ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)» MYOFFICE DOCUMENT APPLICATION PROGRAMMING INTERFACE (API). БИБЛИОТЕКА ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#

РУКОВОДСТВО ПРОГРАММИСТА

3.2

Версия 1

На 288 листах

Дата публикации: 17.12.2024

Москва
2024

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

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

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

СОДЕРЖАНИЕ

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

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.

1.4 Системные требования

Сборку приложения можно осуществить с помощью утилит командной строки. Убедитесь, что используемая версия инструментария позволяет выполнить сборку 64-разрядного кода.

Полный перечень требований к программному и аппаратному обеспечению приведен в документе «МойОфис Комплект Средств Разработки (SDK). MyOffice Document Application Programming Interface (API). Системные требования».

2 ПОДГОТОВКА К РАБОТЕ

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_3.2.zip
Linux	MyOfficeSDKDocumentAPI_CSharp_Linux_3.2.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_3.2`, 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`,
`libcrypto-ossl.dll`,
`libcrypto-ossl.lib`,
`libssl-ossl.dll`,
`libssl-ossl.lib`,
`sbb.dll`,
`sbb.lib`
	`libDocumentAPI.so`,
	`NCT.MyOfficeSDK.dll`,
	`libctypto.so`
	`libctypto.so.1.1`
	`libssl.so`,
	`libssl.so.1.1`,
	`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, Word!");
                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).

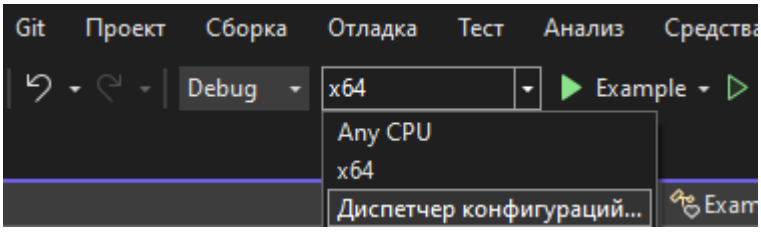


Рисунок 1 – Выбор диспетчера конфигурации в Microsoft Visual Studio

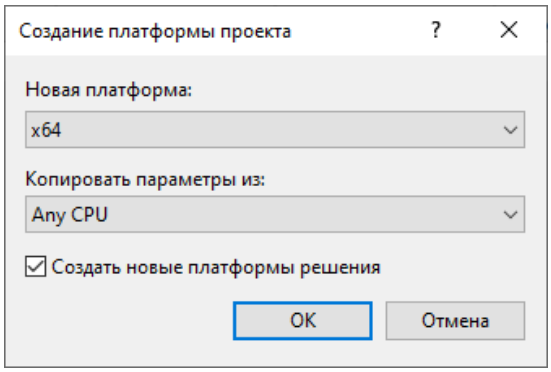


Рисунок 2 – Окно создания платформы проекта

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`, `libcrypto-ossl.dll`, `libssl-ossl.dll`, `sbb.dll` и папку `Resources` из папки `MyOfficeSDKDocumentAPI_CSharp_3.2` каталога установки MyOffice Document API.

Далее в Microsoft Visual Studio в окне **Обозреватель решений** (см. Рисунок 3) нужно выбрать раздел **Зависимости,** нажать правую кнопку мыши и выбрать пункт **Добавить ссылку на модель COM**.

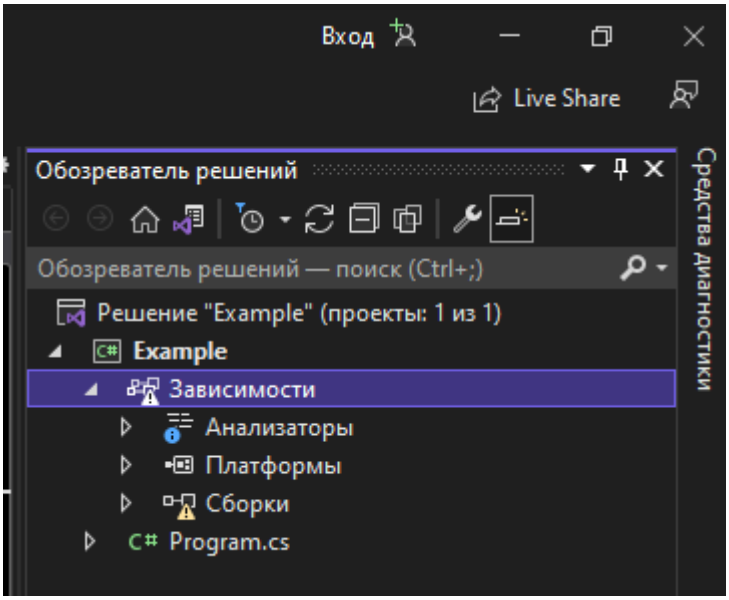


Рисунок 3 – Окно обозревателя решений Microsoft Visual Studio

В открывшемся окне нажать кнопку **Обзор** и выбрать из папки `MyOfficeSDKDocumentAPI_CSharp_3.2` каталога установки MyOffice Document API файл `NCT.MyOfficeSDK.dll` (см. Рисунок 4).

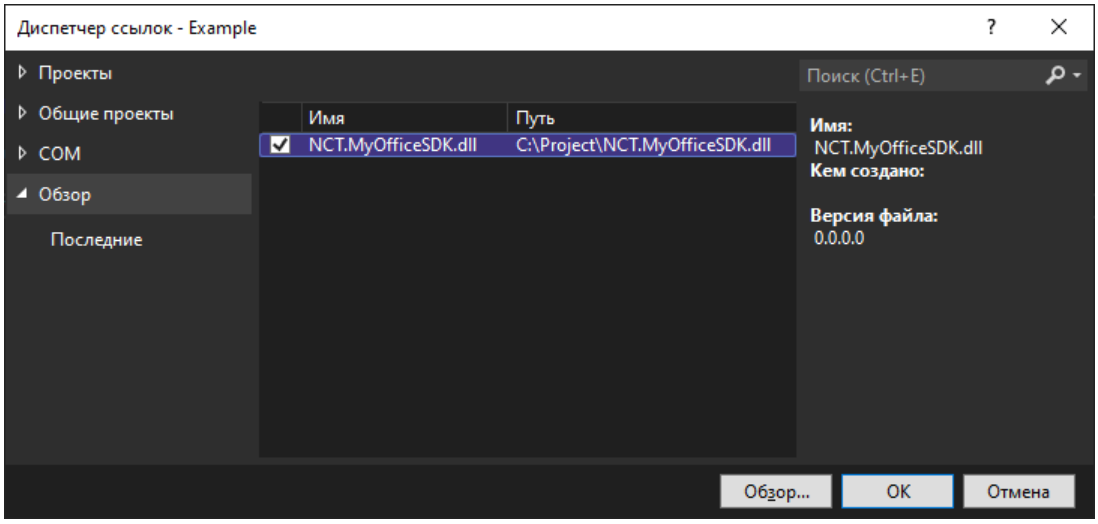


Рисунок 4 – Окно выбора файла библиотеки

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 5):

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

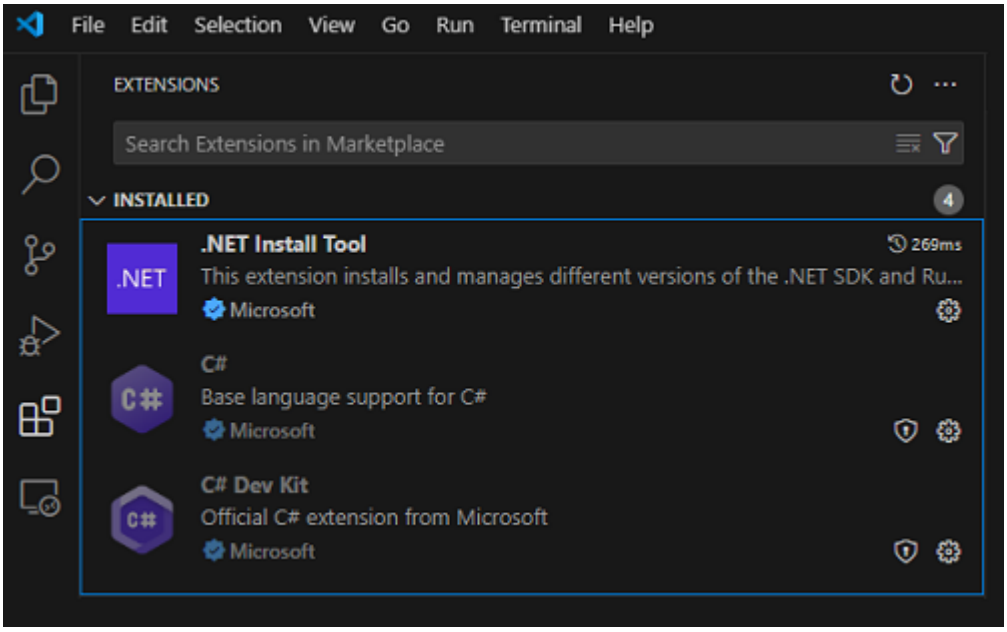


Рисунок 5 – Установка расширений C# в Microsoft Studio Code

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 6):

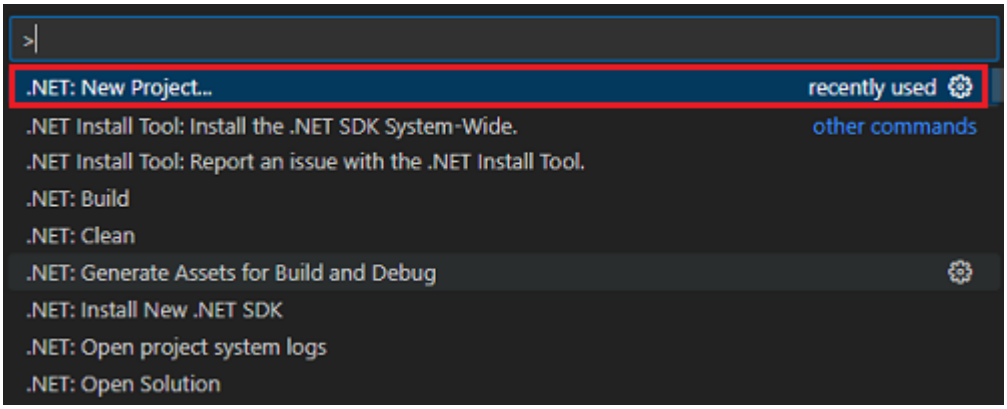


Рисунок 6 – Создание нового проекта

На экране появится список возможных типов проекта, следует выбрать **ConsoleApp** (см. Рисунок 7):

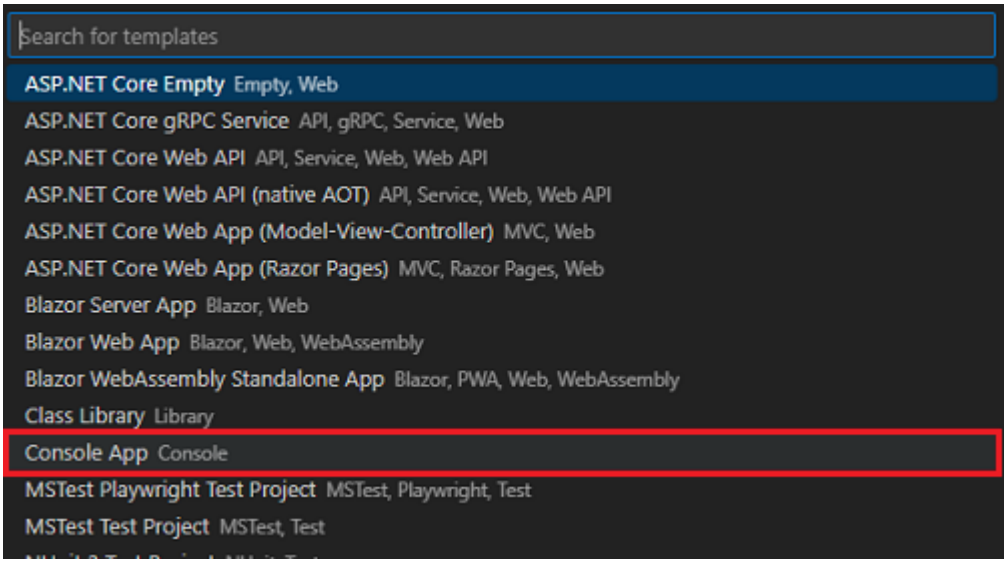


Рисунок 7 – Выбор типа проекта

Далее будет предложено выбрать имя проекта (см. Рисунок 8):

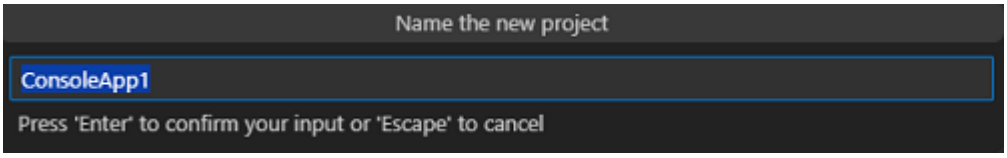


Рисунок 8 – Выбор имени проекта

Выбор папки расположения проекта (см. Рисунок 9):

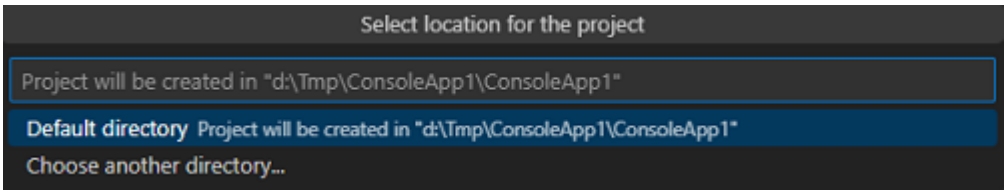


Рисунок 9 – Выбор расположения проекта

Далее следует подтвердить создание проекта (см. Рисунок 10):

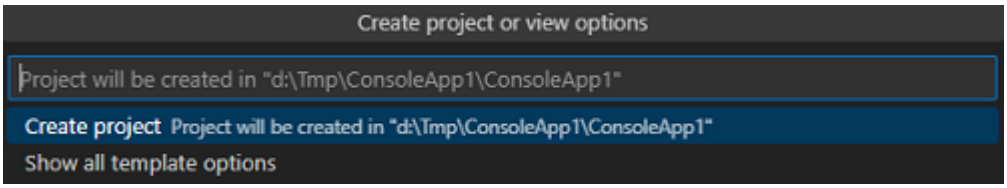


Рисунок 10 – Подтверждение создания проекта

Проект создан, необходимо запустить сборку (меню **Run > Run without debugging**, **Ctrl+F5**), и убедиться, что в строке терминала отображается вывод `"Hello, World!"` (см. Рисунок 11).

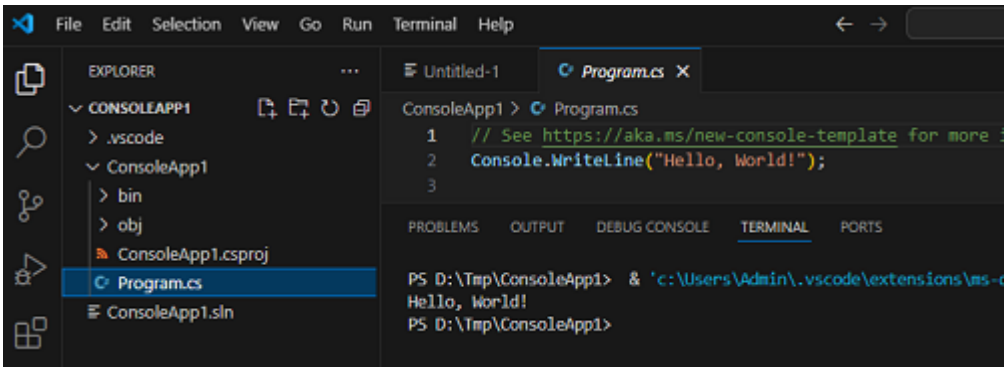


Рисунок 11 – Созданный проект, вывод в консоли

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_3.2\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 12). `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.

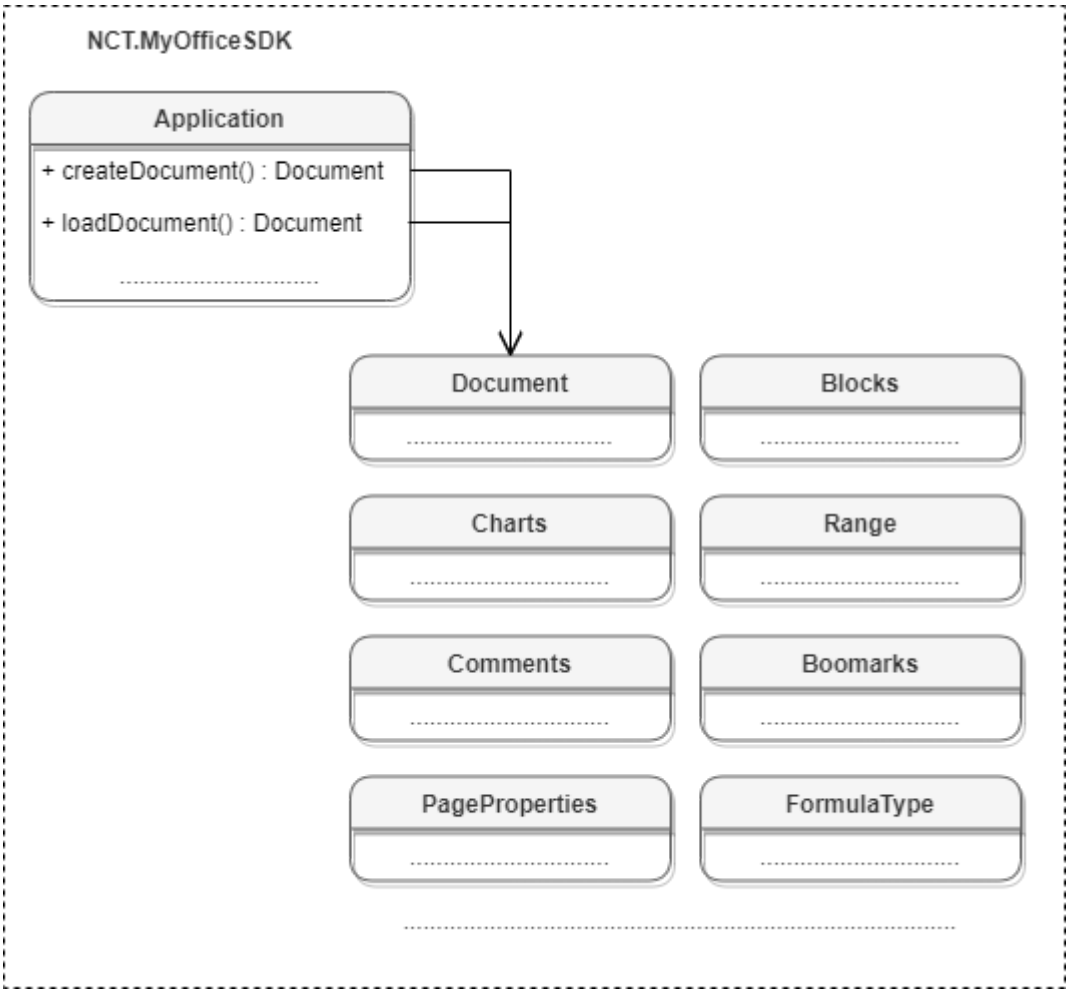


Рисунок 12 – Объектная модель МойОфис C# SDK.

4 РАБОТА С ДОКУМЕНТАМИ

4.1 Работа с текстовым документом

4.1.1 Создание и открытие текстового документа

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

Примеры создания текстового документа:

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

```
var documentSettings = new DocumentSettings();
documentSettings.documentType = DocumentType.Workbook;
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 13 изображена объектная модель классов, относящихся к работе с секциями текстового документа.

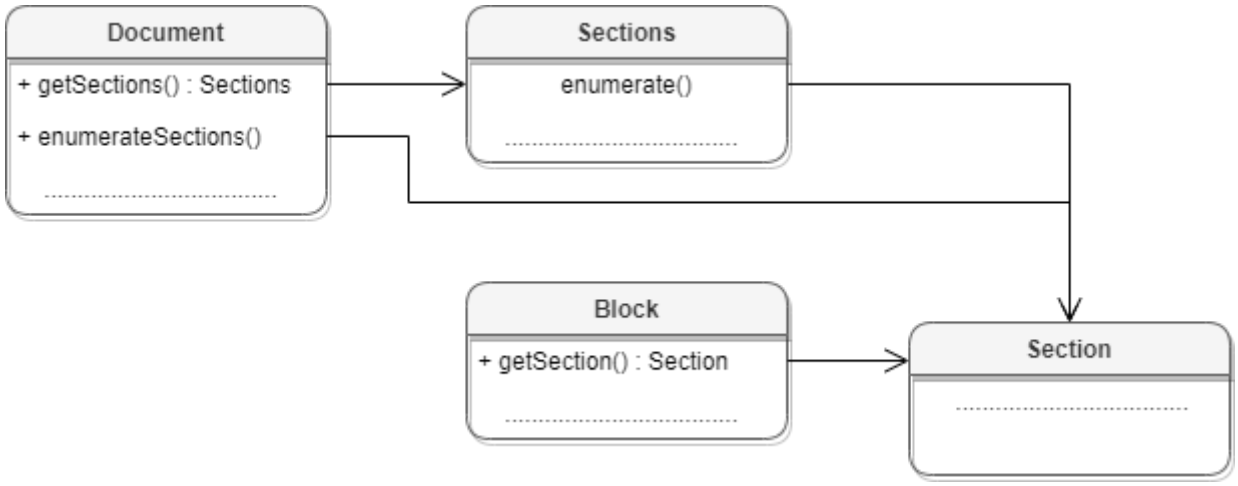


Рисунок 13 – Объектная модель классов для работы с секциями

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:

```
Application app = new Application();

// Загрузка текстового документа
// Документ sections.docx содержит несколько
// разделов (sections). На каждой странице присутствует
```

```
// верхний (header) и нижний (footer) колонтитул.
var doc = app.loadDocument("sections.docx");

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

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

```
var section = doc.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().

```
Range range = document.getRange();
Image insertedImage = range.getBegin().insertImage("C://Tmp//123.jpg", new
SizeU(50, 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 14).

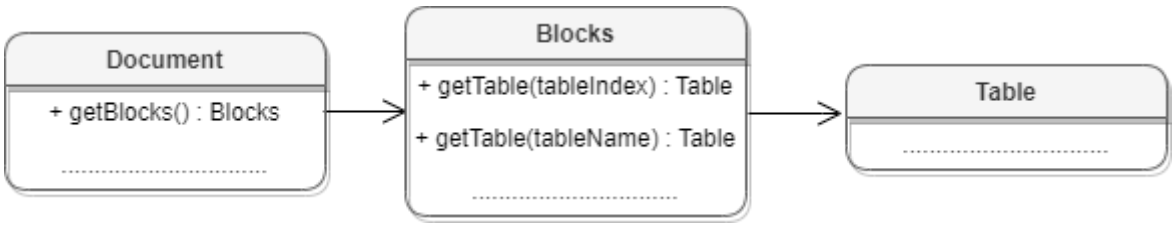


Рисунок 14 – Объектная модель для работы с таблицами

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 15).

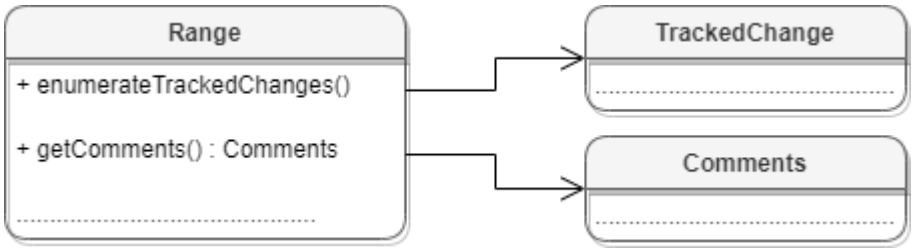


Рисунок 15 – Инструменты рецензирования документа

4.1.8 Работа с элементами управления

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

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

```
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(' ', '_'));
```

Поле с выпадающим списком 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

4.2.1 Создание и открытие табличного документа

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

Пример создания табличного документа:

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

Метод `Application.loadDocument` открывает документ, находящийся по указанному пути.

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 = new VectorString();
workbookSettings.sheetNames.Add("New List");
workbookSettings.printingScope = new
PrintingScope(PrintingScope.Type.PrintArea);
workbookSettings.pageProperties = new PageProperties(100, 200);
workbookSettings.scale = 90;
document.exportAs(filePath, ExportFormat.PDFA1, workbookSettings);
```

4.2.3 Диаграммы

Работа с диаграммами доступна только в табличных документах (см. Рисунок 16).

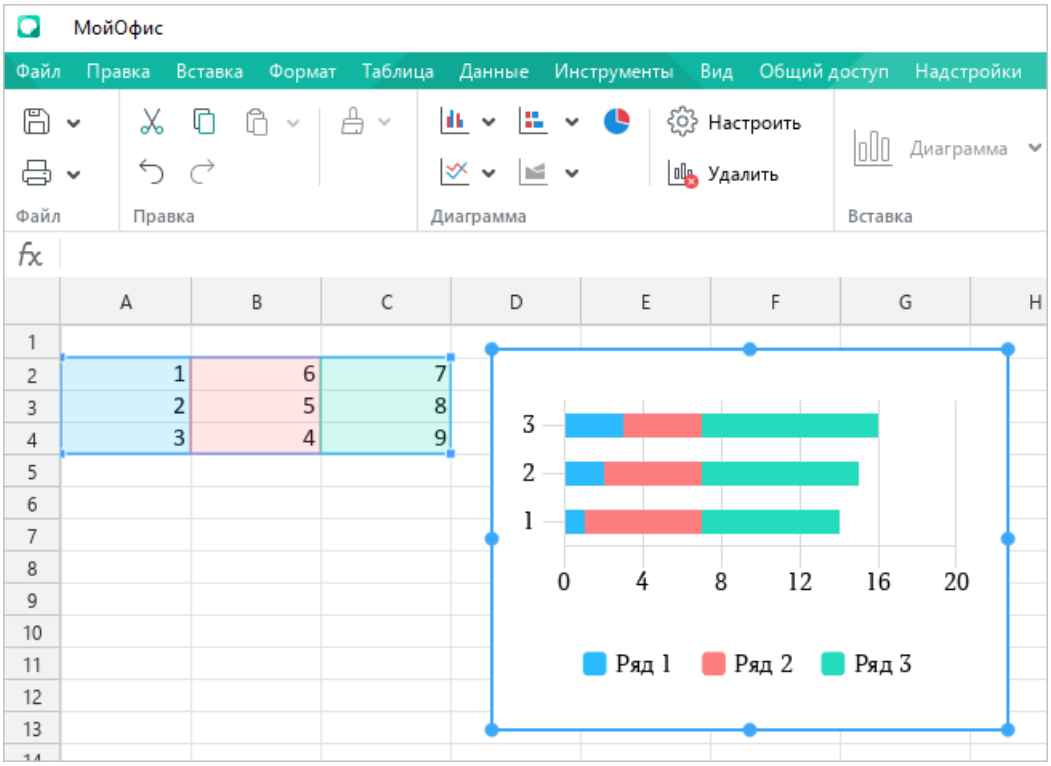


Рисунок 16 – Пример отображения диаграммы в «МойОфис Таблица»

На рисунке 17 изображена объектная модель классов, относящихся к работе с диаграммами.

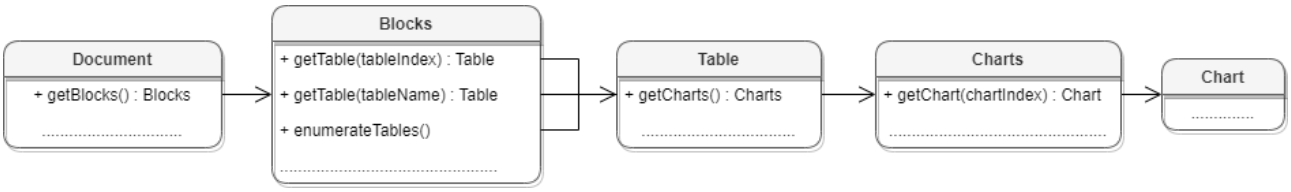


Рисунок 17 – Классы для работы с диаграммами

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

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

Создание и удаление диаграмм в текущей версии не поддерживаются.

4.2.4 Копирование ячеек в табличном документе

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

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

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

var leftTopCellPositoin = new CellPosition(0, 0);
var rightBottomCellPositoin = new CellPosition(1, 1);
var srcCellRangePosition = new CellRangePosition(leftTopCellPositoin,
rightBottomCellPositoin);

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.

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

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

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

```
if (document.getCalculationMode() == 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);
```

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 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:

– 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()`.

4.2.6.1 Вставка изображения

Вставка изображений в табличный документ на данный момент не поддерживается.

4.2.6.2 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)
{
    ..........................
}
```

4.2.7 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 18).

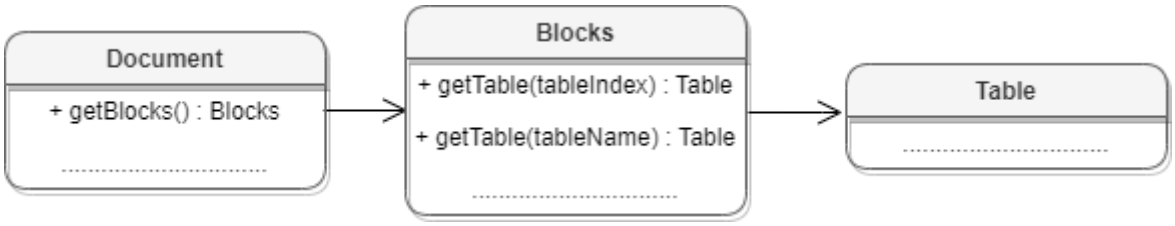


Рисунок 18 – Объектная модель для работы с таблицами

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();
}
```

4.2.8 Работа со сводными таблицами

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

приведена на рисунке 19.

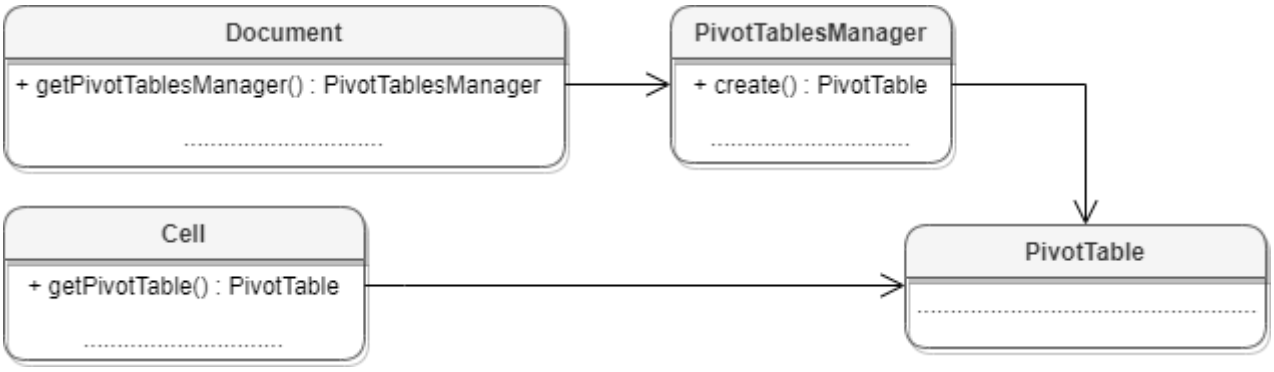


Рисунок 19 – Сводные таблицы

4.2.8.1 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());
```

4.2.8.2 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();
```

4.2.8.3 Получение неподдерживаемых свойств сводной таблицы

Для получения неподдерживаемых свойств сводной таблицы используется метод PivotTable.getUnsupportedFeatures().

Пример:

```
// Получаем неподдерживаемые свойства сводной таблицы
var pivotUnsuportedFeatures = pivotTable.getUnsupportdeFeatures();
```

4.2.8.4 Получение флагов отображения общих итогов для строк и колонок

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

Пример:

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

4.2.8.5 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);
```

4.2.8.6 Получение и применение фильтра для сводной таблицы

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

Пример:

```
// По названию поля сводной таблицы получаем фильтр
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();
```

4.2.8.7 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);
}
```

4.2.8.8 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);
}
```

4.2.8.9 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);
}
```

4.2.8.10 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);
}
```

4.2.8.11 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);
```

4.2.8.12 Обновление сводной таблицы

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

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

4.2.9 Работа с фильтрами

Работа с фильтрами возможна только в табличном документе. Диаграмма взаимодействия объектов приведена на рисунке 20.

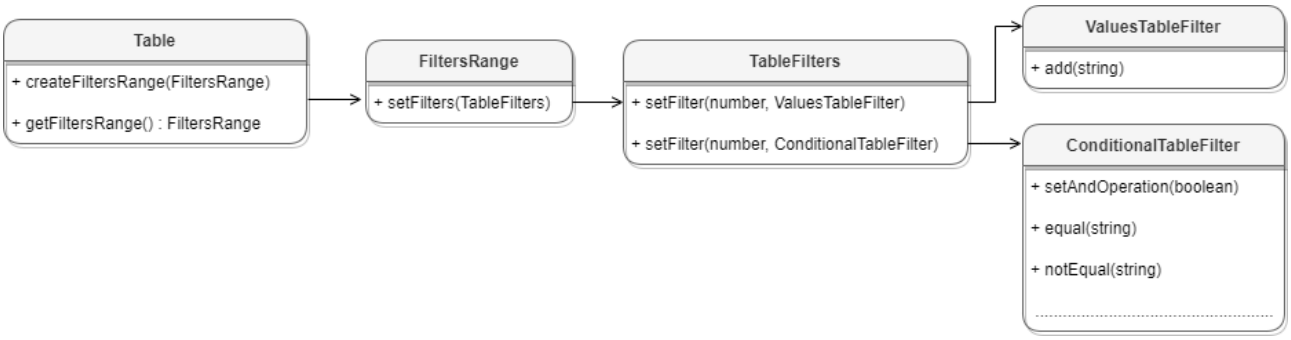


Рисунок 20 – Объектная модель таблиц для работы с фильтрами

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

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 в текстовом или табличном документе необходимо создать экземпляр класса `Search` посредством вызова `createSearch(document)`, затем использовать метод `Search.findText` (см. Рисунок 21).

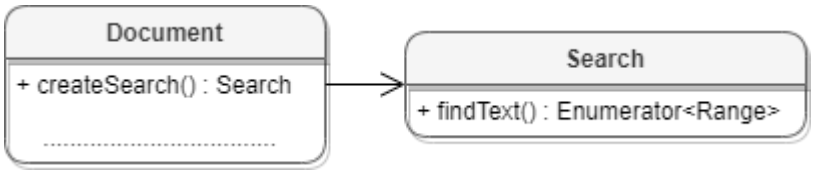


Рисунок 21 – Объектная модель для поиска в документе

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());
}
```

4.5 Работа с макрокомандами

Класс `Scripts` предоставляет доступ к списку макрокоманд документа. На рисунке 22 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)`.

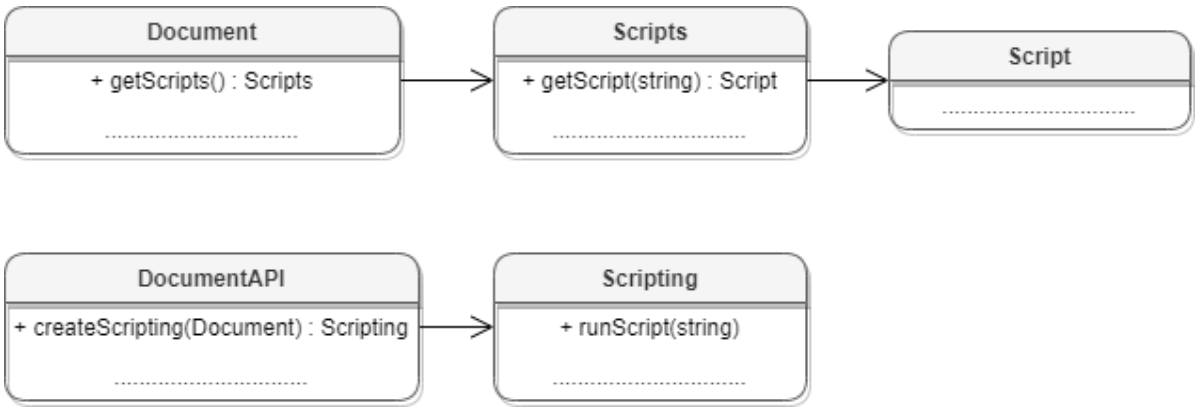


Рисунок 22 – Объектная модель таблиц для работы с макрокомандами

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 23).

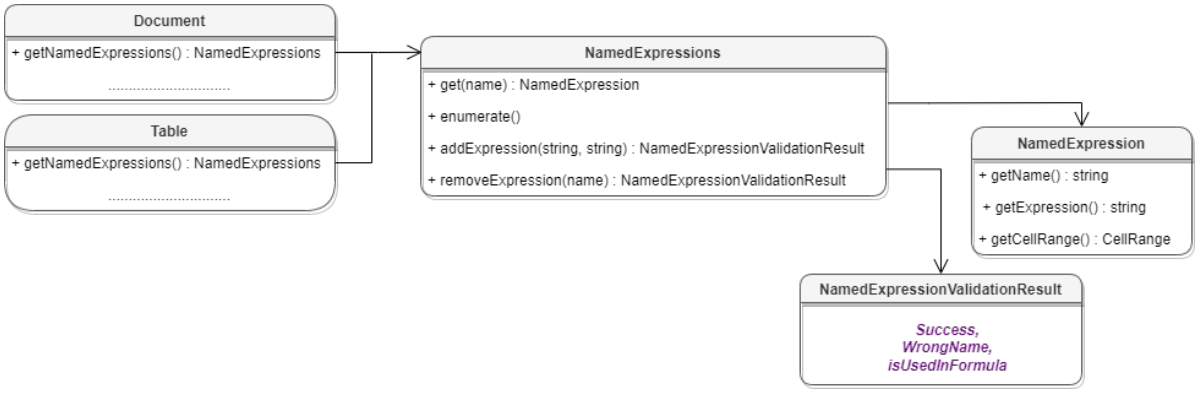


Рисунок 23 – Таблицы для работы с именованными диапазонами

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 свойств именованного диапазона

Пример:

```
var table = document.getBlocks().getTable(0);
var namedExpressions = table.getNamedExpressions();
//  Получить именованное выражение с именем "Alice_Age"
var expression = namedExpressions.get("Alice_Age");
```

```
//  Далее используются поля структуры NamedExpression:
var name = expression.getName();
var formula = expression.getExpression();
var range = expression.getCellRange();
```

4.6.3 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
}
```

4.6.4 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);
```

4.6.5 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);
```

4.6.6 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());
```

4.7 Работа 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.clearColumnnGroups().

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

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

4.7.2 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

4.8.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 24):

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

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

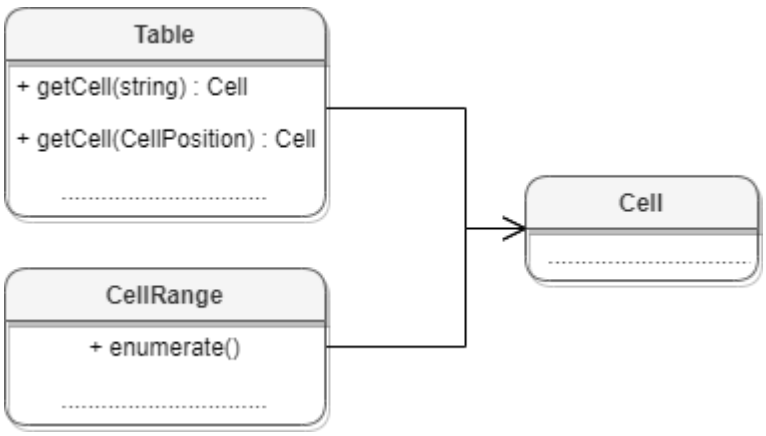


Рисунок 24 – Объектная модель для работы с ячейками таблиц

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 используется метод `Cell.getFormattedValue()`.

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 этим настройками используются методы `Cell.getParagraphProperties()` и `Cell.setParagraphProperties()`.

Пример установки и получения свойств параграфа ячейки:

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

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

Управление настройками текста ячейки (шрифт, цвет) производится через соответствующий ему диапазон. Класс Cell позволяет получить диапазон для всего контента с помощью метода `Cell.getRange()`. Далее, метод `Range.getTextProperties()` позволяет получить экземпляр класса `TextProperties`, представляющий свойства текста.

После изменения значения свойств их необходимо применить к тексту ячейки с помощью метода `Range.setTextProperties()`.

Пример настроек текста ячейки:

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

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

cell.getRange().setTextProperties(textProperties);
```

4.8.3 Форматирование границ ячеек

Для оформления границ ячеек используется класс `Borders` (см. Рисунок 25). 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`.

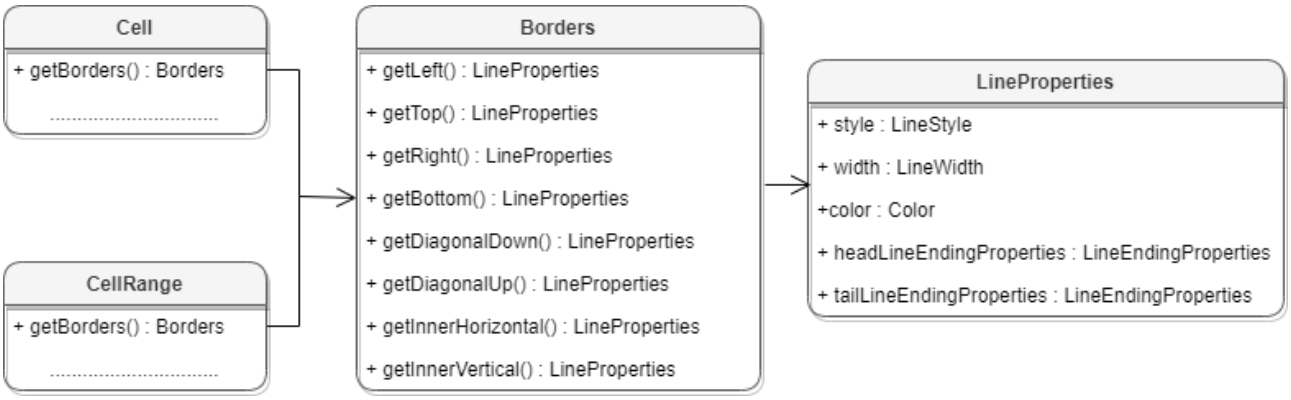


Рисунок 25 – Классы для работы с границами ячеек

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();
```

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();
```

5 ГЛОБАЛЬНЫЕ МЕТОДЫ

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");
```

6 СПРАВОЧНИК КЛАССОВ

Далее приведено описание классов, структур и методов библиотеки MyOffice Document API для языка программирования C#. Разделы приведены в алфавитном порядке.

6.1 Класс AbsoluteFrame

Класс `AbsoluteFrame` описывает прямоугольную область медиаобъекта, находящегося в абсолютной позиции документа (см. Рисунок 26). Предназначен для получения и изменения свойств позиции медиаобъектов. Используется в табличном документе. Может быть получен с помощью метода `Frame.getAbsoluteFrame()`.

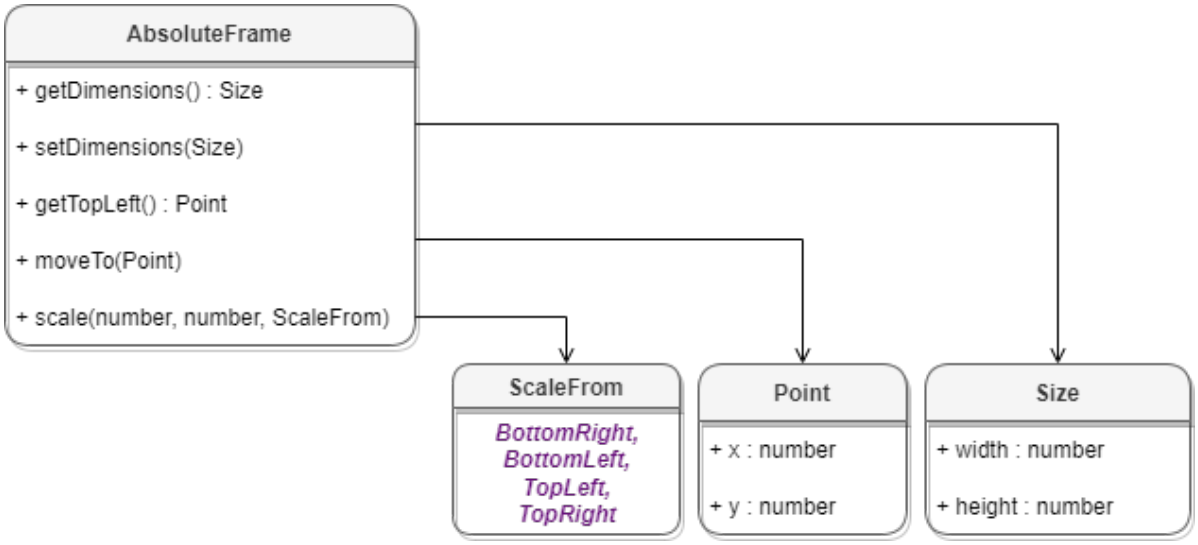


Рисунок 26 – Объектная модель класса `AbsoluteFrame`

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);
    }
}
```

6.1.1 Метод AbsoluteFrame.getDimensions

Метод возвращает задает размеры встроенного объекта, тип - SizeU.

Пример:

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

6.1.2 Метод AbsoluteFrame.getTopLeft

Метод возвращает позицию верхней левой точки объекта, тип - PointU.

Пример:

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

6.1.3 Метод AbsoluteFrame.moveTo

Метод задает позицию встроенного объекта. В качестве параметров передаются координаты объекта.

Пример:

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

6.1.4 Метод AbsoluteFrame.scale

Метод масштабирует объект. В качестве параметров выступают новая длина, новая ширина, а также якорь ScaleFrom, относительно которого производится масштабирование.

Пример:

```
var frame = mediaObject.getFrame();
var absoluteFrame = frame.getAbsoluteFrame();
```

```
if (absoluteFrame != null)
{
    absoluteFrame.scale(100, 100, ScaleFrom.TopLeft);
}
```

6.1.5 Метод AbsoluteFrame.setDimensions

Метод позволяет задать размер встроенного 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);
}
```

6.2 Класс AccountingCellFormatting

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

Описание полей класса AccountingCellFormatting представлено в таблице 2.

Таблица 2 – Описание полей класса AccountingCellFormatting

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

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());
```

6.3 Класс Alignment

Тип `Alignment` содержит варианты горизонтального выравнивания текста, в том числе в ячейке таблицы (см. `ParagraphProperties`).

Варианты горизонтального выравнивания текста:

- `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;
```

6.4 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();
```

6.4.1 Метод Application.createDocument

Метод `Application.createDocument` создает новый документ. В качестве параметра метода используются `DocumentType` или `DocumentSettings`. Возвращает тип `Document`.

Существуют следующие варианты реализации метода:

```
Document createDocument(DocumentType documentType);
Document createDocument(DocumentSettings documentSettings);
```

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

6.4.2 Метод Application.getMessenger

Метод `Application.getMessenger` возвращает объект `Messenger`, реализующий логирование событий.

Пример:

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

6.4.3 Метод Application.loadDocument

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

Существуют следующие варианты реализации метода:

```
Document loadDocument(String filePath);
Document loadDocument(String filePath, LoadDocumentSettings loadSettings);
```

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

Внимание! Данный метод использует текстовый параметр в формате unicode string. Если символы данной строки отличаются от латиницы и кодируются двумя байтами (например, кириллица), то необходимо принять дополнительные меры, описанные в разделе Использование кодировок. В противном случае возникнет исключение `DocumentAPI.UnknownError` с сообщением «Invalid UTF-8».

6.5 Класс Block

Класс `Block` является базовой для всех блоков документа. От нее наследуются классы `Paragraph`, `Table`, `Shape`, `Field` (см. Рисунок 27).

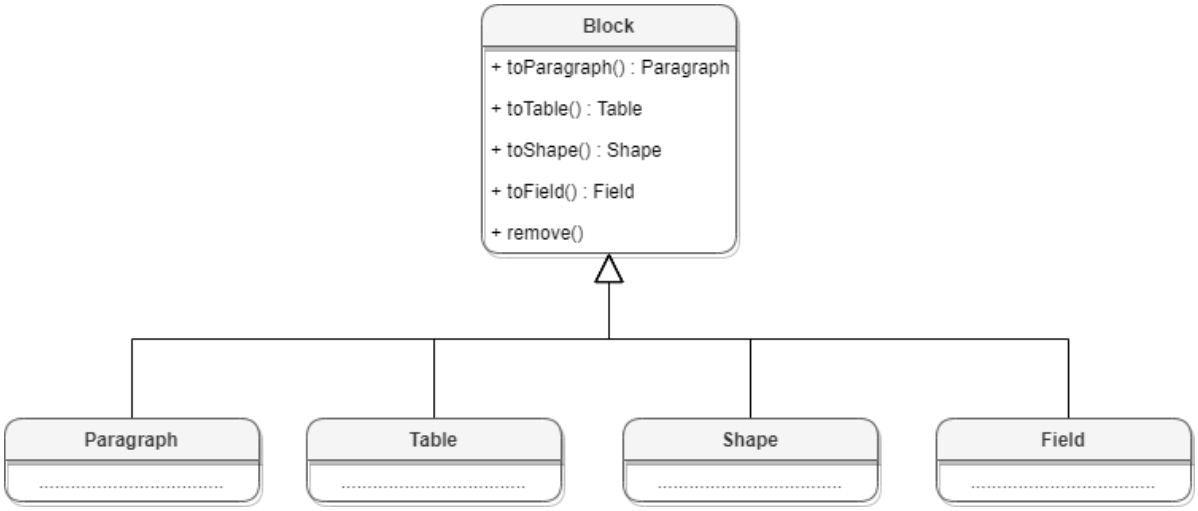


Рисунок 27 – Объектная модель класса `Block`

6.5.1 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());
}
```

6.5.2 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());
}
```

6.5.3 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();
}
```

6.5.4 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());
}
```

6.6 Класс Blocks

Класс `Blocks` обеспечивает доступ к блокам `Block` документа или диапазона документа (см. Рисунок 28). Объект класса `Blocks` может быть получен вызовом метода `Document.getBlocks` или `HeaderFooter.getBlocks`.

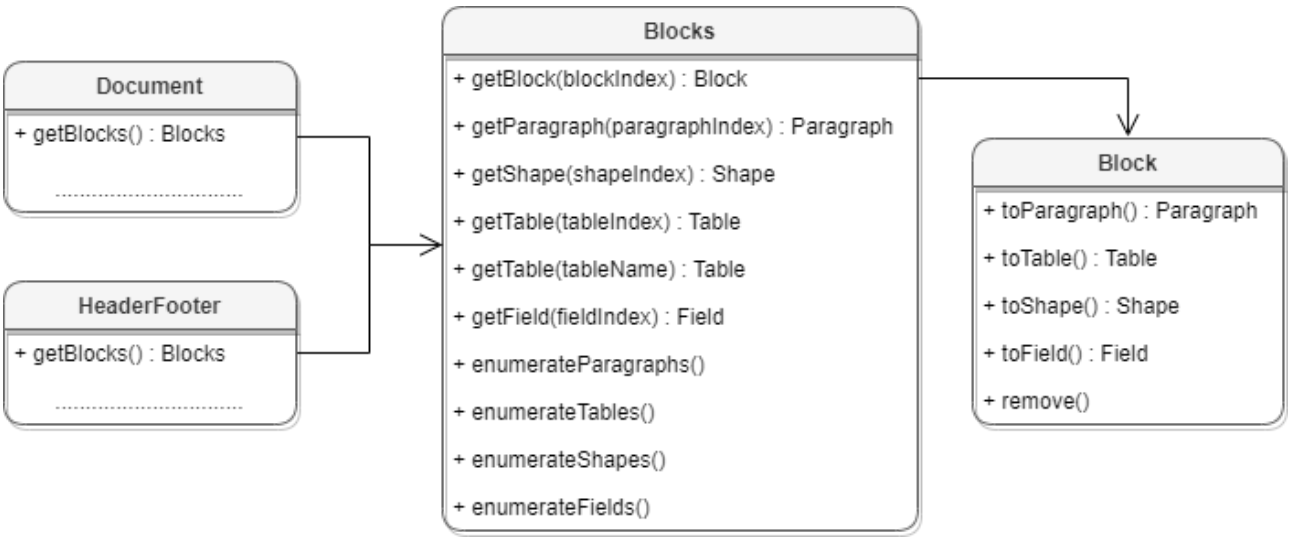


Рисунок 28 – Объектная модель класса `Blocks`

6.6.1 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");
}
```

6.6.2 Метод Blocks.GetEnumerator

Позволяет реализовать перечисление объектов `Block`.

Пример:

```
BlocksEnumerator blocksEnumerator = document.getBlocks().GetEnumerator();
foreach (var block in blocksEnumerator)
{
    Console.WriteLine(block.getRange().extractText());
}
```

6.6.3 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");
}
```

6.6.4 Метод Blocks.getFieldsEnumerator

Позволяет перечислить объекты типа `Field`.

Пример:

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

6.6.5 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");
}
```

6.6.6 Метод Blocks.getParagraphsEnumerator

Позволяет реализовать перечисление абзацев `Paragraph`.

Пример:

```
ParagraphsEnumerator paragraphsEnumerator =
document.getBlocks().getParagraphsEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
```

```
    Console.WriteLine(paragraph.getRange().extractText());
}
```

6.6.7 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");
}
```

6.6.8 Метод Blocks.getShapesEnumerator

Позволяет перечислить объекты типа Shape.

Пример:

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

6.6.9 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("Sheet1");
```

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

6.6.10 Метод Blocks.getTablesEnumerator

Позволяет перечислить объекты типа `Table`.

Пример:

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

6.7 Класс Bookmarks

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

6.7.1 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());
}
```

6.7.2 Метод Bookmarks.removeBookmark

Удаляет закладку по ее названию.

Пример:

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

6.8 Класс Borders

Класс `Borders` предназначен для оформления границ ячейки таблицы. Список методов приведен в таблице 3. В качестве аргумента используется тип `LineProperties`, который содержит такие параметры линии, как тип, толщина, цвет.

Таблица 3 – 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
`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
`Borders getLeft(LineProperties lp)`	Получение левой границы ячейки
`Borders getRight(LineProperties lp)`	Получение правой границы ячейки
`Borders getTop(LineProperties lp)`	Получение верхней границы ячейки
`Borders getBottom(LineProperties lp)`	Получение нижней границы ячейки
`Borders getDiagonalDown(LineProperties lp)`	Получение диагональной линии
`Borders getDiagonalUp(LineProperties lp)`	Получение диагональной линии
`Borders getInnerHorizontal(LineProperties lp)`	Получение внутренних горизонтальных границ ячейки
`Borders getInnerVertical(LineProperties lp)`	Получение внутренних вертикальных границ 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 = borders.setLeft(lineProperties);
borders = borders.setTop(lineProperties);
borders = borders.setRight(lineProperties);
borders = borders.setBottom(lineProperties);
cell.setBorders(borders);
```

6.9 Класс CalculationMode

Режимы пересчета формул в документе представлены в таблице 4. Класс `CalculationMode` используется в методах `Document.getCalculationMode()` и `Document.setCalculationMode()`.

Таблица 4 – Режимы пересчета формул

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

6.10 Класс CaseSensitive

Параметры настройки учета регистра при поиске (см. метод `Search.FindText()`) представлены в таблице 5.

Таблица 5 – Параметры регистра при поиске

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

6.11 Класс Cell

Класс `Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 29).

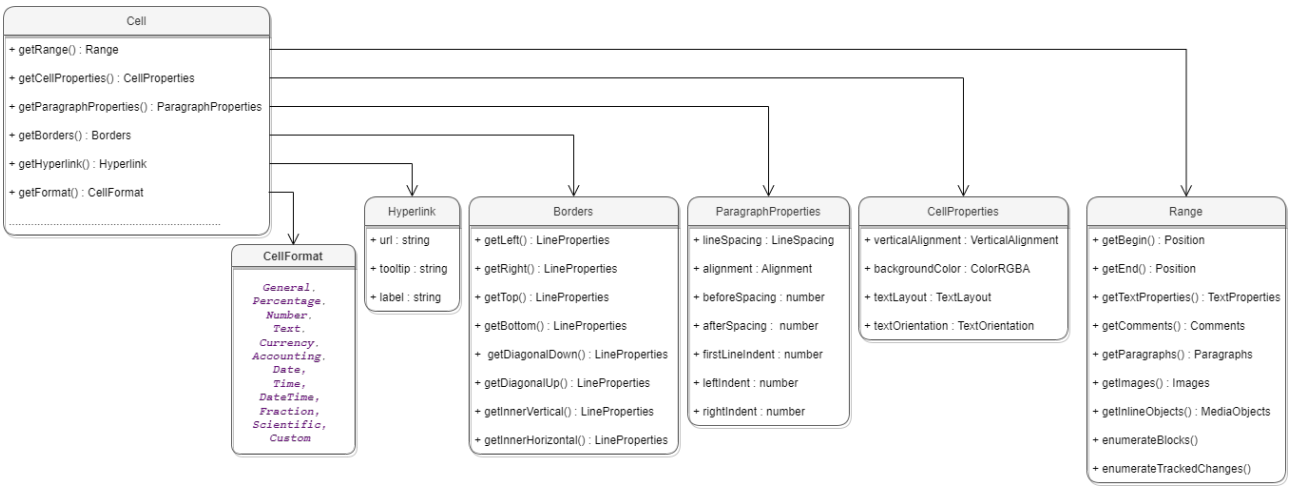


Рисунок 29 – Объектная модель ячейки таблиц

6.11.1 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());
```

6.11.2 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);
```

6.11.3 Метод Cell.getCustomFormat

Возвращает строку формата ячейки.

Пример:

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

6.11.4 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());
```

6.11.5 Метод Cell.getFormattedValue

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

Пример:

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

6.11.6 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());
```

6.11.7 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();
```

6.11.8 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);
```

6.11.9 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());
}
```

6.11.10 Метод Cell.getProtectionProperties

Метод возвращает параметры защиты ячейки 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);
```

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

6.11.11 Метод Cell.getRange

Метод возвращает объект `Range` для управления содержимым ячейки.

6.11.12 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());
```

6.11.13 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());
```

6.11.14 Метод Cell.isProtected

Метод возвращает статус защиты от редактирования ячейки в табличном документе.

Вызов:

```
public bool isProtected()
```

Методы `setProtectionProperties()` и `getProtectionProperties()` позволяют задать и получить параметры защиты ячейки (`CellProtectionProperties`).

6.11.15 Метод Cell.setBool

Устанавливает для ячейки значение логического типа.

Пример:

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

6.11.16 Метод Cell.setBorders

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

6.11.17 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);
```

6.11.18 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");
```

6.11.19 Метод Cell.setCustomFormat

Устанавливает формат ячейки.

Пример:

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

6.11.20 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
```

6.11.21 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());
```

Внимание! Данный метод использует текстовый параметр в формате unicode string. Если символы данной строки отличаются от латиницы и кодируются двумя байтами (например, кириллица), то необходимо принять дополнительные меры, описанные в разделе Использование кодировок. В противном случае возникнет исключение `DocumentAPI.UnknownError` с сообщением «Invalid UTF-8».

6.11.22 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)");
```

6.11.23 Метод Cell.setNumber

Устанавливает для ячейки значение числового типа.

Пример:

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

6.11.24 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);
```

6.11.25 Метод Cell.setProtectionProperties

Метод задает параметры защиты ячейки в табличном документе.

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

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

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

6.11.26 Метод Cell.setText

Устанавливает для ячейки значение строкового типа.

Пример:

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

6.11.27 Метод Cell.unmerge

Разъединяет несколько ячеек, которые были объединены ранее.

Пример:

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

6.12 Класс CellFormat

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

Таблица 6 – Поддерживаемые форматы ячеек таблицы

Наименование константы	Описание
`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».

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

Использование данных констант позволяет установить выбранный формат. При этом будет 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.

6.13 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`.

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);
```

6.13.1 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;
```

6.13.2 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;
```

6.13.3 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());
```

6.14 Класс CellProperties

Класс `CellProperties` предназначен для форматирования содержимого в ячейках таблицы. Описание полей представлено в таблице 7.

Для задания свойств ячейки используется метод `Cell.setCellProperties()`. Для получения свойств ячейки используется метод `Cell.getCellProperties()`. Иерархия классов и полей для работы с `CellProperties` отображена на рисунке 30.

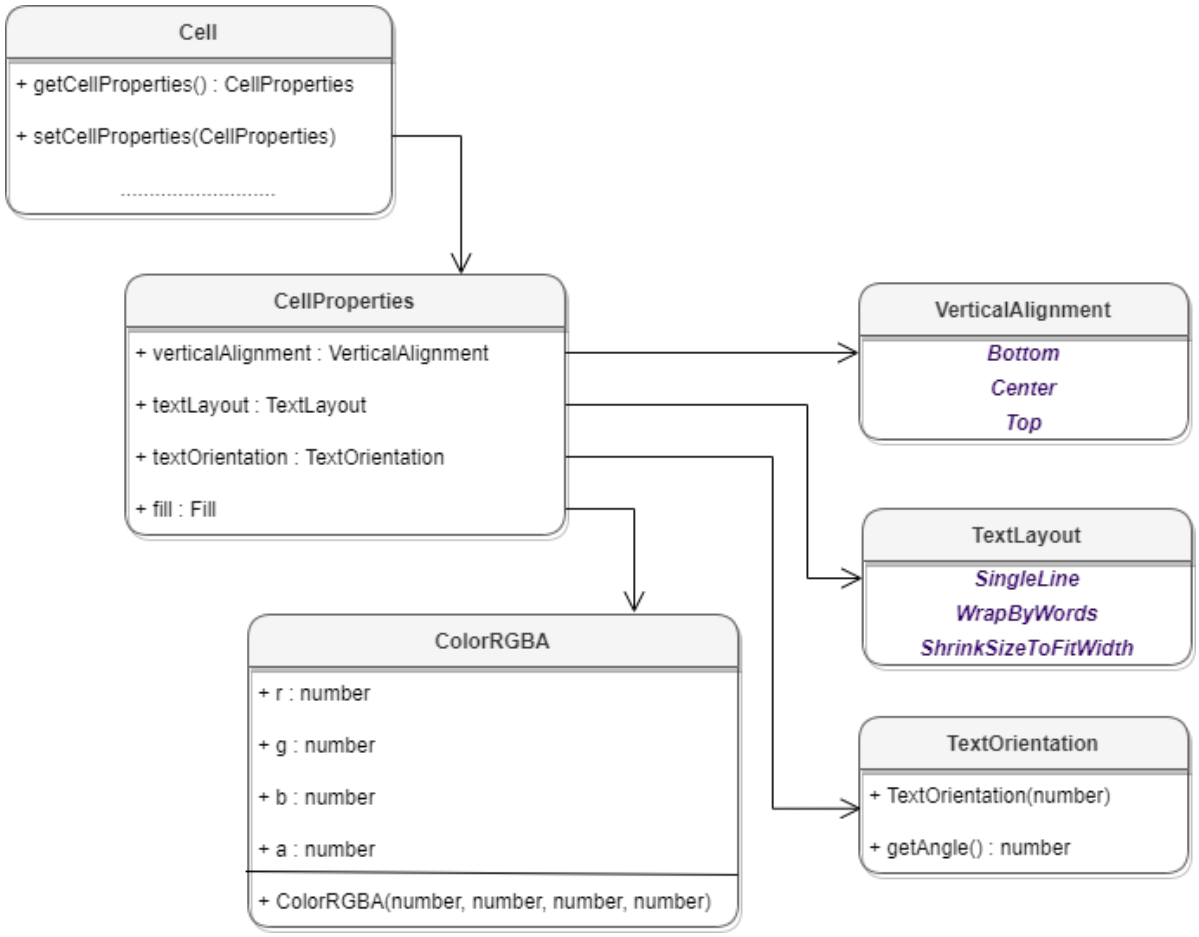


Рисунок 30 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 7 – Описание полей класса `CellProperties`

Поле	Тип	Значение
`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 RGBColorRGBAA(255, 255, 0, 255)));
cellProps.textOrientation = new TextOrientation(45);

cell.setCellProperties(cellProps);
```

6.15 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 8 – 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);
```

6.16 Класс CellRange

Класс `CellRange` описывает диапазон ячеек таблицы.

Пример:

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

6.16.1 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));
```

6.16.2 Метод CellRange.containsCell

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

Примеры:

```
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.

6.16.3 Метод CellRange.copyInto

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

Данный метод реализован только в табличных документах.

Пример (только для табличного документа):

```
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);
```

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

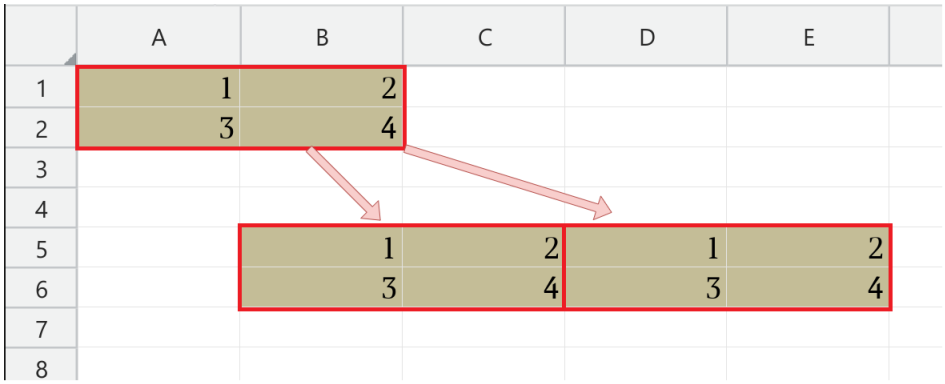


Рисунок 31 – Копирование ячеек табличного документа

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

6.16.4 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());
```

6.16.5 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());
```

6.16.6 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);
```

6.16.7 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());
}
```

6.16.8 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());
```

6.16.9 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());
```

6.16.10 Метод CellRange.getProtectionProperties

Метод возвращает параметры защиты диапазона ячеек 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. Метод `setProtectionProperties()` позволяет задать параметры защиты диапазона ячеек. Вы также можете использовать метод `isProtected()`, чтобы узнать, защищены ли ячейки диапазона от редактирования.

6.16.11 Метод CellRange.getTable

Возвращает таблицу `Table`, содержащую текущий диапазон.

6.16.12 Метод CellRange.getTableRange

Возвращает положение текущего диапазона ячеек в таблице (объект `CellRangePosition`).

6.16.13 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);
```

6.16.14 Метод CellRange.isProtected

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

Вызов:

```
public bool? isProtected()
```

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

6.16.15 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();
```

6.16.16 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.

6.16.17 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);
```

6.16.18 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);
```

6.16.19 Метод CellRange.setProtectionProperties

Метод задает параметры защиты диапазона ячеек в табличном документе.

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

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

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

6.17 Класс CellRangePosition

Класс `CellRangePosition` представляет положение диапазона ячеек в таблице. Используется в качестве поля `tableRange` таблицы `TableRangeInfo`, а также в методах `Table.getCellRange()`, `Chart.setRange()`. По умолчанию диапазон включает одну ячейку в позиции 0,0 что соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.

Описание полей класса `CellRangePosition` представлено в таблице 9.

Таблица 9 – Поля класса `CellRangePosition`

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

bottomRight	CellPosition	Содержит позицию правой нижней ячейки таблицы прямоугольного диапазона.

Примеры:

```
Table table = document.getBlocks().getTable(0);
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
CellRange range = table.getCellRange(cellRangePosition);
```

```
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("top left row:" + tableRange.topLeft.row + ", top left
column:" + tableRange.topLeft.column);
```

6.17.1 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)];
```

6.18 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. 32.

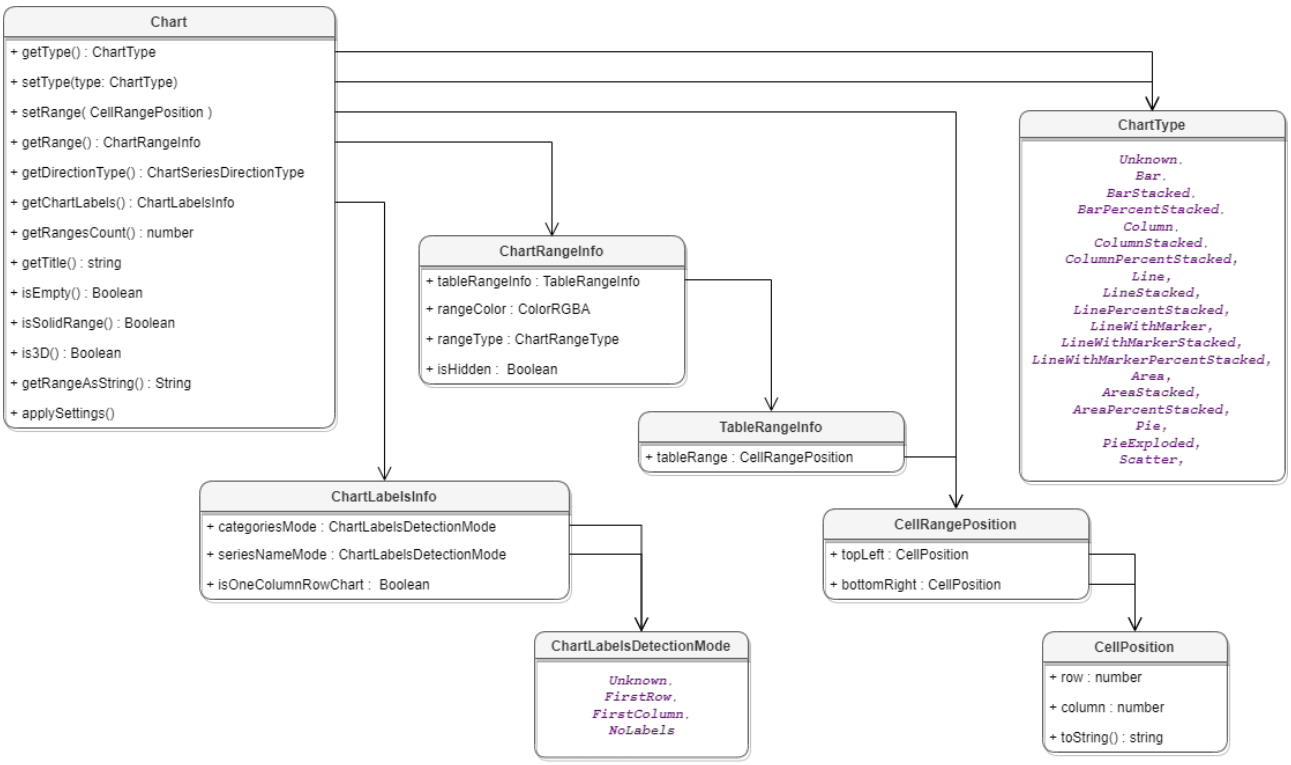


Рисунок 32 – Объектная модель класса `Chart`

6.18.1 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);
```

6.18.2 Метод Chart.getChartLabels

Метод возвращает коллекцию меток диаграммы типа ChartLabelsInfo.

Пример:

```
Chart chart = charts.getChart(0);
ChartLabelsInfo chartlabelsInfo = chart.getChartLabels();
Console.WriteLine(chartlabelsInfo.getType());
```

6.18.3 Метод Chart.getDirectionType

Метод возвращает направление ChartSeriesDirectionType серий диаграммы.

Пример:

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.getDirectionType());
```

6.18.4 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);
}
```

6.18.5 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);
    }
}
```

6.18.6 Метод Chart.getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

Пример:

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.getRangeAsString());
```

6.18.7 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());
}
```

6.18.8 Метод Chart.getTitle

Метод возвращает заголовок диаграммы.

Пример:

```
Chart chart = charts.getChart(0);
String title = chart.getTitle();
if (title != null) {
    Console.WriteLine(chart.getTitle());
}
```

6.18.9 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());
    }
}
```

6.18.10 Метод Chart.is3D

Метод возвращает `true`, если диаграмма трехмерная.

Пример:

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.is3D());
```

6.18.11 Метод Chart.isEmpty

Метод возвращает `true`, если диаграмма не содержит значений.

Пример:

```
Chart chart = charts.getChart(0);
Console.WriteLine(chart.isEmpty());
```

6.18.12 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());
```

6.18.13 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());
```

6.18.14 Метод Chart.setRect

Метод задает область расположения диаграммы.

Внимание ! 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));
```

6.18.15 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());
}
```

6.19 Класс ChartLabelsDetectionMode

Класс описывает режимы автоматического определения меток диаграмм.

Поля класса соответствуют следующим режимам автоматического определения меток диаграмм:

- `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);
```

6.20 Класс ChartLabelsInfo

Класс `ChartLabelsInfo` описывает настройки автоматического определения меток диаграммы. Инициализируется конструктором:

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

Параметры конструктора:

- `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.

Описание полей класса `ChartLabelsInfo` представлено в таблице 10.

Таблица 10 – Описание полей класса `ChartLabelsInfo`

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

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);
```

6.21 Класс ChartRangeInfo

Класс `ChartRangeInfo` описывает серию диаграммы. Инициализируется конструктором:

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

Параметры конструктора:

– `tableRangeInfo` - диапазон ячеек, тип `TableRangeInfo`;

– `color` – цвет серии диаграммы, тип `ColorRGBA`;

– `hidden` – видимость серии, тип Boolean;

– `rangeType` – тип диапазона исходных данных диаграммы, тип `ChartRangeType`.

Описание полей класса представлено в таблице 11.

Таблица 11 – Описание полей класса `ChartRangeInfo`

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

Пример:

```
Chart chart = charts.getChart(0);
ChartRangeInfo chartRangeInfo = chart.getRange(0);
Console.WriteLine(chartRangeInfo.rangeColor);
Console.WriteLine(chartRangeInfo.rangeType);
```

6.22 Класс ChartRangeType

Класс описывает тип диапазона исходных данных диаграммы.

Возможные значения:

– `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);
```

6.23 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());
}
```

6.23.1 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());
    }
}
```

6.23.2 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));
}
```

6.23.3 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());
}
```

6.24 Класс ChartSeriesDirectionType

Класс описывает направление серий диаграмм.

Возможные значения:

- `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);
```

6.25 Класс ChartType

Перечисление `ChartType` описывает все поддерживаемые типы диаграмм.

Поля класса соответствуют следующим типам диаграмм:

- `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.

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);
}
```

6.26 Класс CheckBoxControl

Представляет собой флаговую кнопку (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());
```

6.27 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);
```

6.27.1 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);
}
```

6.27.2 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);
}
```

6.27.3 Метод Color.getTransforms

Метод возвращает текущую трансформацию цвета `ColorTransforms`.

Пример:

```
var color = new Color(new ColorRGBA(255, 0, 0, 255));
var transforms = color.getTransforms();
```

6.27.4 Метод Color.setTransforms

Метод устанавливает трансформацию цвета `ColorTransforms`.

Пример:

```
var color = new Color(new ColorRGBA(255, 0, 0, 255));
var colorTransforms = new ColorTransforms();
colorTransforms.apply(new ColorRGBA(55, 146, 179, 200));
color.setTransforms(colorTransforms);
```

6.28 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`).

Для создания нового объекта используется один из конструкторов:

```
ColorRGBA()
ColorRGBA(byte r, byte g, byte b, byte a)
```

Описание полей класса `ColorRGBA` представлено в таблице 12.

Таблица 12 – Описание полей класса `ColorRGBA`

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

6.29 Класс ColorTransforms

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

Пример:

```
var color = new Color(new ColorRGBA(255, 0, 0, 255));
var colorTransforms = new ColorTransforms();
colorTransforms.apply(new ColorRGBA(55, 146, 179, 200));
color.setTransforms(colorTransforms);
var transforms = color.getTransforms();
```

6.29.1 Метод ColorTransforms.apply

Метод устанавливает цвет трансформации `ColorRGBA`.

Пример:

```
var colorTransforms = new ColorTransforms();
colorTransforms.apply(new ColorRGBA(55, 146, 179, 200));
```

6.30 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`.

6.30.1 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);
    }
}
```

6.30.2 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());
    }
}
```

6.30.3 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());
        }
    }
}
```

6.30.4 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());
    }
}
```

6.30.5 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());
    }
}
```

6.31 Класс Comments

Класс `Comments` содержит коллекцию комментариев диапазона (см. Рисунок 33).

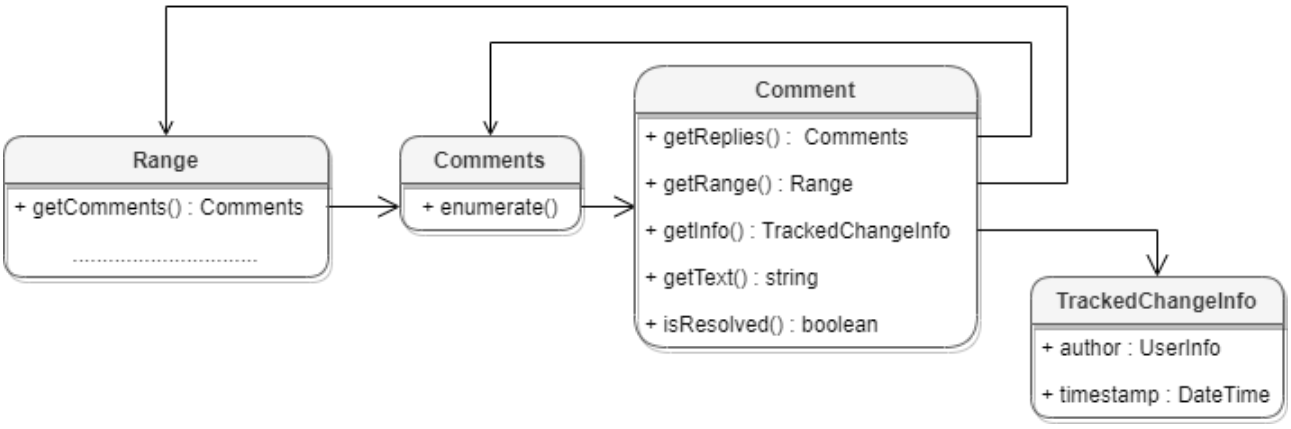


Рисунок 33 – Объектная модель классов для работы с комментариями

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

Пример:

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

6.31.1 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);
}
```

6.32 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.

6.32.1 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);
```

6.32.2 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.

```
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);
```

Пример:

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

Пример использования приведен в разделе Работа с фильтрами.

6.33 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();
```

6.34 Класс ContentControl

Базовый класс для элементов управления (см. Работа с элементами управления).

Наследники:

- `CheckBoxControl`
- `DatePickerControl`
- `DropListControl`
- `InputFieldControl`

6.34.1 Метод ContentControl.canEdit

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

6.34.2 Метод ContentControl.getTitle

Метод возвращает название элемента управления.

6.34.3 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();
```

6.35 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");
```

6.36 Класс CurrencyCellFormatting

Класс содержит параметры для денежного формата ячеек таблицы. Описание полей класса `CurrencyCellFormatting` представлено в таблице 13.

Таблица 13 – Описание полей класса `CurrencyCellFormatting`

Поле	Описание
`CurrencyCellFormatting.decimalPlaces`	Количество десятичных позиций
`CurrencyCellFormatting.symbol`	Символ денежной единицы
`CurrencyCellFormatting.localeCode`	Идентификатор кода языка (MS-LCID)
`CurrencyCellFormatting.useRedForNegative`	Использовать красный цвет для отрицательных значений
`CurrencyCellFormatting.useBracketsForNegative`	Использовать скобки для отрицательных значений
`CurrencyCellFormatting.hideSign`	Скрывать знак «минус» для отрицательных значений
`CurrencyCellFormatting.useThousandsSeparator`	Использовать разделитель для тысячных
`CurrencyCellFormatting.currencySignPlacement`	Варианты размещения знака валюты `CurrencySignPlacement`

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());
```

6.37 Класс CurrencySignPlacement

Класс `CurrencySignPlacement` .определяет варианты размещения знака валюты до значения ($12.00), либо после (12.00 Р). Используется в поле `LocaleInfo.currencyFormat`.

Описание полей таблицы `CurrencySignPlacement` представлено в таблице 14.

Таблица 14 – Описание вариантов расположения знаков валюты

Поле	Описание
`CurrencySignPlacement.Prefix`	Символ валюты перед значением
`CurrencySignPlacement.Suffix`	Символ валюты после значения

6.38 Класс DatePatterns

Форматы даты представлены в таблице 15. Пример использования см. в главе `DateTimeCellFormatting`.

Таблица 15 – Форматы даты

Наименование константы	Описание
`DatePatterns.DayMonthTextLongYearLong`	`'mmmm dd, yyyy'` для языка `en_US`
`DatePatterns.FullDate`	`'день недели, mmmm dd, yyyy'` для языка `en_US`

Наименование константы	Описание
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'

6.39 Класс DatePickerControl

Представляет собой поле с календарем, используемое 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);
```

6.40 Класс DateTime

Класс DateTime предоставляет дату и время с точностью до секунды. Используется для поля TrackedChangeInfo.timeStamp. Описание полей класса DateTime представлено в таблице 16.

Таблица 16 – Описание полей класса DateTime

Поле	Тип	Описание
DateTime.year	Дата	Год
DateTime.month	Дата	Месяц
DateTime.day	Дата	День

Поле	Тип	Описание
DateTime.hour	Время	Часы
DateTime.minute	Время	Минуты
DateTime.second	Время	Секунды

6.41 Класс DateTimeCellFormatting

Класс содержит параметры для формата ячеек таблицы типа Дата и Время, используется в качестве аргумента метода Cell.setFormat(). Описание полей класса DateTimeCellFormatting представлено в таблице 17.

Таблица 17 – Описание полей класса DateTimeCellFormatting

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

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());
```

6.42 Класс DateTimeFormat

В таблице 18 представлены варианты масштабирования при печати табличных документов. Используется в качестве параметра метода CellRange.insertCurrentDateTime().

Таблица 18 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
DateTimeFormat.DateTime	Дата/время

Наименование константы	Описание
`DateTimeFormat.Date`	Дата
`DateTimeFormat.Time`	Время

6.43 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);
    ..........................................
}
```

6.43.1 Метод Document.areMirroredMarginsEnabled

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

Пример:

```
Console.WriteLine(document.areMirroredMarginsEnabled());
```

6.43.2 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.

6.43.3 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.

6.43.4 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.

Внимание! Данный метод использует текстовый параметр в формате unicode string. Если символы данной строки отличаются от латиницы и кодируются двумя байтами (например, кириллица), то необходимо принять дополнительные меры, описанные в разделе Использование кодировок. В противном случае возникнет исключение `DocumentAPI.UnknownError` с сообщением «Invalid UTF-8».

6.43.5 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.

6.43.6 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());
    }
}
```

6.43.7 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());
}
```

6.43.8 Метод Document.getCalculationMode

Метод возвращает текущий режим пересчета формул в документе `CalculationMode`. Чтобы изменить этот режим, используйте метод `Document.setCalculationMode()`.

6.43.9 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());
}
```

6.43.10 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());
```

6.43.11 Метод Document.getFormulaType

Метод возвращает поддерживаемую адресацию ячеек FormulaType документа.

Пример:

```
FormulaType formulaType = document.getFormulaType();
```

6.43.12 Метод Document.getNamedExpressions

Используется для получения списка именованных диапазонов NamedExpressions.

Пример:

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

6.43.13 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) {
    ..........................
}
```

6.43.14 Метод Document.getRange

Метод предоставляет доступ ко всему диапазону Range документа.

Пример:

```
Range range = document.getRange();
if (range != null) {
    Console.WriteLine(range.extractText());
}
```

6.43.15 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());
}
```

6.43.16 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());
}
```

6.43.17 Метод Document.getSectionsEnumerator

Позволяет перечислить секции документа, тип Section.

Пример:

```
SectionsEnumerator sectionsEnumerator = document.getSectionsEnumerator();
foreach (var section in sectionsEnumerator)
{
```

```
    Console.WriteLine(section.getRange().extractText());
}
```

6.43.18 Метод Document.isCalculatedOnSave

Метод возвращает состояние функции пересчета формул при сохранении документа. Используйте метод `Document.setCalculatedOnSave()`, чтобы включить или отключить эту функцию.

6.43.19 Метод Document.isChangesTrackingEnabled

Метод возвращает текущее состояние отслеживания изменений в документе (`true` - включены).

Пример:

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

6.43.20 Метод Document.isStructureProtected

Метод возвращает состояние защиты от изменения структуры табличного документа.

Вызов:

```
    public bool isStructureProtected()
```

Методы `setStructureProtection()` и `removeStructureProtection()` позволяют установить и снять защиту от изменений структуры.

6.43.21 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 34.

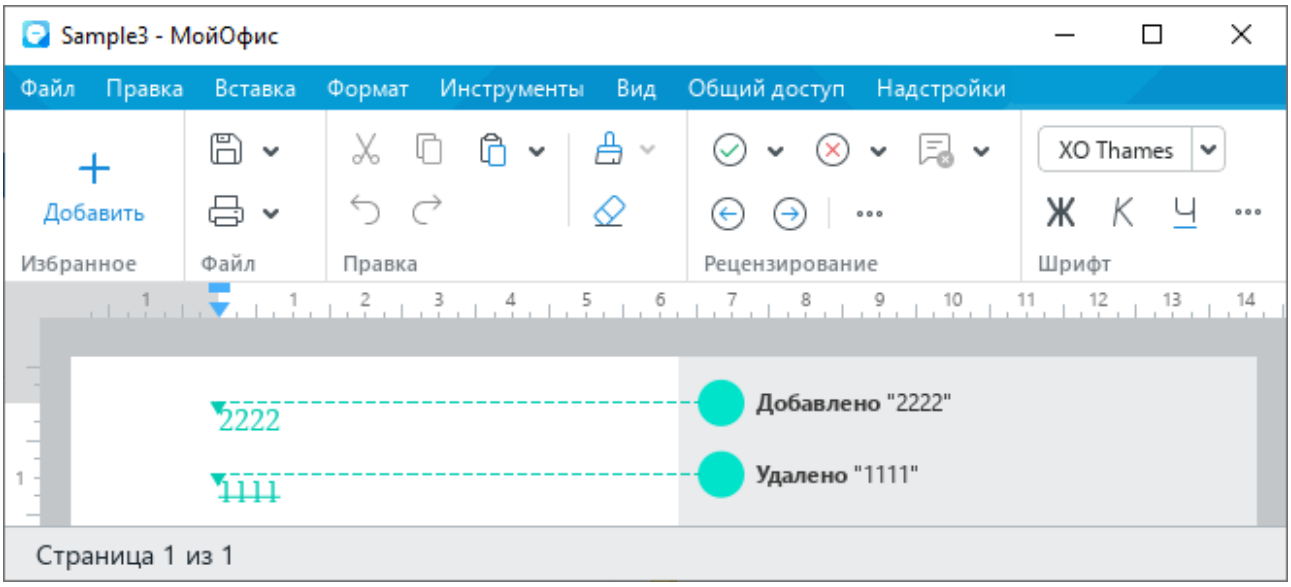


Рисунок 34 – Результат выполнения метода `merge`

6.43.22 Метод Document.removeStructureProtection

Метод снимает защиту от изменений структуры табличного документа.

Вызов:

```
public void removeStructureProtection(string password)
```

Параметры:

– `password`: (необязательный) пароль для снятия защиты, тип `string`.

Пример:

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

Метод `setStructureProtection()` позволяет установить защиту от изменений структуры. Вы также можете использовать метод `isStructureProtected()`, чтобы узнать текущее состояние защиты. Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `DocumentAPI.IncorrectPasswordError`.

6.43.23 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.

Внимание! Данный метод использует текстовый параметр в формате unicode string. Если символы данной строки отличаются от латиницы и кодируются двумя байтами (например, кириллица), то необходимо принять дополнительные меры, описанные в разделе Использование кодировок. В противном случае возникнет исключение `DocumentAPI.UnknownError` с сообщением «Invalid UTF-8».

6.43.24 Метод Document.setCalculatedOnSave

Метод позволяет включить и отключить функцию пересчета формул при сохранении документа. Используйте метод `Document.isCalculatedOnSave()`, чтобы узнать текущее состояние этой функции.

6.43.25 Метод Document.setCalculationMode

Метод позволяет задать режим пересчета формул в документе `CalculationMode`. Используйте метод `Document.getCalculationMode()`, чтобы получить текущий режим пересчета.

6.43.26 Метод Document.setChangesTrackingEnabled

Метод управляет состоянием отслеживания изменений в документе (включены или выключены).

Пример:

```
document.setChangesTrackingEnabled(true);
```

6.43.27 Метод Document.setFormulaType

Метод устанавливает поддерживаемую адресацию ячеек `FormulaType` документа.

Пример:

```
document.setFormulaType(FormulaType.A1);
```

6.43.28 Метод Document.setMirroredMarginsEnabled

Метод позволяет включать и выключать зеркальные поля в документе.

Пример:

```
document.setMirroredMarginsEnabled(true);
```

6.43.29 Метод Document.setPageOrientation

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

Пример:

```
document.setPageOrientation(PageOrientation.Landscape);
```

6.43.30 Метод Document.setPageProperties

Метод устанавливает свойство PageProperties в документе.

Пример:

```
PageProperties pageProperties = new PageProperties();
pageProperties.width = 100;
pageProperties.height = 200;
document.setPageProperties(pageProperties);
```

6.43.31 Метод Document.setStructureProtection

Метод устанавливает защиту от изменений структуры табличного документа.

Вызов:

```
public void setStructureProtection(string password)
```

Параметры:

– `password`: (необязательный) пароль для установки защиты, тип `string`.

Пример:

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

Метод `removeStructureProtection()` позволяет снять защиту от изменений структуры. Вы также можете использовать метод `isStructureProtected()`, чтобы узнать текущее состояние защиты. Если метод `setStructureProtection()` применяется к документу с уже защищенной структурой, возникает исключение `DocumentAPI.SpreadsheetProtectionError`.

6.44 Класс DocumentFormat

В таблице 19 приведены поддерживаемые форматы документов, которые используются в поле `documentFormat` класса `SaveDocumentSettings`.

Таблица 19 – Форматы документов

Константа	Описание
`PlainText`	Используется для работы с файлами TXT.
`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.
`OXML`	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML.
`ODF`	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010).
`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.
`PDF`	Используется для работы с документами в формате Portable Document Format (PDF) версии 1.4.
`PDFA`	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

6.45 Класс DocumentSettings

Класс `DocumentSettings` предоставляет общие настройки документа и используется в `Application.createDocument`.

Описание полей класса `DocumentSettings` представлено в таблице 20.

Таблица 20 – 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

6.46 Класс DocumentType

В таблице 21 приведены поддерживаемые типы документов, которые используются при создании документа `Application.createDocument`, `DocumentSettings`.

Таблица 21 - Типы документов

Наименование константы	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.

6.47 Класс DropListControl

Представляет собой поле с выпадающим списком 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"));
```

6.48 Класс DSVSettings

Класс `DSVSettings` предоставляет настройки, необходимые для работы с файлами CSV (comma-separated value) и DSV (delimiter-separated value). Используется в `SaveDocumentSettings`, `LoadDocumentSettings`.

Описание полей класса `DSVSettings` представлено в таблице 22.

Таблица 22 – Описание полей класса `DSVSettings`

Поле	Описание
`DSVSettings.autofit`	Признак необходимости автоматического подстраивания ширины столбца под размер данных в ячейке
`DSVSettings.startBlockIndex`	Индекс блока документа сохранения

Поле	Описание
DSVSettings.lastBlockIndex	Индекс блока документа для окончания сохранения

Пример:

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

6.49 Класс Encoding

В таблице 23 приведены поддерживаемые кодировки документов. Используется в LoadDocumentSettings.

Таблица 23 - Кодировки документов

Наименование константы	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

6.50 Класс ExportFormat

В таблице 24 приведены поддерживаемые форматы экспорта документов (см. Document.exportAs).

Таблица 24 - Форматы экспорта документов

Константа	Описание
PDFA1	Используется для работы с документами в формате Portable Document Format (PDF).

6.51 Класс Field

Класс `Field` предназначен для реализации некоторых полей, например, содержания (см. класс `Block`).

6.52 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");
```

6.52.1 Метод Fill.getColor

Метод возвращает цвет заполнения `Color`.

6.52.2 Метод Fill.getUrl

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

6.52.3 Метод Fill.isNoFill

Метод возвращает `true`, если заполнения нет.

6.53 Класс FiltersRange

Класс `FiltersRange` реализует диапазон таблицы, позволяющий манипулировать фильтрами столбцов. Пример использования приведен в разделе Работа с фильтрами.

6.53.1 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();
```

6.53.2 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();
```

6.53.3 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.

6.53.4 Метод FiltersRange.setFilters

Метод `FiltersRange.setFilters` устанавливает фильтры `TableFilters` для

столбцов диапазона. Фильтрация выполняется с использованием логической операции AND по столбцам. Нумерация столбцов начинается относительно левой позиции диапазона фильтрации. Если номер столбца в фильтрах превышает диапазон фильтрации, то фильтр будет успешно добавлен, но фильтрация для этого столбца будет пропущена. Такое поведение полезно, если необходимо изменить размер диапазона фильтрации при этом сохранить ранее определенные фильтры. Диапазон фильтрации должен существовать до вызова этого метода. В настоящее время DocumentAPI позволяет создавать диапазон фильтрации только для всего листа табличного документа. Для создания или изменения размера существующего диапазона используется метод `Table.createFiltersRange()`.

Вид метода `FiltersRange.setFilters`:

```
void setFilters(const TableFilters& filters);
```

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.

6.54 Класс FormulaType

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 25. Используется в `Document.getFormulaType()`, `Document.setFormulaType()`, `DocumentSettings`.

Таблица 25 – Системы адресации ячеек в табличном документе

Константа	Система адресации ячеек	Описание
`A1`	A1	Вариант A1 соответствует наиболее распространенной системе адресации ячеек, при которой столбцы задаются буквами, а строки – числами.

Константа	Система адресации ячеек	Описание
R1C1	R1C1	Вариант R1C1 соответствует альтернативной системе адресации ячеек, при которой столбцы и строки задаются числами.

6.55 Класс FractionCellFormatting

Класс содержит параметры для дробного формата ячеек таблицы. Используется в качестве аргумента метода Cell.setFormat(). Описание полей класса FractionCellFormatting представлено в таблице 26.

Таблица 26 – Описание полей класса FractionCellFormatting

Поле	Описание
FractionCellFormatting.minNumeratorDigits	Количество позиций числителя
FractionCellFormatting.minDenominatorDigits	Количество позиций знаменателя
FractionCellFormatting denominatorValue	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());
```

6.56 Класс Frame

Класс `Frame` описывает прямоугольную область графического объекта документа. Предназначен для получения и изменения свойств графических объектов. Для расположения в позиции текста документа используется `InlineFrame`, в таблице - `AbsoluteFrame` (см. Рисунок 35).

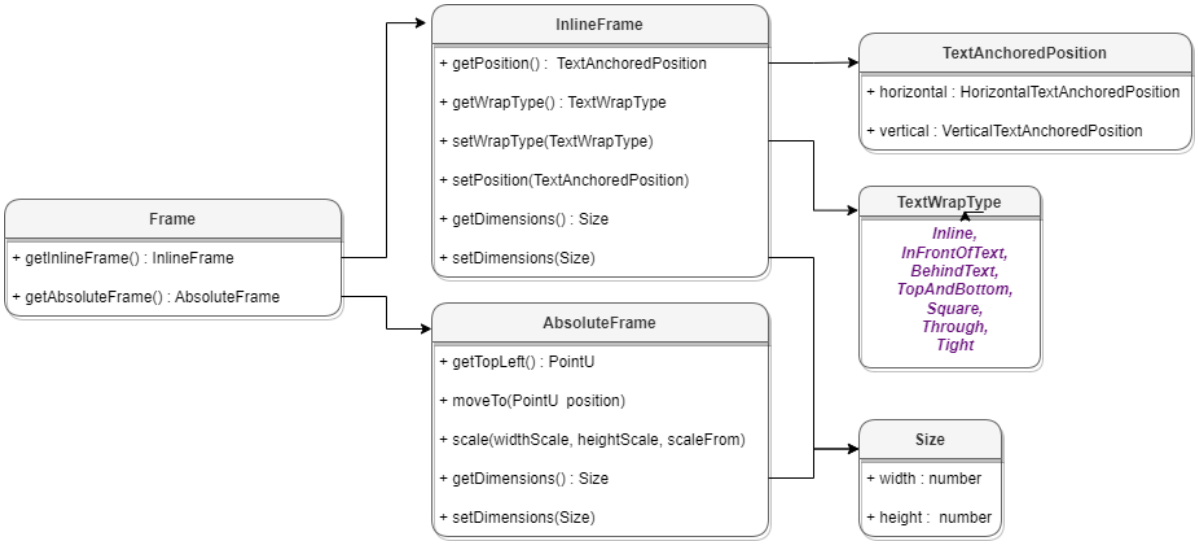


Рисунок 35 – Объектная модель класса `Frame`

Пример:

```
MediaObjects 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());
    }
}
```

6.56.1 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);
    }
}
```

6.56.2 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());
    }
}
```

6.57 Класс FrozenRangePosition

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

6.57.1 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());
```

6.57.2 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());
```

6.57.3 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());
```

6.57.4 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());
```

6.57.5 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());
```

6.57.6 Метод FrozenRangePosition.isArea

Возвращает `true` если диапазон является непрерывной областью.

Пример:

```
frozenRangePosition = FrozenRangePosition.createFrozenArea(2, 2);
Console.WriteLine(frozenRangePosition.isArea());
```

6.57.7 Метод FrozenRangePosition.isCols

Возвращает `true` если диапазон состоит из колонок.

Пример:

```
frozenRangePosition = FrozenRangePosition.createFrozenCols(0, 2);
Console.WriteLine(frozenRangePosition.isCols());
```

6.57.8 Метод FrozenRangePosition.isRows

Возвращает `true` если диапазон состоит из строк.

Пример:

```
frozenRangePosition = DocumentAPI.FrozenRangePosition.createFrozenRows(0, 2);
Console.WriteLine(frozenRangePosition.isRows());
```

6.57.9 Метод FrozenRangePosition.isRowsCols

Возвращает `true` если диапазон содержит строки и колонки.

Пример:

```
frozenRangePosition = FrozenRangePosition.createFrozenArea(2, 2);
Console.WriteLine(frozenRangePosition.isRowsCols());
```

6.57.10 Оператор ==

Метод используется для определения эквивалентности двух объектов `FrozenRangePosition`.

6.57.11 Оператор !=

Метод используется для определения неэквивалентности двух объектов `FrozenRangePosition`.

6.58 Класс HeaderFooter

Класс `HeaderFooter` определяет колонтитул текстового документа.

6.58.1 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());
    }
}
```

6.58.2 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());
}
```

6.58.3 Метод HeaderFooter.getType

Метод предоставляет информацию о типе колонтитула (HeaderFooterType).

Пример:

```
foreach (var headerFooter in headersEnumerator)
{
    HeaderFooterType headerFooterType = headerFooter.getType();
    Console.WriteLine(headerFooterType);
}
```

6.59 Класс HeaderFooterType

Класс HeaderFooterType описывает типы колонтитулов – верхний колонтитул HeaderFooterType.Header и нижний колонтитул HeaderFooterType.Footer.

6.60 Класс HeadersFooters

Класс HeadersFooters представляет коллекцию верхних и нижних колонтитулов раздела (см. Рисунок 36). Доступ к колонтитулам осуществляется посредством методов Section.getHeaders(), Section.getFooters().

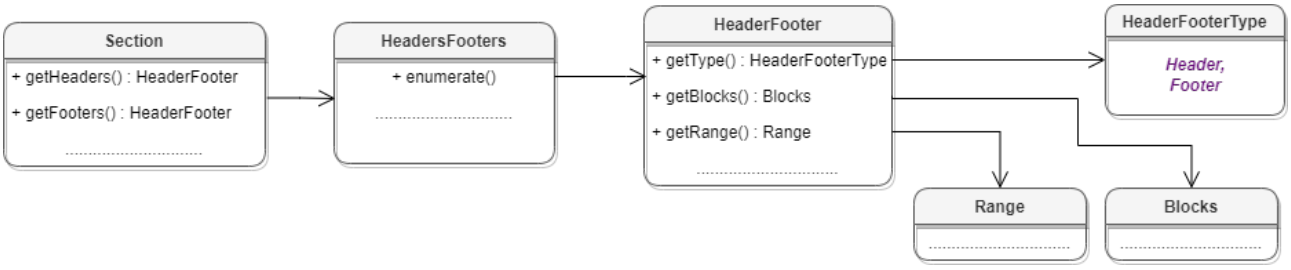


Рисунок 36 – Классы для работы с колонтитулами

6.60.1 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());
    }
}
```

6.61 Класс HorizontalAnchorAlignment

В таблице 27 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали (см. описание класса `HorizontalTextAnchoredPosition`).

Таблица 27 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

Наименование константы	Описание
`HorizontalAnchorAlignment.Left`	По верхнему краю
`HorizontalAnchorAlignment.Right`	По нижнему краю
`HorizontalAnchorAlignment.Center`	По центру
`HorizontalAnchorAlignment.Inside`, `HorizontalAnchorAlignment.Outside`	По границам

6.62 Класс HorizontalRelativeTo

В таблице 28 представлены типы размещения объекта относительно закрепленной позиции по горизонтали (см. описание класса `HorizontalTextAnchoredPosition`).

Таблица 28 – Типы размещения объекта относительно закрепленной позиции по горизонтали

Наименование константы	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
`HorizontalRelativeTo.PageRightMargin`	Правое поле страницы
`HorizontalRelativeTo.PageInsideMargin`	Внутреннее поле страницы

Наименование константы	Описание
`HorizontalRelativeTo.PageOutsideMargin`	Внешнее поле страницы

6.63 Класс HorizontalTextAnchoredPosition

Класс `HorizontalTextAnchoredPosition` предназначен для управления относительным положением объекта со смещением или выравниванием по горизонтали (см. описание класса `TextAnchoredPosition`).

Описание полей класса `HorizontalTextAnchoredPosition` представлено в таблице 29.

Таблица 29 – Описание полей класса `HorizontalTextAnchoredPosition`

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

6.64 Класс Hyperlink

Класс `Hyperlink` описывает свойства ссылки. Объект `Hyperlink` может быть получен посредством вызова метода `Cell.getHyperlink()`.

Таблица 30 – Описание полей класса `Hyperlink`

Поле	Тип	Описание
`Hyperlink.url`	Строка	Адрес ссылки
`Hyperlink.tooltip`	Строка	Текст подсказки
`Hyperlink.label`	Строка	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}");
```

6.64.1 Оператор ==

Метод используется для определения эквивалентности двух объектов `Hyperlink`.

6.64.2 Оператор !=

Метод используется для определения неэквивалентности двух объектов `Hyperlink`.

6.65 Класс Image

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

6.65.1 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());
}
```

6.65.2 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;
    }
}
```

6.66 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
}
```

6.66.1 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
}
```

6.67 Класс InlineFrame

Класс `InlineFrame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 37). Предназначен для получения и изменения свойств позиции графических объектов. Используется в текстовом документе. Может быть получен с помощью метода `Frame.getInlineFrame()`.

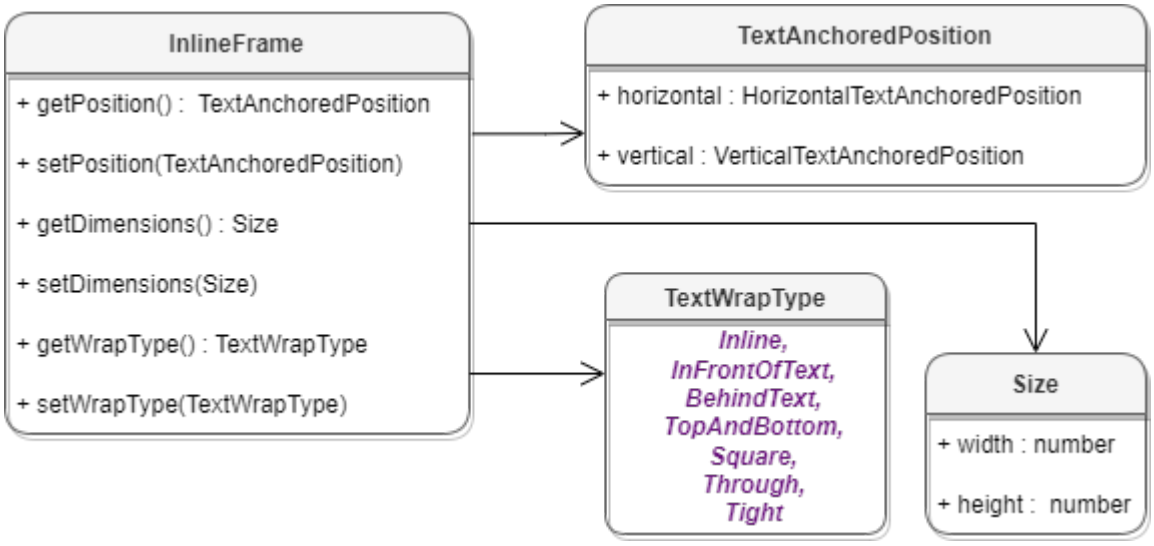


Рисунок 37 – Объектная модель класса `InlineFrame`

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);
    }
}
```

6.67.1 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);
    }
}
```

6.67.2 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);
    }
}
```

6.67.3 Метод InlineFrame.getWrapType

Метод возвращает вариант обтекания текстом встроенного объекта (см. TextWrapType).

Пример:

```
Console.WriteLine(inlineFrame.getWrapType());
```

6.67.4 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);
    }
}
```

6.67.5 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);
```

6.67.6 Метод InlineFrame.setWrapType

Метод устанавливает вариант обтекания текстом встроенного объекта (см. TextWrapType).

Пример:

```
inlineFrame.setWrapType(TextWrapType.Inline);
Console.WriteLine(inlineFrame.getWrapType());
```

6.68 Класс InputFieldControl

Представляет собой текстовое поле 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(' ', '_'));
```

6.69 Класс Insets

Класс `Insets` предназначен для задания полей, например, страницы. Поля класса `Insets` представлены в таблице 31. Используется в поле `margins` класса `PageProperties`.

Таблица 31 – Описание полей класса `Insets`

Поле	Тип	Описание
`left`	`int`	Левая граница поля
`top`	`int`	Верхняя граница поля
`right`	`int`	Правая граница поля
`bottom`	`int`	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);
```

6.70 Класс LineEndingProperties

Класс `LineEndingProperties` содержит варианты оформления окончаний линий. Описание полей класса `LineEndingProperties` представлено в таблице 32. Используется в полях `headLineEndingProperties` и `tailLineEndingProperties` класса `LineProperties`.

Таблица 32 – Описание полей класса `LineEndingProperties`

Поле	Тип	Описание
`LineEndingProperties.style`	`LineEndingStyle`	Стиль окончания линии

Поле	Тип	Описание
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);
```

6.71 Класс LineEndingStyle

В таблице 33 приведены типы окончания линии. Используется в поле `style` класса LineEndingProperties.

Таблица 33 – Типы окончания линии

Наименование константы	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);
```

6.72 Класс LineProperties

Класс `LineProperties` предназначен для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 38).

Поля класса:

- `style` - тип линии, допустимые значения представлены в разделе `LineStyle`;
- `width` - толщина линии;
- `color` - цвет линии, тип - `Color`;
- `headLineEndingProperties` - тип начала линии, тип - `LineEndingProperties`;
- `tailLineEndingProperties` - тип окончания линии, тип - `LineEndingProperties`.

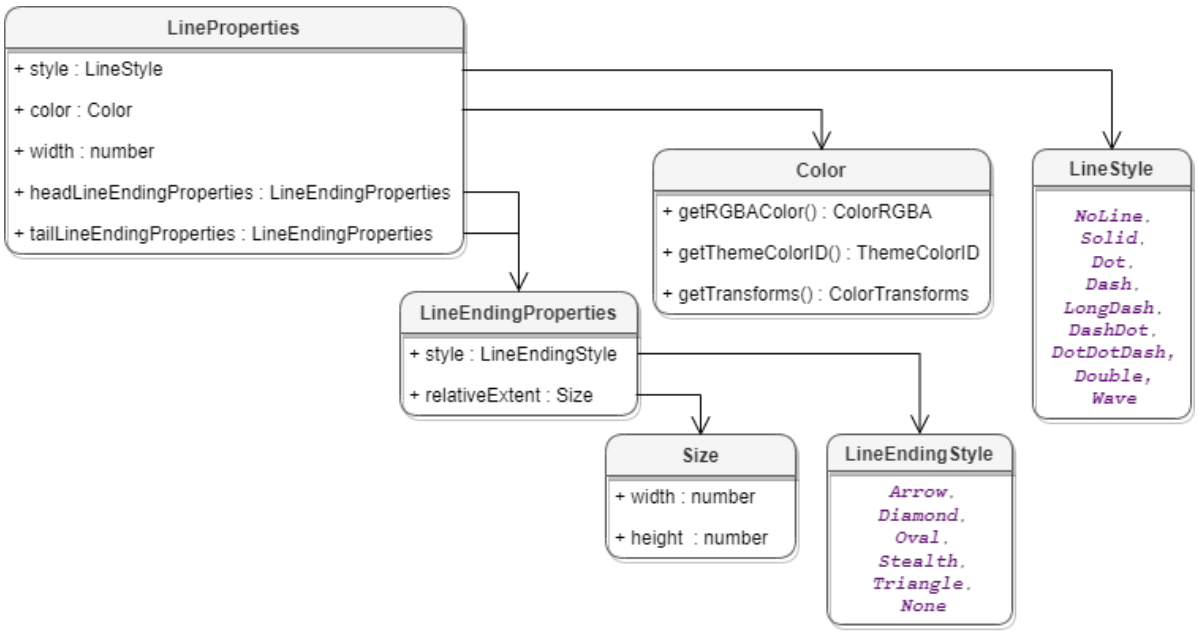


Рисунок 38 – Свойства границ ячеек

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);
```

6.73 Класс LineSpacing

Класс LineSpacing задает межстрочный интервал абзаца. Поля класса приведены в таблице 34. Для управления значением межстрочного интервала используются значения, представленные в разделе `LineSpacingRule`.

Таблица 34 – Параметры межстрочного интервала

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

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;
```

6.74 Класс LineSpacingRule

Класс `LineSpacingRule` содержит типы межстрочного интервалов.

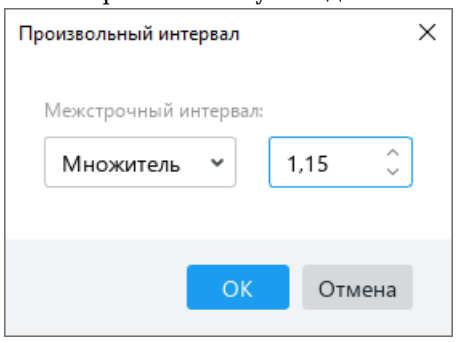
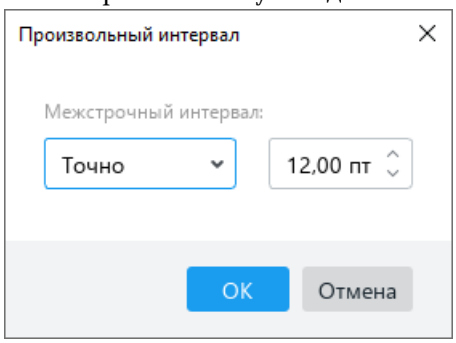
Типы межстрочных интервалов:

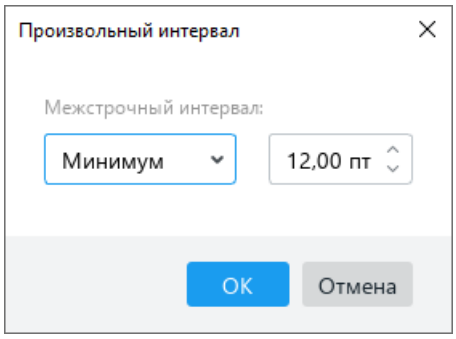
- `Multiple` – межстрочный интервал с использованием множителя;
- `Exact` – межстрочный интервал с использованием точного значения;
- `AtLeast` – межстрочный интервал с использованием минимального значения.

В таблице 35 представлены описания правил формирования межстрочного интервала текстового абзаца.

Таблица 35 – Виды межстрочного интервала

Наименование константы	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)`

Наименование константы	Описание
	В данном примере используется значение множителя **1.15**. Действие команды аналогично ручной настройке межстрочного интервала в диалоге «**Произвольный интервал**» (см. документ «МойОфис Текст. Руководство пользователя»). 
`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** (см. документ «МойОфис Текст. Руководство пользователя»). 
`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**» (см. документ «МойОфис Текст. Руководство пользователя»).

Наименование константы	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);
}
```

6.75 Класс LineStyle

В таблице 36 приведены типы линий. Используется в поле `style` класса `LineProperties`.

Таблица 36 – Типы линий

Наименование константы	Описание
`LineStyle.NoLine`	Нет линии
`LineStyle.Solid`	
`LineStyle.Dot`	
`LineStyle.Dash`	
`LineStyle.LongDash`	
`LineStyle.DashDot`	
`LineStyle.DotDotDash`	

Наименование константы	Описание
LineStyle.Double	
LineStyle.Wave	

Пример:

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

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Solid;
```

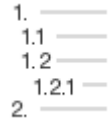
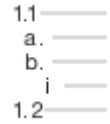
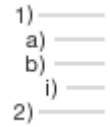
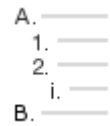
6.76 Класс ListSchema

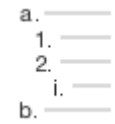
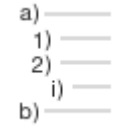
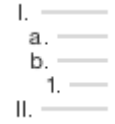
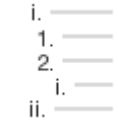
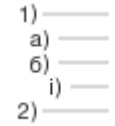
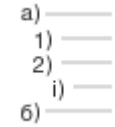
Класс `ListSchema` содержит типы схем форматирования списков, которые могут быть применены к абзацам текста. Данные константы используются в методах `Paragraph.getListSchema()`, `Paragraph.setListSchema()`.

Описания схем форматирования списков представлены в таблице 37.

Таблица 37 – Описания схем списков

Тип схемы списка	Описание типа схемы списка	Изображение
ListSchema.Unknown	Неизвестно	
ListSchema.UnknownBullet	Список без маркера	Соответствует варианту «нет»
ListSchema.UnknownNumbering	Нумерация без номера	Соответствует варианту «нет»
ListSchema.BulletCircleSolid	Список с маркерами в виде круга	
ListSchema.BulletCircleContour	Список с маркерами в виде окружности	

Тип схемы списка	Описание типа схемы списка	Изображение
`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	
`ListSchema.EnumeratorLatinUppercaseDot`	Нумерация латинскими прописными буквами с точкой	

Тип схемы списка	Описание типа схемы списка	Изображение
`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	

Пример:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.setListSchema(ListSchema.BulletCheckmark);
    Console.WriteLine(paragraph.getListSchema());
}
```

6.77 Класс LoadDocumentSettings

Класс `LoadDocumentSettings` предоставляет дополнительные настройки, необходимые для загрузки документов из файла (см. `Application.loadDocument`).

Описание полей класса `LoadDocumentSettings` представлено в таблице 38.

Таблица 38 – Описание полей класса LoadDocumentSettings

Поле	Описание
LoadDocumentSettings.commonDocumentSettings	Общие настройки документа DocumentSettings.
LoadDocumentSettings.encoding	Кодировка документа Encoding.
LoadDocumentSettings.dsvSettings	DSVSettings, настройки для работы с файлами CSV и DSV .
LoadDocumentSettings.documentPassword	Пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows.

6.78 Класс LocaleInfo

Класс LocaleInfo предоставляет информацию о локализации. Используется в поле localeInfo класса DocumentSettings.

Описание полей LocaleInfo представлено в таблице 39.

Таблица 39 – Описание полей класса LocaleInfo

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

Поле	Описание
`LocaleInfo.longDatePattern`	Заданный «длинный» формат отображения даты в текущем регионе (например, 'dddd, mmmm d, yyyy' для en_US).
`LocaleInfo.timePattern`	Заданный формат отображения времени в текущем регионе (например, 'h:mm AM/PM' для en_US).

6.79 Класс MediaObject

Класс `MediaObject` представляет собой встроенный объект документа. Может быть получен посредством использования метода `MediaObjects.getEnumerator()`.

6.79.1 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();
    ..................................
}
```

6.79.2 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");
    }
}
```

6.79.3 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");
```

```
    }
}
```

6.80 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 39).

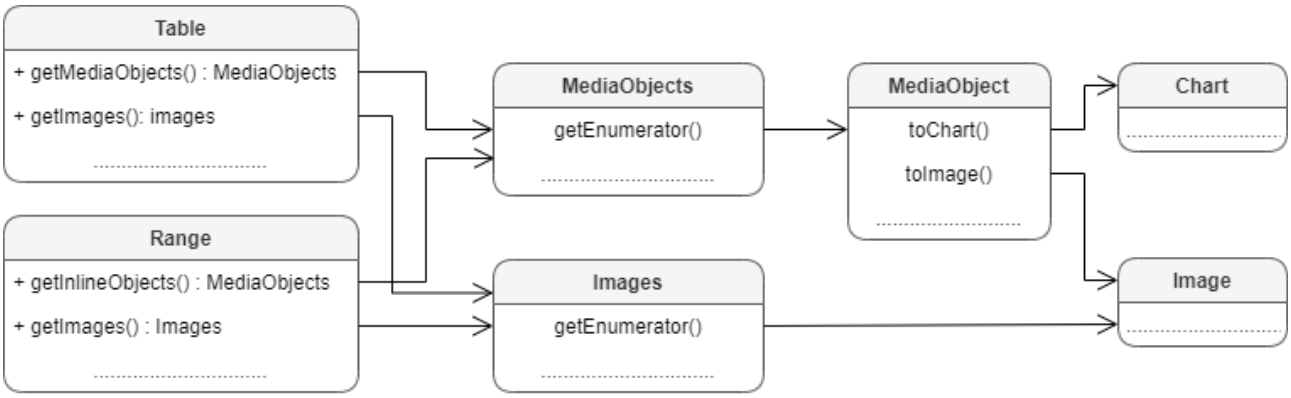


Рисунок 39 – Встроенные объекты

6.80.1 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)
{
    .....................................
}
```

6.81 Класс Message

Класс `Message` предназначен для формирования событий лога.

6.81.1 Класс Message.Severity

Класс `Message.Severity` (Таблица 40) описывает уровни сообщений лога (информация, предупреждение, ошибка).

Таблица 40 – Описание уровней лога `Message.Severity`

Поле	Описание
`Message.Severity.Info`	Информация
`Message.Severity.Warning`	Предупреждение
`Message.Severity.Error`	Ошибка

6.81.2 Метод Message.getSeverity

Метод возвращает уровень лога `Message.Severity`.

6.81.3 Метод Message.getText

Метод возвращает текст сообщения.

6.81.4 Метод Message.makeError

Метод создает сообщение типа `Message.Severity.Error` с заданным текстом.

6.81.5 Метод Message.makeInfo

Метод создает сообщение типа `Message.Severity.Info` с заданным текстом.

6.81.6 Метод Message.makeWarning

Метод создает сообщение типа `Message.Severity.Warning` с заданным текстом.

6.82 Класс Messenger

Служит для организации логов приложений.

6.82.1 Метод Messenger.notify

Метод используется для создания события лога

Пример:

```
MessageHandler messageHandler = new MessageHandler();
Messenger messenger= application.getMessenger();
messenger.notify(Message.makeWarning("Warning"));
```

6.82.2 Метод Messenger.subscribe

Метод служит для подписки на события лога.

Пример:

```
MessageHandler messageHandler = new MessageHandler();
Messenger messenger= application.getMessenger();
messenger.subscribe(messageHandler);
```

6.83 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());
}
```

6.83.1 Метод NamedExpression.getCellRange

Возвращает диапазон ячеек `CellRange`. Пример см. в `NamedExpression`.

6.83.2 Метод NamedExpression.getExpression

Возвращает текст именованного диапазона (формулы). Пример см. в `NamedExpression`.

6.83.3 Метод NamedExpression.getName

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

6.84 Класс NamedExpressions

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

6.84.1 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);
```

6.84.2 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());
}
```

6.84.3 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());
}
```

6.84.4 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);
```

6.85 Класс NumberCellFormatting

Класс содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода `Cell.setFormat()`. Описание полей класса `NumberCellFormatting` представлено в таблице 41.

Таблица 41 – Описание полей класса `NumberCellFormatting`

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

6.86 Класс PageFieldOrder

Класс `PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем класса `PivotTableLayoutSettings`. Описание полей класса представлено в таблице 42.

Таблица 42 – Описание полей класса `PageFieldOrder`

Поле	Описание
`PageFieldOrder.DownThenOver`	Вниз, затем поперек
`PageFieldOrder.OverThenDown`	Поперек, затем вниз

6.87 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
VectorUInt pages = new VectorUInt(3);
pages[0] = 1;
pages[1] = 13;
pages[2] = 25;
PageNumbers pageNumbers = new PageNumbers(pages);
```

```
// Диапазон страниц
PageNumbers pageNumbers = new PageNumbers(1, 20);
```

6.87.1 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));
```

6.87.2 Метод PageNumbers.getLast

Метод `PageNumbers.getLast` возвращает последний номер страницы.

Пример:

```
PageNumbers pageNumbers = new PageNumbers(1, 20);
Console.WriteLine(pageNumbers.getLast());
```

6.88 Класс PageOrientation

Тип `PageOrientation` определяет варианты ориентации страницы документа: Альбомная `PageOrientation.Landscape` или Книжная `PageOrientation.Portrait`. Может быть использована для получения / установки ориентации страниц для секции или документа в методах `Section.setPageOrientation`, `Section.getPageOrientation`.

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());
}
```

6.89 Класс PageParity

Варианты выбора страниц для экспорта и печати представлены в таблице 43. Используется в `PageNumbers`.

Таблица 43 – Варианты выбора страниц для экспорта и печати

Наименование константы	Описание
`PageParity.Odd`	Только нечетные страницы

Наименование константы	Описание
`PageParity.Even`	Только четные страницы
`PageParity.All`	Все страницы

6.90 Класс PageProperties

Класс `PageProperties` предоставляет такие свойства страницы как высота, ширина, размеры полей. Описание полей приведено в таблице 44. Используется в `Document.setPageProperties()`, `Section.getPageProperties()`, `Section.setPageProperties()`.

Таблица 44 – Описание полей класса `PageProperties`

Поле	Описание
`height`	Высота страницы
`width`	Ширина страницы
`margins`	Поля страницы, тип - `Insets`

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);
```

6.91 Класс Paragraph

Класс `Paragraph` предоставляет доступ к свойствам абзаца (см. Рисунок 40).

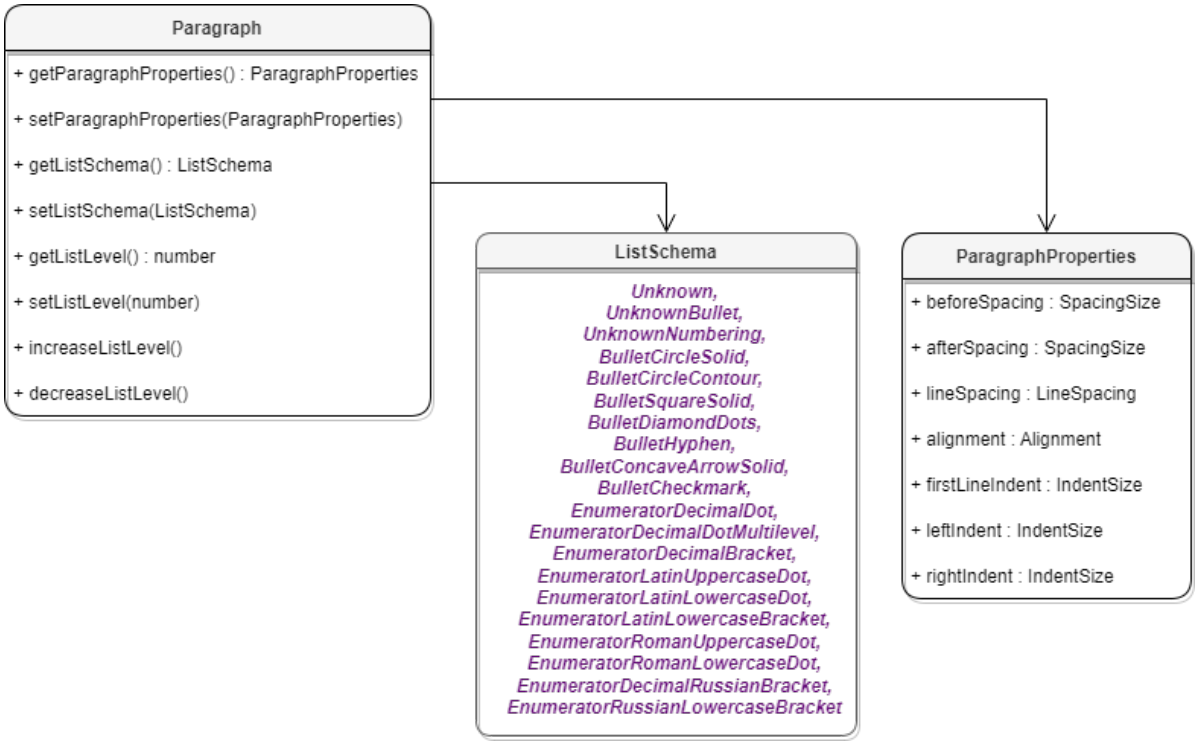


Рисунок 40 – Объектная модель классов для работы со свойствами параграфа

6.91.1 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());
}
```

6.91.2 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());
}
```

6.91.3 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());
}
```

6.91.4 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);
}
```

6.91.5 Метод Paragraph.increaseListLevel

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

Пример:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.increaseListLevel();
    Console.WriteLine(paragraph.getListLevel());
}
```

6.91.6 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());
}
```

6.91.7 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());
}
```

6.91.8 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);
}
```

6.92 Класс ParagraphProperties

Класс `ParagraphProperties` предназначен для управления свойствами форматирования (см. Рисунок 41). Класс `ParagraphProperties` используется в методах `Paragraph.getParagraphProperties()` и `Paragraph.setParagraphProperties()`.

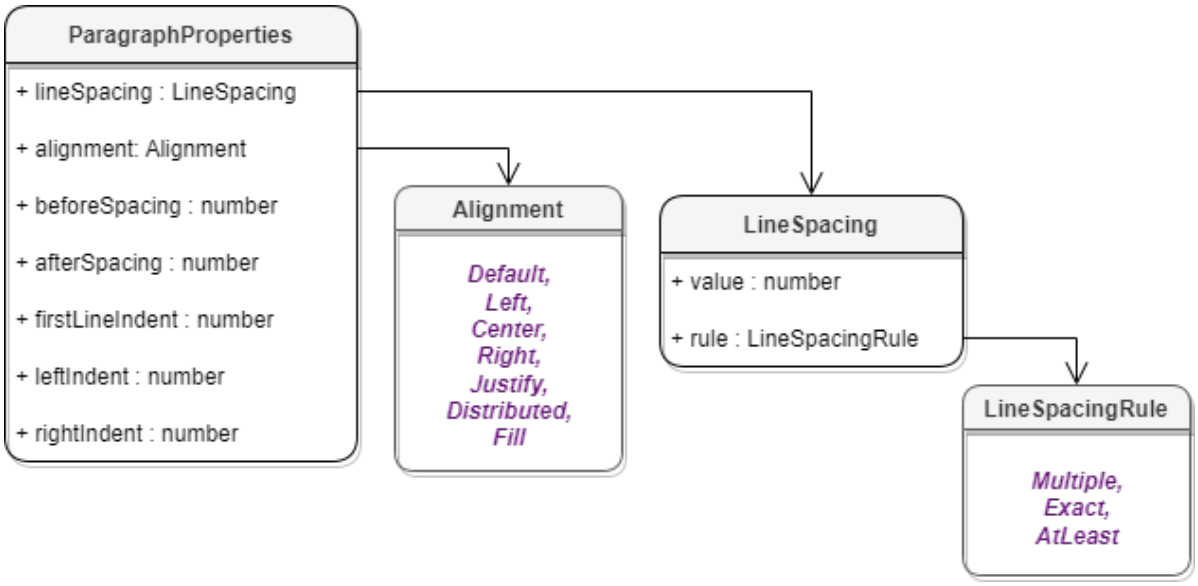
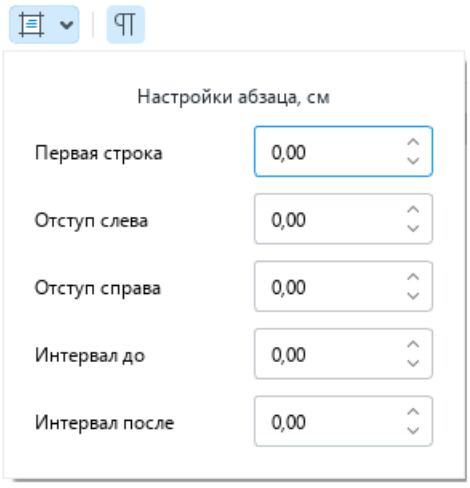


Рисунок 41 – Объектная модель классов для работы со свойствами параграфа

Описание полей класса `ParagraphProperties` представлено в таблице 45.

Таблица 45 – Описание полей класса `ParagraphProperties`

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

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

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);
}
```

6.93 Класс Paragraphs

Класс `Paragraphs` предоставляет доступ к коллекции абзацев типа `Paragraph` (см. Рисунок 42). Коллекция абзацев может быть получена из объекта `Range` посредством использования метода `Range.getParagraphs()`.

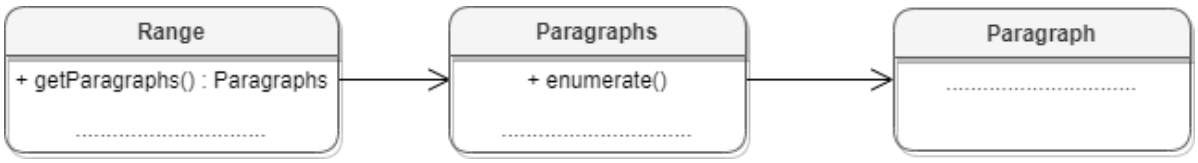


Рисунок 42 – Объектная модель для работы со списком абзацев

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();
```

6.93.1 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();
```

6.93.2 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());
}
```

6.93.3 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();
```

6.93.4 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);
```

6.93.5 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);
```

6.94 Класс PercentageCellFormatting

Содержит параметр для процентного формата ячеек таблицы, используется в качестве аргумента метода `Cell.setFormat()`. Описание полей класса `PercentageCellFormatting` представлено в таблице 46.

Таблица 46 – Описание полей класса `PercentageCellFormatting`

Поле	Описание
`PercentageCellFormatting.decimalPlaces`	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());
```

6.95 Класс PivotTable

Класс для представления сводной таблицы. Может быть получен из ячейки `Cell.getPivotTable()`, либо при создании новой сводной таблицы `PivotTablesManager.create()`.

6.95.1 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());
```

6.95.2 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();
```

6.95.3 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);
}
```

6.95.4 Метод PivotTable.getFieldCategories

Метод возвращает список категорий PivotTableFieldCategories, содержащих заданное поле fieldName.

Пример:

```
PivotTableFieldCategories categories = pivotTable.getFieldCategories("Age");
PivotTableFieldCategorysEnumerator enumerator = categories.GetEnumerator();
foreach (var PivotTableFieldCategory in enumerator)
{
    PivotTableFieldCategory pivotTableFieldCategory = enumerator.Current;
    Console.WriteLine(pivotTableFieldCategory);
}
```

6.95.5 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());
}
```

6.95.6 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());
}
```

6.95.7 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);
}
```

6.95.8 Метод PivotTable.getFilter

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

Пример:

```
PivotTableFilter pivotTableFilter = pivotTable.getFilter("Age");
Console.WriteLine(pivotTableFilter.getFieldName());
```

6.95.9 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));
}
```

6.95.10 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);
}
```

6.95.11 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());
```

6.95.12 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);
```

6.95.13 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);
```

6.95.14 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);
}
```

6.95.15 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());
}
```

6.95.16 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();
}
```

6.95.17 Метод PivotTable.getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства PivotTableUnsupportedFeature сводной таблицы.

Пример:

```
PivotTableUnsupportedFeatures unsupportedFeatures =
pivotTable.getUnsupportedFeatures();
PivotTableUnsupportedFeatures.PivotTableUnsupportedFeaturesEnumerator enumerator
= unsupportedFeatures.GetEnumerator();
while (enumerator.MoveNext()) {
    Console.WriteLine(enumerator.Current);
}
```

6.95.18 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);
}
```

6.95.19 Метод PivotTable.isColumnGrandTotalEnabled

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

Пример:

```
Console.WriteLine(pivotTable.isColumnGrandTotalEnabled());
```

6.95.20 Метод PivotTable.isRowGrandTotalEnabled

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

Пример:

```
Console.WriteLine(pivotTable.isRowGrandTotalEnabled());
```

6.95.21 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();
}
```

6.95.22 Метод PivotTable.update

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

Пример:

```
PivotTableUpdateResult updateResult = pivotTable.update();
if (updateResult == PivotTableUpdateResult.FieldAlreadyEnabled) {
    ............
}
```

6.96 Класс PivotTableCaptions

Класс `PivotTableCaptions` хранит все пользовательские заголовки сводной таблицы (см. `PivotTable.getPivotTableCaptions()`). Описание полей таблицы представлено в таблице 47.

Таблица 47 – Описание полей класса `PivotTableCaptions`

Поле	Описание
`PivotTableCaptions.errorCaption`	Алиас для значений, которые возвращают ошибку

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

6.97 Класс PivotTableCategoryField

`PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. Таблицу 48). Объект может быть получен посредством вызовов `PivotTable.getRowFields()`, `PivotTable.getColumnFields()`.

Таблица 48 – Описание полей `PivotTableCategoryField`

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

6.98 Класс PivotTableEditor

Предназначен для редактирования сводных таблиц. Возвращается посредством метода `PivotTable.createPivotTableEditor()`.

6.98.1 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();
```

6.98.2 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");
}
```

6.98.3 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();
```

6.98.4 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();
```

6.98.5 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 категориии. Метод возвращает объект `PivotTableEditor`.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.moveField("CC",
PivotTableFieldCategory.Values, 0);
pivotTableEditor.apply();
```

6.98.6 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();
```

6.98.7 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();
```

6.98.8 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();
```

6.98.9 Метод PivotTableEditor.setFilter

Метод задает фильтр PivotTableFilter сводной таблицы. Если фильтр не может быть применен, вызывается исключение PivotTableError. 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);
}
PivotTableUpdateResult pivotTableUpdateResult = pivotTableEditor.apply();
```

6.98.10 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();
```

6.98.11 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();
```

6.98.12 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();
```

6.98.13 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);
```

6.99 Класс PivotTableField

Класс `PivotTableField` содержит свойства полей сводной таблицы (см. Таблицу 49). Объект может быть получен посредством вызова `PivotTable.getFieldsList()`.

Таблица 49 – Описание полей класса `PivotTableField`

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

6.100 Класс PivotTableFieldCategories

Класс обеспечивает доступ к списку категорий поля сводной таблицы. Объект может быть получен посредством использования метода `PivotTable.getFieldCategories()`.

6.100.1 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");
    PivotTableFieldCategorysEnumerator enumerator = categories.GetEnumerator();
    foreach (var pivotTableFieldCategory in enumerator)
    {
        Console.WriteLine(pivotTableFieldCategory);
    }
}
```

6.101 Класс PivotTableFieldCategory

Класс `PivotTableFieldCategory` описывает флаги, которые задают категорию области полей. Описание полей представлено в таблице 50. Пример использования см. в `PivotTable.getFieldCategories()`.

Таблица 50 – Описание полей класса `PivotTableFieldCategory`

Поле	Описание
`PivotTableFieldCategory.Pages`	Область фильтров
`PivotTableFieldCategory.Rows`	Область строк
`PivotTableFieldCategory.Columns`	Область колонок
`PivotTableFieldCategory.Values`	Область значений

6.102 Класс PivotTableFieldProperties

`PivotTableFieldProperties` содержит свойства поля `PivotTableField` сводной таблицы (см. Таблицу 51).

Таблица 51 – Описание полей класса `PivotTableFieldProperties`

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

6.103 Класс PivotTableFields

Класс `PivotTableFields` представляет собой список полей сводной таблицы. Может быть получен посредством использования метода `PivotTable.getFieldsList()`.

6.103.1 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);
}
```

6.104 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();
}
```

6.104.1 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());
}
```

6.104.2 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());
}
```

6.104.3 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));
    }
}
```

6.104.4 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));
    }
}
```

6.104.5 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));
    }
}
```

6.105 Класс PivotTableFilters

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

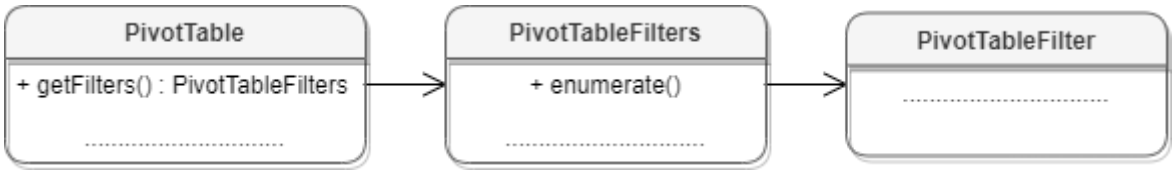


Рисунок 43 – Объектная модель классов для работы с фильтрами

6.105.1 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());
    }
}
```

6.106 Класс PivotTableFunction

Класс `PivotTableFunction` описывает функции, которые могут быть использованы в сводных таблицах. Описание полей представлено в таблице 52. Используется в качестве поля `subtotalFunctions` класса `PivotTableCategoryField`.

Таблица 52 – Описание полей класса `PivotTableFunction`

Поле	Описание
`PivotTableFunction.Auto`	Автозаполнение
`PivotTableFunction.Sum`	Суммирует все числовые данные

Поле	Описание
`PivotTableFunction.Count`	Количество всех ячеек
`PivotTableFunction.CountNums`	Количество числовых ячеек
`PivotTableFunction.Average`	Среднее значение
`PivotTableFunction.Max`	Наибольшее значение
`PivotTableFunction.Min`	Наименьшее значение
`PivotTableFunction.Product`	Произведение всех ячеек
`PivotTableFunction.StdDeviation`	Стандартное смещенное отклонение
`PivotTableFunction.StdDeviationPopulation`	Стандартное несмещенное отклонение
`PivotTableFunction.Variance`	Смещенная дисперсия
`PivotTableFunction.VariancePopulation`	Несмещенная дисперсия

6.107 Класс PivotTableItem

`PivotTableItem` описывает элемент сводной таблицы (см. Рисунок 44). См. пример в главе `PivotTableItems.GetEnumerator()`.

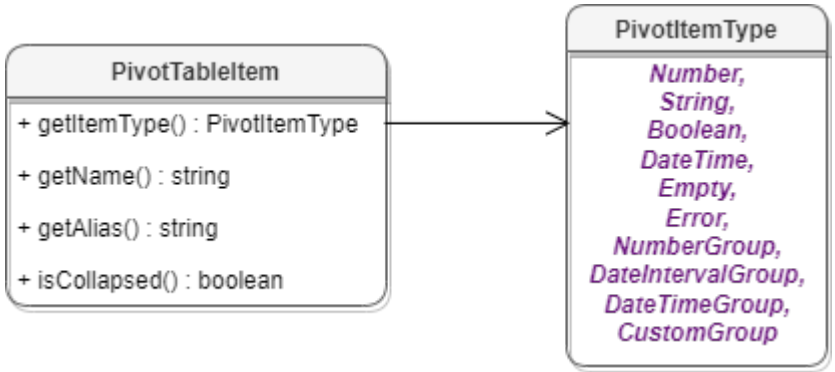


Рисунок 44 – Класс `PivotTableItem`

6.107.1 Метод PivotTableItem.getAlias

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем), тип - строка. См. пример в главе PivotTableItems.GetEnumerator().

6.107.2 Метод PivotTableItem.getItemType

Метод возвращает тип PivotTableItemType элемента сводной таблицы. См. пример в главе PivotTableItems.GetEnumerator().

6.107.3 Метод PivotTableItem.getName

Метод возвращает имя элемента сводной таблицы, тип - строка. См. пример в главе PivotTableItems.GetEnumerator().

6.107.4 Метод PivotTableItem.isCollapsed

Метод возвращает `true`, если элемент сводной таблицы свернут. См. пример в главе PivotTableItems.GetEnumerator().

6.108 Класс PivotTableItems

Класс обеспечивает доступ к списку элементов сводной таблицы PivotTable (см. Рисунок 45).

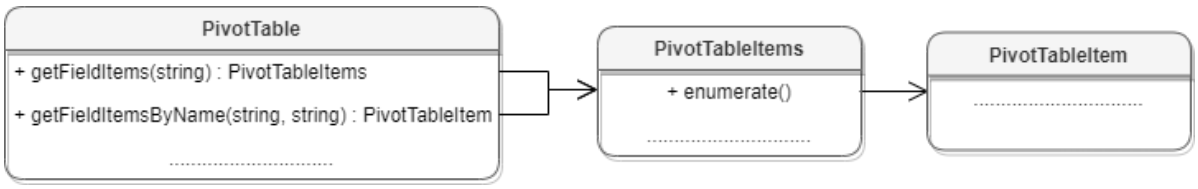


Рисунок 45 – Объектная модель классов для работы с элементами сводных таблиц

6.108.1 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());
    }
}
```

6.109 Класс PivotTableItemType

Класс `PivotTableItemType` содержит возможные типы элементов сводной таблицы. Описание полей представлено в таблице 53.

Таблица 53 – Описание полей класса `PivotTableItemType`

Поле	Описание
`PivotTableItemType.Number`	Числовой
`PivotTableItemType.String`	Строковый
`PivotTableItemType.Boolean`	Логический
`PivotTableItemType.DateTime`	Дата / время
`PivotTableItemType.Empty`	Пустой тип
`PivotTableItemType.Error`	Ошибка
`PivotTableItemType.NumberGroup`	Интервальная группировка
`PivotTableItemType.DateIntervalGroup`	Интервальная группировка по датам
`PivotTableItemType.DateTimeGroup`	Группировка по дате / времени
`PivotTableItemType.CustomGroup`	Пользовательская (произвольная) группировка

Пример:

```
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());
```

```
    }
}
```

6.110 Класс PivotTableLayoutSettings

Класс `PivotTableLayoutSettings` содержит настройки отображения сводной таблицы. Данный объект может быть получен в результате вызова `PivotTable.getPivotTableLayoutSettings()` и установлен методом `PivotTableEditor.setLayoutSettings()`. Описание полей таблицы представлено в таблице 54.

Таблица 54 – Описание полей класса `PivotTableLayoutSettings`

Поле	Описание
`PivotTableLayoutSettings.reportLayout`	Настройка вида макета сводной таблицы (`PivotTableReportLayout`: компактный, табличный, структурный)
`PivotTableLayoutSettings.valueFieldsOrientation`	Настраивает положение значений в случае, если в сводной таблице более двух полей значений. Тип - `ValueFieldsOrientation`
`PivotTableLayoutSettings.pageFieldOrder`	Настройка порядка полей фильтров (`PageFieldOrder`: вниз, затем поперек или сначала поперек, потом вниз)
`PivotTableLayoutSettings.indentForCompactLayout`	Размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей)
`PivotTableLayoutSettings.pageFieldWrapCount`	Настройка связана с `pageFieldOrder`, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д)
`PivotTableLayoutSettings.isMergeAndCenterLabelsEnabled`	Настройка позволяет объединить ячейки заголовков
`PivotTableLayoutSettings.useGridDropZones`	Флаг, отвечающий за отображение классического вида (как в Excel 2003). Влияет только на расположение полей в отчете

Поле	Описание
PivotTableLayoutSettings.displayFieldCaptions	Флаг, отвечающий за отображение заголовков полей

6.111 Класс PivotTablePageField

Содержит свойства поля из области фильтров (см. Таблицу 55). Объект может быть получен посредством вызова PivotTable.getPageFields().

Таблица 55 – Описание полей класса PivotTablePageField

Поле	Описание
PivotTablePageField.fieldProperties	Свойства поля PivotTableFieldProperties

6.112 Класс PivotTableReportLayout

Класс PivotTableReportLayout описывает внешний вид отчетов сводной таблицы. Является полем класса PivotTableLayoutSettings. Описание полей представлено в таблице 56.

Таблица 56 – Описание полей класса PivotTableReportLayout

Поле	Описание
PivotTableReportLayout.Compact	Компактный вид
PivotTableReportLayout.Tabular	Табличный вид
PivotTableReportLayout.Outline	Структурный вид

6.113 Класс PivotTablesManager

Класс PivotTablesManager используется для создания сводных таблиц, содержит метод create(). Может быть получена вызовом Document.getPivotTablesManager().

Пример:

```
PivotTablesManager pivotTablesManager = document.getPivotTablesManager();
```

6.113.1 Метод PivotTablesManager.create

Метод создает сводную таблицу `PivotTable` на основе диапазона исходных данных `CellRange`.

Если местоположение не задано, создается новый лист (таблица), и сводная таблица будет расположена по умолчанию.

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
CellRange cellRange = sheet.getCellRange("B3:C4");
PivotTablesManager pivotTablesManager = document.getPivotTablesManager();
PivotTable pivotTable = pivotTablesManager.create(cellRange);
```

6.114 Класс PivotTableUnsupportedFeature

Класс `PivotTableUnsupportedFeature` описывает неподдерживаемую функциональность сводных таблиц. Получение неподдерживаемой функциональности сводных таблиц описано в `PivotTable.getUnsupportedFeatures()`. Описание полей класса представлено в таблице 57.

Таблица 57 – Описание полей класса `PivotTableUnsupportedFeature`

Поле	Описание
`PivotTableUnsupportedFeature.CalculatedField`	Вычисляемые поля
`PivotTableUnsupportedFeature.CalculatedItem`	Вычисляемые элементы
`PivotTableUnsupportedFeature.CollapsedValues`	Свернутые поля
`PivotTableUnsupportedFeature.ShowDataAs`	Вычисления ("Show data" как в MS Excel)

6.115 Класс PivotTableUpdateResult

В таблице 58 приведены константы, которые соответствуют возможным результатам обновления сводной таблицы (см. методы `PivotTable.update()`, `PivotTableEditor:apply()`).

Таблица 58 – Результаты обновления сводной таблицы

Наименование константы	Описание
`PivotTableUpdateResult.Success`	Успешное обновление таблицы
`PivotTableUpdateResult.NoPivotTable`	Сводная таблица не найдена
`PivotTableUpdateResult.NoSuchFieldInCategory`	Не найдено поле в категории
`PivotTableUpdateResult.NoSuchFieldInPivotTable`	Не найдено поле в сводной таблице
`PivotTableUpdateResult.InvalidIndex`	Ошибка в индексе
`PivotTableUpdateResult.FieldAlreadyEnabled`	Поле уже существует
`PivotTableUpdateResult.MovingFieldToTheSameCategoryForbidden`	Попытка перемещения поля в рамках текущей категории
`PivotTableUpdateResult.InvalidFunction`	Неправильная функция
`PivotTableUpdateResult.InvalidCategory`	Неправильная область
`PivotTableUpdateResult.InvalidDataSourceRange`	Ошибка диапазона исходных данных
`PivotTableUpdateResult.NoDataRowsInDataSource`	В исходных данных нет строк с данными
`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.Canceled`	Обновление сводной таблицы отменено

6.116 Класс PivotTableValueField

`PivotTableValueField` содержит свойства поля сводной таблицы, использующегося как значение столбец (см. Таблицу 59). Таблица может быть получена посредством вызова `PivotTable.getValueFields()`.

Таблица 59 – Описание полей класса `PivotTableValueField`

Поле	Описание
`PivotTableValueField.baseFieldName`	Оригинальное поле на основе которого было создано данное поле, тип - строка.
`PivotTableValueField.valueFieldName`	Автоматический уникальный псевдоним такой как “Sum of %имя поля%“, тип – строка.
`PivotTableValueField.cellNumberFormat`	Числовой формат типа `CellFormat` для конкретного поля значений.
`PivotTableValueField.totalFunction`	Агрегирующая функция `PivotTableFunction` поля значений (SUM, COUNT, MAX и т.д).
`PivotTableValueField.customFormula`	Вычисляемая формула для поля значений, тип - строка.

6.117 Класс PointU

Класс `PointU` представляет собой точку в двухмерном пространстве.

Список свойств:

- `x` – координата точки по оси x;
- `y` – координата точки по оси y.

Примеры:

```
PointU point = new PointU(50, 50);
```

```
PointU point = new PointU();
point.x = 50;
point.y = 50;
```

6.118 Класс Position

Класс `Position` представляет местоположение в текстовом документе. Используется для обозначения начала и конца диапазона `Range`.

6.118.1 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();
```

6.118.2 Метод Position.insertBookmark

Вставляет закладку с наименованием в текущую позицию.

Пример:

```
document.getRange().getEnd().insertBookmark("Bookmark example");
```

6.118.3 Метод Position.insertImage

Вставляет изображение из файла в текущую позицию. Возвращает объект `Image`, содержащий вставленное изображение. На данный момент 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));
```

Внимание! Данный метод использует текстовый параметр в формате unicode string. Если символы данной строки отличаются от латиницы и кодируются двумя байтами (например, кириллица), то необходимо принять дополнительные меры, описанные в разделе Использование кодировок. В противном случае возникнет исключение `DocumentAPI.UnknownError` с сообщением «Invalid UTF-8».

6.118.4 Метод Position.insertLineBreak

Метод предназначен для вставки перевода строки.

Пример:

```
Range range = document.getRange();
Position endPosition = range.getEnd();
endPosition.insertLineBreak();
```

6.118.5 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();
```

6.118.6 Метод Position.insertSectionBreak

Вставляет разрыв раздела в текущую позицию. В качестве параметра выступает тип SectionBreakType. При вызове без параметра используется тип SectionBreakType.NextPage.

Примеры:

```
Range range = document.getRange();
Position endPosition = range.getEnd();
endPosition.insertSectionBreak();
```

```
endPosition.insertSectionBreak(SectionBreakType.EvenPage);
```

6.118.7 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.

6.118.8 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");
```

Внимание! Данный метод использует текстовый параметр в формате unicode string. Если символы данной строки отличаются от латиницы и кодируются двумя байтами (например, кириллица), то необходимо принять дополнительные меры, описанные в разделе Использование кодировок. В противном случае возникнет исключение `DocumentAPI.UnknownError` с сообщением «Invalid UTF-8».

6.118.9 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);
```

6.118.10 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);
```

6.119 Класс PresentationExportSettings

Класс `PresentationExportSettings` предоставляет настройки, необходимые для экспорта презентационных документов (см. `Document.exportAs`). Поле объекта `PresentationExportSettings.skipHiddenSlides` позволяет включить в созданный документ скрытые слайды.

Пример:

```
PresentationExportSettings exportSettings = new PresentationExportSettings();
exportSettings.skipHiddenSlides = false;
document.exportAs(filePath, ExportFormat.PDFA1, exportSettings);
```

6.120 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);
```

6.120.1 Метод PrintingScope.getCellRange

Метод возвращает диапазон ячеек таблицы (см. `CellRangePosition`).

6.120.2 Метод PrintingScope.usePrintArea

Метод возвращает `true`, если область печати должна использоваться во время печати, экспорта и т. д.

6.120.3 Тип PrintingScope.Type

Варианты выбора диапазона страниц для экспорта и печати представлены в таблице 60. Используется в `PrintingScope`

Таблица 60 – Диапазон страниц для экспорта и печати

Наименование константы	Описание
`PrintingScope.Type.PrintArea`	Выбранная область печати (по умолчанию)
`PrintingScope.Type.WholeSheet`	Печать всего документа

6.121 Класс Range

Класс `Range` предоставляет доступ к диапазону документа. На рисунке 46 изображена объектная модель классов, относящихся к работе с диапазонами.

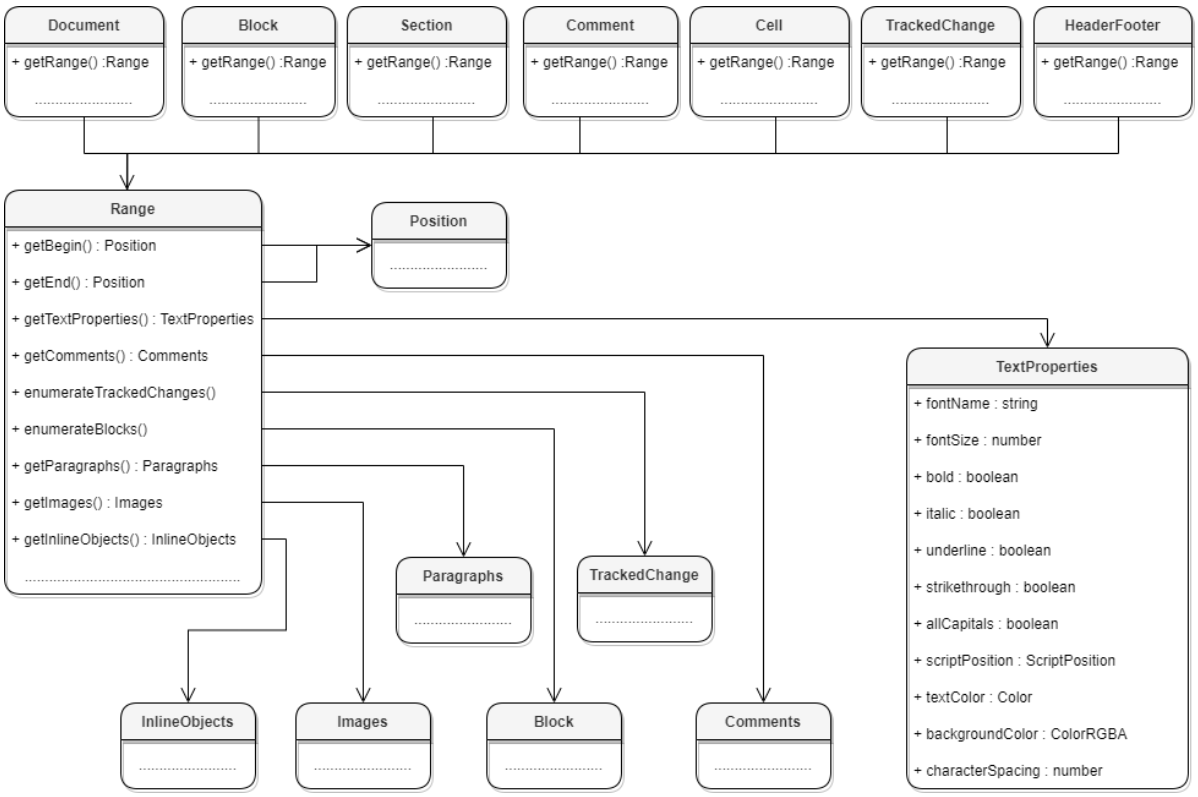


Рисунок 46 – Объектная модель для работы с классом `Range`

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());
}
```

6.121.1 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`.

6.121.2 Метод Range.extractText

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и другие объекты игнорируются.

Пример для текстового документа:

```
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());
}
```

6.121.3 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");
}
```

6.121.4 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());
    }
}
```

6.121.5 Метод Range.getComments

Обеспечивает доступ к комментариям в диапазоне.

Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам. Если дат нет, то порядок комментариев не определен.

Пример:

```
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());
}
```

6.121.6 Метод Range.getEnd

Метод возвращает позицию в конце диапазона, не включая последний символ paragraph mark.

Пример для текстового документа:

```
Position endDocPosition = document.getRange().getEnd();
endDocPosition.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 endDocPos = cellRange.getEnd();
    endDocPos.insertText("API");
}
```

6.121.7 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());
}
```

6.121.8 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)
{
    ..............................
}
```

6.121.9 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());
    }
}
```

6.121.10 Метод Range.getTextProperties

Метод возвращает объект с текущими настройками форматирования для фрагмента текстового документа. Описание настроек форматирования осуществляется с помощью объекта TextProperties.

Пример для текстового документа:

```
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);
}
```

6.121.11 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());
}
```

6.121.12 Метод Range.isContentLocked

Метод возвращает значение `true`, если изменения содержимого диапазона запрещены.

Пример для текстового документа:

```
Range range = document.getRange();
Console.WriteLine(range.isContentLocked());
```

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.isContentLocked());
}
```

6.121.13 Метод Range.lockContent

Метод запрещает изменения содержимого диапазона.

Внимание ! 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());
}
```

6.121.14 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());
}
```

6.121.15 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");
}
```

Внимание! Данный метод использует текстовый параметр в формате unicode string. Если символы данной строки отличаются от латиницы и кодируются двумя байтами (например, кириллица), то необходимо принять дополнительные меры, описанные в разделе Использование кодировок. В противном случае возникнет исключение `DocumentAPI.UnknownError` с сообщением «Invalid UTF-8».

6.121.16 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);
}
```

6.121.17 Метод Range.unlockContent

Метод разрешает изменения содержимого диапазона.

Внимание ! 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());
}
```

6.122 Класс RangeBorders

Класс `RangeBorders` оставлен для совместимости. Вместо него следует использовать класс `Borders`.

6.123 Класс RectU

Класс `RectU` описывает прямоугольник в двухмерном пространстве.

Список свойств:

- `topLeft` – координата левой верхней вершины прямоугольника;
- `bottomRight` – 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);
```

6.124 Класс SaveDocumentSettings

Класс `SaveDocumentSettings` предоставляет настройки, используемые для сохранения документа в файл (см. `Document.saveAs()`). Описание полей класса `SaveDocumentSettings` представлено в таблице 61.

Таблица 61 – Описание полей класса `SaveDocumentSettings`

Поле	Описание
`SaveDocumentSettings.documentFormat`	Формат документа `DocumentFormat`.
`SaveDocumentSettings.documentType`	Тип документа `DocumentType`.
`SaveDocumentSettings.documentPassword`	Пароль для защиты электронного документа от несанкционированного доступа.
`SaveDocumentSettings.isTemplate`	Флаг, обозначающий, что документ должен быть сохранен как шаблон.
`SaveDocumentSettings.dsvSettings`	Структура `DSVSettings`, необходимая для сохранения в формате DSV.

6.125 Класс ScaleFrom

Варианты якоря для масштабирования `AbsoluteFrame` представлены в таблице 62. Используется в качестве поля `scaleFrom` метода `AbsoluteFrame.scale()`.

Таблица 62 – Варианты для якоря масштабирования

Наименование константы	Описание
`BottomRight`	Правый нижний угол
`BottomLeft`	Левый нижний угол
`TopLeft`	Левый верхний угол
`TopRight`	Правый верхний угол

Пример:

```
TextProperties textProperties = new TextProperties();
textProperties.scriptPosition = ScriptPosition.NormalScript;
document.getRange().setTextProperties(textProperties);
```

6.126 Класс ScientificCellFormatting

Класс содержит параметры для экспоненциального формата ячеек таблицы. Используется в качестве аргумента метода `Cell.setFormat()`. Описание полей класса `ScientificCellFormatting` представлено в таблице 63.

Таблица 63 – Описание полей класса `ScientificCellFormatting`

Поле	Описание
`ScientificCellFormatting.decimalPlaces`	Количество десятичных позиций

Поле	Описание
`ScientificCellFormatting.minExponentDigits`	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());
```

6.127 Класс Script

Класс `Script` предназначен для управления отдельной макрокомандой. Содержит свойства `Name` и `Body`.

6.127.1 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());
    }
}
```

6.127.2 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());
    }
}
```

6.127.3 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());
}
```

6.127.4 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());
    }
}
```

6.128 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");
```

6.128.1 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");
```

6.129 Класс ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 64. Используется в качестве поля `scriptPosition` класса `TextProperties`.

Таблица 64 – Типы надстрочного и подстрочного форматирования

Наименование константы	Описание
`SuperScript`	Надстрочный знак (верхний индекс)
`SubScript`	Подстрочный знак (нижний индекс)
`NoramlScript`	Без указания индекса

Пример:

```
TextProperties textProperties = new TextProperties();
textProperties.scriptPosition = ScriptPosition.NormalScript;
document.getRange().setTextProperties(textProperties);
```

6.130 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());
    }
}
```

6.130.1 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());
    }
}
```

6.130.2 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());
}
```

6.130.3 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");
```

```
    }
}
```

6.130.4 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);
}
```

6.131 Класс Search

Класс `Search` предоставляет доступ к механизму поиска фрагментов документа, открытого в редакторе текста или таблиц.

6.131.1 Метод Search.findText

Метод выполняет поиск строки с учетом и без учета регистра:

- во всем документе;
- в выбранном диапазоне;
- в выбранном диапазоне ячеек;
- в таблице.

Результат возвращается в виде диапазона `Range`, содержащего искомый фрагмент.

Если строка не обнаружена, возвращается пустой диапазон.

Возможно использование следующих вариантов метода:

```
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.

6.132 Класс Section

Класс `Section` представляет собой раздел в документе.

6.132.1 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);
}
```

6.132.2 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);
}
```

6.132.3 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());
}
```

6.132.4 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);
}
```

6.132.5 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());
}
```

6.132.6 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());
}
```

6.132.7 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);
}
```

6.133 Класс SectionBreakType

Таблица `SectionBreakType` описывает варианты разрыва страниц, используется в `Position.InsertSectionBreak()`.

Описание полей класса `SectionBreakType` представлено в таблице 65.

Таблица 65 – Описание полей класса `SectionBreakType`

Поле	Описание
`SectionBreakType.NextPage`	Следующий раздел начинается с новой страницы
`SectionBreakType.Continious`	Следущий 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

6.134 Класс Sections

Класс `Sections` используется для доступа к коллекции секций документа. Описание секции см. в разделе `Section`.

6.134.1 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());
}
```

6.135 Класс Shape

Класс `Shape` представляет собой фигуру, содержит методы для установки и получения свойств `ShapeProperties`.

6.135.1 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);
}
```

6.135.2 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);
}
```

6.136 Класс ShapeProperties

Класс описывает свойства фигуры и используется в Shape.getShapeProperties, Shape.setShapeProperties.

Содержит следующие поля:

– verticalAlignment - вертикальное выравнивание, тип VerticalAlignment;

– borderProperties - свойства границ фигуры, тип LineProperties;

– fill - свойства заполнения фигуры, тип Fill;

– shapeTextLayout - свойства текста внутри фигуры, тип ShapeTextLayout.

Пример:

```
ShapeProperties shapeProperties = shape.getShapeProperties();
shapeProperties.verticalAlignment = ...
shapeProperties.borderProperties = ...
shapeProperties.fill = ...
shapeProperties.shapeTextLayout = ...
shape.setShapeProperties(shapeProperties);
```

6.136.1 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);
}
```

6.136.2 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);
```

6.136.3 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);
}
```

6.136.4 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);
}
```

6.137 Класс ShapeTextLayout

Класс ShapeTextLayout описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 66. Используется в качестве поля shapeTextLayout класса ShapeProperties.

Таблица 66 – Описание полей класса ShapeTextLayout

Поле	Описание
ShapeTextLayout.DoNotAutoFit	Размещение текста в фигуре по умолчанию
ShapeTextLayout.FitShapeExtentToText	Расширение фигуры под текст
ShapeTextLayout.FitTextToShape	Заполнение фигуры текстом

6.138 Класс SizeU

Класс SizeU описывает размер объекта в двухмерном пространстве.

Список свойств:

- width – ширина объекта в двухмерном пространстве;
- height – высота объекта в двухмерном пространстве.

Примеры:

```
SizeU size = new SizeU(50, 50);
```

```
SizeU size = new SizeU();
size.width = 50;
size.height = 50;
```

6.139 Класс Table

Класс `Table` предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 47).

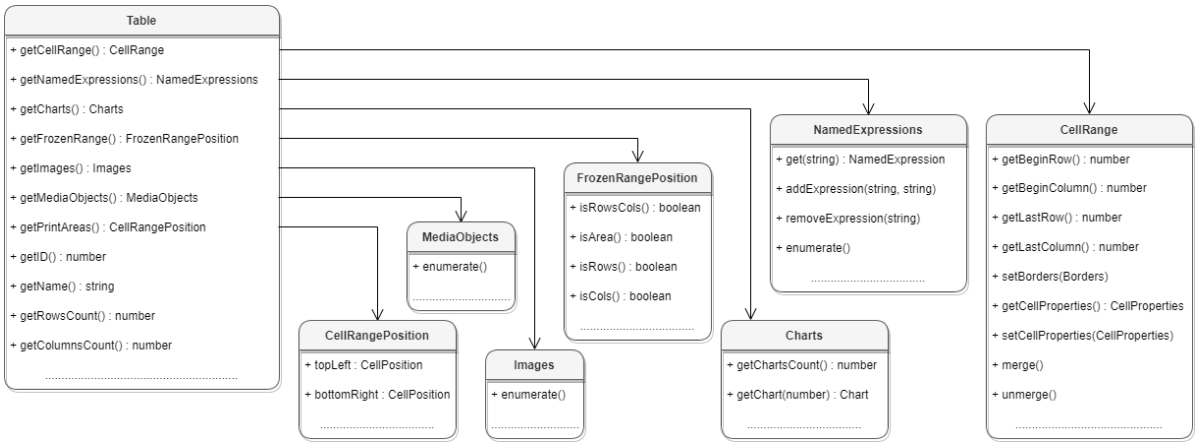


Рисунок 47 – Структура полей класса `Table`

Класс `Table` наследуется от класса `Block` и обладает его базовыми методами `Block.getRange`, `Block.getSection`, `Block.remove`.

6.139.1 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.

6.139.2 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.

6.139.3 Метод Table.createFiltersRange

Метод `Table.createFiltersRange` задает диапазон, который используется как диапазон фильтрации.

```
FiltersRange createFiltersRange(const CellRangePosition& range);
```

В качестве параметра используется диапазон ячеек типа `CellRangePosition`. Метод возвращает `FiltersRange`.

Разрешен только один диапазон фильтрации на таблицу. Это означает, что данный метод удаляет ранее определенный диапазон фильтрации. При этом есть исключение: если новый диапазон начинается с той же позиции, предыдущие фильтры будут сохранены.

Диапазон фильтрации должен включать дополнительную строку, которая используется как заголовок таблицы. Эта строка никогда не фильтруется.

Метод может быть использован только в табличных документах.

Метод может формировать следующие исключения:

– `NotImplementedError`: метод вызывается для неподдерживаемого документа
– `IncorrectArgumentError`: диапазон слишком мал или имеет недопустимые элементы
– `DocumentModificationError`: диапазон пересекает объединенные ячейки или содержит сводную таблицу
– `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.

6.139.4 Метод Table.duplicate

Для создания копии листа в табличном документе используется метод `duplicate`. Созданная копия листа размещается после копируемого листа. Для того, чтобы избежать повторяющихся имен, к имени нового листа добавляется индекс. Метод может быть использован только в табличном документе.

Пример:

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

6.139.5 Метод Table.freeze

Метод `freeze` закрепляет заданную область `FrozenRangePosition` таблицы.

Пример:

```
var frozenRangePosition = FrozenRangePosition.createFrozenCols(0, 2);
table.freeze(frozenRangePosition);
```

6.139.6 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());
```

6.139.7 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);
```

6.139.8 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());
}
```

6.139.9 Метод Table.getColumnsCount

Метод позволяет получить количество столбцов таблицы.

Пример:

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

6.139.10 Метод Table.getFiltersRange

Метод `Table.getFiltersRange` возвращает текущий диапазон фильтрации, принадлежащий таблице. Рабочий лист табличного документа может содержать только один диапазон фильтрации.

```
FiltersRange getFiltersRange();
```

Метод возвращает FiltersRange, если диапазон фильтрации существует.

Метод может быть использован только в табличных документах.

При использовании данного метода может произойти исключение InvalidObjectError в случае, если диапазон недействителен.

Пример:

```
FiltersRange filtersRange = sheet.getFiltersRange();
CellRange cellRange = filtersRange.getCellRange();
Console.WriteLine(cellRange.getBeginRow());
```

Более подробный пример приведен в разделе Работа с фильтрами.

6.139.11 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());
```

6.139.12 Метод Table.getImages

Метод предназначен для получения списка изображений в таблице.

```
Images images = table.getImages();
ImagesEnumerator imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    ......................
}
```

6.139.13 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)
{
    ........................
}
```

6.139.14 Метод Table.getName

Метод позволяет получить имя листа табличного документа.

Пример:

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

6.139.15 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());
}
```

6.139.16 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();
```

6.139.17 Метод Table.getProtectionProperties

Метод возвращает параметры защиты от изменений листа табличного документа.

Вызов:

```
public TableProtectionProperties getProtectionProperties()
```

Возвращает:

– TableProtectionProperties: свойства защиты листа документа (null, если защита листа не установлена).

Используйте методы setProtection() и removeProtection(), чтобы установить и снять защиту от изменений листа. Вы также можете использовать метод isProtected(), чтобы узнать, установлена ли защита на лист.

6.139.18 Метод Table.getRowsCount

Метод позволяет получить количество строк таблицы.

Пример:

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

6.139.19 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());
```

6.139.20 Метод Table.groupColumns

Метод предназначен для группировки столбцов таблицы, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
groupColumns(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет начата группировка столбцов;
- `columnsCount` – количество столбцов для группировки.

6.139.21 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.

6.139.22 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);
```

6.139.23 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);
```

6.139.24 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);
```

6.139.25 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);
```

6.139.26 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.

6.139.27 Метод Table.isProtected

Метод возвращает состояние защиты от изменений листа табличного документа.

Вызов:

```
public bool isProtected()
```

Используйте методы `setProtection()` и `removeProtection()`, чтобы установить и снять защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод `getProtectionProperties()`.

6.139.28 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.

6.139.29 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);
}
```

6.139.30 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);
```

6.139.31 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();
}
```

6.139.32 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.

6.139.33 Метод Table.removeProtection

Метод снимает защиту от изменений с 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");
```

Используйте метод `setProtection()`, чтобы установить защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод `getProtectionProperties()`. Вы также можете использовать метод `isProtected()`, чтобы узнать, установлена ли защита на лист.

Если введенный пароль не совпадает с заданным при установке защиты, возникает исключение `DocumentAPI.IncorrectPasswordError`.

6.139.34 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.

6.139.35 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.

6.139.36 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.

6.139.37 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.

6.139.38 Метод Table.setColumnWidth

Метод устанавливает ширину столбца таблицы в пунктах (1/72 дюйма).

Вызов:

```
setColumnWidth( columnIndex, width )
```

Параметры:

- `columnIndex` – индекс столбца в таблице, для которого устанавливается значение ширины. Индексация столбцов начинается с нуля.
- `width` – ширина столбца в пунктах (1/72 дюйма).

Пример:

```
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");

// Установить ширину столбца в 400 pt
table.setColumnWidth(1, 400);
```

6.139.39 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.

6.139.40 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));
```

6.139.41 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());
```

6.139.42 Метод Table.setProtection

Метод устанавливает защиту от изменений на лист 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");
```

Используйте метод `removeProtection()`, чтобы снять защиту от изменений листа. Получить текущие параметры защиты листа позволяет метод `getProtectionProperties()`. Вы также можете использовать метод `isProtected()`, чтобы узнать, установлена ли защита на лист.

Метод `setProtectionProperties()` объектов `Cell` и `CellRange` позволяет задать параметры защиты для ячеек (например, разрешить редактирование определенных ячеек в документе перед установкой защиты).

Если метод `setProtection()` применяется к уже защищенному листу, возникает исключение `DocumentAPI.SpreadsheetProtectionError`.

6.139.43 Метод Table.setRowHeight

Метод устанавливает высоту строки таблицы в пунктах (1/72 дюйма).

Вызов:

```
setRowHeight(rowIndex, height)
```

Параметры:

- `rowIndex` – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- `height` – высота строки в пунктах (1/72 дюйма).

– `rowHeightRule` – точность значения (`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);
```

6.139.44 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.

6.139.45 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);
```

6.139.46 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);
```

6.139.47 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.

6.139.48 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.

6.140 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.

6.140.1 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();
```

6.140.2 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);
```

6.140.3 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.

6.141 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 67 – 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

Поле	Значение по умолчанию	Описание
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");
```

6.142 Класс TableRangeInfo

Класс TableRangeInfo описывает диапазон ячеек таблицы (см. ChartRangeInfo).

Описание полей класса TableRangeInfo представлено в таблице 68.

Таблица 68 – Поля класса TableRangeInfo

Поле	Тип	Описание
tableRange	CellRangePosition	Диапазон ячеек

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("Top left row:" + tableRange.topLeft.row + ", top left
column:" + tableRange.topLeft.column);
```

6.143 Класс TextAnchoredPosition

Класс `TextAnchoredPosition` представляет позицию объекта на странице текстового документа (см. `InlineFrame.setPosition()`). Описание полей представлено в таблице 69.

Таблица 69 – Описание полей класса `TextAnchoredPosition`

Поле	Описание
`TextAnchoredPosition.horizontal`	Позиция по горизонтали `HorizontalTexAnchoredPosition`
`TextAnchoredPosition.vertical`	Позиция по вертикали `VerticalTextAnchoredPosition`

Пример:

```
TextAnchoredPosition textAnchoredPosition = new TextAnchoredPosition();
textAnchoredPosition.horizontal = new
HorizontalTextAnchoredPosition(HorizontalRelativeTo.Character);
textAnchoredPosition.vertical = new
VerticalTextAnchoredPosition(VerticalRelativeTo.Character);
frame.setPosition(textAnchoredPosition);
```

6.144 Класс TextExportSettings

Класс `TextExportSettings` предоставляет настройки, необходимые для экспорта текстовых документов (см. `Document.exportAs`). Поле объекта `TextExportSettings.pageNumbers` является экземпляром класса `PageNumbers`, в котором содержатся настройки страниц для экспорта текстовых документов.

Пример:

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

6.145 Класс TextProperties

Класс `DocumentAPI.TextProperties` содержит поля, задающие параметры текста. На рисунке 48 изображена объектная модель класса `DocumentAPI.TextProperties`.

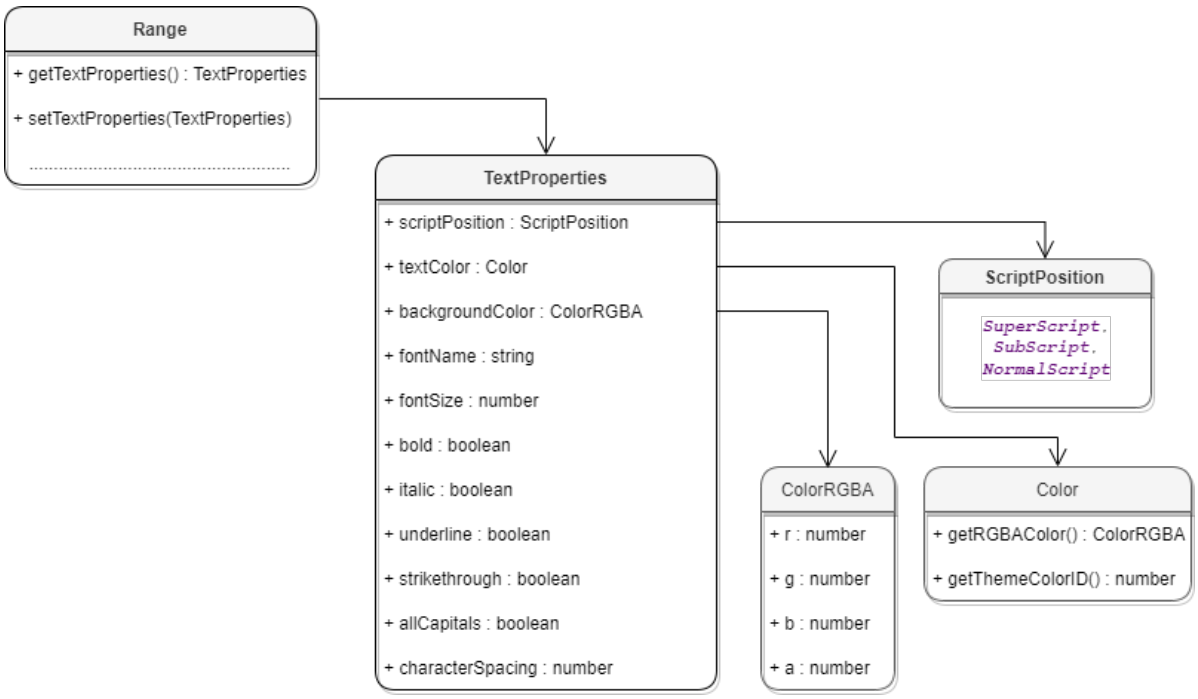


Рисунок 48 – Объектная модель для работы с классом `DocumentAPI.TextProperties`

Описание полей класса `TextProperties` представлено в таблице 70. Свойства `TextProperties` применяются к диапазону текста `Range` (методы `Range.getTextProperties()`, `Range.setTextProperties()`).

Таблица 70 – Описание полей класса `TextProperties`

Поле	Тип	Описание
`TextProperties.fontName`	Строковое	Наименование шрифта, использованного для оформления фрагмента документа.
`TextProperties.fontSize`	Числовое	Размер шрифта, использованного для оформления фрагмента документа.
`TextProperties.bold`	Логическое	Значение `true` устанавливает жирное начертание для указанного фрагмента

Поле	Тип	Описание
		текста.
TextProperties.italic	Логическое	Значение true устанавливает начертание курсивом для указанного фрагмента текста.
TextProperties.underline	Логическое	Значение true устанавливает подчеркивание для указанного фрагмента текста.
TextProperties.strikethrough	Логическое	Значение true устанавливает начертание «зачеркнутый» для указанного фрагмента текста.
TextProperties.allCapitals	Логическое	Значение true устанавливает все буквы указанного фрагмента текста как прописные. Значение false устанавливает все буквы указанного фрагмента текста как строчные.
TextProperties.scriptPosition	ScriptPosition	Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме.
TextProperties.textColor	Color	Цвет указанного фрагмента документа.
TextProperties.backgroundColor	ColorRGBA	Цвет фона указанного фрагмента документа.
TextProperties.characterSpacing	Числовое	Размер межсимвольного интервала.

Пример:

```
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);
}
```

6.146 Класс TextWrapType

В таблице 71 представлены варианты обтекания текстом встроенного объекта. Используется в InlineFrame.setWrapType(), InlineFrame.getWrapType().

Таблица 71 – Варианты обтекания текстом встроенного объекта

Наименование константы	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

Пример:

```
var frame = inlineObject.getFrame();
frame.setWrapType(TextWrapType.Inline);
```

6.147 Класс ThemeColorID

В таблице 72 представлены типы идентификаторов цветов тем. Используется в `Color`.

Таблица 72 – Типы идентификаторов цветов тем

Наименование константы	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

Наименование константы	Описание
`ThemeColorID.Accent4`	Акцент4
`ThemeColorID.Accent5`	Акцент5
`ThemeColorID.Accent6`	Акцент6
`ThemeColorID.Hyperlink`	Гиперссылка
`ThemeColorID.FollowedHyperlink`	Следующая гиперссылка

6.148 Класс TimePatterns

Форматы времени представлены в таблице 73. Пример использования см. в разделе `DateTimeCellFormatting`.

Таблица 73 – Форматы времени

Наименование константы	Описание
`TimePatterns.ShortTime`	`'hh:mm AM/PM'` для языка `en_US`
`TimePatterns.LongTime`	`'hh:mm:ss AM/PM'` для языка `en_US`

6.149 Класс TimeZone

Класс `TimeZone` предоставляет настройки, необходимые для экспорта текстовых документов, см. `DocumentSettings`.

Поле класса `TimeZone.offsetInSecondsToUTC` (числовой тип) содержит значение, с помощью которого задается смещение или разность между временем в данном часовом поясе и временем в формате UTC (Всемирное координированное время).

6.150 Класс TrackedChange

Класс `TrackedChange` представляет отслеживаемое изменение в диапазоне документа (см. Рисунок 49).

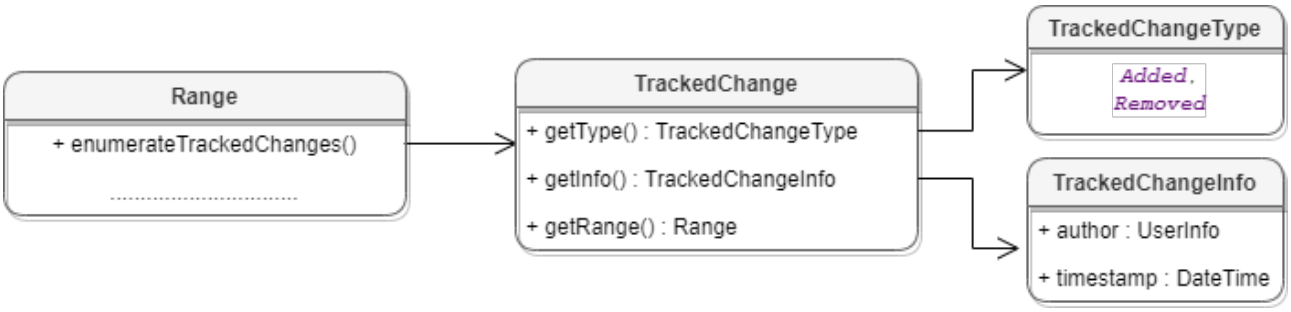


Рисунок 49 – Объектная модель классов для работы с отслеживаемыми изменениями

Для получения списка отслеживаемых изменений используется метод `Range.getTrackedChangesEnumerator()`.

Пример:

```
var trackedChangesEnumerator =
document.getRange().getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator)
{
    ........................................
}
```

6.150.1 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);
}
```

6.150.2 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());
}
```

6.150.3 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());
}
```

6.151 Класс TrackedChangeInfo

Класс `TrackedChangeInfo` содержит информацию об отслеживаемых изменениях. Используется в `TrackedChange.getInfo`, `Comment.getInfo`. Описание полей представлено в таблице 74.

Таблица 74 – 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);
};
```

6.152 Класс TrackedChangeType

Класс `TrackedChangeType` содержит типы отслеживаемых изменений, возвращается методом `TrackedChange.getType()`.

Типы отслеживаемых изменений:

- `Added` – добавленные изменения;
- `Removed` – удаленные изменения.

Пример:

```
var range = document.getRange();
TrackedChangesEnumerator trackedChangesEnumerator =
range.getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator)
{
    if (trackedChange.getType() == TrackedChangeType.Added)
        Console.WriteLine("Added");
    else
        Console.WriteLine("Removed");
}
```

6.153 Класс UserInfo

Класс `UserInfo` предоставляет информацию о пользователе, используется в `TrackedChangeInfo`, `DocumentSettings`.

Описание полей класса `UserInfo` представлено в таблице 75.

Таблица 75 – Описание полей класса `UserInfo`

Поле	Описание	Тип
`UserInfo.name`	Имя пользователя	Строка
`UserInfo.email`	Адрес электронной почты пользователя	Строка

6.154 Класс ValueFieldsOrientation

Класс `ValueFueldsOrientation` описывает варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений. Является полем класса `PivotTableLayoutSettings`. Описание полей представлено в таблице 76.

Таблица 76 – Описание полей `ValueFueldsOrientation`

Поле	Описание
`ValueFueldsOrientation.ByRows`	По строкам
`ValueFueldsOrientation.ByColumns`	По столбцам

6.155 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.

6.155.1 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");
```

6.155.2 Метод ValuesTableFilter.clear

Метод `ValuesTableFilter.clear` удаляет все элементы фильтра.

Пример:

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

6.156 Класс VectorString

Тип `VectorString` представляет собой коллекцию данных типа `string`. Используется в качестве контейнера для имен листов в классе `WorkbookExportSettings`.

Примеры использования:

```
// Содание 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();
```

6.157 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:

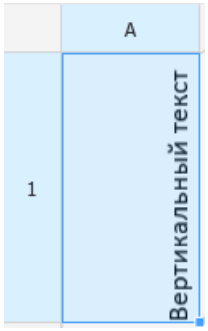
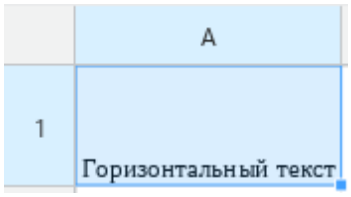
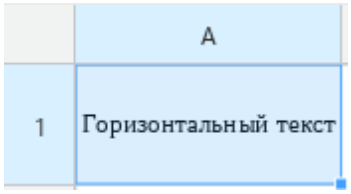
```
// Содание вектора
VectorUInt vector = new VectorUInt(3);
// Добавление новых элементов
vector.Add(1);
vector.Add(2);
vector.Add(3);
// Добавление другого вектора
vector.AddRange(vector);
// Определение размера
vector.Capacity = 3;
Console.WriteLine(vector.Capacity);
// Очистка содержимого
vector.Clear();
// 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);
}
```

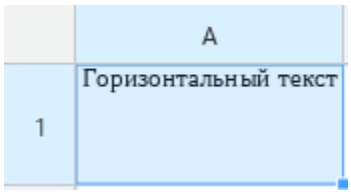
```
// Получение подмножества элементов
VectorUInt range = vector.GetRange(0, 2);
// Вставка элемента по индексу
vector.Insert(0, 2);
// Вставка вектора по индексу
vector.InsertRange(0, vector);
// Удаление по индексу
vector.RemoveAt(0);
// Удаление диапазона
vector.RemoveRange(0, 2);
// Преобразование в массив
uint[] array = vector.ToArray();
```

6.158 Класс VerticalAlignment

В таблице 77 представлены константы,описывающие варианты выравнивания текста по вертикали. Используется в CellProperties, ShapeProperties.

Таблица 77 – Виды выравнивания текста по вертикали

Наименование константы	Представление в интерфейсе	
VerticalAlignment.Bottom		
VerticalAlignment.Center		

Наименование константы	Представление в интерфейсе	
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);
```

6.159 Класс VerticalAnchorAlignment

В таблице 78 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали (см. описание класса VerticalTextAnchoredPosition).

Таблица 78 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
VerticalAnchorAlignment.Top	По верхнему краю
VerticalAnchorAlignment.Bottom	По нижнему краю
VerticalAnchorAlignment.Center	По центру
VerticalAnchorAlignment.Inside, VerticalAnchorAlignment.Outside	По границам

6.160 Класс VerticalRelativeTo

В таблице 79 представлены типы размещения объекта относительно закрепленной позиции по вертикали (см. описание класса VerticalTextAnchoredPosition).

Таблица 79 – Типы размещения объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
VerticalRelativeTo.Character	Символ

Наименование константы	Описание
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

6.161 Класс VerticalTextAnchoredPosition

Класс VerticalTextAnchoredPosition предназначен для управления относительным положением объекта со смещением или выравниванием по вертикали (см. описание класса TextAnchoredPosition).

Описание полей класса VerticalTextAnchoredPosition представлено в таблице 80.

Таблица 80 – Описание полей класса VerticalTextAnchoredPosition

Поле	Описание
VerticalTextAnchoredPosition.relativeTo	Тип размещения объекта относительно закрепленной позиции по вертикали VerticalRelativeTo
VerticalTextAnchoredPosition.offset	Смещение объекта
VerticalTextAnchoredPosition.aligment	Тип выравнивания объекта относительно закрепленной позиции по вертикали VerticalAnchorAlignment

6.162 Класс WorkbookExportSettings

Класс WorkbookExportSettings предоставляет настройки, необходимые для экспорта табличных документов (см. Document.exportAs).

Описание полей класса WorkbookExportSettings представлено в таблице 81.

Таблица 81 – Описание полей класса `WorkbookExportSettings`

Поле	Описание
`WorkbookExportSettings.sheetNames`	Представляет коллекцию имен листов для экспорта, тип `VectorString`. Если коллекция пуста, экспортируются все листы.
`WorkbookExportSettings.printingScope`	Представляет область печати (весь документ, область печати, пользовательский диапазон и т. д.) `PrintingScope`.
`WorkbookExportSettings.pageProperties`	Представляют свойства страницы для выходного документа (высота и ширина страницы в пунктах pt) `PageProperties`.
`WorkbookExportSettings.scale`	Представляет масштаб экспорта выходного документа в процентах (например, 50,0%, 150,63%, 400,0% и т. д.).

Пример:

```
WorkbookExportSettings workbookSettings = new WorkbookExportSettings();
workbookSettings.sheetNames = new VectorString();
workbookSettings.sheetNames.Add("Лист2");
workbookSettings.printingScope = new
PrintingScope(PrintingScope.Type.PrintArea);
workbookSettings.pageProperties = new PageProperties(100, 200);
workbookSettings.scale = 90;
document.exportAs(filePath, ExportFormat.PDFA1, workbookSettings);
```

6.163 Исключения

6.163.1 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");
```

6.163.2 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");
```

6.163.3 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");
```

6.163.4 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");
```

6.163.5 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");
```

6.163.6 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");
```

6.163.7 Класс ForbiddenActionError

Исключение `ForbiddenActionError` наследуется от `System.ApplicationException` и возникает в случае выполнения запрещенной операции.

Пример:

```
throw new DocumentAPI.ForbiddenActionError("Forbidden action");
```

6.163.8 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");
```

6.163.9 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");
```

6.163.10 Класс NoSuchElementError

Исключение `NoSuchElementError` наследуется от `System.ApplicationException` и возникает в случае, когда элемент не существует.

Пример:

```
throw new DocumentAPI.NoSuchElementError("Element not exists");
```

6.163.11 Класс NotImplementedError

Исключение `NotImplementedError` наследуется от `System.ApplicationException` и возникает в случае, если обнаружена нереализованная функциональность.

Пример:

```
throw new DocumentAPI.NotImplementedError("Not implemented");
```

6.163.12 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");
```

6.163.13 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");
```

6.163.14 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");
```

6.163.15 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");
```

6.163.16 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");
```

6.163.17 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 ИСПОЛЬЗОВАНИЕ КОДИРОВОК

Некоторые методы принимают текстовые параметры в формате unicode string. При этом наличие двухбайтовых символов (например, кириллица) приводит к возникновению исключения DocumentAP.UnknownError с сообщением «Invalid UTF-8». В этом случае рекомендуется использовать функцию кодирования, например, такую, как описана ниже:

```
public static string win1251ToUnicode(string value) {
    System.Text.Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

    Encoding windows = Encoding.Default;
    Encoding unicode = Encoding.Unicode;
    Encoding sp = Encoding.GetEncoding(1251);

    if (sp != null && !String.IsNullOrEmpty(value)) {
        // First get bytes in windows encoding
        byte[] wbytes = windows.GetBytes(value);

        // Check if CodePage to use is different from current Windows one
        if (windows.CodePage != sp.CodePage) {
            // Convert to Unicode using SP code page
            byte[] ubytes = Encoding.Convert(sp, unicode, wbytes);
            return unicode.GetString(ubytes);
        } else {
            // Directly convert to Unicode using windows code page
            byte[] ubytes = Encoding.Convert(windows, unicode, wbytes);
            return unicode.GetString(ubytes);
        }
    } else {
        return value;
    }
}
```

Пример загрузки документа:

```
string filePath = win1251ToUnicode("C:/Рабочий каталог/spreadsheet.xlsx");
var document = application.loadDocument(filePath);
```

Пример сохранения документа:

```
string filePath = win1251ToUnicode("C:/Рабочий каталог/spreadsheet.xlsx");
document.saveAs(filePath, saveDocumentSettings);
```

Пример экспорта документа:

```
string filePath = win1251ToUnicode("C:/Рабочий каталог/spreadsheet.pdf");
document.exportAs(filePath, ExportFormat.PDFA1, textExportSettings);
```

Пример заполнения ячейки таблицы:

```
string cellText = win1251ToUnicode("Түсіндіруде, дәлелде келтірілген ерекше жағдай");
table.getCell("A1").setText(cellText);
```

```
table.getCell("A1").setFormattedValue(cellText);
```

Пример вставки текста в документ:

```
var document = application.createDocument(DocumentType.Text);
string rangeText = win1251ToUnicode("Количество просмотров");
document.getRange().getBegin().insertText(rangeText);
```

Пример замены текста в диапазоне:

```
string filePath = win1251ToUnicode("C:/Рабочий каталог/Sample.docx");
var doc = application.loadDocument(filePath, loadSettings);
Range range = doc.getRange();
string rangeText = win1251ToUnicode("Набор переменных");
range.replaceText(rangeText);
```

8 КОНТРОЛЬ 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);
}
```