first, first)*.
`NotEOW`	Указывает, что выражения "\>" и "\b" не должны совпадать с подмножеством *[last, last)*.
`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 *[from, last)*, при этом *from! = last*, если может существовать более длинная последовательность символов *[from, to)*, из которых *[from, last)* - 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`

Флаг	Описание
	игнорируются алгоритмами регулярных выражений (RE.7) и итераторов (RE.8).
`DotNewLine`	Указывает, что выражение "." не распознается как символ новой строки. Это инверсия модификатора "s/" языка Perl.
`NotDotNull`	Указывает, что выражение "." не распознается как символ null ('\0').
`Posix`	Указывает, что выражение должно быть обработано в соответствии с правилом POSIX *"leftmost-longest"*, независимо от того, какое выражение было скомпилировано. Эти правила плохо работают со многими специфичными для Perl моментами, например, такими как ленивые (*"non-greedy"*) 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.

12.3 Функция Re.search

Ищет скомпилированное регулярное выражение по заданной строке. Метод возвращает найденные подстроки.

Вызов:

```
Re.search(subject, matchFlags, pattern)
```

Параметры:

- `subject` (`string`) – исходная строка;
- `matchFlags` (`int`) – флаги, задающие правила применения регулярного выражения;
- `pattern` (`string`, `Regex`) – строка шаблона или скомпилированный шаблон.

Возвращает:

- `matches` (`object`) – подстроки, найденные в соответствии с шаблоном;
- `err` (`string`) - сообщение об ошибке или `nil`.

12.4 Функция Re.replace

Находит в заданной строке все фрагменты, удовлетворяющие регулярному выражению. Каждый найденный фрагмент форматируется в соответствии с форматтером и заменяет собой исходный текст.

Вызов:

```
Re.replace(subject, formatter, matchFlags, pattern)
```

Параметры:

- `subject` (`string`) – исходная строка для поиска;

– `formatter` (`string`) – строка, задающая форматирование найденных фрагментов;
– `matchFlags` (`int`) – флаги, задающие правила применения регулярного выражения, а также флаги, специфичные для замены;
– `pattern` (`string`, `Regex`) – строка шаблона или скомпилированный шаблон.

Возвращает:

– `newString` (`string`) – новая строка с замененными подстроками;
– `err` (`string`) - сообщение об ошибке или nil.

12.5 Флаги, используемые для замены

Эти флаги определены в пространстве имен `Re.Replace`. Они используются в алгоритме, используемом методом `Re.replace()` и находятся в таблице 84

Таблица 84 – Описание флагов для функции `Re.replace()`

Флаг	Описание
`FormatDefault`	Когда к исходному тексту применяется регулярное выражение, и находится очередной фрагмент для замены, новая строка формируется с использованием функции замены *ECMAScript*, `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.
`FormatSed`	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`.
`FormatPerl`	В случае, когда регулярное выражение применяется для замены в строке, новая строка формируется `по правилам Perl 5`.
`FormatLiteral`	В случае, когда регулярное выражение применяется для замены в строке, новая строка будет являться строковой копией заменяемого текста.
`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.

Флаг	Описание
`FormatAll`	Все синтаксические расширения включены, включая условные замены (*?ddexpression1:expression2*). Для дополнительных деталей см. Руководство по форматированию строк Boost.

13 Функции для работы с датой и временем

В данном разделе описаны функции, предназначенные для работы с датой и временем: os.clock(), os,date(), os.difftime(), os.time().

Примеры задач, которые позволяют решать данные функции:

Преобразование DATETIME в POSIX:

```
datetime = {year = 2013, month = 09, day = 13, hour = 21, min = 40, sec = 15}
seconds_since_epoch = os.time(datetime)
print(tostring(seconds_since_epoch))
```

Преобразование POSIX в DATETIME:

```
seconds_since_epoch  = 1379094015
datetime = os.date("!*t",seconds_since_epoch)
print( "year = "          .. tostring(datetime.year) .. " " ..
       "month = "         .. tostring(datetime.month) .. " " ..
       "day = "           .. tostring(datetime.day) .. " " ..
       "hour = "          .. tostring(datetime.hour) .. " " ..
       "min = "           .. tostring(datetime.min) .. " " ..
       "sec = "           .. tostring(datetime.sec) .. " " ..
       "weekday = "       .. tostring(datetime.wday) .. " " ..
       "day of year = "   .. tostring(datetime.yday) .. " " ..
       "iddst = "         .. tostring(datetime.isdst))
```

Текущее время в формате POSIX:

```
print(os.time())
```

Текущее время в формате DATETIME:

```
datetime = os.date("!*t",os.time())
print(tostring(datetime.year) .. " " ..
      tostring(datetime.month) .. " " ..
      tostring(datetime.day) .. " " ..
      tostring(datetime.hour) .. " " ..
      tostring(datetime.min) .. " " ..
      tostring(datetime.sec) .. " " ..
      tostring(datetime.wday) .. " " ..
      tostring(datetime.yday) .. " " ..
      tostring(datetime.isdst))
```

Текущий месяц:

```
print(os.date("%B", os.time()))
```

День недели по дате:

```
print(os.date("%w",os.time({ year=1997, month = 11, day = 10})))
```

Порядковый номер недели в году по дате:

```
print(os.date("%W",os.time({ year=1997, month = 11, day = 10})))
```

Вывод даты / времени в различных форматах:

```
print(os.date("%d.%m.%Y"))
print(os.date("%X",os.time()))
print(os.date("Сейчас %H часов %M минут %S секунд",os.time()))
print(os.date())
```

13.1 Функция os.clock

Функция `os.clock` возвращает время от начала запуска приложения / процесса. Время возвращается в секундах с точностью до миллисекунд.

Типичное применение - измерение времени выполнения фрагмента кода.

Пример измерения времени выполнения фрагмента кода:

```
local x = os.clock()
local s = 0
for i=1,100000 do s = s + i end
print(string.format("elapsed time: %.2f\n", os.clock() - x))
```

13.2 Функция os.date

Функция `os.date(format, time)` возвращает форматированную дату / время. В качестве первого аргумента выступает формат, вторым аргументом является время в секундах. Оба аргумента не обязательны. При отсутствии второго аргумента будет использован заданный формат и текущая дата / время. Вызов без аргументов вернет текущую дату / время в формате 07.05.2024 13:33:10.

В строке формата могут быть использованы следующие опции:

%a - день недели, сокр. (англ.) (пример, Wed)

%A - день недели, полностью (англ.) (пример, Wednesday)

%b - месяц, сокр. (англ.) (пример, Sep)

%B - месяц, полностью (англ.) (пример, September)

%c - дата и время (по-умолчанию) (пример, 03/22/15 22:28:11)

%d - день месяца (пример, 22) [диапазон, 01-31]

%H - час, в 24-х часовом формате (пример, 23) [диапазон, 00-23]

%I - час, в 12-и часовом формате (пример, 11) [диапазон, 01-12]

%M - минута (пример, 48) [диапазон, 00-59]

%m - месяц (пример, 09) [диапазон, 01-12]

%p - время суток "am", или "pm"

%S - секунда (пример, 10) [диапазон, 00-59]

%w - день недели (пример, 3) [диапазон, 0-6, соответствует Sunday-Saturday]

%x - дата (пример, 09/16/98)

%X - время (пример, 23:48:10)

%Y - год, 4 цифры (пример, 2015)

%y - год, 2 цифры (пример, 15) [00-99]

%% - символ "%"

*t - вернет таблицу

!*t - вернет таблицу (по Гринвичу)

Если параметр `format` начинается с '!', то время форматируется в соответствии с универсальным глобальным временем (по Гринвичу). После этого опционального символа, если format равен "*t", то date возвращает таблицу со следующими полями: `year` (год, четыре цифры), `month` (месяц, 1 – 12), day (день, 1 – 31), `hour` (час, 0 – 23), `min` (минуты, 0 – 59), `sec` (секунды, 0 – 61), `wday` (день недели, воскресенью соответствует 1), `yday` (день года), и `isdst` (флаг дневного времени суток, тип boolean).

Примеры:

```
print (os.date("%x")) --> 07.05.2024
print (os.date("%c")) --> 25/04/07 10:10:05
```

13.3 Функция os.difftime

Функция `os.diftime(t1, t2)` возвращает число секунд, прошедших от времени t1 до времени t2.

Пример сравнения двух дат в днях:

```
reference = os.time{day=15, year=2024, month=2}
daysfrom = os.difftime(os.time(), reference) / (24 * 60 * 60) -- seconds in a
day
wholedays = math.floor(daysfrom)
print(wholedays)
```

13.4 Функция os.time

Функция `os.time()` возвращает время в формате posix (количество секунд, прошедших с 00:00:00 1 января 1970 года).

При вызове без аргументов возвращает текущее время.

Аргументом может являться таблица с обязательными ключами `year`, `month`, `day`, и необязательными `hour`, `min`, `sec`, `isdst`.

Пример:

```
log (os.time()) -- текущее время в формате posix

local datetime = {year = 2017, month = 03, day = 1, hour = 14, min = 23, sec =
8}
log(os.time(datetime))  -- время, переданное в параметре
```

14 Класс Matches

Класс `Matches` содержит результат функций `Re.match()` и `Re.search()`.

14.1 Метод getFirst

Вызов:

```
position, err = matches:getFirst(group)
```

Параметры:

- `group (int, string)` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

Возвращает:

- `position (int)` – первая позиция (в байтах) исходной строки;
- `err (string)` - сообщение об ошибке или `nil`.

14.2 Метод getLength

Вызов:

```
position, err = matches:getLength(group)
```

Параметры:

- `group (int, string)` – позиция (или имя группы) найденных результатов, начинающаяся с 1.

Возвращает:

- `length (int)` – длина исходной строки в байтах;
- `err (string)` - сообщение об ошибке или `nil`.

14.3 Метод getSize

Вызов:

```
size, err = matches:getSize()
```

Возвращает:

- `size (int)` – количество найденных групп;
- `err (string)` - сообщение об ошибке или `nil`.

14.4 Метод getString

Вызов:

```
substr, err = matches:getString(group, subject)
```

Параметры:

- `group` (`int`, `string`) – позиция (или имя группы) найденных результатов, начинающаяся с 1;
- `subject` (`string`) – исходная строка. **Внимание**: объект `Matches` сохраняет только смещения и не хранит исходную строку. Таким образом, необходимо передать ту же строку, которая использовалась для поиска.

Возвращает:

- `substr` (`string`) – найденная подстрока;
- `err` (`string`) - сообщение об ошибке или `nil`.

14.5 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)
```