ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

Программное обеспечение МойОфис Комплект Средств Разработки (SDK)

MyOffice Document Application Programming Interface (API)

Библиотека для языка программирования C#

26.2.0

Руководство программиста

На 528 листах

Версия документа: 1

Дата публикации: 18.06.2026

Москва
2026

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

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

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

Таблица 1 - Сокращения и расшифровки

Сокращение	Расшифровка
ОС	Операционная система
MyOffice Document API	Программное обеспечение «МойОфис Комплект Средств Разработки (SDK). MyOffice Document API. Библиотека для языка программирования C#»
API	Application Programming Interface (программный интерфейс приложения)
IDE	Integrated Development Environment (интегрированная среда разработки)
SDK	Software Development Kit (комплект для разработки программного обеспечения)

1 ОБЩИЕ СВЕДЕНИЯ

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

1.1 Назначение библиотеки

Библиотека MyOffice Document API для языка программирования C# предназначена для использования в составе прикладных информационных систем или отдельных приложений на платформе Microsoft.NET Framework для решения задач по созданию и наполнению текстовых и табличных документов в пакетном режиме.

1.2 Библиотека MyOffice Document API для языка программирования C#

Библиотека MyOffice Document API для языка программирования C# предоставляет возможность выполнения следующих операций:

1. Создание, открытие, сохранение изменений в электронных текстовых и табличных документах в следующих форматах:
   - текстовые и табличные документы, создаваемые с помощью Microsoft Office в формате OOXML, расширения файлов DOCX и XLSX;
   - текстовые и табличные документы, создаваемые с помощью LibreOffice в формате ODF, расширения файлов ODT и ODS;
   - текстовые и табличные документы, создаваемые с помощью МойОфис в формате ODF, расширения файлов XODT и XODS;
   - экспорт документов в формате PDF.
2. Изменение содержимого документов в пакетном режиме, в том числе:
   - добавление, удаление, изменение текста абзаца;
   - вставка, удаление, форматирование таблиц в текстовом документе;
   - вставка, удаление, переименование отдельных листов в табличном документе;
   - установка значения ячейки электронной таблицы и расчет формул;
   - оформление документа с использованием различных шрифтов и цветового оформления.
3. Поиск и замена фрагмента текста в документе.
4. Управление режимом рецензирования документа, отслеживание изменений в документе.
5. Управление закладками в текстовом документе.

6. Написание и запуск макрокоманд.

Для управления содержимым документа используется объектная модель, представляющая собой совокупность структур данных текстового или табличного документа.

1.3 Уровень подготовки пользователя

Пользователь MyOffice Document API должен иметь:

1. Опыт разработки на языке программирования C# для платформы Microsoft.NET Framework.
2. Навык работы со стандартными офисными приложениями.

2 ПОДГОТОВКА К РАБОТЕ

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

2.1 Дистрибутив

Дистрибутив C# MyOffice Document API поставляется в виде архивных файлов (см. Таблицу 2).

Таблица 2 – Список дистрибутивов C# Document API

ОС	Дистрибутив
Microsoft Windows	MyOfficeSDKDocumentAPI_CSharp_Win_26.2.zip
Linux	MyOfficeSDKDocumentAPI_CSharp_Linux_26.2.zip

2.2 Установка

Для установки MyOffice Document API необходимо извлечь содержимое архивного файла дистрибутива в выбранный каталог для установки MyOffice Document API.

После извлечения в каталоге установки MyOffice Document API будет создана папка `MyOfficeSDKDocumentAPI_CSharp_26.2`, содержащая файлы, приведенные в Таблице 3.

Таблица 3 – Состав дистрибутивов C# Document API

Windows	Linux	Описание
каталог `Resources`		ресурсы приложения

`DocumentAPI.dll`
`NCT.MyOfficeSDK.dll`
`sbb.dll`
`sbb.lib`
	`libDocumentAPI.so`
	`NCT.MyOfficeSDK.dll`
	`libsbb.so`
		файлы библиотек

`EULA_ru.hml`		текст лицензионного соглашения на использование набора средств разработки «МойОфис SDK»
`TPL_ru.html`		перечисление библиотек, использовавшихся при

	разработке приложения

2.3 Пример приложения

Сборка приложения осуществляется с использованием IDE (MS Visual Studio или MS Visual Studio Code). Ниже приведен тестовый пример исходного кода, содержащий методы MyOffice Document API, которые позволят создать текстовый документ, вставить в него текст, а затем сохранить документ с заданным именем и расширением:

```
using NCT.MyOfficeSDK;

namespace Example {
    class Program {
        static void Main(string[] args) {
            try {
                Application application = new Application();
                var doc = application.createDocument(DocumentType.Text);
                doc.getRange().getBegin().insertText("Hello, World!");
                var outputPath = "Example.docx";
                doc.saveAs(outputPath);
            } catch {
                Console.WriteLine("Error");
            }
        }
    }
}
```

2.4 Сборка приложения в среде Microsoft Visual Studio

В данном разделе описаны настройка и сборка проекта в среде Microsoft Visual Studio (OS Windows) с использованием библиотеки MyOffice Document API для языка C#.

Для начала необходимо запустить Microsoft Visual Studio и создать новый проект, выбрав следующие настройки:

- тип проекта: консольное приложение C#;
- имя проекта: Example;
- папка расположения: любая, например, `C:\Project`;

– имя решения: Example.

После создания проекта необходимо открыть **Диспетчер конфигураций** (см. Рисунок 1) и создать платформу проекта x64 (см. Рисунок 2).

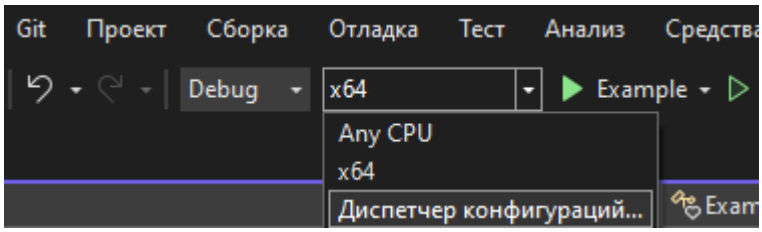


Рисунок 1 – Выбор диспетчера конфигурации в Microsoft Visual Studio

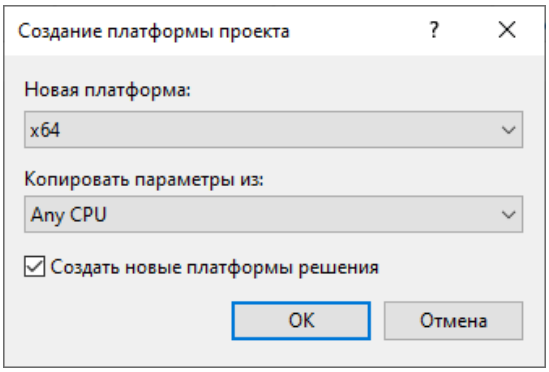


Рисунок 2 – Окно создания платформы проекта

Для предварительной сборки нужно выбрать пункт меню **Собрать решение** во вкладке **Сборка**.

После предварительной сборки необходимо открыть папку проекта и перейти в каталог `\bin\x64\Debug\<NET_CORE_API_FOLDER>` (название папки `NET_CORE_API_FOLDER` зависит от версии .NET, например, она может называться `netcoreapp3.1`, `net6.0`, `net8.0`. Данная папка может отсутствовать, в этом случае нужно найти папку, содержащую файл `Example.exe`, например, `\bin\x64\Debug`).

Скопировать в текущий каталог файлы `DocumentAPI.dll`, `NCT.MyOfficeSDK.dll`, `sbb.dll` и папку `Resources` из папки `MyOfficeSDKDocumentAPI_CSharp_26.2` каталога установки MyOffice Document API.

Далее в Microsoft Visual Studio в окне **Обозреватель решений** (см. Рисунок 1) нужно выбрать раздел **Зависимости,** нажать правую кнопку мыши и выбрать пункт **Добавить ссылку на модель COM**.

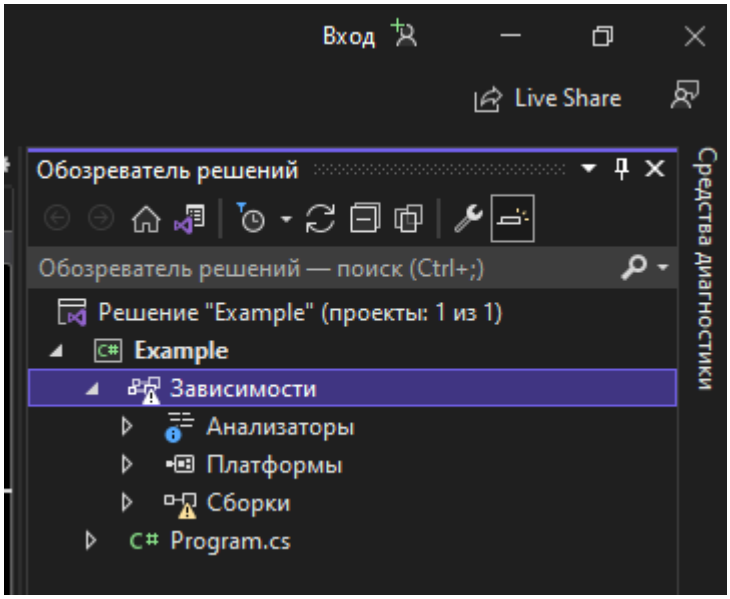


Рисунок 3 – Окно обозревателя решений Microsoft Visual Studio

В открывшемся окне нажать кнопку **Обзор** и выбрать из папки `MyOfficeSDKDocumentAPI_CSharp_26.2` каталога установки MyOffice Document API файл `NCT.MyOfficeSDK.dll` (см. Рисунок 1).

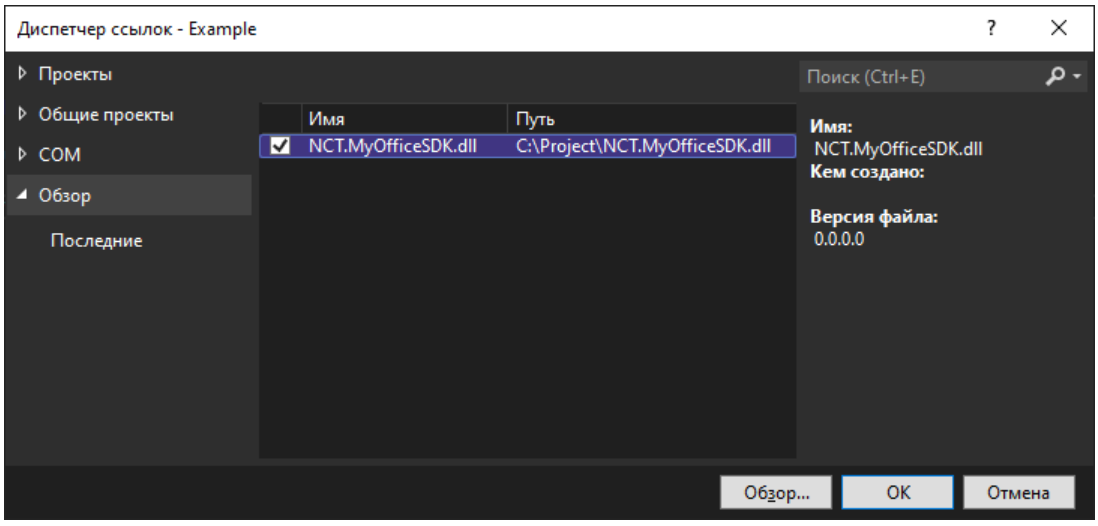


Рисунок 4 – Окно выбора файла библиотеки

Для окончательной сборки приложения в командном меню Microsoft Visual Studio нужно выбрать пункт **Сборка > Собрать решение**.

Для проверки работоспособности MyOffice Document API необходимо запустить собранное приложение, выбрав в командном меню пункт **Отладка > Запуск без отладки**.

В результате выполнения приложения в каталоге проекта в папке `\bin\x64\Debug\<NET_CORE_API_FOLDER>` (название папки `NET_CORE_API_FOLDER` зависит от версии .NET, например, она может называться `netcoreapp3.1`, `net6.0`, `net8.0`) будет создан файл `Example.docx`, а в окне консоли отладки Microsoft Visual Studio не будет сообщений об ошибках.

2.5 Сборка приложения в среде Microsoft Visual Studio Code

В данном разделе описаны настройка и сборка проекта в среде MS Visual Studio Code с использованием библиотеки MyOffice Document API для языка C#.

Описание является актуальным как для OS Windows, так и для OS Linux, т.к. среда VS Code может быть установлена на обе операционные системы.

Для начала необходимо запустить Microsoft Visual Studio Code и установить следующие расширения (см. Рисунок 1):

- .NET Install Tool (SDK and Runtime);
- C#, Base language support;
- C# Dev Kit (Official C# extension from Microsoft).

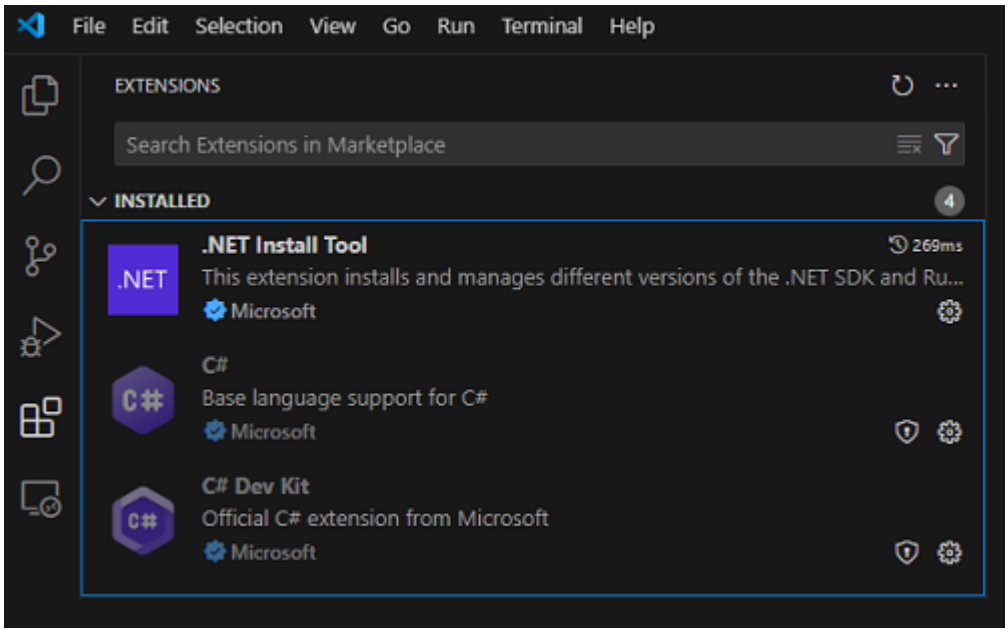


Рисунок 5 – Установка расширений C# в Microsoft Studio Code

Далее следует нажать сочетание клавиш **Ctrl+Shift+P**, на экране появится поле ввода с возможностью выбора. Для создания проекта необходимо выбрать **.NET: New Project** (см. Рисунок 1):

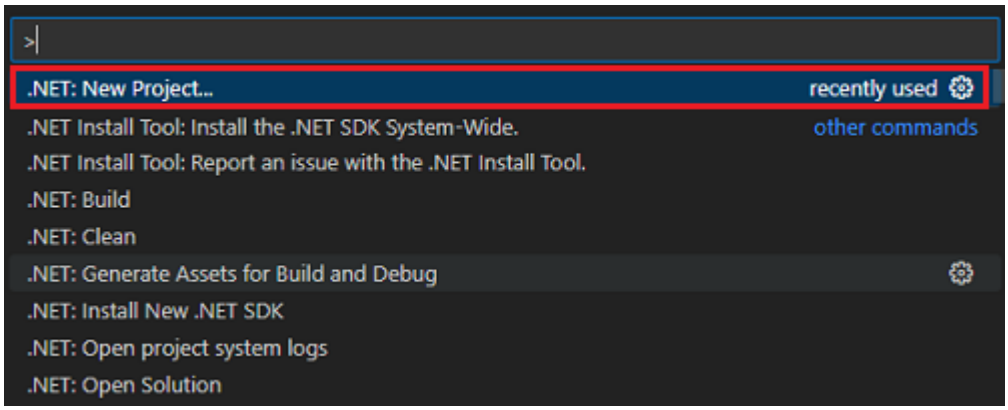


Рисунок 6 – Создание нового проекта

На экране появится список возможных типов проекта, следует выбрать **ConsoleApp** (см. Рисунок 1):

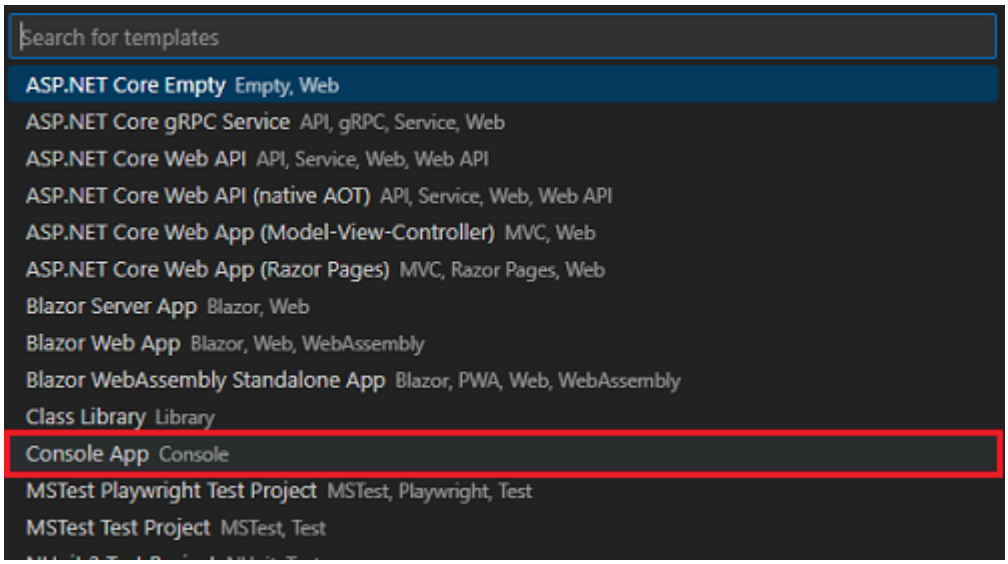


Рисунок 7 – Выбор типа проекта

Далее будет предложено выбрать имя проекта (см. Рисунок 1):

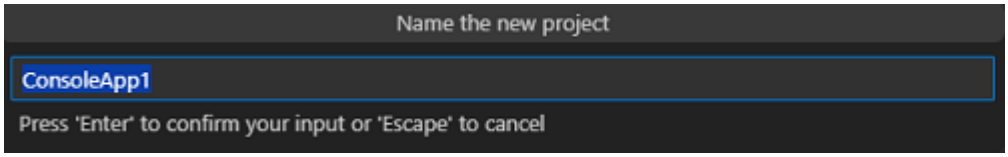


Рисунок 8 – Выбор имени проекта

Выбор папки расположения проекта (см. Рисунок 1):

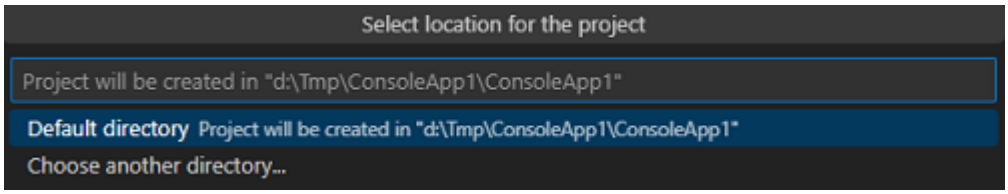


Рисунок 9 – Выбор расположения проекта

Далее следует подтвердить создание проекта (см. Рисунок 1):

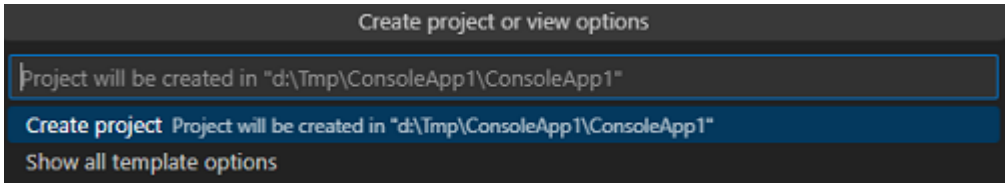


Рисунок 10 – Подтверждение создания проекта

Проект создан, необходимо запустить сборку (меню **Run > Run without debugging**, **Ctrl+F5**), и убедиться, что в строке терминала отображается вывод `"Hello, World!"` (см. Рисунок 1).

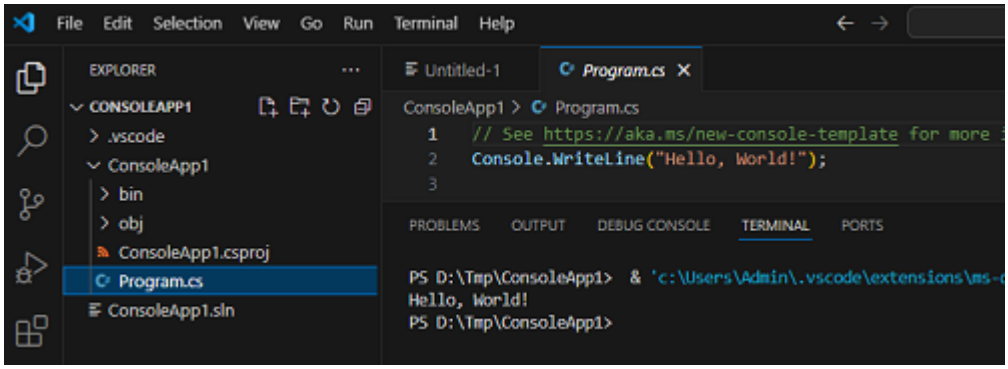


Рисунок 11 – Созданный проект, вывод в консоли

Далее необходимо заменить содержимое файла `Program.cs` программным кодом, приведенным в разделе Пример приложения.

В существующий файл `<APP_NAME>.csproj` добавить следующую секцию `<ItemGroup>`:

```
<Project Sdk="Microsoft.NET.Sdk">
  ......................
  <ItemGroup>
    <Reference Include="NCT.MyOfficeSDK">
      <HintPath>NCT.MyOfficeSDK.dll</HintPath>
    </Reference>
  </ItemGroup>
  ......................
</Project>
```

Затем перейти в каталог `\bin\x64\Debug\<NET_CORE_API_FOLDER>` (название папки `NET_CORE_API_FOLDER` зависит от версии .NET, например, она может называться `netcoreapp3.1, net6.0, net8.0`).

Скопировать в указанный каталог файлы библиотек, приведенные в таблице раздела Установка, а также папку `Resources`.

Скопировать файл `NCT.MyOfficeSDK.dll` в корневую папку приложения, где находится файл `Program.cs`.

Для проверки работоспособности MyOffice Document API необходимо запустить собранное приложение, выбрав в командном меню пункт **Run > Start Debugging** или **Run > Start Without Debugging**.

В результате выполнения приложения в каталоге проекта в папке `\bin\x64\Debug\<NET_CORE_API_FOLDER>` (название папки `NET_CORE_API_FOLDER` зависит от версии .NET, например, она может называться `netcoreapp3.1, net6.0, net8.0`) будет создан файл `Example.docx`, а в окне консоли отладки Microsoft Visual Studio Code не будет сообщений об ошибках.

2.6 Распространение разработанных приложений

Для распространения разработанных приложений, использующих вызовы MyOffice Document API, нужно обеспечить наличие следующих объектов в каталоге с распространяемым приложением `<APP_NAME>`:

1. Исполняемый файл приложения `<APP_NAME>.exe`, библиотека `<APP_NAME>.dll`, файл конфигурации `<APP_NAME>.runtimeconfig.json`.

2. Папка `Resources`, содержащая ресурсы библиотеки (скопировать папку `MyOfficeSDKDocumentAPI_CSharp_26.2\Resources` каталога установки MyOffice Document API).
3. Файлы библиотек, приведенных в таблице раздела Установка.

3 ОБЪЕКТНАЯ МОДЕЛЬ МОЙОФИС C# SDK

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

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

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

Для этого используются классы, расположенные в пространстве имен (namespace) `NCT.MyOfficeSDK` (см. Рисунок 1). `NCT.MyOfficeSDK` содержит основной класс `NCT.MyOfficeSDK.Application`, который служит для создания и открытия документа, а также классы и функции для представления документа и всех его составляющих, которые поддерживает МойОфис: абзацы, таблицы, ячейки, рисунки, колонтитулы и т.д.

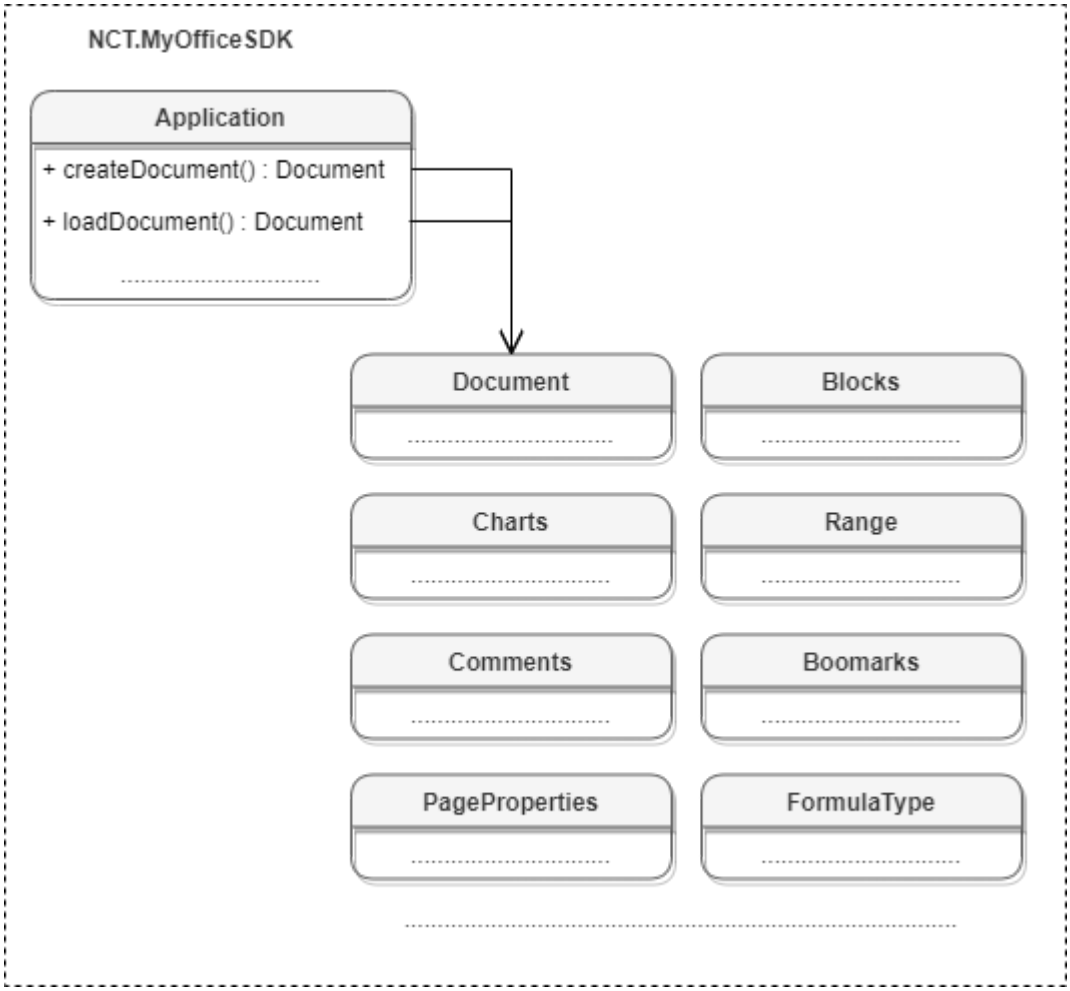


Рисунок 12 – Объектная модель МойОфис C# SDK.

4 РАБОТА С ДОКУМЕНТАМИ

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

4.1 Работа с текстовым документом

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

4.1.1 Создание и открытие текстового документа

Метод `Application.createDocument` создает документ. В качестве параметра используются `DocumentType` или `DocumentSettings`.

Примеры создания текстового документа

```
var document = application.createDocument(DocumentType.Text);
```

```
var documentSettings = new DocumentSettings();
documentSettings.documentType = DocumentType.Text;
documentSettings.localeInfo = new LocaleInfo();
documentSettings.localeInfo.decimalSeparator = ",";
var document = application.createDocument(documentSettings);
```

Метод `Application.loadDocument` загружает документ. В качестве параметра используется путь к документу. Дополнительно может быть использован параметр `LoadDocumentSettings`.

Примеры загрузки текстового документа

```
var document = application.loadDocument("C:/Work/text.docx");
```

```
DocumentSettings documentSettings = new DocumentSettings();
documentSettings.documentType = DocumentType.Text;
LoadDocumentSettings loadSettings = new LoadDocumentSettings();
loadSettings.commonDocumentSettings = documentSettings;

Application application = new Application();
var document = application.loadDocument("C:/Work/text.docx", loadSettings);
```

4.1.2 Сохранение и экспорт текстового документа

Метод `Document.saveAs` сохраняет документ по указанному пути.

Примеры сохранения текстового документа

```
document.saveAs(filePath);
```

```
SaveDocumentSettings saveDocumentSettings = new SaveDocumentSettings();

saveDocumentSettings.documentFormat = DocumentFormat.OXML;
saveDocumentSettings.documentType = DocumentType.Text;
saveDocumentSettings.documentPassword = "password";
saveDocumentSettings.isTemplate = false;

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

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

Метод `Document.exportAs` экспортирует документ в файл по указанному пути с заданным форматом типа `ExportFormat`.

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

Примеры экспорта текстового документа

```
document.exportAs(filePath, ExportFormat.PDFA1);
```

```
TextExportSettings textExportSettings = new TextExportSettings();
textExportSettings.pageNumbers = new PageNumbers(PageParity.Even);
document.exportAs(filePath, ExportFormat.PDFA1, textExportSettings);
```

4.1.3 Разделы (секции) документа

На рисунке 1 изображена объектная модель классов, относящихся к работе с секциями текстового документа.

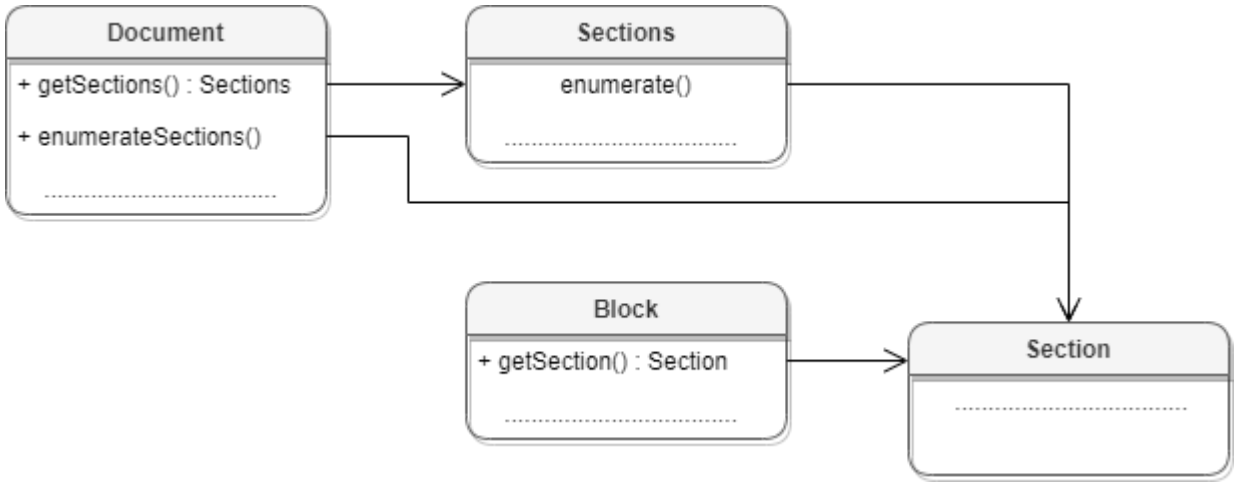


Рисунок 13 – Объектная модель классов для работы с секциями

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

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

- получение объекта `Sections` с помощью вызова `Document.getSections()`;
- перечисление всех доступных секций `Section` с помощью вызова `Document.getSectionsEnumerator()`;
- получение секции `Section` вызовом метода `Block.getSection()` для блока, который входит в секцию.

Примеры

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getPageProperties().width);
}
```

```
SectionsEnumerator sectionsEnumerator = document.getSectionsEnumerator();
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getPageProperties().width);
}
```

```
Block block = document.getBlocks().getBlock(0);
Section section = block.getSection();
if (section == null)
{
    section = block.getSection();
    Console.WriteLine(section.getPageProperties().width);
}
```

4.1.3.1 Работа с колонтитулами раздела

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

Пример

```
var section = document.getSectionsEnumerator().Current;
var header = section.getHeaders().GetEnumerator().Current;
var footer = section.getFooters().GetEnumerator().Current;

Console.WriteLine(header.getRange().extractText());
Console.WriteLine(footer.getRange().extractText());
```

4.1.3.2 Управление ориентацией и свойствами страниц раздела

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

```
var section = document.getBlocks().getBlock(0).getSection();
section.setPageOrientation(PageOrientation.Portrait);
```

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

```
PageProperties pageProps = new PageProperties();
pageProps.width = 350.0f;
pageProps.height = 800.0f;

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

Ориентация страниц может быть установлена для каждого раздела документа. Список разделов документа может быть получен из объекта Document.

```
var sectionsEnumerator = document.getSectionsEnumerator();
foreach (var section in sectionsEnumerator) {
    section.setPageOrientation(PageOrientation.Portrait);
}
```

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

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

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

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

4.1.4 Встроенные объекты в текстовом документе

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

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

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

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

4.1.4.1 Вставка изображения

Для вставки изображения используется метод Position.insertImage().

Вставка изображения в текстовый документ

```
Range range = document.getRange();
Image insertedImage = range.getBegin().insertImage("C://Tmp//123.jpg", new
SizeU(50, 50));
```

Вставка изображения в колонтитулы текстового документа

```
var sections = document.getSections();
foreach (Section section in sections) {
    var footers = section.getFooters();
    foreach (HeaderFooter footer in footers) {
        Position pos = footer.getRange().getBegin();
        pos.insertImage("logo.jpg", new SizeU(100, 50));
    }
}
```

4.1.4.2 Перечисление встроенных объектов

Перечисление графических объектов в текстовом документе

```
var mediaObjects = document.getRange().getInlineObjects();
var enumerator = mediaObjects.GetEnumerator();
// Перебор коллекции встроенных объектов
foreach (var mediaObject in enumerator) {
    var image = mediaObject.toImage();
    if (image != null) {
        Console.WriteLine("Image");
    } else {
        Console.WriteLine("Shape");
    }
}
```

Перечисление изображений в текстовом документе

```
var images = document.getRange().getImages();
var enumerator = images.GetEnumerator();
foreach (var image in enumerator) {
```

```
    Console.WriteLine("Image");
}
```

Перечисление изображений в таблице текстового документа

```
Table table = document.getBlocks().getTable(0);
Images images = table.getImages();
ImagesEnumerator imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator) {
    Console.WriteLine("Image");
}
```

4.1.5 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены на страницах документа. Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 1).

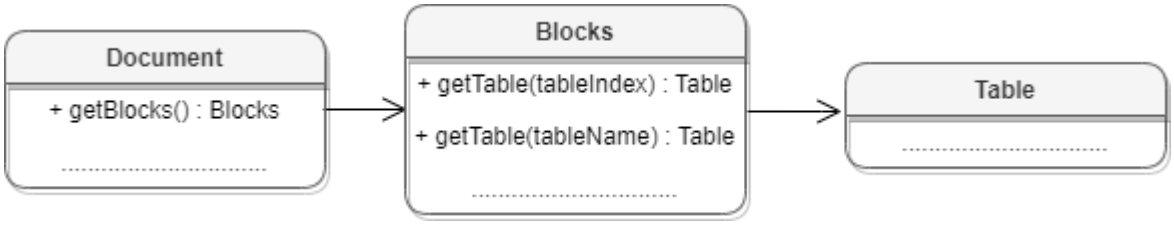


Рисунок 14 – Объектная модель для работы с таблицами

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()`.

```
TablesEnumerator tablesEnumerator = document.getBlocks().getTablesEnumerator();
foreach (var table in tablesEnumerator)
```

```
{
    Console.WriteLine(table.getName());
}
```

Получение таблицы текстового документа

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

```
Table table = document.getBlocks().getTable(0);
```

```
Table table = document.getBlocks().getTable("Таблица1");
```

Вставка таблицы в текстовый документ

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

```
Range range = document.getRange();
Position beginPosition = range.getBegin();
Table table = beginPosition.insertTable(3, 3, "Table");
```

Переименование таблицы

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

```
String tableName = "Table1";
Table table = document.getBlocks().getTable(0);
table.setName(tableName);
table = document.getBlocks().getTable(tableName);
```

Удаление таблицы

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

```
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    table.remove();
}
```

4.1.6 Работа с закладками

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

Доступны следующие операции с закладками:

Вставка закладки в указанное местоположение

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

Удаление закладки с заданным именем

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

Поиск закладки по имени

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

Замена текстового содержимого закладки

```
Bookmarks bookmarks = document.getBookmarks();
Range bookmarkRange = bookmarks.getBookmarkRange("Bookmark");
if (bookmarkRange != null) {
    bookmarkRange.getBegin().replaceText("New bookmark text");
}
```

Вставка текста в закладку

```
Bookmarks bookmarks = document.getBookmarks();
Range bookmarkRange = bookmarks.getBookmarkRange("Bookmark");
if (bookmarkRange != null) {
    bookmarkRange.getBegin().insertText("New bookmark text");
}
```

Удаление содержимого закладки

```
Bookmarks bookmarks = document.getBookmarks();
Range bookmarkRange = bookmarks.getBookmarkRange("Bookmark");
if (bookmarkRange != null) {
```

```
    bookmarkRange.getBegin().removeBackward();
}
```

Получение текстового содержимого закладки

```
Bookmarks bookmarks = document.getBookmarks();
Range bookmarkRange = bookmarks.getBookmarkRange("Bookmark");
if (bookmarkRange != null) {
    Console.WriteLine(bookmarkRange.extractText());
}
```

Вставка таблицы в закладку

```
Bookmarks bookmarks = document.getBookmarks();
Range bookmarkRange = bookmarks.getBookmarkRange("Bookmark");
if (bookmarkRange != null) {
    bookmarkRange.getEnd().insertTable(3, 3, "signers_list");
}
```

4.1.7 Рецензирование документов

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

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

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

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

Пример

```
document.setChangesTrackingEnabled(true);
Console.WriteLine(document.isChangesTrackingEnabled());
```

Инструменты рецензирования применяются к диапазону документа, по этой причине методы доступа к ним находятся в классе `Range` (см. Рисунок 1).

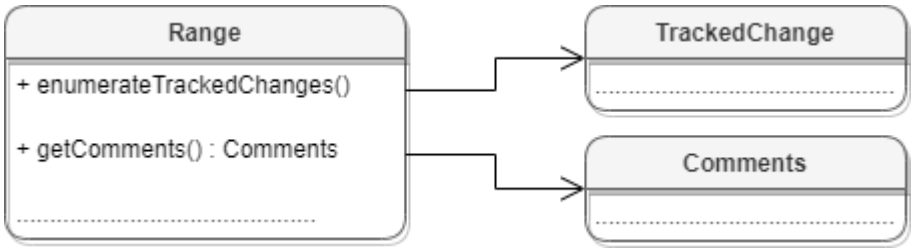


Рисунок 15 – Инструменты рецензирования документа

4.1.8 Работа с элементами управления

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

Используйте метод `Document.getContentControls()` чтобы получить элементы управления из текущего документа:

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

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

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

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

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

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

```
inputField.setValue(inputField.getValue().Replace(' ', '_'));
```

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

```
DropListControl comboBox = controls.findByTitle("select").toDropList();
comboBox.setValue(comboBox.getChoices().IndexOf("two"));
```

4.2 Работа с табличным документом

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

4.2.1 Создание и открытие табличного документа

Метод `Application.createDocument` создает документ. В качестве параметра используется класс `DocumentType` или `DocumentSettings`. Для создания табличного документа необходимо выбрать тип `DocumentType.Workbook`.

Пример создания табличного документа

```
var document = application.createDocument(DocumentType.Workbook);
```

Метод `Application.loadDocument` открывает документ. В качестве параметра используется путь к документу. Дополнительно может быть использован параметр `LoadDocumentSettings`.

Примеры загрузки табличного документа

```
var document = application.loadDocument("C:/Work/sheet.xlsx");
```

```
DocumentSettings documentSettings = new DocumentSettings();
documentSettings.documentType = DocumentType.Workbook;
LoadDocumentSettings loadSettings = new LoadDocumentSettings();
loadSettings.commonDocumentSettings = documentSettings;

Application application = new Application();
var document = application.loadDocument("C:/Work/sheet.xlsx", loadSettings);
```

4.2.2 Сохранение и экспорт табличного документа

Метод `Document.saveAs` сохраняет документ по указанному пути.

Примеры сохранения табличного документа

```
document.saveAs(filePath);
```

```
SaveDocumentSettings saveDocumentSettings = new SaveDocumentSettings();

saveDocumentSettings.documentFormat = DocumentFormat.OXML;
saveDocumentSettings.documentType = DocumentType.Workbook;
saveDocumentSettings.documentPassword = "password";
saveDocumentSettings.isTemplate = false;

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

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

Метод `Document.exportAs` экспортирует документ в файл по указанному пути с заданным форматом типа `ExportFormat`.

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

Примеры экспорта табличного документа

```
document.exportAs(filePath, ExportFormat.PDFA1);
```

```
WorkbookExportSettings workbookSettings = new WorkbookExportSettings();
workbookSettings.sheetNames = ["Лист1", "Лист3"];
workbookSettings.printingScope = new PrintingScope(PrintingScope.Type.PrintArea);
workbookSettings.pageProperties = new PageProperties(100, 200);
workbookSettings.scale = 90;
string filePath = "C:\\work\\Sheet.pdf";
document.exportAs(filePath, ExportFormat.PDFA1, workbookSettings);
```

4.2.3 Диаграммы

Работа с диаграммами доступна только в табличных документах (см. Рисунок 1).

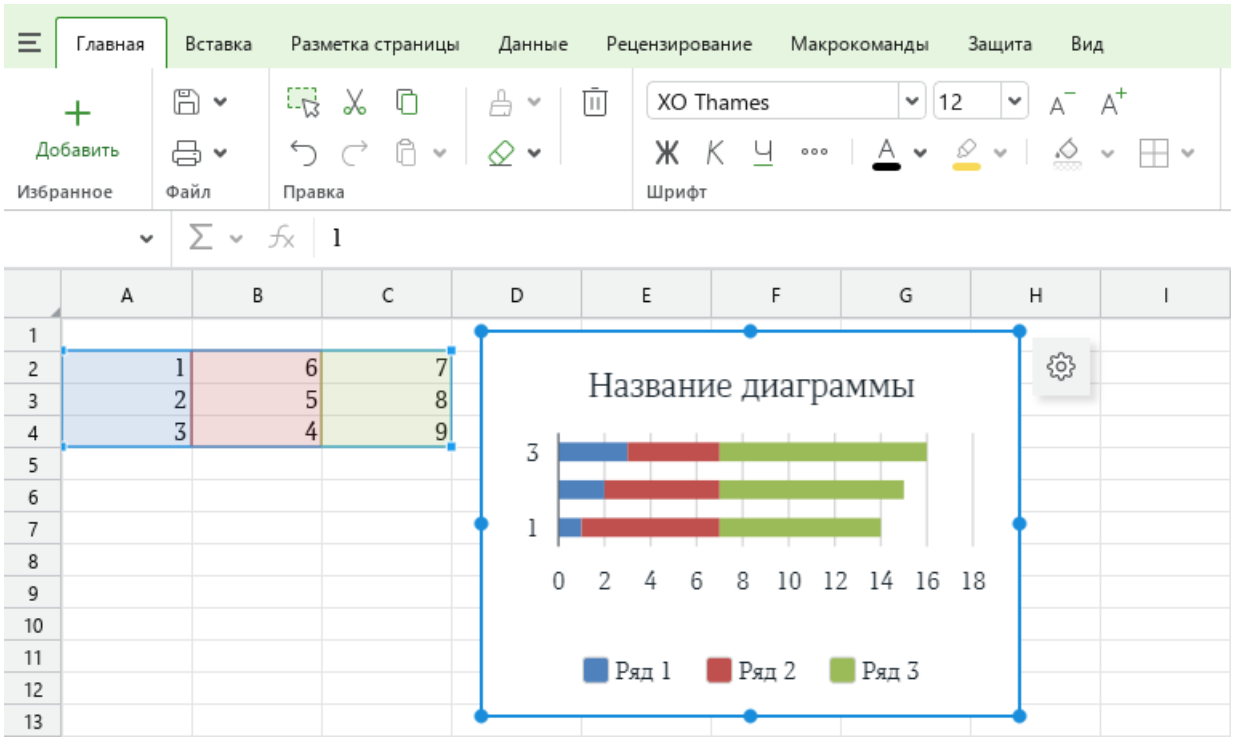


Рисунок 16 – Пример отображения диаграммы в приложении «МояТаблица»

На рисунке 1 изображена объектная модель классов, относящихся к работе с диаграммами.

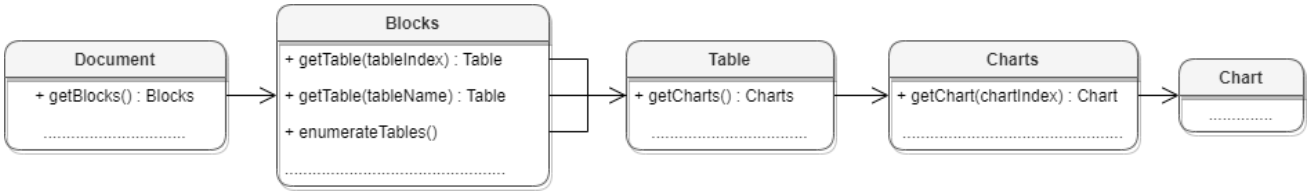


Рисунок 17 – Классы для работы с диаграммами

Для доступа к списку диаграмм используется метод таблицы (листа документа) `Table.getCharts()`.

Для получения диаграммы `Chart` используется метод `Charts.getChart()`.

Удаление диаграмм в текущей версии не поддерживаются.

4.2.4 Копирование ячеек в табличном документе

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

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

```
Table table = document.getBlocks().getTable(0);

var leftTopCellPosition = new CellPosition(0, 0);
var rightBottomCellPosition = new CellPosition(1, 1);
var srcCellRangePosition = new CellRangePosition(leftTopCellPosition,
rightBottomCellPosition);

var strTargetRange = "E6:F7";
var sheetList = document.getBlocks().getTable(0);
var sourceRange = sheetList.getCellRange(srcCellRangePosition);
var destRange = sheetList.getCellRange(strTargetRange);

sourceRange.copyInto(destRange);
```

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

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

Методы `CellRange.copyInto()` и `CellRange.moveInto()` также позволяют копировать и перемещать ячейки между документами и листами документа:

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

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

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

sourceRange.copyInto(targetRange);
```

4.2.5 Работа с формулами

Метод Cell.setFormula позволяет поместить формулу в ячейку таблицы:

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

Также при создании формулы можно использовать именованные диапазоны для обозначения группы ячеек.

Используйте метод Cell.isFormula, чтобы определить, содержит ли текущая ячейка формулу. Из ячейки с формулой, можно получить текст формулы (Cell.getFormulaAsString) или результат вычисления (Cell.getRawValue или Cell.getFormattedValue):

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("C1");
if (cell.isFormula()) {
    cell.getFormulaAsString();    // =AVERAGE(B:B)
    cell.getRawValue();           // 1.5
    cell.getFormattedValue();     // 150.0%
}
```

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

```
if (document.getCalculationMode() == CalculationMode.Auto)
    document.setCalculationMode(CalculationMode.Manual);
```

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

```
if (document.isCalculatedOnSave())
    document.setCalculatedOnSave(false);
```

Эту настройку пересчета формул можно задать непосредственно при сохранении документа. Для этого используйте поле SaveDocumentSettings.allowCalculation.

Если автоматический пересчет формул отключен, вы можете обновить значения всех формул в документе с помощью метода Document.calculateOutdatedFormulas или

использовать метод `calculate()` объектов `Table`, `CellRange` или `Cell` для пересчета формул, находящихся в определенном диапазоне:

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

4.2.6 Проверка данных

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

Проверка данных по списку значений

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

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

```
DataValidation dvList = new DataValidation();
dvList.setType(DataValidationType.List);
dvList.setFormula1("UK; Italy; Germany; Austria; Brazil");
dvList.setShowDropDown(true);
```

```
cellRange.setDataValidation(dvList);
```

Проверка данных в формате Дата

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

1. Создайте экземпляр класса `DataValidation`.
2. Вызовите метод `DataValidation.setType()` с параметром `DataValidationType.Date` для создания проверки дат.
3. Используйте метод `DataValidation.setOperator()`, чтобы задать оператор сравнения.
4. Передайте дату для сравнения в метод `DataValidation.setFormula1()`. Метод принимает значения в виде строки в формате мм/дд/гггг, а также формулы, именованные диапазоны и адреса.
5. Если выбран оператор сравнения `Between` или `NotBetween`, задайте вторую дату для промежутка с помощью метода `DataValidation.setFormula2()`.
6. Используя метод `CellRange.setDataValidation()` примените проверку данных к диапазону ячеек.

```
DataValidation dvDate = new DataValidation();
dvDate.setType(DataValidationType.Date);
dvDate.setOperator(DataValidationOperator.Between);
dvDate.setFormula1("06/01/2024");
dvDate.setFormula2("08/31/2024");

cellRange.setDataValidation(dvDate);
```

Отображение сообщений об ошибках

Добавьте визуальное предупреждение о вводе недопустимых значений:

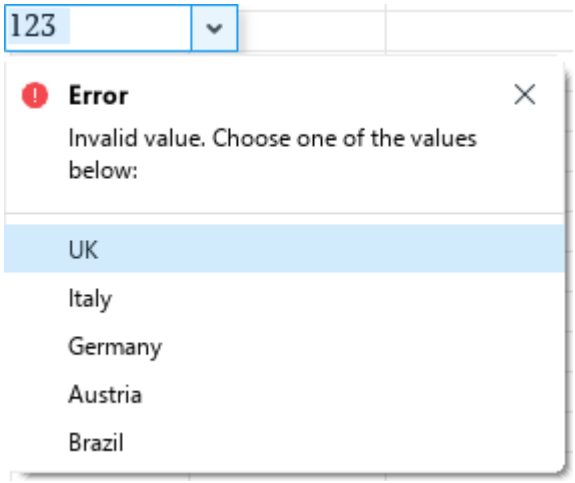


Рисунок 19 – Ошибка проверки данных

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

```
DataValidation validation = new DataValidation();
// ...
// Настройки сообщения об ошибке:
validation.setShowErrorMessage(true);
validation.setErrorStyle(DataValidationErrorStyle.Stop);
validation.setErrorTitle("Error");
validation.setErrorMessage("Invalid value. Choose one of the values below:");

cellRange.setDataValidation(validation);
```

Отображение подсказок

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

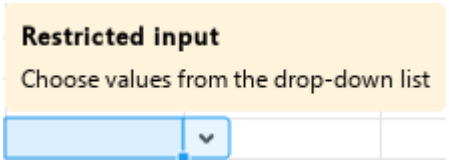


Рисунок 21 – Подсказка при вводе значения в ячейку с проверкой

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

```
DataValidation validation = new DataValidation();
// ...
// Настройки подсказки:
validation.setShowInputMessage(true);
validation.setPromptTitle("Restricted input");
validation.setPrompt("Choose values from the drop-down list");

cellRange.setDataValidation(validation);
```

Обработка результатов проверки

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

```
if (cell.getDataValidation() == null) {
    Console.WriteLine("No validation applied");
} else if (cell.checkDataValidation().isValid()) {
    Console.WriteLine("Validation passed");
} else {
    DataValidation validation = cell.checkDataValidation().getDataValidation();
    Console.WriteLine(validation.getErrorMessage());
}
```

Получение настроек проверок

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

Удаление проверок

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

4.2.7 Встроенные объекты в табличном документе

Редакторы таблиц МойОфис поддерживают графические объекты типа Image, Shape, Chart.

Объектная модель табличного документа в части управления изображениями развивается и дополняется возможностями. На данный момент доступны следующие операции:

- вставка изображений в табличный документ;
- перечисление встроенных объектов, находящихся в табличном документе, определение их типа и геометрических размеров;
- перемещение встроенных объектов, изменение их размеров и масштаба.

Доступ ко встроенным объектам табличного документа осуществляется посредством использования методов Range.getInlineObjects(), Table.getImages(), Table.getMediaObjects().

4.2.7.1 Вставка изображения

Для вставки изображения используется метод Table.insertImage().

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

4.2.7.2 Перечисление встроенных объектов

Список изображений в листе табличного документа может быть получен с помощью метода Table.getImages(), вызванного у объекта листа документа.

Перечисление изображений табличного документа

```
Table firstSheet = document.getBlocks().getTable(0);
Images images = firstSheet.getImages();
ImagesEnumerator imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    ..........................
}
```

Список всех встроенных объектов в листе табличного документа может быть получен с помощью метода `Table.getMediaObjects()`, вызванного у объекта листа документа.

Перечисление встроенных объектов табличного документа

```
Table firstSheet = document.getBlocks().getTable(0);
MediaObjects mediaObjects = firstSheet.getMediaObjects();
MediaObjectsEnumerator mediaObjectsEnumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    ..........................
}
```

4.2.8 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 1).

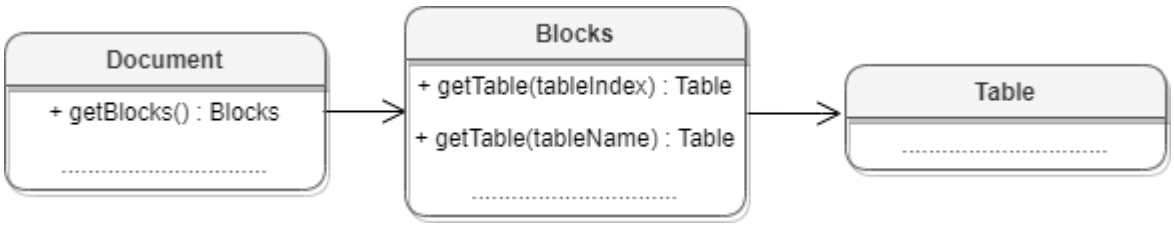


Рисунок 22 – Объектная модель для работы с таблицами

Получение листа табличного документа

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

```
Table table = document.getBlocks().getTable(0);
```

```
Table table = document.getBlocks().getTable("Лист1");
```

Перечисление страниц табличного документа

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

```
TablesEnumerator tablesEnumerator = document.getBlocks().getTablesEnumerator();
foreach (var table in tablesEnumerator)
{
    Console.WriteLine(table.getName());
}
```

Также доступен вариант перечисления листов документа посредством использования метода Blocks.getEnumerator() с дальнейшим преобразованием блока в таблицу.

```
BlocksEnumerator blocksEnumerator = document.getBlocks().GetEnumerator();
foreach (var block in blocksEnumerator)
{
    Table table = block.toTable();
    if (table != null)
    {
        Console.WriteLine(table.getName());
    }
}
```

Вставка страницы в табличный документ

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

```
Position position = document.getRange().getEnd();
position.insertTable(4, 3, "Лист2");
```

Переименование страницы

Для переименования таблицы используется метод Table.setName().

```
String tableName = "Table1";
Table table = document.getBlocks().getTable(0);
table.setName(tableName);
table = document.getBlocks().getTable(tableName);
```

Скрытие и отображение страниц табличного документа

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

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

Копирование страницы

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

```
Table table = document.getBlocks().getTable(0);
table.duplicate();
```

Удаление страницы

Для удаления таблицы используется метод Table.remove().

```
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    table.remove();
}
```

4.2.9 Работа со сводными таблицами

Сводная таблица - инструмент обработки данных, служащий для их обобщения и удобства обработки. Схема взаимодействия объектов, связанных со сводными таблицами, приведена на рисунке 1.

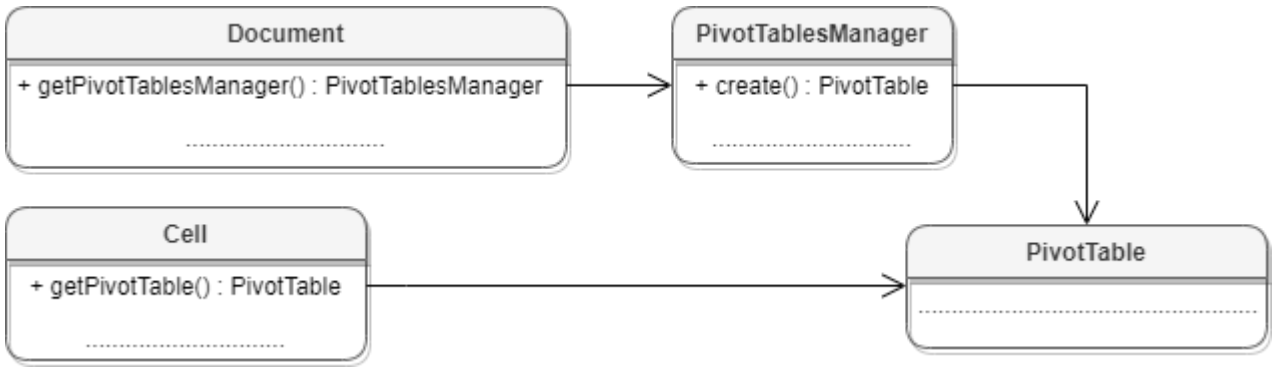


Рисунок 23 – Сводные таблицы

4.2.9.1 Создание сводной таблицы

Метод PivotTablesManager.create используется для добавления сводной таблицы в документ.

Пример

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

4.2.9.2 Получение диапазона исходных данных сводной таблицы

Для получения диапазона исходных данных сводной таблицы используется метод PivotTable.getSourceRange().

Пример

```
var table = document.getBlocks().getTable(1);
// Получаем ячейку, находящуюся в диапазоне исходных данных сводной таблицы
var pivotRootCell = table.getCell(new CellPosition(2, 0));

// Получаем сводную таблицу
var pivotTable = pivotRootCell.getPivotTable();

// Получаем диапазон исходных данных сводной таблицы
var sourceCellRange = pivotTable.getSourceRange();
```

```
// Для получения границ диапазона используем поля CellRange
Console.WriteLine(sourceCellRange.getBeginRow());
Console.WriteLine(sourceCellRange.getBeginColumn());
Console.WriteLine(sourceCellRange.getLastRow());
Console.WriteLine(sourceCellRange.getLastColumn());
```

4.2.9.3 Получение диапазона размещения сводной таблицы

Для получения диапазона размещения сводной таблицы используется метод PivotTable.getPivotRange().

Пример

```
// Получаем диапазон размещения сводной таблицы
var pivotCellRange = pivotTable.getPivotRange();
```

4.2.9.4 Получение флагов отображения общих итогов для строк и колонок

Для получения флагов отображения общих итогов для строк и колонок используются методы PivotTable.isRowGrandTotalEnabled(), PivotTable.isColumnGrandTotalEnabled().

Пример

```
// Получаем флаги отображения общих итогов для строк и колонок
var isRowGrandTotalEnabled = pivotTable.isRowGrandTotalEnabled();
var isColGrandTotalEnabled = pivotTable.isColumnGrandTotalEnabled();
```

4.2.9.5 Получение заголовков сводной таблицы

Для получения заголовков сводной таблицы используется метод PivotTable.getPivotTableCaptions().

Пример

```
PivotTableCaptions captions = pivotTable.getPivotTableCaptions();

// Используем поля структуры PivotTableCaptions:
Console.WriteLine(captions.emptyCaption);
Console.WriteLine(captions.errorCaption);
```

```
Console.WriteLine(captions.rowHeaderCaption);
Console.WriteLine(captions.columnHeaderCaption);
Console.WriteLine(captions.valuesHeaderCaption);
```

4.2.9.6 Получение и применение фильтра для сводной таблицы

Для работы с фильтрами сводной таблицы используются методы PivotTable.getFilter(), PivotTableEditor.setFilter().

Пример

```
// По названию поля сводной таблицы получаем фильтр
var filter = pivotTable.getFilter("Category");

// Делаем элементы `Car` и `Technology` скрытыми
filter.setHidden("Car", true);
filter.setHidden("Technology", true);

// Делаем элемент `Furniture` видимым
filter.setHidden("Furniture", false);

// Применяем фильтр к сводной таблице
pivotTable.createPivotTableEditor().setFilter(filter).apply();
```

4.2.9.7 Получение полей из области фильтров

Для получения полей из области фильтров используется метод PivotTable.getPageFields().

Пример

```
// Получение полей из области фильтров
PivotTablePageFields pageFields = pivotTable.getPageFields();
// Перебираем все поля из области фильтров
foreach (var field in pageFields)
{
    var fieldProps = field.fieldProperties;

    //  Далее используем поля структуры PivotTableFieldProperties:
    Console.WriteLine(fieldProps.fieldName);
    Console.WriteLine(fieldProps.fieldAlias);
```

```
    Console.WriteLine(fieldProps.subtotalAlias);
}
```

4.2.9.8 Получение полей из области значений

Для получения полей из области значений используется метод PivotTable.getValueFields().

Пример

```
// Получение полей из области значений
PivotTableValueFields valueFields = pivotTable.getValueFields();
// Перебираем все поля из области значений
foreach (var valueField in valueFields)
{
    //  Далее используем поля структуры PivotTableValueField
    Console.WriteLine(valueField.baseFieldName);
    Console.WriteLine(valueField.valueFieldName);
    Console.WriteLine(valueField.cellNumberFormat);
    Console.WriteLine(valueField.totalFunction);
    Console.WriteLine(valueField.customFormula);
}
```

4.2.9.9 Получение полей из области строк

Для получения полей из области строк используется метод PivotTable.getRowFields().

Пример

```
// Получение полей из области строк
PivotTableCategoryFields rowFields = pivotTable.getRowFields();
// Перебираем все поля из области строк
foreach (var rowField in rowFields)
{
    var fieldProperties = rowField.fieldProperties;
    var subtotalFunctions = rowField.subtotalFunctions;

    //  Далее используем поля структуры PivotTableCategoryField:
    Console.WriteLine(fieldProperties.fieldName);
```

```
    Console.WriteLine(fieldProperties.fieldAlias);
    Console.WriteLine(fieldProperties.subtotalAlias);
}
```

4.2.9.10 Получение полей из области колонок

Для получения полей из области колонок используется метод PivotTable.getColumnFields().

Пример

```
// Получение полей из области колонок
PivotTableCategoryFields columnFields = pivotTable.getColumnFields();
// Перебираем все поля из области колонок
foreach (var columnField in columnFields)
{
    var fieldProperties = columnField.fieldProperties;
    var subtotalFunctions = columnField.subtotalFunctions;

    //  Далее используем поля структуры PivotTableCategoryField:
    Console.WriteLine(fieldProperties.fieldName);
    Console.WriteLine(fieldProperties.fieldAlias);
    Console.WriteLine(fieldProperties.subtotalAlias);
}
```

4.2.9.11 Получение настроек отображения сводной таблицы

Для получения настроек отображения сводной таблицы используется метод PivotTable.getPivotTableLayoutSettings().

Пример

```
PivotTableLayoutSettings layoutSettings =
                pivotTable.getPivotTableLayoutSettings();
//  Далее используем поля структуры PivotTableLayoutSettings:
Console.WriteLine(layoutSettings.reportLayout);
Console.WriteLine(layoutSettings.pageFieldOrder);
Console.WriteLine(layoutSettings.useGridDropZones);
Console.WriteLine(layoutSettings.pageFieldWrapCount);
Console.WriteLine(layoutSettings.displayFieldCaptions);
```

```
Console.WriteLine(layoutSettings.indentForCompactLayout);
Console.WriteLine(layoutSettings.valueFieldsOrientation);
Console.WriteLine(layoutSettings.isMergeAndCenterLabelsEnabled);
```

4.2.9.12 Обновление сводной таблицы

Для обновления сводной таблицы используется метод PivotTable.update(). Метод возвращает значение типа PivotTableUpdateResult.

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

4.2.10 Работа с фильтрами

Работа с фильтрами возможна только в табличном документе. Диаграмма взаимодействия объектов приведена на рисунке 1.

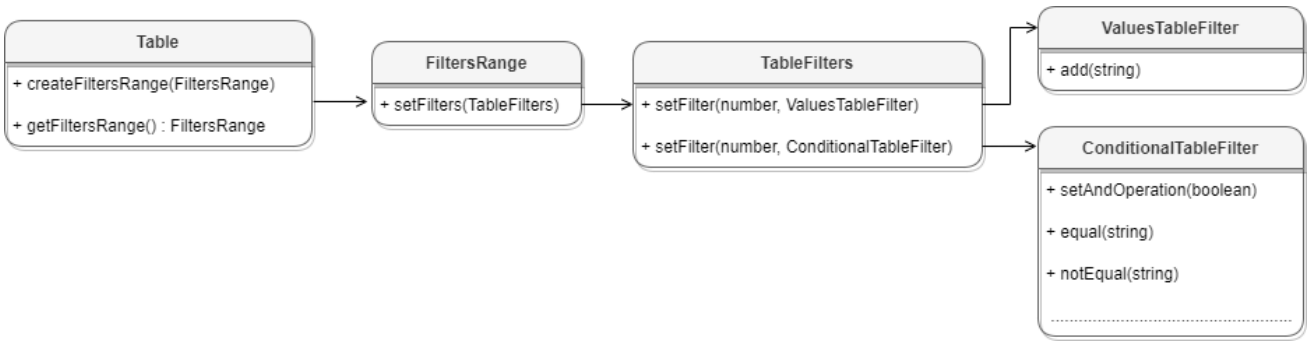


Рисунок 24 – Объектная модель таблиц для работы с фильтрами

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

```
Table sheet = document.getBlocks().getTable("Лист1");

CellRangePosition cellRange = new CellRangePosition(1, 1, 8, 2);
FiltersRange filtersRange = sheet.createFiltersRange(cellRange);

ValuesTableFilter johnPaulFilter = new ValuesTableFilter();
johnPaulFilter.add("John");
johnPaulFilter.add("Paul");

ConditionalTableFilter songFilter = new ConditionalTableFilter();
songFilter.setAndOperation(true);
songFilter.notEqual("");
songFilter.notBegins("TODO");

TableFilters tableFilters = new TableFilters();
tableFilters.setFilter(0, johnPaulFilter);
tableFilters.setFilter(1, songFilter);

filtersRange.setFilters(tableFilters);
```

4.3 Встроенные объекты

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

4.3.1 Определение типа встроенных объектов

Для определения типа графического объекта (Image/Chart/Shape) могут быть использованы методы MediaObject.toImage(), MediaObject.toChart(). В случае, если объект существует, метод вернет ненулевой объект.

Пример

```
var image = mediaObject.toImage();
if (image != null) {
    Console.WriteLine("Image");
```

```
} else {
    var chart = mediaObject.toChart();
    if (chart != null) {
        Console.WriteLine("Chart");
    } else {
        Console.WriteLine("Shape");
    }
}
```

4.3.2 Работа со встроенными объектами

Перечисление встроенных объектов описано в разделах Встроенные объекты в текстовом документе и Встроенные объекты в табличном документе.

Остальные методы работы со встроенными объектами общие для текстовых и табличных документов, и зависят от типа `Frame`, в котором находятся:

1. Получение размеров

Размеры встроенного объекта могут быть получены из объектов `InlineFrame` или `AbsoluteFrame`, которые, в свою очередь, могут быть получены посредством использования методов `MediaObject.getFrame()`, `Image.getFrame()`, `Chart.getFrame()` (см раздел `Frame`).

```
var inlineFrame = frame.getInlineFrame();
if (inlineFrame != null) {
   Console.WriteLine(inlineFrame.getDimensions().width);
}
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
   Console.WriteLine(absoluteFrame.getDimensions().width);
}
```

2. Получение текущей позиции

С помощью методов `InlineFrame.getPosition()`, `AbsoluteFrame.getTopLeft()` можно получить текущую позицию объекта.

```
var inlineFrame = frame.getInlineFrame();
if (inlineFrame != null) {
    TextAnchoredPosition textAnchoredPosition = frame.getPosition();
```

```
    Console.WriteLine(textAnchoredPosition.horizontal.alignment);
}
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
    PointU topLeftPos = absoluteFrame.getTopLeft();
    Console.WriteLine(topLeftPos.x);
}
```

3. Установка размеров

С помощью методов `InlineFrame.setDimensions()`, `AbsoluteFrame.setDimensions()` можно изменить размеры встроенных объектов

```
var inlineFrame = frame.getInlineFrame();
SizeU frameDimensions = new SizeU(50, 50);
if (inlineFrame != null) {
    inlineFrame.setDimensions(frameDimensions);
}
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
    absoluteFrame.setDimensions(frameDimensions);
}
```

4. Установка позиции

Для объекта `AbsoluteFrame` используется метод `AbsoluteFrame.moveTo()`

```
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
   absoluteFrame.moveTo(new PointU(100, 100));
}
```

Для объекта `InlineFrame` используется метод `InlineFrame.setPosition()`. Примеры использования с различными параметрами приведены в разделе описании метода.

5. Масштабирование размеров

Для объекта `AbsoluteFrame` используется метод `AbsoluteFrame.scale()`

```
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
```

```
    absoluteFrame.scale(100, 100, ScaleFrom.TopLeft);
}
```

6. Установка обтекания текстом

Для `InlineFrame` вариант обтекания текстом графического объекта `TextWrapType` может быть задан посредством использованием метода `InlineFrame.setWrapType()`.

```
inlineFrame.setWrapType(TextWrapType.Inline);
```

4.4 Поиск в документе

Для поиска диапазона в текстовом или табличном документе необходимо создать экземпляр класса `Search` посредством вызова `createSearch(document)`, затем использовать метод `Search.findText` (см. Рисунок 1).

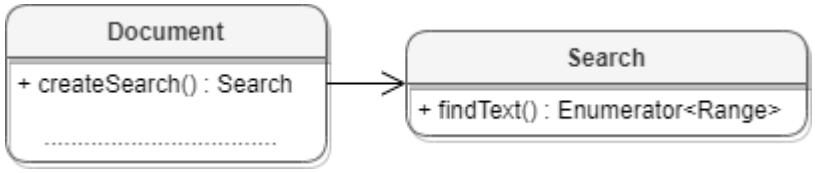


Рисунок 25 – Объектная модель для поиска в документе

Примеры поиска в документе

```
// Поиск по всему документу
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult = search.findText("API", CaseSensitive.Yes);
```

```
// Поиск в диапазоне блока
Range firstBlockRange = document.getBlocks().getBlock(0).getRange();
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult = search.findText("API", firstBlockRange,
CaseSensitive.No);
```

```
// Поиск в диапазоне ячеек
Table table = document.getBlocks().getTable(0);
CellRange cellRange = table.getCellRange("A1:B2");
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult = search.findText("API", cellRange,
CaseSensitive.Yes);
```

```
// Поиск в таблице
Table table = document.getBlocks().getTable(0);
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult = search.findText("API", table, CaseSensitive.Yes);
```

```
// Отображение результатов поиска
while (searchResult.MoveNext())
{
    var range = searchResult.Current;
    Console.WriteLine(range.extractText());
}
```

Для поиска ячеек в табличном документе используйте методы Table.find и CellRange.find.

Пример поиска ячеек в табличном документе

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

TableSearchSettings searchProps = new TableSearchSettings();
searchProps.caseSensitive = CaseSensitive.No;
searchProps.matchBehaviour = TableSearchSettings.MatchBehaviour.Glob;
searchProps.searchIn = TableSearchSettings.SearchProperty.Value;
searchProps.wholeWords = true;

CellsEnumerator results = sheet.find("*eye", searchProps);
foreach (Cell cell in results) {
    Console.WriteLine(cell.getFormattedValue()); // Steeleye Stout
}
```

Пример форматирования результатов поиска

```
Table sheet = document.getBlocks().getTable(0);
Search search = DocumentAPI.createSearch(document);
string template = "МойОфис";
var ranges = search.findText(template, sheet);
foreach (Range range in ranges) {
    TextProperties props = range.getTextProperties();
    props.bold = true;
```

```
    range.setTextProperties(props);
}
```

4.5 Работа с макрокомандами

Класс `Scripts` предоставляет доступ к списку макрокоманд документа. На рисунке 1 изображена объектная модель классов, относящихся к работе с макрокомандами.

Класс `Scripts` предназначен для доступа к списку макрокоманд, доступен через метод `Document.getScripts()`, класс `Scripting` служит для запуска макрокоманд, доступен через `DocumentAPI.createScripting(document)`.

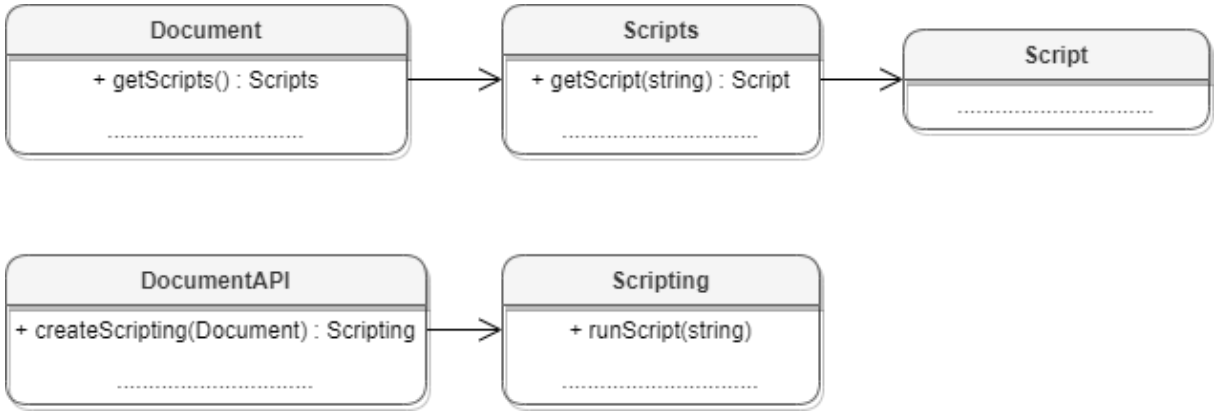


Рисунок 26 – Объектная модель таблиц для работы с макрокомандами

Доступны следующие операции:

- получение списка макрокоманд;
- добавление макрокоманды;
- получение макрокоманды по имени;
- удаление макрокоманды;
- запуск макрокоманды.

4.6 Работа с именованными диапазонами

Именованный диапазон – это диапазон ячеек таблицы, которому присвоено имя. Преимуществом именованного диапазона является его информативность: использование имени в текстовом формате выглядит более удобным, чем адреса ячеек. 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. Взаимодействие объектов, связанных с именованными диапазонами, описано на диаграмме (см. Рисунок 1).

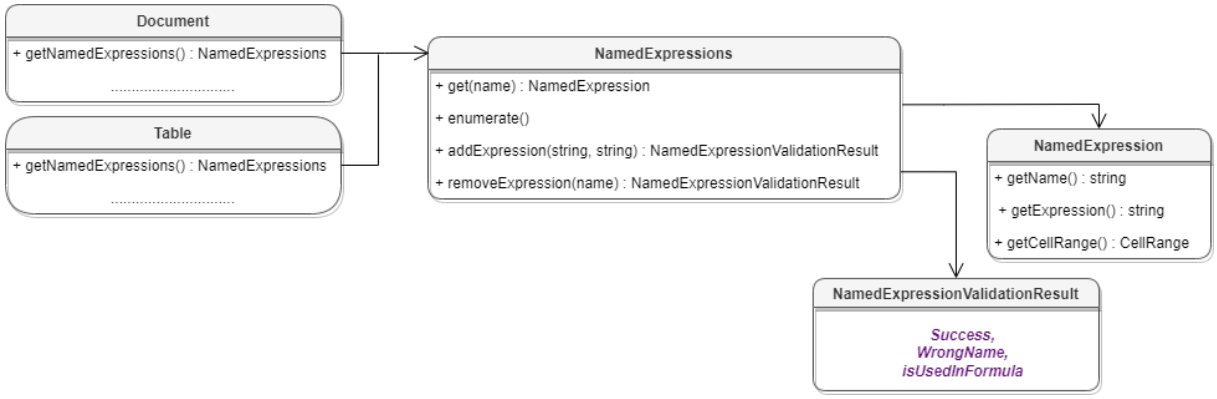


Рисунок 27 – Таблицы для работы с именованными диапазонами

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

4.6.1 Доступ к именованным диапазонам

Доступ к именованным диапазонам осуществляется посредством методов `Document.getNamedExpressions()` и `Table.getNamedExpressions()`.

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

Table table = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = table.getNamedExpressions();
```

4.6.2 Получение коллекции именованных диапазонов

Для перечисления именованных диапазонов используется объект `NamedExpressionsEnumerator`, который может быть получен с помощью метода `NamedExpressions.getEnumerator()`.

Примеры

```
// Коллекция именованных выражений
var table = document.getBlocks().getTable(0);
var namedExpressions = table.getNamedExpressions();
var enumerator = namedExpressions.getEnumerator();

// Перебор коллекции именованных выражений
```

```
foreach (var namedExpression in enumerator)
    // Использование полей структуры NamedExpression
    Console.WriteLine(namedExpression.getName());
    Console.WriteLine(namedExpression.getExpression());
    Console.WriteLine(namedExpression.getCellRange());
}
```

```
NamedExpressionsEnumerator enumerator = namedExpressions.getEnumerator();
foreach (var namedExpression in enumerator)
{
    // use namedExpression
}
```

4.6.3 Добавление именованного диапазона

Для добавления именованного диапазона используется метод NamedExpressions.addExpression().

```
String expressionName = "Покупки";
String expressionValue = "=Формула покупки!$E$6:$E$14";
namedExpressions.addExpression(expressionName, expressionValue);
```

4.6.4 Получение параметров именованного диапазона

Для получения детальной информации об именованном диапазоне используются методы NamedExpression.getName, NamedExpression.getExpression, NamedExpression.getCellRange.

```
Console.WriteLine(namedExpression.getName());
Console.WriteLine(namedExpression.getExpression());
CellRange cellRange = namedExpression.getCellRange();
Console.WriteLine(cellRange.getBeginColumn());
Console.WriteLine(cellRange.getLastColumn());
```

4.6.5 Переименование именованного диапазона

Для переименования именованного диапазона используется метод NamedExpression.setName().

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

4.6.6 Удаление именованного диапазона

Для удаления именованного диапазона используется метод NamedExpressions.removeExpression().

```
String expressionName = "Покупки";
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpression namedExpression = namedExpressions.get(expressionName);
namedExpressions.removeExpression(expressionName);
```

4.7 Работа со строками и столбцами таблиц

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

4.7.1 Группировка строк и колонок таблицы

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

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

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

4.7.2 Управление видимостью строк / колонок

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

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

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

Пример для текстового и табличного редактора

```
Table table = document.getBlocks().getTable(0);
Console.WriteLine(table.isRowVisible(3));
Console.WriteLine(table.isColumnVisible(1));
```

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

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

Пример для табличного редактора

```
uint beginRow = 1;
uint lastRow = 3;
uint beginColumn = 2;
uint lastColumn = 3;

bool visibility = false;

table.setRowsVisible(beginRow, lastRow - beginRow + 1, visibility);
table.setColumnsVisible(beginColumn, lastColumn - beginColumn + 1, visibility);
```

4.8 Работа с ячейками таблиц

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

4.8.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 1):

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

– из диапазона ячеек методом перечисления CellRange.getEnumerator().

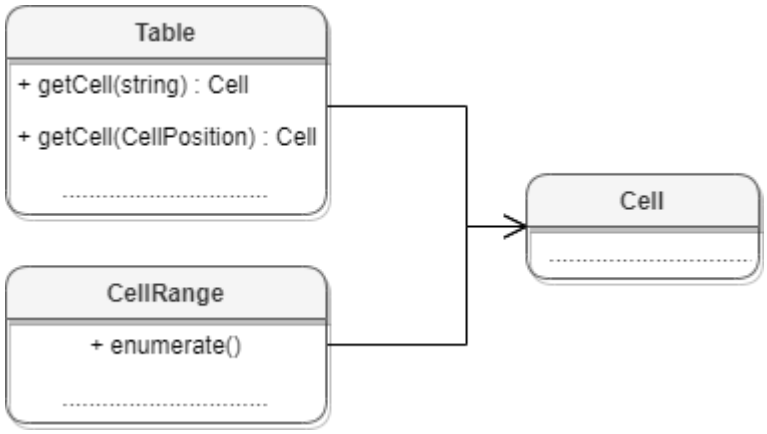


Рисунок 28 – Объектная модель для работы с ячейками таблиц

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

Пример

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
```

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

Пример

```
Table firstSheet = document.getBlocks().getTable("Лист1");
CellRange cellRange = firstSheet.getCellRange("B3:C4");
CellsEnumerator cellRangesEnumerator = cellRange.GetEnumerator();
foreach (var cell in cellRangesEnumerator)
{
    Console.WriteLine(cell.getFormattedValue());
}
```

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

Примеры

```
Table table1 = document.getBlocks().getTable(0);
Table table2 = document.getBlocks().getTable(1);

CellRange cellRange1 = table1.getCellRange("A1:C4");
```

```
CellRange cellRange2 = table2.getCellRange("A1:C4");

Cell cell1 = table1.getCell("A1");
Cell cell2 = table1.getCell("C4");
Cell cell3 = table1.getCell("E4");

Console.WriteLine(cellRange1.containsCell(cell1));
Console.WriteLine(cellRange1.containsCell(cell2));
Console.WriteLine(cellRange1.containsCell(cell3));

Console.WriteLine(cellRange2.containsCell(cell1));
Console.WriteLine(cellRange2.containsCell(cell2));
Console.WriteLine(cellRange2.containsCell(cell3));
```

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

Примеры

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
cell.setText("Текст");
Console.WriteLine(cell.getFormattedValue());

cell.setNumber(10);
Console.WriteLine(cell.getFormattedValue());

cell.setFormula("=SUM(B2:B3)");
Console.WriteLine(cell.getFormattedValue());

cell.setBool(false);
Console.WriteLine(cell.getFormattedValue());

cell.setFormattedValue("12:39");
Console.WriteLine(cell.getFormattedValue());
```

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

Пример

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
cell.setFormattedValue("22.07.2020");
Console.WriteLine(cell.getFormattedValue());

cell.setFormattedValue("12:39");
Console.WriteLine(cell.getFormattedValue());
```

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

Пример

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
cell.setFormat(CellFormat.Accounting);
cell.setNumber(12);
Console.WriteLine(cell.getFormattedValue());
```

Для получения значения ячейки используются методы `Cell.getFormattedValue()`, `Cell.getNumberValue()` и `Cell.getBoolValue()`.

Пример

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
Console.WriteLine(cell.getFormattedValue());
```

4.8.2 Форматирование ячеек

При работе с ячейками таблиц можно использовать следующие варианты форматирования:

- форматирование параметров ячейки `CellProperties`, например, цвет фона, угол поворота текста;
- форматирование абзаца ячейки, например, отступы абзаца, межстрочный интервал текста;
- форматирование текста, например, цвет текста, начертание;
- задание параметров границ ячеек.

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

Пример установки и получения свойств абзаца ячейки

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

ParagraphProperties paragraphProperties = cell.getParagraphProperties();
paragraphProperties.alignment = Alignment.Center;
cell.setParagraphProperties(paragraphProperties);
```

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

Пример настроек текста ячейки

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell(new CellPosition(0,1));

TextProperties textProperties = cell.getTextProperties();
textProperties.bold = true;
textProperties.italic = true;
textProperties.textColor = new Color(new ColorRGBA(55, 146, 179, 200));

cell.setTextProperties(textProperties);
```

4.8.3 Форматирование границ ячеек

Для оформления границ ячеек используется класс Borders (см. Рисунок 1). Он описывает свойства полей, соответствующих границам и диагоналям ячейки: `Left`, `Right`, `Top`, `Bottom`, `DiagonalDown`, `DiagonalUp`, `InnerHorizontal`, `InnerVertical`.

Каждая граница ячейки описывается классом LineProperties, который, в свою очередь, обладает свойствами LineStyle, LineEndingProperties, Color, LineWidth.

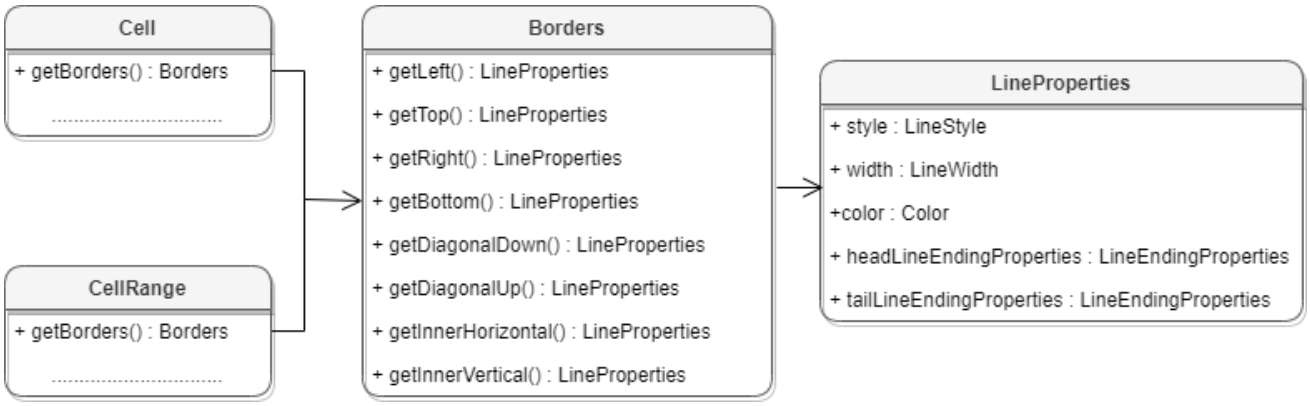


Рисунок 29 – Классы для работы с границами ячеек

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

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

Пример настройки границ ячеек

```
Table firstSheet = document.getBlocks().getTable("Лист1");
CellRange cellRange = firstSheet.getCellRange("F3:H7");

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Solid;
lineProperties.width = 1.5f;
lineProperties.color = new Color(new ColorRGBA(55, 146, 179, 200));

Borders borders = new Borders();
borders.setOuter(lineProperties);

cellRange.setBorders(borders);
```

4.8.4 Объединение и разделение ячеек таблицы

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

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

Пример

```
// Объединение ячеек A1 и A2 на первом листе табличного документа
Table firstSheet = document.getBlocks().getTable(0);
firstSheet.getCellRange("A1:A2").merge();
```

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

Пример

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

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 используется метод `Cell.unmerge()`.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
// Ячейка A1 является результатом объединения диапазона A1:A2
firstSheet.getCell("A1").unmerge();
```

4.9 Защита документов

Данный раздел содержит способы защиты содержимого текстовых и табличных документов:

- Защита диапазона текстового документа
- Защита листа табличного документа
- Защита структуры табличного документа

4.9.1 Защита диапазона текстового документа

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

Используйте метод `Range.lockContent()` для установки защиты:

```
// Защита первого абзаца документа:
Range textRange =
document.getRange().getBegin().getCurrentRange(TextUnit.Paragraph);
textRange.lockContent();
```

При попытке редактирования защищенного фрагмента возникает исключение `DocumentModificationError`:

```
Position pos = textRange.getContentEnd().getPreviousPosition(10);
pos.insertText("My Text"); // DocumentModificationError
```

Чтобы определить, защищен ли диапазон документа, используйте метод `Range.isContentLocked()`. Он возвращает `true`, если текущий диапазон защищен целиком или содержит защищенные фрагменты:

```
Console.WriteLine(textRange.isContentLocked()); // True
Console.WriteLine(document.getRange().isContentLocked()); // True
```

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

```
textRange.unlockContent();
// или
document.getRange().unlockContent();
```

Также вы можете снять защиту только с части диапазона:

```
// Снятие защиты с последнего предложения защищенного абзаца:
textRange.getContentEnd().getPreviousRange(TextUnit.Sentence).unlockContent();
```

4.9.2 Защита листа табличного документа

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

Для настройки параметров защиты ячеек используется объект `CellProtectionProperties`. Он позволяет настроить возможность редактирования ячеек и видимость формул на защищенном листе. Вызовите метод

Cell.setProtectionProperties() или CellRange.setProtectionProperties(), чтобы применить эти параметры к ячейке или диапазону. Параметры ячеек должны быть заданы до установки защиты.

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

```
Table sheet = document.getBlocks().getTable(0);
// Разрешает редактировать все ячейки листа:
CellRange cells = sheet.getCellRange("A1:Z300");
CellProtectionProperties cellProtection = new CellProtectionProperties();
cellProtection.lockedForChanges = false;
cellProtection.formulasNotDisplayed = false;
cells.setProtectionProperties(cellProtection);
// Задает ячейки, которые будут защищены от изменения:
CellRange protectedCells = sheet.getCellRange("A1:H8");
CellProtectionProperties cellProtection1 = new CellProtectionProperties();
cellProtection1.lockedForChanges = true;
cellProtection1.formulasNotDisplayed = false;
protectedCells.setProtectionProperties(cellProtection1);
// Запрещает вставку и удаление строк и столбцов:
TableProtectionProperties tableProtection = new TableProtectionProperties();
tableProtection.deleteColumns = false;
tableProtection.deleteRows = false;
tableProtection.filterData = true;
tableProtection.formatCells = true;
tableProtection.formatColumns = true;
tableProtection.formatRows = true;
tableProtection.insertAndEditObjects = true;
tableProtection.insertAndEditPivotTables = true;
tableProtection.insertColumns = false;
tableProtection.insertLinks = true;
tableProtection.insertRows = false;
tableProtection.selectProtectedCells = true;
tableProtection.sortData = true;
// Устанавливает защиту на лист и пароль для ее снятия:
sheet.setProtection(tableProtection, "password");
```

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

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

```
Cell cell = sheet.getCell("F6");
Console.WriteLine(cell.isProtected()); // True
```

Текущие настройки защиты ячеек возвращают методы Cell.getProtectionProperties() и CellRange.getProtectionProperties().

Статус защиты листа документа можно получить используя метод Table.isProtected():

```
Console.WriteLine(sheet.isProtected()); // True
```

Метод Table.getProtectionProperties() позволяет получить настройки защиты листа.

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

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

4.9.3 Защита структуры табличного документа

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

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

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

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

```
Table sheet = document.getBlocks().getTable(0);
sheet.remove(); // SpreadsheetProtectionError
```

Используйте метод Document.isStructureProtected() для получения состояния защиты структуры документа:

```
Console.WriteLine(document.isStructureProtected()); // True
```

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

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

4.10 Локализация документов

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

Пример получения даты для стандартной локализации

```
Document newDocument = application.createDocument(DocumentType.Workbook);
Table sheet = newDocument.getRange().getBegin().insertTable(10, 10, "Sheet1");
CellRange cellRange = sheet.getCellRange("A1");
cellRange.insertCurrentDateTime(DateTimeFormat.DateTime);
Cell cell = sheet.getCell("A1");
Console.WriteLine(cell.getFormattedValue()); // 05/23/2025 10:53 AM
```

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

Пример получения даты для русской локализации

```
DocumentSettings settings = new DocumentSettings();
settings.documentType = DocumentType.Workbook;
settings.localeInfo = new LocaleInfo();
settings.localeInfo.localeName = "ru_RU";

Document newDocument = application.createDocument(settings);
Table sheet = newDocument.getRange().getBegin().insertTable(10, 10, "Лист1");
CellRange cellRange = sheet.getCellRange("A1");
cellRange.insertCurrentDateTime(DateTimeFormat.DateTime);
```

```
Cell cell = sheet.getCell("A1");
Console.WriteLine(cell.getFormattedValue()); // 23.05.2025 10:53
```

5 ГЛОБАЛЬНЫЕ МЕТОДЫ

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

5.1 Метод DocumentAPI.createScripting

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

Пример

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

5.2 Метод DocumentAPI.createSearch

Метод инициализирует механизм поиска для текущего документа. Возвращает объект `Search`, с помощью которого выполняются поисковые запросы.

Пример

```
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult = search.findText("API");
```

5.3 Метод DocumentAPI.exportWorksheetToHtml

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

Вызов

```
public static HtmlFragments exportWorksheetToHtml(Table table, WorksheetHtmlExportSettings settings)
```

Параметры

– `table`: лист табличного документа, тип `Table`.

– `settings`: настройки генерации HTML документа, тип `WorksheetHtmlExportSettings`.

Возвращает

– HTML элементы, тип `HtmlFragments`.

Ограничения

– Структура возвращаемых элементов может быть изменена в будущих релизах.

– Поддерживается только генерация текста и стилей для ячеек, строк и столбцов.

Пример

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

WorksheetHtmlExportSettings settings = new WorksheetHtmlExportSettings();
settings.exportHeaders = false;
settings.exportMissingBorders = false;
HtmlFragments html = DocumentAPI.exportWorksheetToHtml(sheet, settings);

File.Create("style.css").Close();
File.WriteAllText("style.css", html.rootCss);

string body = html.body;
body = body.Insert(0, "<head>\r\n  <link rel=\"stylesheet\"
href=\"style.css\">\r\n</head>\r\n");

File.Create("doc.html").Close();
File.WriteAllText("doc.html", body);
```

5.4 Методы условного форматирования

Данный раздел содержит глобальные методы, которые могут использоваться для создания и приведения операторов условного форматирования.

5.4.1 Метод DocumentAPI.castToAboveAverageConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к AboveAverageConditionalFormatOperator.

Вызов

```
      public              static              AboveAverageConditionalFormatOperator
castToAboveAverageConditionalFormat(ConditionalFormatOperator
conditionalFormatOperator)
```

Параметры

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

Возвращает

– оператор для правил "Выше среднего" и "Ниже среднего", тип `AboveAverageConditionalFormatOperator`.

– `null`, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.AboveAverage)
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule.getOperator());
}
```

5.4.2 Метод DocumentAPI.castToBinaryConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к `BinaryConditionalFormatOperator`.

Вызов

```
public static BinaryConditionalFormatOperator
castToBinaryConditionalFormat(ConditionalFormatOperator
conditionalFormatOperator)
```

Параметры

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator`.

Возвращает

– оператор для правил "Между" и "Не между", тип `BinaryConditionalFormatOperator`.

– `null`, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.Binary)
        binaryOperator =
DocumentAPI.castToBinaryConditionalFormat(rule.getOperator());
}
```

5.4.3 Метод DocumentAPI.castToColorScaleConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к ColorScaleConditionalFormatOperator.

Вызов

```
public static ColorScaleConditionalFormatOperator
castToColorScaleConditionalFormat(ConditionalFormatOperator
conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правила "Цветовая шкала", тип ColorScaleConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.ColorScale)
        aboveOperator =
DocumentAPI.castToColorScaleConditionalFormat(rule.getOperator());
}
```

5.4.4 Метод DocumentAPI.castToDataBarConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к DataBarConditionalFormatOperator.

Вызов

```
public static DataBarConditionalFormatOperator castToDataBarConditionalFormat(ConditionalFormatOperator conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правила "Гистограмма", тип DataBarConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.DataBar)
        aboveOperator = DocumentAPI.castToDataBarConditionalFormat(rule.getOperator());
}
```

5.4.5 Метод DocumentAPI.castToIconSetConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к IconSetConditionalFormatOperator.

Вызов

```
public static IconSetConditionalFormatOperator castToIconSetConditionalFormat(ConditionalFormatOperator conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правила "Значки", тип IconSetConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.IconSet)
        aboveOperator =
DocumentAPI.castToIconSetConditionalFormat(rule.getOperator());
}
```

5.4.6 Метод DocumentAPI.castToNullaryConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к NullaryConditionalFormatOperator.

Вызов

```
public static NullaryConditionalFormatOperator
castToNullaryConditionalFormat(ConditionalFormatOperator
conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правил без параметров, тип NullaryConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.Nullary)
        aboveOperator =
DocumentAPI.castToNullaryConditionalFormat(rule.getOperator());
}
```

5.4.7 Метод DocumentAPI.castToTextConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к TextConditionalFormatOperator.

Вызов

```
public static TextConditionalFormatOperator castToTextConditionalFormat(ConditionalFormatOperator conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правила "Текст", тип TextConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.Text)
        aboveOperator = DocumentAPI.castToTextConditionalFormat(rule.getOperator());
}
```

5.4.8 Метод DocumentAPI.castToTopBottomConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к TopBottomConditionalFormatOperator.

Вызов

```
public static TopBottomConditionalFormatOperator castToTopBottomConditionalFormat(ConditionalFormatOperator conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правила "Наибольшие и наименьшие значения", тип TopBottomConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.TopBottom)
        aboveOperator =
DocumentAPI.castToTopBottomConditionalFormat(rule.getOperator());
}
```

5.4.9 Метод DocumentAPI.castToUnaryConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к UnaryConditionalFormatOperator.

Вызов

```
public static UnaryConditionalFormatOperator
castToUnaryConditionalFormat(ConditionalFormatOperator conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правил "Больше", "Меньше", "Равно" и "Не равно", тип UnaryConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.Unary)
        aboveOperator =
DocumentAPI.castToUnaryConditionalFormat(rule.getOperator());
}
```

5.4.10 Метод DocumentAPI.castToUniquenessConditionalFormat

Метод позволяет привести базовый оператор условного форматирования к UniquenessConditionalFormatOperator.

Вызов

```
public static UniquenessConditionalFormatOperator castToUniquenessConditionalFormat(ConditionalFormatOperator conditionalFormatOperator)
```

Параметры

– conditionalFormatOperator: оператор условного форматирования, тип ConditionalFormatOperator.

Возвращает

– оператор для правила "Уникальные и повторяющиеся значения", тип UniquenessConditionalFormatOperator.

– null, если приведение невозможно.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.Uniqueness)
        aboveOperator = DocumentAPI.castToUniquenessConditionalFormat(rule.getOperator());
}
```

5.4.11 Метод DocumentAPI.createAboveAverageConditionalFormatOperator

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

Вызов

```
public static ConditionalFormatOperator createAboveAverageConditionalFormatOperator(ConditionalFormatAboveAverageCondition condition, int? stdDev)
```

Параметры

– `condition`: условие применения форматирования, тип `ConditionalFormatAboveAverageCondition`.

– `stdDev`: (необязательный) количество стандартных отклонений, тип `int`.

Возвращает

– оператор условного форматирования, тип `ConditionalFormatOperator`.

5.4.12 Метод DocumentAPI.createBinaryConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правил "Между" и "Не между". Пример использования смотри в разделе `BinaryConditionalFormatOperator`.

Вызов

```
public static ConditionalFormatOperator createBinaryConditionalFormatOperator(ConditionalFormatBinaryCondition condition, string firstArgument, string secondArgument)
```

Параметры

– `condition`: условие применения форматирования, тип `ConditionalFormatBinaryCondition`.

– `firstArgument`: первый аргумент, тип `string`.

– `secondArgument`: второй аргумент, тип `string`.

Возвращает

– оператор условного форматирования, тип `ConditionalFormatOperator`.

5.4.13 Метод DocumentAPI.createColorScaleConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правила "Цветовая шкала". Пример использования смотри в разделе `ColorScaleConditionalFormatOperator`.

Вызов

```
public static ConditionalFormatOperator createColorScaleConditionalFormatOperator(ConditionalFormatColorScaleEntries entries)
```

Параметры

– `entries`: набор правил применения цветов, тип `ConditionalFormatColorScaleEntries`.

Возвращает

– оператор условного форматирования, тип `ConditionalFormatOperator`.

5.4.14 Метод DocumentAPI.createDataBarConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правила "Гистограмма". Пример использования смотри в разделе `DataBarConditionalFormatOperator`.

Вызов

```
public static ConditionalFormatOperator createDataBarConditionalFormatOperator(ConditionalFormatDataBarParams params_)
```

Параметры

– `params_`: настройки гистограммы, тип `ConditionalFormatDataBarParams`.

Возвращает

– оператор условного форматирования, тип `ConditionalFormatOperator`.

5.4.15 Метод DocumentAPI.createIconSetConditionalFormatOperator

Метод создает оператор, который содержит настройки применения форматирования для правила "Значки". Пример использования смотри в разделе `IconSetConditionalFormatOperator`.

Вызов

```
public static ConditionalFormatOperator createIconSetConditionalFormatOperator(ConditionalFormatIconSetEntries entries, bool isValueShown)
```

Параметры

– `entries`: набор правил отображения значков, тип `ConditionalFormatIconSetEntries`.

– `isValueShown`: `true`, чтобы показывать значение ячейки при примененном форматировании, в ином случае – `false`.

Возвращает

– оператор условного форматирования, тип ConditionalFormatOperator.

5.4.16 Метод DocumentAPI.createNullaryConditionalFormatOperator

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

Вызов

```
public static ConditionalFormatOperator createNullaryConditionalFormatOperator(ConditionalFormatNullaryCondition condition)
```

Параметры

– condition: условие применения форматирования, тип ConditionalFormatNullaryCondition.

Возвращает

– оператор условного форматирования, тип ConditionalFormatOperator.

5.4.17 Метод DocumentAPI.createTextConditionalFormatOperator

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

Вызов

```
public static ConditionalFormatOperator createTextConditionalFormatOperator(ConditionalFormatTextCondition condition, string argument)
```

Параметры

– condition: условие применения форматирования, тип ConditionalFormatTextCondition.

– argument: аргумент условия, тип string.

Возвращает

– оператор условного форматирования, тип ConditionalFormatOperator.

5.4.18 Метод DocumentAPI.createTopBottomConditionalFormatOperator

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

Вызов

```
public static ConditionalFormatOperator createTopBottomConditionalFormatOperator(ConditionalFormatTopBottomCondition condition, uint value, bool usePercent)
```

Параметры

– `condition`: условие применения форматирования, тип ConditionalFormatTopBottomCondition.

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

– `usePercent`: `true`, чтобы выделять определенный процент значений; чтобы выделять определенное количество значений – `false`.

Возвращает

– оператор условного форматирования, тип ConditionalFormatOperator.

5.4.19 Метод DocumentAPI.createUnaryConditionalFormatOperator

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

Вызов

```
public static ConditionalFormatOperator createUnaryConditionalFormatOperator(ConditionalFormatUnaryCondition condition, string argument)
```

Параметры

– `condition`: условие применения форматирования, тип ConditionalFormatUnaryCondition.

– `argument`: аргумент условия, тип `string`.

Возвращает

– оператор условного форматирования, тип ConditionalFormatOperator.

5.4.20 Метод DocumentAPI.createUniquenessConditionalFormatOperator

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

Вызов

```
public static ConditionalFormatOperator createUniquenessConditionalFormatOperator(ConditionalFormatUniquenessCondition condition)
```

Параметры

– condition: условие применения форматирования, тип ConditionalFormatUniquenessCondition.

Возвращает

– оператор условного форматирования, тип ConditionalFormatOperator.

6 СПРАВОЧНИК КЛАССОВ

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

6.1 Класс AboveAverageConditionalFormatOperator

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

Конструкторы

```
public AboveAverageConditionalFormatOperator(ConditionalFormatAboveAverageCondition condition)
```

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

Пример

Количество
2
5
10
1
3
4
6
15
20
2
7

Рисунок 30 – Пример создания правила "Выше среднего"

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

CellRange cellRange = sheet.getCellRange("F2:F12");
var cellRangePosition = cellRange.getTableRange();
```

```
var aboveStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(0, 255, 0, 150)));
aboveStyle.cellProperties = cellProperties;

var aboveOperator =
DocumentAPI.createAboveAverageConditionalFormatOperator(ConditionalFormatAboveAve
rageCondition.Above);

var aboveRule = new ConditionalFormatRule(aboveOperator, aboveStyle,
cellRangePosition, false);
rules.addRule(aboveRule);
```

6.1.1 Метод AboveAverageConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
public ConditionalFormatAboveAverageCondition getCondition()
```

Возвращает

– условие применения форматирования, тип ConditionalFormatAboveAverageCondition.

6.1.2 Метод AboveAverageConditionalFormatOperator.getStdDev

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

Вызов

```
public int? getStdDev()
```

Возвращает

– количество стандартных отклонений, тип `int`.

– `null`, если отклонения не заданы.

6.1.3 Метод AboveAverageConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.2 Класс AbsoluteFrame

Класс AbsoluteFrame описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 1). Предназначен для получения и изменения свойств позиции медиаобъектов. Используется в табличном документе. Может быть получен с помощью метода Frame.getAbsoluteFrame().

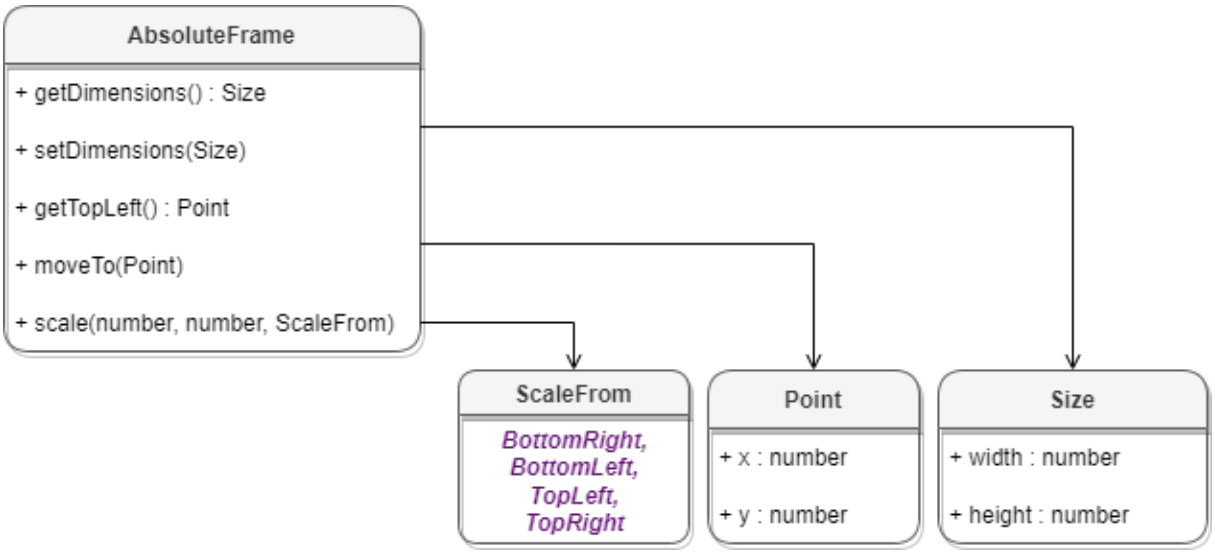


Рисунок 31 – Объектная модель класса AbsoluteFrame

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
Images images = table.getImages();
ImagesEnumerator imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    var frame = image.getFrame();
    var absoluteFrame = frame.getAbsoluteFrame();
    if (absoluteFrame != null) {
       Console.WriteLine(absoluteFrame.getDimensions().width);
```

```
    }
}
```

6.2.1 Метод AbsoluteFrame.getDimensions

Метод возвращает размеры встроенного объекта.

Вызов

```
public SizeU getDimensions()
```

Возвращает

– размер объекта, тип SizeU.

Пример

```
var frame = mediaObject.getFrame();
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
    Console.WriteLine(absoluteFrame.getDimensions().width);
}
```

6.2.2 Метод AbsoluteFrame.getTopLeft

Метод возвращает позицию верхней левой точки объекта.

Вызов

```
public PointU getTopLeft()
```

Возвращает

– координаты верхней левой точки объекта, тип PointU.

– null, если координаты невозможно определить.

Пример

```
var frame = mediaObject.getFrame();
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
   PointU topLeftPos = absoluteFrame.getTopLeft();
}
```

6.2.3 Метод AbsoluteFrame.moveTo

Метод изменяет позицию встроенного объекта.

Вызов

```
public void moveTo(PointU position)
```

Параметры

– `position`: новые координаты верхней левой точки объекта, тип `PointU`.

Пример

```
var frame = mediaObject.getFrame();
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
   absoluteFrame.moveTo(new PointU(100, 100));
}
```

6.2.4 Метод AbsoluteFrame.scale

Метод изменяет размер объекта.

Вызов

```
public void scale(float widthScale, float heightScale, ScaleFrom scaleFrom)
```

Параметры

– `widthScale`: горизонтальный коэффициент масштабирования, тип `float`.

– `heightScale`: вертикальный коэффициент масштабирования, тип `float`.

– `scaleFrom`: точка объекта, относительно которой производится масштабирование, тип `ScaleFrom`.

Пример

```
var frame = mediaObject.getFrame();
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
    absoluteFrame.scale(1.5, 1.5, ScaleFrom.TopLeft);
}
```

6.2.5 Метод AbsoluteFrame.setDimensions

Метод позволяет задать размер встроенного объекта.

Вызов

```
public void setDimensions(SizeU size)
```

Параметры

– `size`: размер объекта, тип `SizeU`.

Пример

```
var frame = mediaObject.getFrame();
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
    absoluteFrame.setDimensions(new SizeU(50, 50));
    Console.WriteLine(absoluteFrame.getDimensions().width);
}
```

6.3 Класс AccountingCellFormatting

Класс содержит параметры финансового формата ячеек таблицы и используется в качестве аргумента метода `Cell.setFormat()`.

Таблица 4 – Описание полей класса `AccountingCellFormatting`

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

Поле	Тип	Описание
AccountingCellFormatting.currencySignPlacement	CurrencySignPlacement	Тип размещения знака валюты

Пример

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

AccountingCellFormatting cellFormat = new AccountingCellFormatting();
cellFormat.decimalPlaces = 2;
cellFormat.symbol = "Руб";

cell.setFormat(cellFormat);
Console.WriteLine(cell.getFormattedValue());
```

6.4 Перечисление Alignment

Перечисление `Alignment` содержит варианты горизонтального выравнивания текста, в том числе в ячейке таблицы. Используется в поле `ParagraphProperties.alignment`.

Таблица 5 – Варианты горизонтального выравнивания текста

Значение	Описание
Alignment.Default	Выравнивание текста по умолчанию
Alignment.Left	Выравнивание текста по левому краю
Alignment.Center	Выравнивание текста по центру
Alignment.Right	Выравнивание по правому краю
Alignment.Justify	Выравнивание по ширине
Alignment.Distributed	Распределенное выравнивание, при применении которого между словами добавляются пробелы так, чтобы оба края каждой строки были выровнены по обеим сторонам. Последняя строка в абзаце также выравнивается по обеим сторонам, но если строка состоит из одного слова, то выравнивание по правой стороне не осуществляется
Alignment.Fill	Распределение текста по горизонтали – заполнение строки текстом

Пример

```
ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
paragraphProperties.alignment = Alignment.Center;
```

6.5 Класс Application

Класс `Application` управляет параметрами и объектами приложения. Предоставляет интерфейс для создания и загрузки документов. Допустимо использование только одного объекта `Application` для всего сеанса обработки документа.

Пример

```
Application application = new Application();
```

6.5.1 Метод Application.createDocument

Метод создает новый документ. Примеры использования приведены в разделах Создание и открытие текстового документа и Создание и открытие табличного документа.

Вызов

```
public Document createDocument(DocumentType documentType)
public Document createDocument(DocumentSettings settings)
```

Параметры

– `documentType`: тип нового документа, тип `DocumentType`.

– `settings`: настройки нового документа, тип `DocumentSettings`.

Возвращает

– новый документ, тип `Document`.

6.5.2 Метод Application.getMessenger

Метод возвращает объект, реализующий логирование событий.

Вызов

```
public Messenger getMessenger()
```

Возвращает

– объект для логирования событий, тип `Messenger`.

Пример

```
Application application = new Application();
MessageHandler handler = new MessageHandler();
Messenger messenger = application.getMessenger();
Connection connection = messenger.subscribe(handler);
```

6.5.3 Метод Application.loadDocument

Метод загружает существующий текстовый или табличный документ из файла, находящегося по указанному пути. Формат и тип документа определяются из расширения файла, если не указаны явно с помощью параметра `settings`.

Примеры использования приведены в разделах Создание и открытие текстового документа и Создание и открытие табличного документа.

Вызов

```
public Document loadDocument(string filePath, LoadDocumentSettings settings)
```

Параметры

- `filePath`: путь до файла, который включает его название и расширение, тип `string`.
- `settings`: (необязательный) настройки загрузки документов, тип `LoadDocumentSettings`.

Возвращает

- загруженный документ, тип `Document`.

6.6 Класс BinaryConditionalFormatOperator

Класс `BinaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил "Между" и "Не между". Используется при создании правила условного форматирования (`ConditionalFormatRule`). В качестве аргументов могут использоваться как значения, так и формулы.

Конструктор

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

Пример

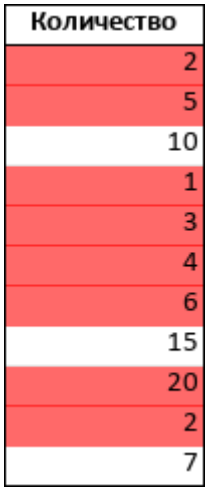


Рисунок 32 – Пример создания правила "Не между"

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

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

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

var binaryOperator =
DocumentAPI.createBinaryConditionalFormatOperator(ConditionalFormatBinaryConditio
n.NotBetween, "=$F$12", "=$F$9");

var binaryRule = new ConditionalFormatRule(binaryOperator, binaryStyle,
cellRangePosition, false);
rules.addRule(binaryRule);
```

6.6.1 Метод BinaryConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
public ConditionalFormatBinaryCondition getCondition()
```

Возвращает

– условие применения форматирования, тип ConditionalFormatBinaryCondition.

6.6.2 Метод BinaryConditionalFormatOperator.getFirstArgument

Метод возвращает первый аргумент условия.

Вызов

```
public string getFirstArgument()
```

Возвращает

– первый аргумент, тип string.

6.6.3 Метод BinaryConditionalFormatOperator.getSecondArgument

Метод возвращает второй аргумент условия.

Вызов

```
public string getSecondArgument()
```

Возвращает

– второй аргумент, тип string.

6.6.4 Метод BinaryConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.7 Класс Block

Класс Block является базовым для всех блоков документа. От него наследуются классы Paragraph, Table, Shape, Field (см. Рисунок 1).

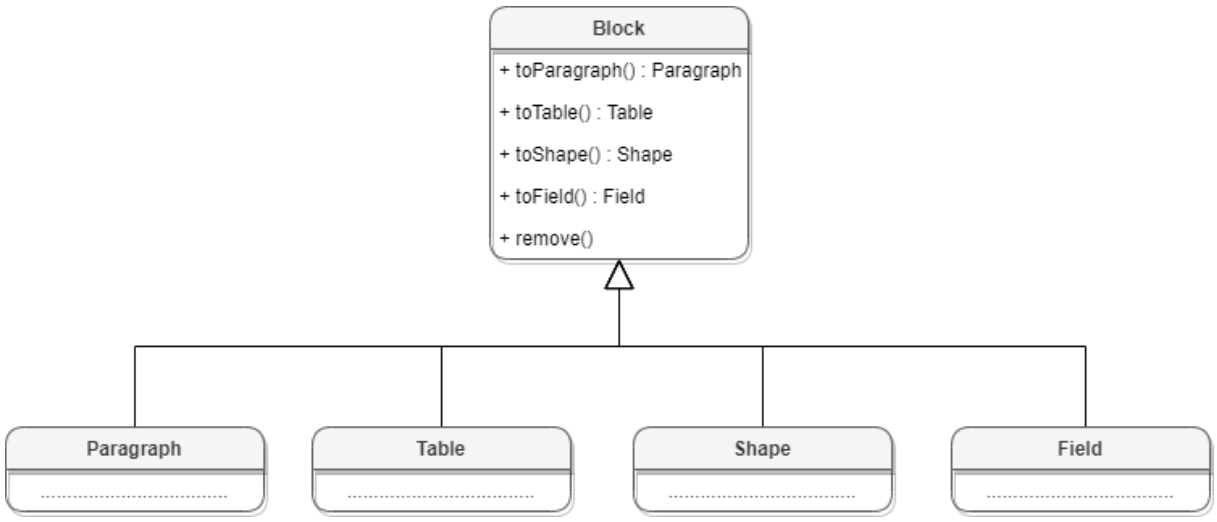


Рисунок 33 – Объектная модель класса `Block`

6.7.1 Метод Block.getRange

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

Пример

```
Blocks blocks = document.getBlocks();
Block block = blocks.getBlock(0);
if (block != null) {
    Console.WriteLine(block.getRange().extractText());
}
```

6.7.2 Метод Block.getSection

Метод возвращает раздел `Section`, содержащий блок.

Пример

```
Blocks blocks = document.getBlocks();
Block block = blocks.getBlock(0);
if (block != null) {
    Section section = block.getSection();
    Console.WriteLine(section.getRange().extractText());
}
```

6.7.3 Метод Block.remove

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

Пример

```
Blocks blocks = document.getBlocks();
Block block = blocks.getBlock(0);
if (block != null) {
    block.remove();
}
```

6.7.4 Методы toParagraph, toTable, toShape, toField

Преобразует объект `Block` в объект соответствующего типа.

Пример

```
Blocks blocks = document.getBlocks();
Block block = blocks.getBlock(0);
if (block != null) {
    Paragraph paragraph = block.toParagraph();
    Console.WriteLine(paragraph.getRange().extractText());
}
```

6.8 Класс Blocks

Класс `Blocks` обеспечивает доступ к блокам `Block` документа или диапазона документа (см. Рисунок 1). Объект класса `Blocks` может быть получен вызовом метода `Document.getBlocks` или `HeaderFooter.getBlocks`.

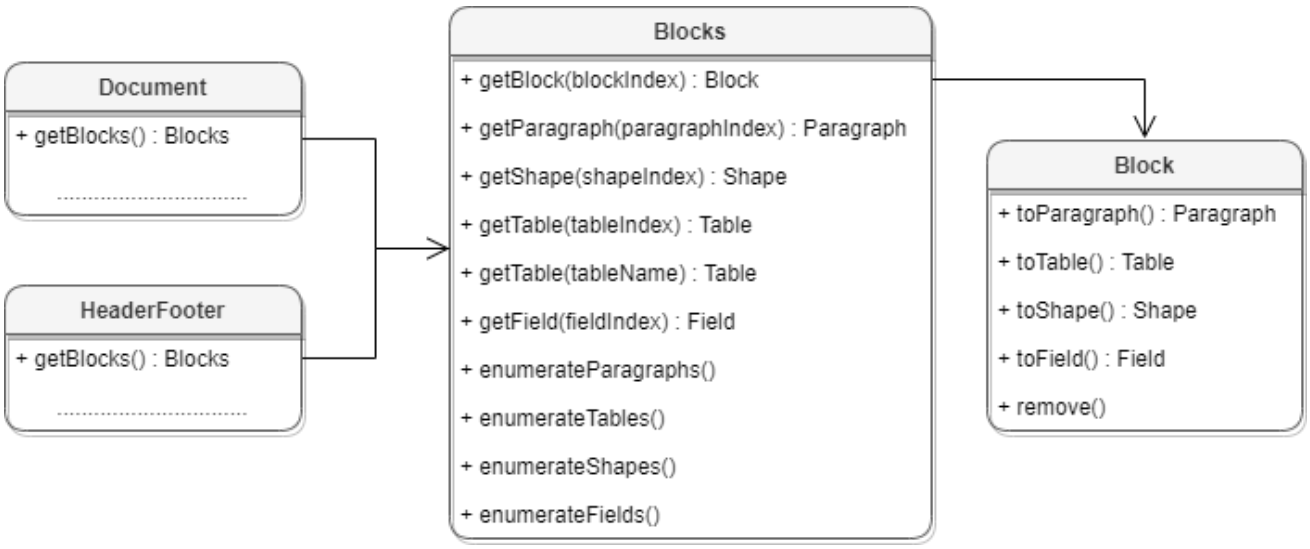


Рисунок 34 – Объектная модель класса `Blocks`

6.8.1 Метод Blocks.getBlock

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

Пример

```
Block block = document.getBlocks().getBlock(0);
if (block != null)
{
    Console.WriteLine(block.getRange().extractText());
}
else
{
    Console.WriteLine("No blocks found");
}
```

6.8.2 Метод Blocks.GetEnumerator

Позволяет реализовать перечисление объектов `Block`.

Пример

```
BlocksEnumerator blocksEnumerator = document.getBlocks().GetEnumerator();
foreach (var block in blocksEnumerator)
```

```
{
    Console.WriteLine(block.getRange().extractText());
}
```

6.8.3 Метод Blocks.getField

Возвращает объект типа `Field` по заданному индексу.

Пример

```
Field field = document.getBlocks().getField(0);
if (field != null) {
    Console.WriteLine(field.getRange().extractText());
} else {
    Console.WriteLine("No fields found");
}
```

6.8.4 Метод Blocks.getFieldsEnumerator

Позволяет перечислить объекты типа `Field`.

Пример

```
FieldsEnumerator fieldsEnumerator = document.getBlocks().getFieldsEnumerator();
foreach (var field in fieldsEnumerator)
{
    Console.WriteLine(field.getRange().extractText());
}
```

6.8.5 Метод Blocks.getParagraph

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

Пример

```
Paragraph paragraph = document.getBlocks().getParagraph(0);
if (paragraph != null) {
    Console.WriteLine(paragraph.getRange().extractText());
} else {
    Console.WriteLine("No paragraphs found");
}
```

6.8.6 Метод Blocks.getParagraphsEnumerator

Позволяет реализовать перечисление абзацев Paragraph.

Пример

```
ParagraphsEnumerator paragraphsEnumerator =
document.getBlocks().getParagraphsEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    Console.WriteLine(paragraph.getRange().extractText());
}
```

6.8.7 Метод Blocks.getShape

Возвращает фигуру Shape по заданному индексу.

Пример

```
Shape shape = document.getBlocks().getShape(0);
if (shape != null) {
    Console.WriteLine(shape.getRange().extractText());
} else {
    Console.WriteLine("No shapes found");
}
```

6.8.8 Метод Blocks.getShapesEnumerator

Позволяет перечислить объекты типа Shape.

Пример

```
ShapesEnumerator shapesEnumerator = document.getBlocks().getShapesEnumerator();
foreach (var shape in shapesEnumerator)
{
    Console.WriteLine(shape.getRange().extractText());
}
```

6.8.9 Метод Blocks.getTable

Для табличного документа метод возвращает страницу документа, для текстового документа - таблицу. В качестве параметра используется индекс или имя таблицы. При использовании индекса нумерация таблиц начинается с нуля.

Варианты реализации метода:

```
Table getTable(int tableIndex);
Table getTable(String tableName);
```

Примеры

```
Table table = document.getBlocks().getTable(0);
```

```
Table table = document.getBlocks().getTable("Лист1");
```

Примеры использования метода `getTable()` также приведены в разделах Работа с таблицами текстового документа и Работа с листами табличного документа.

6.8.10 Метод Blocks.getTablesEnumerator

Позволяет перечислить объекты типа `Table`.

Пример

```
TablesEnumerator tablesEnumerator = document.getBlocks().getTablesEnumerator();
foreach (var table in tablesEnumerator)
{
    Console.WriteLine(table.getRange().extractText());
}
```

6.9 Класс Bookmarks

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

6.9.1 Метод Bookmarks.getBookmarkRange

Возвращает экземпляр объекта `Range` для дальнейшей работы с содержимым закладки.

Пример

```
Bookmarks bookmarks = document.getBookmarks();
Range bookmarkRange = bookmarks.getBookmarkRange("Bookmark");
if (bookmarkRange != null) {
    bookmarkRange.replaceText("New bookmark text");
    Console.WriteLine(bookmarkRange.extractText());
}
```

6.9.2 Метод Bookmarks.removeBookmark

Удаляет закладку по ее названию.

Пример

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

6.10 Класс Borders

Класс `Borders` предназначен для оформления границ ячейки таблицы. Список методов приведен в таблице 6. Методы установки границ возвращают объект `Borders` и используют в качестве параметра объект `LineProperties`, который содержит такие настройки линии, как тип, толщина, цвет.

Таблица 6 – Описание методов класса `Borders`

Метод	Описание
`Borders setLeft(LineProperties lp)`	Установка левой границы ячейки
`Borders setRight(LineProperties lp)`	Установка правой границы ячейки
`Borders setTop(LineProperties lp)`	Установка верхней границы ячейки
`Borders setBottom(LineProperties lp)`	Установка нижней границы ячейки
`Borders setDiagonalDown(LineProperties lp)`	Установка диагональной линии 
`Borders setDiagonalUp(LineProperties lp)`	Установка диагональной линии 
`Borders setOuter(LineProperties lp)`	Установка внешних границ ячейки

Метод	Описание
`Borders setDiagonals(LineProperties lp)`	Установка обеих типов диагональных линий одновременно
`Borders setInnerHorizontal(LineProperties lp)`	Установка внутренних горизонтальных границ ячейки
`Borders setInnerVertical(LineProperties lp)`	Установка внутренних вертикальных границ ячейки
`Borders setInner(LineProperties lp)`	Установка внутренних границ ячейки
`Borders setAll(LineProperties lp)`	Установка всех границ ячейки
`LineProperties getLeft()`	Получение левой границы ячейки
`LineProperties getRight()`	Получение правой границы ячейки
`LineProperties getTop()`	Получение верхней границы ячейки
`LineProperties getBottom()`	Получение нижней границы ячейки
`LineProperties getDiagonalDown()`	Получение диагональной линии
`LineProperties getDiagonalUp()`	Получение диагональной линии
`LineProperties getInnerHorizontal()`	Получение внутренних горизонтальных границ ячейки
`LineProperties getInnerVertical()`	Получение внутренних вертикальных границ ячейки
`bool isEmpty()`	Возвращает `true`, если не установлена ни одна граница ячейки

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("C3");

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Solid;
lineProperties.width = 1.5f;
lineProperties.color = new Color(new ColorRGBA(55, 146, 179, 200));

Borders borders = new Borders();
borders.setLeft(lineProperties).setRight(lineProperties);
borders.setTop(lineProperties).setBottom(lineProperties);
cell.setBorders(borders);
```

6.11 Перечисление CalculationMode

Режимы пересчета формул в документе представлены в таблице 7. Перечисление `CalculationMode` используется в методах `Document.getCalculationMode()` и `Document.setCalculationMode()`.

Таблица 7 – Режимы пересчета формул

Значение	Описание
`CalculationMode.Auto`	Формулы пересчитываются автоматически при изменении данных.
`CalculationMode.Manual`	Формулы пересчитываются вручную.

6.12 Перечисление CaseSensitive

Параметры настройки учета регистра при поиске (см. метод `Search.FindText()` и поле `TableSearchSettings.caseSensitive`) представлены в таблице 8.

Таблица 8 – Параметры регистра при поиске

Значение	Описание
`CaseSensitive.Yes`	Поиск с учетом регистра
`CaseSensitive.No`	Поиск без учета регистра

6.13 Класс Cell

Класс `Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 1).

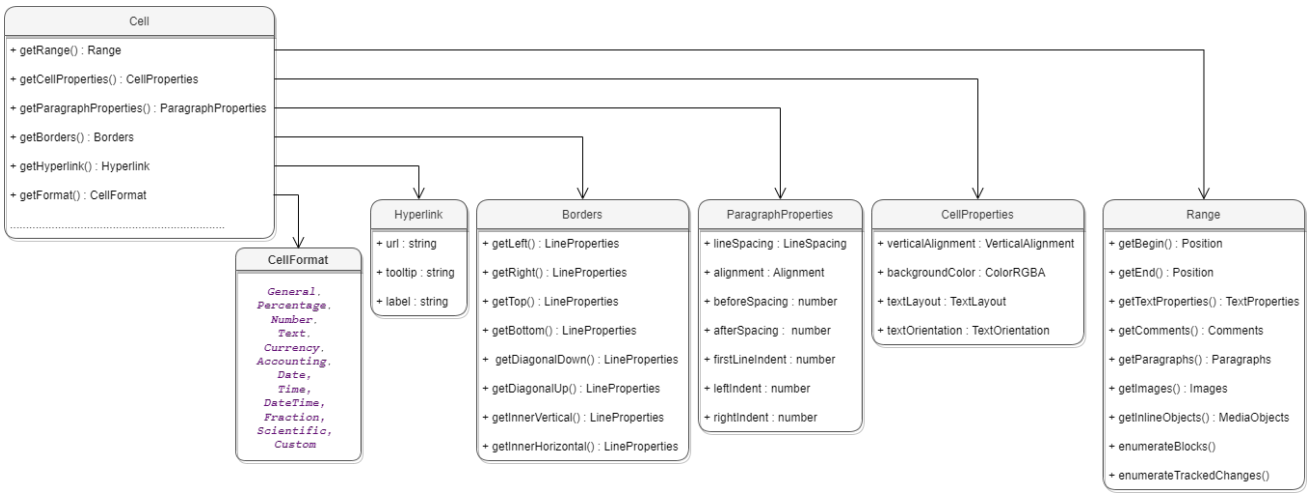


Рисунок 35 – Объектная модель ячейки таблиц

6.13.1 Метод Cell.calculate

Метод пересчитывает формулу в текущей ячейке.

Вызов

```
public void calculate()
```

6.13.2 Метод Cell.checkDataValidation

Метод проверяет значение ячейки согласно заданной в ней проверке данных.

Вызов

```
public DataValidationResult checkDataValidation()
```

Возвращает

– результат проверки значения ячейки, тип `DataValidationResult`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("C10");
DataValidationResult result = cell.checkDataValidation();
if (!result.isValid())
    Console.WriteLine(result.getDataValidation().getErrorMessage());
```

6.13.3 Метод Cell.getBoolValue

Метод возвращает логическое значение ячейки.

Вызов

```
public bool? getBoolValue()
```

Возвращает

– логическое значение ячейки, тип `bool`.

– `null`, если ячейка не содержит логического значения.

Пример

```
Table sheet = document.getBlocks().getTable(0);
sheet.getCell("D2").setNumber(1);
sheet.getCell("E2").setNumber(100);
```

```
sheet.getCell("C2").setFormula("=D2*100=E2");


sheet.getCell("C2").getBoolValue(); // True
```

6.13.4 Метод Cell.getBorders

Позволяет получить границы ячейки.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");
Borders borders = cell.getBorders();
Console.WriteLine(borders.getLeft());
```

6.13.5 Метод Cell.getCellProperties

Позволяет получить свойства CellProperties ячейки.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
CellProperties cellProperties = cell.getCellProperties();
Console.WriteLine(cellProperties.verticalAlignment);
```

6.13.6 Метод Cell.getColumnIndex

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

Вызов

```
public uint getColumnIndex()
```

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("A3");
Console.WriteLine(cell.getColumnIndex()); // 0
Console.WriteLine(cell.getRowIndex()); // 2
```

6.13.7 Метод Cell.getCurrentRegion

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

Вызов

```
public CellRange getCurrentRegion()
```

Возвращает

– диапазон ячеек, тип CellRange.

Пример

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

6.13.8 Метод Cell.getCustomFormat

Возвращает строку формата ячейки.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");
Console.WriteLine(cell.getCustomFormat());
```

6.13.9 Метод Cell.getDataValidation

Метод возвращает настройки проверки данных для текущей ячейки.

Вызов

```
public DataValidation getDataValidation()
```

Возвращает

– настройки проверки данных, тип DataValidation.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("E4");
DataValidation cellDV = cell.getDataValidation();
Console.WriteLine(cellDV.getType());
Console.WriteLine(cellDV.getFormula1());
```

6.13.10 Метод Cell.getFormat

Метод возвращает формат ячейки. Список поддерживаемых форматов ячеек приведен в разделе CellFormat.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");
NumberCellFormatting cellFormatting = new NumberCellFormatting();
cellFormatting.decimalPlaces = 2;
cell.setFormat(cellFormatting);
Console.WriteLine(cell.getFormat());
```

6.13.11 Метод Cell.getFormattedValue

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

Библиотеки Document API по умолчанию работают с документами используя английскую локализацию. Поэтому, при взаимодействии с документом через API, значения могут отличаться от отображаемых в интерфейсе редактора. Раздел Локализация документов содержит информацию о том, как работать с локализованными значениями.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");
cell.setNumber(2.3);
Console.WriteLine(cell.getFormattedValue());
```

6.13.12 Метод Cell.getFormulaAsString

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
Console.WriteLine(cell.getFormulaAsString());
```

6.13.13 Метод Cell.getHyperlink

Возвращает первый объект в ячейке типа Hyperlink.

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");
Hyperlink hyperlink = cell.getHyperlink();
```

6.13.14 Метод Cell.getMergedRange

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

Вызов

```
public CellRange getMergedRange()
```

Возвращает

– диапазон ячеек, тип CellRange.

Пример

```
Table sheet = document.getBlocks().getTable(0);
sheet.getCellRange("D2:F6").merge();
Cell cell = sheet.getCell("E4");
CellRange mergedRange = cell.getMergedRange();
Console.WriteLine(mergedRange.getAddress(new CellRangeAddressSettings())); //
D2:F6
```

6.13.15 Метод Cell.getNote

Метод возвращает текст заметки для текущей ячейки.

Вызов

```
public string getNote()
```

Возвращает

– текст заметки.

Пример

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

Методы `Cell.setNote` и `Cell.removeNote` позволяют добавить и удалить заметку.

6.13.16 Метод Cell.getNumberValue

Метод возвращает числовое значение ячейки.

Вызов

```
public float? getNumberValue()
```

Возвращает

– числовое значение ячейки, тип `float`.

– `null`, если ячейка не содержит числового значения.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("D2");
cell.setNumber(0.1);


cell.getNumberValue();
```

6.13.17 Метод Cell.getParagraphProperties

Возвращает свойства абзаца `ParagraphProperties`, находящегося в ячейке.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
ParagraphProperties paragraphProperties = cell.getParagraphProperties();
Console.WriteLine(paragraphProperties.alignment);
```

6.13.18 Метод Cell.getPivotTable

Возвращает сводную таблицу `PivotTable`, относящуюся к ячейке.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null) {
    Console.WriteLine(pivotTable.getSourceRangeAddress());
}
```

6.13.19 Метод Cell.getProtectionProperties

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

Вызов

```
public CellProtectionProperties getProtectionProperties()
```

Возвращает

– `CellProtectionProperties`: свойства защиты ячейки (`null`, если ячейка находится в сводной таблице).

Пример

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

CellProtectionProperties cellProps = cell.getProtectionProperties();
cellProps.lockedForChanges = false;
cellProps.formulasNotDisplayed = false;

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

6.13.20 Метод Cell.getRange

Метод возвращает объект `Range` для управления содержимым ячейки.

6.13.21 Метод Cell.getRawValue

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");
Console.WriteLine(cell.getRawValue());
```

6.13.22 Метод Cell.getResolvedBorders

Метод возвращает границы ячейки с учетом границ окружающих ячеек.

Вызов

```
public Borders getResolvedBorders()
```

Возвращает

– границы ячейки, тип Borders.

Пример

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

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Solid;
lineProperties.width = 1.5f;
Borders borders = new Borders();
borders.setLeft(lineProperties).setRight(lineProperties);
cellBorders.setBorders(borders);

Cell cell = sheet.getCell("A2");
cell.getBorders().getRight(); // null
cell.getResolvedBorders().getRight().width; // 1,5
```

6.13.23 Метод Cell.getRowIndex

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

Вызов

```
public uint getRowIndex()
```

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("A3");
Console.WriteLine(cell.getColumnIndex()); // 0
Console.WriteLine(cell.getRowIndex()); // 2
```

6.13.24 Метод Cell.getTable

Метод возвращает таблицу `Table`, в которой находится текущая ячейка.

Вызов

```
public Table getTable()
```

Пример

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

6.13.25 Метод Cell.getTextProperties

Метод возвращает текущие настройки форматирования текста для ячейки.

Вызов

```
public TextProperties getTextProperties()
```

Возвращает

– свойства форматирования текста, тип `TextProperties`.

Поля возвращаемого объекта `TextProperties` могут быть `null`, если ячейка содержит текст с разными настройками.

Пример

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

TextProperties props = cell.getTextProperties();
props.backgroundColor = new ColorRGBA(0, 0, 255, 200);
props.textColor = new Color(new ColorRGBA(255, 0, 0, 255));

cell.setTextProperties(props);
```

Метод Cell.setTextProperties позволяет задать настройки форматирования текста в ячейке.

6.13.26 Метод Cell.isContentEmpty

Метод позволяет определить содержит ли ячейка данные.

Вызов

```
public bool isContentEmpty()
```

Возвращает

– `true`, если ячейка не содержит данные, в ином случае – `false`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell1 = sheet.getCell("A1");
Cell cell2 = sheet.getCell("A2");
Cell cell3 = sheet.getCell("A3");
cell1.setText("123");
cell2.setTextProperties(new TextProperties() { bold = true });

cell1.isEmpty(); // false
cell1.isContentEmpty(); // false

cell2.isEmpty(); // false
cell2.isContentEmpty(); // true

cell3.isEmpty(); // true
cell3.isContentEmpty(); // true
```

6.13.27 Метод Cell.isEmpty

Метод позволяет определить содержит ли ячейка данные или настройки форматирования.

Вызов

```
public bool isEmpty()
```

Возвращает

– `true`, если ячейка не содержит данные или настройки форматирования, в ином случае – `false`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell1 = sheet.getCell("A1");
Cell cell2 = sheet.getCell("A2");
Cell cell3 = sheet.getCell("A3");
cell1.setText("123");
cell2.setTextProperties(new TextProperties() { bold = true });

cell1.isEmpty(); // false
cell1.isContentEmpty(); // false

cell2.isEmpty(); // false
cell2.isContentEmpty(); // true

cell3.isEmpty(); // true
cell3.isContentEmpty(); // true
```

6.13.28 Метод Cell.isFormula

Метод позволяет определить, содержит ли текущая ячейка формулу.

Вызов

```
public bool isFormula()
```

Возвращает

– `true`, если ячейка содержит формулу, в ином случае – `false`.

6.13.29 Метод Cell.isInMergedRange

Метод позволяет определить находится ли ячейка в объединенном диапазоне.

Вызов

```
public bool isInMergedRange()
```

Возвращает

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

Пример

```
Table sheet = document.getBlocks().getTable(0);
sheet.getCellRange("A1:C1").merge();
```

```
Cell cell1 = sheet.getCell("B1");
Cell cell2 = sheet.getCell("B2");

Console.WriteLine(cell1.isInMergedRange()); // True
Console.WriteLine(cell2.isInMergedRange()); // False
```

6.13.30 Метод Cell.isPivotTableRoot

Метод позволяет определить является ли ячейка основанием сводной таблицы.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("A1");
Console.WriteLine(cell.isPivotTableRoot());
```

6.13.31 Метод Cell.isProtected

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

Вызов

```
public bool isProtected()
```

6.13.32 Метод Cell.removeNote

Метод удаляет заметку из текущей ячейки.

Вызов

```
public void removeNote()
```

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A3");
if (cell.getNote() != null)
    cell.removeNote();
```

Методы `Cell.setNote` и `Cell.getNote` позволяют добавить заметку и получить её текст.

6.13.33 Метод Cell.setBool

Устанавливает для ячейки значение логического типа.

Пример

```
Cell cell = sheet.getCell("A1");
cell.setBool(true);
```

6.13.34 Метод Cell.setBorders

Метод предназначен для установки границ ячейки. Примеры использования приведены в разделе Borders.

6.13.35 Метод Cell.setCellProperties

Позволяет установить свойства ячейки CellProperties.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
CellProperties cellProperties = cell.getCellProperties();
cellProperties.verticalAlignment = VerticalAlignment.Center;
cell.setCellProperties(cellProperties);
```

6.13.36 Метод Cell.setContent

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

Пример

```
Cell cell = firstSheet.getCell("A1");
cell.setContent("=A2+A3");
```

6.13.37 Метод Cell.setCustomFormat

Устанавливает формат ячейки.

Пример

```
Cell cell = firstSheet.getCell("A1");
cell.setCustomFormat("0,00");
```

6.13.38 Метод Cell.setFormat

Метод устанавливает формат ячейки. Существуют несколько вариантов использования метода.

Варианты вызова метода

```
setFormat(cellFormat)
```

Где **cellFormat** – формат ячейки типа CellFormat.

```
setFormat(accountingCellFormatting)
```

Где **accountingCellFormatting** – формат ячейки типа AccountingCellFormatting.

```
setFormat(percentageCellFormatting)
```

Где **percentageCellFormatting** – формат ячейки типа PercentageCellFormatting.

```
setFormat(numberCellFormatting)
```

Где **numberCellFormatting** – формат ячейки типа NumberCellFormatting.

```
setFormat(currencyCellFormatting)
```

Где **currencyCellFormatting** – формат ячейки типа CurrencyCellFormatting.

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

Где **dateTimeCellFormatting** – формат ячейки типа DateTimeCellFormatting, **typeFormat -** формат даты/времени типа CellFormat.

```
setFormat(fractionCellFormatting)
```

Где **fractionCellFormatting** – формат ячейки типа FractionCellFormatting.

```
setFormat(scientificCellFormatting)
```

Где **scientificCellFormatting** – формат ячейки типа ScientificCellFormatting.

Примеры использования

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

// Формат : Общий
cell.setFormat(CellFormat.General);
```

```
Console.WriteLine(cell.getFormat()); // 0
Console.WriteLine(cell.getRange().extractText()); // 2,3

// Формат : Процентный
PercentageCellFormatting percentageCellFormatting = new
PercentageCellFormatting();
percentageCellFormatting.decimalPlaces = 1;
cell.setFormat(percentageCellFormatting);
Console.WriteLine(cell.getFormat()); // 1
Console.WriteLine(cell.getRange().extractText()); // 230,0%

// Формат : Числовой
NumberCellFormatting numberCellFormatting = new NumberCellFormatting();
numberCellFormatting.decimalPlaces = 2;
cell.setFormat(numberCellFormatting);
Console.WriteLine(cell.getFormat()); // 2
Console.WriteLine(cell.getRange().extractText()); // 2,30

// Формат : Денежный
CurrencyCellFormatting currencyCellFormatting = new CurrencyCellFormatting();
currencyCellFormatting.symbol = "$";
cell.setFormat(currencyCellFormatting);
Console.WriteLine(cell.getFormat()); // 4
Console.WriteLine(cell.getRange().extractText()); // 2,30$

// Формат : Финансовый
AccountingCellFormatting accountingCellFormatting = new
AccountingCellFormatting();
accountingCellFormatting.symbol = "₽";
cell.setFormat(accountingCellFormatting);
Console.WriteLine(cell.getFormat()); // 5
Console.WriteLine(cell.getRange().extractText()); // 2,30₽

// Формат : Дата / Время
DateTimeCellFormatting dateTimeCellFormatting = new DateTimeCellFormatting();
dateTimeCellFormatting.dateListID = DatePatterns.FullDate;
dateTimeCellFormatting.timeListID = TimePatterns.ShortTime;
cell.setFormat(dateTimeCellFormatting, CellFormat.DateTime);
```

```
Console.WriteLine(cell.getFormat()); // 8
Console.WriteLine(cell.getRange().extractText()); // понедельник, 1 января 1900
г. 7 : 12

// Формат : Дробный
FractionCellFormatting fractionCellFormatting = new FractionCellFormatting();
fractionCellFormatting.minNumeratorDigits = 2;
cell.setFormat(fractionCellFormatting);
Console.WriteLine(cell.getFormat()); // 9
Console.WriteLine(cell.getRange().extractText()); // 2 2 / 7

// Формат : Научный
ScientificCellFormatting cellFormatting = new ScientificCellFormatting();
cellFormatting.decimalPlaces = 5;
cell.setFormat(cellFormatting);
Console.WriteLine(cell.getFormat()); // 10
Console.WriteLine(cell.getRange().extractText()); // 2, 30000E+00
```

6.13.39 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 `CellFormat.Text`.

Список поддерживаемых форматов см. в разделе `CellFormat`.

Пример

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
cell.setFormattedValue("22.07.2020");
Console.WriteLine(cell.getFormattedValue());

cell.setFormattedValue("12:39");
Console.WriteLine(cell.getFormattedValue());
```

6.13.40 Метод Cell.setFormula

Метод позволяет вставить формулу в ячейку табличного документа.

Пример

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

6.13.41 Метод Cell.setNote

Метод добавляет заметку в текущую ячейку.

Вызов

```
public void setNote(string noteText)
```

Параметры

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A3");
if (cell.getNote() == null)
    cell.setNote("New Note");
```

Методы `Cell.getNote` и `Cell.removeNote` позволяют получить текст заметки или удалить её.

6.13.42 Метод Cell.setNumber

Устанавливает для ячейки значение числового типа.

Пример

```
Cell cell = sheet.getCell("A1");
cell.setNumber(0.0001);
```

6.13.43 Метод Cell.setParagraphProperties

Устанавливает свойства абзаца `ParagraphProperties`, находящегося в ячейке.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
ParagraphProperties paragraphProperties = cell.getParagraphProperties();
paragraphProperties.alignment = Alignment.Center;
cell.setParagraphProperties(paragraphProperties);
```

6.13.44 Метод Cell.setProtectionProperties

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

Вызов

```
public void setProtectionProperties(CellProtectionProperties protectionProps)
```

Параметры

– `protectionProps`: свойства защиты ячейки, тип `CellProtectionProperties`.

Пример

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

CellProtectionProperties cellProps = cell.getProtectionProperties();
cellProps.lockedForChanges = false;
cellProps.formulasNotDisplayed = false;

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

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

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

6.13.45 Метод Cell.setText

Устанавливает для ячейки значение строкового типа.

Пример

```
Cell cell = sheet.getCell("A1");
cell.setText("One");
```

6.13.46 Метод Cell.setTextProperties

Метод задает настройки форматирования текста для ячейки.

Вызов

```
public void setTextProperties(TextProperties props)
```

Параметры

– `props`: свойства форматирования текста, тип `TextProperties`.

Пример

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

TextProperties props = cell.getTextProperties();
props.backgroundColor = new ColorRGBA(0, 0, 255, 200);
props.textColor = new Color(new ColorRGBA(255, 0, 0, 255));

cell.setTextProperties(props);
```

Метод `Cell.getTextProperties` позволяет получить текущие настройки форматирования текста в ячейке.

6.13.47 Метод Cell.unmerge

Разъединяет несколько ячеек, которые были объединены ранее.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");
cell.unmerge();
```

6.14 Перечисление CellFormat

По умолчанию при создании документа всем ячейкам присваивается формат «Общий». Полный список форматов представлен в таблице 9. Перечисление `CellFormat` используется в методах `Cell.getFormat()` и `Cell.setFormat()`, а также в поле `PivotTableValueField.cellNumberFormat`.

Таблица 9 – Поддерживаемые форматы ячеек таблицы

Значение	Описание
`CellFormat.General`	Формат ячейки «Общий». В этом формате в ячейке отображаются первые 9 символов числа, остальные доступны для просмотра в строке формул. Для дробных чисел в формате «Общий» незначащие нули в дробной части не отображаются. Числа, состоящие более чем из 12 символов, переводятся в экспоненциальную форму после завершения ввода в ячейку.
`CellFormat.Percentage`	Формат ячейки «Процентный». Этот формат используется для представления чисел как процентов. При применении формата «Процентный» введенное число умножается на 100 и обозначается знаком «%».
`CellFormat.Number`	Формат ячейки «Числовой». Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя.
`CellFormat.Text`	Формат ячейки «Текстовый».
`CellFormat.Currency`	Формат ячейки «Денежный». Этот формат используется для представления чисел со знаком или кодом валюты.
`CellFormat.Accounting`	Формат ячейки «Финансовый». Этот формат применяется для чисел, используемых в бухгалтерских документах. В формате «Финансовый» введенное

Значение	Описание
	число автоматически дополняется названием валюты, которая соответствует настройкам системы компьютера. Отрицательные числа в формате «Финансовый» заключаются в круглые скобки в ячейке, а в строке формул остаются в том виде, в котором они были введены.
`CellFormat.Date`	Формат ячейки «Дата». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ.
`CellFormat.Time`	Формат ячейки «Время». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.
`CellFormat.DateTime`	Формат ячейки «Дата + Время»
`CellFormat.Fraction`	Формат ячейки «Дробный». Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель).
`CellFormat.Scientific`	Формат ячейки «Экспоненциальный». Экспоненциальный (или научный) формат используется для представления больших чисел в короткой форме. Все введенные числа длиной более 12 символов автоматически переводятся в этот формат. В ячейке число в формате «Экспоненциальный» представлено следующим образом: – целая часть, всегда состоящая из одной цифры; – разделитель целой и дробной части; – дробная часть, по умолчанию состоящая из двух цифр; – показатель степени числа 10 в виде `E<знак показателя степени> <показатель степени>`.
`CellFormat.Custom`	Пользовательский формат.

Использование данных значений позволяет установить желаемый формат. При этом будут использованы параметры формата по умолчанию.

Примеры использования

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
cell.setFormat(CellFormat.General);
cell = firstSheet.getCell("B2");
```

```
cell.setFormat(CellFormat.Percentage);
cell = firstSheet.getCell("B3");
cell.setFormat(CellFormat.Number);
```

Результат

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

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

6.15 Класс CellPosition

Класс `CellPosition` позволяет задать координаты ячейки листа табличного документа или таблицы в составе текстового документа. Также используется для описания полей `topLeft`, `rightBottom` класса `CellRangePosition`.

Конструкторы

```
public CellPosition()
```

```
public CellPosition(uint rowArg, uint columnArg)
```

– `rowArg` – индекс строки ячейки (индексация начинается с нуля), тип `uint`;

– `columnArg` – индекс колонки ячейки, тип `uint`.

Примеры

```
Table table = document.getBlocks().getTable(0);
Cell cell = table.getCell(new CellPosition(2, 0));
```

```
Table table = document.getBlocks().getTable("List11");
Charts charts = table.getCharts();
ChartRangeInfo rangeInfo = charts.getChart(0).getRange(0);
TableRangeInfo cellRangePosition = rangeInfo.tableRangeInfo;
CellRangePosition tableRange = cellRangePosition.tableRange;
CellPosition topLeftCellPosition = tableRange.topLeft;
Console.WriteLine("top left row:", topLeftCellPosition.row, ", top left column:",
topLeftCellPosition.column);
```

6.15.1 Поле CellPosition.column

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

Примеры

```
CellPosition cellPosition = new CellPosition(1, 2);
```

```
CellPosition cellPosition = new CellPosition();
cellPosition.column = 3;
```

6.15.2 Поле CellPosition.row

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

Примеры

```
CellPosition cellPosition = new CellPosition(1, 2);
```

```
CellPosition cellPosition = new CellPosition();
cellPosition.row = 3;
```

6.15.3 Метод CellPosition.toString

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

Пример

```
CellPosition cellPosition = new CellPosition(0, 0);
Console.WriteLine(cellPosition.toString());
```

6.16 Класс CellProperties

Класс `CellProperties` предназначен для форматирования содержимого в ячейках таблицы. Описание полей представлено в таблице 10.

Для задания свойств ячеек используются методы `Cell.setCellProperties()` и `CellRange.setCellProperties()`. Для получения свойств ячеек используются методы `Cell.getCellProperties()` и `CellRange.getCellProperties()`. Иерархия классов и полей для работы с `CellProperties` отображена на рисунке 1.

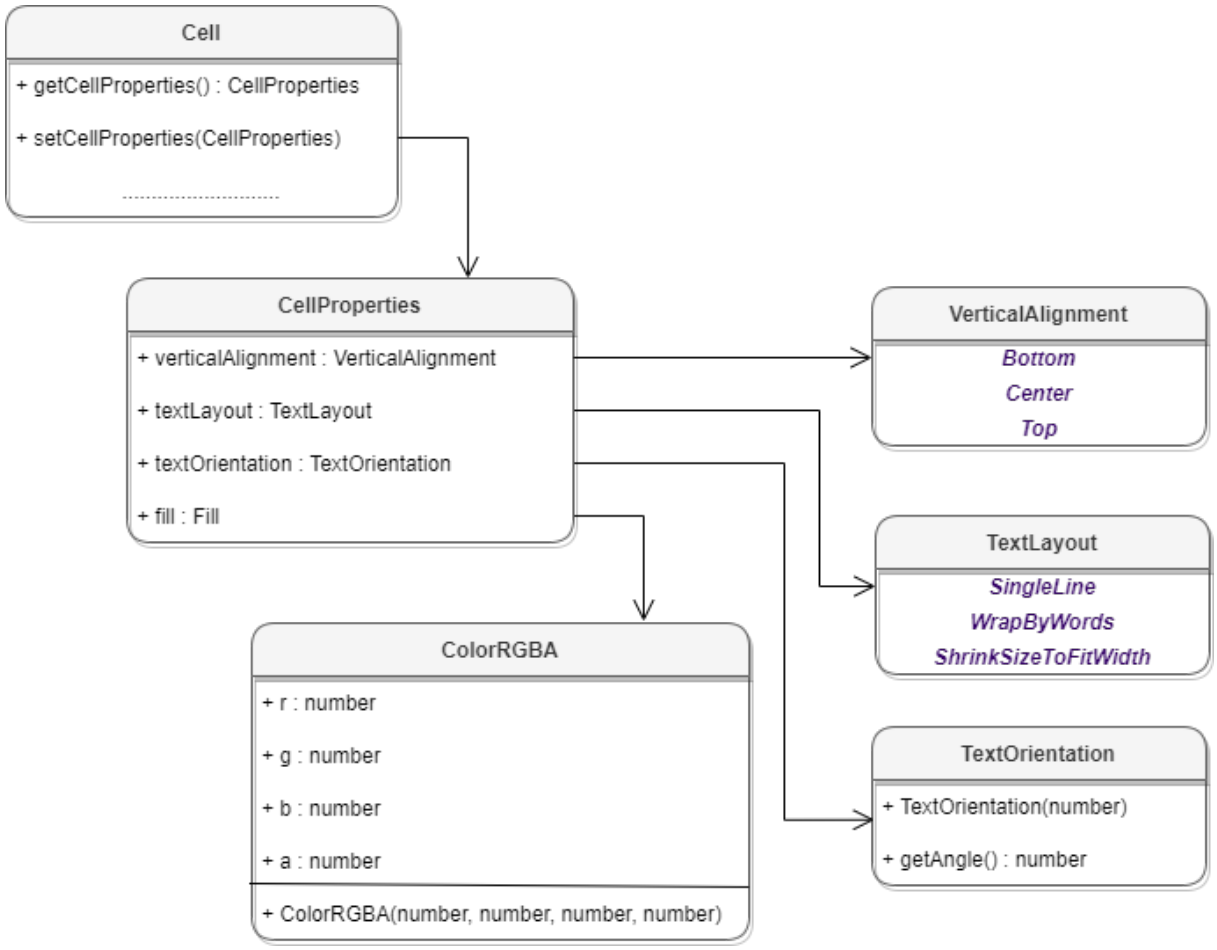


Рисунок 36 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 10 – Описание полей класса `CellProperties`

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

Пример

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

CellProperties cellProps = cell.getCellProperties();
cellProps.verticalAlignment = VerticalAlignment.Center;
```

```
cellProps.textLayout = TextLayout.ShrinkSizeToFitWidth;
cellProps.fill = new Fill(new Color(new ColorRGBA(255, 255, 0, 255)));
cellProps.textOrientation = new TextOrientation(45);

cell.setCellProperties(cellProps);
```

6.17 Класс CellProtectionProperties

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

Таблица 11 – Описание полей класса `CellProtectionProperties`

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

Пример

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

CellProtectionProperties cellProps = cell.getProtectionProperties();
cellProps.lockedForChanges = false;
cellProps.formulasNotDisplayed = false;

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

6.17.1 Метод CellProtectionProperties.toString

Метод возвращает текстовое представление текущих параметров защиты ячеек в формате: `[lockedForChanges: #, formulasNotDisplayed: #]`.

Вызов

```
public string toString()
```

Возвращает

– текстовое представление параметров защиты ячеек, тип `string`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("A1");
Console.WriteLine(cell.getProtectionProperties().toString()); //
[lockedForChanges: true, formulasNotDisplayed: false]
```

6.18 Класс CellRange

Класс `CellRange` описывает диапазон ячеек таблицы.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
```

6.18.1 Метод CellRange.autoFill

Метод заполняет диапазон ячеек, используя данные из этого диапазона в качестве источника. Целевой диапазон вычисляется из начальной позиции исходного диапазона и последней позиции (аргумент `destination`, тип `CellPosition`). Таким образом, конечный диапазон всегда полностью содержит исходный диапазон.

Метод `autoFill` автоматически интерполирует исходные точки и находит алгоритм аппроксимации, который используется для экстраполяции значений в диапазоне ячеек назначения. Результат выполнения метода в текстовом редакторе может отличаться от табличного редактора из-за разных типов данных в ячейках.

Возвращает `true`, если ячейки успешно заполнены и `false` в других случаях (например, если диапазон ячеек назначения содержит формулу или сводную таблицу и т. д.). Вызывает `OutOfRangeError`, если исходный или целевой диапазоны находятся за пределами таблицы.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
cellRange.autoFill(new CellPosition(2, 0));
```

6.18.2 Метод CellRange.calculate

Метод пересчитывает формулы в текущем диапазоне ячеек.

Вызов

```
public void calculate()
```

6.18.3 Метод CellRange.clearDataValidations

Метод убирает проверку данных из текущего диапазона ячеек.

Вызов

```
public void clearDataValidations()
```

Пример

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

6.18.4 Метод CellRange.clearFormat

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

Вызов

```
public void clearFormat(bool skipHiddenCells)
```

Параметры

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

6.18.5 Метод CellRange.clearStyle

Метод очищает форматирование диапазона ячеек.

Вызов

```
public void clearStyle(bool skipHiddenCells)
```

Параметры

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

6.18.6 Метод CellRange.containsCell

Метод определяет принадлежность ячейки диапазону. В качестве параметра выступает объект `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

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
Cell cell = firstSheet.getCell("A1");
Console.WriteLine(cellRange.containsCell(cell));
```

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

6.18.7 Метод CellRange.copyInto

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

Данный метод реализован только в табличных документах.

Вызов

```
public void copyInto(CellRange destination, CellRangePastingSettings pastingSettings)
```

Параметры

– `destination`: целевой диапазон, тип `CellRange`.

– `pastingSettings`: (необязательный) параметры копирования, тип `CellRangePastingSettings`.

Пример (только для табличного документа)

```
Table table = document.getBlocks().getTable(0);
var sourceRange = table.getCellRange("A1:B2");
var destRange = table.getCellRange("C3:D4");
sourceRange.copyInto(destRange);
```

Пример копирования ячеек между листами

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

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

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

sourceRange.copyInto(destRange);
```

Пример настройки копирования ячеек

```
Table sheet = document.getBlocks().getTable(0);
CellRange sourceRange = sheet.getCellRange("A1:C3");
CellRange destRange = sheet.getCellRange("D4:F6");

CellRangePastingSettings settings = new CellRangePastingSettings();
settings.formulasPastingPolicy = FormulasPastingPolicy.AsValues;
settings.stylesPastingPolicy = StylesPastingPolicy.PlainText;

sourceRange.copyInto(destRange, settings);
```

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 1).

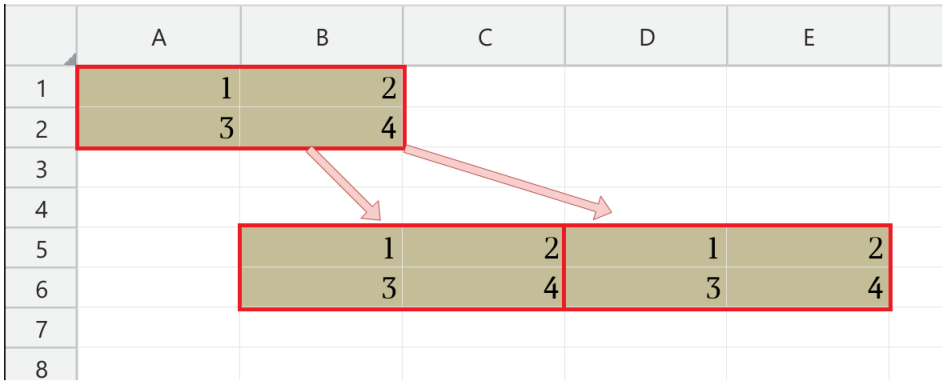


Рисунок 37 – Копирование ячеек табличного документа

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

6.18.8 Метод CellRange.find

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

> ⚠ Библиотеки Document API по умолчанию работают с документами используя английскую локализацию. Поэтому, при взаимодействии с документом через API, значения могут отличаться от отображаемых в интерфейсе редактора. Раздел Локализация документов содержит информацию о том, как работать с локализованными значениями.

Вызов

```
public CellsEnumerator find(string string_, TableSearchSettings settings)
```

Параметры

– `string_`: поисковый запрос, тип `string`.

– `settings`: (необязательный) параметры поиска, тип `TableSearchSettings`.

Возвращает

– список ячеек, соответствующих поисковому запросу, тип `CellsEnumerator`.

Пример

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

TableSearchSettings searchProps = new TableSearchSettings();
```

```
searchProps.caseSensitive = CaseSensitive.No;
searchProps.matchBehaviour = TableSearchSettings.MatchBehaviour.Glob;
searchProps.searchIn = TableSearchSettings.SearchProperty.Value;
searchProps.wholeWords = true;

CellsEnumerator results = cellRange.find("*eye", searchProps);
foreach (Cell cell in results) {
    Console.WriteLine(cell.getFormattedValue()); // Steeleye Stout
}
```

6.18.9 Метод CellRange.getAddress

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

Вызов

```
public string getAddress(CellRangeAddressSettings addressSettings)
```

Параметры

– `addressSettings`: настройки форматирования адреса, тип `CellRangeAddressSettings`.

Возвращает

– адрес диапазона ячеек.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange range = sheet.getCellRange("A1:C11");

CellRangeAddressSettings addressSettings = new CellRangeAddressSettings();
addressSettings.isFileNameRequired = true;
addressSettings.isWorksheetNameRequired = true;
addressSettings.addressFormat = CellRangeAddressFormat.A1;
addressSettings.isAbsoluteRow = true;
addressSettings.isAbsoluteColumn = false;

Console.WriteLine(range.getAddress(addressSettings)); // '[sheet.xods]
Data'!$A1:$C11
```

6.18.10 Метод CellRange.getBeginColumn

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
Console.WriteLine(cellRange.getBeginColumn());
```

6.18.11 Метод CellRange.getBeginRow

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
Console.WriteLine(cellRange.getBeginRow());
```

6.18.12 Метод CellRange.getCellProperties

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
CellProperties cellProperties = cellRange.getCellProperties();
Console.WriteLine(cellProperties.fill.getColor().getRGBAColor().r);
```

6.18.13 Метод CellRange.getEnumerator

Метод возвращает коллекцию ячеек в диапазоне.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
CellsEnumerator cellEnumerator = cellRange.GetEnumerator();
```

```
foreach (var cell in cellEnumerator)
{
    Console.WriteLine(cell.getFormattedValue());
}
```

6.18.14 Метод CellRange.getLastColumn

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
Console.WriteLine(cellRange.getLastColumn());
```

6.18.15 Метод CellRange.getLastRow

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
Console.WriteLine(cellRange.getLastRow());
```

6.18.16 Метод CellRange.getParagraphProperties

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

Вызов

```
public ParagraphProperties getParagraphProperties(bool skipHiddenCells)
```

Параметры

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

Возвращает

– свойства форматирования абзацев, тип `ParagraphProperties`.

Пример

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

ParagraphProperties paragraphProperties = cellRange.getParagraphProperties();
paragraphProperties.alignment = Alignment.Center;

cellRange.setParagraphProperties(paragraphProperties, false);
```

Свойства возвращаемого объекта ParagraphProperties могут быть null, если выбранный диапазон содержит ячейки с разными параметрами. Метод CellRange.setParagraphProperties позволяет задать настройки форматирования абзацев, находящихся в диапазоне ячеек.

6.18.17 Метод CellRange.getProtectionProperties

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

Вызов

```
public CellProtectionProperties getProtectionProperties()
```

Возвращает

– CellProtectionProperties: свойства защиты диапазона ячеек (null, если диапазон содержит только ячейки сводной таблицы).

Пример

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

CellProtectionProperties cellProps = cellRange.getProtectionProperties();
cellProps.lockedForChanges = false;
cellProps.formulasNotDisplayed = false;

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

Свойства возвращаемого объекта CellProtectionProperties могут быть null, если текущий диапазон содержит ячейки с разными параметрами.

6.18.18 Метод CellRange.getTable

Возвращает таблицу `Table`, содержащую текущий диапазон.

6.18.19 Метод CellRange.getTableRange

Возвращает положение текущего диапазона ячеек в таблице (объект `CellRangePosition`).

6.18.20 Метод CellRange.getTextProperties

Метод возвращает текущие настройки форматирования текста для диапазона ячеек.

Вызов

```
public TextProperties getTextProperties(bool skipHiddenCells)
```

Параметры

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

Возвращает

– свойства форматирования текста, тип `TextProperties`.

Поля возвращаемого объекта `TextProperties` могут быть `null`, если диапазон содержит ячейки с разными настройками.

Пример

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

TextProperties props = cellRange.getTextProperties();
props.backgroundColor = new ColorRGBA(0, 0, 255, 200);
props.textColor = new Color(new ColorRGBA(255, 0, 0, 255));

cellRange.setTextProperties(props);
```

Метод `CellRange.setTextProperties` позволяет задать настройки форматирования текста в диапазоне ячеек.

6.18.21 Метод CellRange.insert

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

Вызов

```
public void insert(CellShiftAxis shiftAxis)
```

Параметры

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

Пример

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

cellRange.insert(CellShiftAxis.Vertical);
```

6.18.22 Метод CellRange.insertCurrentDateTime

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

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange cellRange = sheet.getCellRange("A1");
cellRange.insertCurrentDateTime(DateTimeFormat.DateTime);
```

6.18.23 Метод CellRange.intersect

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

Вызов

```
public CellRange intersect(CellRange other)
```

Параметры

– `other`: диапазон ячеек, тип `CellRange`.

Возвращает

– диапазон пересечения, тип `CellRange`.

– `null`, если диапазоны ячеек не пересекаются.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange cellRange1 = sheet.getCellRange("F2:F6");
CellRange cellRange2 = sheet.getCellRange("A4:H6");
CellRange cells = cellRange1.intersect(cellRange2);
Console.WriteLine(cells.getAddress(new CellRangeAddressSettings())); // F4:F6
```

6.18.24 Метод CellRange.isContentEmpty

Метод позволяет определить содержат ли ячейки диапазона данные.

Вызов

```
public bool isContentEmpty()
```

Возвращает

– `true`, если все ячейки диапазона не содержат данные, в ином случае – `false`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange range1 = sheet.getCellRange("A1:B1");
CellRange range2 = sheet.getCellRange("A2:B2");
CellRange range3 = sheet.getCellRange("A3:B3");
sheet.getCell("A1").setText("123");
range2.setTextProperties(new TextProperties() { bold = true });

range1.isEmpty(); // false
range1.isContentEmpty(); // false

range2.isEmpty(); // false
range2.isContentEmpty(); // true

range3.isEmpty(); // true
range3.isContentEmpty(); // true
```

6.18.25 Метод CellRange.isEmpty

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

Вызов

```
public bool isEmpty()
```

Возвращает

– `true`, если все ячейки диапазона не содержат данные или настройки форматирования, в ином случае – `false`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange range1 = sheet.getCellRange("A1:B1");
CellRange range2 = sheet.getCellRange("A2:B2");
CellRange range3 = sheet.getCellRange("A3:B3");
sheet.getCell("A1").setText("123");
range2.setTextProperties(new TextProperties() { bold = true });

range1.isEmpty(); // false
range1.isContentEmpty(); // false

range2.isEmpty(); // false
range2.isContentEmpty(); // true

range3.isEmpty(); // true
range3.isContentEmpty(); // true
```

6.18.26 Метод CellRange.isProtected

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

Вызов

```
public bool? isProtected()
```

Метод `isProtected()` возвращает `null`, если часть ячеек диапазона защищена от редактирования, а часть – нет.

6.18.27 Метод CellRange.merge

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

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
cellRange.merge();
```

6.18.28 Метод CellRange.moveInto

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

Данный метод реализован только в табличных документах.

Пример (только для табличного документа)

```
Table table = document.getBlocks().getTable(0);
var sourceRange = table.getCellRange("A1:B2");
var destRange = table.getCellRange("C3:D4");
sourceRange.moveInto(destRange);
```

Пример перемещения ячеек между документами

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

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

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

sourceRange.moveInto(destRange);
```

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

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

6.18.29 Метод CellRange.remove

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

Вызов

```
public void remove(CellShiftAxis shiftAxis)
```

Параметры

– `shiftAxis`: направление сдвига ячеек (влево или вверх), тип `CellShiftAxis`.

Пример

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

cellRange.remove(CellShiftAxis.Horizontal);
```

6.18.30 Метод CellRange.removeContent

Метод очищает содержимое диапазона ячеек.

Вызов

```
public void removeContent(bool skipHiddenCells)
```

Параметры

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

6.18.31 Метод CellRange.setArrayFormula

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

Вызов

```
public void setArrayFormula(string arrayFormula)
```

Параметры

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

Примеры

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

```
Table sheet = document.getBlocks().getTable(0);
CellRange cellRange = sheet.getCellRange("C1:C7");
cellRange.setArrayFormula("=A1:A7 > 0");
Cell cell = sheet.getCell("C3");
Console.WriteLine(cell.getFormattedValue());  // TRUE
Console.WriteLine(cell.getFormulaAsString()); // {=A1:A7>0}
```

6.18.32 Метод CellRange.setBorders

Метод предназначен для установки границ диапазона ячеек. Отдельные границы устанавливаются с помощью методов класса Borders.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A1");

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Dash;
lineProperties.width = 1.5f;
lineProperties.color = new Color(new ColorRGBA(255, 0, 0, 255));

Borders newBorders = new Borders();
newBorders.setLeft(lineProperties);
newBorders.setRight(lineProperties);
newBorders.setTop(lineProperties);
newBorders.setBottom(lineProperties);

cell.setBorders(newBorders);
```

6.18.33 Метод CellRange.setCellProperties

Метод предназначен для установки свойств CellProperties ячеек диапазона.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
CellProperties cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(55, 146, 179, 200)));
cellRange.setCellProperties(cellProperties);
```

6.18.34 Метод CellRange.setDataValidation

Метод устанавливает проверку данных для текущего диапазона ячеек.

Вызов

```
public void setDataValidation(DataValidation dataValidation)
```

Параметры

– dataValidation: настройки проверки данных, тип DataValidation.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange cellRange = sheet.getCellRange("E2:E10");
DataValidation dvList = new DataValidation();
dvList.setType(DataValidationType.List);
dvList.setFormula1("UK; Italy; Germany; Austria; Brazil");
dvList.setShowDropDown(true);

cellRange.setDataValidation(dvList);
```

6.18.35 Метод CellRange.setParagraphProperties

Метод задает настройки форматирования абзацев, находящихся в диапазоне ячеек.

Вызов

```
public void setParagraphProperties(ParagraphProperties paraProperties, bool
skipHiddenCells)
```

Параметры

– paraProperties: свойства форматирования абзацев, тип ParagraphProperties.

– `skipHiddenCells`: (необязательный, по умолчанию `true`) не применять настройки к отфильтрованным и скрытым ячейкам диапазона, тип `bool`.

Пример

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

ParagraphProperties paragraphProperties = cellRange.getParagraphProperties();
paragraphProperties.alignment = Alignment.Center;

cellRange.setParagraphProperties(paragraphProperties, false);
```

Метод `CellRange.getParagraphProperties` позволяет получить текущие настройки форматирования абзацев, находящихся в диапазоне ячеек.

6.18.36 Метод CellRange.setProtectionProperties

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

Вызов

```
public void setProtectionProperties(CellProtectionProperties protectionProps)
```

Параметры

– `protectionProps`: свойства защиты диапазона ячеек, тип `CellProtectionProperties`.

Пример

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

CellProtectionProperties cellProps = cellRange.getProtectionProperties();
cellProps.lockedForChanges = false;
cellProps.formulasNotDisplayed = false;

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

Метод `setProtectionProperties()` позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой

защиты). Для установки защиты используйте метод Table.setProtection() после задания параметров защиты ячеек. Если метод setProtectionProperties() применяется к ячейкам уже защищенного листа, возникает исключение SpreadsheetProtectionError.

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

6.18.37 Метод CellRange.setTextProperties

Метод задает настройки форматирования текста для диапазона ячеек.

Вызов

```
public void setTextProperties(TextProperties textProperties, bool skipHiddenCells)
```

Параметры

– textProperties: свойства форматирования текста, тип TextProperties.

– skipHiddenCells: (необязательный, по умолчанию true) не применять настройки к отфильтрованным и скрытым ячейкам диапазона, тип bool.

Пример

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

TextProperties props = cellRange.getTextProperties();
props.backgroundColor = new ColorRGBA(0, 0, 255, 200);
props.textColor = new Color(new ColorRGBA(255, 0, 0, 255));

cellRange.setTextProperties(props, false);
```

Метод CellRange.getTextProperties позволяет получить текущие настройки форматирования текста в диапазоне ячеек.

6.18.38 Метод CellRange.sort

Метод сортирует строки текущего диапазона в соответствии с заданными условиями.

Вызов

```
public void sort(SortingConditions sortingConditions)
```

Параметры

– `sortingConditions`: коллекция условий сортировки ячеек, тип `SortingConditions`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange range = sheet.getCellRange("A1:C11");

SortingConditions conditions = new SortingConditions();
conditions.add(0, SortingDirection.Descending);
conditions.add(1, SortingDirection.Ascending);

range.sort(conditions);
```

Метод вызывает `DocumentModificationError` в следующих случаях:

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

6.18.39 Метод CellRange.textToColumns

Метод разделяет текст в каждой ячейке диапазона на отдельные горизонтальные ячейки. Символы-разделители и настройки разделения задаются с помощью объекта `TextToColumnsSettings`. Данный метод можно использовать только у диапазона шириной в одну ячейку.

Вызов

```
public void textToColumns(TextToColumnsSettings settings)
```

Параметры

– `settings`: параметры разделения текста по ячейкам, тип `TextToColumnsSettings`.

Пример

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

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

Рисунок 39 – Разделение текста диапазона по ячейкам

```
Table sheet = document.getBlocks().getTable(0);
Cell cell1 = sheet.getCell("A1");
Cell cell2 = sheet.getCell("A2");
Cell cell3 = sheet.getCell("A3");
cell1.setText("1, Laptop, Moscow, 1500, 8, 'fragile, express'");
cell2.setText("2, UPS, 'Saint Petersburg', 600, 12, 'heavy, premium'");
cell3.setText("3, Mouse, Kazan, 200, 5, 'free shipping, standard'");

TextToColumnsSettings settings = new TextToColumnsSettings();
settings.useCommaDelimiter = true;
settings.useSpaceDelimiter = true;
settings.treatMultipleDelimitersAsOne = true;
settings.textQualifier = TextToColumnsSettings.TextQualifier.SingleQuotes;

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

6.19 Перечисление CellRangeAddressFormat

Перечисление `CellRangeAddressFormat` содержит форматы отображения адреса диапазона ячеек. Используется в поле `CellRangeAddressSettings.addressFormat`.

Таблица 12 – Описание форматов отображения адреса

Значение	Описание
`CellRangeAddressFormat.A1`	Отображение адреса в формате A1 (`A1:C11`)
`CellRangeAddressFormat.R1C1`	Отображение адреса в формате R1C1 (`R[0]C[0]:R[10]C[2]`)

6.20 Класс CellRangeAddressSettings

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

Таблица 13 – Описание полей класса `CellRangeAddressSettings`

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

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange range = sheet.getCellRange("A1:C11");

CellRangeAddressSettings addressSettings = new CellRangeAddressSettings();
addressSettings.isFileNameRequired = true;
addressSettings.isWorksheetNameRequired = true;
addressSettings.addressFormat = CellRangeAddressFormat.A1;
addressSettings.isAbsoluteRow = true;
addressSettings.isAbsoluteColumn = false;

Console.WriteLine(range.getAddress(addressSettings)); // '[sheet.xods]
Data'!$A1:$C11
```

6.21 Класс CellRangePastingSettings

Класс `CellRangePastingSettings` предназначен для настройки параметров копирования ячеек. Данный класс используется в методе `CellRange.copyInto()`.

Таблица 14 – Описание полей класса CellRangePastingSettings

Поле	Тип	Описание
CellRangePastingSettings.stylesPastingPolicy	StylesPastingPolicy	Режим копирования форматирования
CellRangePastingSettings.formulasPastingPolicy	FormulasPastingPolicy	Режим копирования формул

6.22 Класс CellRangePosition

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

Конструкторы

```
public CellRangePosition()
```

```
public CellRangePosition(CellPosition topLeftArg, CellPosition bottomRightArg)
```

– topLeftArg – позиция левой верхней ячейки диапазона, тип CellPosition;

– bottomRightArg – позиция правой нижней ячейки диапазона, тип CellPosition.

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

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

– topLeftColumn – индекс левого столбца диапазона, тип uint;

– bottomRightRow – индекс нижней строки диапазона, тип uint;

– bottomRightColumn – индекс правого столбца диапазона, тип uint.

Таблица 15 – Поля класса CellRangePosition

Поле	Тип	Описание
CellRangePosition.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.
`CellRangePosition.bottomRight`	`CellPosition`	Содержит позицию правой нижней ячейки таблицы прямоугольного диапазона.

Примеры

```
Table table = document.getBlocks().getTable(0);
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
CellRange range = table.getCellRange(cellRangePosition);
```

```
Chart chart = charts.getChart(0);
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
chart.setRange(cellRangePosition);
Console.WriteLine(chart.getRangeAsString());
```

6.22.1 Метод CellRangePosition.toString

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

Пример

```
Table table = document.getBlocks().getTable(0);
Charts charts = table.getCharts();
ChartRangeInfo rangeInfo = charts.getChart(0).getRange(0);
TableRangeInfo cellRangePosition = rangeInfo.tableRangeInfo;
CellRangePosition tableRange = cellRangePosition.tableRange;
Console.WriteLine(tableRange.toString()); // [topLeft: (row: 0, column: 0),
bottomRight: (row: 5, column: 5)];
```

6.23 Перечисление CellShiftAxis

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

Таблица 16 – Описание направлений сдвига ячеек

Значение	Описание
`CellShiftAxis.Horizontal`	Сдвигает ячейки по горизонтали (вправо при вставке, влево при удалении)
`CellShiftAxis.Vertical`	Сдвигает ячейки по вертикали (вниз при вставке, вверх при удалении)

Пример

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

cellRange.insert(CellShiftAxis.Vertical);
```

6.24 Класс Chart

Класс `Chart` представляет диаграмму в табличном документе и описывает все ее элементы (заголовок, легенда, тип, данные, диапазон и т.д). Объектная модель класса `Chart` приведена на Рис. 2.

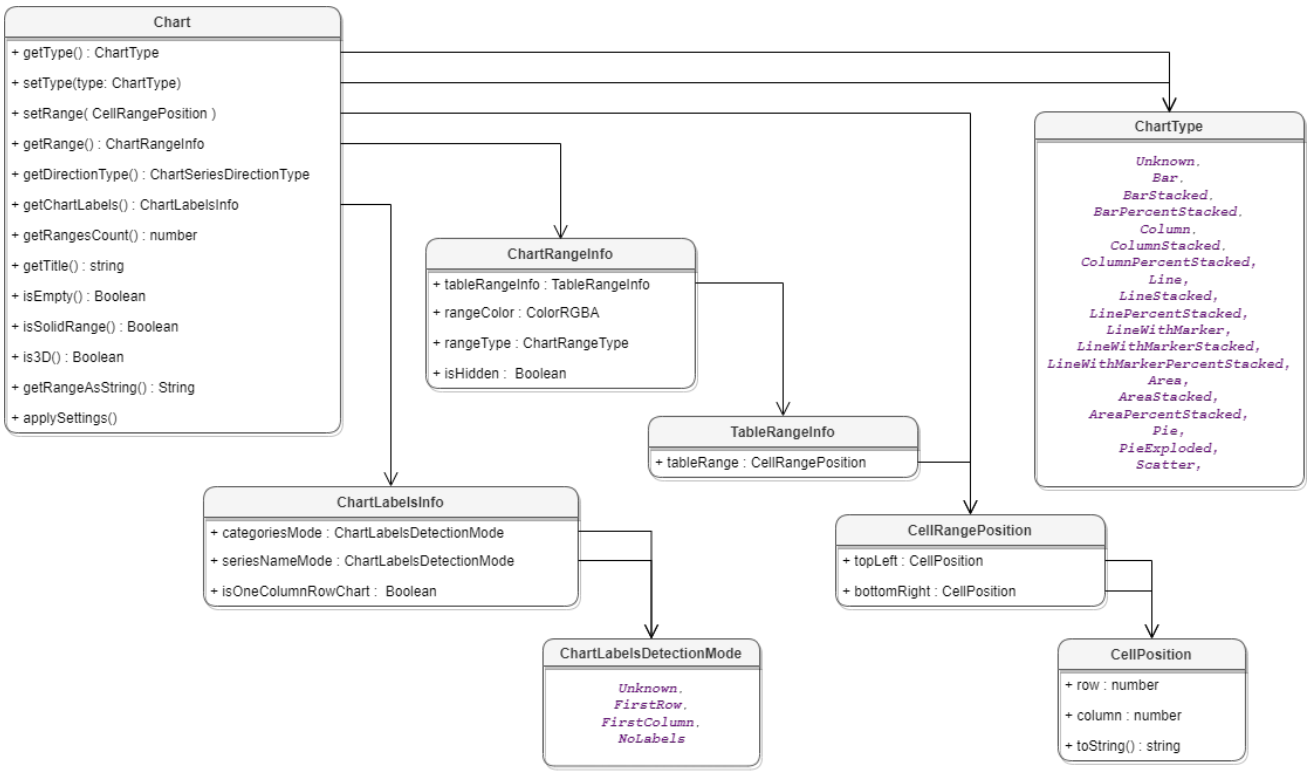


Рисунок 40 – Объектная модель класса `Chart`

6.24.1 Метод Chart.applySettings

Метод позволяет обновить параметры текущей выбранной диаграммы.

Вызов

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

Параметры

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

– `directionType` – направление серий `ChartSeriesDirectionType`;

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

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

Пример

```
CellRange cellRange = sheetDocumentPage.getCellRange("B3:C4");
ChartLabelsInfo chartLabelsInfo = new
ChartLabelsInfo(ChartLabelsDetectionMode.FirstColumn,
ChartLabelsDetectionMode.FirstRow, false);
chart.applySettings(cellRange, null, "Title", chartLabelsInfo);
```

6.24.2 Метод Chart.getChartLabels

Метод возвращает коллекцию меток диаграммы типа `ChartLabelsInfo`.

Пример

```
Chart chart = charts.getChart(0);
ChartLabelsInfo chartLabelsInfo = chart.getChartLabels();
Console.WriteLine(chartLabelsInfo.getType());
```

6.24.3 Метод Chart.getDirectionType

Метод возвращает направление `ChartSeriesDirectionType` серий диаграммы.

Пример

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.getDirectionType());
```

6.24.4 Метод Chart.getFrame

Метод аналогичен методу InlineObject.getFrame(), он возвращает свойства позиции изображения типа Frame.

Пример для текстового документа

```
Chart chart = charts.getChart(0);
Frame frame = chart.getFrame();
var absoluteFrame = frame.getAbsoluteFrame();
if (absoluteFrame != null) {
   Console.WriteLine(absoluteFrame.getDimensions().width);
}
```

6.24.5 Метод Chart.getRange

Метод возвращает диапазон ячеек ChartRangeInfo, содержащий исходные данные диаграммы. Параметр метода – индекс диапазона.

Пример

```
Charts charts = sheetDocumentPage.getCharts();
Chart chart = charts.getChart(0);
if (chart != null) {
    NCT.MyOfficeSDK.ChartRangeInfo chartRangeInfo = chart.getRange(0);
    if (chartRangeInfo != null) {
        Console.WriteLine(chartRangeInfo.rangeType);
    }
}
```

6.24.6 Метод Chart.getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

Пример

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.getRangeAsString());
```

6.24.7 Метод Chart.getRangesCount

Метод возвращает количество серий диаграммы.

Пример

```
Charts charts = sheetDocumentPage.getCharts();
Chart chart = charts.getChart(0);
if (chart != null) {
    Console.WriteLine(chart.getRangesCount());
}
```

6.24.8 Метод Chart.getTitle

Метод возвращает заголовок диаграммы.

Пример

```
Chart chart = charts.getChart(0);
String title = chart.getTitle();
if (title != null) {
    Console.WriteLine(chart.getTitle());
}
```

6.24.9 Метод Chart.getType

Метод возвращает тип диаграммы ChartType.

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Chart chart = charts.getChart(0);
    if (chart != null) {
        Console.WriteLine(chart.getType());
    }
}
```

6.24.10 Метод Chart.is3D

Метод возвращает `true`, если диаграмма трехмерная.

Пример

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.is3D());
```

6.24.11 Метод Chart.isEmpty

Метод возвращает `true`, если диаграмма не содержит значений.

Пример

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.isEmpty());
```

6.24.12 Метод Chart.isSolidRange

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

Пример

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.isSolidRange());
```

6.24.13 Метод Chart.setRange

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

Пример

```
Chart chart = charts.getChart(0);
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
chart.setRange(cellRangePosition);
Console.WriteLine(chart.getRangeAsString());
```

6.24.14 Метод Chart.setRect

Метод задает область расположения диаграммы.

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

Пример

```
Chart chart = charts.getChart(0);
chart.setRect(new RectU(0, 0, 20, 20));
```

6.24.15 Метод Chart.setType

Метод устанавливает тип диаграммы `ChartType`. В качестве параметра передается новый тип диаграммы.

Пример

```
Charts charts = sheetDocumentPage.getCharts();
Chart chart = charts.getChart(0);
if (chart != null) {
    chart.setType(ChartType.ColumnStacked);
    Console.WriteLine(chart.getType());
}
```

6.25 Перечисление ChartLabelsDetectionMode

Перечисление `ChartLabelsDetectionMode` содержит режимы автоматического определения меток диаграмм. Используется в полях `ChartLabelsInfo.categoriesMode` и `ChartLabelsInfo.seriesNameMode`.

Таблица 17 – Режимы автоматического определения меток диаграмм

Значение	Описание
`ChartLabelsDetectionMode.Unknown`	Неопределенный тип
`ChartLabelsDetectionMode.FirstRow`	Метка на первой строке
`ChartLabelsDetectionMode.FirstColumn`	Метка на первой колонке
`ChartLabelsDetectionMode.NoLabels`	Не отрисовывать метки

Пример

```
CellRange cellRange = sheetDocumentPage.getCellRange("B3:C4");
ChartLabelsInfo chartLabelsInfo = new
ChartLabelsInfo(ChartLabelsDetectionMode.FirstColumn,
ChartLabelsDetectionMode.FirstRow, false);
chart.applySettings(cellRange, null, "Title", chartLabelsInfo);
```

6.26 Класс ChartLabelsInfo

Класс `ChartLabelsInfo` описывает настройки автоматического определения меток диаграммы. Используется в методах `Chart.applySettings()` и `Chart.getChartLabels()`.

Конструктор

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

Параметры

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

Таблица 18 – Описание полей класса `ChartLabelsInfo`

Поле	Тип	Описание
`ChartLabelsInfo.categoriesMode`	`ChartLabelsDetectionMode`	Режим автоматического определения меток для категорий
`ChartLabelsInfo.seriesNameMode`	`ChartLabelsDetectionMode`	Режим автоматического определения меток для серий
`ChartLabelsInfo.isOneColumnRowChart`	`bool`	Поле содержит `true`, если диапазон диаграммы содержит

Поле	Тип	Описание
		только одну строку или одну колонку

Примеры

```
ChartLabelsInfo chartInfo = new
ChartLabelsInfo(ChartLabelsDetectionMode.FirstRow,
ChartLabelsDetectionMode.NoLabels, false);
```

```
Chart chart = charts.getChart(0);
ChartLabelsInfo chartLabelsInfo = chart.getChartLabels();
Console.WriteLine(chartLabelsInfo.seriesNameMode);
Console.WriteLine(chartLabelsInfo.categoriesMode);
```

6.27 Класс ChartRangeInfo

Класс `ChartRangeInfo` описывает серию диаграммы. Используется в методе `Chart.getRange()`.

Конструктор

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

Таблица 19 – Описание полей класса `ChartRangeInfo`

Поле	Тип	Описание
`ChartRangeInfo.cellRange`	`CellRange`	Диапазон ячеек диаграммы
`ChartRangeInfo.rangeColor`	`ColorRGBA`	Цвет для отрисовки серии
`ChartRangeInfo.isHidden`	`bool`	Задает видимость серии диаграммы
`ChartRangeInfo.rangeType`	`ChartRangeType`	Тип диапазона диаграммы

Пример

```
Chart chart = charts.getChart(0);
ChartRangeInfo chartRangeInfo = chart.getRange(0);
Console.WriteLine(chartRangeInfo.rangeColor);
Console.WriteLine(chartRangeInfo.rangeType);
```

6.28 Перечисление ChartRangeType

Перечисление `ChartRangeType` содержит типы диапазонов исходных данных диаграммы. Используется в поле `ChartRangeInfo.rangeType`.

Таблица 20 – Типы диапазонов

Значение	Описание
`ChartRangeType.Series`	Серии
`ChartRangeType.SeriesName`	Имена серий
`ChartRangeType.Categories`	Области
`ChartRangeType.DataPoint`	Разметка данных

Пример

```
Chart chart = charts.getChart(0);
ChartRangeInfo chartRangeInfo = chart.getRange(0);
Console.WriteLine(chartRangeInfo.rangeType);
```

6.29 Класс Charts

Класс `Charts` обеспечивает доступ к списку диаграмм табличного документа. Доступ к списку диаграмм осуществляется с помощью метода `Table.getCharts()`.

Пример получения списка диаграмм

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Console.WriteLine(charts.getChartsCount());
}
```

6.29.1 Метод Charts.addChart

Метод добавляет новую диаграмму на лист.

Вызов

```
public Chart addChart(CellRangePosition dataRange, ChartType type, RectU
rect)
```

Параметры

– `dataRange`: диапазон ячеек с исходными данными для диаграммы, тип `CellRangePosition`.

– `type`: тип диаграммы, тип `ChartType`.

– `rect`: область расположения диаграммы, тип `RectU`.

Возвращает

– новая диаграмма, тип `Chart`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange cellRange = sheet.getCellRange("B2:C18");
Charts charts = sheet.getCharts();
Chart newChart = charts.addChart(cellRange.getTableRange(), ChartType.Doughnut,
new RectU(400, 200, 800, 600));
```

6.29.2 Метод Charts.getChart

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

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Chart chart = charts.getChart(0);
    if (chart != null) {
        Console.WriteLine(chart.getRangeAsString());
    }
}
```

6.29.3 Метод Charts.getChartIndexByDrawingIndex

Метод возвращает индекс диаграммы по индексу отрисовки `drawingIndex`.

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Console.WriteLine(charts.getChartIndexByDrawingIndex(0));
}
```

6.29.4 Метод Charts.getChartsCount

Метод возвращает общее количество диаграмм в табличном документе.

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Console.WriteLine(charts.getChartsCount());
}
```

6.30 Перечисление ChartSeriesDirectionType

Перечисление `ChartSeriesDirectionType` содержит направления серий диаграмм. Используется в методах `Chart.applySettings()` и `Chart.getDirectionType()`.

Таблица 21 – Направления серий диаграмм

Значение	Описание
`ChartSeriesDirectionType.Unknown`	Неопределенный тип
`ChartSeriesDirectionType.ByRow`	Серии направлены по строкам
`ChartSeriesDirectionType.ByColumn`	Серии направлены по колонкам

Пример

```
Chart chart = charts.getChart(0);
ChartSeriesDirectionType chartSeriesDirectionType = chart.getDirectionType();
Console.WriteLine(chartSeriesDirectionType);
```

6.31 Перечисление ChartType

Перечисление `ChartType` содержит все поддерживаемые типы диаграмм. Используется в методах `Charts.addChart()`, `Chart.getType()` и `Chart.setType()`.

Таблица 22 – Типы диаграмм

Значение	Описание
`ChartType.Unknown`	Неопределенный тип
`ChartType.Bar`	Линейчатая диаграмма с группировкой
`ChartType.BarStacked`	Линейчатая диаграмма с накоплением
`ChartType.BarPercentStacked`	Линейчатая нормированная диаграмма с накоплением
`ChartType.Column`	Гистограмма с группировкой
`ChartType.ColumnStacked`	Гистограмма с накоплением
`ChartType.ColumnPercentStacked`	Нормированная гистограмма с накоплением
`ChartType.Line`	Стандартный график
`ChartType.LineStacked`	График с накоплением
`ChartType.LinePercentStacked`	Нормированный график с накоплением
`ChartType.LineWithMarker`	Стандартный график с маркерами
`ChartType.LineWithMarkerStacked`	График с накоплением и маркерами
`ChartType.LineWithMarkerPercentStacked`	Нормированный график с накоплением и маркерами
`ChartType.Area`	Стандартная диаграмма с областями
`ChartType.AreaStacked`	Диаграмма с областями с накоплением
`ChartType.AreaPercentStacked`	Нормированная диаграмма с областями с накоплением
`ChartType.Pie`	Круговая диаграмма
`ChartType.PieExploded`	Круговая диаграмма с отделенными секторами
`ChartType.Scatter`	Диаграмма рассеяния
`ChartType.Doughnut`	Кольцевая диаграмма

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Chart chart = charts.getChart(0);
    ChartType chartType = chart.getType();
    Console.WriteLine(chartType);
}
```

6.32 Класс CheckBoxControl

Представляет собой элемент управления "Флажок" в документе. Является наследником класса `ContentControl`. Методы `CheckBoxControl.getValue()` и `CheckBoxControl.setValue(bool)` позволяют получить и задать значение этого элемента управления.

Пример

```
ContentControls controls = document.getContentControls();
CheckBoxControl checkBox = controls.findByTitle("check1").toCheckBox();
checkBox.setValue(!checkBox.getValue());
```

6.33 Класс Color

Класс `Color` представляет либо цветовой объект RGBA, либо заданные цвета идентификатора темы. В качестве параметров конструктора используются объекты `ColorRGBA`, `ThemeColorID`.

Пример

```
Color rgbaColor = new Color(new ColorRGBA(255, 0, 0, 255));
Color themeColor = new Color(ThemeColorID.Text1);
```

6.33.1 Метод Color.getRGBAColor

Метод возвращает цвет `ColorRGBA`.

Пример

```
Color color = new Color(new ColorRGBA(255, 0, 0, 255));
ColorRGBA rgbaColor = color.getRGBAColor();
```

```
if (rgbaColor != null) {
    Console.WriteLine(rgbaColor.r);
}
```

6.33.2 Метод Color.getThemeColorID

Метод возвращает цвет идентификатора темы ThemeColorID.

Пример

```
Color color = new Color(new ColorRGBA(255, 0, 0, 255));
ThemeColorID? themeColorId = color.getThemeColorID();
if (themeColorId == null && themeColorId.HasValue) {
    Console.WriteLine(themeColorId.Value);
}
```

6.33.3 Метод Color.getTransforms

Метод возвращает примененные к цвету трансформации.

Вызов

```
public ColorTransforms getTransforms()
```

Возвращает

– трансформации цвета, тип ColorTransforms.

– null, если трансформации не заданы.

Пример

```
ColorTransforms transforms = color.getTransforms();
foreach (ColorTransform transform in transforms) {
    ColorTransformType type = transform.getType();
}
```

6.33.4 Метод Color.setTransforms

Метод применяет заданные трансформации к текущему цвету.

Вызов

```
public void setTransforms(ColorTransforms transforms)
```

Параметры

– `transforms`: трансформации цвета, тип `ColorTransforms`.

Пример

```
Color color = new Color(ThemeColorID.FollowedHyperlink);
ColorTransforms colorTransforms = new ColorTransforms();
colorTransforms.addTransform(ColorTransformType.RedModulation,
1.5f).addTransform(ColorTransformType.Tint, 0.3f);
color.setTransforms(colorTransforms);
```

6.34 Класс ColorRGBA

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

Конструкторы

```
ColorRGBA()
ColorRGBA(byte r, byte g, byte b, byte a)
```

Таблица 23 – Описание полей класса `ColorRGBA`

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

Примеры использования

```
ColorRGBA rgba = new ColorRGBA();
rgba.r = 0;
rgba.g = 0;
rgba.b = 255;
rgba.a = 200;
// r: 0, g: 0, b: 255, a: 200
```

```
Console.WriteLine("r:" + rgba.r + ", g:" + rgba.g + ", b:" + rgba.b + ", a:" +
rgba.a);
```

```
rgba = new ColorRGBA(55, 146, 179, 200);
// r: 55, g: 146, b: 179, a: 200
Console.WriteLine("r:" + rgba.r + ", g:" + rgba.g + ", b:" + rgba.b + ", a:" +
rgba.a);
```

```
LineProperties lineProps = new LineProperties();
lineProps.color = new Color(rgba);
```

6.35 Класс ColorScaleConditionalFormatOperator

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

Конструктор

```
public
ColorScaleConditionalFormatOperator(ConditionalFormatColorScaleEntries entries)
```

Пример

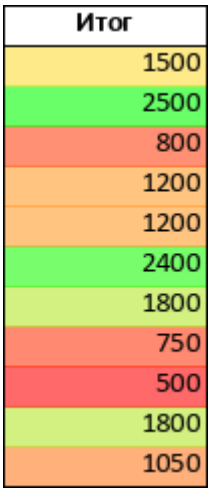


Рисунок 41 – Пример создания правила "Цветовая шкала" с тремя пороговыми значениями

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

var entries = new ConditionalFormatColorScaleEntries();
```

```
var value1 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Min, "0", false);
var entry1 = new ConditionalFormatColorScaleEntry(value1, new Color(new
ColorRGBA(255, 0, 0, 150)));
entries.addEntry(entry1);
var value2 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "50",
false);
var entry2 = new ConditionalFormatColorScaleEntry(value2, new Color(new
ColorRGBA(255, 220, 56, 150)));
entries.addEntry(entry2);
var value3 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Max, "0", false);
var entry3 = new ConditionalFormatColorScaleEntry(value3, new Color(new
ColorRGBA(0, 255, 0, 150)));
entries.addEntry(entry3);

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

var colorScaleOperator =
DocumentAPI.createColorScaleConditionalFormatOperator(entries);

var colorScaleRule = new ConditionalFormatRule();
colorScaleRule.setRange(cellRangePosition);
colorScaleRule.setOperator(colorScaleOperator);
rules.addRule(colorScaleRule);
```

6.35.1 Метод ColorScaleConditionalFormatOperator.getEntries

Метод возвращает набор правил применения цветов для текущего оператора.

Вызов

```
public ConditionalFormatColorScaleEntries getEntries()
```

Возвращает

– набор правил применения цветов, тип ConditionalFormatColorScaleEntries.

6.35.2 Метод ColorScaleConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.35.3 Метод ColorScaleConditionalFormatOperator.setEntries

Метод задает набор правил применения цветов для текущего оператора.

Вызов

```
public void setEntries(ConditionalFormatColorScaleEntries entries)
```

Параметры

– entries: набор правил применения цветов, тип ConditionalFormatColorScaleEntries.

6.36 Класс ColorTransform

Класс ColorTransform представляет собой трансформацию цвета для объектов ColorRGBA и Color. Является элементом коллекции ColorTransforms.

Пример

```
ColorTransforms transforms = color.getTransforms();
foreach (ColorTransform transform in transforms) {
    ColorTransformType type = transform.getType();
}
```

6.36.1 Метод ColorTransform.getType

Метод возвращает тип текущей трансформации цвета.

Вызов

```
public ColorTransformType getType()
```

Возвращает

– тип трансформации цвета, тип ColorTransformType.

6.36.2 Метод ColorTransform.getValue

Метод возвращает значение текущей трансформации цвета.

Вызов

```
public float? getValue()
```

Возвращает

– значение трансформации, тип `float`.

– `null`, если значение не задано.

6.37 Класс ColorTransforms

Класс `ColorTransforms` представляет собой коллекцию трансформаций цвета для объектов `ColorRGBA` и `Color` (см. методы `Color.setTransforms` и `Color.getTransforms`). Элементами коллекции являются объекты класса `ColorTransform`.

Примеры

```
ColorTransforms colorTransforms = new ColorTransforms();
colorTransforms.addTransform(ColorTransformType.Tint,
0.7f).addTransform(ColorTransformType.Complement);
ColorRGBA newColor = colorTransforms.apply(new ColorRGBA(55, 146, 179, 200));
```

```
Color color = new Color(ThemeColorID.FollowedHyperlink);
ColorTransforms colorTransforms = new ColorTransforms();
colorTransforms.addTransform(ColorTransformType.RedModulation,
1.5f).addTransform(ColorTransformType.Tint, 0.3f);
color.setTransforms(colorTransforms);
```

6.37.1 Метод ColorTransforms.addTransform

Метод добавляет новую трансформацию цвета в коллекцию.

Вызов

```
public ColorTransforms addTransform(ColorTransformType transformType,
float? transformValue)
```

Параметры

– `transformType`: тип трансформации, тип `ColorTransformType`.

– `transformValue`: (необязательный) значение трансформации, тип `float`. Допустимые значения для каждого типа трансформации перечислены в описании `ColorTransformType`.

Возвращает

– текущая коллекция трансформаций цвета, тип `ColorTransforms`.

Пример

```
ColorTransforms colorTransforms = new ColorTransforms();
colorTransforms.addTransform(ColorTransformType.Tint,
0.7f).addTransform(ColorTransformType.Complement);
ColorRGBA newColor = colorTransforms.apply(new ColorRGBA(55, 146, 179, 200));
```

6.37.2 Метод ColorTransforms.apply

Метод применяет текущие трансформации к заданному цвету.

Вызов

```
public ColorRGBA apply(ColorRGBA color)
```

Параметры

– `color`: исходный цвет, тип `ColorRGBA`.

Возвращает

– цвет после применения трансформаций, тип `ColorRGBA`.

Пример

```
ColorTransforms colorTransforms = new ColorTransforms();
colorTransforms.addTransform(ColorTransformType.Shade, 0.1f);
colorTransforms.addTransform(ColorTransformType.Inverse);
ColorRGBA newColor = colorTransforms.apply(new ColorRGBA(55, 146, 179, 200));
```

6.37.3 Метод ColorTransforms.clearTransforms

Метод очищает коллекцию трансформаций цвета.

Вызов

```
public void clearTransforms()
```

6.38 Перечисление ColorTransformType

Перечисление `ColorTransformType` содержит типы трансформаций цвета. Используется в методах `ColorTransforms.addTransform()` и `ColorTransform.getType()`.

Таблица 24 – Типы трансформаций цвета

Значение	Описание	Значения параметра
`ColorTransformType.Alpha`	Задает значение Alpha-канала (прозрачности) для модели RGBA. Значение задается в процентах от 255.	0,0-1,0
`ColorTransformType.Red`	Задает значение красного канала для модели RGBA. Значение задается в процентах от 255.	0,0-1,0
`ColorTransformType.Green`	Задает значение зеленого канала для модели RGBA. Значение задается в процентах от 255.	0,0-1,0
`ColorTransformType.Blue`	Задает значение синего канала для модели RGBA. Значение задается в процентах от 255.	0,0-1,0
`ColorTransformType.AlphaModulation`	Умножает значение Alpha-канала на заданный коэффициент. Результат трансформации не может превысить 255.	>=0,0
`ColorTransformType.RedModulation`	Умножает значение красного канала на заданный коэффициент. Результат трансформации не может превысить 255.	>=0,0
`ColorTransformType.GreenModulation`	Умножает значение зеленого канала на заданный коэффициент. Результат трансформации не может превысить 255.	>=0,0
`ColorTransformType.BlueModulation`	Умножает значение синего канала на заданный коэффициент. Результат трансформации не может превысить 255.	>=0,0
`ColorTransformType.AlphaOffset`	Увеличивает или уменьшает значение Alpha-канала на заданный процент.	>=-1,0

Значение	Описание	Значения параметра
	Результат трансформации не может быть меньше 0 и больше 255.	
`ColorTransformType.RedOffset`	Увеличивает или уменьшает значение красного канала на заданный процент. Результат трансформации не может быть меньше 0 и больше 255.	>=-1,0
`ColorTransformType.GreenOffset`	Увеличивает или уменьшает значение зеленого канала на заданный процент. Результат трансформации не может быть меньше 0 и больше 255.	>=-1,0
`ColorTransformType.BlueOffset`	Увеличивает или уменьшает значение синего канала на заданный процент. Результат трансформации не может быть меньше 0 и больше 255.	>=-1,0
`ColorTransformType.Hue`	Задает положительный угол поворота на окружности цветового тона (Hue) в модели HSL.	0,0-360,0
`ColorTransformType.HueModulation`	Умножает значение тона (Hue) в модели HSL на заданный коэффициент. Результат трансформации не может привысить 360.	>=0,0
`ColorTransformType.HueOffset`	Увеличивает или уменьшает угол поворота на окружности цветового тона (Hue) на заданное количество градусов. Результат трансформации не может быть меньше 0 и больше 360.	-360,0-360,0
`ColorTransformType.Luminance`	Задает значение канала светлоты (Luminance) в модели HSL. Значение задается в процентах.	0,0-1,0
`ColorTransformType.LuminanceModulation`	Умножает значение канала светлоты (Luminance) в модели HSL на заданный коэффициент. Результат трансформации не может привысить 100.	>=0,0

Значение	Описание	Значения параметра
`ColorTransformType.LuminanceOffset`	Увеличивает или уменьшает значение канала светлоты (Luminance) на заданный процент. Результат трансформации не может быть меньше 0 и больше 100.	>=-1,0
`ColorTransformType.Saturation`	Задает значение канала насыщенности (Saturation) в модели HSL. Значение задается в процентах.	0,0-1,0
`ColorTransformType.SaturationModulation`	Умножает значение канала насыщенности (Saturation) в модели HSL на заданный коэффициент. Результат трансформации не может привысить 100.	>=0,0
`ColorTransformType.SaturationOffset`	Увеличивает или уменьшает значение канала насыщенности (Saturation) на заданный процент. Результат трансформации не может быть меньше 0 и больше 100.	>=-1,0
`ColorTransformType.Shade`	Умножает значение каждого канала (R, G, B) на заданный параметр, делая цвет темнее (0,0 - полностью черный, 1,0 - исходный цвет).	0,0-1,0
`ColorTransformType.Tint`	Делает цвет светлее, смешивая его с белым. Параметр задает процент осветления (0,0 - исходный цвет, 1,0 - полностью белый).	0,0-1,0
`ColorTransformType.Complement`	Сдвигает цветовой тон (Hue) на 180° (Меняет цвет на противоположный на цветовом круге).	-
`ColorTransformType.Inverse`	Инвертирует каждый RGB канал цвета.	-
`ColorTransformType.Gamma`	Преобразует цвет из линейного пространства в пространство sRGB.	-
`ColorTransformType.InverseGamma`	Преобразует цвет из гамма-пространства (sRGB) в линейное пространство.	-

Значение	Описание	Значения параметра
`ColorTransformType.Gray`	Преобразует цвет в оттенок серого, сохраняя яркость.	-

6.39 Класс Comment

Класс `Comment` предоставляет доступ к следующим свойствам комментария:

- диапазон текста `Range`, который описывает комментарий;
- текст комментария;
- информация о комментарии `TrackedChangeInfo`;
- признак того, что комментарий принят;
- список ответов на комментарий `Comments`.

6.39.1 Метод Comment.getInfo

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

Пример

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Console.WriteLine(comment.getInfo().author);
    }
}
```

6.39.2 Метод Comment.getRange

Метод возвращает диапазон документа `Range`, которому соответствует комментарий.

Пример

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
```

```
    {
        Range commentRange = comment.getRange();
        Console.WriteLine(commentRange.extractText());
    }
}
```

6.39.3 Метод Comment.getReplies

Метод предоставляет доступ к ответам на комментарии. Ответы представляются классом Comments так же, как и сами комментарии документа.

Пример

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Comments replies = comment.getReplies();
        CommentsEnumerator repliesEnumerator = replies.GetEnumerator();
        foreach (var reply in repliesEnumerator)
        {
            Console.WriteLine(reply.isResolved());
        }
    }
}
```

6.39.4 Метод Comment.getText

Метод возвращает текст комментария.

Пример

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Console.WriteLine(comment.getText());
```

```
    }
}
```

6.39.5 Метод Comment.isResolved

Метод возвращает значение `true`, если комментарий принят.

Пример

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Console.WriteLine(comment.isResolved());
    }
}
```

6.40 Класс Comments

Класс `Comments` содержит коллекцию комментариев диапазона (см. Рисунок 2).

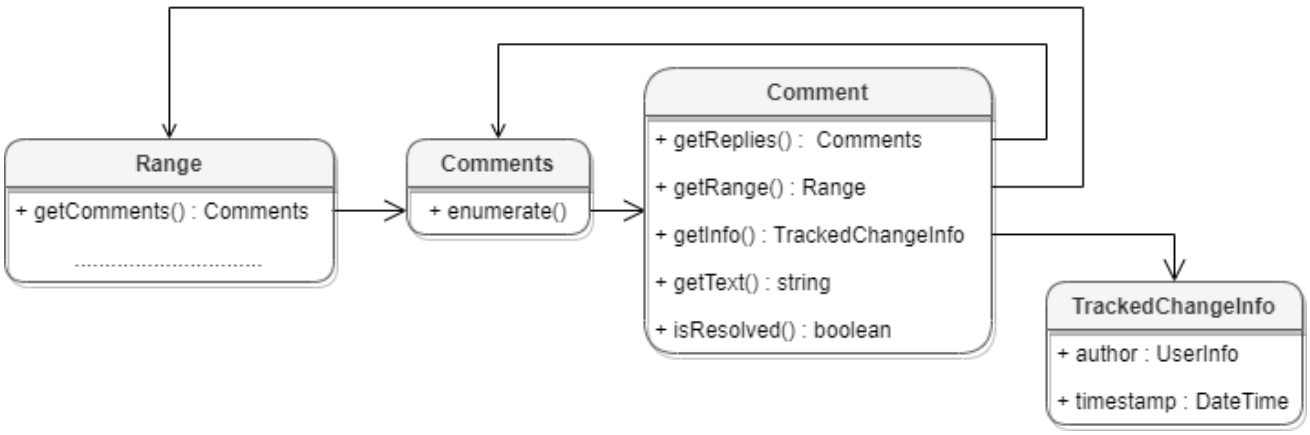


Рисунок 42 – Объектная модель классов для работы с комментариями

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

Пример

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

6.40.1 Метод Comments.GetEnumerator

Метод используется для перечисления комментариев диапазона.

Пример

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
foreach (var comment in commentsEnumerator)
{
    Console.WriteLine(comment.getInfo().author);
}
```

6.41 Перечисление ConditionalFormatAboveAverageCondition

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

Таблица 25 – Условия применения форматирования

Значение	Описание
`ConditionalFormatAboveAverageCondition.Above`	Выше среднего
`ConditionalFormatAboveAverageCondition.EqualAbove`	Выше среднего или равно
`ConditionalFormatAboveAverageCondition.Below`	Ниже среднего
`ConditionalFormatAboveAverageCondition.EqualBelow`	Ниже среднего или равно

6.42 Перечисление ConditionalFormatBinaryCondition

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

Таблица 26 – Условия применения форматирования

Значение	Описание
`ConditionalFormatBinaryCondition.Between`	Между, включительно
`ConditionalFormatBinaryCondition.NotBetween`	Не между

6.43 Класс ConditionalFormatCellStyle

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

Таблица 27 – Описание полей класса `ConditionalFormatCellStyle`

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

6.44 Класс ConditionalFormatColorScaleEntries

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

Пример

```
var entries = new ConditionalFormatColorScaleEntries();
var value1 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Min, "0", false);
```

```
var entry1 = new ConditionalFormatColorScaleEntry(value1, new Color(new
ColorRGBA(255, 0, 0, 150)));
entries.addEntry(entry1);
// ...
var colorScaleOperator = new ColorScaleConditionalFormatOperator(entries);
```

6.44.1 Метод ConditionalFormatColorScaleEntries.addEntry

Метод добавляет правило применения цвета в текущий набор.

Вызов

```
public void addEntry(ConditionalFormatColorScaleEntry entry)
```

Параметры

– `entry`: правило применения цвета, тип `ConditionalFormatColorScaleEntry`.

6.44.2 Метод ConditionalFormatColorScaleEntries.getEntriesCount

Метод возвращает количество правил в текущем наборе.

Вызов

```
public uint getEntriesCount()
```

Возвращает

– количество правил, тип `uint`.

6.44.3 Метод ConditionalFormatColorScaleEntries.getEntry

Метод возвращает правило по его индексу.

Вызов

```
public ConditionalFormatColorScaleEntry getEntry(uint index)
```

Параметры

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

Возвращает

– правило применения цвета, тип `ConditionalFormatColorScaleEntry`.

6.44.4 Метод ConditionalFormatColorScaleEntries.setEntry

Метод заменяет правило под заданным индексом.

Вызов

```
public void setEntry(uint index, ConditionalFormatColorScaleEntry entry)
```

Параметры

– `index`: индекс правила для замены, индексация начинается с нуля, тип `uint`.

– `entry`: новое правило применения цвета, тип `ConditionalFormatColorScaleEntry`.

6.45 Класс ConditionalFormatColorScaleEntry

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

Конструктор

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

Пример

```
var entries = new ConditionalFormatColorScaleEntries();
var value1 = new ConditionalFormatValueObject(ConditionalFormatValueObjectType.Min, "0", false);
var entry1 = new ConditionalFormatColorScaleEntry(value1, new Color(new ColorRGBA(255, 0, 0, 150)));
entries.addEntry(entry1);
```

6.45.1 Метод ConditionalFormatColorScaleEntry.getColor

Метод возвращает цвет, который используется в текущем правиле.

Вызов

```
public Color getColor()
```

Возвращает

– используемый цвет, тип `Color`.

6.45.2 Метод ConditionalFormatColorScaleEntry.getValueObject

Метод возвращает пороговое значение для текущего правила.

Вызов

```
public ConditionalFormatValueObject getValueObject()
```

Возвращает

– пороговое значение для текущего правила, тип `ConditionalFormatValueObject`.

6.45.3 Метод ConditionalFormatColorScaleEntry.setColor

Метод позволяет задать цвет для текущего правила.

Вызов

```
public void setColor(Color color)
```

Параметры

– `color`: цвет для правила, тип `Color`.

6.45.4 Метод ConditionalFormatColorScaleEntry.setValueObject

Метод позволяет задать пороговое значение для текущего правила.

Вызов

```
public void setValueObject(ConditionalFormatValueObject valueObject)
```

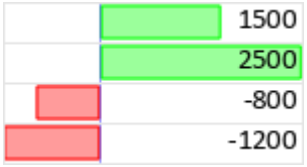
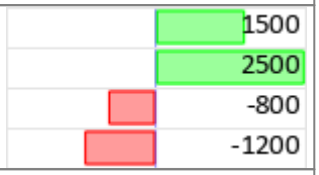
Параметры

– `valueObject`: пороговое значение для текущего правила, тип `ConditionalFormatValueObject`.

6.46 Перечисление ConditionalFormatDataBarAxisPosition

Перечисление `ConditionalFormatDataBarAxisPosition` содержит варианты расположения оси между положительными и отрицательными значениями гистограммы. Используется в поле `ConditionalFormatDataBarParams.axisPosition`.

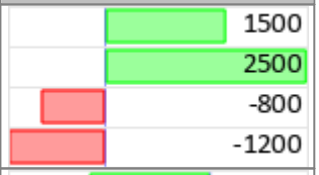
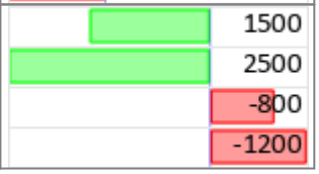
Таблица 28 – Варианты расположения оси

Значение	Описание	Изображение
`ConditionalFormatDataBarAxisPosition.Auto`	Ось расположена в зависимости от соотношения максимального положительного и отрицательного значений	
`ConditionalFormatDataBarAxisPosition.Middle`	Ось расположена посередине ячейки	
`ConditionalFormatDataBarAxisPosition.None`	Ось отсутствует, положительные и отрицательные значения направлены в одну сторону	

6.47 Перечисление ConditionalFormatDataBarDirection

Перечисление `ConditionalFormatDataBarDirection` содержит варианты направления гистограммы в ячейке. Используется в поле `ConditionalFormatDataBarParams.direction`.

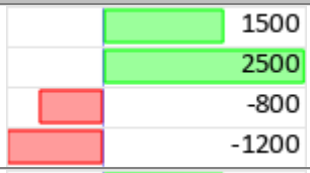
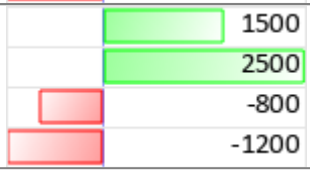
Таблица 29 – Варианты направления гистограммы

Значение	Описание	Изображение
`ConditionalFormatDataBarDirection.LeftToRight`	Гистограмма для положительных значений рисуется слева направо	
`ConditionalFormatDataBarDirection.RightToLeft`	Гистограмма для положительных значений рисуется справа налево	

6.48 Перечисление ConditionalFormatDataBarFillType

Перечисление `ConditionalFormatDataBarDirection` содержит варианты заливки гистограммы в ячейке. Используется в поле `ConditionalFormatDataBarParams.fillType`.

Таблица 30 – Варианты заливки гистограммы

Значение	Описание	Изображение
`ConditionalFormatDataBarFillType.Solid`	Сплошная заливка	
`ConditionalFormatDataBarFillType.Gradient`	Градиент	

6.49 Класс ConditionalFormatDataBarParams

Класс `ConditionalFormatDataBarParams` предназначен для настройки параметров гистограммы условного форматирования. Используется при создании экземпляра класса `DataBarConditionalFormatOperator`.

Конструктор

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

Таблица 31 – Описание полей класса `ConditionalFormatDataBarParams`

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

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

6.50 Класс ConditionalFormatDocumentRules

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

Пример

```
ConditionalFormatDocumentRules rules = document.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.AboveAverage)
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule.getOperator());
}
```

6.50.1 Метод ConditionalFormatDocumentRules.removeAllRules

Метод удаляет все правила условного форматирования из коллекции.

Вызов

```
public void removeAllRules()
```

Пример

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

6.51 Перечисление ConditionalFormatIconSet

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

Таблица 32 – Наборы значков условного форматирования

Значение	Набор значков
`ConditionalFormatIconSet.ThreeArrows`	
`ConditionalFormatIconSet.ThreeArrowsGray`	
`ConditionalFormatIconSet.ThreeFlags`	
`ConditionalFormatIconSet.ThreeTrafficLights1`	
`ConditionalFormatIconSet.ThreeTrafficLights2`	
`ConditionalFormatIconSet.ThreeSigns`	
`ConditionalFormatIconSet.ThreeSymbols`	
`ConditionalFormatIconSet.ThreeSymbols2`	
`ConditionalFormatIconSet.ThreeStars`	
`ConditionalFormatIconSet.ThreeTriangles`	
`ConditionalFormatIconSet.ThreeSmilies`	Не поддерживается
`ConditionalFormatIconSet.ThreeColorSmilies`	Не поддерживается
`ConditionalFormatIconSet.FourArrows`	

Значение	Набор значков
ConditionalFormatIconSet.FourArrowsGray	
ConditionalFormatIconSet.FourRedToBlack	
ConditionalFormatIconSet.FourRating	
ConditionalFormatIconSet.FourTrafficLights	
ConditionalFormatIconSet.FiveArrows	
ConditionalFormatIconSet.FiveArrowsGray	
ConditionalFormatIconSet.FiveRating	
ConditionalFormatIconSet.FiveQuarters	
ConditionalFormatIconSet.FiveBoxes	
ConditionalFormatIconSet.NoIcons	Пустой набор

Пример

```
var entries = new ConditionalFormatIconSetEntries();
var value1 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "0",
false);
var entry1 = new ConditionalFormatIconSetEntry(value1,
ConditionalFormatIconSet.FiveBoxes, 0);
entries.addEntry(entry1);
```

6.52 Класс ConditionalFormatIconSetEntries

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

Пример

```
var entries = new ConditionalFormatIconSetEntries();
var value1 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "0",
```

```
false);
var entry1 = new ConditionalFormatIconSetEntry(value1,
ConditionalFormatIconSet.FiveBoxes, 0);
entries.addEntry(entry1);
// ...
var iconSetOperator = new IconSetConditionalFormatOperator(entries, true);
```

6.52.1 Метод ConditionalFormatIconSetEntries.addEntry

Метод добавляет правило отображения значка в текущий набор.

Вызов

```
public void addEntry(ConditionalFormatIconSetEntry entry)
```

Параметры

– `entry`: правило отображения значка, тип `ConditionalFormatIconSetEntry`.

6.52.2 Метод ConditionalFormatIconSetEntries.getEntriesCount

Метод возвращает количество правил в текущем наборе.

Вызов

```
public uint getEntriesCount()
```

Возвращает

– количество правил, тип `uint`.

6.52.3 Метод ConditionalFormatIconSetEntries.getEntry

Метод возвращает правило по его индексу.

Вызов

```
public ConditionalFormatIconSetEntry getEntry(uint index)
```

Параметры

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

Возвращает

– правило отображения значка, тип `ConditionalFormatIconSetEntry`.

6.52.4 Метод ConditionalFormatIconSetEntries.setEntry

Метод заменяет правило под заданным индексом.

Вызов

```
public void setEntry(uint index, ConditionalFormatIconSetEntry entry)
```

Параметры

– `index`: индекс правила для замены, индексация начинается с нуля, тип `uint`.

– `entry`: новое правило отображения значка, тип `ConditionalFormatIconSetEntry`.

6.53 Класс ConditionalFormatIconSetEntry

Класс `ConditionalFormatIconSetEntry` представляет собой правило отображения значка для условного форматирования. Используется в методах `ConditionalFormatIconSetEntries.addEntry()`, `ConditionalFormatIconSetEntries.getEntry()` и `ConditionalFormatIconSetEntries.setEntry()`.

Конструкторы

```
public ConditionalFormatIconSetEntry()
```

```
public ConditionalFormatIconSetEntry(ConditionalFormatValueObject valueObject, ConditionalFormatIconSet iconSet, byte iconIndex)
```

Пример

```
var entries = new ConditionalFormatIconSetEntries();
var value1 = new ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "0", false);
var entry1 = new ConditionalFormatIconSetEntry(value1, ConditionalFormatIconSet.FiveBoxes, 0);
entries.addEntry(entry1);
```

6.53.1 Метод ConditionalFormatIconSetEntry.getIconIndex

Метод возвращает индекс используемого значка из текущего набора.

Вызов

```
public byte getIconIndex()
```

Возвращает

– индекс значка в наборе, индексация начинается с нуля, тип `byte`.

6.53.2 Метод ConditionalFormatIconSetEntry.getIconSet

Метод возвращает набор, значок из которого используется в текущем правиле.

Вызов

```
public ConditionalFormatIconSet getIconSet()
```

Возвращает

– набор значков, тип `ConditionalFormatIconSet`.

6.53.3 Метод ConditionalFormatIconSetEntry.getValueObject

Метод возвращает пороговое значение для текущего правила.

Вызов

```
public ConditionalFormatValueObject getValueObject()
```

Возвращает

– пороговое значение для текущего правила, тип `ConditionalFormatValueObject`.

6.53.4 Метод ConditionalFormatIconSetEntry.setIconIndex

Метод позволяет задать значок по его индексу в наборе.

Вызов

```
public void setIconIndex(byte iconIndex)
```

Параметры

– `iconIndex`: индекс значка в наборе, индексация начинается с нуля, тип `byte`.

6.53.5 Метод ConditionalFormatIconSetEntry.setIconSet

Метод позволяет задать набор значков для использования в текущем правиле.

Вызов

```
public void setIconSet(ConditionalFormatIconSet iconSet)
```

Параметры

– `iconSet`: набор значков, тип `ConditionalFormatIconSet`.

6.53.6 Метод ConditionalFormatIconSetEntry.setValueObject

Метод позволяет задать пороговое значение для текущего правила.

Вызов

```
public void setValueObject(ConditionalFormatValueObject valueObject)
```

Параметры

– `valueObject`: пороговое значение для текущего правила, тип `ConditionalFormatValueObject`.

6.54 Перечисление ConditionalFormatNullaryCondition

Перечисление `ConditionalFormatNullaryCondition` содержит условия применения форматирования для правил без параметров. Используется в методе `NullaryConditionalFormatOperator.getCondition()`.

Таблица 33 – Условия применения форматирования

Значение	Описание
`ConditionalFormatNullaryCondition.IsBlank`	Пустая ячейка
`ConditionalFormatNullaryCondition.IsNotBlank`	Непустая ячейка
`ConditionalFormatNullaryCondition.IsError`	Ячейка с ошибкой формулы
`ConditionalFormatNullaryCondition.IsNotError`	Ячейка без ошибки формулы
`ConditionalFormatNullaryCondition.Yesterday`	Вчера
`ConditionalFormatNullaryCondition.Today`	Сегодня
`ConditionalFormatNullaryCondition.Tomorrow`	Завтра
`ConditionalFormatNullaryCondition.InLast7Days`	Последние 7 дней
`ConditionalFormatNullaryCondition.LastWeek`	Прошлая неделя
`ConditionalFormatNullaryCondition.ThisWeek`	Эта неделя
`ConditionalFormatNullaryCondition.NextWeek`	Следующая неделя
`ConditionalFormatNullaryCondition.LastMonth`	Прошлый месяц

Значение	Описание
`ConditionalFormatNullaryCondition.ThisMonth`	Этот месяц
`ConditionalFormatNullaryCondition.NextMonth`	Следующий месяц

6.55 Класс ConditionalFormatOperator

Класс `ConditionalFormatOperator` представляет собой базовый класс для операторов условного форматирования. Используется в методах `ConditionalFormatRule.getOperator()`, `ConditionalFormatRule.setOperator()`, `ConditionalFormatRuleProxy.getOperator()` и `ConditionalFormatRuleProxy.setOperator()`.

Наследники

- `AboveAverageConditionalFormatOperator`
- `BinaryConditionalFormatOperator`
- `ColorScaleConditionalFormatOperator`
- `DataBarConditionalFormatOperator`
- `IconSetConditionalFormatOperator`
- `NullaryConditionalFormatOperator`
- `TextConditionalFormatOperator`
- `TopBottomConditionalFormatOperator`
- `UnaryConditionalFormatOperator`
- `UniquenessConditionalFormatOperator`

6.55.1 Метод ConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public virtual ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип `ConditionalFormatOperatorType`.

6.56 Перечисление ConditionalFormatOperatorType

Перечисление `ConditionalFormatOperatorType` содержит типы операторов условного форматирования. Используется в методах `ConditionalFormatOperator.getType()` и `ConditionalFormatRuleProxy.getType()`.

Таблица 34 – Типы операторов условного форматирования

Значение	Описание
`ConditionalFormatOperatorType.Nullary`	Оператор без аргументов
`ConditionalFormatOperatorType.Unary`	Больше, меньше, равно, не равно
`ConditionalFormatOperatorType.Binary`	Между и не между
`ConditionalFormatOperatorType.Text`	"Текст"
`ConditionalFormatOperatorType.TopBottom`	Наибольшие и наименьшие значения
`ConditionalFormatOperatorType.AboveAverage`	Выше и ниже среднего
`ConditionalFormatOperatorType.Uniqueness`	Уникальные и повторяющиеся значения
`ConditionalFormatOperatorType.IconSet`	"Значки"
`ConditionalFormatOperatorType.ColorScale`	"Цветовая шкала"
`ConditionalFormatOperatorType.DataBar`	"Гистограмма"

6.57 Класс ConditionalFormatRule

Класс `ConditionalFormatRule` представляет собой правило условного форматирования. Используется в методе `ConditionalFormatTableRules.addRule()`.

Конструкторы

```
public ConditionalFormatRule()
```

```
public ConditionalFormatRule(ConditionalFormatOperator conditionalFormatOperator, ConditionalFormatCellStyle cellStyle, CellRangePosition cellRangePosition, bool stopCalculations)
```

```
public ConditionalFormatRule(ConditionalFormatOperator conditionalFormatOperator, ConditionalFormatCellStyle cellStyle, CellRangePositions cellRangePositions, bool stopCalculations)
```

Пример

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

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

var aboveStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(0, 255, 0, 150)));
aboveStyle.cellProperties = cellProperties;

var aboveOperator =
DocumentAPI.createAboveAverageConditionalFormatOperator(ConditionalFormatAboveAve
rageCondition.Above);

var aboveRule = new ConditionalFormatRule(aboveOperator, aboveStyle,
cellRangePosition, false);
rules.addRule(aboveRule);
```

6.57.1 Метод ConditionalFormatRule.getOperator

Метод возвращает оператор условного форматирования.

Вызов

```
public ConditionalFormatOperator getOperator()
```

Возвращает

– оператор условного форматирования, тип `ConditionalFormatOperator`.

6.57.2 Метод ConditionalFormatRule.getRanges

Метод возвращает диапазоны ячеек таблицы, к которым применяется текущее правило.

Вызов

```
public CellRangePositions getRanges()
```

Возвращает

– диапазоны ячеек таблицы, тип `CellRangePositions`.

6.57.3 Метод ConditionalFormatRule.getStopCalculations

Метод позволяет определить, будут ли применяться другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
public bool getStopCalculations()
```

Возвращает

– `true`, если другие правила не применяются к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

6.57.4 Метод ConditionalFormatRule.getStyle

Метод возвращает настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
public ConditionalFormatCellStyle getStyle()
```

Возвращает

– настройки параметров отображения ячеек, тип `ConditionalFormatCellStyle`.

6.57.5 Метод ConditionalFormatRule.getUUID

Метод возвращает уникальный идентификатор для текущего правила.

Вызов

```
public string getUUID()
```

Возвращает

– уникальный идентификатор правила, тип `string`.

6.57.6 Метод ConditionalFormatRule.setOperator

Метод позволяет задать оператор условного форматирования.

Вызов

```
public void setOperator(ConditionalFormatOperator
conditionalFormatOperator)
```

Параметры

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator` или его наследники.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
var entries = new ConditionalFormatIconSetEntries();
// ...
var iconSetOperator = DocumentAPI.createIconSetConditionalFormatOperator(entries,
true);

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

var iconSetRule = new ConditionalFormatRule();
iconSetRule.setOperator(iconSetOperator);
iconSetRule.setRange(cellRangePosition);
rules.addRule(iconSetRule);
```

6.57.7 Метод ConditionalFormatRule.setRange

Метод позволяет задать диапазон ячеек таблицы, к которому будет применено текущее правило.

Вызов

```
public void setRange(CellRangePosition cellRangePosition)
```

Параметры

– `cellRangePosition`: диапазон ячеек таблицы, тип `CellRangePosition`.

6.57.8 Метод ConditionalFormatRule.setRanges

Метод позволяет задать диапазоны ячеек таблицы, к которым будет применено текущее правило.

Вызов

```
public void setRanges(CellRangePositions cellRangePositions)
```

Параметры

– `cellRangePositions`: диапазоны ячеек таблицы, тип `CellRangePositions`.

6.57.9 Метод ConditionalFormatRule.setStopCalculations

Метод позволяет задать, применять ли другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
public void setStopCalculations(bool stopCalculations)
```

Параметры

– `stopCalculations`: `true`, чтобы не применять другие правила к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

6.57.10 Метод ConditionalFormatRule.setStyle

Метод позволяет задать настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
public void setStyle(ConditionalFormatCellStyle style)
```

Параметры

– `style`: настройки параметров отображения ячеек, тип `ConditionalFormatCellStyle`.

6.57.11 Метод ConditionalFormatRule.setUUID

Метод позволяет задать уникальный идентификатор для текущего правила.

Вызов

```
public void setUUID(string uuid)
```

Параметры

– `uuid`: уникальный идентификатор правила, тип `string`.

6.58 Класс ConditionalFormatRuleProxy

Класс `ConditionalFormatRuleProxy` представляет собой правило условного форматирования, как элемент коллекции `ConditionalFormatDocumentRules` или

ConditionalFormatTableRules. Используется в методе ConditionalFormatTableRules.getRule().

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
ConditionalFormatRuleProxy firstRule = rules.getRule(0);
firstRule.remove();
```

6.58.1 Метод ConditionalFormatRuleProxy.getData

Метод возвращает объект ConditionalFormatRule для текущего правила.

Вызов

```
public ConditionalFormatRule getData()
```

Возвращает

– правило условного форматирования, тип ConditionalFormatRule.

6.58.2 Метод ConditionalFormatRuleProxy.getIndex

Метод возвращает индекс правила в коллекции.

Вызов

```
public uint getIndex()
```

Возвращает

– индекс правила в коллекции, индексация начинается с нуля, тип `uint`.

6.58.3 Метод ConditionalFormatRuleProxy.getOperator

Метод возвращает оператор условного форматирования.

Вызов

```
public ConditionalFormatOperator getOperator()
```

Возвращает

– оператор условного форматирования, тип ConditionalFormatOperator.

6.58.4 Метод ConditionalFormatRuleProxy.getRanges

Метод возвращает диапазоны ячеек таблицы, к которым применяется текущее правило.

Вызов

```
public CellRangePositions getRanges()
```

Возвращает

– диапазоны ячеек таблицы, тип `CellRangePositions`.

6.58.5 Метод ConditionalFormatRuleProxy.getStopCalculations

Метод позволяет определить, будут ли применяться другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
public bool getStopCalculations()
```

Возвращает

– `true`, если другие правила не применяются к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

6.58.6 Метод ConditionalFormatRuleProxy.getStyle

Метод возвращает настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
public ConditionalFormatCellStyle getStyle()
```

Возвращает

– настройки параметров отображения ячеек, тип `ConditionalFormatCellStyle`.

6.58.7 Метод ConditionalFormatRuleProxy.getTableName

Метод возвращает название листа, который содержит текущее правило.

Вызов

```
public string getTableName()
```

Возвращает

– название листа, тип `string`.

6.58.8 Метод ConditionalFormatRuleProxy.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.58.9 Метод ConditionalFormatRuleProxy.moveTo

Метод позволяет задать новый индекс правила в коллекции.

Вызов

```
public void moveTo(uint newIndex)
```

Параметры

– `newIndex`: новый индекс правила в коллекции, тип `uint`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
ConditionalFormatRuleProxy lastRule = rules.getRule(rules.getRuleCount() - 1);
lastRule.moveTo(0);
```

6.58.10 Метод ConditionalFormatRuleProxy.remove

Метод удаляет текущее правило из коллекции.

Вызов

```
public void remove()
```

6.58.11 Метод ConditionalFormatRuleProxy.setOperator

Метод позволяет задать оператор условного форматирования.

Вызов

```
public void setOperator(ConditionalFormatOperator
conditionalFormatOperator)
```

Параметры

– `conditionalFormatOperator`: оператор условного форматирования, тип `ConditionalFormatOperator` или его наследники.

6.58.12 Метод ConditionalFormatRuleProxy.setRange

Метод позволяет задать диапазон ячеек таблицы, к которому будет применено текущее правило.

Вызов

```
public void setRange(CellRangePosition cellRangePosition)
```

Параметры

– `cellRangePosition`: диапазон ячеек таблицы, тип `CellRangePosition`.

6.58.13 Метод ConditionalFormatRuleProxy.setRanges

Метод позволяет задать диапазоны ячеек таблицы, к которым будет применено текущее правило.

Вызов

```
public void setRanges(CellRangePositions cellRangePositions)
```

Параметры

– `cellRangePositions`: диапазоны ячеек таблицы, тип `CellRangePositions`.

6.58.14 Метод ConditionalFormatRuleProxy.setStopCalculations

Метод позволяет задать, применять ли другие правила условного форматирования к ячейкам, которые попадают под условия текущего правила.

Вызов

```
public void setStopCalculations(bool stopCalculations)
```

Параметры

– `stopCalculations`: `true`, чтобы не применять другие правила к ячейкам, которые попадают под текущее правило, в ином случае – `false`.

6.58.15 Метод ConditionalFormatRuleProxy.setStyle

Метод позволяет задать настройки параметров отображения ячеек, которые попадают под условие форматирования.

Вызов

```
public void setStyle(ConditionalFormatCellStyle style)
```

Параметры

– `style`: настройки параметров отображения ячеек, тип `ConditionalFormatCellStyle`.

6.59 Класс ConditionalFormatTableRules

Класс `ConditionalFormatTableRules` представляет собой коллекцию правил условного форматирования для листа табличного документа. Элементы коллекции представлены объектами класса `ConditionalFormatRuleProxy`. Используется в методе `Table.getConditionalFormatRules()`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.AboveAverage)
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule.getOperator());
}
```

6.59.1 Метод ConditionalFormatTableRules.addRule

Метод добавляет заданное правило в коллекцию.

Вызов

```
public void addRule(ConditionalFormatRule rule)
```

Параметры

– `rule`: правило условного форматирования, тип `ConditionalFormatRule`.

Пример

```
var aboveOperator =
DocumentAPI.createAboveAverageConditionalFormatOperator(ConditionalFormatAboveAve
rageCondition.Above);
var aboveRule = new ConditionalFormatRule(aboveOperator, aboveStyle,
cellRangePosition, false);
rules.addRule(aboveRule);
```

6.59.2 Метод ConditionalFormatTableRules.getRule

Метод возвращает правило по его индексу.

Вызов

```
public ConditionalFormatRuleProxy getRule(uint index)
```

Параметры

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

Возвращает

– Правило условного форматирования, тип `ConditionalFormatRuleProxy`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
ConditionalFormatRuleProxy rule = rules.getRule(rules.getRuleCount() - 1);
```

6.59.3 Метод ConditionalFormatTableRules.getRuleCount

Метод возвращает количество правил в коллекции.

Вызов

```
public uint getRuleCount()
```

Возвращает

– количество правил в коллекции, тип `uint`.

6.59.4 Метод ConditionalFormatTableRules.removeAllRules

Метод удаляет все правила условного форматирования из коллекции.

Вызов

```
public void removeAllRules()
```

6.59.5 Метод ConditionalFormatTableRules.removeRulesFromRange

Метод удаляет правила условного форматирования, которые попадают в заданный диапазон ячеек.

Вызов

```
public void removeRulesFromRange(CellRangePosition cellRangePosition)
```

Параметры

– `cellRangePosition`: диапазон ячеек таблицы, тип `CellRangePosition`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange cellRange = sheet.getCellRange("A2:F12");
var cellRangePosition = cellRange.getTableRange();
ConditionalFormatTableRules rules = sheet.getConditionalFormatRules();
rules.removeRulesFromRange(cellRangePosition);
```

6.60 Перечисление ConditionalFormatTextCondition

Перечисление `ConditionalFormatTextCondition` содержит условия применения форматирования для правила "Текст". Используется в методе `TextConditionalFormatOperator.getCondition()`.

Таблица 35 – Условия применения форматирования

Значение	Описание
`ConditionalFormatTextCondition.ContainsText`	Содержит
`ConditionalFormatTextCondition.DoesNotContainText`	Не содержит
`ConditionalFormatTextCondition.BeginsWith`	Начинается с
`ConditionalFormatTextCondition.EndsWith`	Заканчивается на

6.61 Перечисление ConditionalFormatTopBottomCondition

Перечисление `ConditionalFormatTopBottomCondition` содержит условия применения форматирования для правила "Наибольшие и наименьшие значения". Используется в методе `TopBottomConditionalFormatOperator.getCondition()`.

Таблица 36 – Условия применения форматирования

Значение	Описание
`ConditionalFormatTopBottomCondition.Top`	Наибольшие значения
`ConditionalFormatTopBottomCondition.Bottom`	Наименьшие значения

6.62 Перечисление ConditionalFormatUnaryCondition

Перечисление `ConditionalFormatUnaryCondition` содержит условия применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Используется в методе `UnaryConditionalFormatOperator.getCondition()`.

Таблица 37 – Условия применения форматирования

Значение	Описание
`ConditionalFormatUnaryCondition.Equal`	Равно
`ConditionalFormatUnaryCondition.NotEqual`	Не равно
`ConditionalFormatUnaryCondition.Greater`	Больше
`ConditionalFormatUnaryCondition.GreaterOrEqual`	Больше или равно
`ConditionalFormatUnaryCondition.Less`	Меньше
`ConditionalFormatUnaryCondition.LessOrEqual`	Меньше или равно
`ConditionalFormatUnaryCondition.Expression`	Позволяет использовать формулу в качестве аргумента

Пример

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

Рисунок 43 – Пример создания правила с формулой

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

var unaryStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(255, 0, 0, 150)));
unaryStyle.cellProperties = cellProperties;

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

var unaryOperator = new
UnaryConditionalFormatOperator(ConditionalFormatUnaryCondition.Expression,
"=ISBLANK(H2)");

var unaryRule = new ConditionalFormatRule(unaryOperator, unaryStyle,
cellRangePosition, false);
rules.addRule(unaryRule);
```

6.63 Перечисление ConditionalFormatUniquenessCondition

Перечисление `ConditionalFormatUniquenessCondition` содержит условия применения форматирования для правила "Уникальные и повторяющиеся значения". Используется в методе `UniquenessConditionalFormatOperator.getCondition()`.

Таблица 38 – Условия применения форматирования

Значение	Описание
`ConditionalFormatUniquenessCondition.Duplicate`	Повторяющиеся значения
`ConditionalFormatUniquenessCondition.Unique`	Уникальные значения

6.64 Класс ConditionalFormatValueObject

Класс `ConditionalFormatValueObject` представляет собой пороговое значение для применения подправил условного форматирования. Используется в правилах, созданных с помощью следующих операторов: `ColorScaleConditionalFormatOperator`,

DataBarConditionalFormatOperator и IconSetConditionalFormatOperator.

Конструкторы

```
public ConditionalFormatValueObject()
```

```
public ConditionalFormatValueObject(ConditionalFormatValueObjectType type, string value, bool isStrictCompare)
```

Пример

```
var value1 = new ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "0", false);
var entry1 = new ConditionalFormatIconSetEntry(value1, ConditionalFormatIconSet.FiveBoxes, 0);
```

6.64.1 Метод ConditionalFormatValueObject.getType

Метод возвращает тип текущего порогового значения.

Вызов

```
public ConditionalFormatValueObjectType getType()
```

Возвращает

– тип порогового значения, тип ConditionalFormatValueObjectType.

6.64.2 Метод ConditionalFormatValueObject.getValue

Метод возвращает значение порога.

Вызов

```
public string getValue()
```

Возвращает

– значение порога, тип `string`.

6.64.3 Метод ConditionalFormatValueObject.isStrictCompare

Метод позволяет определить, используется ли строгое сравнение в текущем пороговом значении (">" вместо ">=").

Вызов

```
public bool isStrictCompare()
```

Возвращает

– `true`, если используется строгое сравнение, в ином случае – `false`.

6.64.4 Метод ConditionalFormatValueObject.setStrictCompare

Метод позволяет использовать строгое сравнение в текущем пороговом значении (">" вместо ">=").

Вызов

```
public void setStrictCompare(bool isStrictCompare)
```

Параметры

– `isStrictCompare`: `true`, чтобы использовать строгое сравнение, в ином случае – `false`.

6.64.5 Метод ConditionalFormatValueObject.setType

Метод позволяет задать тип текущего порогового значения.

Вызов

```
public void setType(ConditionalFormatValueObjectType type)
```

Параметры

– `type`: тип порогового значения, тип `ConditionalFormatValueObjectType`.

6.64.6 Метод ConditionalFormatValueObject.setValue

Метод позволяет задать значение порога.

Вызов

```
public void setValue(string value)
```

Параметры

– `value`: значение порога, тип `string`.

6.65 Перечисление ConditionalFormatValueObjectType

Перечисление `ConditionalFormatValueObjectType` содержит типы пороговых значений, которые используются в правилах условного форматирования. Используется в

методах ConditionalFormatValueObject.getType() и ConditionalFormatValueObject.setType().

Таблица 39 – Типы пороговых значений

Значение	Описание
ConditionalFormatValueObjectType.Percentile	Процентиль
ConditionalFormatValueObjectType.Value	Фиксированное число
ConditionalFormatValueObjectType.Percent	Процент от наибольшего числа
ConditionalFormatValueObjectType.Formula	Результат формулы
ConditionalFormatValueObjectType.Min	Наименьшее число
ConditionalFormatValueObjectType.Max	Наибольшее число
ConditionalFormatValueObjectType.AutoMin	Автоматически устанавливает нижнее значение (только для правила "Гистограмма")
ConditionalFormatValueObjectType.AutoMax	Автоматически устанавливает верхнее значение (только для правила "Гистограмма")

Для корректного задания пороговых значений, при использовании типов Min, Max, AutoMin и AutoMax, необходимо дополнительно указать значение порога – 0.

Примеры

```
var value1 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "0",
false);
var entry1 = new ConditionalFormatIconSetEntry(value1,
ConditionalFormatIconSet.FiveBoxes, 0);
```

```
var value1 = new ConditionalFormatValueObject();
value1.setType(ConditionalFormatValueObjectType.AutoMin);
value1.setValue("0");
var value2 = new ConditionalFormatValueObject();
value2.setType(ConditionalFormatValueObjectType.AutoMax);
value2.setValue("0");

var dataBarParams = new ConditionalFormatDataBarParams(value1, value2);
```

6.66 Класс ConditionalTableFilter

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

Конструктор по умолчанию:

```
ConditionalTableFilter(bool andOperation = false);
```

Параметр

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

Конструктор копирования:

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

Пример использования приведен в разделе Работа с фильтрами.

6.66.1 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

```
ConditionalTableFilter songFilter = new ConditionalTableFilter();
songFilter.setAndOperation(true);
```

6.66.2 Методы добавления условий

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

- Критерии **match** и **notMatch** не могут быть сохранены для документов формата OXML.
- Критерии **aboveAverage**, **belowAverage**, **topValues**, **bottomValues**, **topPercent** и **bottomPercent** могут быть сохранены для документов формата OXML только если они являются единственными в фильтре.

```
void equal(string value)
void notEqual(string value)
void less(string value)
void lessOrEqual(string value)
void greater(string value)
void greaterOrEqual(string value)
void match(string value)
void notMatch(string value)
void begins(string value)
void notBegins(string value)
void ends(string value)
void notEnds(string value)
void contains(string value)
void notContains(string value)
void aboveAverage()
void belowAverage()
void topValues(ushort numValues)
void bottomValues(ushort numValues)
void topPercent(ushort percentage)
void bottomPercent(ushort percentage)
```

Пример

```
ConditionalTableFilter songFilter = new ConditionalTableFilter();
songFilter.notEqual("");
songFilter.notBegins("TODO");
```

Пример использования приведен в разделе Работа с фильтрами.

6.67 Класс Connection

Класс `Connection` реализует соединение между `Messenger` и клиентом. Содержит один метод `unsubscribe` для разрыва соединения.

Пример

```
MessageHandler messageHandler = new MessageHandler();
Messenger messenger = application.getMessenger();
Connection connection = messenger.subscribe(messageHandler);
.....................................................
connection.unsubscribe();
```

6.68 Перечисление ContainingTableFilter

Перечисление `ContainingTableFilter` содержит возможные типы фильтров столбцов таблицы. Используется в методе `TableFilters.getFilterType()`.

Таблица 40 – Типы фильтров

Значение	Описание
`ContainingTableFilter.None`	Фильтр отсутствует
`ContainingTableFilter.Values`	Фильтр по значению (`ValuesTableFilter`)
`ContainingTableFilter.Conditional`	Фильтр по условию (`ConditionalTableFilter`)
`ContainingTableFilter.Color`	Не поддерживается

Пример

```
ValuesTableFilter valuesFilter;
if (filters.getFilterType(0) == ContainingTableFilter.Values)
    valuesFilter = filters.getAsValueFilter(0);
```

6.69 Класс ContentControl

Базовый класс для элементов управления (см. Работа с элементами управления).

Наследники

- `CheckBoxControl`
- `DatePickerControl`
- `DropListControl`
- `InputFieldControl`

6.69.1 Метод ContentControl.canEdit

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

6.69.2 Метод ContentControl.getTitle

Метод возвращает название элемента управления.

6.69.3 Методы toCheckBox, toInputField, toDatePicker, toDropList

Методы преобразуют объект `ContentControl` в объект соответствующего типа. Возвращают `null`, если преобразование невозможно.

Пример

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

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

6.70 Класс ContentControls

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

Пример

```
ContentControls controls = document.getContentControls();
ContentControl textField = controls.findByTitle("input");
```

6.71 Класс CurrencyCellFormatting

Класс содержит параметры для денежного формата ячеек таблицы.

Таблица 41 – Описание полей класса `CurrencyCellFormatting`

Поле	Тип	Описание
`CurrencyCellFormatting.decimalPlaces`	`byte`	Количество десятичных позиций
`CurrencyCellFormatting.symbol`	`string`	Символ денежной единицы
`CurrencyCellFormatting.localeCode`	`int`	Идентификатор кода языка (MS-LCID)
`CurrencyCellFormatting.useRedForNegative`	`bool`	Использовать красный цвет для отрицательных значений
`CurrencyCellFormatting.useBracketsForNegative`	`bool`	Использовать скобки для отрицательных значений
`CurrencyCellFormatting.hideSign`	`bool`	Скрывать знак «минус» для отрицательных значений
`CurrencyCellFormatting.useThousandsSeparator`	`bool`	Использовать разделитель для тысячных
`CurrencyCellFormatting.currencySignPlacement`	`CurrencySignPlacement`	Варианты размещения знака валюты

Экземпляр данного класса используется в качестве аргумента метода `Cell.setFormat()`, см. пример.

Пример

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

CurrencyCellFormatting cellFormat = new CurrencyCellFormatting();
cellFormat.decimalPlaces = 2;
cellFormat.useThousandsSeparator = true;
```

```
cellFormat.useRedForNegative = true;
cellFormat.useBracketsForNegative = true;
cellFormat.hideSign = false;
cellFormat.currencySignPlacement = CurrencySignPlacement.Suffix;

cell.setFormat(cellFormat);
Console.WriteLine(cell.getFormattedValue());
```

6.72 Перечисление CurrencySignPlacement

Перечисление `CurrencySignPlacement` содержит варианты размещения знака валюты. Используется в полях `LocaleInfo.currencyFormat` и `CurrencyCellFormatting.currencySignPlacement`.

Таблица 42 – Описание вариантов расположения знаков валюты

Значение	Описание
`CurrencySignPlacement.Prefix`	Символ валюты перед значением ($12.00)
`CurrencySignPlacement.Suffix`	Символ валюты после значения (12.00 Р)

6.73 Класс DataBarConditionalFormatOperator

Класс `DataBarConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Гистограмма". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

Конструктор

```
public DataBarConditionalFormatOperator(ConditionalFormatDataBarParams params_)
```

Пример

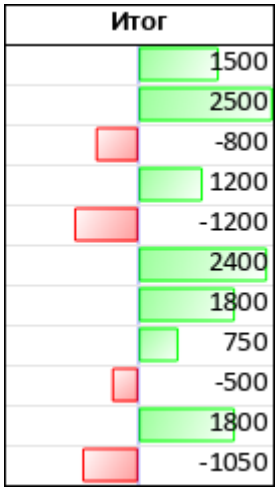


Рисунок 44 – Пример создания правила "Гистограмма"

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

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

var value1 = new ConditionalFormatValueObject();
value1.setType(ConditionalFormatValueObjectType.AutoMin);
value1.setValue("0");
var value2 = new ConditionalFormatValueObject();
value2.setType(ConditionalFormatValueObjectType.AutoMax);
value2.setValue("0");

var dataBarParams = new ConditionalFormatDataBarParams(value1, value2);
dataBarParams.barFill = new Color(new ColorRGBA(0, 255, 0, 100));
dataBarParams.borderColor = new Color(new ColorRGBA(0, 255, 0, 255));
dataBarParams.negativeBarFill = new Color(new ColorRGBA(255, 0, 0, 100));
dataBarParams.negativeBorderColor = new Color(new ColorRGBA(255, 0, 0, 255));
dataBarParams.axisPosition = ConditionalFormatDataBarAxisPosition.Middle;
dataBarParams.axisColor = new Color(new ColorRGBA(0, 0, 255, 100));
dataBarParams.fillType = ConditionalFormatDataBarFillType.Gradient;
dataBarParams.direction = ConditionalFormatDataBarDirection.LeftToRight;
dataBarParams.isValueVisible = true;

var dataBarOperator =
```

```
DocumentAPI.createDataBarConditionalFormatOperator(dataBarParams);

var dataBarRule = new ConditionalFormatRule();
dataBarRule.setRange(cellRangePosition);
dataBarRule.setOperator(dataBarOperator);
rules.addRule(dataBarRule);
```

6.73.1 Метод DataBarConditionalFormatOperator.getAxisColor

Метод возвращает цвет оси между положительными и отрицательными значениями.

Вызов

```
public Color getAxisColor()
```

Возвращает

– цвет оси, тип Color.

6.73.2 Метод DataBarConditionalFormatOperator.getAxisPosition

Метод возвращает позицию оси между положительными и отрицательными значениями.

Вызов

```
public ConditionalFormatDataBarAxisPosition getAxisPosition()
```

Возвращает

– расположение оси, тип ConditionalFormatDataBarAxisPosition.

6.73.3 Метод DataBarConditionalFormatOperator.getBarFill

Метод возвращает цвет заливки гистограммы для положительных значений.

Вызов

```
public Color getBarFill()
```

Возвращает

– цвет заливки для положительных значений, тип Color.

6.73.4 Метод DataBarConditionalFormatOperator.getBorderColor

Метод возвращает цвет границ гистограммы для положительных значений.

Вызов

```
public Color getBorderColor()
```

Возвращает

– цвет границ для положительных значений, тип Color.

6.73.5 Метод DataBarConditionalFormatOperator.getDirection

Метод возвращает направление гистограммы в ячейках.

Вызов

```
public ConditionalFormatDataBarDirection getDirection()
```

Возвращает

– направление гистограммы, тип ConditionalFormatDataBarDirection.

6.73.6 Метод DataBarConditionalFormatOperator.getFillType

Метод возвращает тип заливки гистограммы.

Вызов

```
public ConditionalFormatDataBarFillType getFillType()
```

Возвращает

– тип заливки гистограммы, тип ConditionalFormatDataBarFillType.

6.73.7 Метод DataBarConditionalFormatOperator.getLowerThreshold

Метод возвращает нижнее пороговое значение гистограммы.

Вызов

```
public ConditionalFormatValueObject getLowerThreshold()
```

Возвращает

– нижнее пороговое значение гистограммы, тип ConditionalFormatValueObject.

6.73.8 Метод DataBarConditionalFormatOperator.getMaxLength

Метод возвращает максимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
public double getMaxLength()
```

Возвращает

– максимальная длина гистограммы, в процентах, тип `double`.

6.73.9 Метод DataBarConditionalFormatOperator.getMinLength

Метод возвращает минимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
public double getMinLength()
```

Возвращает

– минимальная длина гистограммы, в процентах, тип `double`.

6.73.10 Метод DataBarConditionalFormatOperator.getNegativeBarFill

Метод возвращает цвет заливки гистограммы для отрицательных значений.

Вызов

```
public Color getNegativeBarFill()
```

Возвращает

– цвет заливки для отрицательных значений, тип `Color`.

6.73.11 Метод DataBarConditionalFormatOperator.getNegativeBorderColor

Метод возвращает цвет границ гистограммы для отрицательных значений.

Вызов

```
public Color getNegativeBorderColor()
```

Возвращает

– цвет границ для отрицательных значений, тип `Color`.

6.73.12 Метод DataBarConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип `ConditionalFormatOperatorType`.

6.73.13 Метод DataBarConditionalFormatOperator.getUpperThreshold

Метод возвращает верхнее пороговое значение гистограммы.

Вызов

```
public ConditionalFormatValueObject getUpperThreshold()
```

Возвращает

– верхнее пороговое значение гистограммы, тип `ConditionalFormatValueObject`.

6.73.14 Метод DataBarConditionalFormatOperator.getValueVisibility

Метод позволяет определить показывается ли значение в ячейке с гистограммой.

Вызов

```
public bool getValueVisibility()
```

Возвращает

– `true`, если значение показывается в ячейке с гистограммой, в ином случае – `false`.

6.73.15 Метод DataBarConditionalFormatOperator.setAxisColor

Метод позволяет задать цвет оси между положительными и отрицательными значениями.

Вызов

```
public void setAxisColor(Color color)
```

Параметры

– `color`: цвет оси, тип `Color`.

6.73.16 Метод DataBarConditionalFormatOperator.setAxisPosition

Метод позволяет задать позицию оси между положительными и отрицательными значениями.

Вызов

```
public void setAxisPosition(ConditionalFormatDataBarAxisPosition position)
```

Параметры

– `position`: расположение оси, тип `ConditionalFormatDataBarAxisPosition`.

6.73.17 Метод DataBarConditionalFormatOperator.setBarFill

Метод позволяет задать цвет заливки гистограммы для положительных значений.

Вызов

```
public void setBarFill(Color fill)
```

Параметры

– `fill`: цвет заливки для положительных значений, тип `Color`.

6.73.18 Метод DataBarConditionalFormatOperator.setBorderColor

Метод позволяет задать цвет границ гистограммы для положительных значений.

Вызов

```
public void setBorderColor(Color color)
```

Параметры

– `color`: цвет границ для положительных значений, тип `Color`.

6.73.19 Метод DataBarConditionalFormatOperator.setDirection

Метод позволяет задать направление гистограммы в ячейках.

Вызов

```
public void setDirection(ConditionalFormatDataBarDirection direction)
```

Параметры

– `direction`: направление гистограммы, тип `ConditionalFormatDataBarDirection`.

6.73.20 Метод DataBarConditionalFormatOperator.setFillType

Метод позволяет задать тип заливки гистограммы.

Вызов

```
public void setFillType(ConditionalFormatDataBarFillType fillType)
```

Параметры

– `fillType`: тип заливки гистограммы, тип `ConditionalFormatDataBarFillType`.

6.73.21 Метод DataBarConditionalFormatOperator.setLowerThreshold

Метод позволяет задать нижнее пороговое значение гистограммы.

Вызов

```
public void setLowerThreshold(ConditionalFormatValueObject lowerThreshold)
```

Параметры

– `lowerThreshold`: нижнее пороговое значение гистограммы, тип `ConditionalFormatValueObject`.

6.73.22 Метод DataBarConditionalFormatOperator.setMaxLength

Метод задает максимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
public void setMaxLength(double maxLength)
```

Параметры

– `maxLength`: максимальная длина гистограммы, в процентах, тип `double`.

6.73.23 Метод DataBarConditionalFormatOperator.setMinLength

Метод задает минимальную длину гистограммы, в процентах от ширины ячейки.

Вызов

```
public void setMinLength(double minLength)
```

Параметры

– `minLength`: минимальная длина гистограммы, в процентах, тип `double`.

6.73.24 Метод DataBarConditionalFormatOperator.setNegativeBarFill

Метод позволяет задать цвет заливки гистограммы для отрицательных значений.

Вызов

```
public void setNegativeBarFill(Color fill)
```

Параметры

– `fill`: цвет заливки для отрицательных значений, тип `Color`.

6.73.25 Метод DataBarConditionalFormatOperator.setNegativeBorderColor

Метод позволяет задать цвет границ гистограммы для отрицательных значений.

Вызов

```
public void setNegativeBorderColor(Color color)
```

Параметры

– `color`: цвет границ для отрицательных значений, тип `Color`.

6.73.26 Метод DataBarConditionalFormatOperator.setUpperThreshold

Метод позволяет задать верхнее пороговое значение гистограммы.

Вызов

```
public void setUpperThreshold(ConditionalFormatValueObject upperThreshold)
```

Параметры

– `upperThreshold`: верхнее пороговое значение гистограммы, тип `ConditionalFormatValueObject`.

6.73.27 Метод DataBarConditionalFormatOperator.setValueVisibility

Метод задает видимость значения в ячейке с гистограммой.

Вызов

```
public void setValueVisibility(bool visible)
```

Параметры

– `visible`: `true`, чтобы показывать значение в ячейке с гистограммой, в ином случае – `false`.

6.74 Класс DataValidation

Класс `DataValidation` представляет собой настройки проверки данных (см. Проверка данных). Используется в методах `CellRange.setDataValidation()`, `Cell.getDataValidation()` и `DataValidationResult.getDataValidation()`.

6.74.1 Метод DataValidation.clear

Метод очищает все настройки текущей проверки данных.

Вызов

```
public void clear()
```

Пример

```
Cell cell = sheet.getCell("C4");
DataValidation validation = cell.getDataValidation();
if (!validation.isEmpty())
    validation.clear();
```

6.74.2 Метод DataValidation.getAllowBlank

Метод возвращает разрешен ли ввод пустых значений.

Вызов

```
public bool getAllowBlank()
```

Возвращает

– `true`, если проверка данных разрешает ввод пустых значений, в ином случае – `false`.

6.74.3 Метод DataValidation.getErrorMessage

Метод возвращает текст сообщения об ошибке.

Вызов

```
public string getErrorMessage()
```

Возвращает

– текст сообщения об ошибке, тип `string`.

6.74.4 Метод DataValidation.getErrorStyle

Метод возвращает значение, которое определяет поведение редактора при вводе недопустимых значений.

Вызов

```
public DataValidationErrorStyle getErrorStyle()
```

Возвращает

– поведение редактора при вводе недопустимых значений, тип DataValidationErrorStyle.

6.74.5 Метод DataValidation.getErrorTitle

Метод возвращает заголовок сообщения об ошибке.

Вызов

```
public string getErrorTitle()
```

Возвращает

– заголовок сообщения об ошибке, тип `string`.

6.74.6 Метод DataValidation.getFormula1

Метод возвращает первый аргумент проверки данных. При проверке по списку значений, первым и единственным аргументом может быть:

- формула, результатом которой является диапазон ячеек (`"=INDIRECT(''[Source.xods]Data'!$E$2:$E$6")`);
- именованный диапазон (`"=Countries"`);
- адрес диапазона ячеек (`"='Data'!$E$2:$E$6"`);
- значения, разделенные точкой с запятой (`"UK; Italy; Germany; Austria; Brazil"`).

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()-7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=StartDate"`);
- адрес ячейки (`"='Data'!C2"`);
- дата в формате мм/дд/гггг (`"12/31/2024"`).

Вызов

```
public string getFormula1()
```

Возвращает

– первый аргумент проверки данных, тип `string`.

6.74.7 Метод DataValidation.getFormula2

Метод возвращает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов `Between` и `NotBetween`. Второй аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()+7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=EndDate"`);
- адрес ячейки (`"='Data'!C2"`);
- дата в формате мм/дд/гггг (`"12/31/2024"`).

Вызов

```
public string getFormula2()
```

Возвращает

– второй аргумент проверки данных, тип `string`.

6.74.8 Метод DataValidation.getOperator

Метод возвращает оператор сравнения введенного значения с аргументом/аргументами. Используется только при проверке дат.

Вызов

```
public DataValidationOperator getOperator()
```

Возвращает

– оператор сравнения, тип `DataValidationOperator`.

6.74.9 Метод DataValidation.getPrompt

Метод возвращает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
public string getPrompt()
```

Возвращает

– текст подсказки, тип `string`.

6.74.10 Метод DataValidation.getPromptTitle

Метод возвращает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
public string getPromptTitle()
```

Возвращает

– заголовок подсказки, тип `string`.

6.74.11 Метод DataValidation.getShowDropDown

Метод возвращает значение, которое определяет показывать ли выпадающий список значений. Используется только при проверке данных по списку значений.

Вызов

```
public bool getShowDropDown()
```

Возвращает

– `true`, если выпадающий список значений показывается, в ином случае – `false`.

6.74.12 Метод DataValidation.getShowErrorMessage

Метод возвращает значение, которое определяет показывать ли сообщение об ошибке при вводе недопустимых значений.

Вызов

```
public bool getShowErrorMessage()
```

Возвращает

– `true`, если сообщение об ошибке показывается, в ином случае – `false`.

6.74.13 Метод DataValidation.getShowInputMessage

Метод возвращает значение, которое определяет показывать ли подсказку при вводе данных в ячейку с проверкой.

Вызов

```
public bool getShowInputMessage()
```

Возвращает

– `true`, если подсказка показывается, в ином случае – `false`.

6.74.14 Метод DataValidation.getType

Метод возвращает тип проверки данных.

Вызов

```
public DataValidationType getType()
```

Возвращает

– тип проверки данных, тип `DataValidationType`.

6.74.15 Метод DataValidation.isEmpty

Метод позволяет определить наличие заданных настроек проверки данных в текущем объекте `DataValidation`.

Вызов

```
public bool isEmpty()
```

Возвращает

– `true`, если объект не содержит заданных настроек проверки данных, в ином случае – `false`.

Пример

```
Cell cell = sheet.getCell("C4");
DataValidation validation = cell.getDataValidation();
if (!validation.isEmpty())
    validation.clear();
```

6.74.16 Метод DataValidation.setAllowBlank

Метод позволяет разрешить ввод пустых значений.

Вызов

```
public void setAllowBlank(bool allowBlank)
```

Параметры

– `allowBlank`: `true`, чтобы разрешить ввод пустых значений, в ином случае – `false`.

6.74.17 Метод DataValidation.setErrorMessage

Метод задает текст сообщения об ошибке.

Вызов

```
public void setErrorMessage(string errorMessage)
```

Параметры

– `errorMessage`: текст сообщения об ошибке, тип `string`.

Пример

```
DataValidation validation = new DataValidation();
// ...
// Настройки сообщения об ошибке:
validation.setShowErrorMessage(true);
validation.setErrorStyle(DataValidationErrorStyle.Warning);
validation.setErrorTitle("Error");
validation.setErrorMessage("Invalid value");

cellRange.setDataValidation(validation);
```

6.74.18 Метод DataValidation.setErrorStyle

Метод задает поведение редактора при вводе недопустимых значений.

Вызов

```
public void setErrorStyle(DataValidationErrorStyle errorStyle)
```

Параметры

– `errorStyle`: поведение редактора при вводе недопустимых значений, тип `DataValidationErrorStyle`.

Пример

```
DataValidation validation = new DataValidation();
// ...
// Настройки сообщения об ошибке:
validation.setShowErrorMessage(true);
validation.setErrorStyle(DataValidationErrorStyle.Warning);
validation.setErrorTitle("Error");
validation.setErrorMessage("Invalid value");
```

```
cellRange.setDataValidation(validation);
```

6.74.19 Метод DataValidation.setErrorTitle

Метод задает заголовок сообщения об ошибке.

Вызов

```
public void setErrorTitle(string errorTitle)
```

Параметры

– `errorTitle`: заголовок сообщения об ошибке, тип `string`.

Пример

```
DataValidation validation = new DataValidation();
// ...
// Настройки сообщения об ошибке:
validation.setShowErrorMessage(true);
validation.setErrorStyle(DataValidationErrorStyle.Warning);
validation.setErrorTitle("Error");
validation.setErrorMessage("Invalid value");

cellRange.setDataValidation(validation);
```

6.74.20 Метод DataValidation.setFormula1

Метод задает первый аргумент проверки данных. При проверке по списку значений, первым и единственным аргументом может быть:

- формула, результатом которой является диапазон ячеек (`"=INDIRECT(''[Source.xods]Data'!$E$2:$E$6")`);
- именованный диапазон (`"=Countries"`);
- адрес диапазона ячеек (`"='Data'!$E$2:$E$6"`);
- значения, разделенные точкой с запятой (`"UK; Italy; Germany; Austria; Brazil"`).

При проверке дат аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()-7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=StartDate"`);

- адрес ячейки (`"='Data'!C2"`);
- дата в формате мм/дд/гггг (`"12/31/2024"`).

Вызов

```
public void setFormula1(string formula1)
```

Параметры

– `formula1`: первый аргумент проверки данных, тип `string`.

Пример

```
DataValidation dvList = new DataValidation();
dvList.setType(DataValidationType.List);
dvList.setFormula1("UK; Italy; Germany; Austria; Brazil");
dvList.setShowDropDown(true);

cellRange.setDataValidation(dvList);
```

6.74.21 Метод DataValidation.setFormula2

Метод задает второй аргумент проверки данных. Используется только при проверке дат с помощью операторов `Between` и `NotBetween`. Второй аргумент может принимать следующие значения:

- формула, результатом которой является дата (`"=TODAY()+7"`);
- именованный диапазон, состоящий из одной ячейки с датой (`"=EndDate"`);
- адрес ячейки (`"='Data'!C2"`);
- дата в формате мм/дд/гггг (`"12/31/2024"`).

Вызов

```
public void setFormula2(string formula2)
```

Параметры

– `formula2`: второй аргумент проверки данных, тип `string`.

Пример

```
DataValidation dvDate = new DataValidation();
dvDate.setType(DataValidationType.Date);
dvDate.setOperator(DataValidationOperator.Between);
dvDate.setFormula1("06/01/2024");
dvDate.setFormula2("08/31/2024");
```

```
cellRange.setDataValidation(dvDate);
```

6.74.22 Метод DataValidation.setOperator

Метод задает оператор сравнения введенного значения с аргументом/аргументами. Используется только при проверке дат.

Вызов

```
public void setOperator(DataValidationOperator dataValidationOperator)
```

Параметры

– `dataValidationOperator`: оператор сравнения, тип `DataValidationOperator`.

Пример

```
DataValidation dvDate = new DataValidation();
dvDate.setType(DataValidationType.Date);
dvDate.setOperator(DataValidationOperator.Between);
dvDate.setFormula1("06/01/2024");
dvDate.setFormula2("08/31/2024");

cellRange.setDataValidation(dvDate);
```

6.74.23 Метод DataValidation.setPrompt

Метод задает текст подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
public void setPrompt(string prompt)
```

Параметры

– `prompt`: текст подсказки, тип `string`.

Пример

```
DataValidation validation = new DataValidation();
// ...
// Настройки подсказки:
validation.setShowInputMessage(true);
validation.setPromptTitle("Restricted input");
```

```
validation.setPrompt("Choose values from the drop-down list");


cellRange.setDataValidation(validation);
```

6.74.24 Метод DataValidation.setPromptTitle

Метод задает заголовок подсказки, которая показывается при вводе данных в ячейку с проверкой.

Вызов

```
public void setPromptTitle(string promptTitle)
```

Параметры

– `promptTitle`: заголовок подсказки, тип `string`.

Пример

```
DataValidation validation = new DataValidation();
// ...
// Настройки подсказки:
validation.setShowInputMessage(true);
validation.setPromptTitle("Restricted input");
validation.setPrompt("Choose values from the drop-down list");


cellRange.setDataValidation(validation);
```

6.74.25 Метод DataValidation.setShowDropDown

Метод определяет показывать ли выпадающий список значений. Используется только при проверке данных по списку значений.

Вызов

```
public void setShowDropDown(bool showDropDown)
```

Параметры

– `showDropDown`: `true`, чтобы показывать выпадающий список значений, в ином случае – `false`.

Пример

```
DataValidation dvList = new DataValidation();
dvList.setType(DataValidationType.List);
```

```
dvList.setFormula1("UK; Italy; Germany; Austria; Brazil");
dvList.setShowDropDown(true);

cellRange.setDataValidation(dvList);
```

6.74.26 Метод DataValidation.setShowErrorMessage

Метод определяет показывать ли сообщение об ошибке при вводе недопустимых значений.

Вызов

```
public void setShowErrorMessage(bool showErrorMessage)
```

Параметры

– `showErrorMessage`: `true`, чтобы показывать сообщение об ошибке, в ином случае – `false`.

Пример

```
DataValidation validation = new DataValidation();
// ...
// Настройки сообщения об ошибке:
validation.setShowErrorMessage(true);
validation.setErrorStyle(DataValidationErrorStyle.Warning);
validation.setErrorTitle("Error");
validation.setErrorMessage("Invalid value");

cellRange.setDataValidation(validation);
```

6.74.27 Метод DataValidation.setShowInputMessage

Метод определяет показывать ли подсказку при вводе данных в ячейку с проверкой.

Вызов

```
public void setShowInputMessage(bool showInputMessage)
```

Параметры

– `showInputMessage`: `true`, чтобы показывать подсказку, в ином случае – `false`.

Пример

```
DataValidation validation = new DataValidation();
// ...
// Настройки подсказки:
validation.setShowInputMessage(true);
validation.setPromptTitle("Restricted input");
validation.setPrompt("Choose values from the drop-down list");

cellRange.setDataValidation(validation);
```

6.74.28 Метод DataValidation.setType

Метод задает тип проверки данных.

Вызов

```
public void setType(DataValidationType type)
```

Параметры

– `type`: тип проверки данных, тип `DataValidationType`.

Пример

```
DataValidation dvDate = new DataValidation();
dvDate.setType(DataValidationType.Date);
dvDate.setOperator(DataValidationOperator.Between);
dvDate.setFormula1("06/01/2024");
dvDate.setFormula2("08/31/2024");

cellRange.setDataValidation(dvDate);
```

6.75 Перечисление DataValidationErrorStyle

Перечисление `DataValidationErrorStyle` содержит варианты поведения редактора при вводе недопустимых значений. Используется в методах `DataValidation.getErrorStyle()` и `DataValidation.setErrorStyle()`.

Таблица 43 – Варианты поведения редактора при вводе недопустимых значений

Значение	Описание
`DataValidationErrorStyle.Stop`	Запрещает ввод в ячейку недопустимого значения

Значение	Описание
`DataValidationErrorStyle.Warning`	Позволяет ввести в ячейку недопустимое значение после соответствующего предупреждения
`DataValidationErrorStyle.Information`	Поведение идентично значению `Warning`

6.76 Перечисление DataValidationOperator

Перечисление `DataValidationOperator` содержит операторы сравнения введенной даты с аргументом/аргументами. Используется в методах `DataValidation.getOperator()` и `DataValidation.setOperator()`.

Таблица 44 – Описание операторов сравнения дат

Значение	Описание
`DataValidationOperator.Between`	Дата должна попадать в интервал между двумя аргументами включительно
`DataValidationOperator.NotBetween`	Дата должна быть вне интервала между двумя аргументами
`DataValidationOperator.Equal`	Дата должна совпадать со значением первого аргумента
`DataValidationOperator.NotEqual`	Дата не должна совпадать со значением первого аргумента
`DataValidationOperator.LessThan`	Дата должна быть раньше первого аргумента
`DataValidationOperator.LessThanOrEqual`	Дата должна быть раньше первого аргумента или совпадать с ним
`DataValidationOperator.GreaterThan`	Дата должна быть позже первого аргумента
`DataValidationOperator.GreaterThanOrEqual`	Дата должна быть позже первого аргумента или совпадать с ним

6.77 Класс DataValidationResult

Класс `DataValidationResult` представляет собой результат проверки значения ячейки (см. Проверка данных). Используется в методе `Cell.checkDataValidation()`.

6.77.1 Метод DataValidationResult.getDataValidation

Метод возвращает настройки проверки данных, если ячейка содержит недопустимое значение.

Вызов

```
public DataValidation getDataValidation()
```

Возвращает

– настройки проверки данных, тип DataValidation.

– null, если ячейка содержит допустимое значение.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("C10");
DataValidationResult result = cell.checkDataValidation();
if (!result.isValid())
    Console.WriteLine(result.getDataValidation().getErrorMessage());
```

6.77.2 Метод DataValidationResult.isValid

Метод возвращает является ли значение ячейки допустимым при текущих настройках проверки данных.

Вызов

```
public bool isValid()
```

Возвращает

– true, если значение ячейки допустимо или проверка данных отсутствует, в ином случае – false.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("C10");
DataValidationResult result = cell.checkDataValidation();
if (!result.isValid())
    Console.WriteLine(result.getDataValidation().getErrorMessage());
```

6.78 Перечисление DataValidationType

Перечисление DataValidationType содержит типы проверки данных. Используется в методах DataValidation.getType() и DataValidation.setType().

Таблица 45 – Описание типов проверки данных

Значение	Описание
DataValidationType.None	Проверка данных отсутствует
DataValidationType.Integer	Не поддерживается
DataValidationType.Fractional	Не поддерживается
DataValidationType.List	Проверка данных по списку доступных значений
DataValidationType.Date	Проверка данных в формате **Дата**
DataValidationType.Time	Не поддерживается
DataValidationType.TextLength	Не поддерживается
DataValidationType.Custom	Не поддерживается

6.79 Перечисление DatePatterns

Форматы даты представлены в таблице 46. Пример использования см. в главе DateTimeCellFormatting.

Таблица 46 – Форматы даты

Значение	Описание
DatePatterns.DayMonthTextLongYearLong	'mmmm dd, yyyy' для языка en_US
DatePatterns.FullDate	'день недели, mmmm dd, yyyy' для языка en_US
DatePatterns.DayMonthNumberLongYearLong	'mm/dd/yyyy' для языка en_US
DatePatterns.DayMonthNumberLongYearShort	'mm/dd/yy' для языка en_US
DatePatterns.DayMonthNumberShortYearShort	'm/dd/yy' для языка en_US
DatePatterns.DayMonthTextShort	'dd-mmm' для языка en_US
DatePatterns.MonthTextShortYearShort	'mmm-yy' для языка en_US
DatePatterns.DayMonthTextShortYearShort	'mmm dd, yy' для языка en_US
DatePatterns.DayMonthYearLongNonLocalizableHyphen	Нелокализуемый шаблон 'dd-mm-yyyy'
DatePatterns.DayMonthYearLongNonLocalizableSlash	Нелокализуемый шаблон 'dd/mm/yyyy'

6.80 Класс DatePickerControl

Представляет собой элемент управления "Выбор даты", используемый для ввода дат в документе. Является наследником класса `ContentControl`. Методы `DatePickerControl.getValue()` и `DatePickerControl.setValue(DateTime)` позволяют получить и задать значение этого элемента управления.

Пример

```
ContentControls controls = document.getContentControls();
DatePickerControl startDate = controls.findByTitle("startDate").toDatePicker();
DatePickerControl endDate = controls.findByTitle("endDate").toDatePicker();
var value = startDate.getValue();
value.year = (ushort)(value.year + 1);
endDate.setValue(value);
```

6.81 Класс DateTime

Класс `DateTime` предоставляет дату и время с точностью до секунды. Используется для поля `TrackedChangeInfo.timeStamp`.

Таблица 47 – Описание полей класса `DateTime`

Поле	Тип	Описание
`DateTime.year`	`ushort`	Год
`DateTime.month`	`byte`	Месяц
`DateTime.day`	`byte`	День
`DateTime.hour`	`byte`	Часы
`DateTime.minute`	`byte`	Минуты
`DateTime.second`	`byte`	Секунды

6.82 Класс DateTimeCellFormatting

Класс `DateTimeCellFormatting` содержит параметры для формата ячеек таблицы типа Дата и Время, используется в качестве аргумента метода `Cell.setFormat()`.

Таблица 48 – Описание полей класса `DateTimeCellFormatting`

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

Пример

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

DateTimeCellFormatting cellFormat = new DateTimeCellFormatting();
cellFormat.dateListID = DatePatterns.DayMonthNumberLongYearShort;
cellFormat.timeListID = TimePatterns.LongTime;

cell.setFormat(cellFormat, CellFormat.DateTime);
Console.WriteLine(cell.getFormattedValue());
```

6.83 Перечисление DateTimeFormat

В таблице 49 представлены варианты форматирования даты и времени. Используется в качестве параметра метода `CellRange.insertCurrentDateTime()`.

Таблица 49 – Варианты форматирования даты и времени

Значение	Описание
`DateTimeFormat.DateTime`	Дата и время
`DateTimeFormat.Date`	Дата
`DateTimeFormat.Time`	Время

6.84 Класс Document

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

Пример

```
Blocks blocks = document.getBlocks();
if (blocks != null) {
    Paragraph paragraph = blocks.getParagraph(0);
    ..........................................
}
```

6.84.1 Метод Document.areMirroredMarginsEnabled

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

Пример

```
Console.WriteLine(document.areMirroredMarginsEnabled());
```

6.84.2 Метод Document.calculateAllFormulas

Метод пересчитывает все формулы в документе. Используйте этот метод после изменения данных табличного документа, если функция автоматического пересчета формул отключена. Чтобы узнать текущее состояние этой функции, воспользуйтесь методом `Document.getCalculationMode()`.

Также вы можете использовать метод `Document.calculateOutdatedFormulas()` для пересчета только формул, данные которых были изменены, и метод `calculate()` объектов `Table`, `CellRange` или `Cell` для пересчета формул, находящихся в определенном диапазоне.

6.84.3 Метод Document.calculateOutdatedFormulas

Метод пересчитывает формулы, данные которых были изменены. Используйте этот метод после изменения данных табличного документа, если функция автоматического пересчета формул отключена. Чтобы узнать текущее состояние этой функции, воспользуйтесь методом `Document.getCalculationMode()`.

Также вы можете использовать метод `calculate()` объектов `Table`, `CellRange` или `Cell` для пересчета формул, находящихся в определенном диапазоне.

6.84.4 Метод Document.exportAs

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

Расширенные версии метода позволяют указать дополнительные настройки экспорта документа:

- для текстовых документов – класс `TextExportSettings`;
- для табличных документов – класс `WorkbookExportSettings`;
- для презентационных документов – класс `PresentationExportSettings`.

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

Существуют следующие варианты реализации метода:

```
exportAs(String filePath, ExportFormat exportFormat);
exportAs(String filePath, ExportFormat exportFormat, TextExportSettings
textExportSettings);
exportAs(String filePath, ExportFormat exportFormat, WorkbookExportSettings
workbookExportSettings);
exportAs(String filePath, ExportFormat exportFormat, PresentationExportSettings
presentationExportSettings);
```

Примеры использования метода `exportAs` приведены в разделах Сохранение и экспорт текстового документа и Сохранение и экспорт табличного документа.

6.84.5 Метод Document.getAbsoluteFilePath

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

Пример

```
var documentPath = document.getAbsoluteFilePath();
```

Ограничения:

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

6.84.6 Метод Document.getBlocks

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

Пример

```
Blocks blocks = document.getBlocks();
if (blocks != null) {
    Paragraph paragraph = blocks.getParagraph(0);
    if (paragraph != null) {
        Console.WriteLine(paragraph.getListLevel());
    }
}
```

6.84.7 Метод Document.getBookmarks

Метод предоставляет доступ к списку закладок Bookmarks.

Пример

```
Bookmarks bookmarks = document.getBookmarks();
if (bookmarks != null) {
    Range range = bookmarks.getBookmarkRange("Bookmark");
    range.replaceText("New bookmark text");
    Console.WriteLine(range.extractText());
}
```

6.84.8 Метод Document.getCalculationMode

Метод возвращает текущий режим пересчета формул в документе CalculationMode. Чтобы изменить этот режим, используйте метод Document.setCalculationMode().

6.84.9 Метод Document.getComments

Метод обеспечивает доступ к комментариям документа, возвращает объект Comments.

Пример

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
foreach (var comment in commentsEnumerator)
```

```
{
    Console.WriteLine(comment.getText());
}
```

6.84.10 Метод Document.getConditionalFormatRules

Метод позволяет получить коллекцию правил условного форматирования для табличного документа.

Вызов

```
public ConditionalFormatDocumentRules getConditionalFormatRules()
```

Возвращает

– коллекция правил условного форматирования для документа, тип ConditionalFormatDocumentRules.

Пример

```
ConditionalFormatDocumentRules rules = document.getConditionalFormatRules();
foreach (ConditionalFormatRuleProxy rule in rules) {
    if (rule.getType() == ConditionalFormatOperatorType.AboveAverage)
        aboveOperator =
DocumentAPI.castToAboveAverageConditionalFormat(rule.getOperator());
}
```

6.84.11 Метод Document.getContentControls

Метод предоставляет доступ к списку элементов управления ContentControls, содержащихся в документе (см. Работа с элементами управления).

Пример

```
ContentControls controls = document.getContentControls();
foreach (var control in controls)
    Console.WriteLine(control.getTitle());
```

6.84.12 Метод Document.getFormulaType

Метод возвращает поддерживаемую адресацию ячеек FormulaType документа.

Пример

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

6.84.13 Метод Document.getNamedExpressions

Используется для получения списка именованных диапазонов NamedExpressions.

Пример

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

6.84.14 Метод Document.getPivotTablesManager

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

Пример

```
PivotTablesManager pivotTablesManager = document.getPivotTablesManager();
if (pivotTablesManager != null) {
    ..........................
}
```

6.84.15 Метод Document.getRange

Метод предоставляет доступ ко всему диапазону Range документа.

Пример

```
Range range = document.getRange();
if (range != null) {
    Console.WriteLine(range.extractText());
}
```

6.84.16 Метод Document.getScripts

Метод предоставляет доступ к списку макрокоманд Scripts, содержащихся в документе.

Пример

```
Scripts scripts = document.getScripts();
ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
foreach (var script in scriptsEnumerator)
{
    Console.WriteLine(script.getName());
}
```

6.84.17 Метод Document.getSections

Возвращает объект типа Sections.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getRange().extractText());
}
```

6.84.18 Метод Document.getSectionsEnumerator

Позволяет перечислить секции документа, тип Section.

Пример

```
SectionsEnumerator sectionsEnumerator = document.getSectionsEnumerator();
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getRange().extractText());
}
```

6.84.19 Метод Document.getTextStyles

Метод возвращает объект, который позволяет взаимодействовать со стилями абзацев в документе.

Вызов

```
public TextStyles getTextStyles()
```

Возвращает

– объект для работы со стилями абзацев, тип TextStyles.

Пример

```
TextStyles styles = document.getTextStyles();
foreach (TextStyle style in styles.getEnumerator())
    Console.WriteLine(style.getName());
```

6.84.20 Метод Document.getVBAModules

Метод позволяет получить коллекцию VBA макрокоманд из текущего документа.

Вызов

```
public VBAModules getVBAModules()
```

Возвращает

– коллекция VBA макрокоманд, тип VBAModules.

Пример

```
VBAModules modules = document.getVBAModules();
foreach (VBAModule module in modules) {
    module.getName();
}
```

6.84.21 Метод Document.isCalculatedOnSave

Метод возвращает состояние функции пересчета формул при сохранении документа. Используйте метод Document.setCalculatedOnSave(), чтобы включить или отключить эту функцию.

6.84.22 Метод Document.isChangesTrackingEnabled

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

Пример

```
Console.WriteLine(document.isChangesTrackingEnabled());
```

6.84.23 Метод Document.isStructureProtected

Метод возвращает состояние защиты от изменения структуры табличного документа (см. Защита структуры табличного документа).

Вызов

```
public bool isStructureProtected()
```

6.84.24 Метод Document.merge

Метод `Document.merge` сравнивает текущий документ с другим документом, который передается в параметре типа `Document`.

Метод возвращает объект `Document`, содержащий результат сравнения в виде отслеживаемых изменений.

Пример

```
var firstDoc = application.loadDocument("C:/Tmp/Sample1.docx");
var secondDoc = application.loadDocument("C:/Tmp/Sample2.docx");

var mergedDoc = firstDoc.merge(secondDoc);
var outputFilePath = "C:/Tmp/Sample3.docx";

mergedDoc.saveAs(outputFilePath);
```

Результат выполнения данного примера (сравнение двух документов, содержащих "1111" и "2222") приведен на рисунке 2.

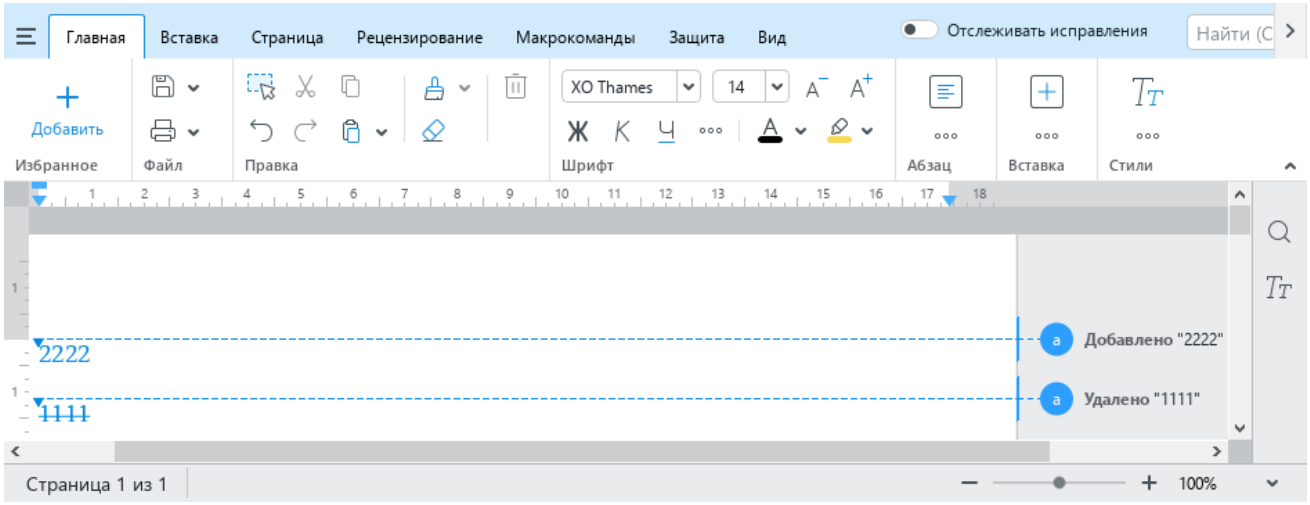


Рисунок 45 – Результат выполнения метода `merge`

6.84.25 Метод Document.removeStructureProtection

Метод снимает защиту от изменений структуры табличного документа (см. Защита структуры табличного документа).

Вызов

```
public void removeStructureProtection(string password)
```

Параметры

– `password`: (необязательный) пароль для снятия защиты, тип `string`.

Пример

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

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

6.84.26 Метод Document.saveAs

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

При необходимости, в качестве второго аргумента можно использовать объект класса `SaveDocumentSettings`, которая содержит формат документа `DocumentFormat`, тип документа `DocumentType` и пароль для защиты документа от несанкционированного доступа.

Существуют следующие варианты реализации метода:

```
Document saveAs(String filePath);
Document saveAs(String filePath, SaveDocumentSettings saveDocumentSettings);
```

Примеры использования метода `saveAs` приведены в разделах Сохранение и экспорт текстового документа и Сохранение и экспорт табличного документа.

6.84.27 Метод Document.setCalculatedOnSave

Метод позволяет включить и отключить функцию пересчета формул при сохранении документа. Используйте метод `Document.isCalculatedOnSave()`, чтобы узнать текущее состояние этой функции.

6.84.28 Метод Document.setCalculationMode

Метод позволяет задать режим пересчета формул в документе `CalculationMode`. Используйте метод `Document.getCalculationMode()`, чтобы получить текущий режим пересчета.

6.84.29 Метод Document.setChangesTrackingEnabled

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

Пример

```
document.setChangesTrackingEnabled(true);
```

6.84.30 Метод Document.setFormulaType

Метод устанавливает поддерживаемую адресацию ячеек FormulaType документа.

Пример

```
document.setFormulaType(FormulaType.A1);
```

6.84.31 Метод Document.setMirroredMarginsEnabled

Метод позволяет включать и выключать зеркальные поля в документе.

Пример

```
document.setMirroredMarginsEnabled(true);
```

6.84.32 Метод Document.setPageOrientation

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

Пример

```
document.setPageOrientation(PageOrientation.Landscape);
```

6.84.33 Метод Document.setPageProperties

Метод устанавливает свойство PageProperties в документе.

Пример

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

6.84.34 Метод Document.setStructureProtection

Метод устанавливает защиту от изменений структуры табличного документа (см. Защита структуры табличного документа).

Вызов

```
public void setStructureProtection(string password)
```

Параметры

– `password`: (необязательный) пароль для установки защиты, тип `string`.

Пример

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

Если метод `setStructureProtection()` применяется к документу с уже защищенной структурой, возникает исключение `SpreadsheetProtectionError`.

6.85 Перечисление DocumentFormat

В таблице 50 приведены поддерживаемые форматы документов, которые используются в поле `SaveDocumentSettings.documentFormat`.

Таблица 50 – Форматы документов

Значение	Описание
`DocumentFormat.PlainText`	Используется для работы с файлами TXT.
`DocumentFormat.DSV`	Используется для работы с табличными данными в текстовой форме (CSV, DSV). Строка текста содержит одно или несколько полей данных, разделенных запятыми или иным разделителем.
`DocumentFormat.OXML`	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML.
`DocumentFormat.ODF`	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010).
`DocumentFormat.HTML`	Используется для работы с веб-документами в формате HTML. Работа с веб-документами в формате HTML средствами Document API в настоящий момент не поддерживается.
`DocumentFormat.PDF`	Используется для работы с документами в формате Portable Document Format (PDF) версии 1.4.

Значение	Описание
`DocumentFormat.PDFA`	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

6.86 Класс DocumentSettings

Класс `DocumentSettings` предоставляет общие настройки документа и используется в методе `Application.createDocument` и поле `LoadDocumentSettings.commonDocumentSettings`.

Таблица 51 – Описание полей класса `DocumentSettings`

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

6.87 Перечисление DocumentType

В таблице 52 приведены поддерживаемые типы документов, которые используются при создании документа `Application.createDocument`, `DocumentSettings`.

Таблица 52 - Типы документов

Значение	Описание
`DocumentType.Text`	Используется для работы с текстовыми документами в форматах DOCX, ODT, XODT, TXT.
`DocumentType.Workbook`	Используется для работы с табличными документами в форматах XLSX, ODS, XODS.
`DocumentType.Presentation`	Используется для работы с презентационными документами в форматах PPTX, ODP, XODP.

6.88 Класс DropListControl

Представляет собой элемент управления "Выпадающий список" в документе. Является наследником класса `ContentControl`. Методы `DropListControl.getValue()` и

`DropListControl.setValue(int)` позволяют получить и задать значение этого элемента управления. Метод `DropListControl.getChoices()` возвращает коллекцию элементов, находящихся в выпадающем списке.

Пример

```
ContentControls controls = document.getContentControls();
DropListControl comboBox = controls.findByTitle("select").toDropList();
comboBox.setValue(comboBox.getChoices().IndexOf("two"));
```

6.89 Класс DSVSettings

Класс `DSVSettings` предоставляет настройки, необходимые для работы с файлами CSV (comma-separated value) и DSV (delimiter-separated value). Используется в `SaveDocumentSettings`, `LoadDocumentSettings`.

Таблица 53 – Описание полей класса `DSVSettings`

Поле	Тип	Описание
`DSVSettings.autofit`	`bool`	Признак необходимости автоматического подстраивания ширины столбца под размер данных в ячейке
`DSVSettings.startBlockIndex`	`uint`	Индекс первого листа документа для сохранения
`DSVSettings.lastBlockIndex`	`uint`	Индекс последнего листа документа для сохранения

Пример

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

6.90 Перечисление Encoding

В таблице 54 приведены поддерживаемые кодировки документов. Используется в `LoadDocumentSettings`.

Таблица 54 - Кодировки документов

Значение	Кодировка
`Encoding.Unknown`	Невозможно определить кодировку
`Encoding.UTF8`	UTF8
`Encoding.UTF16BE`	UTF16BE
`Encoding.UTF16LE`	UTF16LE
`Encoding.UTF32BE`	UTF32BE
`Encoding.UTF32LE`	UTF32LE
`Encoding.Windows1250`	Windows1250
`Encoding.Windows1251`	Windows1251
`Encoding.Windows1252`	Windows1252
`Encoding.ISO8859Part5`	ISO8859Part5
`Encoding.KOI8R`	KOI8R
`Encoding.KOI8U`	KOI8U
`Encoding.CP866`	CP866

6.91 Перечисление ExportFormat

В таблице 55 приведены поддерживаемые форматы экспорта документов (см. `Document.exportAs`).

Таблица 55 - Форматы экспорта документов

Значение	Описание
`ExportFormat.PDFA1`	Используется для работы с документами в формате Portable Document Format (PDF).

6.92 Класс Field

Класс `Field` предназначен для реализации некоторых полей, например, содержания (см. класс `Block`).

6.93 Класс Fill

Класс определяет заполнение фигуры, используется в качестве поля `fill` классов `ShapeProperties`, `CellProperties`.

Варианты заполнения:

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

Примеры

```
// Без заполнения
shapeProperties.fill = new Fill();
```

```
// Заполнение цветом
shapeProperties.fill = new Fill(new Color(new ColorRGBA(55, 146, 179, 200)));
```

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

6.93.1 Метод Fill.getColor

Метод возвращает цвет заполнения `Color`.

6.93.2 Метод Fill.getUrl

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

6.93.3 Метод Fill.isNoFill

Метод возвращает `true`, если заполнения нет.

6.94 Класс FiltersRange

Класс `FiltersRange` реализует диапазон таблицы, позволяющий манипулировать фильтрами столбцов. Пример использования приведен в разделе Работа с фильтрами.

6.94.1 Метод FIltersRange.clear

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

При использовании данного метода может произойти исключение InvalidObjectError в случае, если диапазон фильтрации недействителен.

Пример

```
FiltersRange filtersRange = sheet.createFiltersRange(cellRange);
.....................................
filtersRange.clear();
```

6.94.2 Метод FIltersRange.eraseFilters

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

При использовании данного метода может произойти исключение InvalidObjectError в случае, если диапазон фильтрации недействителен.

Пример

```
FiltersRange filtersRange = sheet.createFiltersRange(cellRange);
.....................................
filtersRange.eraseFilters();
```

6.94.3 Метод FiltersRange.getCellRange

Метод `FiltersRange.getCellRange` возвращает диапазон ячеек, содержащий текущий объект фильтрации. Возвращаемый диапазон включает строку заголовка. Если объект фильтрации не определен, то возвращается `null`.

Пример

```
CellRange cellRange = filtersRange.getCellRange();
Console.WriteLine(cellRange.getBeginRow());
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.94.4 Метод FiltersRange.getFilters

Метод возвращает фильтры столбцов, которые находятся в текущем диапазоне фильтрации.

Вызов

```
public TableFilters getFilters()
```

Возвращает

– фильтры столбцов, тип TableFilters.

– null, если диапазон фильтрации недействителен.

Пример

```
FiltersRange filtersRange = sheet.createFiltersRange(cellRange);
// ...
TableFilters filters = filtersRange.getFilters();
```

6.94.5 Метод FiltersRange.setFilters

Метод FiltersRange.setFilters устанавливает фильтры TableFilters для столбцов диапазона. Фильтрация выполняется с использованием логической операции AND по столбцам. Нумерация столбцов начинается относительно левой позиции диапазона фильтрации. Если номер столбца в фильтрах превышает диапазон фильтрации, то фильтр будет успешно добавлен, но фильтрация для этого столбца будет пропущена. Такое поведение полезно, если необходимо изменить размер диапазона фильтрации при этом сохранить ранее определенные фильтры. Диапазон фильтрации должен существовать до вызова этого метода. В настоящее время DocumentAPI позволяет создавать диапазон фильтрации только для всего листа табличного документа. Для создания или изменения размера существующего диапазона используется метод Table.createFiltersRange().

Вызов

```
public void setFilters(TableFilters filter)
```

Метод использует параметр типа TableFilters.

При использовании данного метода может произойти исключение InvalidObjectError в случае, если диапазон фильтрации недействителен.

Пример

```
FiltersRange filtersRange = sheet.createFiltersRange(cellRange);
.....................................
TableFilters tableFilters = new TableFilters();
tableFilters.setFilter(0, johnPaulFilter);
tableFilters.setFilter(1, songFilter);
.....................................
filtersRange.setFilters(tableFilters);
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.95 Класс FootnoteEndnote

Класс `FootnoteEndnote` представляет собой сноску в текстовом документе.

6.95.1 Метод FootnoteEndnote.getPosition

Метод возвращает позицию сноски в тексте.

Вызов

```
public Position getPosition()
```

Возвращает

– позиция сноски в тексте, тип `Position`.

Пример

```
Range range = document.getRange();
FootnotesEndnotes notes = range.getFootnotesEndnotes();
foreach (FootnoteEndnote note in notes)
    note.getPosition().removeForward();
```

6.95.2 Метод FootnoteEndnote.getRange

Метод возвращает диапазон содержимого сноски.

Вызов

```
public Range getRange()
```

Возвращает

– диапазон содержимого сноски, тип `Range`.

Пример

```
Range range = document.getRange();
FootnotesEndnotes notes = range.getFootnotesEndnotes();
foreach (FootnoteEndnote note in notes)
    Console.WriteLine(note.getRange().extractText());
```

6.95.3 Метод FootnoteEndnote.getType

Метод возвращает тип текущей сноски.

Вызов

```
public FootnoteEndnoteType getType()
```

Возвращает

– тип сноски, тип FootnoteEndnoteType.

6.96 Перечисление FootnoteEndnoteType

Перечисление FootnoteEndnoteType содержит типы сносок в текстовом документе. Используется в методе FootnoteEndnote.getType().

Таблица 56 – Описание типов сносок

Значение	Описание
FootnoteEndnoteType.Footnote	Сноска
FootnoteEndnoteType.Endnote	Концевая сноска

6.97 Класс FootnotesEndnotes

Класс FootnotesEndnotes представляет собой коллекцию сносок и концевых сносок. Используется в методе Range.getFootnotesEndnotes().

Пример

```
Range range = document.getRange();
FootnotesEndnotes notes = range.getFootnotesEndnotes();
foreach (FootnoteEndnote note in notes) {
    Console.WriteLine(note.getType());
    Console.WriteLine(note.getRange().extractText());
}
```

6.98 Перечисление FormulasPastingPolicy

Перечисление FormulasPastingPolicy содержит режимы копирования формул в ячейках. Используется в качестве поля formulasPastingPolicy класса CellRangePastingSettings.

Таблица 57 – Режимы копирования формул

Значение	Описание
FormulasPastingPolicy.AsFormulas	Копирует сами формулы

Значение	Описание
`FormulasPastingPolicy.AsValues`	Копирует результаты формул

6.99 Перечисление FormulaType

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 58. Используется в `Document.getFormulaType()`, `Document.setFormulaType()`, `DocumentSettings`.

Таблица 58 – Системы адресации ячеек в табличном документе

Значение	Описание	Пример
`FormulaType.A1`	Наиболее распространенная система адресации ячеек, при которой столбцы задаются буквами, а строки – числами	`='Лист1'!$D$20:$F$25`
`FormulaType.R1C1`	Альтернативная система адресации ячеек, при которой столбцы и строки задаются числами	`='Лист1'!R20C4:R25C6`
`FormulaType.OpenFormula`	Система адресации ячеек в рамках стандарта ODF	`=['Лист1'.$D$20:.$F$25]`

6.100 Класс FractionCellFormatting

Класс `FractionCellFormatting` содержит параметры для дробного формата ячеек таблицы. Используется в качестве аргумента метода `Cell.setFormat()`.

Таблица 59 – Описание полей класса `FractionCellFormatting`

Поле	Тип	Описание
`FractionCellFormatting.minNumeratorDigits`	`int`	Количество позиций числителя
`FractionCellFormatting.minDenominatorDigits`	`int`	Количество позиций знаменателя
`FractionCellFormatting.denominatorValue`	`int`	Знаменатель

Пример

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

FractionCellFormatting cellFormat = new FractionCellFormatting();
cellFormat.denominatorValue = 22;
cellFormat.minDenominatorDigits = 3;
cellFormat.minNumeratorDigits = 2;

cell.setFormat(cellFormat);
Console.WriteLine(cell.getFormattedValue());
```

6.101 Класс Frame

Класс `Frame` описывает прямоугольную область графического объекта документа. Предназначен для получения и изменения свойств графических объектов. Для расположения в позиции текста документа используется `InlineFrame`, в таблице - `AbsoluteFrame` (см. Рисунок 2).

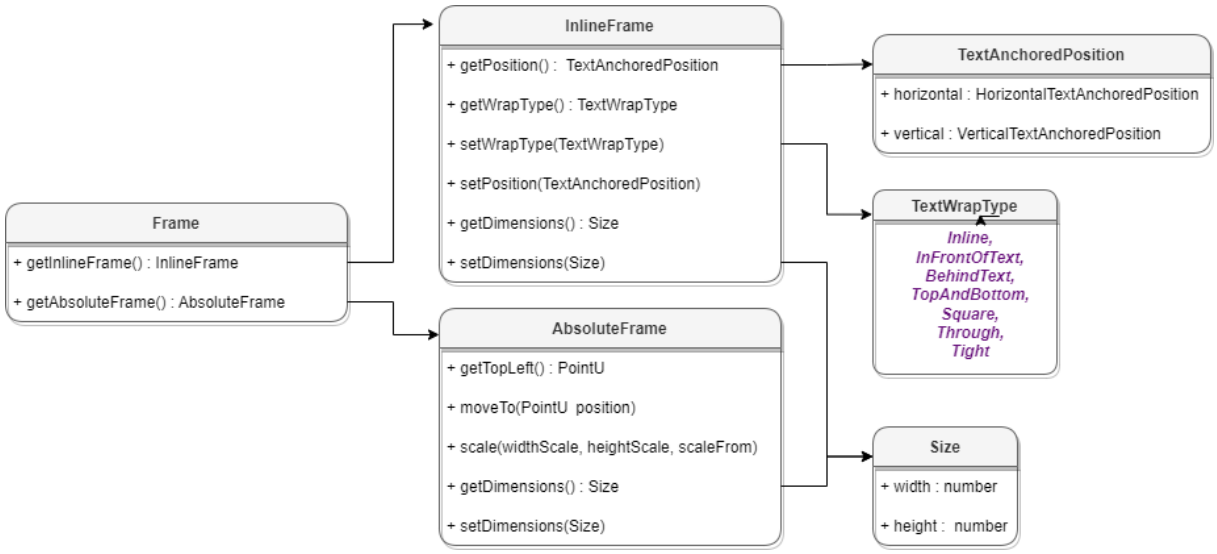


Рисунок 46 – Объектная модель класса `Frame`

Пример

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator enumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in enumerator)
```

```
{
    var frame = mediaObject.getFrame();
    var absoluteFrame = frame.getAbsoluteFrame();
    if (absoluteFrame != null) {
       Console.WriteLine(absoluteFrame.getDimensions().width);
    }
    var inlineFrame = frame.getInlineFrame();
    if (inlineFrame != null) {
       Console.WriteLine(inlineFrame.getWrapType());
    }
}
```

6.101.1 Метод Frame.getAbsoluteFrame

Метод возвращает тип `AbsoluteFrame` если объект представляет данный тип, либо `null` в противном случае.

Пример

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator enumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in enumerator)
{
    var frame = mediaObject.getFrame();
    var absoluteFrame = frame.getAbsoluteFrame();
    if (absoluteFrame != null) {
       Console.WriteLine(absoluteFrame.getDimensions().width);
    }
}
```

6.101.2 Метод Frame.getInlineFrame

Метод возвращает тип `InlineFrame` если объект представляет данный тип, либо `null` в противном случае.

Пример

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator enumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in enumerator)
```

```
{
    var inlineFrame = frame.getInlineFrame();
    if (inlineFrame != null) {
       Console.WriteLine(inlineFrame.getWrapType());
    }
}
```

6.102 Класс FrozenRangePosition

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

6.102.1 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 frozenRangePosition = new FrozenRangePosition();
Console.WriteLine(frozenRangePosition.isRowsCols());
```

```
FrozenRangePosition frozenRangePosition = new FrozenRangePosition(0, 2, 5, 5);
Console.WriteLine(frozenRangePosition.isRowsCols());
```

6.102.2 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 frozenRangePosition = FrozenRangePosition.create(0, 2, 5, 5);
Console.WriteLine(frozenRangePosition.isRowsCols());
```

6.102.3 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 = FrozenRangePosition.createFrozenArea(0, 2);
Console.WriteLine(frozenRangePosition.isArea());
```

6.102.4 Метод FrozenRangePosition.createFrozenCols

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

Вызов

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

Пример

```
frozenRangePosition = FrozenRangePosition.createFrozenCols(0, 2);
Console.WriteLine(frozenRangePosition.isRowsCols());
```

6.102.5 Метод FrozenRangePosition.createFrozenRows

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

Вызов

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

Пример

```
frozenRangePosition = FrozenRangePosition.createFrozenRows(0, 2);
Console.WriteLine(frozenRangePosition.isRowsCols());
```

6.102.6 Метод FrozenRangePosition.isArea

Возвращает `true` если диапазон является непрерывной областью.

Пример

```
FrozenRangePosition frozenRangePosition = FrozenRangePosition.createFrozenArea(2,
 2);
Console.WriteLine(frozenRangePosition.isArea());
```

6.102.7 Метод FrozenRangePosition.isCols

Возвращает `true` если диапазон состоит из колонок.

Пример

```
FrozenRangePosition frozenRangePosition = FrozenRangePosition.createFrozenCols(0,
 2);
Console.WriteLine(frozenRangePosition.isCols());
```

6.102.8 Метод FrozenRangePosition.isRows

Возвращает `true` если диапазон состоит из строк.

Пример

```
FrozenRangePosition frozenRangePosition = FrozenRangePosition.createFrozenRows(0,
 2);
Console.WriteLine(frozenRangePosition.isRows());
```

6.102.9 Метод FrozenRangePosition.isRowsCols

Возвращает `true` если диапазон содержит строки и колонки.

Пример

```
FrozenRangePosition frozenRangePosition = FrozenRangePosition.createFrozenArea(2,
 2);
Console.WriteLine(frozenRangePosition.isRowsCols());
```

6.103 Класс HeaderFooter

Класс `HeaderFooter` определяет колонтитул текстового документа.

6.103.1 Метод HeaderFooter.getBlocks

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

Пример

```
foreach (var headerFooter in headersEnumerator)
{
    Blocks blocks = headerFooter.getBlocks();
    BlocksEnumerator blocksEnumerator = blocks.GetEnumerator();
    foreach (var block in blocksEnumerator)
    {
        Console.WriteLine(block.getRange().extractText());
    }
}
```

6.103.2 Метод HeaderFooter.getRange

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

Пример

```
foreach (var headerFooter in headersEnumerator)
{
    Range range = headerFooter.getRange();
    Console.WriteLine(range.extractText());
}
```

6.103.3 Метод HeaderFooter.getType

Метод предоставляет информацию о типе колонтитула (HeaderFooterType).

Пример

```
foreach (var headerFooter in headersEnumerator)
{
    HeaderFooterType headerFooterType = headerFooter.getType();
    Console.WriteLine(headerFooterType);
}
```

6.104 Перечисление HeaderFooterType

Перечисление `HeaderFooterType` содержит типы колонтитулов. Используется в методе `HeaderFooter.getType()`.

Таблица 60 – Типы колонтитулов

Значение	Описание
`HeaderFooterType.Header`	Верхний колонтитул
`HeaderFooterType.Footer`	Нижний колонтитул

6.105 Класс HeadersFooters

Класс `HeadersFooters` представляет коллекцию верхних и нижних колонтитулов раздела (см. Рисунок 2). Доступ к колонтитулам осуществляется посредством методов `Section.getHeaders()`, `Section.getFooters()`.

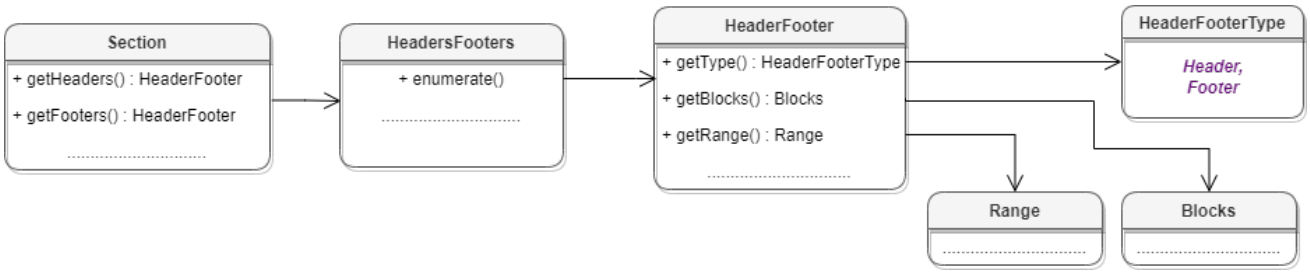


Рисунок 47 – Классы для работы с колонтитулами

6.105.1 Метод HeadersFooters.GetEnumerator

Метод возвращает коллекцию колонтитулов.

Пример

```
Block block = document.getBlocks().getBlock(0);
if (block != null)
{
    Section section = block.getSection();
    HeadersFooters headersFooters = section.getHeaders();
    HeaderFootersEnumerator headersEnumerator = headersFooters.GetEnumerator();
    foreach (var header in headersEnumerator)
```

```
    {
        Console.WriteLine(header.getType());
    }
}
```

6.106 Перечисление HorizontalAnchorAlignment

В таблице 61 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали (см. описание класса HorizontalTextAnchoredPosition).

Таблица 61 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

Значение	Описание
HorizontalAnchorAlignment.Left	По верхнему краю
HorizontalAnchorAlignment.Right	По нижнему краю
HorizontalAnchorAlignment.Center	По центру
HorizontalAnchorAlignment.Inside HorizontalAnchorAlignment.Outside	По границам

6.107 Перечисление HorizontalRelativeTo

В таблице 62 представлены типы размещения объекта относительно закрепленной позиции по горизонтали (см. описание класса HorizontalTextAnchoredPosition).

Таблица 62 – Типы размещения объекта относительно закрепленной позиции по горизонтали

Значение	Описание
HorizontalRelativeTo.Character	Символ
HorizontalRelativeTo.Column	Столбец
HorizontalRelativeTo.ColumnLeftMargin	Левое поле столбца
HorizontalRelativeTo.ColumnRightMargin	Правое поле столбца
HorizontalRelativeTo.ColumnInsideMargin	Внутреннее поле столбца
HorizontalRelativeTo.ColumnOutsideMargin	Внешнее поле столбца
HorizontalRelativeTo.Page	Страница
HorizontalRelativeTo.PageContent	Содержимое страницы
HorizontalRelativeTo.PageLeftMargin	Левое поле страницы

Значение	Описание
`HorizontalRelativeTo.PageRightMargin`	Правое поле страницы
`HorizontalRelativeTo.PageInsideMargin`	Внутреннее поле страницы
`HorizontalRelativeTo.PageOutsideMargin`	Внешнее поле страницы

6.108 Класс HorizontalTextAnchoredPosition

Класс `HorizontalTextAnchoredPosition` предназначен для управления относительным положением объекта со смещением или выравниванием по горизонтали (см. описание класса `TextAnchoredPosition`).

Таблица 63 – Описание полей класса `HorizontalTextAnchoredPosition`

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

6.109 Класс HtmlFragments

Класс `HtmlFragments` содержит HTML элементы, сгенерированные в результате работы метода `DocumentAPI.exportWorksheetToHtml()`.

Таблица 64 – Описание полей класса `HtmlFragments`

Поле	Тип	Описание
`HtmlFragments.rootCss`	`string`	Содержит CSS стили для сгенерированных элементов и соответствующие селекторы
`HtmlFragments.body`	`string`	Содержит сгенерированные HTML элементы внутри div контейнера с `class="myoffice-worksheet"`

6.110 Класс Hyperlink

Класс `Hyperlink` описывает свойства ссылки. Объект `Hyperlink` может быть получен посредством вызова метода `Cell.getHyperlink()`.

Таблица 65 – Описание полей класса `Hyperlink`

Поле	Тип	Описание
`Hyperlink.url`	`string`	Адрес ссылки
`Hyperlink.tooltip`	`string`	Текст подсказки
`Hyperlink.label`	`string`	Текст описания

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A3");
Hyperlink hyperlink = cell.getHyperlink();
Console.WriteLine($"{hyperlink.url} {hyperlink.tooltip} {hyperlink.label}");
```

6.111 Класс IconSetConditionalFormatOperator

Класс `IconSetConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Значки". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

Конструктор

```
public IconSetConditionalFormatOperator(ConditionalFormatIconSetEntries entries, bool isValueShown)
```

Пример

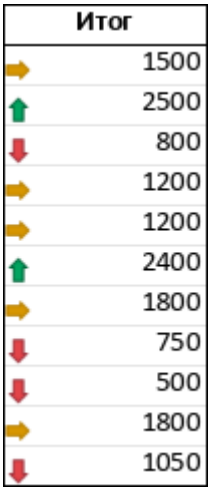


Рисунок 48 – Пример создания правила "Значки"

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

var entries = new ConditionalFormatIconSetEntries();
var value1 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "0",
false);
var entry1 = new ConditionalFormatIconSetEntry(value1,
ConditionalFormatIconSet.ThreeArrows, 0);
entries.addEntry(entry1);
var value2 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "33",
false);
var entry2 = new ConditionalFormatIconSetEntry(value2,
ConditionalFormatIconSet.ThreeArrows, 1);
entries.addEntry(entry2);
var value3 = new
ConditionalFormatValueObject(ConditionalFormatValueObjectType.Percent, "67",
false);
var entry3 = new ConditionalFormatIconSetEntry(value3,
ConditionalFormatIconSet.ThreeArrows, 2);
entries.addEntry(entry3);

var iconSetOperator = DocumentAPI.createIconSetConditionalFormatOperator(entries,
true);

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

var iconSetRule = new ConditionalFormatRule();
iconSetRule.setOperator(iconSetOperator);
iconSetRule.setRange(cellRangePosition);
rules.addRule(iconSetRule);
```

6.111.1 Метод IconSetConditionalFormatOperator.getEntries

Метод возвращает набор правил отображения значков для текущего оператора.

Вызов

```
public ConditionalFormatIconSetEntries getEntries()
```

Возвращает

– набор правил отображения значков, тип ConditionalFormatIconSetEntries.

6.111.2 Метод IconSetConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.111.3 Метод IconSetConditionalFormatOperator.isValueShown

Метод позволяет определить, показывается ли значение ячейки при примененном условном форматировании.

Вызов

```
public bool isValueShown()
```

Возвращает

– `true`, если значение ячейки показывается при примененном форматировании, в ином случае – `false`.

6.111.4 Метод IconSetConditionalFormatOperator.setEntries

Метод задает набор правил отображения значков для текущего оператора.

Вызов

```
public void setEntries(ConditionalFormatIconSetEntries entries)
```

Параметры

– `entries`: набор правил отображения значков, тип ConditionalFormatIconSetEntries.

6.111.5 Метод IconSetConditionalFormatOperator.setValueShown

Метод позволяет настроить отображение значения ячейки при примененном условном форматировании.

Вызов

```
public void setValueShown(bool isValueShown)
```

Параметры

– `isValueShown`: `true`, чтобы показывать значение ячейки при примененном форматировании, в ином случае – `false`.

6.112 Класс Image

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

6.112.1 Метод Image.getFrame

Метод аналогичен методу `InlineObject.getFrame()`, он возвращает свойства позиции изображения типа `Frame`.

Пример для текстового документа

```
var images = document.getRange().getImages();
var imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    Console.WriteLine(image.getFrame());
}
```

6.112.2 Метод Image.remove

Метод удаляет изображение из документа.

Пример для текстового документа

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator enumerator = mediaObjects.getEnumerator();
foreach (var mediaObject in enumerator) {
    Image image = mediaObject.toImage();
```

```
    if (image != null) {
        image.remove();
        break;
    }
}
```

6.112.3 Метод Image.replaceURL

Метод заменяет текущее изображение.

Вызов

```
public void replaceURL(string newURL)
```

Параметры

– `newURL`: путь к новому файлу изображения, тип `string`.

Пример

```
Range range = document.getRange();
Images images = range.getImages();
foreach (var image in images) {
    image.replaceURL("logo2025.png");
    image.getFrame().getAbsoluteFrame().setDimensions(new SizeU(150, 150));
}
```

6.113 Класс Images

Класс `Images` используется для доступа к коллекции изображений. Объект может быть получен в текстовом документе посредством вызова метода `Range.getImages()`.

Пример

```
var images = document.getRange().getImages();
var imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    Console.WriteLine(image); // NCT.MyOfficeSDK.Image
}
```

6.113.1 Метод Images.GetEnumerator

Метод позволяет перечислить коллекцию изображений.

Пример для текстового документа

```
var images = document.getRange().getImages();
var imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    Console.WriteLine(image); // NCT.MyOfficeSDK.Image
}
```

6.114 Класс InlineFrame

Класс `InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 2). Предназначен для получения и изменения свойств позиции графических объектов. Используется в текстовом документе. Может быть получен с помощью метода `Frame.getInlineFrame()`.

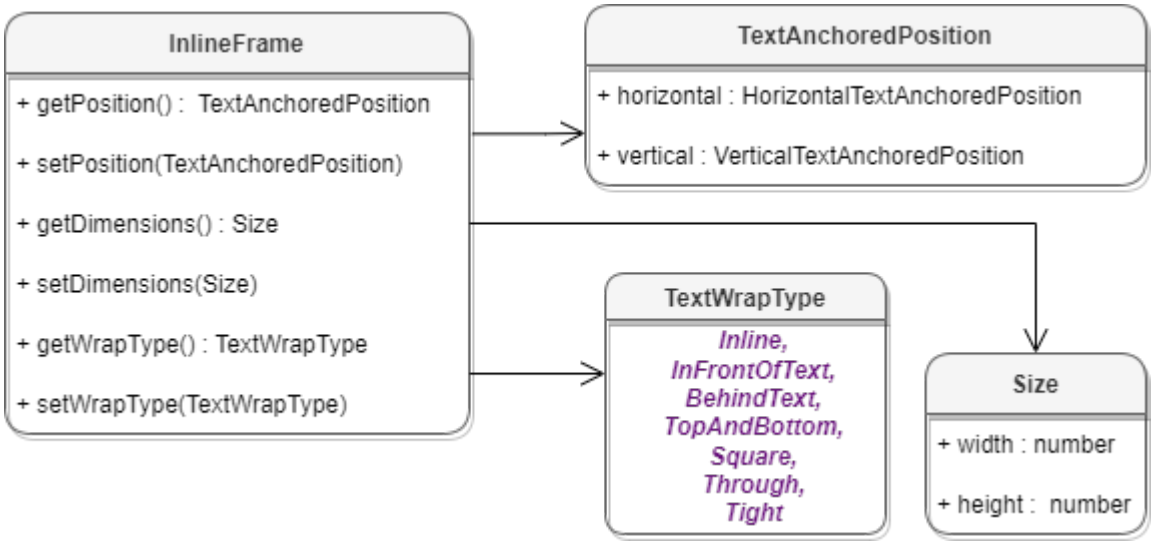


Рисунок 49 – Объектная модель класса `InlineFrame`

Пример для текстового документа

```
Range range = document.getRange();
MediaObjects mediaObjects = range.getInlineObjects();
MediaObjectsEnumerator mediaObjectsEnumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
```

```
{
    var frame = mediaObject.getFrame();
    var inlineFrame = frame.getInlineFrame();
    if (inlineFrame != null) {
       Console.WriteLine(inlineFrame.getDimensions().width);
    }
}
```

6.114.1 Метод InlineFrame.getDimensions

Метод возвращает задает размеры встроенного объекта, тип - SizeU.

Пример

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator enumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in enumerator)
{
    var frame = mediaObject.getFrame();
    var inlineFrame = frame.getInlineFrame();
    if (inlineFrame != null) {
       Console.WriteLine(inlineFrame.getDimensions().width);
    }
}
```

6.114.2 Метод InlineFrame.getPosition

Метод возвращает позицию встроенного объекта, тип TextAnchoredPosition.

Пример

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator enumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in enumerator)
{
    var frame = mediaObject.getFrame();
    var inlineFrame = frame.getInlineFrame();
    if (inlineFrame != null) {
       TextAnchoredPosition textAnchoredPosition = frame.getPosition();
       Console.WriteLine(textAnchoredPosition.horizontal.alignment);
```

```
    }
}
```

6.114.3 Метод InlineFrame.getWrapType

Метод возвращает вариант обтекания текстом встроенного объекта (см. TextWrapType).

Пример

```
Console.WriteLine(inlineFrame.getWrapType());
```

6.114.4 Метод InlineFrame.setDimensions

Метод позволяет задать размер встроенного объекта (тип SizeU).

Пример

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator enumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in enumerator)
{
    var frame = mediaObject.getFrame();
    var inlineFrame = frame.getInlineFrame();
    if (inlineFrame != null) {
        SizeU frameDimensions = new SizeU(50, 50);
        inlineFrame.setDimensions(frameDimensions);
        Console.WriteLine(inlineFrame.getDimensions().width);
    }
}
```

6.114.5 Метод InlineFrame.setPosition

Метод задает положение встроенного объекта, тип аргумента TextAnchoredPosition. Новая позиция может быть установлена только для встроенных объектов, у которых тип обтекания текстом не является типом TextWrapType.Inline.

Примеры

Для установки позиции стиль обтекания текстом должен отличаться от TextWrapType.Inline. Предварительно следует изменить его на другой тип.

```
var wrapType = inlineFrame.getWrapType();
if (wrapType == TextWrapType.Inline)
{
    inlineFrame.setWrapType(TextWrapType.TopAndBottom);
}
```

Используя классы HorizontalTextAnchoredPosition, VerticalTextAnchoredPosition, можно задать положение встроенных объектов в текстовом документе с учетом относительного смещения.

```
var position = new TextAnchoredPosition();

position.horizontal = new
HorizontalTextAnchoredPosition(HorizontalRelativeTo.Page, 12.0f);
position.vertical = new
VerticalTextAnchoredPosition(VerticalRelativeTo.PageTopMargin, 122.0f);

inlineFrame.setPosition(position);
```

Используя классы HorizontalTextAnchoredPosition, VerticalTextAnchoredPosition, можно задать положение встроенных объектов в текстовом документе с учетом относительного выравнивания.

```
var position = new TextAnchoredPosition();

position.horizontal = new
HorizontalTextAnchoredPosition(HorizontalRelativeTo.Page,
HorizontalAnchorAlignment.Center);
position.vertical = new
VerticalTextAnchoredPosition(VerticalRelativeTo.PageTopMargin,
VerticalAnchorAlignment.Top);

inlineFrame.setPosition(position);
```

Используя типы смещения HorizontalRelativeTo.Column и VerticalRelativeTo.Page, можно установить абсолютное положение встроенного объекта в текстовом документе.

```
var position = new TextAnchoredPosition();

position.horizontal = new
HorizontalTextAnchoredPosition(HorizontalRelativeTo.Column, 125.f);
position.vertical = new VerticalTextAnchoredPosition(VerticalRelativeTo.Page,
345.f);

inlineFrame.setPosition(position);
```

6.114.6 Метод InlineFrame.setWrapType

Метод устанавливает вариант обтекания текстом встроенного объекта (см. TextWrapType).

Пример

```
inlineFrame.setWrapType(TextWrapType.Inline);
Console.WriteLine(inlineFrame.getWrapType());
```

6.115 Класс InputFieldControl

Представляет собой элемент управления "Поле ввода" в документе. Является наследником класса ContentControl. Методы `InputFieldControl.getValue()` и `InputFieldControl.setValue(string)` позволяют получить и задать значение этого элемента управления.

Пример

```
ContentControls controls = document.getContentControls();
InputFieldControl inputField = controls.findByTitle("input").toInputField();
inputField.setValue(inputField.getValue().Replace(' ', '_'));
```

6.116 Класс Insets

Класс `Insets` предназначен для задания полей, например, страницы. Используется в поле `margins` класса PageProperties.

Таблица 66 – Описание полей класса `Insets`

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

Пример

```
PageProperties pageProperties = new PageProperties();
Insets insets = new Insets();
insets.left = 0;
insets.top = 0;
insets.right = 100;
insets.bottom = 100;
pageProperties.margins = insets;
document.setPageProperties(pageProperties);
```

6.117 Класс LineEndingProperties

Класс `LineEndingProperties` содержит варианты оформления окончаний линий. Используется в полях `headLineEndingProperties` и `tailLineEndingProperties` класса `LineProperties`.

Таблица 67 – Описание полей класса `LineEndingProperties`

Поле	Тип	Описание
`LineEndingProperties.style`	`LineEndingStyle`	Стиль окончания линии
`LineEndingProperties.relativeExtent`	`SizeU`	Размер окончания линии относительно ее ширины

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("C3");

LineProperties lineProperties = new LineProperties();
```

```
lineProperties.headLineEndingProperties = new LineEndingProperties();
lineProperties.headLineEndingProperties.style = LineEndingStyle.Arrow;
lineProperties.headLineEndingProperties.relativeExtent = new SizeU(2, 2);

lineProperties.tailLineEndingProperties = new LineEndingProperties();
lineProperties.tailLineEndingProperties.style = LineEndingStyle.Arrow;
lineProperties.tailLineEndingProperties.relativeExtent = new SizeU(2, 2);

lineProperties.style = LineStyle.Solid;
lineProperties.width = 1.5f;
lineProperties.color = new Color(new ColorRGBA(55, 146, 179, 200));

Borders borders = new Borders();
borders.setTop(lineProperties);
cell.setBorders(borders);
```

6.118 Перечисление LineEndingStyle

В таблице 68 приведены типы окончания линии. Используется в поле `style` класса `LineEndingProperties`.

Таблица 68 – Типы окончания линии

Значение	Описание
`LineEndingStyle.Arrow`	
`LineEndingStyle.Diamond`	
`LineEndingStyle.Oval`	
`LineEndingStyle.Stealth`	
`LineEndingStyle.Triangle`	
`LineEndingStyle.None`	

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("C3");
```

```
LineProperties lineProperties = new LineProperties();
lineProperties.headLineEndingProperties = new LineEndingProperties();
lineProperties.headLineEndingProperties.style = LineEndingStyle.Arrow;

Borders borders = new Borders();
borders.setTop(lineProperties);
cell.setBorders(borders);
```

6.119 Класс LineProperties

Класс `LineProperties` предназначен для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 2).

Таблица 69 – Описание полей класса `LineProperties`

Поле	Тип	Описание
`LineProperties.style`	`LineStyle`	Тип линии
`LineProperties.width`	`float`	Толщина линии
`LineProperties.color`	`Color`	Цвет линии
`LineProperties.headLineEndingProperties`	`LineEndingProperties`	Тип начала линии
`LineProperties.tailLineEndingProperties`	`LineEndingProperties`	Тип окончания линии

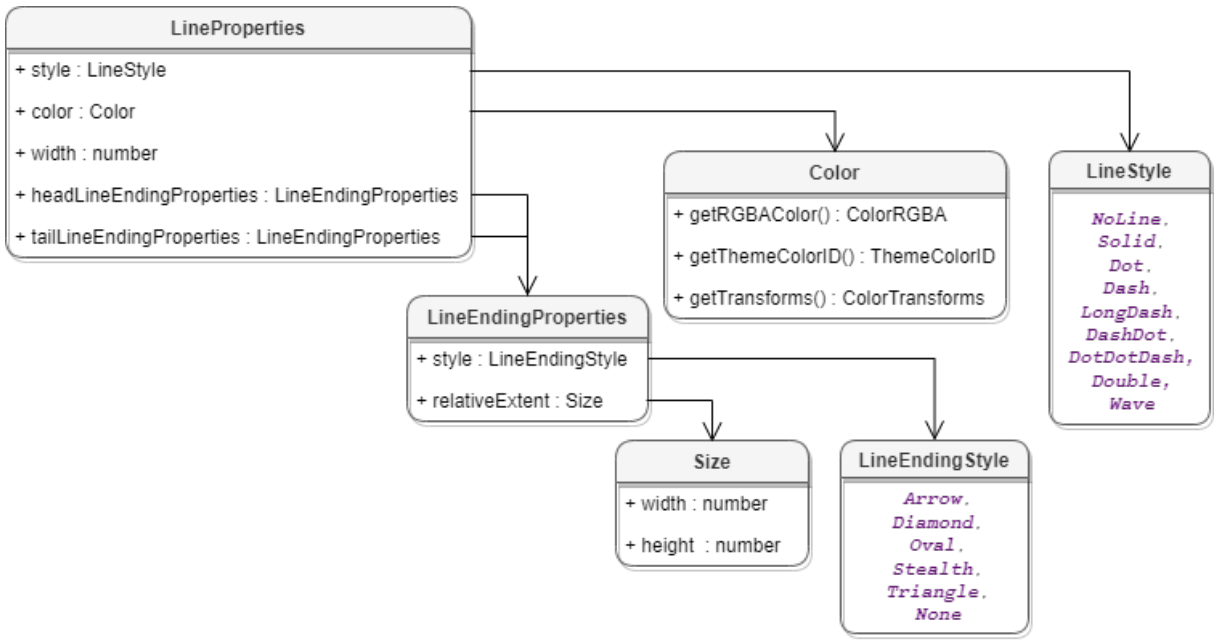


Рисунок 50 – Свойства границ ячеек

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("C3");

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Solid;
lineProperties.width = 1.5f;
lineProperties.color = new Color(new ColorRGBA(55, 146, 179, 200));

Borders borders = new Borders();
borders.setTop(lineProperties);
cell.setBorders(borders);
```

6.120 Класс LineSpacing

Класс `LineSpacing` задает межстрочный интервал абзаца. Для управления значением межстрочного интервала используются значения, представленные в разделе `LineSpacingRule`.

Таблица 70 – Параметры межстрочного интервала

Поле	Тип	Описание
`LineSpacing.value`	`float`	Значение межстрочного интервала
`LineSpacing.rule`	`LineSpacingRule`	Правило формирования межстрочного интервала

Пример

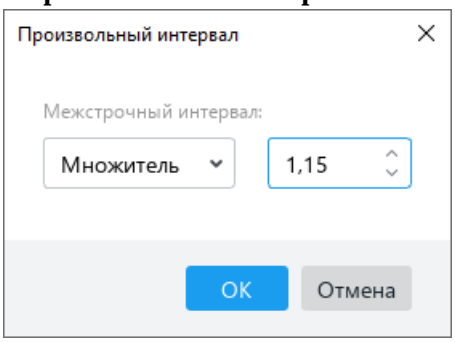
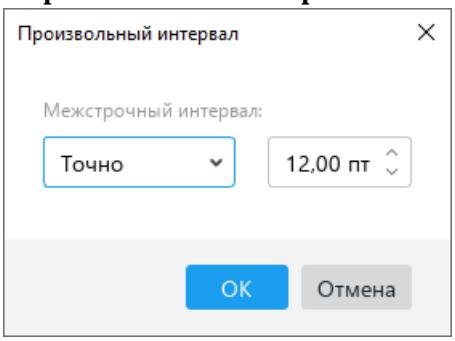
```
Paragraph paragraph = paragraphsEnumerator.Current;
ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
// Конструктор
paragraphProperties.lineSpacing = new LineSpacing(5.0f,
LineSpacingRule.Multiple);
// Обращение к полям
paragraphProperties.lineSpacing.value = 1;
paragraphProperties.lineSpacing.rule = LineSpacingRule.Exact;
```

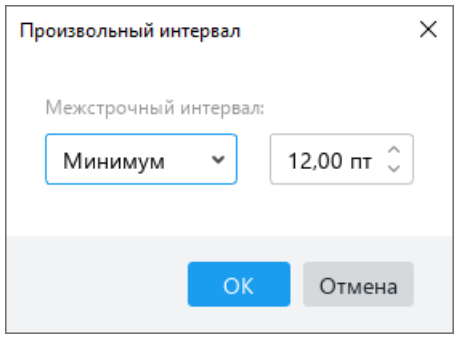
6.121 Перечисление LineSpacingRule

Перечисление `LineSpacingRule` содержит типы межстрочного интервалов. В таблице 71 представлены описания правил формирования межстрочного интервала текстового абзаца.

Таблица 71 – Виды межстрочного интервала

Значение	Описание
`LineSpacingRule.Multiple`	Установка произвольного межстрочного интервала с использованием множителя. При вызове необходимо указать значение множителя, например: `pPr.lineSpacing= new LineSpacing(1.15f, LineSpacingRule.Multiple)` В данном примере используется значение множителя **1.15**.

Значение	Описание
	Действие команды аналогично ручной настройке межстрочного интервала в диалоге «**Произвольный интервал**». 
`LineSpacingRule.Exact`	Установка произвольного межстрочного интервала с использованием точного значения. При вызове необходимо указать точное значение, например: `pPr.lineSpacing= new LineSpacing(12.0f, LineSpacingRule.Exact)` В данном примере используется точное значение **12.0**. Действие команды аналогично ручной настройке межстрочного интервала в диалоге «**Произвольный интервал**». 
`LineSpacingRule.AtLeast`	Установка произвольного межстрочного интервала с использованием минимального значения. При вызове необходимо указать минимальное значение, например: `pPr.lineSpacing= new LineSpacing(12.0f, LineSpacingRule.AtLeast)` В данном примере используется минимальное значение **12.0**. Действие команды аналогично ручной настройке межстрочного интервала в диалоге «**Произвольный интервал**».

Значение	Описание
	

Пример

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    ParagraphProperties paraProps = paragraph.getParagraphProperties();
    paraProps.lineSpacing = new LineSpacing(5.0f, LineSpacingRule.Multiple);
    paragraph.setParagraphProperties(paraProps);
}
```

6.122 Перечисление LineStyle

В таблице 72 приведены типы линий. Используется в поле `style` класса `LineProperties`.

Таблица 72 – Типы линий

Значение	Описание
`LineStyle.NoLine`	Нет линии
`LineStyle.Solid`	
`LineStyle.Dot`	
`LineStyle.Dash`	
`LineStyle.LongDash`	
`LineStyle.DashDot`	

Значение	Описание
LineStyle.DotDotDash	
LineStyle.Double	
LineStyle.Wave	

Пример

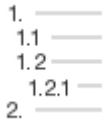
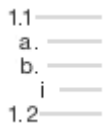
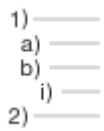
```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("C3");

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Solid;
```

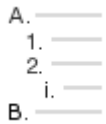
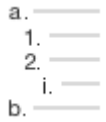
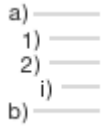
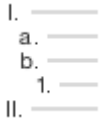
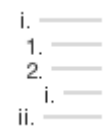
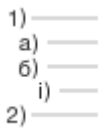
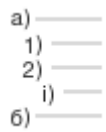
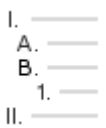
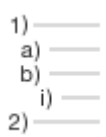
6.123 Перечисление ListSchema

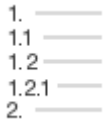
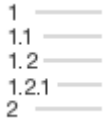
Перечисление ListSchema содержит типы схем форматирования списков, которые могут быть применены к абзацам текста. Данные значения используются в методах Paragraph.getListSchema(), Paragraph.setListSchema().

Таблица 73 – Описания схем форматирования списков

Значение	Тип схемы списка	Изображение
ListSchema. Unknown	Неизвестно	
ListSchema. UnknownBullet	Список без маркера	Соответствует варианту «нет»
ListSchema. UnknownNumbering	Нумерация без номера	Соответствует варианту «нет»
ListSchema. BulletCircleSolid	Список с маркерами в виде круга	

Значение	Тип схемы списка	Изображение
`ListSchema.BulletCircleContour`	Список с маркерами в виде окружности	
`ListSchema.BulletSquareSolid`	Список с маркерами в виде квадрата	
`ListSchema.BulletDiamondDots`	Список с маркерами в виде четырех ромбов	
`ListSchema.BulletHyphen`	Список с маркерами в виде дефиса	
`ListSchema.BulletConcaveArrowSolid`	Список с маркерами в виде вогнутой стрелки	
`ListSchema.BulletCheckmark`	Список с маркерами в виде галочки	
`ListSchema.EnumeratorDecimalDot`	Десятичная нумерация с точкой	
`ListSchema.EnumeratorDecimalDotMultiLevel`	Многоуровневая десятичная нумерация с точкой	
`ListSchema.EnumeratorDecimalBracket`	Десятичная нумерация со скобкой	

Значение	Тип схемы списка	Изображение
`ListSchema.EnumeratorLatinUppercaseDot`	Нумерация латинскими прописными буквами с точкой	
`ListSchema.EnumeratorLatinLowercaseDot`	Нумерация латинскими строчными буквами с точкой	
`ListSchema.EnumeratorLatinLowercaseBracket`	Нумерация латинскими строчными буквами со скобкой	
`ListSchema.EnumeratorRomanUppercaseDot`	Нумерация римскими прописными цифрами с точкой	
`ListSchema.EnumeratorRomanLowercaseDot`	Нумерация римскими строчными цифрами с точкой	
`ListSchema.EnumeratorDecimalRussianBracket`	Десятичная нумерация через запятую со скобкой	
`ListSchema.EnumeratorRussianLowercaseBracket`	Нумерация с русскими строчными буквами со скобкой	
`ListSchema.EnumeratorOutlineDefault`	Нумерация римскими прописными цифрами с точкой	
`ListSchema.EnumeratorNumberValue`	Десятичная нумерация со скобкой, после третьего уровня вложенности – нумерация в скобках	

Значение	Тип схемы списка	Изображение
ListSchema.EnumeratorDecimalDotMultiLevelHeading	Десятичная нумерация без левого отступа с точкой	
ListSchema.EnumeratorDecimalDotNoTrailingDotMultiLevelHeading	Десятичная нумерация без левого отступа	

Пример

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.setListSchema(ListSchema.BulletCheckmark);
    Console.WriteLine(paragraph.getListSchema());
}
```

6.124 Класс LoadDocumentSettings

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

Таблица 74 – Описание полей класса `LoadDocumentSettings`

Поле	Тип	Описание
LoadDocumentSettings.commonDocumentSettings	DocumentSettings	Общие настройки документа
LoadDocumentSettings.encoding	Encoding	Кодировка документа
LoadDocumentSettings.dsvSettings	DSVSettings	Настройки для работы с файлами CSV и DSV
LoadDocumentSettings.documentPassword	string	Пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows.

6.125 Класс LocaleInfo

Класс `LocaleInfo` предоставляет информацию о локализации. Используется в поле `localeInfo` класса `DocumentSettings`.

Таблица 75 – Описание полей класса `LocaleInfo`

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

6.126 Класс MediaObject

Класс `MediaObject` представляет собой встроенный объект документа. Может быть получен посредством использования метода `MediaObjects.getEnumerator()`.

6.126.1 Метод MediaObject.getFrame

Метод возвращает свойства позиции встроенного объекта. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют класс `Frame`.

Пример для текстового документа

```
var mediaObjectsEnumerator =
document.getRange().getInlineObjects().GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    var frame = mediaObject.getFrame();
    ..................................
}
```

6.126.2 Метод MediaObject.toChart

Метод возвращает диаграмму `Chart`, связанную со встроенным объектом текстового документа. Если объект не является изображением, метод возвращает `null`.

Пример

```
var mediaObjectsEnumerator =
document.getRange().getInlineObjects().GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    var image = mediaObject.toImage();
    if (image != null)
    {
        Console.WriteLine("Текущий объект является изображением");
    }
    else
    {
        Console.WriteLine("Текущий объект является фигурой");
```

```
    }
}
```

6.126.3 Метод MediaObject.toImage

Метод возвращает изображение Image, связанное со встроенным объектом. Если объект не является изображением, метод возвращает `null`.

Пример для текстового документа

```
var mediaObjectsEnumerator =
document.getRange().getInlineObjects().GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    var image = mediaObject.toImage();
    if (image != null)
    {
        Console.WriteLine("Текущий объект является изображением");
    }
    else
    {
        Console.WriteLine("Текущий объект является фигурой");
    }
}
```

Пример для табличного документа

```
var mediaObjectsEnumerator =
document.getRange().getInlineObjects().GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    var image = mediaObject.toImage();
    if (image != null)
    {
        Console.WriteLine("Текущий объект является изображением");
    }
    else
    {
        Console.WriteLine("Текущий объект является фигурой");
```

```
    }
}
```

6.127 Класс MediaObjects

Класс `MediaObjects` предназначен для доступа к коллекции графических объектов в текстовом документе. Объект может быть получен вызовом методов `Range.getInlineObjects()`, `Table.getMediaObjects()` (см. Рисунок 2).

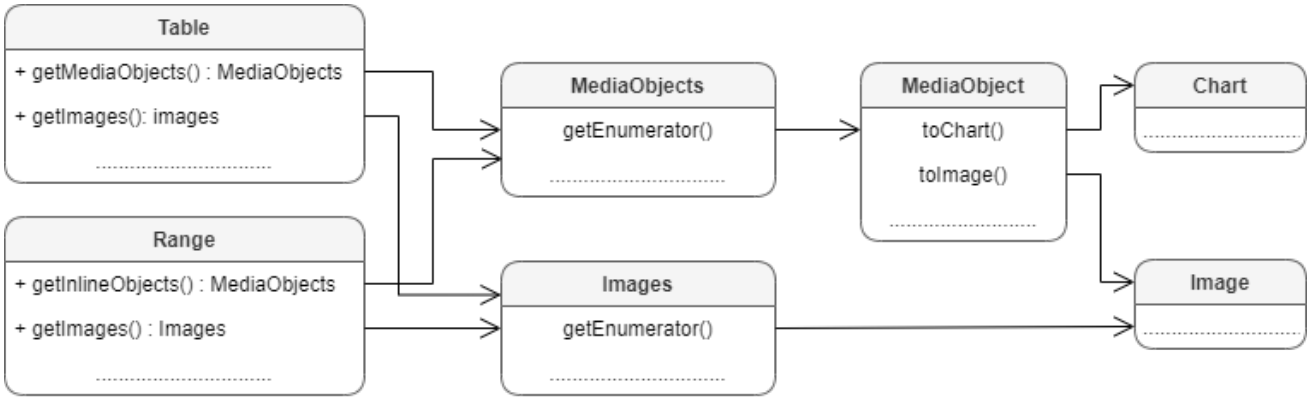


Рисунок 51 – Встроенные объекты

6.127.1 Метод MediaObjects.getEnumerator

Метод позволяет перечислить коллекцию встроенных объектов в текстовом документе.

Пример для текстового документа

```
MediaObjects mediaObjects = document.getRange().getInlineObjects();
MediaObjectsEnumerator mediaObjectsEnumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    .....................................
}
```

6.128 Класс Message

Класс `Message` предназначен для формирования событий лога.

6.128.1 Перечисление Message.Severity

Перечисление `Message.Severity` (Таблица 76) описывает уровни сообщений лога (информация, предупреждение, ошибка).

Таблица 76 – Описание уровней лога `Message.Severity`

Значение	Описание
`Message.Severity.Info`	Информация
`Message.Severity.Warning`	Предупреждение
`Message.Severity.Error`	Ошибка

6.128.2 Метод Message.getSeverity

Метод возвращает уровень лога `Message.Severity`.

6.128.3 Метод Message.getText

Метод возвращает текст сообщения.

6.128.4 Метод Message.makeError

Метод создает сообщение типа `Message.Severity.Error` с заданным текстом.

6.128.5 Метод Message.makeInfo

Метод создает сообщение типа `Message.Severity.Info` с заданным текстом.

6.128.6 Метод Message.makeWarning

Метод создает сообщение типа `Message.Severity.Warning` с заданным текстом.

6.129 Класс Messenger

Служит для организации логов приложений.

6.129.1 Метод Messenger.notify

Метод используется для создания события лога

Пример

```
MessageHandler messageHandler = new MessageHandler();
Messenger messenger= application.getMessenger();
messenger.notify(Message.makeWarning("Warning"));
```

6.129.2 Метод Messenger.subscribe

Метод служит для подписки на события лога.

Пример

```
MessageHandler messageHandler = new MessageHandler();
Messenger messenger= application.getMessenger();
messenger.subscribe(messageHandler);
```

6.130 Класс MetaInfo

Класс `MetaInfo` содержит дополнительную информацию о сохраняемом документе. Используется в поле `SaveDocumentSettings.documentMetaInfo`.

Таблица 77 – Описание полей класса `MetaInfo`

Поле	Тип	Описание
`MetaInfo.absoluteDocumentDir`	`string`	Абсолютный путь до папки с файлом
`MetaInfo.documentFileName`	`string`	Название файла

6.131 Класс NamedExpression

Класс описывает структуру именованного диапазона.

Пример

```
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpressionsEnumerator enumerator = namedExpressions.getEnumerator();
foreach (var namedExpression in enumerator)
```

```
{
    Console.WriteLine(namedExpression.getName());
    Console.WriteLine(namedExpression.getExpression());
    CellRange cellRange = namedExpression.getCellRange();
    Console.WriteLine(cellRange.getBeginColumn());
    Console.WriteLine(cellRange.getLastColumn());
}
```

6.131.1 Метод NamedExpression.getCellRange

Возвращает диапазон ячеек `CellRange`. Пример см. в `NamedExpression`.

6.131.2 Метод NamedExpression.getExpression

Возвращает текст именованного диапазона (формулы). Пример см. в `NamedExpression`.

6.131.3 Метод NamedExpression.getName

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

6.131.4 Метод NamedExpression.setName

Метод позволяет переименовать именованный диапазон.

Вызов

```
public void setName(string newName)
```

Параметры

– `newName`: новое имя диапазона, тип `string`.

Пример

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

6.132 Класс NamedExpressions

Класс для представления списка именованных диапазонов. Может быть получен с помощью методов Document.getNamedExpressions(), Table.getNamedExpressions().

6.132.1 Метод NamedExpressions.addExpression

Добавляет новый диапазон в список именованных диапазонов.

Пример

```
String expressionName = "Покупки";
String expressionValue = "=Формула покупки!$E$6:$E$14";
namedExpressions.addExpression(expressionName, expressionValue);
```

6.132.2 Метод NamedExpressions.get

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

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpression namedExpression = namedExpressions.get("Продажи");
if (namedExpression != null) {
    Console.WriteLine(namedExpression.getName());
}
```

6.132.3 Метод NamedExpressions.getEnumerator

Позволяет получить доступ ко всему списку именованных диапазонов.

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpressionsEnumerator enumerator = namedExpressions.getEnumerator();
foreach (var namedExpression in enumerator)
{
```

```
    Console.WriteLine(namedExpression.getName());
}
```

6.132.4 Метод NamedExpressions.removeExpression

Удаляет диапазон по заданному имени.

Пример

```
String expressionName = "Покупки";
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpression namedExpression = namedExpressions.get(expressionName);
namedExpressions.removeExpression(expressionName);
```

6.133 Класс NullaryConditionalFormatOperator

Класс `NullaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил без параметров. Используется при создании правила условного форматирования (`ConditionalFormatRule`).

Конструктор

```
    public    NullaryConditionalFormatOperator(ConditionalFormatNullaryCondition
condition)
```

Пример

Дата заказа
10 января 2025 г.
15 февраля 2025 г.
1 марта 2025 г.
20 апреля 2025 г.
5 мая 2025 г.
18 марта 2025 г.
2 июня 2025 г.
15 июля 2025 г.
1 августа 2025 г.
20 сентября 2025 г.
1 октября 2025 г.

Рисунок 52 – Пример создания правила для дат в прошлом месяце

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

```
var nullaryStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(255, 0, 0, 150)));
nullaryStyle.cellProperties = cellProperties;

CellRange cellRange = sheet.getCellRange("B2:B12");
var cellRangePosition = cellRange.getTableRange();

var nullaryOperator =
DocumentAPI.createNullaryConditionalFormatOperator(ConditionalFormatNullaryCondit
ion.LastMonth);

var nullaryRule = new ConditionalFormatRule(nullaryOperator, nullaryStyle,
cellRangePosition, false);
rules.addRule(nullaryRule);
```

6.133.1 Метод NullaryConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
public ConditionalFormatNullaryCondition getCondition()
```

Возвращает

– условие применения форматирования, тип ConditionalFormatNullaryCondition.

6.133.2 Метод NullaryConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.134 Класс NumberCellFormatting

Класс `NumberCellFormatting` содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода `Cell.setFormat()`.

Таблица 78 – Описание полей класса `NumberCellFormatting`

Поле	Тип	Описание
`NumberCellFormatting.decimalPlaces`	`byte`	Количество десятичных позиций
`NumberCellFormatting.useThousandsSeparator`	`bool`	Использовать разделитель для тысячных
`NumberCellFormatting.useRedForNegative`	`bool`	Использовать красный цвет для отрицательных значений
`NumberCellFormatting.useBracketsForNegative`	`bool`	Использовать скобки для отрицательных значений
`NumberCellFormatting.hideSign`	`bool`	Скрывать знак «минус» для отрицательных значений

Пример

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

NumberCellFormatting cellFormat = new NumberCellFormatting();
cellFormat.decimalPlaces = 2;
cellFormat.useThousandsSeparator = true;
cellFormat.useRedForNegative = true;
cellFormat.useBracketsForNegative = true;
cellFormat.hideSign = false;

cell.setFormat(cellFormat);
Console.WriteLine(cell.getFormattedValue());
```

6.135 Перечисление PageFieldOrder

Перечисление `PageFieldOrder` содержит виды отображения полей из области фильтров. Используется в поле `PivotTableLayoutSettings.pageFieldOrder`.

Таблица 79 – Виды отображения полей из области фильтров

Значение	Описание
`PageFieldOrder.DownThenOver`	Вниз, затем поперек
`PageFieldOrder.OverThenDown`	Поперек, затем вниз

6.136 Класс PageNumbers

Класс `PageNumbers` используется в качестве поля `pageNumbers` класса `TextExportSettings` и представляет собой коллекцию страниц для экспорта.

Позволяет установить следующие типы страниц для экспорта:

- нечетные, четные страницы, тип `PageParity`;
- список конкретных номеров страниц, тип `VectorUInt`;
- диапазон страниц с указанием начальной и конечной страницы.

Примеры

```
// Четные страницы
PageNumbers pageNumbers = new PageNumbers(PageParity.Even);
```

```
// Конкретные номера страниц
PageNumbers pageNumbers = new PageNumbers([1,13,25]);
```

```
// Диапазон страниц
PageNumbers pageNumbers = new PageNumbers(1, 20);
```

6.136.1 Метод PageNumbers.contains

Метод служит для проверки вхождения заданного номера страницы в коллекцию номеров страниц `PageNumbers`.

Пример

```
PageNumbers pageNumbers = new PageNumbers(1, 20);
Console.WriteLine(pageNumbers.contains(2));
```

6.136.2 Метод PageNumbers.getLast

Метод `PageNumbers.getLast` возвращает последний номер страницы.

Пример

```
PageNumbers pageNumbers = new PageNumbers(1, 20);
Console.WriteLine(pageNumbers.getLast());
```

6.137 Перечисление PageOrientation

Перечисление `PageOrientation` содержит варианты ориентации страницы документа. Может быть использована для получения / установки ориентации страниц для секции или документа в методах `Section.setPageOrientation()`, `Section.getPageOrientation()`.

Таблица 80 – Варианты ориентации страницы документа

Значение	Описание
`PageOrientation.Landscape`	Альбомная ориентация
`PageOrientation.Portrait`	Книжная ориентация

Примеры

```
Block block = document.getBlocks().getBlock(0);
if (block != null) {
    Section section = block.getSection();
    section.setPageOrientation(PageOrientation.Portrait);
    Console.WriteLine(section.getPageOrientation());
}
```

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    section.setPageOrientation(PageOrientation.Portrait);
```

```
    Console.WriteLine(section.getPageOrientation());
}
```

6.138 Перечисление PageParity

Варианты выбора страниц для экспорта и печати представлены в таблице 81. Используется в PageNumbers.

Таблица 81 – Варианты выбора страниц для экспорта и печати

Значение	Описание
PageParity.Odd	Только нечетные страницы
PageParity.Even	Только четные страницы
PageParity.All	Все страницы

6.139 Класс PageProperties

Класс PageProperties предоставляет такие свойства страницы как высота, ширина, размеры полей. Используется в Document.setPageProperties(), Section.getPageProperties(), Section.setPageProperties().

Таблица 82 – Описание полей класса PageProperties

Поле	Тип	Описание
PageProperties.height	float	Высота страницы
PageProperties.width	float	Ширина страницы
PageProperties.margins	Insets	Поля страницы

Примеры

```
PageProperties pageProperties = section.getPageProperties();
pageProperties.height = 100;
pageProperties.width = 200;
section.setPageProperties(pageProperties);
```

```
PageProperties pageProperties = new PageProperties();
pageProperties.height = 100;
```

```
pageProperties.width = 200;
document.setPageProperties(pageProperties);
```

```
PageProperties pageProperties = new PageProperties(100, 200);
document.setPageProperties(pageProperties);
```

6.140 Класс Paragraph

Класс `Paragraph` предоставляет доступ к свойствам абзаца (см. Рисунок 2).

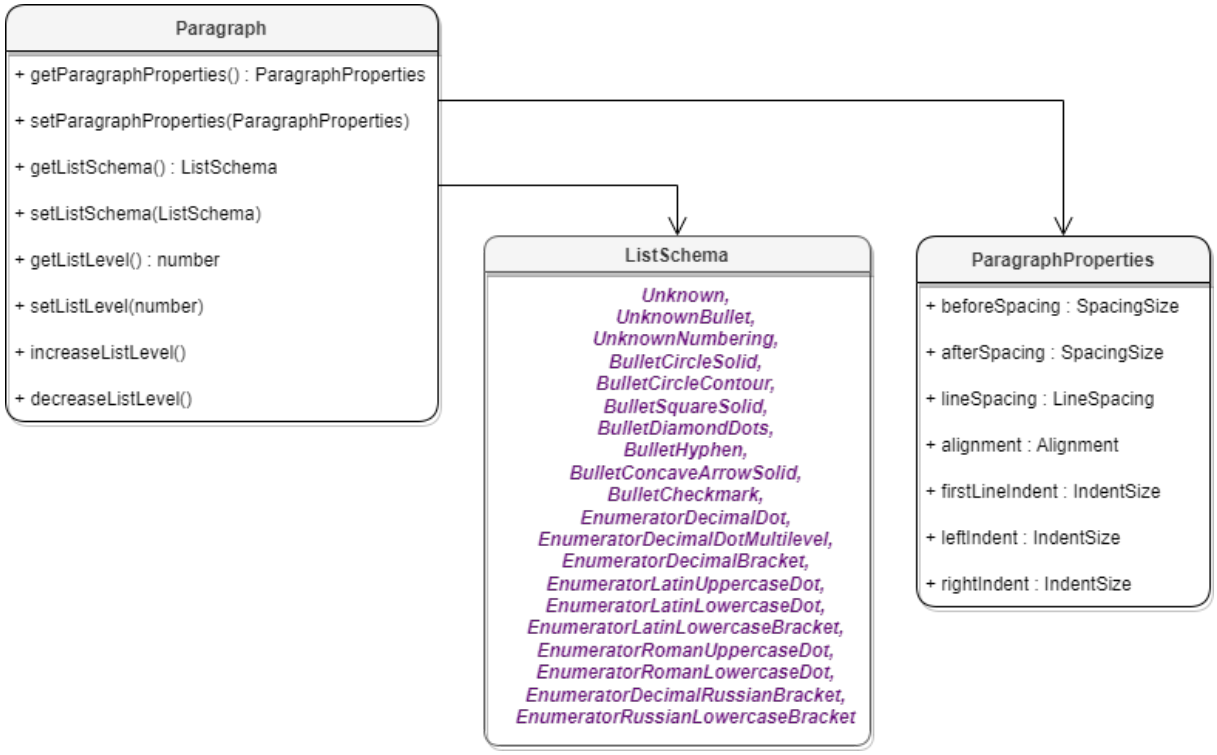


Рисунок 53 – Объектная модель классов для работы со свойствами абзаца

6.140.1 Метод Paragraph.decreaseListLevel

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

Пример

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.decreaseListLevel();
```

```
    Console.WriteLine(paragraph.getListLevel());
}
```

6.140.2 Метод Paragraph.getListLevel

Метод позволяет получить глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    Console.WriteLine(paragraph.getListLevel());
}
```

6.140.3 Метод Paragraph.getListSchema

Метод возвращает схему форматирования абзаца ListSchema, если схема нумерации установлена для абзаца. Данный метод используется только в текстовом документе.

Пример

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    Console.WriteLine(paragraph.getListSchema());
}
```

6.140.4 Метод Paragraph.getParagraphProperties

Метод предоставляет доступ к классу, определяющему такие свойства абзаца ParagraphProperties, как выравнивание текста, межстрочные интервалы, отступы и т. д.

Пример для текстового документа

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
}
```

Пример для табличного документа

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
NCT.MyOfficeSDK.Range range = cell.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
}
```

6.140.5 Метод Paragraph.getResolvedStyle

Метод возвращает комбинированный стиль текущего абзаца. Комбинированным стилем является стиль всего абзаца, общий стиль всех его частей или стиль по умолчанию. Данный стиль отображается в интерфейсе редактора.

Вызов

```
public TextStyle getResolvedStyle()
```

Возвращает

– комбинированный стиль текущего абзаца, тип TextStyle.

Пример

```
Paragraph paragraph = document.getRange().getBegin().getParagraph();
Console.WriteLine(paragraph.getResolvedStyle().getName());
```

6.140.6 Метод Paragraph.getStyle

Метод возвращает стиль текущего абзаца.

Вызов

```
public TextStyle getStyle()
```

Возвращает

– стиль текущего абзаца, тип TextStyle.

Пример

```
Paragraphs paragraphs = document.getRange().getParagraphs();
foreach (Paragraph paragraph in paragraphs)
    Console.WriteLine(paragraph.getStyle().getName());
```

6.140.7 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

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.increaseListLevel();
    Console.WriteLine(paragraph.getListLevel());
}
```

6.140.8 Метод Paragraph.setListLevel

Метод позволяет установить глубину вложенности элемента списка.

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

Пример

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.setListLevel(1);
    Console.WriteLine(paragraph.getListLevel());
}
```

6.140.9 Метод Paragraph.setListSchema

Метод позволяет установить тип маркированного или нумерованного списка `ListSchema`. Данный метод используется только в текстовом документе.

Пример

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.setListSchema(ListSchema.BulletCircleSolid);
    Console.WriteLine(paragraph.getListSchema());
}
```

6.140.10 Метод Paragraph.setParagraphProperties

Метод предназначен для обновления свойств форматирования абзаца ParagraphProperties.

Пример для текстового документа

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
    paragraphProperties.alignment = Alignment.Left;
    paragraph.setParagraphProperties(paragraphProperties);
}
```

Пример для табличного документа

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
NCT.MyOfficeSDK.Range range = cell.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
    paragraphProperties.alignment = Alignment.Left;
    paragraph.setParagraphProperties(paragraphProperties);
}
```

6.140.11 Метод Paragraph.setStyle

Метод задает стиль абзаца.

Вызов

```
public void setStyle(TextStyle textStyle)
```

Параметры

– `textStyle`: стиль абзаца, тип `TextStyle`.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle normalStyle = styles.get(PredefinedTextStyle.Normal);
Paragraphs paragraphs = document.getRange().getParagraphs();
foreach (Paragraph paragraph in paragraphs)
    paragraph.setStyle(normalStyle);
```

6.141 Класс ParagraphProperties

Класс `ParagraphProperties` предназначен для управления свойствами форматирования абзаца (см. Рисунок 2). Класс `ParagraphProperties` используется в методах `Paragraph.getParagraphProperties()`, `Paragraph.setParagraphProperties()`, `Cell.getParagraphProperties()`, `Cell.setParagraphProperties()`, `CellRange.getParagraphProperties()` и `CellRange.setParagraphProperties()`.

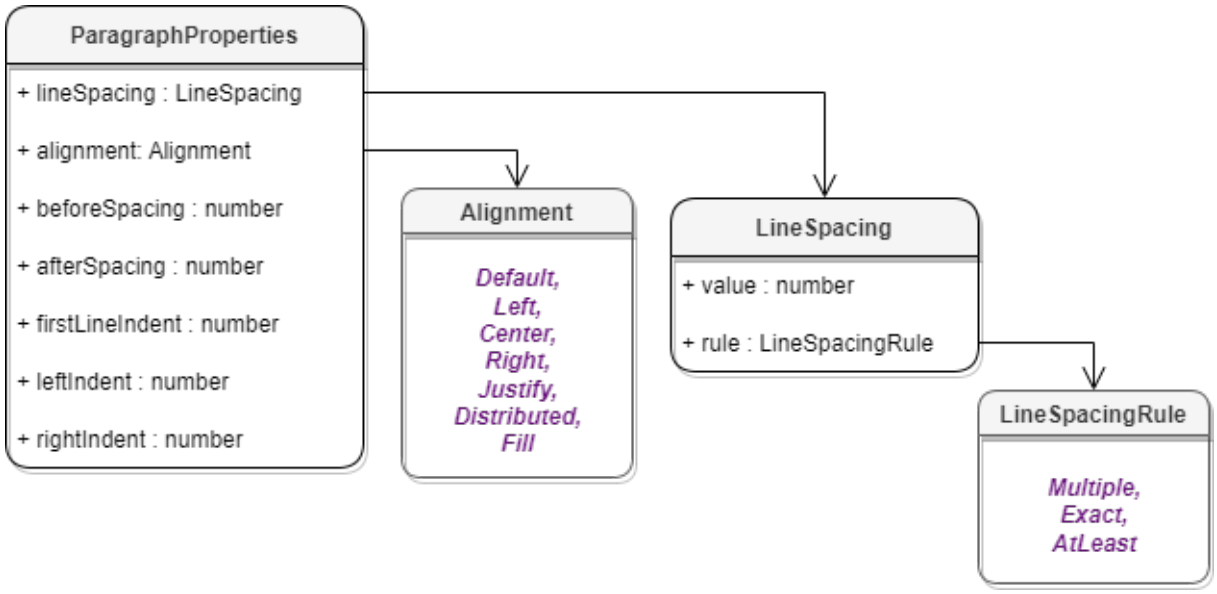
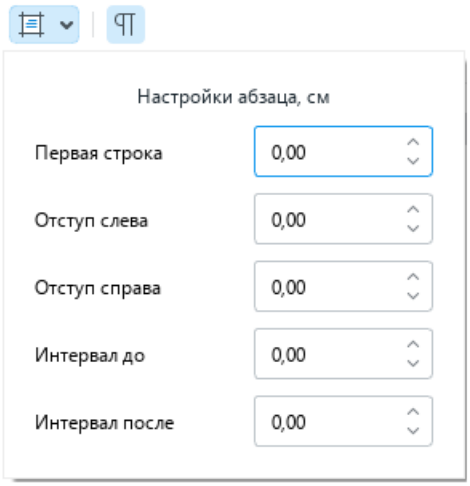


Рисунок 54 – Объектная модель классов для работы со свойствами абзаца

Описание полей класса `ParagraphProperties` представлено в таблице 83.

Таблица 83 – Описание полей класса `ParagraphProperties`

Поле	Тип	Описание
`ParagraphProperties.beforeSpacing`	`float`	Установка величины расстояния до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца,** (см. рисунок ниже), в поле **Интервал до**. 
`ParagraphProperties.afterSpacing`	`float`	Установка величины расстояния после абзаца. При работе с пользовательским интерфейсом приложения соответствует значению,

Поле	Тип	Описание
		указанному в диалоговом окне **Настройки абзаца**, в поле **Интервал после**.
`ParagraphProperties.lineSpacing`	`LineSpacing`	Расстояние между строк одного абзаца (межстрочный интервал).
`ParagraphProperties.alignment`	`Alignment`	Выравнивание текстового фрагмента по горизонтали.
`ParagraphProperties.firstLineIndent`	`float`	Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Первая строка**.
`ParagraphProperties.leftIndent`	`float`	Расстояние от левого поля документа до абзаца (отступ слева). При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ слева**.
`ParagraphProperties.rightIndent`	`float`	Расстояние от правого поля документа до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ справа**.

Пример для текстового документа

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.setListSchema(ListSchema.BulletCheckmark);
    ParagraphProperties paraProps = paragraph.getParagraphProperties();
    paraProps.afterSpacing = 28.3f;     // соответствует 1 см
    paraProps.beforeSpacing = 28.3f;    // соответствует 1 см
    paraProps.firstLineIndent = 28.3f; // соответствует 1 см
    paraProps.leftIndent = 28.3f;       // соответствует 1см
    paraProps.rightIndent = 28.3f;      // соответствует 1см
    paraProps.alignment = NCT.MyOfficeSDK.Alignment.Center;
```

```
    paraProps.lineSpacing = new LineSpacing(5.0f, LineSpacingRule.Multiple);
}
```

Пример для табличного документа

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
NCT.MyOfficeSDK.Range range = cell.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
}
```

6.142 Класс Paragraphs

Класс `Paragraphs` предоставляет доступ к коллекции абзацев типа `Paragraph` (см. Рисунок 2). Коллекция абзацев может быть получена из объекта `Range` посредством использования метода `Range.getParagraphs()`.

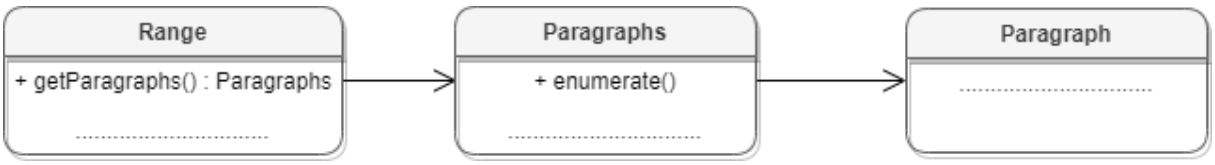


Рисунок 55 – Объектная модель для работы со списком абзацев

Пример для текстового документа

```
Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
```

Пример для табличного документа

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
Range range = cell.getRange();
Paragraphs paragraphs = range.getParagraphs();
```

6.142.1 Метод Paragraphs.decreaseListLevel

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

Пример

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.decreaseListLevel();
```

6.142.2 Метод Paragraphs.GetEnumerator

Метод позволяет перечислить коллекцию абзацев.

Пример

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    Console.WriteLine(paragraph.getRange().extractText());
}
```

6.142.3 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

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.increaseListLevel();
```

6.142.4 Метод Paragraphs.setListLevel

Метод устанавливает глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.setListLevel(1);
```

6.142.5 Метод Paragraphs.setListSchema

Метод устанавливает тип маркированного или нумерованного списка `ListSchema`. Данный метод используется только в текстовом документе.

Пример

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.setListSchema(NCT.MyOfficeSDK.ListSchema.BulletCheckmark);
```

6.143 Класс PercentageCellFormatting

Содержит параметр для процентного формата ячеек таблицы, используется в качестве аргумента метода `Cell.setFormat()`.

Таблица 84 – Описание полей класса `PercentageCellFormatting`

Поле	Тип	Описание
`PercentageCellFormatting.decimalPlaces`	`byte`	Количество десятичных позиций

Пример

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

PercentageCellFormatting cellFormat = new PercentageCellFormatting();
cellFormat.decimalPlaces = 2;

cell.setFormat(cellFormat);
Console.WriteLine(cell.getFormattedValue());
```

6.144 Класс PivotTable

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

6.144.1 Метод PivotTable.areAllFiltersInDefaultState

Метод позволяет определить, применен ли к сводной таблице хоть один фильтр.

Вызов

```
public bool areAllFiltersInDefaultState()
```

Возвращает

– `true`, если к сводной таблице не применен ни один фильтр, в ином случае – `false`.

6.144.2 Метод PivotTable.changeSourceRange

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

Пример

```
pivotTable.changeSourceRange("I3:K5");
CellRange sourceRange = pivotTable.getSourceRange();
Console.WriteLine(sourceRange.getBeginColumn() + ", " +
sourceRange.getLastColumn());
```

6.144.3 Метод PivotTable.createPivotTableEditor

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

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor.addField("Age", PivotTableFieldCategory.Rows);
pivotTableEditor.apply();
```

6.144.4 Метод PivotTable.getColumnFields

Метод возвращает список полей PivotTableCategoryField из области колонок.

Пример

```
PivotTableCategoryFields columnFields = pivotTable.getColumnFields();
PivotTableCategoryFields.PivotTableCategoryFieldsEnumerator enumerator =
columnFields.GetEnumerator();
while (enumerator.MoveNext())
{
    PivotTableCategoryField pivotTableCategoryField = enumerator.Current;
    Console.WriteLine(pivotTableCategoryField.fieldProperties.fieldAlias);
    Console.WriteLine(pivotTableCategoryField.fieldProperties.subtotalAlias);
    Console.WriteLine(pivotTableCategoryField.fieldProperties.fieldName);
    PivotTableFunctions subtotalFunctions =
pivotTableCategoryField.subtotalFunctions;
    Console.WriteLine(subtotalFunctions.Count);
}
```

6.144.5 Метод PivotTable.getConditionalLabelFilter

Метод возвращает примененный условный фильтр по подписи.

Вызов

```
    public PivotTableConditionalLabelFilter getConditionalLabelFilter(string
fieldName)
```

Параметры

– `fieldName`: название поля в сводной таблице, тип `string`.

Возвращает

– условный фильтр по подписи, тип PivotTableConditionalLabelFilter.

– `null`, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

Пример

```
var labelOperation = new PivotTableConditionalLabelFilterOperation();
labelOperation.operationType =
PivotTableConditionalLabelFilterOperationType.Contains;
```

```
labelOperation.operand = "to";

var labelFilter = pivotTable.getConditionalLabelFilter("Product Name");
labelFilter.setOperation(labelOperation);

tableEditor.setFilter(labelFilter).apply();
```

6.144.6 Метод PivotTable.getConditionalValueFilter

Метод возвращает примененный условный фильтр по значению.

Вызов

```
public PivotTableConditionalValueFilter getConditionalValueFilter(string fieldName)
```

Параметры

– `fieldName`: название поля в сводной таблице, тип `string`.

Возвращает

– условный фильтр по значению, тип `PivotTableConditionalValueFilter`.

– `null`, если поля с таким именем не существует, или оно не находится в области строк или столбцов.

Пример

```
var valueOperation = PivotTableConditionalValueFilter.getDefaultOperation(PivotTableConditionalValueFilterOperationType.Between);
valueOperation.operand1 = "20";
valueOperation.operand2 = "50";

var valueFilter = pivotTable.getConditionalValueFilter("Product Name");
valueFilter.setOperation(valueOperation);

tableEditor.setFilter(valueFilter).apply();
```

6.144.7 Метод PivotTable.getFieldCategories

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

Пример

```
PivotTableFieldCategories categories = pivotTable.getFieldCategories("Age");
PivotTableFieldCategoriesEnumerator enumerator = categories.GetEnumerator();
foreach (var PivotTableFieldCategory in enumerator)
{
    PivotTableFieldCategory pivotTableFieldCategory = enumerator.Current;
    Console.WriteLine(pivotTableFieldCategory);
}
```

6.144.8 Метод PivotTable.getFieldItems

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

Пример

```
PivotTableItems pivotTableItems = pivotTable.getFieldItems("Age");
PivotTableItemsEnumerator enumerator = pivotTableItems.GetEnumerator();
foreach (var pivotTableItem in enumerator)
{
    Console.WriteLine(pivotTableItem.getAlias());
}
```

6.144.9 Метод PivotTable.getFieldItemsByName

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

Пример

```
PivotTableItems itemsByName = pivotTable.getFieldItemsByName("Ultimate Question
of Life", "42");
PivotTableItemsEnumerator enumerator = itemsByName.GetEnumerator();
foreach (var PivotTableItem in enumerator)
{
    Console.WriteLine(PivotTableItem.getName());
}
```

6.144.10 Метод PivotTable.getFieldsList

Метод возвращает список PivotTableFields всех полей сводной таблицы.

Пример

```
PivotTableFields pivotTableFields = pivotTable.getFieldsList();
PivotTableFields.PivotTableFieldsEnumerator enumerator =
pivotTableFields.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableField pivotTableField = enumerator.Current;
    Console.WriteLine(pivotTableField.customFormula);
}
```

6.144.11 Метод PivotTable.getFilter

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

Пример

```
PivotTableFilter pivotTableFilter = pivotTable.getFilter("Age");
Console.WriteLine(pivotTableFilter.getFieldName());
```

6.144.12 Метод PivotTable.getFilters

Метод возвращает список фильтров PivotTableFilter сводной таблицы.

Пример

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    Console.WriteLine(pivotTableFilter.isHidden(0));
}
```

6.144.13 Метод PivotTable.getPageFields

Метод возвращает список полей PivotTablePageField из области фильтров.

Пример

```
PivotTablePageFields pageFields = pivotTable.getPageFields();
PivotTablePageFields.PivotTablePageFieldsEnumerator enumerator =
pageFields.GetEnumerator();
while (enumerator.MoveNext())
{
    PivotTablePageField pivotTableCategory = enumerator.Current;
    Console.WriteLine(pivotTableCategory.fieldProperties.fieldAlias);
    Console.WriteLine(pivotTableCategory.fieldProperties.subtotalAlias);
    Console.WriteLine(pivotTableCategory.fieldProperties.fieldName);
}
```

6.144.14 Метод PivotTable.getPivotRange

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

Пример

```
PivotTable pivotTable = pivotTablesManager.create(cellRange);
CellRange pivotRange = pivotTable.getPivotRange();
Console.WriteLine(pivotRange.getBeginColumn() + ", " +
pivotRange.getLastColumn());
```

6.144.15 Метод PivotTable.getPivotTableCaptions

Метод возвращает информацию PivotTableCaptions о всех заголовках сводной таблицы.

Пример

```
PivotTableCaptions pivotTableCaptions = pivotTable.getPivotTableCaptions();
Console.WriteLine(pivotTableCaptions.errorCaption);
Console.WriteLine(pivotTableCaptions.emptyCaption);
Console.WriteLine(pivotTableCaptions.grandTotalCaption);
Console.WriteLine(pivotTableCaptions.valuesHeaderCaption);
Console.WriteLine(pivotTableCaptions.columnHeaderCaption);
Console.WriteLine(pivotTableCaptions.rowHeaderCaption);
```

6.144.16 Метод PivotTable.getPivotTableLayoutSettings

Метод возвращает настройки отображения PivotTableLayoutSettings сводной таблицы.

Пример

```
PivotTableLayoutSettings layoutSettings =
pivotTable.getPivotTableLayoutSettings();
Console.WriteLine(layoutSettings.displayFieldCaptions);
Console.WriteLine(layoutSettings.indentForCompactLayout);
Console.WriteLine(layoutSettings.isMergeAndCenterLabelsEnabled);
Console.WriteLine(layoutSettings.pageFieldOrder);
Console.WriteLine(layoutSettings.pageFieldWrapCount);
Console.WriteLine(layoutSettings.reportLayout);
Console.WriteLine(layoutSettings.useGridDropZones);
Console.WriteLine(layoutSettings.valueFieldsOrientation);
```

6.144.17 Метод PivotTable.getRowFields

Метод возвращает список полей PivotTableCategoryField из области строк.

Пример

```
PivotTableCategoryFields rowFields = pivotTable.getRowFields();
PivotTableCategoryFields.PivotTableCategoryFieldsEnumerator enumerator =
rowFields.GetEnumerator();
while (enumerator.MoveNext())
{
    PivotTableCategoryField pivotTableCategoryField = enumerator.Current;
    Console.WriteLine(pivotTableCategoryField.fieldProperties.fieldAlias);
    Console.WriteLine(pivotTableCategoryField.fieldProperties.subtotalAlias);
    Console.WriteLine(pivotTableCategoryField.fieldProperties.fieldName);
    PivotTableFunctions subtotalFunctions =
pivotTableCategoryField.subtotalFunctions;
    Console.WriteLine(subtotalFunctions.Count);
}
```

6.144.18 Метод PivotTable.getSortingParams

Метод возвращает параметры сортировки, примененные к заданному полю сводной таблицы.

Вызов

```
public PivotTableSortingParams getSortingParams(string fieldName)
```

Параметры

– `fieldName`: название поля таблицы, тип `string`.

Возвращает

– параметры сортировки, тип `PivotTableSortingParams`.

– `null`, если параметры сортировки не удалось получить.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("L1");
PivotTable pivotTable = cell.getPivotTable();
PivotTableSortingParams sorting = pivotTable.getSortingParams("Product");
Console.WriteLine(sorting.sortingType);
Console.WriteLine(sorting.sortingOrder);
```

6.144.19 Метод PivotTable.getSourceRange

Метод возвращает диапазон `CellRange` исходных данных сводной таблицы.

Пример

```
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null) {
    CellRange sourceRange = pivotTable.getSourceRange();
    Console.WriteLine(sourceRange.getBeginColumn());
}
```

6.144.20 Метод PivotTable.getSourceRangeAddress

Метод возвращает текстовое представление диапазона исходных данных сводной таблицы.

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null) {
    pivotTable.remove();
}
```

6.144.21 Метод PivotTable.getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства, которые представлены в сводной таблице.

Вызов

```
public PivotTableUnsupportedFeatures getUnsupportedFeatures()
```

Возвращает

– коллекция значений PivotTableUnsupportedFeature.

Пример

```
PivotTableUnsupportedFeatures features = pivotTable.getUnsupportedFeatures();
foreach (var feature in features) {
    Console.WriteLine(feature);
}
```

6.144.22 Метод PivotTable.getValueFields

Метод возвращает список полей PivotTableValueField из области значений.

Пример

```
PivotTableValueFields valueFields = pivotTable.getValueFields();
PivotTableValueFields.PivotTableValueFieldsEnumerator enumerator =
valueFields.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableValueField pivotTableValueField = enumerator.Current;
    Console.WriteLine(pivotTableValueField.baseFieldName);
    Console.WriteLine(pivotTableValueField.cellNumberFormat);
    Console.WriteLine(pivotTableValueField.customFormula);
    Console.WriteLine(pivotTableValueField.totalFunction);
```

```
    Console.WriteLine(pivotTableValueField.valueFieldName);
}
```

6.144.23 Метод PivotTable.getViewDetailsEnabled

Метод позволяет определить включена ли возможность просмотра детализации данных для значений текущей сводной таблицы.

Вызов

```
public bool getViewDetailsEnabled()
```

Возвращает

– `true`, если включена возможность просмотра детализации данных, в ином случае – `false`.

6.144.24 Метод PivotTable.isColumnGrandTotalEnabled

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

Пример

```
Console.WriteLine(pivotTable.isColumnGrandTotalEnabled());
```

6.144.25 Метод PivotTable.isRowGrandTotalEnabled

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

Пример

```
Console.WriteLine(pivotTable.isRowGrandTotalEnabled());
```

6.144.26 Метод PivotTable.remove

Метод удаляет сводную таблицу.

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null) {
```

```
    pivotTable.remove();
}
```

6.144.27 Метод PivotTable.update

Метод обновляет и полностью пересчитывает сводную таблицу, возвращает PivotTableUpdateResult.

Пример

```
PivotTableUpdateResult updateResult = pivotTable.update();
if (updateResult == PivotTableUpdateResult.FieldAlreadyEnabled) {
    ............
}
```

6.145 Перечисление PivotTableBaseItemPosition

Перечисление PivotTableBaseItemPosition содержит типы сравнения значений для дополнительных вычислений. Используется в PivotTableCalculationDataBaseItem.

Таблица 85 – Описание типов сравнения значений

Значение	Описание
PivotTableBaseItemPosition.Previous	Каждое значение сравнивается с предыдущим
PivotTableBaseItemPosition.Next	Каждое значение сравнивается со следующим

6.146 Класс PivotTableCalculationData

Класс PivotTableCalculationData представляет собой настройки дополнительных вычислений для поля значений сводной таблицы. Используется в методе PivotTableEditor.setAdditionalCalculations() и поле PivotTableValueField.calculationData.

Конструкторы

```
    public PivotTableCalculationData()
```

```
    public PivotTableCalculationData(PivotTableDataCalculationType calculationType)
```

```
    public PivotTableCalculationData(PivotTableDataCalculationType calculationType, PivotTableCalculationDataBaseEntity base_)
```

Таблица 86 – Описание полей класса `PivotTableCalculationData`

Поле	Тип	Описание
`PivotTableCalculationData.calculationType`	`PivotTableDataCalculationType`	Тип вычисления
`PivotTableCalculationData.baseEntity`	`PivotTableCalculationDataBaseEntity`	Данные для вычисления

Пример дополнительных вычислений типа "% от общего итога"

```
Table sheet = document.getBlocks().getTable(0);
PivotTable pivotTable = sheet.getCell("J8").getPivotTable();
PivotTableEditor tableEditor = pivotTable.createPivotTableEditor();

PivotTableCalculationData data = new PivotTableCalculationData();
data.calculationType = PivotTableDataCalculationType.PercentOfTotal;

tableEditor.setAdditionalCalculations("Sum of Total", data);
tableEditor.apply();
```

Пример дополнительных вычислений типа "% от родительского итога" поля Product Name

```
PivotTableCalculationData data = new PivotTableCalculationData();
data.calculationType = PivotTableDataCalculationType.PercentOfParent;
data.baseEntity = new PivotTableCalculationDataBaseEntity("Product Name");
```

Пример дополнительных вычислений типа "Отличие" от предыдущего значения в поле Country

```
PivotTableCalculationData data = new PivotTableCalculationData();
data.calculationType = PivotTableDataCalculationType.Difference;
var baseItem = new
PivotTableCalculationDataBaseItem(PivotTableBaseItemPosition.Previous);
data.baseEntity = new PivotTableCalculationDataBaseEntity("Country", baseItem);
```

6.147 Класс PivotTableCalculationDataBaseEntity

Класс `PivotTableCalculationDataBaseEntity` представляет собой настройки данных для задания дополнительных вычислений. Используется в поле `PivotTableCalculationData.baseEntity`.

Конструкторы

```
public PivotTableCalculationDataBaseEntity()
```

```
public PivotTableCalculationDataBaseEntity(string field)
```

```
public PivotTableCalculationDataBaseEntity(string field, PivotTableCalculationDataBaseItem item)
```

Таблица 87 – Описание полей класса `PivotTableCalculationDataBaseEntity`

Поле	Тип	Описание
`PivotTableCalculationDataBaseEntity.baseField`	`string`	Базовое поле
`PivotTableCalculationDataBaseEntity.baseItem`	`PivotTableCalculationDataBaseItem`	Базовое значение

6.148 Класс PivotTableCalculationDataBaseItem

Класс `PivotTableCalculationDataBaseItem` позволяет задать значение поля для сравнения при дополнительных вычислениях (определенное значение или предыдущее/следующее значение). Используется в поле `PivotTableCalculationDataBaseEntity.baseItem`.

Конструкторы

```
public PivotTableCalculationDataBaseItem(PivotTableItem baseItem)
```

```
public PivotTableCalculationDataBaseItem(PivotTableBaseItemPosition baseItemPosition)
```

6.148.1 Метод PivotTableCalculationDataBaseItem.getItem

Метод возвращает текущее значение для сравнения при дополнительных вычислениях.

Вызов

```
public PivotTableItem getItem()
```

Возвращает

– элемент сводной таблицы, тип PivotTableItem.

– null, если вместо значения задано сравнение.

6.148.2 Метод PivotTableCalculationDataBaseItem.getPosition

Метод возвращает текущий тип сравнения значений для дополнительных вычислений.

Вызов

```
public PivotTableBaseItemPosition? getPosition()
```

Возвращает

– тип сравнения значений, тип PivotTableBaseItemPosition.

– null, если вместо сравнения задано конкретное значение.

6.149 Класс PivotTableCaptions

Класс PivotTableCaptions хранит все пользовательские заголовки сводной таблицы (см. PivotTable.getPivotTableCaptions() и PivotTableEditor.setCaptions()).

Таблица 88 – Описание полей класса PivotTableCaptions

Поле	Тип	Описание
PivotTableCaptions.errorCaption	string	Подпись для значений, которые возвращают ошибку
PivotTableCaptions.emptyCaption	string	Подпись для значений, которые возвращают пустое значение
PivotTableCaptions.grandTotalCaption	string	Подпись общих итогов

Поле	Тип	Описание
PivotTableCaptions.valuesHeaderCaption	string	Подпись поля из области значений; это поле отображается в отчете в случае, если в сводной таблице находится более двух полей из области значений, и макет имеет тип ‘табличный’ или ‘структурный’
PivotTableCaptions.rowHeaderCaption	string	Подпись заголовка строк (видна только при включенном компактном макете, это макет по умолчанию)
PivotTableCaptions.columnHeaderCaption	string	Подпись заголовка колонок (видна только при включенном компактном макете, это макет по умолчанию)

6.150 Класс PivotTableCategoryField

Класс PivotTableCategoryField содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. Таблицу 89). Объект может быть получен посредством вызовов PivotTable.getRowFields(), PivotTable.getColumnFields().

Таблица 89 – Описание полей PivotTableCategoryField

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

6.151 Класс PivotTableConditionalLabelFilter

Класс PivotTableConditionalLabelFilter представляет собой условный фильтр по подписи. Фильтрация производится по строковым значениям, которые содержит поле сводной таблицы. Используется в методах PivotTable.getConditionalLabelFilter() и PivotTableEditor.setFilter().

6.151.1 Метод PivotTableConditionalLabelFilter.getFieldName

Метод возвращает имя поля, с которым ассоциирован фильтр.

Вызов

```
public string getFieldName()
```

Возвращает

– имя поля, тип `string`.

6.151.2 Метод PivotTableConditionalLabelFilter.getOperation

Метод возвращает текущие настройки фильтра.

Вызов

```
public PivotTableConditionalLabelFilterOperation getOperation()
```

Возвращает

– настройки фильтра, тип `PivotTableConditionalLabelFilterOperation`.

6.151.3 Метод PivotTableConditionalLabelFilter.isInDefaultState

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

Вызов

```
public bool isInDefaultState()
```

Возвращает

– `true`, если фильтр не содержит настроек, в ином случае – `false`.

6.151.4 Метод PivotTableConditionalLabelFilter.reset

Метод сбрасывает настройки текущего фильтра.

Вызов

```
public void reset()
```

6.151.5 Метод PivotTableConditionalLabelFilter.setOperation

Метод устанавливает настройки фильтра.

Вызов

```
public void setOperation(PivotTableConditionalLabelFilterOperation operation)
```

Параметры

– `operation`: настройки фильтра, тип `PivotTableConditionalLabelFilterOperation`.

Пример

```
var labelOperation = new PivotTableConditionalLabelFilterOperation();
labelOperation.operationType =
PivotTableConditionalLabelFilterOperationType.Contains;
labelOperation.operand = "to";

var labelFilter = pivotTable.getConditionalLabelFilter("Product Name");
labelFilter.setOperation(labelOperation);

tableEditor.setFilter(labelFilter).apply();
```

6.152 Класс PivotTableConditionalLabelFilterOperation

Класс `PivotTableConditionalLabelFilterOperation` представляет собой настройки условного фильтра по подписи. Используется в методах `PivotTableConditionalLabelFilter.getOperation()` и `PivotTableConditionalLabelFilter.setOperation()`.

Таблица 90 – Описание полей класса `PivotTableConditionalLabelFilterOperation`

Поле	Тип	Описание
`PivotTableConditionalLabelFilterOperation.operationType`	`PivotTableConditionalLabelFilterOperationType`	Тип операции сравнения
`PivotTableConditionalLabelFilterOperation.operand`	`string`	Значение для сравнения

Пример

```
var labelOperation = new PivotTableConditionalLabelFilterOperation();
labelOperation.operationType =
PivotTableConditionalLabelFilterOperationType.Contains;
labelOperation.operand = "to";

var labelFilter = pivotTable.getConditionalLabelFilter("Product Name");
labelFilter.setOperation(labelOperation);

tableEditor.setFilter(labelFilter).apply();
```

6.153 Перечисление PivotTableConditionalLabelFilterOperationType

Перечисление `PivotTableConditionalLabelFilterOperationType` содержит типы операций сравнения, применяемые к фильтру по подписи. Используется в поле `PivotTableConditionalLabelFilterOperation.operationType`.

Таблица 91 – Описание типов операций сравнения

Значение	Описание
`PivotTableConditionalLabelFilterOperationType.Equal`	Равные
`PivotTableConditionalLabelFilterOperationType.NotEqual`	Не равные
`PivotTableConditionalLabelFilterOperationType.BeginsWith`	Начинаются с
`PivotTableConditionalLabelFilterOperationType.NotBeginsWith`	Не начинается с
`PivotTableConditionalLabelFilterOperationType.EndsWith`	Заканчиваются на
`PivotTableConditionalLabelFilterOperationType.NotEndsWith`	Не заканчиваются на
`PivotTableConditionalLabelFilterOperationType.Contains`	Содержат
`PivotTableConditionalLabelFilterOperationType.NotContains`	Не содержат

6.154 Класс PivotTableConditionalValueFilter

Класс `PivotTableConditionalValueFilter` представляет собой условный фильтр по значению. Фильтрация производится по значениям, которые соответствуют полю в строке или столбце сводной таблицы. Используется в методах `PivotTable.getConditionalValueFilter()` и `PivotTableEditor.setFilter()`.

6.154.1 Метод PivotTableConditionalValueFilter.getDefaultOperation

Метод возвращает настройки условного фильтра соответствующие заданной операции сравнения.

Вызов

```
public static PivotTableConditionalValueFilterOperation
getDefaultOperation(PivotTableConditionalValueFilterOperationType operationType)
```

Параметры

– `operationType`: тип операции сравнения, тип `PivotTableConditionalValueFilterOperationType`.

Возвращает

– настройки условного фильтра по значению, тип `PivotTableConditionalValueFilterOperation`.

Пример

```
var valueOperation =
PivotTableConditionalValueFilter.getDefaultOperation(PivotTableConditionalValueFi
lterOperationType.Between);
valueOperation.operand1 = "20";
valueOperation.operand2 = "50";

var valueFilter = pivotTable.getConditionalValueFilter("Product Name");
valueFilter.setOperation(valueOperation);

tableEditor.setFilter(valueFilter).apply();
```

6.154.2 Метод PivotTableConditionalValueFilter.getExpectedOperandType

Метод возвращает тип данных, который должен использоваться с заданной операцией сравнения.

Вызов

```
public static PivotTableConditionalValueFilterOperandType getExpectedOperandType(PivotTableConditionalValueFilterOperationType operationType)
```

Параметры

– `operationType`: тип операции сравнения, тип `PivotTableConditionalValueFilterOperationType`.

Возвращает

– тип данных, тип `PivotTableConditionalValueFilterOperandType`.

6.154.3 Метод PivotTableConditionalValueFilter.getFieldName

Метод возвращает имя поля, с которым ассоциирован фильтр.

Вызов

```
public string getFieldName()
```

Возвращает

– имя поля, тип `string`.

6.154.4 Метод PivotTableConditionalValueFilter.getOperation

Метод возвращает текущие настройки фильтра.

Вызов

```
public PivotTableConditionalValueFilterOperation getOperation()
```

Возвращает

– настройки фильтра, тип `PivotTableConditionalValueFilterOperation`.

6.154.5 Метод PivotTableConditionalValueFilter.isInDefaultState

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

Вызов

```
public bool isInDefaultState()
```

Возвращает

– `true`, если фильтр не содержит настроек, в ином случае – `false`.

6.154.6 Метод PivotTableConditionalValueFilter.reset

Метод сбрасывает настройки текущего фильтра.

Вызов

```
public void reset()
```

6.154.7 Метод PivotTableConditionalValueFilter.setOperation

Метод устанавливает настройки фильтра.

Вызов

```
public void setOperation(PivotTableConditionalValueFilterOperation operation)
```

Параметры

– `operation`: настройки фильтра, тип `PivotTableConditionalValueFilterOperation`.

Пример

```
var valueOperation = PivotTableConditionalValueFilter.getDefaultOperation(PivotTableConditionalValueFilterOperationType.Between);
valueOperation.operand1 = "20";
valueOperation.operand2 = "50";

var valueFilter = pivotTable.getConditionalValueFilter("Product Name");
valueFilter.setOperation(valueOperation);

tableEditor.setFilter(valueFilter).apply();
```

6.155 Перечисление PivotTableConditionalValueFilterOperandType

Перечисление `PivotTableConditionalValueFilterOperandType` содержит типы данных, с которыми могут работать различные операции сравнения. Используется в методе `PivotTableConditionalValueFilter.getExpectedOperandType()`.

Таблица 92 – Описание типов данных

Значение	Описание
`PivotTableConditionalValueFilterOperandType.PositiveInt`	Положительное целое число
`PivotTableConditionalValueFilterOperandType.Percent`	Число с плавающей запятой в диапазоне от 0 до 100
`PivotTableConditionalValueFilterOperandType.Double`	Дробное число
`PivotTableConditionalValueFilterOperandType.NonNegativeDouble`	Положительное дробное число

6.156 Класс PivotTableConditionalValueFilterOperation

Класс `PivotTableConditionalValueFilterOperation` представляет собой настройки условного фильтра по значению. Используется в методах `PivotTableConditionalValueFilter.getDefaultOperation()`, `PivotTableConditionalValueFilter.getOperation()` и `PivotTableConditionalValueFilter.setOperation()`.

Таблица 93 – Описание полей класса `PivotTableConditionalValueFilterOperation`

Поле	Тип	Описание
`PivotTableConditionalValueFilterOperation.operationType`	`PivotTableConditionalValueFilterOperationType`	Тип операции сравнения
`PivotTableConditionalValueFilterOperation.valueFieldIndex`	`uint`	Индекс поля, к которому применяется фильтр
`PivotTableConditionalValueFilterOperation.operand1`	`string`	Первое значение для сравнения
`PivotTableConditionalValueFilterOperation.operand2`	`string`	Второе значение для операции сравнения `Between`

Пример

```
var valueOperation = PivotTableConditionalValueFilter.getDefaultOperation(PivotTableConditionalValueFilterOperationType.Between);
```

```
valueOperation.operand1 = "20";
valueOperation.operand2 = "50";

var valueFilter = pivotTable.getConditionalValueFilter("Product Name");
valueFilter.setOperation(valueOperation);

tableEditor.setFilter(valueFilter).apply();
```

6.157 Перечисление PivotTableConditionalValueFilterOperationType

Перечисление `PivotTableConditionalValueFilterOperationType` содержит типы операций сравнения, применяемые к фильтру по значению. Используется в поле `PivotTableConditionalValueFilterOperation.operationType`.

Таблица 94 – Описание типов операций сравнения

Значение	Описание
`PivotTableConditionalValueFilterOperationType.Equal`	Равные
`PivotTableConditionalValueFilterOperationType.NotEqual`	Не равные
`PivotTableConditionalValueFilterOperationType.Greater`	Больше
`PivotTableConditionalValueFilterOperationType.GreaterOrEqual`	Больше или равные
`PivotTableConditionalValueFilterOperationType.Less`	Меньше
`PivotTableConditionalValueFilterOperationType.LessOrEqual`	Меньше или равные
`PivotTableConditionalValueFilterOperationType.Between`	Между
`PivotTableConditionalValueFilterOperationType.TopPercent`	Не поддерживается
`PivotTableConditionalValueFilterOperationType.BottomPercent`	Не поддерживается
`PivotTableConditionalValueFilterOperationType.TopSum`	Не поддерживается
`PivotTableConditionalValueFilterOperationType.BottomSum`	Не поддерживается
`PivotTableConditionalValueFilterOperationType.TopValues`	Не поддерживается

Значение	Описание
PivotTableConditionalValueFilterOperationType.BottomValues	Не поддерживается

6.158 Перечисление PivotTableDataCalculationType

Перечисление PivotTableDataCalculationType содержит типы дополнительных вычислений для поля значений сводной таблицы. Для некоторых типов вычислений необходимо дополнительно указать данные для сравнения. Используется в поле PivotTableCalculationData.calculationType.

Таблица 95 – Описание типов дополнительных вычислений

Значение	Описание
PivotTableDataCalculationType.Normal	Без вычислений
PivotTableDataCalculationType.Difference	Отличие (от значения заданного поля)
PivotTableDataCalculationType.Percent	% от (значения заданного поля)
PivotTableDataCalculationType.PercentDiff	Отличие в % (от значения заданного поля)
PivotTableDataCalculationType.PercentOfCol	% от итога по столбцу
PivotTableDataCalculationType.PercentOfRow	% от итога по строке
PivotTableDataCalculationType.PercentOfTotal	% от общего итога
PivotTableDataCalculationType.RunTotal	Не поддерживается
PivotTableDataCalculationType.Index	Не поддерживается
PivotTableDataCalculationType.PercentOfParent	% от родительского итога (по заданному полю)
PivotTableDataCalculationType.PercentOfParentCol	% от итога по родительскому столбцу
PivotTableDataCalculationType.PercentOfParentRow	% от итога по родительской строке
PivotTableDataCalculationType.PercentOfRunningTotal	Не поддерживается
PivotTableDataCalculationType.RankAscending	Не поддерживается
PivotTableDataCalculationType.RankDescending	Не поддерживается

6.159 Класс PivotTableEditor

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

6.159.1 Метод PivotTableEditor.addField

Метод добавляет новое поле в сводную таблицу, используя параметры:

– fieldName - имя поля;

– toCategory - категория поля (тип - PivotTableFieldCategory);

– index - позиция в категории. Метод возвращает объект PivotTableEditor.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.addField("CC",
PivotTableFieldCategory.Values, 0);
pivotTableEditor.apply();
```

6.159.2 Метод PivotTableEditor.apply

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

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
if (PivotTableUpdateResult.Success == pivotTableEditor.apply()) {
    Console.WriteLine("Successfully applied");
}
```

6.159.3 Метод PivotTableEditor.disableField

Метод удаляет поле из всех областей. Параметр fieldName - имя поля (тип - строка). Метод возвращает объект PivotTableEditor.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.disableField("Age");
pivotTableEditor.apply();
```

6.159.4 Метод PivotTableEditor.enableField

Метод добавляет поле в область, зависящую от типа поля. Параметр `fieldName` - имя поля. Метод возвращает объект `PivotTableEditor`.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.enableField("Age");
pivotTableEditor.apply();
```

6.159.5 Метод PivotTableEditor.moveField

Метод перемещает поле между категориями, используя параметры:

- `fieldName` - имя поля;
- `toCategory` - область, в которую перемещается поле (тип - `PivotTableFieldCategory`);
- `index` - позиция в новой категории. Метод возвращает объект `PivotTableEditor`.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.moveField("CC",
PivotTableFieldCategory.Values, 0);
pivotTableEditor.apply();
```

6.159.6 Метод PivotTableEditor.removeField

Метод удаляет поле из категории, используя параметры:

- `fieldName` - имя поля;
- `fromCategory` - область, из которой удаляется поле (тип - `PivotTableFieldCategory`).

Метод возвращает объект PivotTableEditor.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.removeField("BB",
PivotTableFieldCategory.Values);
pivotTableEditor.apply();
```

6.159.7 Метод PivotTableEditor.reorderField

Метод изменяет позицию поля в пределах категории, используя параметры:

– `fieldName` - имя поля;

– `category` - область (тип - PivotTableFieldCategory);

– `toIndex` - новая позиция поля.

Метод возвращает объект PivotTableEditor.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.reorderField("CC",
PivotTableFieldCategory.Values, 0);
pivotTableEditor.apply();
```

6.159.8 Метод PivotTableEditor.resetAllFilters

Метод сбрасывает все фильтры, примененные к сводной таблице, и обновляет её.

Вызов

```
public PivotTableEditor resetAllFilters()
```

Возвращает

– текущий редактор сводных таблиц, тип PivotTableEditor.

6.159.9 Метод PivotTableEditor.setAdditionalCalculations

Метод задает дополнительные вычисления для поля значений сводной таблицы.

Вызов

```
public PivotTableEditor setAdditionalCalculations(string valueFieldName,
PivotTableCalculationData calculationData)
```

Параметры

– `valueFieldName`: название поля значений, тип `string`.

– `calculationData`: настройки дополнительных вычислений, тип `PivotTableCalculationData`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

Пример дополнительных вычислений типа "Отличие" от предыдущего значения в поле Country

```
Table sheet = document.getBlocks().getTable(0);
PivotTable pivotTable = sheet.getCell("J8").getPivotTable();
PivotTableEditor tableEditor = pivotTable.createPivotTableEditor();

PivotTableCalculationData data = new PivotTableCalculationData();
data.calculationType = PivotTableDataCalculationType.Difference;
var baseItem = new
PivotTableCalculationDataBaseItem(PivotTableBaseItemPosition.Previous);
data.baseEntity = new PivotTableCalculationDataBaseEntity("Country", baseItem);

tableEditor.setAdditionalCalculations("Sum of Total", data);
tableEditor.apply();
```

6.159.10 Метод PivotTableEditor.setCaptions

Метод задает заголовки сводной таблицы `PivotTableCaptions`, возвращает объект `PivotTableEditor`.

Пример

```
PivotTableCaptions captions = pivotTable.getPivotTableCaptions();
captions.grandTotalCaption = "Общий итог за год";
```

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor.setCaptions(captions);
pivotTableEditor.apply();
```

6.159.11 Метод PivotTableEditor.setColumnFieldsCollapsed

Метод сворачивает или разворачивает все поля в области столбцов сводной таблицы.

Вызов

```
public PivotTableEditor setColumnFieldsCollapsed(bool collapse)
```

Параметры

– `collapse`: `true`, чтобы свернуть поля в области столбцов, в ином случае – `false`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

6.159.12 Метод PivotTableEditor.setFieldAlias

Метод задает альтернативное название для указанного поля.

Вызов

```
public PivotTableEditor setFieldAlias(string fieldName, string newAlias)
```

Параметры

– `fieldName`: название поля, тип `string`.

– `newAlias`: альтернативное название поля, тип `string`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
PivotTable pivotTable = sheet.getCell("J8").getPivotTable();
PivotTableEditor tableEditor = pivotTable.createPivotTableEditor();

tableEditor.setFieldAlias("Country", "Location").apply();
```

6.159.13 Метод PivotTableEditor.setFieldCollapsed

Метод сворачивает или разворачивает заданное поле сводной таблицы.

Вызов

```
public PivotTableEditor setFieldCollapsed(string fieldName, bool collapse)
```

Параметры

– `fieldName`: название поля, тип `string`.

– `collapse`: `true`, чтобы свернуть заданное поле, в ином случае – `false`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

6.159.14 Метод PivotTableEditor.setFilter

Метод задает фильтр `PivotTableFilter`, `PivotTableConditionalLabelFilter` или `PivotTableConditionalValueFilter` сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`. Метод возвращает объект `PivotTableEditor`.

Перегрузки

```
public PivotTableEditor setFilter(PivotTableFilter filter)
```

```
public PivotTableEditor setFilter(PivotTableConditionalLabelFilter filter)
```

```
public PivotTableEditor setFilter(PivotTableConditionalValueFilter filter)
```

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator pivotTableItemsEnumerator =
pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in pivotTableItemsEnumerator)
{
    uint filterSize = pivotTableFilter.getCount();
    for (uint i = 0; i < filterSize; i++)
    {
        pivotTableFilter.setHidden(i, true);
    }
    pivotTableEditor.setFilter(pivotTableFilter);
}
PivotTableUpdateResult pivotTableUpdateResult = pivotTableEditor.apply();
```

6.159.15 Метод PivotTableEditor.setFilters

Метод задает фильтры `PivotTableFilters` сводной таблицы. Если какой-то из фильтров не может быть применен, он пропускается. Метод возвращает объект `PivotTableEditor`.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator pivotTableItemsEnumerator =
pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in pivotTableItemsEnumerator)
{
    uint filterSize = pivotTableFilter.getCount();
    for (uint i = 0; i < filterSize; i++)
    {
        pivotTableFilter.setHidden(i, true);
    }
    pivotTableEditor.setFilter(pivotTableFilter);
}
pivotTableEditor.setFilters(pivotTableFilters);
PivotTableUpdateResult pivotTableUpdateResult = pivotTableEditor.apply();
```

6.159.16 Метод PivotTableEditor.setGrandTotalSettings

Метод задает настройки отображения общего итога. Параметры: `isRowGrandTotalEnabled` – показывать общие итоги для строк, `isColGrandTotalEnabled` – показывать общие итоги для столбцов.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.setGrandTotalSettings(true, true);
pivotTableEditor.apply();
```

6.159.17 Метод PivotTableEditor.setItemCollapsed

Метод сворачивает или разворачивает заданный элемент сводной таблицы.

Вызов

```
public PivotTableEditor setItemCollapsed(PivotTableItem item, bool collapse)
```

Параметры

– `item`: элемент сводной таблицы, тип `PivotTableItem`.

– `collapse`: `true`, чтобы свернуть заданный элемент, в ином случае – `false`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

6.159.18 Метод PivotTableEditor.setLayoutSettings

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

Пример

```
PivotTableLayoutSettings pivotTableLayoutSettings =
pivotTable.getPivotTableLayoutSettings();
pivotTableLayoutSettings.reportLayout = PivotTableReportLayout.Tabular;

PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.setLayoutSettings(pivotTableLayoutSettings);
pivotTableEditor.apply();
```

6.159.19 Метод PivotTableEditor.setRowFieldsCollapsed

Метод сворачивает или разворачивает все поля в области строк сводной таблицы.

Вызов

```
public PivotTableEditor setRowFieldsCollapsed(bool collapse)
```

Параметры

– `collapse`: `true`, чтобы свернуть поля в области строк, в ином случае – `false`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

6.159.20 Метод PivotTableEditor.setSortingByLabel

Метод позволяет отсортировать данные в сводной таблице по названию строк и столбцов.

Вызов

```
public PivotTableEditor setSortingByLabel(string fieldName, PivotTableSortingOrder order)
```

Параметры

– `fieldName`: название поля для сортировки, тип `string`.

– `order`: порядок сортировки, тип `PivotTableSortingOrder`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("L1");
PivotTable pivotTable = cell.getPivotTable();
PivotTableEditor editor = pivotTable.createPivotTableEditor();
editor.setSortingByLabel("Product", PivotTableSortingOrder.Ascending);
editor.apply();
```

6.159.21 Метод PivotTableEditor.setSortingByValue

Метод позволяет отсортировать данные в сводной таблице по значениям и итогам.

Вызов

```
public PivotTableEditor setSortingByValue(string fieldName, PivotTableSortingOrder order, string valueFieldName, VectorString sortingByValueSlice, PivotTableFunction? sortingFieldSubtotal, PivotTableFunction? valueSliceSubtotal)
```

Параметры

– `fieldName`: название поля для сортировки, тип `string`.

– `order`: порядок сортировки, тип `PivotTableSortingOrder`.

– `valueFieldName`: название поля значений, которое содержит данные для сортировки, тип `string`.

– `sortingByValueSlice`: (необязательный) срез данных для сортировки по значениям, содержит название строки или столбца с данными или путь до него во вложенной структуре, тип `VectorString`.

– `sortingFieldSubtotal`: (необязательный) функция для сортировки по подытогам, применяется к заданному в параметре `fieldName` полю, тип `PivotTableFunction`.

– `valueSliceSubtotal`: (необязательный) функция для сортировки по подытогам, применяется к заданному в параметре `sortingByValueSlice` полю, тип `PivotTableFunction`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

Пример сортировки по общим итогам

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("L1");
PivotTable pivotTable = cell.getPivotTable();
PivotTableEditor editor = pivotTable.createPivotTableEditor();
editor.setSortingByValue("Manager", PivotTableSortingOrder.Ascending, "Сумма по полю Total");
editor.apply();
```

Пример сортировки по значениям во вложенном столбце "Russia > Furniture"

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("L1");
PivotTable pivotTable = cell.getPivotTable();
PivotTableEditor editor = pivotTable.createPivotTableEditor();
editor.setSortingByValue("Manager", PivotTableSortingOrder.Descending, "Сумма по полю Total", ["Russia", "Furniture"]);
editor.apply();
```

Пример сортировки по промежуточным итогам в столбце "Russia"

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("L1");
PivotTable pivotTable = cell.getPivotTable();
PivotTableEditor editor = pivotTable.createPivotTableEditor();
editor.setSortingByValue("Manager", PivotTableSortingOrder.Descending, "Сумма по
```

```
полю Total", ["Russia"], null, PivotTableFunction.Auto);
editor.apply();
```

6.159.22 Метод PivotTableEditor.setSubtotalFunctions

Метод задает функции вычисления промежуточных итогов для указанного поля.

Вызов

```
public PivotTableEditor setSubtotalFunctions(string fieldName,
PivotTableFunctions subtotalFunctions)
```

Параметры

– `fieldName`: название поля, тип `string`.

– `subtotalFunctions`: функции для вычисления промежуточных итогов, тип коллекция объектов `PivotTableFunction`.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
PivotTable pivotTable = sheet.getCell("J8").getPivotTable();
PivotTableEditor tableEditor = pivotTable.createPivotTableEditor();

PivotTableFunctions functions = new PivotTableFunctions();
functions.Add(PivotTableFunction.Count);
functions.Add(PivotTableFunction.Average);

tableEditor.setSubtotalFunctions("Product Name", functions).apply();
```

6.159.23 Метод PivotTableEditor.setSubtotalOnTop

Метод задает расположение промежуточных итогов для указанного поля.

Вызов

```
public PivotTableEditor setSubtotalOnTop(string fieldName, bool
isSubtotalOnTop)
```

Параметры

– `fieldName`: название поля, тип `string`.

– `isSubtotalOnTop`: `true`, чтобы отображать промежуточные итоги в заголовке группы, `false` – отображать под группой.

Возвращает

– текущий редактор сводных таблиц, тип `PivotTableEditor`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
PivotTable pivotTable = sheet.getCell("J8").getPivotTable();
PivotTableEditor tableEditor = pivotTable.createPivotTableEditor();

tableEditor.setSubtotalOnTop("Product Name", true).apply();
```

6.159.24 Метод PivotTableEditor.setSummarizeFunction

Метод задает суммирующую функцию для поля из области значений, используя параметры:

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

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

Метод возвращает объект `PivotTableEditor`.

Пример

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFunction summarizeFunction = PivotTableFunction.Average;
pivotTableEditor = pivotTableEditor.setSummarizeFunction("CC",
summarizeFunction);
```

6.159.25 Метод PivotTableEditor.setViewDetailsEnabled

Метод позволяет задать возможность просмотра детализации данных для значений сводной таблицы.

Вызов

```
public PivotTableEditor setViewDetailsEnabled(bool enabled)
```

Параметры

– `enabled`: `true`, чтобы включить возможность просмотра детализации данных, в ином случае – `false`.

Возвращает

– текущий редактор сводных таблиц, тип PivotTableEditor.

Пример

```
Table sheet = document.getBlocks().getTable(0);
PivotTable pivotTable = sheet.getCell("J8").getPivotTable();
PivotTableEditor tableEditor = pivotTable.createPivotTableEditor();

tableEditor.setViewDetailsEnabled(true).apply();
```

6.160 Класс PivotTableField

Класс PivotTableField содержит свойства полей сводной таблицы (см. Таблицу 96). Объект может быть получен посредством вызова PivotTable.getFieldsList().

Таблица 96 – Описание полей класса PivotTableField

Поле	Тип	Описание
PivotTableField.fieldProperties	PivotTableFieldProperties	Свойства полей сводной таблицы
PivotTableField.fieldCategories	PivotTableFieldCategories	Категории полей сводной таблицы
PivotTableField.customFormula	string	Вычисляемая формула

6.161 Класс PivotTableFieldCategories

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

6.161.1 Метод PivotTableFieldCategories.getEnumerator

Метод для перечисления категорий поля PivotTableFieldCategory.

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
```

```
    PivotTableFieldCategories categories = pivotTable.getFieldCategories("Age");
    PivotTableFieldCategoriesEnumerator enumerator = categories.GetEnumerator();
    foreach (var pivotTableFieldCategory in enumerator)
    {
        Console.WriteLine(pivotTableFieldCategory);
    }
}
```

6.162 Перечисление PivotTableFieldCategory

Перечисление `PivotTableFieldCategory` содержит флаги, которые задают категорию области полей. Пример использования см. в `PivotTable.getFieldCategories()`.

Таблица 97 – Категории области полей

Значение	Описание
`PivotTableFieldCategory.Pages`	Область фильтров
`PivotTableFieldCategory.Rows`	Область строк
`PivotTableFieldCategory.Columns`	Область столбцов
`PivotTableFieldCategory.Values`	Область значений

6.163 Класс PivotTableFieldProperties

Класс `PivotTableFieldProperties` содержит свойства поля `PivotTableField` сводной таблицы (см. Таблицу 98).

Таблица 98 – Описание полей класса `PivotTableFieldProperties`

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

6.164 Класс PivotTableFields

Класс `PivotTableFields` представляет собой список полей сводной таблицы. Может быть получен посредством использования метода `PivotTable.getFieldsList()`.

6.164.1 Метод PivotTableFields.getEnumerator

Используется для перечисления полей сводной таблицы.

Пример

```
PivotTableFields pivotTableFields = pivotTable.getFieldsList();
PivotTableFields.PivotTableFieldsEnumerator enumerator =
pivotTableFields.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableField pivotTableField = enumerator.Current;
    Console.WriteLine(pivotTableField.customFormula);
}
```

6.165 Класс PivotTableFilter

Позволяет осуществить доступ к списку фильтров таблицы, каждый из которых обладает свойством видимости. При любом изменении фильтров они должны быть применены к сводной таблице посредством использования методов `PivotTableEditor.setFilter()`, `PivotTableEditor.setFilters()`.

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
    PivotTableFilters pivotTableFilters = pivotTable.getFilters();
    PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
    foreach (var pivotTableFilter in enumerator)
    {
        for (uint i = 0; i < pivotTableFilter.getCount(); i++)
        {
            pivotTableFilter.setHidden(i, false);
```

```
        }
    }
    PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
    pivotTableEditor.setFilters(pivotTableFilters);
    pivotTableEditor.apply();
}
```

6.165.1 Метод PivotTableFilter.getCount

Возвращает количество фильтруемых полей.

Пример

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    Console.WriteLine(pivotTableFilter.getCount());
}
```

6.165.2 Метод PivotTableFilter.getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

Пример

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    Console.WriteLine(pivotTableFilter.getFieldName());
}
```

6.165.3 Метод PivotTableFilter.getName

Возвращает имя поля для заданного индекса фильтра.

Пример

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
```

```
foreach (var pivotTableFilter in enumerator)
{
    for (uint i = 0; i < pivotTableFilter.getCount(); i++)
    {
        Console.WriteLine(pivotTableFilter.getName(i));
    }
}
```

6.165.4 Метод PivotTableFilter.isHidden

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

Пример

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    for (uint i = 0; i < pivotTableFilter.getCount(); i++)
    {
        Console.WriteLine(pivotTableFilter.isHidden(i));
    }
}
```

6.165.5 Метод PivotTableFilter.isInDefaultState

Метод позволяет определить содержит ли текущий фильтр какие-либо настройки.

Вызов

```
public bool isInDefaultState()
```

Возвращает

– `true`, если фильтр не содержит настроек, в ином случае – `false`.

6.165.6 Метод PivotTableFilter.reset

Метод сбрасывает настройки текущего фильтра.

Вызов

```
public void reset()
```

6.165.7 Метод PivotTableFilter.setHidden

Устанавливает видимость поля для заданного индекса, используя параметры:

– `itemName` – индекс поля;

– `hidden` – видимость (`true` – поле скрыто).

Пример

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    for (uint i = 0; i < pivotTableFilter.getCount(); i++)
    {
        pivotTableFilter.setHidden(i, false);
        Console.WriteLine(pivotTableFilter.isHidden(i));
    }
}
```

6.166 Класс PivotTableFilters

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

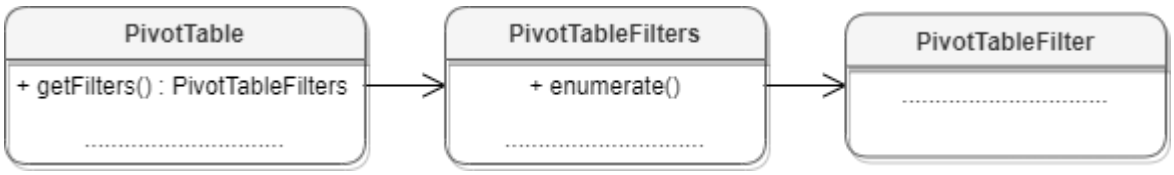


Рисунок 56 – Объектная модель классов для работы с фильтрами

6.166.1 Метод PivotTableFilters.getEnumerator

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

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
```

```
    PivotTableFilters pivotTableFilters = pivotTable.getFilters();
    PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
    foreach (var pivotTableFilter in enumerator)
    {
        Console.WriteLine(pivotTableFilter.getFieldName());
    }
}
```

6.167 Перечисление PivotTableFunction

Перечисление `PivotTableFunction` содержит функции, которые могут быть использованы в сводных таблицах. Используется в качестве поля `subtotalFunctions` класса `PivotTableCategoryField`.

Таблица 99 – Функции сводных таблиц

Значение	Описание
`PivotTableFunction.Auto`	Автозаполнение
`PivotTableFunction.Sum`	Суммирует все числовые данные
`PivotTableFunction.Count`	Количество всех ячеек
`PivotTableFunction.CountNums`	Количество числовых ячеек
`PivotTableFunction.Average`	Среднее значение
`PivotTableFunction.Max`	Наибольшее значение
`PivotTableFunction.Min`	Наименьшее значение
`PivotTableFunction.Product`	Произведение всех ячеек
`PivotTableFunction.StdDeviation`	Стандартное смещенное отклонение
`PivotTableFunction.StdDeviationPopulation`	Стандартное несмещенное отклонение
`PivotTableFunction.Variance`	Смещенная дисперсия

Значение	Описание
`PivotTableFunction.VariancePopulation`	Несмещенная дисперсия

6.168 Класс PivotTableItem

`PivotTableItem` описывает элемент сводной таблицы (см. Рисунок 2). См. пример в главе `PivotTableItems.GetEnumerator()`.

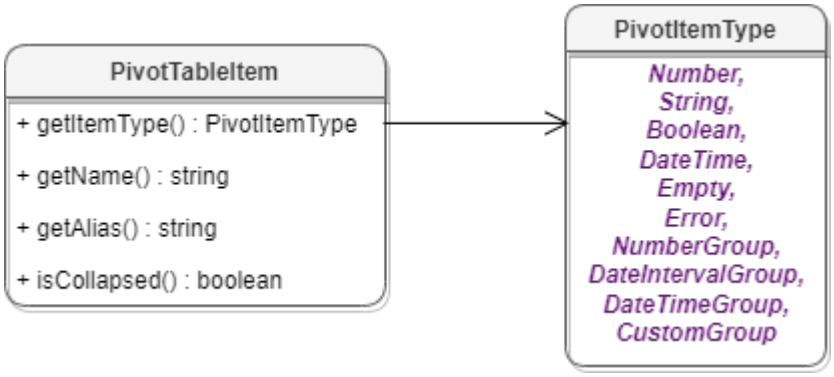


Рисунок 57 – Класс `PivotTableItem`

6.168.1 Метод PivotTableItem.getAlias

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем), тип - строка. См. пример в главе `PivotTableItems.GetEnumerator()`.

6.168.2 Метод PivotTableItem.getItemType

Метод возвращает тип `PivotTableItemType` элемента сводной таблицы. См. пример в главе `PivotTableItems.GetEnumerator()`.

6.168.3 Метод PivotTableItem.getName

Метод возвращает имя элемента сводной таблицы, тип - строка. См. пример в главе `PivotTableItems.GetEnumerator()`.

6.168.4 Метод PivotTableItem.isCollapsed

Метод возвращает `true`, если элемент сводной таблицы свернут. См. пример в главе `PivotTableItems.GetEnumerator()`.

6.169 Класс PivotTableItemForCategory

Класс `PivotTableItemForCategory` представляет собой элемент коллекции `PivotTableSlicePath`.

Таблица 100 – Описание полей класса `PivotTableItemForCategory`

Поле	Тип	Описание
`PivotTableItemForCategory.fieldName`	`string`	Название поля сводной таблицы
`PivotTableItemForCategory.item`	`PivotTableItem`	Элемент сводной таблицы

6.170 Класс PivotTableItems

Класс обеспечивает доступ к списку элементов сводной таблицы `PivotTable` (см. Рисунок 2).

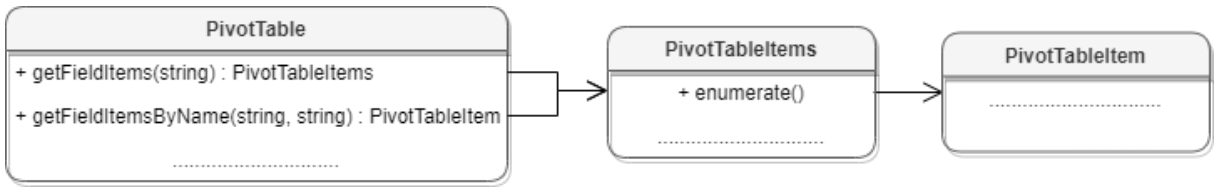


Рисунок 58 – Объектная модель классов для работы с элементами сводных таблиц

6.170.1 Метод PivotTableItems.GetEnumerator

Используется для перечисления элементов сводной таблицы.

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
```

```
    PivotTableItems pivotTableItems = pivotTable.getFieldItems("Age");
    PivotTableItemsEnumerator pivotTableItemsEnumerator =
pivotTableItems.GetEnumerator();
    foreach (var pivotTableItem in pivotTableItemsEnumerator)
    {
        Console.WriteLine(pivotTableItem.getAlias());
        Console.WriteLine(pivotTableItem.getName());
        Console.WriteLine(pivotTableItem.getItemType());
        Console.WriteLine(pivotTableItem.isCollapsed());
    }
}
```

6.171 Перечисление PivotTableItemType

Перечисление `PivotTableItemType` содержит возможные типы элементов сводной таблицы.

Таблица 101 – Типы элементов сводной таблицы

Значение	Описание
`PivotTableItemType.Number`	Числовой
`PivotTableItemType.String`	Строковый
`PivotTableItemType.Boolean`	Логический
`PivotTableItemType.DateTime`	Дата / время
`PivotTableItemType.Empty`	Пустой тип
`PivotTableItemType.Error`	Ошибка
`PivotTableItemType.NumberGroup`	Интервальная группировка
`PivotTableItemType.DateIntervalGroup`	Интервальная группировка по датам
`PivotTableItemType.DateTimeGroup`	Группировка по дате / времени

Значение	Описание
PivotTableItemType.CustomGroup	Пользовательская (произвольная) группировка

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
    PivotTableItems pivotTableItems = pivotTable.getFieldItems("Age");
    PivotTableItemsEnumerator pivotTableItemsEnumerator =
pivotTableItems.GetEnumerator();
    foreach (var pivotTableItem in pivotTableItemsEnumerator)
    {
        Console.WriteLine(pivotTableItem.getItemType());
    }
}
```

6.172 Класс PivotTableLayoutSettings

Класс PivotTableLayoutSettings содержит настройки отображения сводной таблицы. Данный объект может быть получен в результате вызова PivotTable.getPivotTableLayoutSettings() и установлен методом PivotTableEditor.setLayoutSettings().

Таблица 102 – Описание полей класса PivotTableLayoutSettings

Поле	Тип	Описание
PivotTableLayoutSettings.reportLayout	PivotTableReportLayout	Настройка вида макета сводной таблицы (компактный, табличный, структурный).
PivotTableLayoutSettings.valueFieldsOrientation	ValueFieldsOrientation	Настраивает положение значений в случае, если в

Поле	Тип	Описание
		сводной таблице более двух полей значений.
PivotTableLayoutSettings.pageFieldOrder	PageFieldOrder	Настройка порядка полей фильтров (вниз, затем поперек или сначала поперек, потом вниз)
PivotTableLayoutSettings.indentForCompactLayout	float	Размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей)
PivotTableLayoutSettings.pageFieldWrapCount	uint	Настройка связана с pageFieldOrder, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д)
PivotTableLayoutSettings.isMergeAndCenterLabelsEnabled	bool	Настройка позволяет объединить ячейки заголовков
PivotTableLayoutSettings.useGridDropZones	bool	Флаг, отвечающий за отображение классического вида (как в Excel 2003). Влияет только на расположение полей в отчете
PivotTableLayoutSettings.displayFieldCaptions	bool	Флаг, отвечающий за отображение заголовков полей

6.173 Класс PivotTablePageField

Содержит свойства поля из области фильтров (см. Таблицу 103). Объект может быть получен посредством вызова PivotTable.getPageFields().

Конструкторы

```
public PivotTablePageField()
```

```
public PivotTablePageField(string fieldName, string fieldAlias, string subtotalAlias)
```

Параметры

– `fieldName`: название поля, тип `string`.

– `fieldAlias`: псевдоним поля, тип `string`.

– `subtotalAlias`: псевдоним промежуточных итогов поля, тип `string`.

Таблица 103 – Описание полей класса `PivotTablePageField`

Поле	Тип	Описание
`PivotTablePageField.fieldProperties`	`PivotTableFieldProperties`	Свойства поля

6.174 Перечисление PivotTableReportLayout

Перечисление `PivotTableReportLayout` позволяет задать внешний вид отчетов сводной таблицы. Используется в качестве поля `reportLayout` класса `PivotTableLayoutSettings`.

Таблица 104 – Варианты внешнего вида отчетов

Значение	Описание
`PivotTableReportLayout.Compact`	Компактный вид
`PivotTableReportLayout.Tabular`	Табличный вид
`PivotTableReportLayout.Outline`	Структурный вид

6.175 Класс PivotTableSlicePath

Класс `PivotTableSlicePath` представляет собой путь до столбца или строки в сводной таблице. Данный класс является коллекцией объектов `PivotTableItemForCategory`. Используется в поле `PivotTableSortingParams.sortByValueSlice`.

6.176 Класс PivotTablesManager

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

Пример

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

6.176.1 Метод PivotTablesManager.create

Метод создает сводную таблицу на основе диапазона исходных данных.

Вызов

```
public PivotTable create(CellRange cellRange, Cell destination)
```

```
public PivotTable create(string formula, Cell destination)
```

Параметры

– cellRange: диапазон данных для сводной таблицы, тип CellRange.

– formula: формула, результатом которой является диапазон данных для сводной таблицы, тип string.

– destination: (необязательный) местоположение левой верхней ячейки сводной таблицы, тип Cell. Если местоположение не задано, создается новый лист, и сводная таблица будет расположена в его верхнем левом углу.

Возвращает

– созданная сводная таблица, тип PivotTable.

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
CellRange cellRange = sheet.getCellRange("B3:C4");
PivotTablesManager pivotTablesManager = document.getPivotTablesManager();
PivotTable pivotTable = pivotTablesManager.create(cellRange);
```

6.177 Перечисление PivotTableSortingOrder

Перечисление PivotTableSortingOrder позволяет задать порядок сортировки данных в сводной таблице. Используется в методах PivotTableEditor.setSortingByLabel() и

PivotTableEditor.setSortingByValue() и в поле PivotTableSortingParams.sortingOrder.

Таблица 105 – Варианты порядка сортировки данных в сводной таблице

Значение	Описание
PivotTableSortingOrder.Ascending	Сортировка по возрастанию
PivotTableSortingOrder.Descending	Сортировка по убыванию

6.178 Класс PivotTableSortingParams

Класс PivotTableSortingParams содержит настройки сортировки данных в сводной таблице. Данный класс используется в методе PivotTable.getSortingParams().

Таблица 106 – Описание полей класса PivotTableSortingParams

Поле	Тип	Описание
PivotTableSortingParams.sortByValueSlice	PivotTableSlicePath	Срез данных для сортировки по значениям, который содержит путь до столбца или строки с данными
PivotTableSortingParams.sortFieldSubtotal	PivotTableFunction	Функция для сортировки по подытогам, которая применена к текущему полю
PivotTableSortingParams.sortingOrder	PivotTableSortingOrder	Порядок сортировки
PivotTableSortingParams.sortingType	PivotTableSortingType	Тип сортировки
PivotTableSortingParams.valueFieldNameForSorting	string	Название поля значений, которое содержит данные для сортировки
PivotTableSortingParams.valueSliceSubtotal	PivotTableFunction	Функция для сортировки по подытогам, которая применена к sortByValueSlice полю

6.179 Перечисление PivotTableSortingType

Перечисление PivotTableSortingType содержит типы сортировки данных в сводной таблице. Используется в поле PivotTableSortingParams.sortingType.

Таблица 107 – Типы сортировки данных в сводной таблице

Значение	Описание
`PivotTableSortingType.Manual`	Ручная сортировка
`PivotTableSortingType.Label`	Сортировка по названию строк или столбцов
`PivotTableSortingType.Value`	Сортировка по значениям и итогам

6.180 Перечисление PivotTableUnsupportedFeature

Перечисление `PivotTableUnsupportedFeature` описывает неподдерживаемую функциональность сводных таблиц. Получение неподдерживаемой функциональности сводных таблиц описано в `PivotTable.getUnsupportedFeatures()`.

Таблица 108 – Описание значений `PivotTableUnsupportedFeature`

Значение	Описание
`PivotTableUnsupportedFeature.CalculatedItem`	Вычисляемые элементы
`PivotTableUnsupportedFeature.ShowDataAs`	Вычисления ("Show data" как в MS Excel)
`PivotTableUnsupportedFeature.MultipleFilters`	Множественная фильтрация
`PivotTableUnsupportedFeature.GroupFieldFilter`	Условная фильтрация для сгруппированных полей
`PivotTableUnsupportedFeature.UnsupportedFilter`	Неподдерживаемый фильтр

6.181 Перечисление PivotTableUpdateResult

В таблице 109 приведены значения, которые соответствуют возможным результатам обновления сводной таблицы (см. методы `PivotTable.update()`, `PivotTableEditor.apply()`).

Таблица 109 – Результаты обновления сводной таблицы

Значение	Описание
`PivotTableUpdateResult.Success`	Успешное обновление таблицы

Значение	Описание
PivotTableUpdateResult.NoPivotTable	Сводная таблица не найдена
PivotTableUpdateResult.NoSuchFieldInCategory	Не найдено поле в категории
PivotTableUpdateResult.NoSuchFieldInPivotTable	Не найдено поле в сводной таблице
PivotTableUpdateResult.InvalidIndex	Ошибка в индексе
PivotTableUpdateResult.FieldAlreadyEnabled	Поле уже существует
PivotTableUpdateResult.MovingFieldToTheSameCategoryForbidden	Попытка перемещения поля в рамках текущей категории
PivotTableUpdateResult.InvalidFunction	Неправильная функция
PivotTableUpdateResult.InvalidCategory	Неправильная область
PivotTableUpdateResult.InvalidDataSourceRange	Ошибка диапазона исходных данных
PivotTableUpdateResult.NoDataRowsInDataSource	В исходных данных нет строк с данными
PivotTableUpdateResult.EmptyDataSourceHeaders	Пустые заголовки исходных данных
PivotTableUpdateResult.NoReferenceUnderDefine	Попытка обновить или создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу
PivotTableUpdateResult.NoSuchItem	Элемент не найден
PivotTableUpdateResult.CannotExpandCollapseLeafItem	Не удается раскрыть свернутый элемент
PivotTableUpdateResult.AnotherPivotInsideDataSource	Найдена другая сводная таблица в этом же диапазоне
PivotTableUpdateResult.AnotherFieldHasTheSameName	Заданное альтернативное название совпадает с названием существующего поля
PivotTableUpdateResult.InvalidCalculationData	Неверно заданы настройки дополнительных вычислений

Значение	Описание
PivotTableUpdateResult.InvalidConditionalFilterParams	Неверно заданы настройки условного фильтра
PivotTableUpdateResult.Canceled	Обновление сводной таблицы отменено
PivotTableUpdateResult.NoSuchSheetInExternalDocument	Не найден лист во внешнем документе
PivotTableUpdateResult.InvalidSortingParams	Неправильно заданы настройки сортировки

6.182 Класс PivotTableValueField

Класс PivotTableValueField содержит свойства поля сводной таблицы, использующегося как значение столбец (см. Таблицу 110). Класс может быть получена посредством вызова PivotTable.getValueFields().

Таблица 110 – Описание полей класса PivotTableValueField

Поле	Тип	Описание
PivotTableValueField.baseFieldName	string	Оригинальное поле на основе которого было создано данное поле.
PivotTableValueField.valueFieldName	string	Автоматический уникальный псевдоним такой как “Sum of % имя поля%“.
PivotTableValueField.cellNumberFormat	CellFormat	Числовой формат для конкретного поля значений.
PivotTableValueField.totalFunction	PivotTableFunction	Агрегирующая функция поля значений (SUM, COUNT, MAX и т.д).
PivotTableValueField.customFormula	string	Вычисляемая формула для поля значений.
PivotTableValueField.calculationData	PivotTableCalculationData	Настройки дополнительных вычислений для поля значений.

6.183 Класс PointU

Класс `PointU` представляет собой точку в двухмерном пространстве.

Таблица 111 – Описание полей класса `PointU`

Поле	Тип	Описание
`PointU.x`	`float`	Координата точки по оси x
`PointU.y`	`float`	Координата точки по оси y

Примеры

```
PointU point = new PointU(50, 50);
```

```
PointU point = new PointU();
point.x = 50;
point.y = 50;
```

6.184 Класс Position

Класс `Position` представляет местоположение в текстовом документе. Используется для обозначения начала и конца диапазона `Range`.

6.184.1 Метод Position.compare

Метод сравнивает заданную позицию с текущей.

Вызов

```
public int? compare(Position other)
```

Параметры

– `other`: позиция для сравнения с текущей, тип `Position`.

Возвращает

– положительное расстояние, если текущая позиция больше заданной.

– отрицательное расстояние, если текущая позиция меньше заданной.

– 0, если позиции равны.

– `null`, если позиции нельзя сравнить (позиции в разных документах, в тексте и в колонтитуле и т.д.).

Примеры для текстового документа

```
Table table1 = document.getRange().getBegin().insertTable(2, 2, "table1");
Position positionDoc = document.getRange().getBegin();
Position positionTable = table1.getRange().getBegin();
Position positionCell = table1.getCell(new CellPosition(0,
0)).getRange().getBegin();

if (positionDoc.compare(positionTable) + positionTable.compare(positionCell) ==
0)
    // true
```

```
document.getRange().getBegin().insertText("Some text");
Position positionBegin = document.getRange().getBegin();
Position positionEnd = document.getRange().getEnd();

Console.WriteLine(positionBegin.compare(positionEnd)); // -10
Console.WriteLine(positionEnd.compare(positionBegin)); // 10
```

Пример для табличного документа

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

Position firstCellEnd = sheet.getCell("A1").getRange().getEnd();
Position secondCellBegin = sheet.getCell("B1").getRange().getBegin();

Console.WriteLine(firstCellEnd.compare(secondCellBegin)); // 0
```

6.184.2 Метод Position.getBefore

Метод возвращает позицию в текстовом документе, которая находится перед заданной таблицей.

Вызов

```
public static Position getBefore(Table table)
```

Параметры

– `table`: таблица в текстовом документе, тип `Table`.

Возвращает

– позиция перед таблицей, тип `Position`.

Пример

```
Table table = document.getRange().getBegin().insertTable(3, 3, "table1");
document.getRange().getBegin().insertText("Text in the first cell");
Position position = Position.getBefore(table);
position.insertSectionBreak(SectionBreakType.Continuous);
document.getRange().getBegin().insertText("Text before the table");
```

6.184.3 Метод Position.getCell

Метод возвращает ячейку `Cell` для заданной позиции.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A3");
NCT.MyOfficeSDK.Range range = cell.getRange();
Position position = range.getBegin();
Cell cellAtPosition = position.getCell();
```

6.184.4 Метод Position.getCurrentRange

Метод возвращает диапазон, в котором находится текущая позиция. Размер диапазона зависит от выбранной единицы текста.

Вызов

```
public Range getCurrentRange(TextUnit textUnit)
```

Параметры

– `textUnit`: единица текста, определяющая размер диапазона, тип `TextUnit`.

Возвращает

– диапазон документа, содержащий текущую позицию, тип `Range`.

Пример для текстового документа

```
Position docBegin = document.getRange().getBegin();
Range sentence = docBegin.getCurrentRange(TextUnit.Sentence);
Range word = docBegin.getCurrentRange(TextUnit.Word);
Range character = docBegin.getCurrentRange(TextUnit.Character);

Console.WriteLine(sentence.extractText());
// Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
```

```
incididunt ut labore et dolore magna aliqua.
Console.WriteLine(word.extractText()); // Lorem
Console.WriteLine(character.extractText()); // L
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("B2");
Position cellBegin = cell.getRange().getBegin();
Range sentence = cellBegin.getCurrentRange(TextUnit.Sentence);

Console.WriteLine(sentence.extractText()); // First sentence.
```

6.184.5 Метод Position.getNextPosition

Метод возвращает позицию, которая находится на заданном расстоянии после текущей. Если результат выходит за пределы текущего абзаца, метод возвращает конец абзаца.

Вызов

```
public Position getNextPosition(uint count)
```

Параметры

– `count`: (необязательный, по умолчанию `1`) расстояние до следующей позиции.

Возвращает

– следующая позиция, тип `Position`.

Пример для текстового документа

```
document.getRange().getBegin().insertText("New text");
document.getRange().getBegin().getNextPosition(2).insertText("!");
Console.WriteLine(document.getRange().extractText()); // Ne!w text
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("B2");
cell.setText("New text");
cell.getRange().getBegin().getNextPosition(2).insertText("!");

Console.WriteLine(cell.getFormattedValue()); // Ne!w text
```

6.184.6 Метод Position.getNextRange

Метод возвращает диапазон, который расположен после текущего диапазона. Размеры текущего и следующего диапазонов зависят от выбранной единицы текста. Если выбранная единица текста отлична от `Paragraph`, то работа метода `Position.getNextRange` ограничена текущим абзацем.

Вызов

```
public Range getNextRange(TextUnit textUnit)
```

Параметры

– `textUnit`: единица текста, определяющая размеры диапазонов, тип `TextUnit`.

Возвращает

– следующий диапазон документа, тип `Range`.

Пример для текстового документа

```
Position docBegin = document.getRange().getBegin();
Range nextSentence = docBegin.getNextRange(TextUnit.Sentence);
Range nextWord =
docBegin.getNextRange(TextUnit.Word).getBegin().getNextRange(TextUnit.Word);
Range nextCharacter = docBegin.getNextRange(TextUnit.Character);

Console.WriteLine(nextSentence.extractText());
// Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
Console.WriteLine(nextWord.extractText()); // ipsum
Console.WriteLine(nextCharacter.extractText()); // o
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("B2");
Position cellBegin = cell.getRange().getBegin();
Range sentence = cellBegin.getNextRange(TextUnit.Sentence);

Console.WriteLine(sentence.extractText()); // Second sentence.
```

6.184.7 Метод Position.getParagraph

Метод возвращает абзац (Paragraph) для текущей позиции.

Пример для текстового документа

```
Position position = document.getRange().getBegin();
Paragraph paragraphAtPosition = position.getParagraph();
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("B2");
Position position = cell.getRange().getBegin();
Paragraph paragraphAtPosition = position.getParagraph();
```

6.184.8 Метод Position.getPreviousPosition

Метод возвращает позицию, которая находится на заданном расстоянии перед текущей. Если результат выходит за пределы текущего абзаца, метод возвращает начало абзаца.

Вызов

```
public Position getPreviousPosition(uint count)
```

Параметры

– `count`: (необязательный, по умолчанию `1`) расстояние до предыдущей позиции.

Возвращает

– предыдущая позиция, тип Position.

Пример для текстового документа

```
document.getRange().getBegin().insertText("New text");
document.getRange().getContentEnd().getPreviousPosition(2).insertText("!");
Console.WriteLine(document.getRange().extractText()); // New te!xt
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("B2");
cell.setText("New text");
cell.getRange().getContentEnd().getPreviousPosition(2).insertText("!");

Console.WriteLine(cell.getFormattedValue()); // New te!xt
```

6.184.9 Метод Position.getPreviousRange

Метод возвращает диапазон, который расположен перед текущим диапазоном. Размеры текущего и предыдущего диапазонов зависят от выбранной единицы текста. Если выбранная единица текста отлична от `Paragraph`, то работа метода `Position.getPreviousRange` ограничена текущим абзацем.

Вызов

```
public Range getPreviousRange(TextUnit textUnit)
```

Параметры

– `textUnit`: единица текста, определяющая размеры диапазонов, тип `TextUnit`.

Возвращает

– предыдущий диапазон документа, тип `Range`.

Пример для текстового документа

```
Position sentenceBegin =
document.getRange().getBegin().getNextRange(TextUnit.Sentence).getBegin();
Range previousSentence = sentenceBegin.getPreviousRange(TextUnit.Sentence);
Range previousWord = sentenceBegin.getPreviousRange(TextUnit.Word);
while (previousWord.getContentEnd().compare(previousWord.getBegin()) == 1) {
    previousWord = previousWord.getBegin().getPreviousRange(TextUnit.Word);
}
Range previousCharacter = sentenceBegin.getPreviousRange(TextUnit.Character);
previousCharacter =
previousCharacter.getBegin().getPreviousRange(TextUnit.Character);
previousCharacter =
previousCharacter.getBegin().getPreviousRange(TextUnit.Character);

Console.WriteLine(previousSentence.extractText());
// Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
Console.WriteLine(previousWord.extractText()); // aliqua
Console.WriteLine(previousCharacter.extractText()); // a
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("B2");
Position cellEnd = cell.getRange().getContentEnd();
```

```
Range sentence = cellEnd.getPreviousRange(TextUnit.Sentence);


Console.WriteLine(sentence.extractText()); // Second sentence.
```

6.184.10 Метод Position.getStyle

Метод возвращает стиль фрагмента текста, в котором расположена текущая позиция. Данный метод в основном специализируется на получении стилей из документов, созданных в редакторе Microsoft Word. В большинстве случаев рекомендуется использовать метод Paragraph.getStyle() или Paragraph.getResolvedStyle().

Вызов

```
public TextStyle getStyle()
```

Возвращает

– стиль фрагмента, в котором расположена текущая позиция, тип TextStyle.

– null, если стиль не задан.

Пример

```
Position position = document.getRange().getBegin().getNextPosition(12);
Console.WriteLine(position.getStyle().getName());
```

6.184.11 Метод Position.insertBookmark

Вставляет закладку с наименованием в текущую позицию.

Пример

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

6.184.12 Метод Position.insertEndnote

Метод вставляет концевую сноску в текущую позицию.

Вызов

```
public Position insertEndnote()
```

Возвращает

– позиция содержимого концевой сноски, тип Position.

Пример

```
Range range = document.getRange();
Range sentence = range.getBegin().getCurrentRange(TextUnit.Sentence);
Position pos = sentence.getContentEnd().insertEndnote();
pos.insertText("Endnote");
```

6.184.13 Метод Position.insertFootnote

Метод вставляет сноску в текущую позицию.

Вызов

```
public Position insertFootnote()
```

Возвращает

– позиция содержимого сноски, тип `Position`.

Пример

```
Range range = document.getRange();
Range sentence = range.getBegin().getCurrentRange(TextUnit.Sentence);
Position pos = sentence.getContentEnd().insertFootnote();
pos.insertText("Footnote");
```

6.184.14 Метод Position.insertHyperlink

Метод `insertHyperlink` вставляет ссылку в текущую позицию. В качестве параметров передаются адрес ссылки и текст ссылки.

Вызов

```
public void insertHyperlink(string url, string label)
```

Параметры

– `url` – адрес ссылки;

– `label` – текст ссылки.

Пример

```
Position pos = document.getRange().getBegin();
pos.insertHyperlink("https://testhyperlink.com", "Hyperlink");
```

6.184.15 Метод Position.insertImage

Вставляет изображение из файла в текущую позицию. Возвращает объект `Image`, содержащий вставленное изображение. Метод может быть использован только в текстовом документе.

Параметры

– `url` – полный путь к файлу, строка;

– `size` – геометрические размеры изображения для вставки, тип `SizeU`.

Пример

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

6.184.16 Метод Position.insertLineBreak

Метод предназначен для вставки перевода строки.

Пример

```
Range range = document.getRange();
Position endPosition = range.getEnd();
endPosition.insertLineBreak();
```

6.184.17 Метод Position.insertPageBreak

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

Пример

```
Range range = document.getRange();
Position endPosition = range.getEnd();
endPosition.insertPageBreak();
```

6.184.18 Метод Position.insertSectionBreak

Вставляет разрыв раздела в текущую позицию. В качестве параметра выступает тип `SectionBreakType`. При вызове без параметра используется значение `SectionBreakType.NextPage`.

Примеры

```
Range range = document.getRange();
Position endPosition = range.getEnd();
endPosition.insertSectionBreak();
```

```
endPosition.insertSectionBreak(SectionBreakType.EvenPage);
```

6.184.19 Метод Position.insertTable

Метод предназначен для вставки таблицы в текстовый документ или страницы в табличный документ. В качестве параметров передаются число строк, столбцов, а также имя таблицы. Метод возвращает объект таблицы.

```
Table insertTable(int rows, int cols, String tableName);
```

Следует учитывать, что при вставке таблицы к ее имени автоматически добавляется порядковый номер, начинающийся с единицы. Таким образом, вызов

```
Table table = position.insertTable(3, 3, "Table");
```

приведет к созданию таблицы с именем «Table1».

Примеры добавления таблицы в текстовый документ приведены в разделе Работа с таблицами текстового документа.

Примеры добавления страницы в текстовый документ приведены в разделе Работа с листами табличного документа.

6.184.20 Метод Position.insertText

Метод предназначен для вставки текстовой строки в заданное местоположение в документе.

Пример

```
Range range = document.getRange();
Position startPosition = range.getBegin();
startPosition.insertText("Текст в начале строки");
```

6.184.21 Метод Position.removeBackward

Метод удаляет заданное количество объектов (символов, картинок и т.д.) до текущей позиции.

Пример

```
Range range = document.getRange();
Position beginPosition = range.getBegin();
beginPosition.removeBackward(3);
```

6.184.22 Метод Position.removeForward

Метод удаляет заданное количество объектов (символов, картинок и т.д.) после текущей позиции.

Пример

```
Range range = document.getRange();
Position beginPosition = range.getBegin();
beginPosition.removeForward(3);
```

6.185 Перечисление PredefinedTextStyle

Перечисление `PredefinedTextStyle` содержит стандартные стили абзаца. Используется в методах `TextStyles.create()` и `TextStyles.get()`.

Таблица 112 – Описание стандартных стилей абзацев

Значение	Описание
`PredefinedTextStyle.Normal`	Обычный
`PredefinedTextStyle.Heading1`	Заголовок 1
`PredefinedTextStyle.Heading2`	Заголовок 2
`PredefinedTextStyle.Heading3`	Заголовок 3
`PredefinedTextStyle.Heading4`	Заголовок 4
`PredefinedTextStyle.Heading5`	Заголовок 5
`PredefinedTextStyle.Heading6`	Заголовок 6
`PredefinedTextStyle.Heading7`	Заголовок 7

Значение	Описание
`PredefinedTextStyle.Heading8`	Заголовок 8
`PredefinedTextStyle.Heading9`	Заголовок 9
`PredefinedTextStyle.Title`	Название документа
`PredefinedTextStyle.Subtitle`	Подзаголовок
`PredefinedTextStyle.HeaderFooter`	Колонтитул
`PredefinedTextStyle.Footnote`	Сноска
`PredefinedTextStyle.Endnote`	Концевая сноска
`PredefinedTextStyle.Contents1`	Содержание 1
`PredefinedTextStyle.Contents2`	Содержание 2
`PredefinedTextStyle.Contents3`	Содержание 3
`PredefinedTextStyle.Contents4`	Содержание 4
`PredefinedTextStyle.Contents5`	Содержание 5
`PredefinedTextStyle.Contents6`	Содержание 6
`PredefinedTextStyle.Contents7`	Содержание 7
`PredefinedTextStyle.Contents8`	Содержание 8
`PredefinedTextStyle.Contents9`	Содержание 9
`PredefinedTextStyle.Hyperlink`	Ссылка

6.186 Класс PresentationExportSettings

Класс `PresentationExportSettings` предоставляет настройки, необходимые для экспорта презентационных документов (см. `Document.exportAs`).

Таблица 113 – Описание полей класса `PresentationExportSettings`

Поле	Тип	Описание
`PresentationExportSettings.skipHiddenSlides`	`bool`	Исключает из созданного документа скрытые слайды

Пример

```
PresentationExportSettings exportSettings = new PresentationExportSettings();
exportSettings.skipHiddenSlides = false;
document.exportAs(filePath, ExportFormat.PDFA1, exportSettings);
```

6.187 Класс PrintingScope

Класс `PrintingScope` содержит настройки для экспорта табличных документов. Используется в поле `printingScope` класса `WorkbookExportSettings`.

Позволяет создать области печати следующих типов:

- выбранная область печати либо весь документ (см. `PrintingScope.Type`);
- указанный диапазон ячеек (см. `CellRangePosition`).

Примеры

```
// по умолчанию - PrintingScope.Type.PrintArea
PrintingScope printingScope = new PrintingScope();
```

```
// область печати
PrintingScope printingScope = new PrintingScope(PrintingScope.Type.PrintArea);
```

```
// диапазон ячеек
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
PrintingScope printingScope = new PrintingScope(cellRangePosition);
```

6.187.1 Метод PrintingScope.getCellRange

Метод возвращает диапазон ячеек таблицы (см. `CellRangePosition`).

6.187.2 Метод PrintingScope.usePrintArea

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

6.187.3 Перечисление PrintingScope.Type

Варианты выбора диапазона страниц для экспорта и печати представлены в таблице 114. Используется в `PrintingScope`.

Таблица 114 – Диапазон страниц для экспорта и печати

Значение	Описание
`PrintingScope.Type.PrintArea`	Выбранная область печати (по умолчанию)
`PrintingScope.Type.WholeSheet`	Печать всего документа

6.188 Класс PrintTitles

Класс `PrintTitles` представляет собой сквозной заголовок для экспорта документов. Сквозной заголовок – это диапазон ячеек, который будет отображаться на каждой странице экспортируемого листа. Данный класс используется в методах `Table.getPrintTitles()` и `Table.setPrintTitles()`.

Конструкторы

```
PrintTitles()
```

```
PrintTitles(int? top, int? left, int? bottom, int? right)
```

Таблица 115 – Описание полей класса `PrintTitles`

Поле	Тип	Описание
`PrintTitles.bottomRow`	`int`	Индекс нижней строки диапазона
`PrintTitles.leftColumn`	`int`	Индекс левой колонки диапазона
`PrintTitles.rightColumn`	`int`	Индекс правой колонки диапазона
`PrintTitles.topRow`	`int`	Индекс верхней строки диапазона

Пример

```
Table sheet = document.getBlocks().getTable(0);
PrintTitles titles = new PrintTitles(4, 4, 7, 7);
sheet.setPrintTitles(titles);

document.exportAs("exportedDoc.pdf", ExportFormat.PDFA1);
```

6.188.1 Метод PrintTitles.create

Метод создает новый сквозной заголовок с заданными координатами.

Вызов

```
public static PrintTitles create(uint top, uint left, uint bottom, uint right)
```

Параметры

– `top`: индекс верхней строки диапазона.

– `left`: индекс левой колонки диапазона.

– `bottom`: индекс нижней строки диапазона.

– `right`: индекс правой колонки диапазона.

Возвращает

– сквозной заголовок, тип `PrintTitles`.

Пример

```
PrintTitles titles = PrintTitles.create(4, 4, 7, 7);
Console.WriteLine(titles.isRowsCols()); // True
```

6.188.2 Метод PrintTitles.createPrintTitlesCols

Метод создает новый сквозной заголовок состоящий из целых колонок.

Вызов

```
public static PrintTitles createPrintTitlesCols(uint first, uint last)
```

Параметры

– `first`: индекс первой колонки диапазона.

– `last`: индекс последней колонки диапазона.

Возвращает

– сквозной заголовок, тип `PrintTitles`.

Пример

```
PrintTitles titles = PrintTitles.createPrintTitlesCols(0, 2);
Console.WriteLine(titles.isCols()); // True
```

6.188.3 Метод PrintTitles.createPrintTitlesRows

Метод создает новый сквозной заголовок состоящий из целых строк.

Вызов

```
public static PrintTitles createPrintTitlesRows(uint first, uint last)
```

Параметры

– `first`: индекс первой строки диапазона.

– `last`: индекс последней строки диапазона.

Возвращает

– сквозной заголовок, тип `PrintTitles`.

Пример

```
PrintTitles titles = PrintTitles.createPrintTitlesRows(0, 0);
Console.WriteLine(titles.isRows()); // True
```

6.188.4 Метод PrintTitles.isCols

Метод позволяет определить состоит ли сквозной заголовок только из целых колонок.

Вызов

```
public bool isCols()
```

Возвращает

– `true`, если сквозной заголовок состоит из целых колонок, в ином случае – `false`.

6.188.5 Метод PrintTitles.isRows

Метод позволяет определить состоит ли сквозной заголовок только из целых строк.

Вызов

```
public bool isRows()
```

Возвращает

– `true`, если сквозной заголовок состоит из целых строк, в ином случае – `false`.

6.188.6 Метод PrintTitles.isRowsCols

Метод позволяет определить состоит ли сквозной заголовок только из целых строк или столбцов.

Вызов

```
public bool isRowsCols()
```

Возвращает

– `true`, если сквозной заголовок не состоит из целых строк или колонок, в ином случае – `false`.

6.189 Класс Range

Класс `Range` предоставляет доступ к диапазону документа. На рисунке 2 изображена объектная модель классов, относящихся к работе с диапазонами.

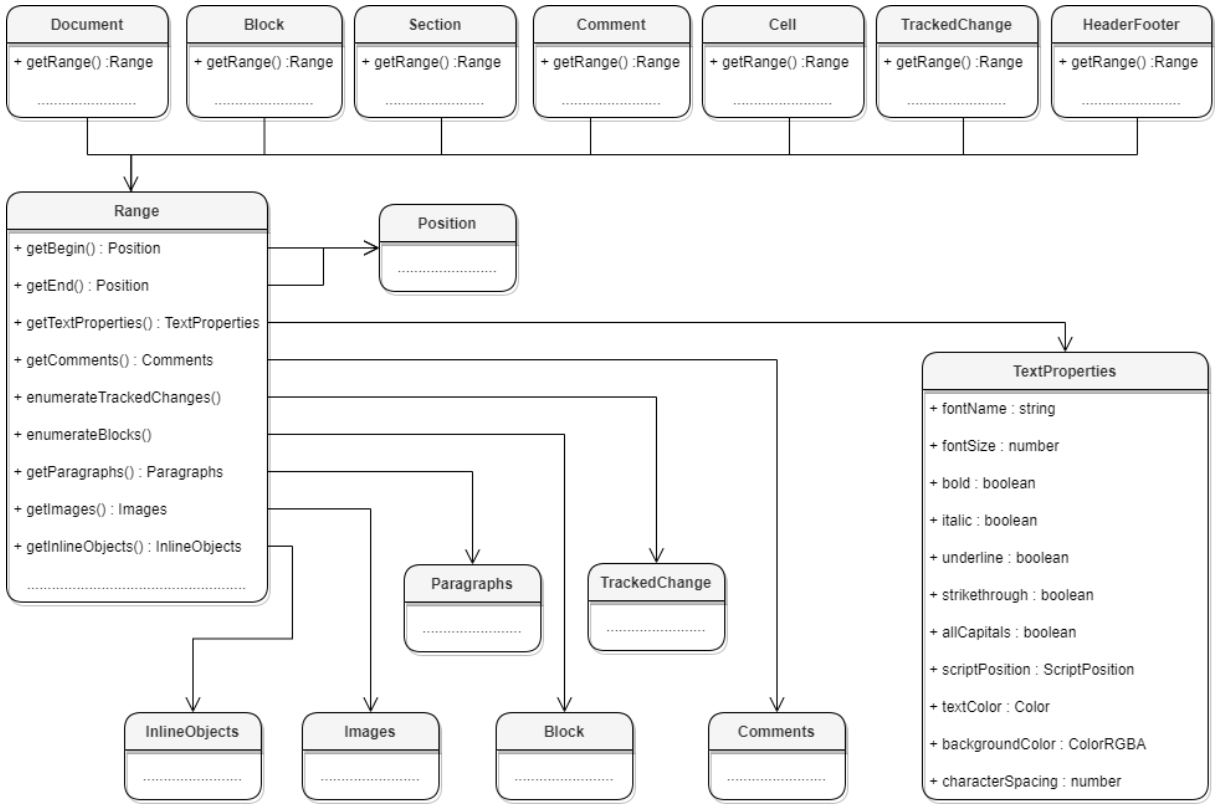


Рисунок 59 – Объектная модель для работы с классом `Range`

Варианты получения диапазона для текстового документа

```
// диапазон всего документа
Range range = document.getRange();

// диапазон блока
Block block = document.getBlocks().getBlock(0);
if (block != null) {
    Range blockRange = block.getRange();
```

```
}

// диапазон секций
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getRange().extractText());
}

// диапазон комментариев
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
foreach (var comment in commentsEnumerator)
{
    Console.WriteLine(comment.getRange().extractText());
}

// диапазон ячейки
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
}

// диапазон верхних колонтитулов
Block block = document.getBlocks().getBlock(0);
if (block != null)
{
    Section section = block.getSection();
    HeadersFooters headers = section.getHeaders();
    HeaderFootersEnumerator headerFootersEnumerator = headers.GetEnumerator();
    foreach (var headerFooter in headerFootersEnumerator)
    {
        Console.WriteLine(headerFooter.getRange().extractText());
    }
}
```

```
// диапазон отслеживаемых изменений
var trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator)
{
    Console.WriteLine(trackedChange.getType().ToString());
    Console.WriteLine(trackedChange.getInfo().author.name);
    Console.WriteLine(trackedChange.getRange().extractText());
}
```

6.189.1 Конструктор Range

Для создания объекта Range вы можете использовать следующий конструктор:

```
Range documentRange = new Range(begin, end)
```

Параметры

– begin: начальная позиция диапазона, тип Position;

– end: конечная позиция диапазона, тип Position.

6.189.2 Метод Range.copyInto

Метод копирует текущий диапазон в заданную позицию.

Вызов

```
public void copyInto(Position destination)
```

Параметры

– destination: конечная позиция, тип Position.

Пример

Этот пример копирует первый абзац в конец документа:

```
Position startPos = document.getRange().getBegin();
Range paragraph = startPos.getCurrentRange(TextUnit.Paragraph);
Position pos = document.getRange().getEnd();
paragraph.copyInto(pos);
```

6.189.3 Метод Range.extractText

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и другие объекты игнорируются.

Пример для текстового документа

```
Range range = document.getRange();
Console.WriteLine(range.ExtractText());
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    Console.WriteLine(cellRange.ExtractText());
}
```

6.189.4 Метод Range.getBegin

Метод возвращает позицию в начале диапазона.

Пример для текстового документа

```
Position beginDocPosition = document.getRange().getBegin();
beginDocPosition.insertText("API");
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    Position beginDocPos = cellRange.getBegin();
    beginDocPos.insertText("API");
}
```

6.189.5 Метод Range.getBlocksEnumerator

Предоставляет возможность итерации по блокам.

Пример для текстового документа

```
Range range = document.getRange();
BlocksEnumerator blocksEnumerator = range.getBlocksEnumerator();
foreach (var block in blocksEnumerator)
{
    Console.WriteLine(block.getRange().extractText());
}
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    BlocksEnumerator blocksEnumerator = cellRange.getBlocksEnumerator();
    foreach (var block in blocksEnumerator)
    {
        Console.WriteLine(block.getRange().extractText());
    }
}
```

6.189.6 Метод Range.getComments

Обеспечивает доступ к комментариям в диапазоне.

Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам. Если дат нет, то порядок комментариев не определен.

Пример

```
var commentsEnumerator = document.getRange().getComments().GetEnumerator();
foreach (var comment in commentsEnumerator)
{
    Console.WriteLine(comment.getText());
    Console.WriteLine(comment.getInfo().author.name);
```

```
    Console.WriteLine(comment.getRange().extractText());
}
```

6.189.7 Метод Range.getContentEnd

Метод возвращает позицию в конце содержимого текущего диапазона.

Вызов

```
public Position getContentEnd()
```

Возвращает

– позиция конца диапазона, тип Position.

Метод `Range.getContentEnd` может использоваться для добавления информации в конец содержимого абзаца/ячейки. Чтобы получить позицию конца диапазона с переходом к следующему абзацу/ячейке, вызовите метод `Range.getEnd`.

Пример для текстового документа

```
document.getRange().getBegin().insertText("Text");
document.getRange().getContentEnd().insertText("#");
Console.WriteLine(document.getRange().extractText()); // Text#\n
```

Пример для табличного документа

```
Cell cell = document.getBlocks().getTable(0).getCell("B2");
cell.setText("Text");
Position cellEnd = cell.getRange().getContentEnd();
cellEnd.insertText("#");
Console.WriteLine(cell.getRawValue()); // Text#
```

6.189.8 Метод Range.getEnd

Метод возвращает позицию в конце диапазона, включая символ перехода на новую строку/ячейку.

Метод `Range.getEnd` может использоваться для перехода к следующему абзацу/ячейке. Чтобы получить позицию конца диапазона для добавления в него информации, вызовите метод `Range.getContentEnd`.

Пример для текстового документа

```
document.getRange().getBegin().insertText("First Paragraph");
Position endDocPosition = document.getRange().getEnd();
```

```
endDocPosition.insertText("Second Paragraph");
Console.WriteLine(document.getRange().extractText()); // First Paragraph\nSecond
Paragraph\n
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
sheet.getCell("B2").setText("Text");
Position cellEnd = sheet.getCell("B2").getRange().getEnd();
cellEnd.insertText("#");
Console.WriteLine(sheet.getCell("B2").getRawValue()); // Text
Console.WriteLine(sheet.getCell("C2").getRawValue()); // #
```

6.189.9 Метод Range.getFootnotesEndnotes

Метод возвращает коллекцию сносок и концевых сносок в текущем диапазоне.

Вызов

```
public FootnotesEndnotes getFootnotesEndnotes()
```

Возвращает

– коллекция сносок и концевых сносок, тип FootnotesEndnotes.

Пример

```
Range range = document.getRange();
FootnotesEndnotes notes = range.getFootnotesEndnotes();
foreach (FootnoteEndnote note in notes) {
    Console.WriteLine(note.getType());
    Console.WriteLine(note.getRange().extractText());
}
```

6.189.10 Метод Range.getImages

Обеспечивает доступ к изображениям (Image) в диапазоне.

Пример

```
Images images = document.getRange().getImages();
ImagesEnumerator imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
```

```
    Console.WriteLine(image.getFrame().getWrapType());
}
```

6.189.11 Метод Range.getInlineObjects

Обеспечивает доступ к перечислению `MediaObjects` графических объектов диапазона.

Пример

```
Range range = document.getRange();
MediaObjects mediaObjects = range.getInlineObjects();
MediaObjectsEnumerator mediaObjectsEnumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    ..............................
}
```

6.189.12 Метод Range.getParagraphs

Обеспечивает доступ к абзацам `Paragraphs` в диапазоне.

Пример для текстового документа

```
Paragraphs paragraphs = document.getRange().getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    Console.WriteLine(paragraph.getRange().extractText());
}
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    Cell cell = table.getCell("B2");
    Range range = cell.getRange();
    Paragraphs paragraphs = range.getParagraphs();
    ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
    foreach (var paragraph in paragraphsEnumerator)
    {
```

```
        Console.WriteLine(paragraph.getRange().extractText());
    }
}
```

6.189.13 Метод Range.getStyle

Метод возвращает стиль фрагментов текста в текущем диапазоне. Данный метод в основном специализируется на получении стилей из документов, созданных в редакторе Microsoft Word. В большинстве случаев рекомендуется использовать метод Paragraph.getStyle() или Paragraph.getResolvedStyle().

Вызов

```
public TextStyle getStyle()
```

Возвращает

– стиль фрагментов текста в текущем диапазоне, тип TextStyle.

– null, если стиль не задан, или фрагменты диапазона содержат разные стили.

Пример

```
Range range = document.getRange().getBegin().getNextRange(TextUnit.Sentence);
Console.WriteLine(range.getStyle().getName());
```

6.189.14 Метод Range.getTextProperties

Метод возвращает объект с текущими настройками форматирования для фрагмента текстового документа. Описание настроек форматирования осуществляется с помощью объекта TextProperties.

Поля возвращаемого объекта TextProperties могут быть null, если фрагмент документа содержит текст с разными настройками.

Пример для текстового документа

```
Range range = document.getRange();
TextProperties textProperties = range.getTextProperties();
Console.WriteLine(textProperties.fontName);
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
```

```
    Range cellRange = cell.getRange();
    TextProperties textProperties = cellRange.getTextProperties();
    Console.WriteLine(textProperties.fontName);
}
```

6.189.15 Метод Range.getTrackedChangesEnumerator

Предоставляет возможность итерации по отслеживаемым изменениям `TrackedChange`. Метод может быть использован только в текстовых документах.

Пример

```
var trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator) {
    Console.WriteLine(trackedChange.getRange().extractText());
}
```

6.189.16 Метод Range.isContentLocked

Метод возвращает значение `true`, если изменения содержимого диапазона запрещены (см. Защита диапазона текстового документа).

Пример для текстового документа

```
Range range = document.getRange();
Console.WriteLine(range.isContentLocked());
```

Пример для таблицы внутри текстового документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    Console.WriteLine(cellRange.isContentLocked());
}
```

6.189.17 Метод Range.lockContent

Метод запрещает изменения содержимого диапазона (см. Защита диапазона текстового документа).

Метод может быть использован только в текстовых документах.

Пример для текстового документа

```
Range range = document.getRange();
range.lockContent();
Console.WriteLine(range.isContentLocked());
```

Пример для таблицы внутри текстового документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    cellRange.lockContent();
    Console.WriteLine(cellRange.isContentLocked());
}
```

6.189.18 Метод Range.moveInto

Метод перемещает текущий диапазон в заданную позицию.

Вызов

```
public void moveInto(Position destination)
```

Параметры

– `destination`: конечная позиция, тип `Position`.

Пример

Этот пример перемещает первое предложение в начало следующего абзаца:

```
Position startPos = document.getRange().getBegin();
Range sentence = startPos.getCurrentRange(TextUnit.Sentence);
Position pos = startPos.getCurrentRange(TextUnit.Paragraph).getEnd();
sentence.moveInto(pos);
```

6.189.19 Метод Range.removeContent

Метод полностью удаляет содержимое диапазона.

Пример для текстового документа

```
Range range = document.getRange();
range.lockContent();
Console.WriteLine(range.ExtractText());
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    cellRange.removeContent();
    Console.WriteLine(cellRange.ExtractText());
}
```

6.189.20 Метод Range.replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример для текстового документа

```
Range range = document.getRange();
range.replaceText("Range text");
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    cellRange.replaceText("New text");
}
```

6.189.21 Метод Range.setHyperlink

Метод `setHyperlink` вставляет ссылку в содержимое диапазона и заменяет его текст текстом ссылки.

Вызов

```
public void setHyperlink(string url, string label)
```

Параметры

– `url` – адрес ссылки;

– `label` – текст ссылки.

Пример для текстового документа

```
Range range = document.getRange();
range.setHyperlink("https://testhyperlink.com", "Hyperlink");
```

Пример для табличного документа

```
Table sheet = document.getBlocks().getTable(0);
Cell cell = sheet.getCell("B2");
Range cellRange = cell.getRange();
cellRange.setHyperlink("https://testhyperlink.com", "Hyperlink");
```

6.189.22 Метод Range.setTextProperties

Метод применяет настройки форматирования `TextProperties` для диапазона.

Пример для текстового документа

```
Range range = document.getRange();
TextProperties textProperties = range.getTextProperties();
textProperties.fontName = "Arial";
range.setTextProperties(textProperties);
```

Пример для табличного документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    TextProperties textProperties = cellRange.getTextProperties();
    textProperties.fontName = "Arial";
    cellRange.setTextProperties(textProperties);
}
```

6.189.23 Метод Range.unlockContent

Метод разрешает изменения содержимого диапазона (см. Защита диапазона текстового документа).

Метод может быть использован только в текстовых документах.

Пример для текстового документа

```
Range range = document.getRange();
range.unlockContent();
Console.WriteLine(range.isContentLocked());
```

Пример для таблицы внутри текстового документа

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    cellRange.unlockContent();
    Console.WriteLine(cellRange.isContentLocked());
}
```

6.190 Класс RangeBorders

Класс `RangeBorders` оставлен для совместимости. Вместо него следует использовать класс `Borders`.

6.191 Класс RectU

Класс `RectU` описывает прямоугольник в двухмерном пространстве.

Таблица 116 – Описание полей класса `RectU`

Поле	Тип	Описание
`RectU.topLeft`	`PointU`	Координата левой верхней вершины прямоугольника
`RectU.bottomRight`	`PointU`	Координата правой нижней вершины прямоугольника

Примеры

```
RectU rect = new RectU(new PointU(50, 50), new PointU(50, 50));
```

```
RectU rect = new RectU();
rect.topLeft = new PointU(50, 50);
rect.bottomRight = new PointU(50, 50);
```

6.192 Класс SaveDocumentSettings

Класс `SaveDocumentSettings` предоставляет настройки, используемые для сохранения документа в файл (см. `Document.saveAs()`).

Таблица 117 – Описание полей класса `SaveDocumentSettings`

Поле	Тип	Описание
`SaveDocumentSettings.documentFormat`	`DocumentFormat`	Формат документа (обязательно)
`SaveDocumentSettings.documentType`	`DocumentType`	Тип документа (обязательно)
`SaveDocumentSettings.documentPassword`	`string`	Пароль для защиты электронного документа от несанкционированного доступа
`SaveDocumentSettings.isTemplate`	`bool`	Флаг, обозначающий, что документ должен быть сохранен как шаблон
`SaveDocumentSettings.dsvSettings`	`DSVSettings`	Настройки для сохранения документа в формате DSV
`SaveDocumentSettings.documentMetaInfo`	`MetaInfo`	Дополнительная информация о сохраняемом файле
`SaveDocumentSettings.allowCalculation`	`bool`	Флаг, обозначающий, что формулы в документе должны быть пересчитаны перед сохранением
`SaveDocumentSettings.vbaEnabled`	`bool`	Позволяет сохранять табличные документы с VBA макрокомандами (.xlsm файлы)

6.193 Перечисление ScaleFrom

Варианты якоря для масштабирования `AbsoluteFrame` представлены в таблице 118. Используется в качестве параметра `scaleFrom` метода `AbsoluteFrame.scale()`.

Таблица 118 – Варианты для якоря масштабирования

Значение	Описание
`ScaleFrom.BottomRight`	Правый нижний угол
`ScaleFrom.BottomLeft`	Левый нижний угол
`ScaleFrom.TopLeft`	Левый верхний угол
`ScaleFrom.TopRight`	Правый верхний угол

6.194 Класс ScientificCellFormatting

Класс `ScientificCellFormatting` содержит параметры для экспоненциального формата ячеек таблицы. Используется в качестве аргумента метода `Cell.setFormat()`.

Таблица 119 – Описание полей класса `ScientificCellFormatting`

Поле	Тип	Описание
`ScientificCellFormatting.decimalPlaces`	`byte`	Количество десятичных позиций
`ScientificCellFormatting.minExponentDigits`	`byte`	Минимальное количество позиций экспоненты

Пример

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

ScientificCellFormatting cellFormat = new ScientificCellFormatting();
cellFormat.decimalPlaces = 2;
cellFormat.minExponentDigits = 3;

cell.setFormat(cellFormat);
Console.WriteLine(cell.getFormattedValue());
```

6.195 Класс Script

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

6.195.1 Метод Script.getBody

Метод возвращает текст макрокоманды в виде строки.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
    foreach (var script in scriptsEnumerator)
    {
        Console.WriteLine(script.getBody());
    }
}
```

6.195.2 Метод Script.getName

Метод возвращает имя макрокоманды.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
    foreach (var script in scriptsEnumerator)
    {
        Console.WriteLine(script.getName());
    }
}
```

6.195.3 Метод Script.setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

Пример

```
foreach (var script in scriptsEnumerator)
{
    script.setName("local scripts = document:getScripts()\nfor script in scripts
: enumerate() do\nprint(script:getName())\nend");
    Console.WriteLine(script.getName());
}
```

6.195.4 Метод Script.setName

Метод устанавливает имя для макрокоманды.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
    foreach (var script in scriptsEnumerator)
    {
        script.setName("Script name");
        Console.WriteLine(script.getName());
    }
}
```

6.196 Класс Scripting

Объект класса `Scripting` может быть получен путем вызова `DocumentAPI.createScripting(document)`, и содержит метод `runScript`, который используется для запуска макрокоманды.

Пример

```
Scripting scripting = DocumentAPI.createScripting(document);
scripting.runScript("ScriptName");
```

6.196.1 Метод Scripting.runScript

Запускает макрокоманду с указанным именем. В случае невозможности запуска макрокоманды вызывает исключение `ScriptExecutionError`.

Пример

```
Scripting scripting = DocumentAPI.createScripting(document);
scripting.runScript("ScriptName");
```

6.197 Перечисление ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 120. Используется в качестве поля `scriptPosition` класса `TextProperties`.

Таблица 120 – Типы надстрочного и подстрочного форматирования

Значение	Описание
`ScriptPosition.SuperScript`	Надстрочный знак (верхний индекс)
`ScriptPosition.SubScript`	Подстрочный знак (нижний индекс)
`ScriptPosition.NormalScript`	Без указания индекса

Пример

```
TextProperties textProperties = new TextProperties();
textProperties.scriptPosition = ScriptPosition.NormalScript;
document.getRange().setTextProperties(textProperties);
```

6.198 Класс Scripts

Класс `Scripts` предоставляет доступ к списку макрокоманд документа. Коллекцию макрокоманд `Scripts` можно получить из документа посредством вызова метода `Document.getScripts()`.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null)
{
    ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
    foreach (var script in scriptsEnumerator)
    {
        Console.WriteLine(script.getName());
    }
}
```

6.198.1 Метод Scripts.GetEnumerator

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null)
{
    ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
    foreach (var script in scriptsEnumerator)
```

```
    {
        Console.WriteLine(script.getName());
    }
}
```

6.198.2 Метод Scripts.getScript

Метод возвращает объект класса Script, описывающий макрокоманду. В качестве аргумента используется имя макрокоманды.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    Script script = scripts.getScript("ScriptName");
    Console.WriteLine(script.getName());
}
```

6.198.3 Метод Scripts.removeScript

Метод удаляет макрокоманду из текущего документа. В качестве аргумента используется имя макрокоманды.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    String scriptName = "Enumerate scripts for document";
    scripts.removeScript(scriptName);
    Script script = scripts.getScript(scriptName);
    if (script == null) {
        Console.WriteLine("Script was removed");
    }
}
```

6.198.4 Метод Scripts.setScript

Метод добавляет макрокоманду в текущий документ. Если макрокоманда с таким именем уже существует, будет обновлено ее содержимое.

Пример

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    String scriptName = "Enumerate scripts for document";
    String scriptCode = "local scripts = document:getScripts()\nfor script in scripts:enumerate() do\nprint(script:getName())\nend";
    scripts.setScript(scriptName, scriptCode);
    Script script = scripts.getScript(scriptName);
}
```

6.199 Класс Search

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

6.199.1 Метод Search.findText

Метод выполняет поиск строки с учетом и без учета регистра:

- во всем документе;
- в выбранном диапазоне;
- в выбранном диапазоне ячеек;
- в таблице.

Результат возвращается в виде диапазона `Range`, содержащего искомый фрагмент.

Если строка не обнаружена, возвращается пустой диапазон.

Библиотеки Document API по умолчанию работают с документами используя английскую локализацию. Поэтому, при взаимодействии с документом через API, значения могут отличаться от отображаемых в интерфейсе редактора. Раздел Локализация документов содержит информацию о том, как работать с локализованными значениями.

Возможно использование следующих вариантов метода:

```
RangesEnumerator findText(string text, CaseSensitive caseSensitive)
RangesEnumerator findText(string text)
RangesEnumerator findText(string text, Range range, CaseSensitive caseSensitive)
RangesEnumerator findText(string text, Range range)
RangesEnumerator findText(string text, CellRange cellRange, CaseSensitive
caseSensitive)
RangesEnumerator findText(string text, CellRange cellRange)
RangesEnumerator findText(string text, Table table, CaseSensitive caseSensitive)
RangesEnumerator findText(string text, Table table)
```

Параметры

- `text` – текст для поиска;
- `caseSensitive` – регистр поиска , тип `CaseSensitive`;
- `range` – диапазон поиска, тип `Range`;
- `cellRange` – диапазон ячеек поиска, тип `CellRange`;
- `table` – таблица для поиска, тип `Table`.

Пример

```
// Поиск по всему документу, отображение результатов поиска
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult = search.findText("API", CaseSensitive.Yes);
while (searchResult.MoveNext())
{
    var range = searchResult.Current;
    Console.WriteLine(range.extractText());
}
```

Дополнительные примеры использования метода `Search.findText` приведены в разделе Поиск в документе.

6.200 Класс Section

Класс `Section` представляет собой раздел в документе.

6.200.1 Метод Section.getFooters

Метод возвращает коллекцию `HeadersFooters` нижних колонтитулов данного

раздела.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    HeadersFooters footers = section.getFooters();
    Console.WriteLine(footers);
}
```

6.200.2 Метод Section.getHeaders

Метод возвращает коллекцию HeadersFooters верхних колонтитулов данного раздела.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    HeadersFooters headers = section.getHeaders();
    Console.WriteLine(headers);
}
```

6.200.3 Метод Section.getPageOrientation

Метод возвращает ориентацию страниц раздела типа PageOrientation.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getPageOrientation());
}
```

6.200.4 Метод Section.getPageProperties

Метод возвращает параметры страниц раздела PageProperties.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    PageProperties pageProperties = section.getPageProperties();
    Console.WriteLine(pageProperties.height);
}
```

6.200.5 Метод Section.getRange

Метод возвращает диапазон Range в документе, соответствующий данному разделу.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    Range range = section.getRange();
    Console.WriteLine(range.extractText());
}
```

6.200.6 Метод Section.setPageOrientation

Метод задает ориентацию страниц раздела типа PageOrientation.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    section.setPageOrientation(PageOrientation.Portrait);
    Console.WriteLine(section.getPageOrientation());
}
```

6.200.7 Метод Section.setPageProperties

Метод устанавливает параметры PageProperties страниц, находящихся в разделе.

Пример

```
Block block = document.getBlocks().getBlock(0);
if (block != null)
{
    Section section = block.getSection();
    PageProperties pageProperties = section.getPageProperties();
    pageProperties.height = 100;
    pageProperties.width = 200;
    section.setPageProperties(pageProperties);
}
```

6.201 Перечисление SectionBreakType

Перечисление SectionBreakType содержит варианты разрыва страниц, используется в Position.insertSectionBreak().

Таблица 121 – Варианты разрыва страниц

Значение	Описание
SectionBreakType.NextPage	Следующий раздел начинается с новой страницы
SectionBreakType.Continuous	Следующий раздел продолжается на текущей странице без вставки разрыва страницы
SectionBreakType.EvenPage	Следующий раздел начинается на ближайшей четной странице
SectionBreakType.OddPage	Следующий раздел начинается на ближайшей нечетной странице

6.202 Класс Sections

Класс Sections используется для доступа к коллекции секций документа. Описание секции см. в разделе Section.

6.202.1 Метод Sections.getEnumerator

Метод позволяет перечислить коллекцию секций документа.

Пример

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getRange().extractText());
}
```

6.203 Класс Shape

Класс `Shape` представляет собой фигуру, содержит методы для установки и получения свойств `ShapeProperties`.

6.203.1 Метод Shape.getShapeProperties

Метод возвращает свойства фигуры `ShapeProperties`.

Пример

```
Shape shape = document.getBlocks().getShape(0);
if (shape != null) {
    ShapeProperties shapeProperties = shape.getShapeProperties();
    Console.WriteLine(shapeProperties.verticalAlignment);
}
```

6.203.2 Метод Shape.setShapeProperties

Метод устанавливает свойства фигуры `ShapeProperties`.

Пример

```
Shape shape = document.getBlocks().getShape(0);
if (shape != null) {
    ShapeProperties shapeProperties = shape.getShapeProperties();
    shapeProperties.verticalAlignment = VerticalAlignment.Center;
```

```
    shape.setShapeProperties(shapeProperties);
}
```

6.204 Класс ShapeProperties

Класс `ShapeProperties` описывает свойства фигуры. Используется в методах `Shape.getShapeProperties()` и `Shape.setShapeProperties()`.

Таблица 122 – Описание полей класса `ShapeProperties`

Поле	Тип	Описание
`ShapeProperties.verticalAlignment`	`VerticalAlignment`	Вертикальное выравнивание
`ShapeProperties.borderProperties`	`LineProperties`	Свойства границ фигуры
`ShapeProperties.fill`	`Fill`	Свойства заполнения фигуры
`ShapeProperties.shapeTextLayout`	`ShapeTextLayout`	Свойства текста внутри фигуры

Пример

```
ShapeProperties shapeProperties = shape.getShapeProperties();
shapeProperties.verticalAlignment = ...
shapeProperties.borderProperties = ...
shapeProperties.fill = ...
shapeProperties.shapeTextLayout = ...
shape.setShapeProperties(shapeProperties);
```

6.204.1 Поле ShapeProperties.borderProperties

Поле предназначено для установки свойств границ фигуры `LineProperties`.

Пример

```
Shape shape = document.getBlocks().getShape(0);
if (shape != null) {
```

```
    ShapeProperties shapeProperties = shape.getShapeProperties();
    LineProperties borderProperties = new LineProperties();
    borderProperties.style = LineStyle.Solid;
    borderProperties.width = 1.5f;
    borderProperties.color = new Color(new ColorRGBA(55, 146, 179, 200));
    shapeProperties.borderProperties = borderProperties;
    shape.setShapeProperties(shapeProperties);
}
```

6.204.2 Поле ShapeProperties.fill

Поле предназначено для установки свойств заполнения фигуры `Fill`.

Пример

```
ShapeProperties shapeProperties = shape.getShapeProperties();
.....................
shapeProperties.fill = new Fill();
.....................
shapeProperties.fill = new Fill(new Color(new ColorRGBA(55, 146, 179, 200)));
.....................
shapeProperties.fill = new Fill("https://fillpattern.url");
.....................
shape.setShapeProperties(shapeProperties);
```

6.204.3 Поле ShapeProperties.shapeTextLayout

Поле предназначено для установки свойств текста внутри фигуры, тип - `ShapeTextLayout`.

Пример

```
Shape shape = document.getBlocks().getShape(0);
if (shape != null) {
    ShapeProperties shapeProperties = shape.getShapeProperties();
    shapeProperties.shapeTextLayout = ShapeTextLayout.FitTextToShape;
    shape.setShapeProperties(shapeProperties);
}
```

6.204.4 Поле ShapeProperties.verticalAlignment

Поле предназначено для установки типа вертикального выравнивания VerticalAlignment.

Пример

```
Shape shape = document.getBlocks().getShape(0);
if (shape != null) {
    ShapeProperties shapeProperties = shape.getShapeProperties();
    shapeProperties.verticalAlignment = VerticalAlignment.Center;
    shape.setShapeProperties(shapeProperties);
}
```

6.205 Перечисление ShapeTextLayout

Перечисление ShapeTextLayout позволяет задать свойства текста, находящегося внутри фигуры. Используется в качестве поля shapeTextLayout класса ShapeProperties.

Таблица 123 – Свойства текста внутри фигуры

Значение	Описание
ShapeTextLayout.DoNotAutoFit	Размещение текста в фигуре по умолчанию
ShapeTextLayout.FitShapeExtentToText	Расширение фигуры под текст
ShapeTextLayout.FitTextToShape	Заполнение фигуры текстом

6.206 Класс SizeU

Класс SizeU описывает размер объекта в двухмерном пространстве.

Таблица 124 – Описание полей классе SizeU

Поле	Тип	Описание
SizeU.width	float	Ширина объекта
SizeU.height	float	Высота объекта

Примеры

```
SizeU size = new SizeU(50, 50);
```

```
SizeU size = new SizeU();
size.width = 50;
size.height = 50;
```

6.207 Класс SortingConditions

Представляет собой коллекцию условий для сортировки строк. Класс `SortingConditions` используется в методе `CellRange.sort()`.

Конструктор по умолчанию:

```
SortingConditions()
```

Конструктор копирования:

```
SortingConditions(SortingConditions other)
```

Порядок условий в коллекции определяет порядок применения сортировки. Например, можно задать дополнительное условие для другого столбца, чтобы отсортировать строки с одинаковыми значениями в первом столбце.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange range = sheet.getCellRange("A1:C11");

SortingConditions conditions = new SortingConditions();
conditions.add(0, SortingDirection.Descending);
conditions.add(1, SortingDirection.Ascending);

range.sort(conditions);
```

6.207.1 Метод SortingConditions.add

Метод добавляет заданное условие сортировки в коллекцию.

Вызов

```
public void add(uint index, SortingDirection sortingDirection)
```

Параметры

– `index`: индекс столбца диапазона для применения сортировки, тип `uint`.

– `sortingDirection`: порядок сортировки, тип `SortingDirection`.

Пример

```
SortingConditions conditions = new SortingConditions();
conditions.add(0, SortingDirection.Descending);
conditions.add(1, SortingDirection.Ascending);
```

6.207.2 Метод SortingConditions.clear

Метод очищает коллекцию условий.

Вызов

```
public void clear()
```

6.208 Перечисление SortingDirection

В таблице 125 представлены варианты порядка сортировки строк. Используется в методе SortingConditions.add().

Таблица 125 – Варианты порядка сортировки строк

Значение	Описание
SortingDirection.Ascending	Сортировка по возрастанию
SortingDirection.Descending	Сортировка по убыванию

Пример

```
SortingConditions conditions = new SortingConditions();
conditions.add(0, SortingDirection.Descending);
conditions.add(1, SortingDirection.Ascending);
```

6.209 Класс StyledTableRange

Класс StyledTableRange представляет собой умную таблицу в табличном документе. Используется в методе Table.getStyledTableRange().

Пример

```
Table sheet = document.getBlocks().getTable(0);
StyledTableRange smartTable = sheet.getStyledTableRange("SmartTable1");
CellRange tableCells = smartTable.getCellRange();
FiltersRange tableFiltersRange = smartTable.getFiltersRange();
```

6.209.1 Метод StyledTableRange.getCellRange

Метод возвращает диапазон ячеек, который принадлежит умной таблице. Данный диапазон также включает строки заголовков и итогов.

Вызов

```
public CellRange getCellRange()
```

Возвращает

– диапазон ячеек, тип CellRange.

Пример

```
Table sheet = document.getBlocks().getTable(0);
StyledTableRange smartTable = sheet.getStyledTableRange("SmartTable1");
CellRange tableCells = smartTable.getCellRange();
Console.WriteLine(tableCells.getTableRange().toString());
```

6.209.2 Метод StyledTableRange.getFiltersRange

Метод возвращает диапазон фильтрации умной таблицы.

Вызов

```
public FiltersRange getFiltersRange()
```

Возвращает

– диапазон фильтрации, тип FiltersRange.

Пример

```
Table sheet = document.getBlocks().getTable(0);
StyledTableRange smartTable = sheet.getStyledTableRange("SmartTable1");
FiltersRange tableFiltersRange = smartTable.getFiltersRange();

TableFilters filters = new TableFilters();
ConditionalTableFilter filter1 = new ConditionalTableFilter();
filter1.begins("T");
ConditionalTableFilter filter2 = new ConditionalTableFilter();
filter2.begins("M");
filters.setFilter(0, filter1);
filters.setFilter(1, filter2);

tableFiltersRange.setFilters(filters);
```

6.210 Перечисление StylesPastingPolicy

Перечисление `StylesPastingPolicy` содержит режимы копирования форматирования ячеек. Используется в качестве поля `stylesPastingPolicy` класса `CellRangePastingSettings`.

Таблица 126 – Режимы копирования форматирования

Значение	Описание
`StylesPastingPolicy.AllContent`	Копирует стили и числовой формат
`StylesPastingPolicy.FormattedText`	Копирует числовой формат
`StylesPastingPolicy.PlainText`	Копирует только текст без форматирования

6.211 Класс Table

Класс `Table` предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 2).

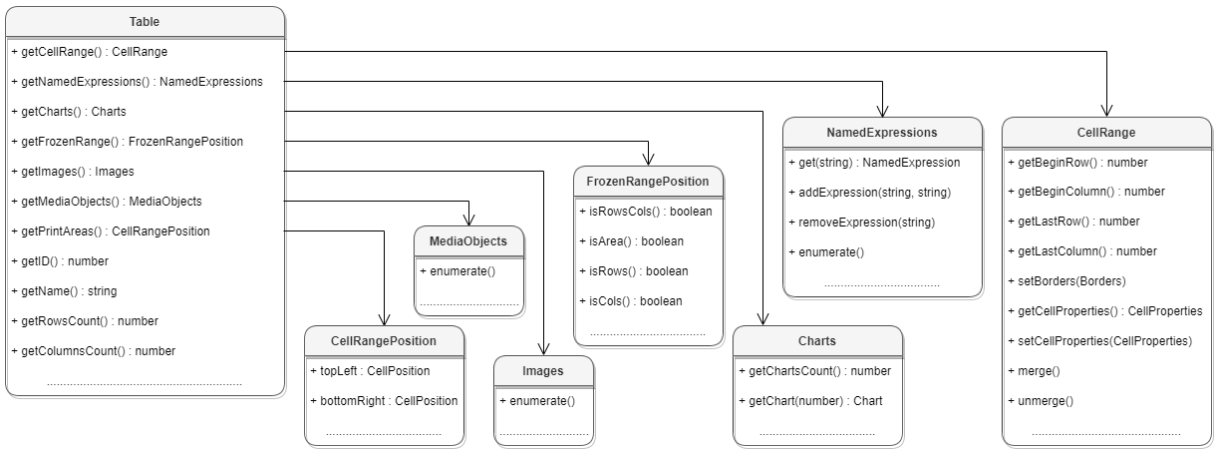


Рисунок 60 – Структура полей класса `Table`

Класс `Table` наследуется от класса `Block` и обладает его базовыми методами `Block.getRange`, `Block.getSection`, `Block.remove`.

6.211.1 Метод Table.clearColumnGroups

Метод предназначен для очистки группированных столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов

```
clearColumnGroups(columnIndex, columnsCount)
```

Параметры

- `columnIndex` – индекс столбца, начиная с которого будет начата очистка групп;
- `columnsCount` – количество столбцов для очистки групп.

6.211.2 Метод Table.clearRowGroups

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

Вызов

```
clearRowGroups(rowIndex, rowCount)
```

Параметры

- `rowIndex` – индекс строки, начиная с которой будет начата очистка групп;
- `rowCount` – количество строк для очистки групп.

6.211.3 Метод Table.createFiltersRange

Метод `Table.createFiltersRange` задает диапазон, который используется как диапазон фильтрации.

```
public FiltersRange createFiltersRange(CellRangePosition range)
```

В качестве параметра используется диапазон ячеек типа `CellRangePosition`. Метод возвращает `FiltersRange`.

Разрешен только один диапазон фильтрации на таблицу. Это означает, что данный метод удаляет ранее определенный диапазон фильтрации. При этом есть исключение: если новый диапазон начинается с той же позиции, предыдущие фильтры будут сохранены.

Диапазон фильтрации должен включать дополнительную строку, которая используется как заголовок таблицы. Эта строка никогда не фильтруется.

Метод может быть использован только в табличных документах.

Метод может формировать следующие исключения:

- `NotImplementedError`: метод вызывается для неподдерживаемого документа
- `IncorrectArgumentError`: диапазон слишком мал или имеет недопустимые элементы

– DocumentModificationError: диапазон пересекает объединенные ячейки или содержит сводную таблицу

– OutOfRangeError: диапазон находится за пределами таблицы

Пример

```
Table sheet = document.getBlocks().getTable("Лист1");
CellRangePosition cellRange = new CellRangePosition(1, 1, 8, 2);
FiltersRange filtersRange = sheet.createFiltersRange(cellRange);
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.211.4 Метод Table.duplicate

Метод используется для создания копии листа табличного документа. Может быть использован только в табличном документе.

При использовании без параметров создает копию после текущего листа. Для того, чтобы избежать повторяющихся имен, к имени нового листа добавляется индекс.

Задайте параметры, чтобы скопировать лист в другой табличный документ. В этом случае копируются все элементы листа, кроме следующих: сводные таблицы, диаграммы, фигуры, изображения, проверки данных, условное форматирование, именованные диапазоны, данные о сортировке и фильтрации.

Вызов

```
public void duplicate()
```

```
public void duplicate(Document other, uint index)
```

Параметры

– `other`: документ, в который копируется лист, тип Document.

– `index`: позиция вставки листа, тип `uint`.

Пример копирования в пределах документа

```
Table table = document.getBlocks().getTable(0);
table.duplicate();
```

Пример копирования в другой документ

```
var tableDocument = application.loadDocument("sheet.xods");
var otherDocument = application.loadDocument("otherSheet.xods");

Table sheet = tableDocument.getBlocks().getTable(0);
sheet.duplicate(otherDocument, 0);
```

```
otherDocument.saveAs("otherSheet.xods");
```

6.211.5 Метод Table.find

Метод выполняет поиск ячеек, соответствующих заданному запросу.

Библиотеки Document API по умолчанию работают с документами используя английскую локализацию. Поэтому, при взаимодействии с документом через API, значения могут отличаться от отображаемых в интерфейсе редактора. Раздел Локализация документов содержит информацию о том, как работать с локализованными значениями.

Вызов

```
public CellsEnumerator find(string string_, TableSearchSettings settings)
```

Параметры

– `string_`: поисковый запрос, тип `string`.

– `settings`: (необязательный) параметры поиска, тип `TableSearchSettings`.

Возвращает

– список ячеек, соответствующих поисковому запросу, тип `CellsEnumerator`.

Пример

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

TableSearchSettings searchProps = new TableSearchSettings();
searchProps.caseSensitive = CaseSensitive.No;
searchProps.matchBehaviour = TableSearchSettings.MatchBehaviour.Glob;
searchProps.searchIn = TableSearchSettings.SearchProperty.Value;
searchProps.wholeWords = true;

CellsEnumerator results = sheet.find("*eye", searchProps);
foreach (Cell cell in results) {
    Console.WriteLine(cell.getFormattedValue()); // Steeleye Stout
}
```

6.211.6 Метод Table.freeze

Метод `freeze` закрепляет заданную область `FrozenRangePosition` таблицы.

Пример

```
var frozenRangePosition = FrozenRangePosition.createFrozenCols(0, 2);
table.freeze(frozenRangePosition);
```

6.211.7 Метод Table.getCell

Метод позволяет получить доступ к отдельной ячейке таблицы. В качестве аргумента может выступать текстовое представление адреса ячейки, либо экземпляр класса `CellPosition`.

Примеры

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
Console.WriteLine(cell.getFormattedValue());
```

```
Table firstSheet = document.getBlocks().getTable(0);
CellPosition cellPosition = new CellPosition(2, 1);
Cell cell = firstSheet.getCell(cellPosition);
Console.WriteLine(cell.getFormattedValue());
```

6.211.8 Метод Table.getCellRange

Метод позволяет получить доступ к диапазону ячеек класса `CellRange`. В качестве аргумента может использоваться строка, описывающая диапазон (`"B3:C4"`), либо объект типа `CellRangePosition`.

Примеры

```
Table firstSheet = document.getBlocks().getTable("Лист1");
CellRange cellRange = firstSheet.getCellRange("B3:C4");
CellsEnumerator cellRangesEnumerator = cellRange.GetEnumerator();
foreach (var cell in cellRangesEnumerator)
{
    Console.WriteLine(cell.getFormattedValue());
}
```

```
Table firstSheet = document.getBlocks().getTable("Лист1");
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
CellRange cellRange = firstSheet.getCellRange(cellRangePosition);
```

6.211.9 Метод Table.getCharts

Для получения списка диаграмм (Charts) таблицы используется метод `Table.getCharts`.

Пример

```
TablesEnumerator tablesEnumerator = document.getBlocks().getTablesEnumerator();
foreach (var table in tablesEnumerator)
{
    Charts charts = table.getCharts();
    Console.WriteLine(charts.getChartsCount());
}
```

6.211.10 Метод Table.getColumnsCount

Метод позволяет получить количество столбцов таблицы.

Пример

```
Table table = document.getBlocks().getTable(0);
Console.WriteLine(table.getColumnsCount());
```

6.211.11 Метод Table.getColumnWidth

Метод возвращает ширину столбца таблицы в пунктах (1/72 дюйма).

Вызов

```
float getColumnWidth(uint columnIndex)
```

Параметры

- `columnIndex` – индекс столбца в таблице, для которого возвращается значение ширины. Индексация столбцов начинается с нуля.

Возвращает

- ширина столбца в пунктах (1/72 дюйма).

Пример

```
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");
var width = table.getColumnWidth(1);
```

Задать ширину столбца таблицы позволяет метод Table.setColumnWidth.

6.211.12 Метод Table.getConditionalFormatRules

Метод позволяет получить коллекцию правил условного форматирования для текущего листа.

Вызов

```
public ConditionalFormatTableRules getConditionalFormatRules()
```

Возвращает

– коллекция правил условного форматирования для листа, тип ConditionalFormatTableRules.

Пример

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

var topBottomStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(255, 0, 0, 150)));
topBottomStyle.cellProperties = cellProperties;

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

var topBottomOperator =
DocumentAPI.createTopBottomConditionalFormatOperator(ConditionalFormatTopBottomCo
ndition.Top, 20, true);

var topBottomRule = new ConditionalFormatRule(topBottomOperator, topBottomStyle,
cellRangePosition, false);
rules.addRule(topBottomRule);
```

6.211.13 Метод Table.getFiltersRange

Метод `Table.getFiltersRange` возвращает текущий диапазон фильтрации, принадлежащий таблице. Рабочий лист табличного документа может содержать только один диапазон фильтрации. Чтобы получить диапазон фильтрации умной таблицы, используйте метод `StyledTableRange.getFiltersRange`.

```
FiltersRange getFiltersRange();
```

Метод возвращает `FiltersRange`, если диапазон фильтрации существует.

Метод может быть использован только в табличных документах.

При использовании данного метода может произойти исключение `InvalidObjectError` в случае, если диапазон недействителен.

Пример

```
FiltersRange filtersRange = sheet.getFiltersRange();
CellRange cellRange = filtersRange.getCellRange();
Console.WriteLine(cellRange.getBeginRow());
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.211.14 Метод Table.getFrozenRange

Существует возможность закрепления диапазона строк и столбцов. Такие диапазоны всегда остаются видимыми на экране в случае, когда пользователь осуществляет навигацию по таблице.

Метод `getFrozenRange` возвращает закрепленный диапазон `FrozenRangePosition`.

Пример

```
var frozenRangePosition = FrozenRangePosition.createFrozenCols(0, 2);
table.freeze(frozenRangePosition);
Console.WriteLine(table.getFrozenRange().isCols());
```

6.211.15 Метод Table.getImages

Метод предназначен для получения списка изображений в таблице.

```
Images images = table.getImages();
ImagesEnumerator imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
```

```
    ......................
}
```

6.211.16 Метод Table.getLabelColor

Метод возвращает цвет ярлыка листа. Ярлыки используются для визуального разделения листов табличного документа.

Вызов

```
public Color getLabelColor()
```

Возвращает

– цвет ярлыка листа, тип `Color`.

– `null`, если цвет не задан.

Пример

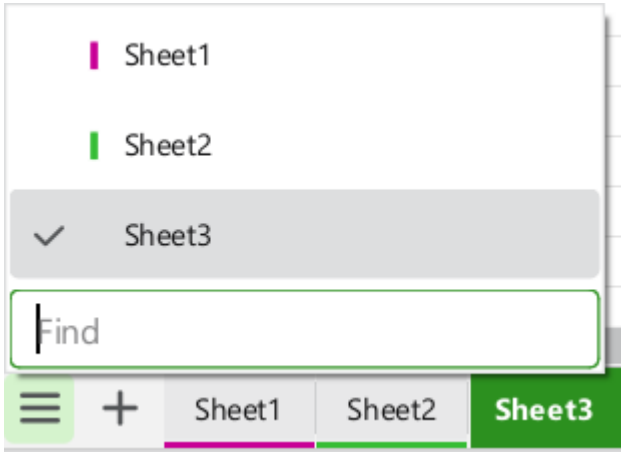


Рисунок 61 – Ярлыки листов табличного документа

```
Table sheet = document.getBlocks().getTable(1);
ColorRGBA rgba = sheet.getLabelColor().getRGBAColor();
Console.WriteLine("r:" + rgba.r + ", g:" + rgba.g + ", b:" + rgba.b + ", a:" +
rgba.a);
```

6.211.17 Метод Table.getLastNonEmptyCellInColumn

Возвращает индекс последней заполненной ячейки в столбце. Последней заполненной считается ячейка, после которой все ячейки пустые.

Вызов

```
public int? getLastNonEmptyCellInColumn(uint columnIndex)
```

Параметры

– `columnIndex`: индекс столбца.

Возвращает

– индекс строки, которая содержит последнюю заполненную ячейку.

– `null`, если все ячейки в столбце пустые.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Console.WriteLine(sheet.getLastNonEmptyCellInColumn(0)); // 10
Console.WriteLine(sheet.getLastNonEmptyCellInRow(0)); // 2
```

6.211.18 Метод Table.getLastNonEmptyCellInRow

Возвращает индекс последней заполненной ячейки в строке. Последней заполненной считается ячейка, после которой все ячейки пустые.

Вызов

```
public int? getLastNonEmptyCellInRow(uint rowIndex)
```

Параметры

– `rowIndex`: индекс строки.

Возвращает

– индекс столбца, который содержит последнюю заполненную ячейку.

– `null`, если все ячейки в строке пустые.

Пример

```
Table sheet = document.getBlocks().getTable(0);
Console.WriteLine(sheet.getLastNonEmptyCellInColumn(0)); // 10
Console.WriteLine(sheet.getLastNonEmptyCellInRow(0)); // 2
```

6.211.19 Метод Table.getMediaObjects

Для получения списка медиаобъектов (`MediaObjects`) таблицы используется метод `Table.getMediaObjects`.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
MediaObjects mediaObjects = firstSheet.getMediaObjects();
MediaObjectsEnumerator mediaObjectsEnumerator = mediaObjects.GetEnumerator();
foreach (var mediaObject in mediaObjectsEnumerator)
{
    ..........................
}
```

6.211.20 Метод Table.getName

Метод позволяет получить имя листа табличного документа.

Пример

```
Table table = document.getBlocks().getTable(0);
Console.WriteLine(table.getName());
```

6.211.21 Метод Table.getNamedExpressions

Для получения списка именованных диапазонов `NamedExpressions` используется метод `Table.getNamedExpressions()`.

Пример

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpression namedExpression = namedExpressions.get("Продажи");
if (namedExpression != null) {
    Console.WriteLine(namedExpression.getName());
}
```

6.211.22 Метод Table.getPrintAreas

Метод `Table.getPrintAreas` возвращает текущие области печати - вектор элементов `CellRangePosition`.

Пример

```
tbl = document.getBlocks().getTable(0);
tbl.setPrintArea(new CellRangePosition(0, 0, 5, 5));
printAreas = tbl.getPrintAreas();
```

6.211.23 Метод Table.getPrintTitles

Метод возвращает сквозной заголовок для текущего листа.

Вызов

```
public PrintTitles getPrintTitles()
```

Возвращает

– сквозной заголовок, тип `PrintTitles`.

– `null`, если сквозной заголовок не установлен.

Пример

```
Table sheet = document.getBlocks().getTable(0);
PrintTitles printTitles = sheet.getPrintTitles();
Console.WriteLine(printTitles.isCols());
Console.WriteLine(printTitles.isRows());
Console.WriteLine(printTitles.isRowsCols());
```

6.211.24 Метод Table.getProtectionProperties

Метод возвращает параметры защиты от изменений листа табличного документа (см. Защита листа табличного документа).

Вызов

```
public TableProtectionProperties getProtectionProperties()
```

Возвращает

– `TableProtectionProperties`: свойства защиты листа документа (`null`, если защита листа не установлена).

6.211.25 Метод Table.getRowHeight

Метод возвращает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов

```
float getRowHeight(uint rowIndex)
```

Параметры

– `rowIndex` – индекс строки в таблице, для которой возвращается значение высоты. Индексация строк начинается с нуля.

Возвращает

– высота строки в пунктах (1/72 дюйма).

Пример

```
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");
var height = table.getRowHeight(1);
```

Задать высоту строки таблицы позволяет метод `Table.setRowHeight`.

6.211.26 Метод Table.getRowsCount

Метод позволяет получить количество строк таблицы.

Пример

```
Table table = document.getBlocks().getTable(0);
Console.WriteLine(table.getRowsCount());
```

6.211.27 Метод Table.getShowZeroValue

Для проверки режима отображения нулевых значений ячеек используется метод `getShowZeroValue`.

Пример

```
Table table = document.getBlocks().getTable(0);
table.setShowZeroValue(false);
Console.WriteLine(table.getShowZeroValue());
```

6.211.28 Метод Table.getStyledTableRange

Метод позволяет получить умную таблицу из листа табличного документа по ее названию.

Вызов

```
public StyledTableRange getStyledTableRange(string name)
```

Параметры

– `name`: название умной таблицы, тип `string`.

Возвращает

– умная таблица, тип `StyledTableRange`.

– `null`, если если умной таблицы с таким названием не существует.

Пример

```
Table sheet = document.getBlocks().getTable(0);
StyledTableRange smartTable = sheet.getStyledTableRange("SmartTable1");
CellRange tableCells = smartTable.getCellRange();
FiltersRange tableFiltersRange = smartTable.getFiltersRange();
```

6.211.29 Метод Table.getUsedRange

Метод возвращает диапазон ячеек, используемый в текущем листе табличного документа. К используемым относятся ячейки, которые содержат:

- данные;
- заметки;
- форматирование;
- объединенные ячейки;
- фильтры;
- сводные таблицы;
- умные таблицы.

Вызов

```
public CellRange getUsedRange()
```

Возвращает

– используемый диапазон ячеек, тип `CellRange`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
CellRange usedRange = sheet.getUsedRange();
```

6.211.30 Метод Table.groupColumns

Метод предназначен для группировки столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов

```
groupColumns(columnIndex, columnsCount)
```

Параметры

- `columnIndex` – индекс столбца, начиная с которого будет начата группировка столбцов;
- `columnsCount` – количество столбцов для группировки.

6.211.31 Метод Table.groupRows

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

Вызов

```
groupRows(rowIndex, rowsCount)
```

Параметры

- `rowIndex` – индекс строки, начиная с которого будет начата группировка строк;
- `rowsCount` – количество строк для группировки.

6.211.32 Метод Table.insertColumnAfter

Метод предназначен для вставки нового столбца после указанной позиции в таблице.

Вызов

```
insertColumnAfter(columnIndex, copyColumnStyle, columnsCount)
```

Параметры

- `columnIndex` – индекс столбца в таблице, после которого производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.

– `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример

```
// Создать в документе новую таблицу 2x2
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");
// Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
table.insertColumnAfter(0, false, 2);
```

6.211.33 Метод Table.insertColumnBefore

Метод предназначен для вставки нового столбца до указанной позиции в таблице.

Вызов

```
insertColumnBefore(columnIndex, copyColumnStyle, columnsCount)
```

Параметры

– `columnIndex` – индекс столбца в таблице, перед которым производится вставка. Индексация столбцов начинается с нуля.

– `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.

– `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример

```
// Создать в документе новую таблицу 2x2
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");
// Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
table.insertColumnBefore(1, false, 2);
```

6.211.34 Метод Table.insertImage

Метод вставляет изображение из файла в заданные координаты на листе табличного документа.

Вызов

```
public Image insertImage(string url, RectU rect)
```

Параметры

– `url`: путь к файлу, тип `string`.

– `rect`: координаты для вставки изображения, тип `RectU`.

Возвращает

– вставленное изображение, тип `Image`.

Пример

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

6.211.35 Метод Table.insertRowAfter

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

Вызов

```
insertRowAfter(rowIndex, copyRowStyle, rowsCount)
```

Параметры

– `rowIndex` – индекс строки в таблице, после которой производится вставка. Индексация строк начинается с нуля.

– `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.

– `rowsCount` – количество вставляемых строк. Значение по умолчанию 1.

Пример

```
// Создать в документе новую таблицу 2x2
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");

// Добавление двух строк в середину таблицы, без наследования настроек форматирования
table.insertRowAfter(0, false, 2);
```

6.211.36 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
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");

// Добавление двух строк в середину таблицы, без наследования настроек форматирования
table.insertRowBefore(1, false, 2);
```

6.211.37 Метод Table.isColumnVisible

Метод `Table.isColumnVisible` позволяет определять видимость столбца по заданному индексу. Индексация столбцов начинается с нуля. Метод возвращает `true` если столбец отображается.

Для задания видимости столбцов таблицы применяется метод `Table.setColumnsVisible`.

Вызов

```
isColumnVisible(columnIndex)
```

Параметр

`columnIndex` – индекс столбца.

Пример

```
Table table = document.getBlocks().getTable(0);
Console.WriteLine(table.isColumnVisible(3));
```

Дополнительный пример использования метода `Table.isColumnVisible` приведен в разделе Управление видимостью строк / колонок.

6.211.38 Метод Table.isProtected

Метод возвращает состояние защиты от изменений листа табличного документа (см. Защита листа табличного документа).

Вызов

```
public bool isProtected()
```

6.211.39 Метод Table.isRowVisible

Метод `Table.isRowVisible` позволяет определять видимость строки по заданному индексу. Индексация строк начинается с нуля. Метод возвращает `true` если строка отображается.

Для задания видимости строк таблицы применяется метод `Table.setRowsVisible`.

Вызов

```
isRowVisible(rowIndex)
```

Параметр

`rowIndex` – индекс строки.

Пример

```
Table table = document.getBlocks().getTable(0);
Console.WriteLine(table.isRowVisible(3));
```

Дополнительный пример использования метода `Table.isRowVisible` приведен в разделе Управление видимостью строк / колонок.

6.211.40 Метод Table.isVisible

Метод возвращает значение true, если лист таблицы в табличном документе отображается в редакторе таблиц.

Пример

```
Table table = document.getBlocks().getTable(0);
if (!table.isVisible()) {
    table.setVisible(true);
}
```

6.211.41 Метод Table.moveTo

Для перемещения листа таблицы по указанному индексу в табличном документе используется метод `moveTo`. Указанный индекс должен быть меньше или равен количеству листов в документе. Индексация листов начинается с нуля. Метод может быть использован только в табличном документе.

Пример

```
Table table = document.getBlocks().getTable(0);
table.moveTo(1);
```

6.211.42 Метод Table.remove

Для удаления таблицы в текстовом документе или листа в табличном документе используется метод `remove()`.

Пример

```
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    table.remove();
}
```

6.211.43 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, необязательный параметр. Значение по умолчанию 1.

6.211.44 Метод Table.removeProtection

Метод снимает защиту от изменений с листа табличного документа (см. Защита листа табличного документа).

Вызов

```
public void removeProtection(string password)
```

Параметры

– `password`: (необязательный) пароль для снятия защиты, тип `string`.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
firstSheet.removeProtection("password");
```

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `IncorrectPasswordError`.

6.211.45 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, необязательный параметр. Значение по умолчанию 1.

6.211.46 Метод Table.removeVisibleColumns

Метод удаляет видимые столбцы таблицы, находящиеся между заданными индексами (включительно).

Вызов

```
public void removeVisibleColumns(uint firstIndex, uint lastIndex)
```

Параметры

- `firstIndex`: индекс первого столбца. Индексация столбцов начинается с нуля.
- `lastIndex`: индекс последнего столбца.

6.211.47 Метод Table.removeVisibleRows

Метод удаляет видимые строки таблицы, находящиеся между заданными индексами (включительно).

Вызов

```
public void removeVisibleRows(uint firstIndex, uint lastIndex)
```

Параметры

- `firstIndex`: индекс первой строки. Индексация строк начинается с нуля.
- `lastIndex`: индекс последней строки.

6.211.48 Метод Table.setColumnsVisible

Метод `Table.setColumnsVisible` позволяет задавать видимость столбцов, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов

```
setColumnsVisible(first, columnsCount, visible)
```

Параметры

`first` – начальный индекс;

`columnsCount` – количество столбцов;

`visible` – видимость.

6.211.49 Метод Table.setColumnWidth

Метод устанавливает ширину столбца таблицы в пунктах (1/72 дюйма).

Вызов

```
void setColumnWidth(uint columnIndex, float width)
```

Параметры

- `columnIndex` – индекс столбца в таблице, для которого устанавливается значение ширины. Индексация столбцов начинается с нуля.
- `width` – ширина столбца в пунктах (1/72 дюйма).

Пример

```
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");

// Установить ширину столбца в 400 pt
table.setColumnWidth(1, 400);
```

Метод Table.getColumnWidth позволяет получить ширину столбца таблицы.

6.211.50 Метод Table.setLabelColor

Метод позволяет задать цвет ярлыка листа. Ярлыки используются для визуального разделения листов табличного документа.

Вызов

```
public void setLabelColor(Color labelColor)
```

Параметры

– `labelColor`: цвет ярлыка листа, тип Color. `null`, чтобы сбросить цвет ярлыка.

Пример

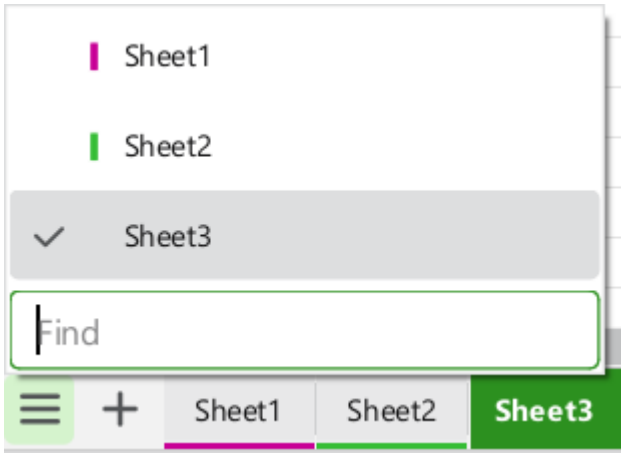


Рисунок 62 – Ярлыки листов табличного документа

```
Table sheet = document.getBlocks().getTable(0);
sheet.setLabelColor(new Color(new ColorRGBA(200, 0, 150, 255)));
```

6.211.51 Метод Table.setName

Метод задает имя таблицы. В случае с табличным документом это текстовое значение будет являться именем страницы. В текстовых документах имя таблицы нигде не отображается, но в дальнейшем его можно использовать для доступа к таблице по имени.

Устанавливаемое значение должно быть уникальным.

Пример использования

```
Table table = document.getBlocks().getTable("Лист1");
if (table != null)
{
    String newTableName = "Лист2";
    table.setName(newTableName);
    table = document.getBlocks().getTable(newTableName);
    if (table != null)
    {
        // Table was renamed
    }
}
```

Примеры использования метода также приведены в разделах Работа с таблицами текстового документа и Работа с листами табличного документа.

6.211.52 Метод Table.setPrintArea

Метод служит для установки и сброса области печати, тип аргумента CellRangePosition.

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
firstSheet.setPrintArea(CellRangePosition(0, 0, 5, 5));
```

6.211.53 Метод Table.setPrintAreas

Метод Table.setPrintAreas задает множественные области печати или экспорта CellRangePositions, где CellRangePositions - вектор из элементов CellRangePosition.

Пример

```
var ranges = new CellRangePositions();
ranges.Add(new CellRangePosition(0, 0, 5, 5));
ranges.Add(new CellRangePosition(1, 2, 5, 5));
table.setPrintAreas(ranges);

var printAreas = table.getPrintAreas();
```

```
Console.WriteLine(printAreas[0].toString());
Console.WriteLine(printAreas[1].toString());
```

6.211.54 Метод Table.setPrintTitles

Метод задает сквозной заголовок (диапазон ячеек, который будет отображаться на каждой странице экспортируемого документа) для текущего листа.

Вызов

```
public void setPrintTitles(PrintTitles range)
```

Параметры

– `range`: сквозной заголовок, тип `PrintTitles`.

Пример

```
Table sheet = document.getBlocks().getTable(0);
PrintTitles titles = new PrintTitles(4, 4, 7, 7);
sheet.setPrintTitles(titles);

document.exportAs("exportedDoc.pdf", ExportFormat.PDFA1);
```

6.211.55 Метод Table.setProtection

Метод устанавливает защиту от изменений на лист табличного документа (см. Защита листа табличного документа).

Вызов

```
public void setProtection(TableProtectionProperties tableProtectionProperties, string password)
```

Параметры

– `tableProtectionProperties`: параметры защиты листа, тип `TableProtectionProperties`;

– `password`: (необязательный) пароль для установки защиты, тип `string`.

Пример

```
TableProtectionProperties tableProps = new TableProtectionProperties();
tableProps.deleteColumns = false;
tableProps.deleteRows = false;
tableProps.filterData = true;
```

```
tableProps.formatCells = true;
tableProps.formatColumns = true;
tableProps.formatRows = true;
tableProps.insertAndEditObjects = false;
tableProps.insertAndEditPivotTables = false;
tableProps.insertColumns = false;
tableProps.insertLinks = true;
tableProps.insertRows = false;
tableProps.selectProtectedCells = true;
tableProps.sortData = true;

Table firstSheet = document.getBlocks().getTable(0);
firstSheet.setProtection(tableProps, "password");
```

Метод `setProtectionProperties()` объектов `Cell` и `CellRange` позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой защиты).

Если метод `setProtection()` применяется к уже защищенному листу, возникает исключение `SpreadsheetProtectionError`.

6.211.56 Метод Table.setRowHeight

Метод устанавливает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов

```
void setRowHeight(uint rowIndex, float height, RowHeightRule? heightRule)
```

Параметры

- `rowIndex` – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- `height` – высота строки в пунктах (1/72 дюйма).
- `heightRule` – точность значения (`RowHeightRule.Exact` – точно, `RowHeightRule.AtLeast` – не меньше).

Пример

```
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");

// Установить высоту строки в 100 pt
table.setRowHeight(1, 100, RowHeightRule.Exact);
```

Метод Table.getRowHeight позволяет получить высоту строки таблицы.

6.211.57 Метод Table.setRowsVisible

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

Вызов

```
setRowsVisible(first, rowsCount, visible)
```

Параметры

first – начальный индекс;

columnsCount – количество строк;

visible – видимость.

6.211.58 Метод Table.setShowZeroValue

Для упрощения чтения таблицы нулевые значения ячеек могут быть скрыты. Для управления скрытием/показом ячеек используется метод setShowZeroValue.

Пример

```
Table table = document.getBlocks().getTable(0);
table.setShowZeroValue(true);
```

6.211.59 Метод Table.setVisible

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

Вызов

```
setVisible(bool visible)
```

Параметр

visible – параметр, задающий видимость листа. Если значение параметра visible равно true, то лист таблицы отображается в редакторе таблиц.

Пример

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

6.211.60 Метод Table.ungroupColumns

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

Вызов

```
ungroupColumns(columnIndex, columnsCount)
```

Параметры

- `columnIndex` – индекс столбца, начиная с которого будет начата разгруппировка столбцов;
- `columnsCount` – количество столбцов для разгруппировки.

6.211.61 Метод Table.ungroupRows

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

Вызов

```
ungroupRows(rowIndex, rowsCount)
```

Параметры

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

6.212 Класс TableFilters

`TableFilters` - это объект, который хранит фильтры столбцов. Фильтры можно применять к диапазону фильтрации `FiltersRange`. При применении фильтров, соответствующие строки рабочего листа будут скрыты.

Конструктор по умолчанию:

```
TableFilters();
```

Конструктор копирования:

```
TableFilters(TableFilters other);
```

Пример использования приведен в разделе Работа с фильтрами.

6.212.1 Метод TableFilters.clear

Метод `TableFilters.clear` удаляет все фильтры столбцов, которые были сохранены ранее.

Пример

```
TableFilters tableFilters = new TableFilters();
tableFilters.setFilter(0, johnPaulFilter);
tableFilters.setFilter(1, songFilter);
.............................
tableFilters.clear();
```

6.212.2 Метод TableFilters.erase

Метод `TableFilters.erase` удаляет фильтр из заданного столбца. В качестве параметра используется индекс столбца.

Пример

```
TableFilters tableFilters = new TableFilters();
tableFilters.setFilter(0, johnPaulFilter);
tableFilters.setFilter(1, songFilter);
.............................
tableFilters.erase(1);
```

6.212.3 Метод TableFilters.getAsConditionalFilter

Метод возвращает фильтр по условию, установленный в заданном столбце.

Вызов

```
public ConditionalTableFilter getAsConditionalFilter(uint column)
```

Параметры

– `column`: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `uint`.

Возвращает

– фильтр по условию, тип `ConditionalTableFilter`.

– `null`, если фильтр отсутствует или он другого типа.

Пример

```
ConditionalTableFilter conditionalFilter;
if (filters.getFilterType(0) == ContainingTableFilter.Conditional)
    conditionalFilter = filters.getAsConditionalFilter(0);
```

6.212.4 Метод TableFilters.getAsValueFilter

Метод возвращает фильтр по значению, установленный в заданном столбце.

Вызов

```
public ValuesTableFilter getAsValueFilter(uint column)
```

Параметры

– `column`: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `uint`.

Возвращает

– фильтр по значению, тип `ValuesTableFilter`.

– `null`, если фильтр отсутствует или он другого типа.

Пример

```
ValuesTableFilter valuesFilter;
if (filters.getFilterType(0) == ContainingTableFilter.Values)
    valuesFilter = filters.getAsValueFilter(0);
```

6.212.5 Метод TableFilters.getFilterType

Метод возвращает тип фильтра, который установлен в заданном столбце.

Вызов

```
public ContainingTableFilter getFilterType(uint column)
```

Параметры

– `column`: индекс столбца, нумерация столбцов начинается с нуля относительно левого края диапазона фильтрации, тип `uint`.

Возвращает

– тип фильтра столбцов таблицы, тип `ContainingTableFilter`.

Пример

```
ValuesTableFilter valuesFilter;
if (filters.getFilterType(0) == ContainingTableFilter.Values)
    valuesFilter = filters.getAsValueFilter(0);
```

6.212.6 Метод TableFilters.setFilter

Метод `TableFilters.setFilter` устанавливает фильтр для конкретного столбца. Нумерация столбцов начинается с нуля, относительно левой позиции диапазона фильтрации.

```
void setFilter(int column, ValuesTableFilter filter);
void setFilter(int column, ConditionalTableFilter filter);
```

Параметры

- `column` – индекс столбца;
- `filter` – вариант фильтра: `ValuesTableFilter` или `ConditionalTableFilter`.

Пример

```
ValuesTableFilter johnPaulFilter = new ValuesTableFilter();
johnPaulFilter.add("John");
johnPaulFilter.add("Paul");

ConditionalTableFilter songFilter = new ConditionalTableFilter();
songFilter.setAndOperation(true);
songFilter.notEqual("");
songFilter.notBegins("TODO");

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

Более подробный пример приведен в разделе Работа с фильтрами.

6.213 Класс TableProtectionProperties

Класс `TableProtectionProperties` предназначен для настройки параметров защиты листа в табличном документе (аналог раздела «Разрешенные действия» в меню

«Управление защитой»). Данный класс используется в методах Table.setProtection() и Table.getProtectionProperties().

Таблица 127 – Описание полей класса TableProtectionProperties

Поле	Значение по умолчанию	Описание
TableProtectionProperties.deleteColumns	false	Разрешить удалять колонки
TableProtectionProperties.deleteRows	false	Разрешить удалять строки
TableProtectionProperties.filterData	false	Разрешить фильтровать данные
TableProtectionProperties.formatCells	false	Разрешить форматировать ячейки
TableProtectionProperties.formatColumns	false	Разрешить форматировать столбцы
TableProtectionProperties.formatRows	false	Разрешить форматировать строки
TableProtectionProperties.insertAndEditObjects	false	Разрешить вставлять и редактировать объекты: изображения, диаграммы, фигуры и текстовые поля
TableProtectionProperties.insertAndEditPivotTables	false	Разрешить вставлять и редактировать сводные таблицы
TableProtectionProperties.insertColumns	false	Разрешить вставлять столбцы
TableProtectionProperties.insertLinks	false	Разрешить вставлять и редактировать ссылки в ячейках
TableProtectionProperties.insertRows	false	Разрешить вставлять строки
TableProtectionProperties.selectProtectedCells	true	Разрешить выделение защищенных ячеек
TableProtectionProperties.sortData	false	Разрешить сортировать данные

Пример

```
TableProtectionProperties tableProps = new TableProtectionProperties();
tableProps.deleteColumns = false;
tableProps.deleteRows = false;
```

```
tableProps.filterData = true;
tableProps.formatCells = true;
tableProps.formatColumns = true;
tableProps.formatRows = true;
tableProps.insertAndEditObjects = false;
tableProps.insertAndEditPivotTables = false;
tableProps.insertColumns = false;
tableProps.insertLinks = true;
tableProps.insertRows = false;
tableProps.selectProtectedCells = true;
tableProps.sortData = true;

Table firstSheet = document.getBlocks().getTable(0);
firstSheet.setProtection(tableProps, "password");
```

6.214 Класс TableSearchSettings

Класс `TableSearchSettings` предназначен для настройки параметров поиска ячеек. Данный класс используется в методах `Table.find()` и `CellRange.find()`.

Таблица 128 – Описание полей класса `TableSearchSettings`

Поле	Тип	Описание
`TableSearchSettings.caseSensitive`	`CaseSensitive`	Позволяет учитывать регистр при поиске
`TableSearchSettings.matchBehaviour`	`TableSearchSettings.MatchBehaviour`	Задает алгоритм сравнения запроса и значения
`TableSearchSettings.searchIn`	`TableSearchSettings.SearchProperty`	Позволяет выбрать значения, по которым производится поиск
`TableSearchSettings.wholeWords`	`bool`	Разбивает текст в ячейке на слова, которые считаются отдельными значениями для сравнения

Пример

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

TableSearchSettings searchProps = new TableSearchSettings();
searchProps.caseSensitive = CaseSensitive.No;
searchProps.matchBehaviour = TableSearchSettings.MatchBehaviour.Glob;
```

```
searchProps.searchIn = TableSearchSettings.SearchProperty.Value;
searchProps.wholeWords = true;

CellsEnumerator results = sheet.find("*eye", searchProps);
foreach (Cell cell in results) {
    Console.WriteLine(cell.getFormattedValue()); // Steeleye Stout
}
```

6.214.1 Перечисление TableSearchSettings.MatchBehaviour

Перечисление `MatchBehaviour` содержит алгоритмы сравнения запроса и значения. Используется в поле `TableSearchSettings.matchBehaviour`.

Таблица 129 – Описание алгоритмов сравнения

Значение	Описание
`TableSearchSettings.MatchBehaviour.Partial`	Значение частично содержит запрос
`TableSearchSettings.MatchBehaviour.Total`	Значение полностью соответствует запросу
`TableSearchSettings.MatchBehaviour.Glob`	Позволяет использовать шаблоны, содержащие символы ***** и **?**, для создания запроса

6.214.2 Перечисление TableSearchSettings.SearchProperty

Перечисление `SearchProperty` содержит значения, по которым производится поиск в таблице. Используется в поле `TableSearchSettings.searchIn`.

Таблица 130 – Описание вариантов поиска в таблице

Значение	Описание
`TableSearchSettings.SearchProperty.Value`	Поиск по значениям ячеек
`TableSearchSettings.SearchProperty.Formula`	Поиск по тексту формул в ячейках
`TableSearchSettings.SearchProperty.Notes`	Поиск по тексту заметок

6.215 Класс TextAnchoredPosition

Класс `TextAnchoredPosition` представляет позицию объекта на странице текстового документа (см. `InlineFrame.setPosition()`).

Таблица 131 – Описание полей класса `TextAnchoredPosition`

Поле	Тип	Описание
`TextAnchoredPosition.horizontal`	`HorizontalTexAnchoredPosition`	Позиция по горизонтали
`TextAnchoredPosition.vertical`	`VerticalTextAnchoredPosition`	Позиция по вертикали

Пример

```
TextAnchoredPosition textAnchoredPosition = new TextAnchoredPosition();
textAnchoredPosition.horizontal = new
HorizontalTextAnchoredPosition(HorizontalRelativeTo.Character);
textAnchoredPosition.vertical = new
VerticalTextAnchoredPosition(VerticalRelativeTo.Character);
frame.setPosition(textAnchoredPosition);
```

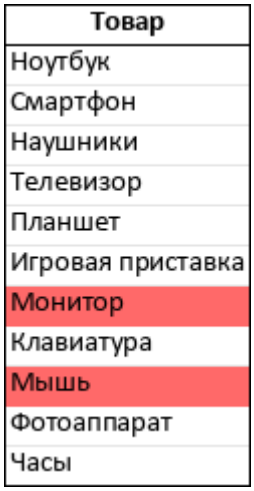
6.216 Класс TextConditionalFormatOperator

Класс `TextConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Текст". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

Конструктор

```
public TextConditionalFormatOperator(ConditionalFormatTextCondition
condition, string argument)
```

Пример



Товар
Ноутбук
Смартфон
Наушники
Телевизор
Планшет
Игровая приставка
Монитор
Клавиатура
Мышь
Фотоаппарат
Часы

Рисунок 63 – Пример создания правила для текстовых значений

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

var textStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(255, 0, 0, 150)));
textStyle.cellProperties = cellProperties;

CellRange cellRange = sheet.getCellRange("A2:A12");
var cellRangePosition = cellRange.getTableRange();

var textOperator =
DocumentAPI.createTextConditionalFormatOperator(ConditionalFormatTextCondition.Be
ginsWith, "M");

var textRule = new ConditionalFormatRule(textOperator, textStyle,
cellRangePosition, false);
rules.addRule(textRule);
```

6.216.1 Метод TextConditionalFormatOperator.getArgument

Метод возвращает аргумент условия.

Вызов

```
public string getArgument()
```

Возвращает

– аргумент условия, тип `string`.

6.216.2 Метод TextConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
public ConditionalFormatTextCondition getCondition()
```

Возвращает

– условие применения форматирования, тип `ConditionalFormatTextCondition`.

6.216.3 Метод TextConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип `ConditionalFormatOperatorType`.

6.217 Класс TextExportSettings

Класс `TextExportSettings` предоставляет настройки, необходимые для экспорта текстовых документов (см. `Document.exportAs`).

Таблица 132 – Описание полей класса `TextExportSettings`

Поле	Тип	Описание
`TextExportSettings.pageNumbers`	`PageNumbers`	Настройки страниц для экспорта текстовых документов
`TextExportSettings.viewMode`	`ViewMode`	Задает то, как будут отображаться исправления в документе
`TextExportSettings.shouldHighlightComments`	`bool`	Выделяет фрагменты текста, которым соответствуют комментарии в документе. Работает только при значениях `WithMarkups` в поле `viewMode`

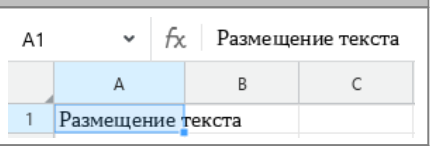
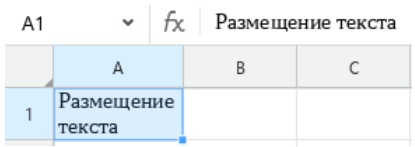
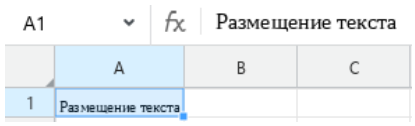
Пример

```
TextExportSettings textExportSettings = new TextExportSettings();
textExportSettings.pageNumbers = new PageNumbers(PageParity.Even);
textExportSettings.viewMode = ViewMode.WithMarkups;
textExportSettings.shouldHighlightComments = true;
document.exportAs("document.pdf", ExportFormat.PDFA1, textExportSettings);
```

6.218 Перечисление TextLayout

В таблице 133 приведены варианты размещения текста в ячейках таблицы. Данное перечисление используется в поле `textLayout` класса `CellProperties`.

Таблица 133 – Варианты размещения текста в ячейках таблицы

Значение	Описание	Отображение
`TextLayout.SingleLine`	Текст располагается в одну строку с наложением на соседние ячейки.	
`TextLayout.WrapByWords`	Текст внутри ячейки переносится по словам. Высота ряда увеличивается чтобы разместить текст полностью.	
`TextLayout.ShrinkSizeToFitWidth`	Текст располагается в одну линию, отображение масштабируется таким образом, чтобы полностью разместиться в ячейке без изменения ее размера. Размер шрифта не изменяется, данная настройка влияет только на отображение содержимого ячейки таблицы.	

Пример

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A3");
CellProperties cellProps = cell.getCellProperties();
cellProps.textLayout = TextLayout.ShrinkSizeToFitWidth;
cell.setCellProperties(cellProps);
```

6.219 Класс TextOrientation

Класс `TextOrientation` содержит угол поворота текста в ячейке, фигуре и т. д. (см. `CellProperties`).

Пример

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

CellProperties cellProps = cell.getCellProperties();
cellProps.textOrientation = new TextOrientation(45);
cell.setCellProperties(cellProps);
```

6.219.1 Метод TextOrientation.getAngle

Возвращает угол направления текста в ячейке. Значение угла указывается в градусах.

Пример

```
CellProperties cellProps = cell.getCellProperties();
cellProps.textOrientation = new TextOrientation(45);
Console.WriteLine(cellProps.textOrientation.getAngle());
```

6.219.2 Метод TextOrientation.isStackedChars

Возвращает `true` в случае, если ориентация текста представляет собой вертикальный столбец.

6.220 Класс TextProperties

Класс `TextProperties` содержит поля, задающие параметры текста. На рисунке 2 изображена объектная модель класса `TextProperties`.

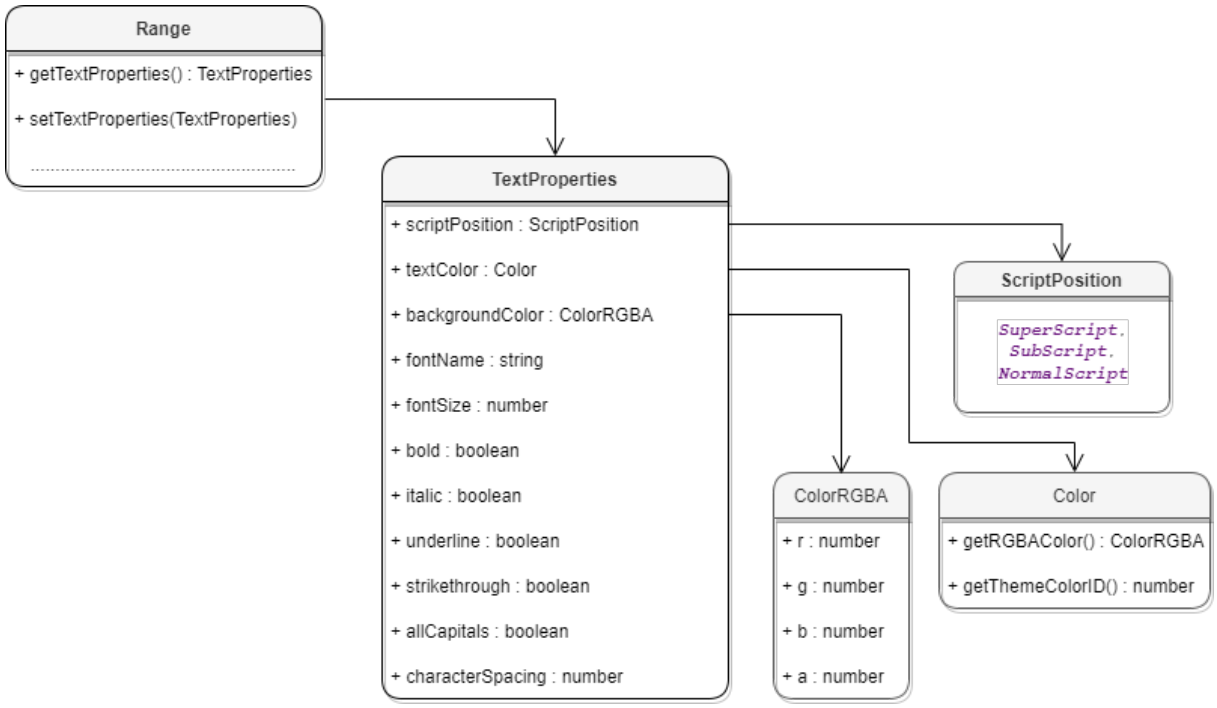


Рисунок 64 – Объектная модель для работы с классом `TextProperties`

Описание полей класса `TextProperties` представлено в таблице 134. Свойства `TextProperties` применяются к диапазону текста `Range` (методы `Range.getTextProperties()`, `Range.setTextProperties()`), ячейке `Cell` (методы `Cell.getTextProperties()`, `Cell.setTextProperties()`) и диапазону ячеек `CellRange` (методы `CellRange.getTextProperties()`, `CellRange.setTextProperties()`).

Таблица 134 – Описание полей класса `TextProperties`

Поле	Тип	Описание
`TextProperties.fontName`	`string`	Наименование шрифта, использованного для оформления фрагмента документа.
`TextProperties.fontSize`	`float`	Размер шрифта, использованного для оформления фрагмента документа.
`TextProperties.bold`	`bool`	Значение `true` устанавливает жирное начертание для указанного фрагмента текста.

Поле	Тип	Описание
TextProperties.italic	bool	Значение true устанавливает начертание курсивом для указанного фрагмента текста.
TextProperties.underline	bool	Значение true устанавливает подчеркивание для указанного фрагмента текста.
TextProperties.strikethrough	bool	Значение true устанавливает начертание «зачеркнутый» для указанного фрагмента текста.
TextProperties.allCapitals	bool	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	float	Размер межсимвольного интервала.

Пример

```
TextProperties textProperties = new TextProperties();
textProperties.fontName = "XO Oriel";
textProperties.fontSize = 20;
// доступ к тексту третьего абзаца
Paragraph paragraph = document.getBlocks().getParagraph(2);
if (paragraph != null) {
    Range range = paragraph.getRange();
    // установить свойства фрагмента текста
    range.setTextProperties(textProperties);
}
```

6.221 Класс TextStyle

Класс `TextStyle` представляет собой стиль абзаца. Используется в методах `Paragraph.getStyle()`, `Paragraph.getResolvedStyle()`, `Paragraph.setStyle()`, `TextStyles.create()`, `TextStyles.get()`, `TextStyle.getParent()`, `TextStyle.getNextParagraphStyle()` и `TextStyle.setNextParagraphStyle()`.

6.221.1 Метод TextStyle.createChild

Метод создает новый стиль на основе текущего стиля абзаца.

Вызов

```
public void createChild(string name)
```

Параметры

– `name`: название нового стиля, тип `string`.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle normalStyle = styles.get(PredefinedTextStyle.Normal);
normalStyle.createChild("myStyle");
TextStyle myStyle = styles.get("myStyle");
TextStyle parentStyle = myStyle.getParent();
Console.WriteLine(myStyle.getName()); // myStyle
Console.WriteLine(parentStyle.getName()); // Normal
```

6.221.2 Метод TextStyle.getName

Метод возвращает название текущего стиля.

Вызов

```
public string getName()
```

Возвращает

– название стиля, тип `string`.

6.221.3 Метод TextStyle.getNextParagraphStyle

Метод возвращает стиль следующего абзаца.

Вызов

```
public TextStyle getNextParagraphStyle()
```

Возвращает

– стиль следующего абзаца, тип TextStyle.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle titleStyle = styles.get(PredefinedTextStyle.Title);
Console.WriteLine(titleStyle.getNextParagraphStyle().getName()); // Normal
```

6.221.4 Метод TextStyle.getParagraphProperties

Метод возвращает настройки форматирования абзаца.

Вызов

```
public ParagraphProperties getParagraphProperties()
```

Возвращает

– настройки форматирования абзаца, тип ParagraphProperties.

6.221.5 Метод TextStyle.getParent

Метод возвращает стиль, на котором основан текущий стиль абзаца.

Вызов

```
public TextStyle getParent()
```

Возвращает

– родительский стиль абзаца, тип TextStyle.

– `null`, если текущий стиль не основан ни на каком стиле.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle normalStyle = styles.get(PredefinedTextStyle.Normal);
normalStyle.createChild("myStyle");
TextStyle myStyle = styles.get("myStyle");
TextStyle parentStyle = myStyle.getParent();
Console.WriteLine(myStyle.getName()); // myStyle
Console.WriteLine(parentStyle.getName()); // Normal
```

6.221.6 Метод TextStyle.getTextProperties

Метод возвращает настройки форматирования текста.

Вызов

```
public TextProperties getTextProperties()
```

Возвращает

– настройки форматирования текста, тип `TextProperties`.

6.221.7 Метод TextStyle.setName

Метод задает название стиля.

Вызов

```
public void setName(string newName)
```

Параметры

– `newName`: название стиля, тип `string`.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle normalStyle = styles.get(PredefinedTextStyle.Normal);
normalStyle.setName("myStyle");
```

6.221.8 Метод TextStyle.setNextParagraphStyle

Метод задает стиль следующего абзаца.

Вызов

```
public void setNextParagraphStyle(TextStyle paragraphStyle)
```

Параметры

– `paragraphStyle`: стиль следующего абзаца, тип `TextStyle`.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle titleStyle = styles.get(PredefinedTextStyle.Title);
titleStyle.setNextParagraphStyle(styles.get(PredefinedTextStyle.Subtitle));
```

6.221.9 Метод TextStyle.setParagraphProperties

Метод задает настройки форматирования абзаца.

Вызов

```
public void setParagraphProperties(ParagraphProperties properties)
```

Параметры

– `properties`: настройки форматирования абзаца, тип `ParagraphProperties`.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle normalStyle = styles.get(PredefinedTextStyle.Normal);
ParagraphProperties props = normalStyle.getParagraphProperties();
props.beforeSpacing = 2.1f;
normalStyle.setParagraphProperties(props);
```

6.221.10 Метод TextStyle.setTextProperties

Метод задает настройки форматирования текста.

Вызов

```
public void setTextProperties(TextProperties textProperties)
```

Параметры

– `textProperties`: настройки форматирования текста, тип `TextProperties`.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle normalStyle = styles.get(PredefinedTextStyle.Normal);
TextProperties props = normalStyle.getTextProperties();
props.fontSize = 20;
normalStyle.setTextProperties(props);
```

6.222 Класс TextStyles

Класс `TextStyles` предоставляет доступ к стилям абзацев в документе. Используется в методе `Document.getTextStyles()`.

6.222.1 Метод TextStyles.create

Метод создает новый стиль абзаца с заданным названием или идентификатором. Если такой стиль уже существует, возникает IncorrectArgumentError.

Вызов

```
public TextStyle create(string newStyleName)
```

```
public TextStyle create(PredefinedTextStyle predefinedTextStyle)
```

Параметры

– newStyleName: название нового стиля, тип string.

– predefinedTextStyle: идентификатор нового стиля, тип PredefinedTextStyle.

Возвращает

– созданный стиль абзаца, тип TextStyle.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle myStyle = styles.create("myStyle");
TextStyle myHeadingStyle = styles.create(PredefinedTextStyle.Heading6);
```

6.222.2 Метод TextStyles.get

Метод возвращает стиль абзаца по его названию или идентификатору.

Вызов

```
public TextStyle get(string styleName)
```

```
public TextStyle get(PredefinedTextStyle predefinedTextStyle)
```

Параметры

– styleName: название стиля, тип string.

– predefinedTextStyle: идентификатор стиля, тип PredefinedTextStyle.

Возвращает

– стиль абзаца, тип TextStyle.

– null, если стиля с заданным названием или идентификатором не существует.

Пример

```
TextStyles styles = document.getTextStyles();
TextStyle normalStyle = styles.get("Normal");
```

```
TextStyle titleStyle = styles.get(PredefinedTextStyle.Title);
Console.WriteLine(normalStyle.getName());
Console.WriteLine(titleStyle.getName());
```

6.222.3 Метод TextStyles.getEnumerator

Метод возвращает перечисление стилей абзацев.

Вызов

```
public TextStylesEnumerator getEnumerator()
```

Возвращает

– перечисление стилей абзацев, тип `TextStylesEnumerator`.

Пример

```
TextStyles styles = document.getTextStyles();
foreach (TextStyle style in styles.getEnumerator())
    Console.WriteLine(style.getName());
```

6.223 Класс TextToColumnsSettings

Класс `TextToColumnsSettings` предназначен для настройки параметров разделения текста по ячейкам. Данный класс используется в методе `CellRange.textToColumns()`.

Таблица 135 – Описание полей класса `TextToColumnsSettings`

Поле	Тип	Описание
`TextToColumnsSettings.customDelimiter`	`string`	Позволяет задать пользовательский разделитель
`TextToColumnsSettings.textQualifier`	`TextQualifier`	Задает символ-ограничитель. Текст между двумя ограничителями записывается в одну ячейку, даже если содержит разделители
`TextToColumnsSettings.treatMultipleDelimitersAsOne`	`bool`	Объединять разделители, расположенные подряд
`TextToColumnsSettings.useCommaDelimiter`	`bool`	Использовать запятую (,) в качестве разделителя
`TextToColumnsSettings.usePercentDelimiter`	`bool`	Использовать процент (%) в качестве разделителя
`TextToColumnsSettings.useSemicolonDelimiter`	`bool`	Использовать точку с запятой (;) в качестве разделителя

Поле	Тип	Описание
TextToColumnsSettings. useSlashDelimiter	bool	Использовать косую черту (/) в качестве разделителя
TextToColumnsSettings. useSpaceDelimiter	bool	Использовать пробел в качестве разделителя
TextToColumnsSettings. useTabDelimiter	bool	Использовать знак табуляции в качестве разделителя

Пример

```
TextToColumnsSettings settings = new TextToColumnsSettings();
settings.useCommaDelimiter = true;
settings.useSpaceDelimiter = true;
settings.treatMultipleDelimitersAsOne = true;
settings.textQualifier = TextToColumnsSettings.TextQualifier.SingleQuotes;
cellRange.textToColumns(settings);
```

6.223.1 Перечисление TextToColumnsSettings.TextQualifier

Перечисление TextQualifier содержит доступные символы-ограничители для горизонтального разделения текста по ячейкам. Текст между двумя ограничителями записывается в одну ячейку, даже если содержит разделители. Используется в поле TextToColumnsSettings.textQualifier.

Таблица 136 – Варианты символов-ограничителей

Значение	Описание
TextToColumnsSettings.TextQualifier.None	Без ограничителей
TextToColumnsSettings.TextQualifier.SingleQuotes	Одинарные кавычки (')
TextToColumnsSettings.TextQualifier.DoubleQuotes	Двойные кавычки (")

Пример

```
TextToColumnsSettings settings = new TextToColumnsSettings();
settings.useCommaDelimiter = true;
settings.useSpaceDelimiter = true;
settings.treatMultipleDelimitersAsOne = true;
```

```
settings.textQualifier = TextToColumnsSettings.TextQualifier.SingleQuotes;
cellRange.textToColumns(settings);
```

6.224 Перечисление TextUnit

В таблице 137 представлены варианты разделения текста на диапазоны. Используется в методах Position.getCurrentRange(), Position.getNextRange() и Position.getPreviousRange().

Таблица 137 – Варианты разделения текста на диапазоны

Значение	Описание
TextUnit.Character	Разделение по символам
TextUnit.Word	Разделение по словам (символы-разделители считаются отдельными словами)
TextUnit.Sentence	Разделение по предложениям (разделители после предложения считаются его частью)
TextUnit.Paragraph	Разделение по абзацам

Пример

```
Range secondSentence =
document.getRange().getBegin().getNextRange(TextUnit.Sentence);
Range firstSentenceLastWord =
secondSentence.getBegin().getPreviousRange(TextUnit.Word);
while
(firstSentenceLastWord.getContentEnd().compare(firstSentenceLastWord.getBegin())
== 1) {
    firstSentenceLastWord =
firstSentenceLastWord.getBegin().getPreviousRange(TextUnit.Word);
}
Range thirdSentenceFirstLetter =
secondSentence.getContentEnd().getCurrentRange(TextUnit.Character);

Console.WriteLine(secondSentence.extractText());
// Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.
```

```
Console.WriteLine(firstSentenceLastWord.extractText());
// aliqua
Console.WriteLine(thirdSentenceFirstLetter.extractText());
// D
```

6.225 Перечисление TextWrapType

В таблице 138 представлены варианты обтекания текстом встроенного объекта. Используется в `InlineFrame.setWrapType()`, `InlineFrame.getWrapType()`.

Таблица 138 – Варианты обтекания текстом встроенного объекта

Значение	Описание
`TextWrapType.Inline`	Встроенный объект располагается в тексте
`TextWrapType.InFrontOfText`	Встроенный объект располагается перед текстом
`TextWrapType.BehindText`	Встроенный объект располагается за текстом
`TextWrapType.TopAndBottom`	Текст располагается сверху и снизу от встроенного объекта
`TextWrapType.Square`	Текст располагается вокруг прямоугольной рамки встроенного объекта
`TextWrapType.Through`	Текст обтекает встроенный объект по сторонам и внутри
`TextWrapType.Tight`	Текст располагается на одинаковых расстояниях от границ объекта

Пример

```
var frame = inlineObject.getFrame();
frame.setWrapType(TextWrapType.Inline);
```

6.226 Перечисление ThemeColorID

В таблице 139 представлены типы идентификаторов цветов тем. Используется в `Color`.

Таблица 139 – Типы идентификаторов цветов тем

Значение	Описание
`ThemeColorID.Background1`	Фон1
`ThemeColorID.Text1`	Текст1
`ThemeColorID.Background2`	Фон2
`ThemeColorID.Text2`	Текст2
`ThemeColorID.Dark1`	Темная1
`ThemeColorID.Dark2`	Темная2
`ThemeColorID.Light1`	Светлая1
`ThemeColorID.Light2`	Светлая2
`ThemeColorID.Accent1`	Акцент1
`ThemeColorID.Accent2`	Акцент2
`ThemeColorID.Accent3`	Акцент3
`ThemeColorID.Accent4`	Акцент4
`ThemeColorID.Accent5`	Акцент5
`ThemeColorID.Accent6`	Акцент6
`ThemeColorID.Hyperlink`	Гиперссылка
`ThemeColorID.FollowedHyperlink`	Посещенная гиперссылка

6.227 Перечисление TimePatterns

Форматы времени представлены в таблице 140. Пример использования см. в разделе `DateTimeCellFormatting`.

Таблица 140 – Форматы времени

Значение	Описание
`TimePatterns.ShortTime`	`'hh:mm AM/PM'` для языка `en_US`
`TimePatterns.LongTime`	`'hh:mm:ss AM/PM'` для языка `en_US`

6.228 Класс TimeZone

Класс `TimeZone` предоставляет настройки, необходимые для экспорта текстовых документов, см. `DocumentSettings`.

Таблица 141 – Описание полей класса `TimeZone`

Поле	Тип	Описание
`TimeZone.offsetInSecondsToUTC`	`int`	Смещение или разность между временем в данном часовом поясе и временем в формате UTC (Всемирное координированное время) в секундах

6.229 Класс TopBottomConditionalFormatOperator

Класс `TopBottomConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Наибольшие и наименьшие значения". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

Конструктор

```
public TopBottomConditionalFormatOperator(ConditionalFormatTopBottomCondition condition, uint value, bool usePercent)
```

Пример

Количество
2
5
10
1
3
4
6
15
20
2
7

Рисунок 65 – Пример создания правила для наибольших 20% значений

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

```
var topBottomStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(255, 0, 0, 150)));
topBottomStyle.cellProperties = cellProperties;

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

var topBottomOperator =
DocumentAPI.createTopBottomConditionalFormatOperator(ConditionalFormatTopBottomCo
ndition.Top, 20, true);

var topBottomRule = new ConditionalFormatRule(topBottomOperator, topBottomStyle,
cellRangePosition, false);
rules.addRule(topBottomRule);
```

6.229.1 Метод TopBottomConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
public ConditionalFormatTopBottomCondition getCondition()
```

Возвращает

– условие применения форматирования, тип ConditionalFormatTopBottomCondition.

6.229.2 Метод TopBottomConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.229.3 Метод TopBottomConditionalFormatOperator.getUsePercent

Метод позволяет определить, выделяется ли определенный процент значений или определенное количество значений.

Вызов

```
public bool getUsePercent()
```

Возвращает

– `true`, если условное форматирование выделяет определенный процент значений; если выделяется определенное количество значений – `false`.

6.229.4 Метод TopBottomConditionalFormatOperator.getValue

Метод возвращает количество/процент значений для выделения.

Вызов

```
public uint getValue()
```

Возвращает

– количество/процент значений для выделения, тип `uint`.

6.230 Класс TrackedChange

Класс `TrackedChange` представляет отслеживаемое изменение в диапазоне документа (см. Рисунок 2).

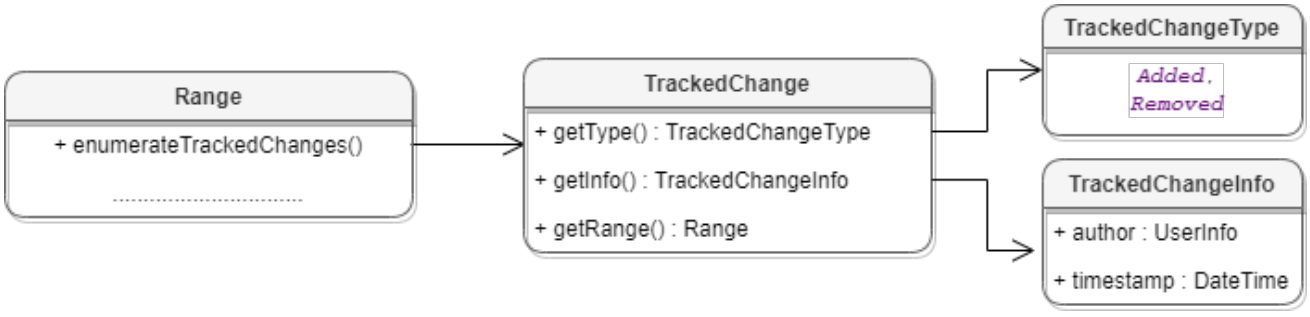


Рисунок 66 – Объектная модель классов для работы с отслеживаемыми изменениями

Для получения списка отслеживаемых изменений используется метод `Range.getTrackedChangesEnumerator()`.

Пример

```
var trackedChangesEnumerator = document.getRange().getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator) {
    ....................................
}
```

6.230.1 Метод TrackedChange.getInfo

Метод позволяет получить информацию об отслеживаемых изменениях TrackedChangeInfo.

Пример

```
var range = document.getRange();
var trackedChangesEnumerator = range.getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator) {
    Console.WriteLine(trackedChange.getInfo().author.name);
}
```

6.230.2 Метод TrackedChange.getRange

Метод возвращает объект Range, который соответствует измененному диапазону внутри абзаца.

Пример

```
var range = document.getRange();
var trackedChangesEnumerator = range.getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator) {
    Console.WriteLine(trackedChange.getRange().extractText());
}
```

6.230.3 Метод TrackedChange.getType

Метод позволяет получить информацию о типе отслеживаемого изменения TrackedChangeType.

Пример

```
var range = document.getRange();
var trackedChangesEnumerator = range.getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator) {
    Console.WriteLine(trackedChange.getType().ToString());
}
```

6.231 Класс TrackedChangeInfo

Класс `TrackedChangeInfo` содержит информацию об отслеживаемых изменениях. Используется в `TrackedChange.getInfo`, `Comment.getInfo`.

Таблица 142 – Описание полей класса `TrackedChangeInfo`

Поле	Тип	Описание
`TrackedChangeInfo.author`	`UserInfo`	Автор изменений
`TrackedChangeInfo.timeStamp`	`DateTime`	Дата и время изменений

Пример

```
var range = document.getRange();
TrackedChangesEnumerator enumerator = range.getTrackedChangesEnumerator();
while (enumerator.MoveNext()) {
    TrackedChange trackedChange = enumerator.Current;
    Console.WriteLine(trackedChange.getInfo().author);
    Console.WriteLine(trackedChange.getInfo().timeStamp);
};
```

6.232 Перечисление TrackedChangeType

Перечисление `TrackedChangeType` содержит типы отслеживаемых изменений, возвращается методом `TrackedChange.getType()`.

Таблица 143 – Типы отслеживаемых изменений

Значение	Описание
`TrackedChangeType.Added`	Добавление содержимого
`TrackedChangeType.Removed`	Удаление содержимого

Значение	Описание
`TrackedChangeType.Formatted`	Форматирование текста
`TrackedChangeType.ParagraphOptions`	Изменение свойств параграфа

Пример

```
var range = document.getRange();
TrackedChangesEnumerator trackedChangesEnumerator =
range.getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator) {
    if (trackedChange.getType() == TrackedChangeType.Added)
        Console.WriteLine("Added");
}
```

6.233 Класс UnaryConditionalFormatOperator

Класс `UnaryConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правил "Больше", "Меньше", "Равно" и "Не равно". Используется при создании правила условного форматирования (`ConditionalFormatRule`).

Конструктор

```
public UnaryConditionalFormatOperator(ConditionalFormatUnaryCondition condition, string argument)
```

Пример

Количество
2
5
10
1
3
4
6
15
20
2
7

Рисунок 67 – Пример создания правила для значений >= 10

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

var unaryStyle = new ConditionalFormatCellStyle();
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(255, 0, 0, 150)));
unaryStyle.cellProperties = cellProperties;

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

var unaryOperator =
DocumentAPI.createUnaryConditionalFormatOperator(ConditionalFormatUnaryCondition.
GreaterOrEqual, "10");

var unaryRule = new ConditionalFormatRule(unaryOperator, unaryStyle,
cellRangePosition, false);
rules.addRule(unaryRule);
```

6.233.1 Метод UnaryConditionalFormatOperator.getArgument

Метод возвращает аргумент условия.

Вызов

```
public string getArgument()
```

Возвращает

– аргумент условия, тип `string`.

6.233.2 Метод UnaryConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
public ConditionalFormatUnaryCondition getCondition()
```

Возвращает

– условие применения форматирования, тип `ConditionalFormatUnaryCondition`.

6.233.3 Метод UnaryConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.234 Класс UniquenessConditionalFormatOperator

Класс `UniquenessConditionalFormatOperator` представляет собой оператор, который содержит настройки применения форматирования для правила "Уникальные и повторяющиеся значения". Используется при создании правила условного форматирования (ConditionalFormatRule).

Конструктор

```
public
UniquenessConditionalFormatOperator(ConditionalFormatUniquenessCondition
condition)
```

Пример

Город
Москва
Санкт-Петербург
Алматы
Минск
Екатеринбург
Казань
Новосибирск
Самара
Астана
Екатеринбург
Минск

Рисунок 68 – Пример создания правила для повторяющихся значений

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

var uniqueStyle = new ConditionalFormatCellStyle();
```

```
var cellProperties = new CellProperties();
cellProperties.fill = new Fill(new Color(new ColorRGBA(255, 0, 0, 150)));
uniqueStyle.cellProperties = cellProperties;

CellRange cellRange = sheet.getCellRange("D2:D12");
var cellRangePosition = cellRange.getTableRange();

var uniqueOperator =
DocumentAPI.createUniquenessConditionalFormatOperator(ConditionalFormatUniqueness
Condition.Duplicate);

var uniqueRule = new ConditionalFormatRule(uniqueOperator, uniqueStyle,
cellRangePosition, false);
rules.addRule(uniqueRule);
```

6.234.1 Метод UniquenessConditionalFormatOperator.getCondition

Метод возвращает условие применения форматирования.

Вызов

```
public ConditionalFormatUniquenessCondition getCondition()
```

Возвращает

– условие применения форматирования, тип ConditionalFormatUniquenessCondition.

6.234.2 Метод UniquenessConditionalFormatOperator.getType

Метод возвращает тип оператора условного форматирования.

Вызов

```
public override ConditionalFormatOperatorType getType()
```

Возвращает

– тип оператора условного форматирования, тип ConditionalFormatOperatorType.

6.235 Класс UserInfo

Класс `UserInfo` предоставляет информацию о пользователе, используется в `TrackedChangeInfo`, `DocumentSettings`.

Таблица 144 – Описание полей класса `UserInfo`

Поле	Тип	Описание
`UserInfo.name`	`string`	Имя пользователя
`UserInfo.email`	`string`	Адрес электронной почты пользователя

6.236 Перечисление ValueFieldsOrientation

Перечисление `ValueFieldsOrientation` содержит варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений. Является полем класса `PivotTableLayoutSettings`.

Таблица 145 – Варианты ориентации значений

Значение	Описание
`ValueFieldsOrientation.ByRows`	По строкам
`ValueFieldsOrientation.ByColumns`	По столбцам

6.237 Класс ValuesTableFilter

Класс `ValuesTableFilter` реализует фильтр, содержащий значения, которые должны быть показаны в диапазоне фильтрации.

Конструктор по умолчанию:

```
ValuesTableFilter();
```

Конструктор копирования:

```
ValuesTableFilter(ValuesTableFilters other);
```

Пример использования приведен в разделе Работа с фильтрами.

6.237.1 Метод ValuesTableFilter.add

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

Пример

```
ValuesTableFilter johnPaulFilter = new ValuesTableFilter();
johnPaulFilter.add("John");
johnPaulFilter.add("Paul");
```

6.237.2 Метод ValuesTableFilter.clear

Метод `ValuesTableFilter.clear` удаляет все элементы фильтра.

Пример

```
ValuesTableFilter johnPaulFilter = new ValuesTableFilter();
johnPaulFilter.add("John");
johnPaulFilter.add("Paul");
// ......................
johnPaulFilter.clear();
```

6.237.3 Метод ValuesTableFilter.remove

Метод удаляет заданное значение из фильтра.

Вызов

```
public void remove(string value)
```

Параметры

– `value`: значение фильтра, тип `string`.

Пример

```
ValuesTableFilter johnPaulFilter = new ValuesTableFilter();
johnPaulFilter.add("John");
johnPaulFilter.add("Paul");
// ...
johnPaulFilter.remove("Sam");
```

6.238 Класс VBAModule

Класс `VBAModule` предоставляет собой VBA макрокоманду. Этот класс используется, как элемент коллекции `VBAModules`.

6.238.1 Метод VBAModule.getCode

Метод возвращает код текущей VBA макрокоманды.

Вызов

```
public string getCode()
```

Возвращает

– код VBA макрокоманды, тип `string`.

6.238.2 Метод VBAModule.getName

Метод возвращает название текущей VBA макрокоманды.

Вызов

```
public string getName()
```

Возвращает

– название VBA макрокоманды, тип `string`.

6.239 Класс VBAModules

Класс `VBAModules` предоставляет собой коллекцию VBA макрокоманд (объектов `VBAModule`), используется в методе `Document.getVBAModules()`.

Пример

```
VBAModules modules = document.getVBAModules();
foreach (VBAModule module in modules) {
    module.getName();
}
```

6.240 Класс VectorString

Тип `VectorString` представляет собой коллекцию данных типа `string`. Используется в качестве контейнера для имен листов в классе `WorkbookExportSettings`.

Примеры использования

```
// Создание вектора
var vector = new VectorString();
// Добавление новых элементов
vector.Add("text1");
vector.Add("text2");
vector.Add("text3");
// Добавление другого вектора
vector.AddRange(vector);
// Определение размера
vector.Capacity = 3;
Console.WriteLine(vector.Capacity);
// Очистка содержимого
vector.Clear();
// Проверка наличия элемента
Console.WriteLine(vector.Contains("text"));
// Сравнение векторов
Console.WriteLine(vector.Equals(vector));
// Определение размера
Console.WriteLine(vector.Count());
// Перечисление элементов
var vectorStringEnumerator = vector.GetEnumerator();
while (vectorStringEnumerator.MoveNext())
{
    String stringValue = vectorStringEnumerator.Current;
    Console.WriteLine(stringValue);
}
// Получение подмножества элементов
VectorString range = vector.GetRange(0, 2);
// Получение индекса элемента
Console.WriteLine(vector.IndexOf("text2"));
// Вставка элемента по индексу
vector.Insert(0, "text4");
// Вставка вектора по индексу
vector.InsertRange(0, vector);
// Индекс последнего вхождения элемента
Console.WriteLine(vector.LastIndexOf("text2"));
```

```
// Удаление по содержимому элемента
vector.Remove("text2");
// Удаление по индексу
vector.RemoveAt(0);
// Удаление диапазона
vector.RemoveRange(0, 2);
// Преобразование в массив
string[] array = vector.ToArray();
```

6.241 Класс VectorUInt

Тип `VectorUInt` представляет собой коллекцию данных типа `uint`. Используется для представления коллекции страниц в `PageNumbers` для экспорта (нечетные, четные, список и т. д.).

Примеры использования

```
// Создание вектора
VectorUInt vector = new VectorUInt(3);
// Добавление новых элементов
vector.Add(1);
vector.Add(2);
vector.Add(3);
// Добавление другого вектора
vector.AddRange(vector);
// Определение размера
vector.Capacity = 3;
Console.WriteLine(vector.Capacity);
// Очистка содержимого
vector.Clear();
// Проверка наличия элемента
Console.WriteLine(vector.Contains(3));
// Сравнение векторов
Console.WriteLine(vector.Equals(vector));
// Определение размера
Console.WriteLine(vector.Count());
// Перечисление элементов
var vectorUIntEnumerator = vector.GetEnumerator();
```

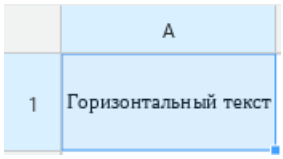
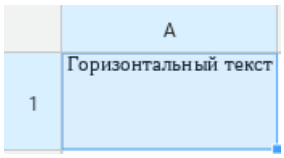
```
while (vectorUIntEnumerator.MoveNext())
{
    uint uintValue = vectorUIntEnumerator.Current;
    Console.WriteLine(uintValue);
}
// Получение подмножества элементов
VectorUInt range = vector.GetRange(0, 2);
// Вставка элемента по индексу
vector.Insert(0, 2);
// Вставка вектора по индексу
vector.InsertRange(0, vector);
// Удаление по индексу
vector.RemoveAt(0);
// Удаление диапазона
vector.RemoveRange(0, 2);
// Преобразование в массив
uint[] array = vector.ToArray();
```

6.242 Перечисление VerticalAlignment

В таблице 146 представлены значения, описывающие варианты выравнивания текста по вертикали. Используется в CellProperties, ShapeProperties.

Таблица 146 – Виды выравнивания текста по вертикали

Значение	Представление в интерфейсе	
VerticalAlignment.Bottom		

Значение	Представление в интерфейсе	
VerticalAlignment.Center		
VerticalAlignment.Top		

Пример

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

CellProperties cellProps = cell.getCellProperties();
cellProps.verticalAlignment = VerticalAlignment.Center;

cell.setCellProperties(cellProps);
```

6.243 Перечисление VerticalAnchorAlignment

В таблице 147 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали (см. описание класса VerticalTextAnchoredPosition).

Таблица 147 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

Значение	Описание
VerticalAnchorAlignment.Top	По верхнему краю
VerticalAnchorAlignment.Bottom	По нижнему краю

Значение	Описание
`VerticalAnchorAlignment.Center`	По центру
`VerticalAnchorAlignment.Inside`, `VerticalAnchorAlignment.Outside`	По границам

6.244 Перечисление VerticalRelativeTo

В таблице 148 представлены типы размещения объекта относительно закрепленной позиции по вертикали (см. описание класса `VerticalTextAnchoredPosition`).

Таблица 148 – Типы размещения объекта относительно закрепленной позиции по вертикали

Значение	Описание
`VerticalRelativeTo.Character`	Символ
`VerticalRelativeTo.BaseLine`	Базовая линия
`VerticalRelativeTo.Paragraph`	Абзац
`VerticalRelativeTo.Page`	Страница
`VerticalRelativeTo.PageContent`	Содержимое страницы
`VerticalRelativeTo.PageTopMargin`	Верхнее поле страницы
`VerticalRelativeTo.PageBottomMargin`	Нижнее поле страницы
`VerticalRelativeTo.PageInsideMargin`	Внутреннее поле страницы
`VerticalRelativeTo.PageOutsideMargin`	Внешнее поле страницы

6.245 Класс VerticalTextAnchoredPosition

Класс `VerticalTextAnchoredPosition` предназначен для управления относительным положением объекта со смещением или выравниванием по вертикали (см. описание класса `TextAnchoredPosition`).

Таблица 149 – Описание полей класса `VerticalTextAnchoredPosition`

Поле	Тип	Описание
`VerticalTextAnchoredPosition.relativeTo`	`VerticalRelativeTo`	Тип размещения объекта относительно

Поле	Тип	Описание
		закрепленной позиции по вертикали
`VerticalTextAnchoredPosition.offset`	`float`	Смещение объекта
`VerticalTextAnchoredPosition.alignment`	`VerticalAnchorAlignment`	Тип выравнивания объекта относительно закрепленной позиции по вертикали

6.246 Перечисление ViewMode

Перечисление `ViewMode` содержит варианты отображения исправлений при экспорте текстового документа. Используется в поле `TextExportSettings.viewMode`.

Таблица 150 – Типы экспортируемых исправлений

Значение	Описание
`ViewMode.Undefined`	Не задано
`ViewMode.WithMarkups`	Экспортирует исходный документ без удаленных фрагментов. Подчеркивает фрагменты текста, для которых есть изменения форматирования.
`ViewMode.Original`	Экспортирует исходный документ.
`ViewMode.OriginalWithMarkups`	Экспортирует исходный документ с выделением изменений.
`ViewMode.Final`	Экспортирует исправленный документ.
`ViewMode.FinalWithMarkups`	Экспортирует исправленный документ с выделением изменений.
`ViewMode.AllChanges`	Экспортирует документ с отображением всех исправлений.

6.247 Класс WorkbookExportSettings

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

Таблица 151 – Описание полей класса `WorkbookExportSettings`

Поле	Тип	Описание
`WorkbookExportSettings.sheetNames`	`VectorString`	Представляет коллекцию имен листов для экспорта. Если коллекция пуста, экспортируются все листы.
`WorkbookExportSettings.printingScope`	`PrintingScope`	Представляет область печати (весь документ, область печати, пользовательский диапазон и т. д.).
`WorkbookExportSettings.pageProperties`	`PageProperties`	Представляют свойства страницы для выходного документа (высота и ширина страницы в пунктах).
`WorkbookExportSettings.scale`	`float`	Представляет масштаб экспорта выходного документа в процентах (например, 50,0%, 150,63%, 400,0% и т. д.).
`WorkbookExportSettings.drawNoneBorders`	`bool`	Добавляет линии сетки в итоговый документ.
`WorkbookExportSettings.printEmptyPages`	`bool`	Включает пустые страницы в итоговый документ.
`WorkbookExportSettings.pageID`	`uint`	Индекс страницы, с которой начинается экспорт.
`WorkbookExportSettings.fitType`	`WorksheetPrinterFitType`	Вариант масштабирования при экспорте табличных документов.

Пример

```
WorkbookExportSettings workbookSettings = new WorkbookExportSettings();
workbookSettings.sheetNames = ["Лист1", "Лист3"];
workbookSettings.printingScope = new PrintingScope(PrintingScope.Type.PrintArea);
workbookSettings.pageProperties = new PageProperties(100, 200);
workbookSettings.scale = 90;
workbookSettings.printEmptyPages = false;
workbookSettings.drawNoneBorders = false;
workbookSettings.fitType = WorksheetPrinterFitType.FitToPage;
string filePath = "C:\\work\\Sheet.pdf";
document.exportAs(filePath, ExportFormat.PDFA1, workbookSettings);
```

6.248 Класс WorksheetHtmlExportSettings

Класс `WorksheetHtmlExportSettings` предназначен для настройки генерации HTML документа на основе листа табличного документа. Данный класс используется в методе `DocumentAPI.exportWorksheetToHtml()`.

Таблица 152 – Описание полей класса `WorksheetHtmlExportSettings`

Поле	Тип	Описание
`WorksheetHtmlExportSettings.exportHeaders`	`bool`	Добавляет заголовки строк и столбцов
`WorksheetHtmlExportSettings.exportMissingBorders`	`bool`	Добавляет линии сетки

6.249 Перечисление WorksheetPrinterFitType

Перечисление `WorksheetPrinterFitType` содержит варианты масштабирования при экспорте табличных документов. Используется в поле `WorkbookExportSettings.fitType`.

Таблица 153 – Варианты масштабирования при экспорте табличных документов

Значение	Описание
`WorksheetPrinterFitType.ActualSize`	Фактический размер
`WorksheetPrinterFitType.ByPageScale`	По масштабу страницы
`WorksheetPrinterFitType.ByPageBreaksOnly`	По разрыву страниц
`WorksheetPrinterFitType.FitToPage`	Вписать в страницу
`WorksheetPrinterFitType.FitToWidth`	Вписать по ширине
`WorksheetPrinterFitType.FitToHeight`	Вписать по высоте

6.250 Исключения

6.250.1 Класс ApplicationCreateError

Исключение `ApplicationCreateError` наследуется от `System.ApplicationException` и возникает в случае, когда объект `Application` не может быть создан.

Пример

```
throw new DocumentAPI.ApplicationCreateError("Can not create application");
```

6.250.2 Класс CoreVersionMismatchError

Исключение `CoreVersionMismatchError` наследуется от `System.ApplicationException` и возникает в случае, когда клиент не поддерживает текущую версию сервера коллаборации.

Пример

```
throw new DocumentAPI.CoreVersionMismatchError("Versions mismatch error");
```

6.250.3 Класс DocumentCreateError

Исключение `DocumentCreateError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть создан.

Пример

```
throw new DocumentAPI.DocumentCreateError("Can not create document");
```

6.250.4 Класс DocumentExportError

Исключение `DocumentExportError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть экспортирован.

Пример

```
throw new DocumentAPI.DocumentExportError("Can not export document");
```

6.250.5 Класс DocumentLoadError

Исключение `DocumentLoadError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть загружен.

Пример

```
throw new DocumentAPI.DocumentLoadError("Can not load document");
```

6.250.6 Класс DocumentModificationError

Исключение `DocumentModificationError` наследуется от `System.ApplicationException` и возникает в случае, когда невозможно выполнить операцию по изменению документа.

Пример

```
throw new DocumentAPI.DocumentModificationError("Can not modify document");
```

6.250.7 Класс DocumentSaveError

Исключение `DocumentSaveError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть сохранен.

Пример

```
throw new DocumentAPI.DocumentSaveError("Can not save document");
```

6.250.8 Класс ForbiddenActionError

Исключение `ForbiddenActionError` наследуется от `System.ApplicationException` и возникает в случае выполнения запрещенной операции.

Пример

```
throw new DocumentAPI.ForbiddenActionError("Forbidden action");
```

6.250.9 Класс IncorrectArgumentError

Исключение `IncorrectArgumentError` наследуется от `System.ApplicationException` и возникает в случае, когда один из аргументов метода или функции имеет недействительное значение.

Пример

```
throw new DocumentAPI.IncorrectArgumentError("Invalid arguments");
```

6.250.10 Класс IncorrectPasswordError

Исключение `IncorrectPasswordError` наследуется от `System.ApplicationException` и возникает в случае ввода неверного пароля.

Пример

```
throw new DocumentAPI.IncorrectPasswordError("Password is incorrect");
```

6.250.11 Класс InvalidObjectError

Исключение `InvalidObjectError` наследуется от `System.ApplicationException` и возникает в случае, когда объект больше не может быть использован.

Пример

```
throw new DocumentAPI.InvalidObjectError("Can not use this object");
```

6.250.12 Класс NoSuchElementError

Исключение `NoSuchElementError` наследуется от `System.ApplicationException` и возникает в случае, когда элемент не существует.

Пример

```
throw new DocumentAPI.NoSuchElementError("Element not exists");
```

6.250.13 Класс NotImplementedError

Исключение `NotImplementedError` наследуется от `System.ApplicationException` и возникает в случае, если обнаружена нереализованная функциональность.

Пример

```
throw new DocumentAPI.NotImplementedError("Not implemented");
```

6.250.14 Класс OutOfRangeError

Исключение `OutOfRangeError` наследуется от `System.ApplicationException` и возникает в случае обнаружения выхода значения за пределы диапазона.

Пример

```
throw new DocumentAPI.OutOfRangeError("Index out of range");
```

6.250.15 Класс ParseError

Исключение `ParseError` наследуется от `System.ApplicationException` и возникает в случае, когда параметр текста не прошел синтаксический анализ.

Пример

```
throw new DocumentAPI.ParseError("Parse error");
```

6.250.16 Класс PivotTableError

Исключение `PivotTableError` наследуется от `System.ApplicationException` и возникает в случае ошибки при работе со сводными таблицами. Например, применение фильтра, который не может быть применен к сводной таблице.

Пример

```
throw new DocumentAPI.PivotTableError("Pivot table error");
```

6.250.17 Класс PositionDocumentsMismatchError

Исключение `PositionDocumentsMismatchError` наследуется от `System.ApplicationException` и возникает в случае, когда несколько позиций относятся к различным документам и не могут быть использованы в одной операции. Данное исключение возникает при попытке пользователя создать диапазон (`Range`), включающий позиции (`Position`), принадлежащие нескольким различным документам, и выполнить операцию для такого диапазона.

Пример

```
throw new DocumentAPI.PositionDocumentsMismatchError("Position mismatch
error");
```

6.250.18 Класс PositionScopeMismatchError

Исключение `PositionScopeMismatchError` наследуется от `System.ApplicationException` и возникает в случае, когда несколько позиций относятся к различным элементам документа и не могут быть использованы в одной операции.

Пример

```
throw new DocumentAPI.PositionScopeMismatchError("Position mismatch error");
```

6.250.19 Класс ScriptExecutionError

Исключение `ScriptExecutionError` наследуется от `System.ApplicationException` и возникает в случае, когда сценарий не удается выполнить.

Пример

```
throw new DocumentAPI.ScriptExecutionError("Can not execute scenario");
```

6.250.20 Класс SpreadsheetProtectionError

Исключение `SpreadsheetProtectionError` наследуется от `System.ApplicationException` и возникает в случае попытки изменения защищенного табличного документа.

Пример

```
throw new DocumentAPI.SpreadsheetProtectionError("Sheet is protected from changes");
```

6.250.21 Класс UnknownError

Исключение `UnknownError` наследуется от `System.ApplicationException` и возникает в случае, когда критическое исключение возникло по неизвестной причине. Приложение должно быть завершено, поскольку возникло неопределенное состояние ядра Document API.

Пример

```
throw new DocumentAPI.UnknownError("Unknown error");
```

7 МЕХАНИЗМ КОНТРОЛЯ ВЕРСИЙ DOCUMENT API

Константы версии Document API `Major` и `Minor` позволяют проверить совместимость предыдущей и текущей версии Document API.

Если была изменена константа `Major` версии Document API, то в Document API произошли обратно несовместимые изменения, и программный код должен быть пересмотрен и обновлен. Обратно несовместимыми изменениями считаются: переименование, удаление или несовместимое изменение подписи существующих классов или методов, а также добавление новых методов, типов и полей класса.

Если была изменена константа `Minor` версии Document API, то в Document API произошли только обратно совместимые изменения, и нет необходимости менять программный код, чтобы он работал с более новой версией Document API. Но гарантируется совместимость только на уровне исходного кода C#, поэтому необходимо перекомпилировать программный код приложения с более новой версией Document API.

Рекомендуется проверить версию Document API до инициализации, как указано ниже:

```
uint ExpectedMajorAPIVersion = 1;
uint ExpectedMinorAPIVersion = 0;
if (!DocumentAPI.isAPIVersionCompatible(ExpectedMajorAPIVersion,
ExpectedMinorAPIVersion)) {
    // Вывод сообщения о несовместимости версии библиотеки Document API и выход
из программы
}
```

Пример проверки совместимости указанной версии Document API с текущей:

```
public class DocumentAPI {
  public static bool isAPIVersionCompatible(uint major, uint minor);
}
```