МойОфис

Комплект Средств Разработки (SDK)

Руководство программиста

MYOFFICE DOCUMENT API (C#)

2022.01

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)»

MYOFFICE DOCUMENT APPLICATION PROGRAMMING INTERFACE (API). БИБЛИОТЕКА ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#

РУКОВОДСТВО ПРОГРАММИСТА

2022.01

На 192 листах

Москва

2023

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. Таблица 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 программы

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/A-1.
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».

1.5 Ограничения

Библиотека MyOffice Document API для языка программирования C# предназначена для использования только в ОС Microsoft Windows.

2 ПОДГОТОВКА К РАБОТЕ

2.1 Дистрибутив

Дистрибутив MyOffice Document API поставляется в виде архивного файла **MyOffice_SDK_Document_API_CSharp_Win_2022.01_x64.zip**.

2.2 Установка

Для установки MyOffice Document API необходимо извлечь содержимое архивного файла дистрибутива в каталог установки MyOffice Document API (**C:\Project** по умолчанию).

После извлечения в каталоге установки MyOffice Document API будет создана папка **MyOfficeSDKDocumentAPI_CSharp_2022.1**, содержащая:

- каталог **Resources**, содержащий ресурсы приложения;
- файлы библиотек **DocumentAPI.dll**, **NCT.MyOfficeSDK.dll**, **libcrypto-ossl.dll**, **libcrypto-ossl.lib**, **libssl-ossl.dll**, **libssl-ossl.lib**, **sbb.dll**, **sbb.lib**;
- файл **EULA_ru.hml**, содержащий текст лицензионного соглашения на использование набора средств разработки «МойОфис SDK»;
- файл **TPL_ru.html**, содержащий перечисление отдельных компонентов приложения.

2.3 Сборка приложения

Сборка приложения может быть осуществлена с использованием IDE. 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 файла:

```
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");
            }
        }
    }
}
```

Для сборки и запуска тестового примера можно использовать IDE Microsoft Visual Studio.

2.3.1 Настройка и сборка приложения в среде Microsoft Visual Studio

В этом разделе описаны настройка и сборка проекта в среде Microsoft Visual Studio 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: **C:\Project**;
- имя решения: Example.

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

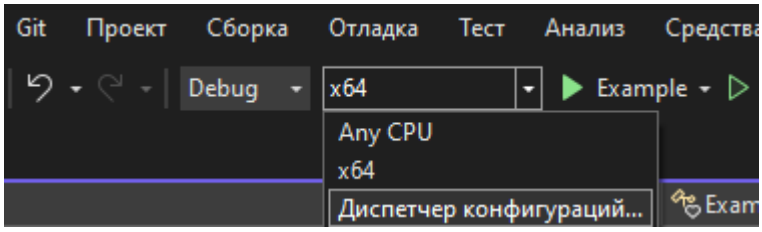


Рисунок 1 – Выбор диспетчера конфигурация в Microsoft Visual Studio

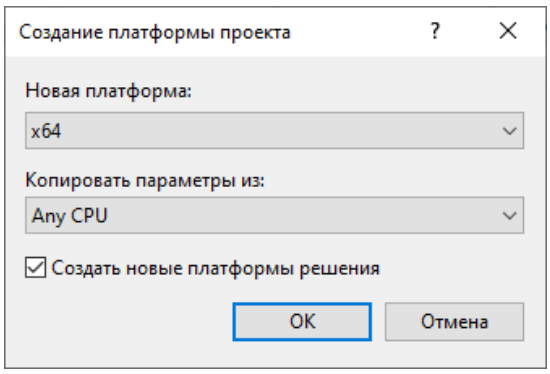


Рисунок 2 – Окно создания платформы проекта

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

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**. 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_2022.1** 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**.

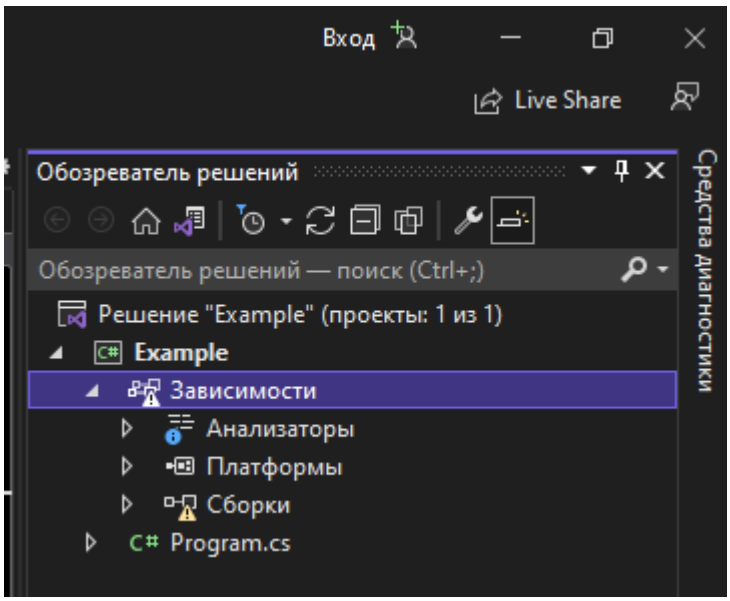


Рисунок 3 – Окно обозревателя решений Microsoft Visual Studio

В открывшемся окне нажать кнопку **Обзор** и выбрать из папки **MyOfficeSDKDocumentAPI_CSharp_2022.1** каталога установки MyOffice Document API файл **NCT.MyOfficeSDK.dll** (см. Рисунок 4).

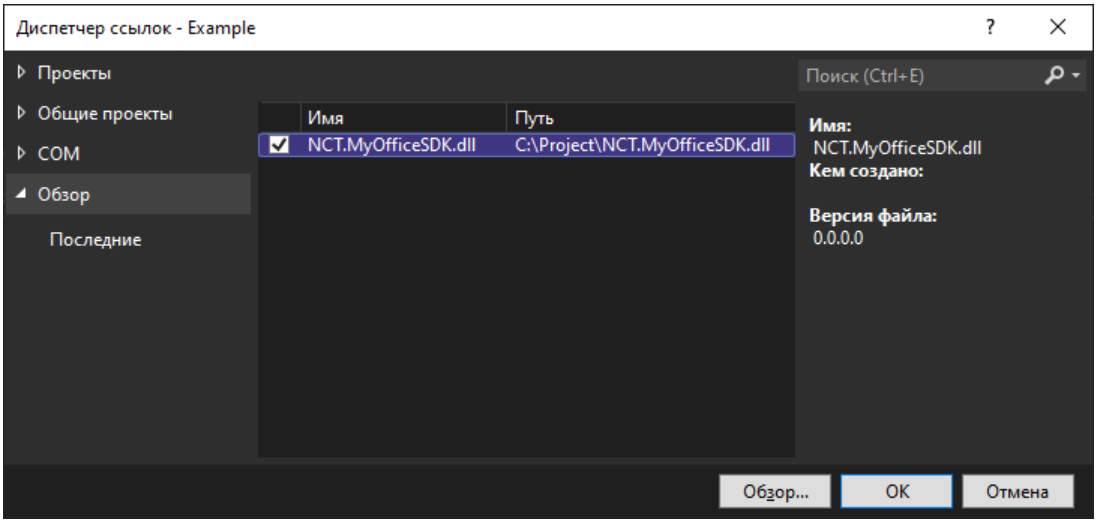


Рисунок 4 – Окно выбора файла библиотеки

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

2.4 Проверка работоспособности

Для проверки работоспособности MyOffice Document API необходимо произвести сборку приложения тестового примера в Microsoft Visual Studio в соответствии с разделом Настройка и сборка приложения а затем 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**) будет создан файл **Example.docx**, а в окне консоли отладки Microsoft Visual Studio отобразится сообщение от выполненного приложения, а также код его завершения.

MyOffice Document API считается работоспособным, если приложение выполнено успешно (код завершения равен нулю).

2.5 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_2022.1\Resources** каталога установки MyOffice Document API).
3. Файл динамической библиотеки **DocumentAPI.dll** (скопировать из папки **MyOfficeSDKDocumentAPI_CSharp_2022.1** каталога установки MyOffice Document API).
4. Файлы **NCT.MyOfficeSDK.dll**, **libcrypto-ossl.dll**, **libssl-ossl.dll, sbb.dll** (скопировать из папки **MyOfficeSDKDocumentAPI_CSharp_2022.1** каталога установки MyOffice Document API).

3 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Приведенные примеры предназначены для ознакомления с основными функциями MyOffice Document API и описывают варианты использования классов, методов, структур и функций при разработке приложений.

3.1 Работа с текстовым документом

3.1.1 Создание и сохранение документа

Для создания документа необходимо получить экземпляр объекта типа `Application` и вызвать метод `createDocument`, который вернет новый документ указанного в параметре типа с настройками по умолчанию.

Возможен также вызов метода `createDocument` с использованием в качестве параметра заданных свойств создаваемого документа.

```
// Инициализация сессии
Application application = new Application();

// Создание текстового документа
var doc = application.createDocument(DocumentType.Text);
```

Сохранение документа осуществляется с помощью метода `saveAs`, содержащегося в объектной модели документа:

```
// Вставка текста
doc.getRange().getBegin().insertText("Hello, Word!");

// Сохранение документа
var outputPath = "Example.docx";
doc.saveAs(outputPath);
```

3.1.2 Работа с текстом

3.1.2.1 Настройка свойств текста

Для настройки свойств текста необходимо создать объект `TextProperties` и инициализировать его поля.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
var doc = app.createDocument(DocumentType.Text);
```

```
// Вставка текста
doc.getRange().getBegin().insertText("Example Text");

var range = doc.getRange();

// Задание свойств текста
TextProperties txtProp = new TextProperties();
txtProp.bold = true;
ColorRGBA color = new ColorRGBA(255, 255, 0, 1);
txtProp.backgroundColor = color;

range.setTextProperties(txtProp);

// Сохранение документа
doc.saveAs("Example.docx");
```

3.1.3 Работа с таблицами

3.1.3.1 Вставка таблицы

Для вставки таблицы необходимо задать ее положение в документе. В примере таблица вставляется в начало документа. В случае успешного выполнения вставки возвращается объект таблицы.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
var doc = app.createDocument(DocumentType.Text);
var pos = doc.getRange().getBegin();

UInt32 rows = 5;
UInt32 columns = 3;
String tableName = "List1";

// Добавление таблицы
var table = pos.insertTable(rows, columns, tableName);
```

```
// Сохранение документа
doc.saveAs("Example.docx");
```

3.1.3.2 Удаление таблицы

Для удаления таблицы необходимо получить объект таблицы. Следует обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
var doc = app.createDocument(DocumentType.Text);
var pos = doc.getRange().getBegin();

UInt32 rows = 3;
UInt32 columns = 5;
String name = "List";

// Вставка таблицы
var table = pos.insertTable(rows, columns, name);

// Удаление таблицы
table.remove();

// Сохранение документа
doc.saveAs("Example.docx");
```

Объект таблицы также может быть получен из объекта типа `Document` по индексу таблицы. Следует обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

```
// Удаление таблицы
var table = doc.getBlocks().getTable(0);
table.remove();
```

3.1.4 Работа с ячейками таблицы

3.1.4.1 Объединение ячеек таблицы

Для объединения ячеек необходим объект `CellRange`. Объект `CellRange` имеет методы объединения и разъединения ячеек.

```
CellRangePosition cellRP = new CellRangePosition(1, 1, 3, 3);
var cellRange = table.getCellRange(cellRP);
cellRange.merge();
```

3.1.4.2 Разъединение ячеек таблицы

Для того чтобы разъединить ячейки, необходимо получить объект ячейки `Cell` из диапазона объединения ячеек. Объект `Cell` содержит метод для разъединения всех ячеек из диапазона объединения, частью которого является эта ячейка.

```
CellPosition cellPos = new CellPosition(1, 2);
var cell = table.getCell(cellPos);
cell.unmerge();
```

3.1.4.3 Поворот текста в ячейке

Для того чтобы задать поворот текста в ячейке, необходимо получить объект ячейки `CellProperties`. Объект `CellProperties` содержит поле `textOrientation`, в котором содержится ориентация текста ячейки.

```
CellProperties cellProps = new CellProperties();
TextOrientation txtOrient = new TextOrientation(90);
cellProps.textOrientation = txtOrient;
table.getCell("B2").setCellProperties(cellProps);
```

3.1.5 Форматирование таблицы

3.1.5.1 Установка свойств форматирования ячеек таблицы

Для форматирования ячеек таблицы необходимо задать набор ячеек для форматирования и применить к ним методы установки необходимых свойств.

```
CellProperties cellProps = new CellProperties();

cellProps.verticalAlignment = VerticalAlignment.Center;
cellProps.backgroundColor = new ColorRGBA(255, 0, 127, 1);

var tableEmployes = doc.getBlocks().getTable(tableName);

var cellOrder = tableEmployes.getCell(new CellPosition(0, 0));

cellOrder.setText("Some text");
```

```
cellOrder.setCellProperties(cellProps);
var cellName = tableEmployes.getCell(new CellPosition(0, 1));

cellName.setText("Another text");
cellName.setCellProperties(cellProps);
```

3.1.5.2 Установка границ ячеек таблицы

Для форматирования ячеек таблицы необходимо задать набор ячеек для форматирования, и применить к ним методы установки необходимых свойств, например, настройки внешних и внутренних границ.

```
// Инициализация сессии
Application app = new Application();

// Создание табличного документа
var doc = app.createDocument(DocumentType.Workbook);
var pos = doc.getRange().getBegin();

// Добавление листа в документ
var table = pos.insertTable(5, 5, "List");

var cellRangeOuter = table.getCellRange(new CellRangePosition(0, 0, 2, 2));

// Установка границ ячеек
LineProperties lineOuter = new LineProperties();
lineOuter.style = LineStyle.Solid;
lineOuter.width = 2.0f;
ColorRGBA color = new ColorRGBA(128, 0, 128, 1);
lineOuter.color = new Color(color);
RangeBorders borders = new RangeBorders();
cellRangeOuter.setBorders(borders.setOuter(lineOuter));

// Сохранение документа
doc.saveAs("Example.docx");
```

3.1.6 Работа с закладками

3.1.6.1 Вставка закладки

Для вставки в документ закладки необходимо задать ее расположение. В примере приведено позиционирование закладки в начале документа. Наименование закладки должно быть уникальным.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
var doc = app.createDocument(DocumentType.Text);

String bookmarkName = "Example";

var pos = doc.getRange().getBegin();

// Вставка закладки
pos.insertBookmark(bookmarkName);

// Сохранение документа
doc.saveAs("Example.docx");
```

3.1.6.2 Изменение содержимого закладки

Для изменения в документе закладки необходимо задать ее наименование. Наименование закладки должно быть уникальным.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
var doc = app.createDocument(DocumentType.Text);
var pos = doc.getRange().getBegin();

var bmk = "Executor";
pos.insertBookmark(bmk);

var collectionBookmarks = doc.getBookmarks();
var rangeExecutor = collectionBookmarks.getBookmarkRange(bmk);
var txtExecutor = "Исполнитель: Иванов И.И.";
```

```
// Изменение содержимого закладки
rangeExecutor.replaceText(txtExecutor);

// Сохранение текстового документа
doc.saveAs("Example.docx");
```

3.1.6.3 Удаление закладки

Для удаления закладки из документа используется метод объектной модели документа `removeBookmark(name)`, где `name` - наименование удаляемой закладки.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
var doc = app.createDocument(DocumentType.Text);
var pos = doc.getRange().getBegin();

var bmk = "Executor";
pos.insertBookmark(bmk);

var collectionBookmarks = doc.getBookmarks();
var range = collectionBookmarks.getBookmarkRange(bmk);

// Удаление закладки
collectionBookmarks.removeBookmark(bmk);

// Сохранение текстового документа
doc.saveAs("Example.docx");
```

3.1.7 Работа с комментариями

3.1.7.1 Получение списка комментариев

```
var commentEnumerator = document.getComments().GetEnumerator();
foreach (var comment in commentEnumerator)
{
    Console.WriteLine(comment.getText());
    Console.WriteLine(comment.getInfo().author.name);
```

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

3.1.7.2 Получение списка ответов

```
var commentEnumerator = document.getComments().GetEnumerator();
foreach (var comment in commentEnumerator)
{
    var repliesEnumerator = comment.getReplies().GetEnumerator();
    foreach (var reply in repliesEnumerator)
    {
        Console.WriteLine(reply.getText());
        Console.WriteLine(reply.getInfo().author).name);
    }
}
```

3.1.8 Экспорт текстового документа

Экспорт текстового документа в формат PDF/A-1.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
var doc = app.createDocument(DocumentType.Text);

// Заполнение текстового документа
doc.getRange().getBegin().insertText("Example Text");

// Экспорт текстового документа в формат PDF
doc.exportAs("Example.pdf", ExportFormat.PDFA1);
```

3.1.9 Поиск в текстовом документе

Для поиска в текстовом документе необходимо с помощью глобальной функции `createSearch` создать экземпляр объекта `Search` и вызвать метод `findText`.

```
// Инициализация сессии
Application app = new Application();

// Создание текстового документа
```

```
var doc = app.createDocument(DocumentType.Text);
var pos = doc.getRange().getBegin();

String txt = "API documentation describes what services an API " +
             "offers and how to use those services, " +
             "aiming to cover everything a client would need to " +
             "know for practical purposes." +
             "Documentation is crucial for the development and "+
             "maintenance of applications using the API.";

// Заполнение текстового документа
doc.getRange().getBegin().insertText(txt);

// Поиск текста
var txtSearch = DocumentAPI.createSearch(doc);
var collect = txtSearch.findText("API");

// Применение свойства BOLD к найденному фрагменту
TextProperties txtProps = new TextProperties();
txtProps.bold = true;

foreach (var range in collect)
{
    range.setTextProperties(txtProps);
}

// Сохранение текстового документа
doc.saveAs("Example.docx");
```

3.1.10 Управление ориентацией и свойствами страниц раздела

Для установки ориентации страницы можно использовать метод `setPageOrientation` объекта типа `Section`, который может быть получен из блока документа.

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

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

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

var section = doc.getBlocks().getBlock(0).getSection();
section.setPageProperties(prop);
```

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

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

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

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

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

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

3.1.11 Работа с отслеживаемыми изменениями

```
// Получение коллекции отслеживаемых изменений из всего диапазона документа
var trackedChangesEnumerator =
document.getRange().getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator)
{
    // Поля структуры TrackedChange:
    Console.WriteLine(trackedChange.getType().ToString());
    Console.WriteLine(trackedChange.getInfo().author.name);
```

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

3.1.12 Работа с колонтитулами раздела

Для получения колонтитулов раздела следует использовать методы `getHeaders` или `getFooters` объекта `Section`.

```
// Инициализация сессии
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());
```

3.2 Работа с табличным документом

3.2.1 Создание и сохранение документа

Для создания документа необходимо получить экземпляр объекта типа `Application` и вызвать метод `createDocument`, который вернет новый документ указанного в параметре типа с настройками по умолчанию. Возможен также вызов метода `createDocument` с использованием в качестве параметра заданных свойств создаваемого документа.

```
// Инициализация сессии
Application app = new Application();
// Создание табличного документа
var doc = app.createDocument(DocumentType.Workbook);
```

Сохранение документа осуществляется с помощью метода `saveAs`, содержащегося в объектной модели документа.

```
var pos = doc.getRange().getBegin();

UInt32 rows = 5;
UInt32 colunms = 3;

// Добавление листа табличного документа
var table = pos.insertTable(rows, colunms, "List");

// Сохранение документа в формате XLSX
doc.saveAs("Example.xlsx");
```

3.2.2 Работа с текстом

3.2.2.1 Настройка свойств текста

Для настройки свойств текста необходимо создать объект `TextProperties` и инициализировать его поля.

```
TextProperties txtProps = new TextProperties();
txtProps.bold = true;
txtProps.textColor = new ColorRGBA(255, 0, 0, 1);
var para = doc.getBlocks().getParagraph(0);
var range = para.getRange();
range.setTextProperties(txtProps);
```

3.2.3 Работа с листами табличного документа

3.2.3.1 Вставка рабочего листа в табличный документ

Для вставки листа необходимо иметь позицию вставки. В примере используется позиция в конце документа. Объект таблицы должен быть возвращен в случае успешного завершения операции вставки.

```
var pos = doc.getRange().getEnd();
var table = pos.insertTable(10, 10, "New List");
```

3.2.3.2 Удаление рабочего листа табличного документа

Для удаления листа документа предварительно его можно получить по названию. Следует обратить внимание на то, что объект листа перестает существовать после удаления из табличного документа.

```
var table = doc.getBlocks().getTable(tableName);
table.remove();
```

Объект листа также может быть получен из объекта `Document` по индексу листа. Индексация листов начинается с нуля. Необходимо обратить внимание на то, что объект листа перестает существовать после удаления из документа.

```
var table = doc.getBlocks().getTable(0);
table.remove();
```

3.2.4 Работа с ячейками таблицы

3.2.4.1 Настройка свойств ячейки

Для настройки свойств ячейки в общем случае необходимо создать объект `CellProperties` и инициализировать необходимые поля.

```
// Инициализация сессии
Application app = new Application();

// Создание табличного документа
var doc = app.createDocument(DocumentType.Workbook);

var pos = doc.getRange().getBegin();

UInt16 rows = 5;
UInt16 columns = 3;
String tableName = "List1";

// Добавление листа табличного документа
var table = pos.insertTable(rows, columns, tableName);

CellProperties cellProps = new CellProperties();

// Задание цвета фона и текста в ячейке
ColorRGBA color = new ColorRGBA(200, 200, 15, 1);
cellProps.backgroundColor = color;

table.getCell("B2").setCellProperties(cellProps);
table.getCell("B2").setText("Example");
```

```
// Сохранение документа в формате XLSX
doc.saveAs("Example.xlsx");
```

Пример задания поворота текста в ячейке:

```
// Инициализация сессии
Application app = new Application();

// Создание табличного документа
var doc = app.createDocument(DocumentType.Workbook);

var pos = doc.getRange().getBegin();

// Добавление листа табличного документа
var table = pos.insertTable(10, 10, "List");

// Вставка содержимого документа
string txt = "API documentation describes whatservices an " +
             "API offers and how to use those services, " +
             "aiming to cover everything a client would " +
             "need to know for practical purposes.";

table.getCell("D5").setText(txt);

CellProperties cellProps = new CellProperties();
cellProps.textOrientation = new TextOrientation(90);
table.getCell("D5").setCellProperties(cellProps);

// Сохранение документа в формате XLSX
doc.saveAs("Example.xlsx");
```

Пример установки выравнивания текста абзаца по ширине ячейки:

```
// Инициализация сессии
Application app = new Application();

// Создание табличного документа
var doc = app.createDocument(DocumentType.Workbook);

var pos = doc.getRange().getBegin();
```

```
// Добавление листа табличного документа
var table = pos.insertTable(10, 10, "List");

// Вставка содержимого документа
string txt = "API documentation describes whatservices an " +
             "API offers and how to use those services, " +
             "aiming to cover everything a client would " +
             "need to know for practical purposes.";


table.getCell("D5").setText(txt);

// Установка выравнивания
ParagraphProperties paraProps = new ParagraphProperties();
paraProps.alignment = Alignment.Justify;
table.getCell("D5").setParagraphProperties(paraProps);

// Сохранение документа в формате XLSX
doc.saveAs("Example.xlsx");}
```

3.2.4.2 Объединение ячеек таблицы

Для объединения ячеек необходим объект `CellRange`. Объект `CellRange` имеет методы объединения и разъединения ячеек.

```
CellRangePosition cellRP = new CellRangePosition(1, 1, 3, 3);
var cellRange = table.getCellRange(cellRP);
cellRange.merge();
```

3.2.4.3 Разъединение ячеек таблицы

Для того чтобы разъединить ячейки, необходимо получить объект ячейки `Cell` из диапазона объединения ячеек. Объект `Cell` содержит метод для разъединения всех ячеек из диапазона объединения, частью которого является эта ячейка.

```
CellPosition cellPos = new CellPosition(1, 2);
var cell = table.getCell(cellPos);
cell.unmerge();
```

3.2.5 Экспорт табличного документа

Экспорт табличного документа в формат PDF/A-1.

```
// Инициализация сессии
Application app = new Application();

// Создание табличного документа
var doc = app.createDocument(DocumentType.Workbook);
var pos = doc.getRange().getBegin();

UInt32 rows = 5;
UInt32 colunms = 3;

// Добавление листа табличного документа
var table = pos.insertTable(rows, colunms, "List");

// Вставка содержимого документа
var txtA1 = "API documentation describes what services an API " +
            "offers and how to use those services, aiming to " +
            "cover everything a client would need to know for " +
            "practical purposes.";

var txtA2 = "Documentation is crucial for the development and " +
            "maintenance of applications using the API.";

var txtA3 = "API documentation is traditionally found in " +
            "documentation files but can also be found in social " +
            "media such as blogs, forums, and Q&A websites.";

table.getCell("A1").setText(txtA1);
table.getCell("A2").setText(txtA2);
table.getCell("A3").setText(txtA3);

// Экспорт документа в формате PDF
doc.exportAs("Example.pdf", ExportFormat.PDFA1);
```

3.2.6 Поиск в табличном документе

Для поиска в табличном документе необходимо с помощью глобальной функции `createSearch` создать для документа экземпляр объекта `Search` и вызвать метод `findText`.

```
// Инициализация сессии
Application app = new Application();

// Создание табличного документа
var doc = app.createDocument(DocumentType.Workbook);
var pos = doc.getRange().getBegin();

UInt32 rows = 5;
UInt32 colunms = 3;

// Добавление листа табличного документа
var table = pos.insertTable(rows, colunms, "List");

// Вставка содержимого документа
var txtA1 = "API documentation describes what services an API " +
            "offers and how to use those services, aiming to " +
            "cover everything a client would need to know for " +
            "practical purposes.";
var txtA2 = "Documentation is crucial for the development and " +
            "maintenance of applications using the API.";
var txtA3 = "API documentation is traditionally found in " +
            "documentation files but can also be found in social " +
            "media such as blogs, forums, and Q&A websites.";

table.getCell("A1").setText(txtA1);
table.getCell("A2").setText(txtA2);
table.getCell("A3").setText(txtA3);

// Поиск текстового фрагмента в документе
var txtSearch = DocumentAPI.createSearch(doc);
var collect = txtSearch.findText("API");

// Установка сввойства тескта для найденных фрагментов
```

```
TextProperties txtProps = new TextProperties();
txtProps.bold = true;

foreach (var range in collect)
{
    // Установка начертания "полужирный"
    range.setTextProperties(txtProps);
}

// Сохранение документа в формате XLSX
doc.saveAs("Example.xlsx");
```

3.2.7 Работа со сводными таблицами

3.2.7.1 Получение диапазона исходных данных сводной таблицы

```
// Загружаем документ, содержащий сводную таблицу
var document = loadDocument();

// Получаем ячейку, находящуюся в диапазоне исходных данных сводной таблицы
var pivotRootCell =
              document.getBlocks().getTable(1).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());
```

3.2.7.2 Получение диапазона размещения сводной таблицы

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

3.2.7.3 Получение неподдерживаемых свойств сводной таблицы

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

3.2.7.4 Получение флагов отображения общих итогов для строк и колонок

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

3.2.7.5 Получение заголовков сводной таблицы

```
// Получение заголовков сводной таблицы
PivotTableCaptions captions = pivotTable.getPivotTableCaptions();

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

3.2.7.6 Получение и применение фильтра для сводной таблицы

```
// 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();
```

3.2.7.7 Получение полей из области фильтров

```
// 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);
}
```

3.2.7.8 Получение полей из области значений

```
// 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);
}
```

3.2.7.9 Получение полей из области строк

```
// 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);
}
```

3.2.7.10 Получение полей из области колонок

```
// 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);
}
```

3.2.7.11 Получение настроек отображения сводной таблицы

```
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);
```

3.2.7.12 Обновление сводной таблицы

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

3.2.8 Работа с именованными выражениями

3.2.8.1 Получение именованного выражения

```
// Загружаем документ, содержащий именованное выражение
var document = loadDocument();

// Получаем именованное выражение с именем “Age_of_Majority”
var ageOfMajority = document.getNamedExpressions().get("Age_of_Majority");
```

3.2.8.2 Получение именованного выражения таблицы

```
// Получаем именованное выражение таблицы с именем “Alice_Age”
var aliceAge =
     document.getBlocks().getTable(0).getNamedExpressions().get("Alice_Age");
```

3.2.8.3 Получение свойств именованного выражения

```
//  Получаем именованное выражение с именем "Alice_Age"
var expression =
     document.getBlocks().getTable(0).getNamedExpressions().get("Alice_Age");
//  Далее используем поля структуры NamedExpression:
var name = expression.getName();
var formula = expression.getExpression();
var range = expression.getCellRange();
```

3.2.8.4 Получение коллекции именованных выражений

```
// Получение коллекции именованных выражений
var enumerator =
        document.getBlocks().getTable(0).getNamedExpressions().getEnumerator();

// Перебор коллекции именованных выражений
foreach (var nameExpression in enumerator)
{
    // Использование полей структуры NamedExpression
    Console.WriteLine(nameExpression.getName());
    Console.WriteLine(nameExpression.getExpression());
    Console.WriteLine(nameExpression.getCellRange());
}
```

3.2.9 Работа со встроенными объектами

Доступ ко встроенным объектам документа осуществляется посредством использования метода `Range::getInlineObjects`.

3.2.9.1 Перечисление встроенных объектов

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
var inlineObjectEnumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in inlineObjectEnumerator)
{
    var image = inlineObject.toImage();
    if (image != null)
    {
        Console.WriteLine("Image");
    }
    else
    {
        Console.WriteLine("Shape");
    }
}
```

3.2.9.2 Изменение позиции встроенного объекта

С помощью объекта типа `Frame` можно задавать такие параметры встроенных объектов как размер, позиция и способ обтекания текстом.

```
// Позиция встроенного объекта не может быть задана,
// если стиль переноса текста - inline.
// Сначала его следует изменить на тип, отличный от inline.
var frame = inlineObject.getFrame();
var wrapType = frame.getWrapType();

if (wrapType == TextWrapType.Inline)
{
    frame.setWrapType(TextWrapType.TopAndBottom);
}
```

Используя классы `HorizontalTextAnchoredPosition`, `VerticalTextAnchoredPosition`, можно задать положение встроенных объектов в текстовом документе с учетом относительного смещения.

```
var frame = inlineObject.getFrame();

var position = new TextAnchoredPosition();

var horizontalPosition = new HorizontalTextAnchoredPosition
    (HorizontalRelativeTo.Page, 12.f);

var verticalPosition = new VerticalTextAnchoredPosition
    (VerticalRelativeTo.PageTopMargin, 122.f);

position.horizontal = horizontalPosition;
position.vertical = verticalPosition;

frame.setPosition(position);
```

Используя классы `HorizontalTextAnchoredPosition`, `VerticalTextAnchoredPosition`, можно задать положение встроенных объектов в текстовом документе с учетом относительного выравнивания.

```
var frame = inlineObject.getFrame();

var position = new TextAnchoredPosition();

var horizontalPosition = new HorizontalTextAnchoredPosition
(HorizontalRelativeTo.Page, HorizontalAnchorAlignment.Center);

var verticalPosition = new VerticalTextAnchoredPosition
(VerticalRelativeTo.PageTopMargin, VerticalAnchorAlignment.Top);

position.horizontal = horizontalPosition;
position.vertical = verticalPosition;

frame.setPosition(position);
```

Используя типы смещения `HorizontalRelativeTo.Column` и `VerticalRelativeTo.Page`, можно установить абсолютное положение встроенного объекта в текстовом документе.

```
var frame = inlineObject.getFrame();

var position = new TextAnchoredPosition();

position.horizontal =
    new HorizontalTextAnchoredPosition(HorizontalRelativeTo.Column, 125.f);
position.vertical =
    new VerticalTextAnchoredPosition(VerticalRelativeTo.Page, 345.f);

frame.setPosition(position);
```

С помощью метода `Frame::setDimensions` можно изменить размеры встроенных объектов

```
var frame = inlineObject.getFrame();

var size = new SizeU(300f, 400f);
frame.setDimensions(size);
```

4 СПРАВОЧНИК КЛАССОВ, СТРУКТУР И МЕТОДОВ

Далее приведено описание классов, структур и методов библиотеки MyOffice Document API для языка программирования C#.

4.1 Типы документов

4.1.1 Тип DocumentType

Тип `DocumentType` определяет поддерживаемые типы документов.

Поддерживаемые типы документов:

- `Text` – используется для работы с текстовыми документами – файлы DOCX, ODT, XODT, TXT;
- `Workbook` – используется для работы с табличными документами – файлы XLSX, ODS, XODS;
- `Presentation` – используется для работы с презентационными документами – файлы PPTX, ODP. Работа с презентационными документами средствами Document API в настоящий момент не поддерживается.

4.2 Форматы документов

4.2.1 Тип DocumentFormat

Тип `DocumentFormat` определяет поддерживаемые форматы документов.

Поддерживаемые форматы документов:

- `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 Dcoument Format (PDF), версии 1.4. Средствами Document API поддерживается только операция экспорта документа в формат PDF/A-1b;

- `PDFA` – используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b). Средствами Document API поддерживается только операция экспорта документа в формат PDF/A-1b.

При работе с файлами XODT или XODS (MyOffice eXtended Open Document Format) используйте значение `DocumentFormat.ODF`.

Средствами MyOffice Document API не поддерживается работа с текстовыми и табличными документами в двоичном формате Microsoft Word (DOC) и Microsoft Excel (XLS, XLSB), а также документами, содержащими макрокоманды VBA (DOCM, XLSM).

4.3 Форматы экспорта документов

4.3.1 Тип ExportFormat

Тип `ExportFormat` определяет поддерживаемые форматы документов для экспорта.

Формат `PDFA1` используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

Средствами Document API поддерживается только операция экспорта документа в формат PDF/A-1b.

4.4 Неподдерживаемые свойства документа SaveUnsupportedFeature

Имеются свойства документа, которые могут быть утеряны при сохранении документа. Тип `SaveUnsupportedFeature` перечисляет все возможные неподдерживаемые свойства.

Варианты неподдерживаемых свойств документа:

- `CommentsInHeaderFooter` – комментарии в колонтитулах;
- `CommentsInFootnote` – комментарии в сносках;
- `MixedNoteAlignment` – параграфы с разным выравниванием в заметках.

4.5 Системы адресации ячеек

4.5.1 Тип FormulaType

Тип `FormulaType` определяет поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе.

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

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

4.6 Кодировки документов

4.6.1 Тип Encoding

Тип `Encoding` определяет поддерживаемые кодировки документов:

- `Unknown`;
- `UTF8`;
- `UTF16BE`;
- `UTF16LE`;
- `UTF32BE`;
- `UTF32LE`;
- `Windows1250`;
- `Windows1251`;
- `Windows1252`;
- `ISO8859Part5`;
- `KOI8R`;
- `KOI8U`;
- `CP866`.

4.7 Типы выравнивания текста

4.7.1 Тип Alignment

Тип `Alignment` содержит все варианты горизонтального выравнивания текста, в том числе в ячейке таблицы.

Варианты горизонтального выравнивания текста:

- `Default` – выравнивание текста по умолчанию;
- `Left` – выравнивание текста по левому краю;
- `Center` – выравнивание текста по центру;
- `Right` – выравнивание по правому краю;
- `Justify` – выравнивание по ширине;
- `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;
- `Fill` – распределение текста по горизонтали – заполнение строки текстом.

4.7.2 Тип TextLayout

Тип `TextLayout` содержит варианты форматирования текста в ячейке таблицы.

Варианты форматирования текста:

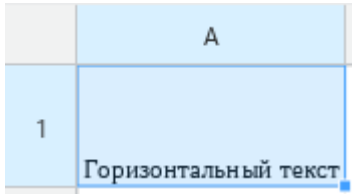
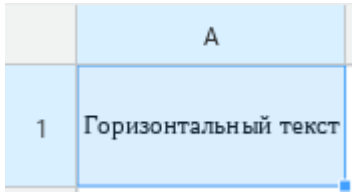
- `SingleLine` – текст располагается в одной строке (если текст длиннее объекта, содержащего его, то текст просто перекрывается с соседними объектами);
- `WrapByWords` – текст выравнивается по высоте объекта, содержащего его, при этом высота объекта может быть увеличена до необходимого размера;
- `ShrinkSizeToFitWidth` – размер текста выбирается таким образом, что текст соответствует ширине объекта, в котором он находится.

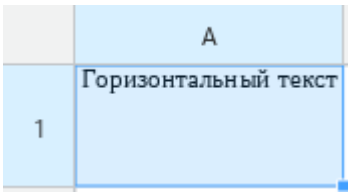
4.7.3 Тип VerticalAlignment

Тип `VerticalAlignment` содержит варианты выравнивания текста в ячейке по вертикали.

Варианты выравнивания текста в ячейке по вертикали представлены в Таблице 2.

Таблица 2 - Варианты выравнивания текста в ячейке по вертикали

Тип выравнивания	Представление в интерфейсе	
`Bottom`		
`Center`		

Тип выравнивания	Представление в интерфейсе	
Top		

4.7.4 Тип TextWrapType

Тип `TextWrapType` содержит варианты обтекания текстом встроенного объекта (изображения, формы и т. д.).

Варианты обтекания текстом:

- `Inline` – встроенный объект располагается в тексте;
- `InFrontOfText` – встроенный объект располагается перед текстом;
- `BehindText` – встроенный объект располагается за текстом;
- `TopAndBottom` – текст располагается сверху и снизу встроенного объекта;
- `Square` – текст располагается вокруг прямоугольной рамки встроенного объекта;
- `Through` – текст обтекает встроенный объект по сторонам и внутри;
- `Tight` – текст обтекает встроенный объект по сторонам.

4.8 Стили линий

4.8.1 Тип LineStyle

Тип `LineStyle` содержит доступные стили линий для оформления границ ячейки.

Варианты стилей линий:

- `NoLine` – нет границ;
- `Solid` – обычная линия;
- `Dot` – пунктирная линия;
- `Dash` – штриховая линия;
- `LongDash` – разомкнутая линия;
- `DashDot` – штрихпунктирная линия;
- `DotDotDash` – штрихпунктирная линия с двумя точками;
- `Double` – двойная линия;

- `Wave` – волнистая линия.

4.9 Типы надстрочного и подстрочного форматирования

4.9.1 Тип ScriptPosition

Тип `ScriptPosition` содержит варианты форматирования отдельного символа в тексте – надстрочный знак, подстрочный знак или обычный символ.

- SuperScript;
- SubScript;
- NormalScript.

4.10 Типы форматов ячеек

4.10.1 Тип CellFormat

Тип `CellFormat` содержит поддерживаемые форматы ячеек таблицы.

Варианты форматов ячеек:

- `General` – общий, например: 12;
- `Percentage` – процентный, например: 120,00%;
- `Number` – числовой, например: 12,00;
- `Text` – текстовый, например: 12;
- `Currency` – денежный, например: 12 000,00₽;
- `Accounting` – финансовый, например: 12 000,00₽;
- `Date` – дата, например: 01.01.2020;
- `Time` – время, например: 0:00:00;
- `DateTime` – дата и время, например: 12.12.2020 0:00:00;
- `Fraction` – дробный, например: 12 1/5;
- `Scientific` – экспоненциальный, например: 1,22E+01;
- `Custom` – пользовательский формат.

Примеры использования:

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

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

4.10.2 Класс AccountingCellFormatting

Класс `AccountingCellFormatting` содержит параметры для финансового формата ячеек таблицы.

Параметры финансового формата ячейки:

- `decimalPlaces` – количество десятичных позиций;
- `symbol` – символ денежной единицы;
- `localeCode` – идентификатор кода языка (MS-LCID);
- `fillSymbol` – символ заполнения;
- `useThousandsSeparator` – использовать разделитель для тысячных;
- `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());
```

4.10.3 Класс PercentageCellFormatting

Класс `PercentageCellFormatting` содержит параметры для процентного формата ячеек таблицы.

Свойство класса:

- `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());
```

4.10.4 Класс NumberCellFormatting

Класс `NumberCellFormatting` содержит параметры для числового формата ячеек таблицы.

Свойства класса:

- `decimalPlaces` – количество десятичных позиций;
- `useThousandsSeparator` – использовать разделитель для тысячных;
- `useRedForNegative` – использовать красный цвет для отрицательных значений;
- `useBracketsForNegative` – использовать скобки для отрицательных значений;
- `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());
```

4.10.5 Класс CurrencyCellFormatting

Класс `CurrencyCellFormatting` содержит параметры для денежного формата ячеек таблицы.

Свойства класса:

- `decimalPlaces` – количество десятичных позиций;

- `symbol` – символ денежной единицы;
- `localeCode` – идентификатор кода языка (MS-LCID);
- `useThousandsSeparator` – использовать разделитель для тысячных.

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 = new CurrencySignPlacement.Suffix;

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

4.10.6 Класс DateTimeCellFormatting

Класс `DateTimeCellFormatting` содержит параметры для формата ячеек таблицы типа Дата и Время.

Свойства класса:

- `dateListID` – формат даты;
- `timeListId` – формат времени.

Пример:

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

DateTimeCellFormatting cellFormat = new DateTimeCellFormatting();
cellFormat.dateListID = DatePatterns.DayMonthNumberLongYearShort;
cellFormat.timeListID = TimePatterns.LongTime;

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

4.10.6.1 Тип DatePatterns

Тип содержит перечисление форматов даты.

Форматы даты:

- `DayMonthTextLongYearLong` – 'mmmm dd, yyyy' для языка en_US;
- `FullDate` - 'день недели, mmmm dd, yyyy' для языка en_US;
- `DayMonthNumberLongYearLong` - 'mm/dd/yyyy' для языка en_US;
- `DayMonthNumberLongYearShort` - 'm/dd/yy' для языка en_US;
- `DayMonthNumberShortYearShort` - 'dd-mmm' для языка en_US;
- `DayMonthTextShort` - 'mmm-yy' для языка en_US;
- `DayMonthTextShortYearShort` - 'mmm dd, yy' для языка en_US;
- `DayMonthYearLongNonLocalizableHyphen` - нелокализуемый шаблон 'dd-mm-yyyy';
- `DayMonthYearLongNonLocalizableSlash` - нелокализуемый шаблон 'dd/mm/yyyy'.

4.10.6.2 Тип TimePatterns

Тип содержит перечисление форматов времени.

Форматы времени:

- `ShortTime` – 'hh:mm AM/PM' для языка en_US;
- `LongTime` - 'hh:mm:ss AM/PM' для языка en_US.

4.10.7 Класс FractionCellFormatting

Класс `FractionCellFormatting` представляет параметры для дробного формата ячеек таблицы.

Свойства класса:

- `minNumeratorDigits` – количество позиций числителя;
- `minDenominatorDigits` – количество позиций знаменателя;
- `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());
```

4.10.8 Класс ScientificCellFormatting

Класс `ScientificCellFormatting` представляет параметры для экспоненциального формата ячеек таблицы.

Свойства класса:

- `decimalPlaces` – количество десятичных позиций;
- `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());
```

4.11 Типы межстрочного интервала

4.11.1 Тип LineSpacingRule

Тип `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.

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);
}
```

4.12 Типы схем форматирования списков

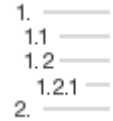
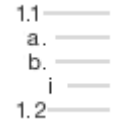
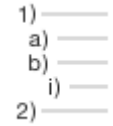
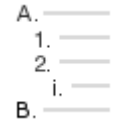
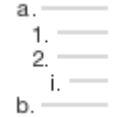
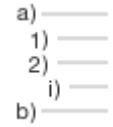
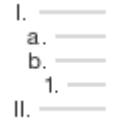
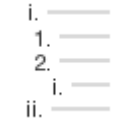
4.12.1 Тип ListSchema

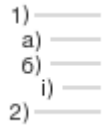
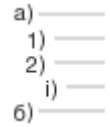
Тип `ListSchema` содержит типы схем форматирования списков, которые могут быть применены к абзацам текста.

Типы схем форматирования списков представлены в Таблице 3:

Таблица 3 - Типы схем списков

Тип схемы списка	Описание типа схемы списка	Изображение
`Unknown`	Неизвестно.	Не определено.
`UnknownBullet`	Список без маркера.	Соответствует варианту «нет».
`UnknownNumbering`	Нумерация без номера.	Соответствует варианту «нет».
`BulletCircleSolid`	Список с маркерами в виде круга.	
`BulletCircleContour`	Список с маркерами в виде окружности.	
`BulletSquareSolid`	Список с маркерами в виде квадрата.	
`BulletDiamondDots`	Список с маркерами в виде четырех ромбов.	

Тип схемы списка	Описание типа схемы списка	Изображение
BulletHyphen	Список с маркерами в виде дефиса.	
BulletConcaveArrowSolid	Список с маркерами в виде вогнутой стрелки.	
BulletCheckmark	Список с маркерами в виде галочки.	
EnumeratorDecimalDot	Десятичная нумерация с точкой.	
EnumeratorDecimalDotMultiLevel	Многоуровневая десятичная нумерация с точкой.	
EnumeratorDecimalBracket	Десятичная нумерация со скобкой.	
EnumeratorLatinUppercaseDot	Нумерация латинскими прописными буквами с точкой.	
EnumeratorLatinLowercaseDot	Нумерация латинскими строчными буквами с точкой.	
EnumeratorLatinLowercaseBracket	Нумерация латинскими строчными буквами со скобкой.	
EnumeratorRomanUppercaseDot	Нумерация римскими прописными цифрами с точкой.	
EnumeratorRomanLowercaseDot	Нумерация римскими строчными цифрами с точкой.	

Тип схемы списка	Описание типа схемы списка	Изображение
`EnumeratorDecimalRussianBracket`	Десятичная нумерация через запятую со скобкой.	
`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());
}
```

4.13 Типы отслеживаемых изменений

4.13.1 Тип TrackedChangeType

Тип `TrackedChangeType` содержит типы отслеживаемых изменений.

Типы отслеживаемых изменений:

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

Пример:

```
TrackedChangesEnumerator trackedChangesEnumerator =
document.getRange().getTrackedChangesEnumerator();
if (trackedChangesEnumerator != null) {
    while (trackedChangesEnumerator.MoveNext()) {
        TrackedChange trackedChange = trackedChangesEnumerator.Current;
        Console.WriteLine(trackedChange.getType());
    }
}
```

4.14 Типы колонтитулов

4.14.1 Тип HeaderFooterType

Тип `HeaderFooterType` содержит типы колонтитулов – верхний колонтитул (`Header`) и нижний колонтитул (`Footer`).

4.15 Варианты размещения знака валюты

4.15.1 Тип CurrencySignPlacement

Тип `CurrencySignPlacement` определяет варианты размещения знака валюты, до значения ($12.00), либо после (12,00 ₽).

Тип `CurrencySignPlacement` включает следующие свойства:

- `Prefix` – знак валюты, размещаемый до значения;
- `Suffix` – знак валюты, размещаемый после значения.

4.16 Тип SectionBreakType

Тип `SectionBreakType` определяет тип начала следующей секции после вставки разрыва раздела.

Поддерживаются следующие типы разрывов разделов документа:

- `NextPage` – следующий раздел начинается с новой страницы;
- `Continious` – следующий раздел продолжается на текущесй странице без вставки разрыва страницы;
- `EvenPage` – следующий раздел начинается на ближайшей четной странице;
- `OddPage` – следующий раздел начинается на ближайшей нечетной странице.

4.17 Варианты ориентации страницы

4.17.1 Тип PageOrientation

Тип `PageOrientation` определяет варианты ориентации страницы документа: Альбомная (`Landscape`) или Книжная (`Portrait`).

4.18 Масштабирование при печати табличных документов

4.18.1 Тип WorksheetPrinterFitType

Тип `WorksheetPrinterFitType` содержит варианты масштабирования при печати табличных документов.

Варианты масштабирования:

- `ActualSize` – фактический размер;

- `ByPageScale` – по масштабу страницы;
- `ByPageBreaksOnly` – по разрыву страниц;
- `FitToPage` – вписать в страницу;
- `FitToWidth` – вписать по ширине;
- `FitToHeight` – вписать по высоте.

4.19 Выбор страниц для экспорта и печати

4.19.1 Тип PageParity

Тип `PageParity` содержит варианты отбора страниц для экспорта, печати документов.

Типы страниц:

- `Odd` – печать только нечетных страниц;
- `Even` – печать только четных страниц;
- `All` – печать всех страниц.

4.20 Типы коллекций

4.20.1 Тип VectorUInt

Тип `VectorUInt` представляет собой коллекцию типа `IEnumerable` данных типа `uint`. Используется для представления коллекции страниц для экспорта (нечетные, четные, список и т. д.).

4.20.2 Тип VectorString

Тип `VectorString` представляет собой коллекцию типа `IEnumerable` данных типа `string`. Используется в качестве контейнера имен листов для печати, экспорта и т. д.

4.21 Типы идентификаторов цветов тем

4.21.1 Тип ThemeColorID

Тип `ThemeColorID` содержит идентификаторы цветов темы.

Типы идентификаторов цветов тем:

- `Background1` – Фон1, значение по умолчанию 0;
- `Text1` – Текст1;
- `Background2` – Фон2;
- `Text2` – Текст2;
- `Dark1` – Темная1;

- `Dark2` – Темная2;
- `Light1` – Светлая1;
- `Light2` – Светлая2;
- `Accent1` – Акцент1;
- `Accent2` – Акцент2;
- `Accent3` – Акцент3;
- `Accent4` – Акцент4;
- `Accent5` – Акцент5;
- `Accent6` – Акцент6;
- `Hyperlink` – Гиперссылка;
- `FollowedHyperlink` – Следующая гиперссылка.

4.22 Типы окончаний линий

4.22.1 Тип LineEndingStyle

Тип `LineEndingStyle` содержит типы окончаний линий.

Типы окончаний линий представлены в Таблице 4:

Таблица 4 - Типы окончаний линий

Тип окончаний линии	Изображение
`Arrow`	
`Diamond`	
`Oval`	
`Stealth`	
`Triangle`	
`None`	

4.23 Типы размещения объекта по горизонтали

4.23.1 Тип HorizontalRelativeTo

Тип `HorizontalRelativeTo` содержит все типы размещения объекта относительно закрепленной позиции по горизонтали.

Типы размещения:

- `Character` – символ;
- `Column` – столбец;
- `ColumnLeftMargin` – левое поле столбца;
- `ColumnRightMargin` – правое поле столбца;
- `ColumnInsideMargin` – внутреннее поле столбца;
- `ColumnOutsideMargin` – внешнее поле столбца;
- `Page` – страница;
- `PageContent` – содержимое страницы;
- `PageLeftMargin` – левое поле страницы;
- `PageRightMargin` – правое поле страницы;
- `PageInsideMargin` – внутреннее поле страницы;
- `PageOutsideMargin` – внешнее поле страницы.

4.24 Типы размещения объекта по вертикали

4.24.1 Тип VerticalRelativeTo

Тип `VerticalRelativeTo` содержит все типы размещения объекта относительно закрепленной позиции по вертикали.

Типы размещения:

- `Character` – символ;
- `BaseLine` – базовая линия;
- `Paragraph` – абзац;
- `Page` – страница;
- `PageContent` – содержимое страницы;
- `PageTopLeftMargin` – верхнее поле страницы;
- `PageBottomMargin` – нижнее поле страницы;
- `PageInsideMargin` – внутреннее поле страницы;
- `PageOutsideMargin` – внешнее поле страницы.

4.25 Типы выравнивания объекта по вертикали

4.25.1 Тип VerticalAnchorAlignment

Тип `VerticalAnchorAlignment` содержит все типы выравнивания объекта относительно закрепленной позиции по вертикали.

Типы выравнивания:

- `Top` – по верхнему краю;
- `Bottom` – по нижнему краю;
- `Center` – по центру;
- `Inside`, `Outside` – по границам.

4.26 Типы выравнивания объекта по горизонтали

4.26.1 Тип HorizontalAnchorAlignment

Тип `HorizontalAnchorAlignment` содержит варианты выравнивания объекта относительно закрепленной позиции по горизонтали.

Типы выравнивания:

- `Left` – по левому краю;
- `Right` – по правому краю;
- `Center` – по центру;
- `Inside`, `Outside` – по границам.

4.27 Классы и структуры, относящиеся к ядру (Core Types)

4.27.1 Класс Application

Управляет параметрами и объектами приложения. Предоставляет интерфейс для создания и загрузки документов. Как правило, создается только один объект класса для всего сеанса работы с документом.

4.27.1.1 Метод Application

Конструктор класса. Получает путь к ресурсам приложения. Если путь не указан, поиск ресурсов производится в подкаталоге `Resources` текущего каталога. При невозможности создания экземпляра класса выбрасывает исключение `ApplicationCreateError`.

Конструктор по умолчанию.

```
Application();
```

Конструктор с указанием пути к ресурсам приложения.

```
Application(string resourcePath);
```

4.27.1.2 Метод createDocument

Создает новый документ указанного в параметрах типа с настройками по умолчанию или с указанными в параметре `settings`. Список поддерживаемых типов документов

приведен в разделе DocumentType. Настройки документа приведены в разделе DocumentSettings.

При невозможности создания документа вызывается исключение DocumentCreateError.

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

4.27.1.3 Метод loadDocument

Загружает существующий текстовый или табличный документ из файла, находящегося по указанному пути `filePath`. Формат и тип документа определяются из расширения файла, если они не указаны явно с помощью параметра `settings`. Настройки описаны в разделе LoadDocumentSettings.

При невозможности загрузки файла выбрасывает исключение DocumentLoadError.

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

4.27.1.4 Метод getMessenger

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

4.27.2 Класс Messenger

Класс `Messenger` предоставляет возможность работы по логированию событий.

4.27.2.1 Метод subscribe

Метод `subscribe` служит для подписки на события лога.

```
Connection subscribe(MessageHandler handler);
```

4.27.2.2 Метод notify

Метод `notify` используется для создания события лога.

```
void notify(Message message);
```

4.27.3 Класс Connection

Класс `Connection` реализует соединение между `Messenger` и клиентом. Содержит один метод `unsubscribe` для разрыва соединения.

4.27.4 Класс Message

Класс `Message` предназначен для формирования событий лога. Внутренний класс `Severity` описывает уровни сообщений лога (информация, предупреждение, ошибка):

- `Info`;
- `Warning`;
- `Error`.

4.27.4.1 Метод getSeverity

Метод возвращает уровень лога `Severity (Info, Warning, Error)`.

4.27.4.2 Метод getText

Метод возвращает текст сообщения.

4.27.4.3 Метод makeInfo

Метод создает сообщение типа `Info` с заданным текстом.

4.27.4.4 Метод makeWarning

Метод создает сообщение типа `Warning` с заданным текстом.

4.27.4.5 Метод makeError

Метод создает сообщение типа `Error` с заданным текстом.

4.27.5 Класс MessageHandler

Класс предназначен для обработки сообщений лога, метод `handle` вызывается при возникновении сообщения.

4.27.6 Класс DocumentSettings

Класс `DocumentSettings` предоставляет настройки, необходимые на протяжении всего периода работы с документом.

Список свойств:

- `userInfo` – информация о пользователе (см. раздел `UserInfo`);
- `documentType` - тип документа (см. раздел `DocumentType`);
- `localeInfo` – информация о локализации (см. раздел `LocaleInfo`);
- `timeZone` – информация о временной зоне (см. раздел `TimeZone`);
- `formulaType` – система адресации ячеек (см. раздел `FormulaType`).

4.27.7 Класс LoadDocumentSettings

Класс `LoadDocumentSettings` предоставляет дополнительные настройки, необходимые для загрузки документа из файла.

Список свойств:

- `commonDocumentSettings` – общие настройки документа (см. раздел `DocumentSettings`);
- `encoding` – кодировка документа (см. раздел `Encoding`);
- `dsvSettings` – настройки, необходимые для работы с файлами CSV (comma-separated value) и DSV (delimiter-separated value) (см. раздел `DSVSettings`);
- `documentPassword` – пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows.

4.27.8 Класс SaveDocumentSettings

Класс `SaveDocumentSettings` предоставляет дополнительные настройки, необходимые для сохранения документа в файл.

Список свойств:

- `documentFormat` – формат документа (см. раздел `DocumentFormat`);
- `documentType` - тип документа (см. раздел `DocumentType`);
- `documentPassword` – пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows;
- `isTemplate` - флаг, обозначающий, что документ должен быть сохранен как шаблон;
- `dsvSettings` – структура `DSVSettings`, необходимая для сохранения в формате DSV.

4.27.9 Класс UserInfo

Класс `UserInfo` предоставляет информацию о пользователе.

Список свойств:

- `name` – имя пользователя;
- `email` – почтовый адрес пользователя.

4.27.10 Класс LocaleInfo

Класс `LocaleInfo` предоставляет информацию о локализации.

Список свойств:

- `localeName` – название локализации представлено в формате <language> <REGION>, где языковой код соответствует стандарту ISO-639, а код региона стандарту ISO-3166, значение по умолчанию - «en_US»;
- `decimalSeparator` – десятичный разделитель, отделяющий целые и дробные части числа;
- `thousandSeparator` – символ, разделяющий группы цифр в числовых значениях;
- `listSeparator` – символ, который используется для разделения элементов списка, например, списка аргументов функции;
- `currencySymbol` – символ валюты;
- `currencyFormat` – расположение знака валюты в текущем регионе;
- `shortDatePattern` – заданный «короткий» формат отображения даты в текущем регионе (например, 'm/d/yy' для en_US);
- `longDatePattern` – заданный «длинный» формат отображения даты в текущем регионе (например, 'dddd, mmmm d, yyyy' для en_US);
- `timePattern` – заданный формат отображения времени в текущем регионе (например, 'h:mm AM/PM' для en_US).

4.27.11 Класс TimeZone

Класс `TimeZone` предоставляет информацию о часовом поясе. Содержит свойство `offsetInSecondsToUTC` с помощью которого задается смещение или разность между временем в данном часовом поясе и временем в формате UTC (Всемирное координированное время).

4.27.12 Класс DSVSettings

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

Список свойств:

- `separators` – символ-разделитель полей данных, значение по умолчанию - запятая;
- `escapeChar` – символ-разделитель для текстовых строк, значение по умолчанию - двойная кавычка;

- `autofit` – признак необходимости автоматического подстраивания ширины столбца под размер данных в ячейке, значение по умолчанию - `false`;
- `startBlockIndex` – поле содержит индекс блока документа для начала сохранения;
- `startBlockIndex` – поле содержит индекс блока документа для окончания сохранения.

4.27.13 Класс PointU

Класс `PointU` представляет собой точку в двухмерном пространстве.

Список свойств:

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

4.27.13.1 Метод PointU

Конструктор по умолчанию.

```
PointU();
```

Конструктор класса устанавливает положение точки в двухмерном пространстве.

```
PointU(float xArg, float yArg);
```

4.27.14 Класс SizeU

Класс `SizeU` описывает размер объекта в двухмерном пространстве.

Список свойств:

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

4.27.14.1 Метод SizeU

Конструктор по умолчанию.

```
SizeU();
```

Конструктор класса устанавливает размер объекта в двухмерном пространстве.

```
SizeU(float widthArg, float heightArg);
```

4.27.15 Класс RectU

Класс `RectU` описывает прямоугольника двухмерном пространстве.

Список свойств:

- `topLeft` – координата левой верхней вершины прямоугольника;
- `bottomRight` – координата правой нижней вершины прямоугольника.

4.27.15.1 Метод RectU

Конструктор по умолчанию.

```
RectU();
```

Конструктор класса устанавливает положение и размер прямоугольника в двухмерном пространстве.

```
RectU(PointU topLeftArg, PointU bottomRightArg);
RectU(float topLeftX, float topLeftY, float bottomRightX, float bottomRightY);
```

4.27.16 Класс ColorRGBA

Класс `ColorRGBA` позволяет настроить пользовательский цвет для оформления элементов документа, текста, границ таблиц и т. п. Для описания цвета используется расширенная цветовая модель RGB, позволяющая установить непрозрачность цвета.

Список свойств:

- `r` – значение компонента красного цвета;
- `g` – значение компонента зеленого цвета;
- `b` – значение компонента синего цвета;
- `a` – значение прозрачности цвета.

4.27.16.1 Метод ColorRGBA

Конструктор по умолчанию. По умолчанию компоненты красного, синего и зеленого цветов устанавливаются в значение 0 (черный цвет), значение прозрачности равно 0 (абсолютно прозрачный).

```
ColorRGBA();
```

Конструктор с параметрами устанавливает заданные значения для компонентов красного, синего и зеленого цветов, а также прозрачности.

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

4.27.17 Класс DateTime

Предоставляет дату и время с точностью до секунды.

Список свойств:

- `year` – год;

- `month` – месяц;
- `day` – день;
- `hour` – часы;
- `minute` – минуты;
- `second` – секунды.

4.28 Классы, структуры и методы объектной модели документа

4.28.1 Класс Document

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

Следующий пример предоставляет доступ к первому абзацу текстового документа:

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

4.28.1.1 Метод 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.

Примеры:

```
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);
```

4.28.1.2 Метод exportAs

Экспортирует документ в файл по указанному пути и с указанным форматом. В настоящее время поддерживается только операция экспорта документа в формат PDF/A-1b. При невозможности экспорта документа выбрасывает исключение DocumentExportError.

```
void exportAs(string filePath, ExportFormat format);
```

Расширенные версии метода позволяют задать дополнительные настройки экспорта документа (например, страницы для экспорта и т. д.).

Для текстовых документов (см. TextExportSettings):

```
void exportAs(string filePath, ExportFormat format,
                               TextExportSettings settings);
```

Для табличных документов (см. WorkbookExportSettings):

```
void exportAs(string filePath, ExportFormat format,
                               WorkbookExportSettings settings);
```

Внимание ! Скрытые листы в табличном документе не экспортируются.

Примеры:

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

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

```
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);
```

4.28.1.3 Метод 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());
    }
}
```

4.28.1.4 Метод 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());
}
```

4.28.1.5 Метод getNamedExpressions

Используется для получения списка именованных выражений NamedExpressions.

Пример:

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

4.28.1.6 Метод getScripts

Метод предоставляет доступ к списку макрокоманд Scripts, содержащихся в документе.

Пример:

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
    while (scriptsEnumerator.MoveNext()) {
        Script script = scriptsEnumerator.Current;
        Console.WriteLine(script.getName());
    };
}
```

4.28.1.7 Метод getRange

Метод предоставляет доступ ко всему диапазону Range документа.

Пример:

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

4.28.1.8 Метод getSections

Возвращает объект типа Sections.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
while (sectionsEnumerator.MoveNext()) {
    Section section = sectionsEnumerator.Current;
    Console.WriteLine(section.getType());
};
```

4.28.1.9 Метод getComments

Метод обеспечивает доступ к комментариям документа, возвращает объект Comments.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
while (commentsEnumerator.MoveNext()) {
    Comment comment = commentsEnumerator.Current;
```

```
    Console.WriteLine(comment.getText());
};
```

4.28.1.10 Метод setChangesTrackingEnabled

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

Пример:

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

4.28.1.11 Метод isChangesTrackingEnabled

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

Пример:

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

4.28.1.12 Метод setMirroredMarginsEnabled

Метод позволяет включать и выключать зеркальные поля в документе.

Пример:

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

4.28.1.13 Метод areMirroredMarginsEnabled

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

Пример:

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

4.28.1.14 Метод 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:

```
Document merged = document.merge(otherDocument);
```

4.28.1.15 Метод setPageProperties

Метод устанавливает свойство PageProperties в документе.

Пример:

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

4.28.1.16 Метод setPageOrientation

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

Пример:

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

4.28.1.17 Метод 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) {
    ..........................
}
```

4.28.1.18 Метод getSectionsEnumerator

Возвращает таблицу объектов типа Section.

Пример:

```
SectionsEnumerator sectionsEnumerator = document.getSectionsEnumerator();
while (sectionsEnumerator.MoveNext()) {
    Section section = sectionsEnumerator.Current;
    Console.WriteLine(section.getPageOrientation());
};
```

4.28.2 Класс Table

Класс `Table` предоставляет доступ к листу электронной таблицы или отдельной таблице в составе текстового документа.

```
class Table : Block
```

4.28.2.1 Наименование таблицы

Метод `getName` позволяет получить наименование листа табличного документа.

```
string getName();
```

Метод `setName` позволяет установить значение `newName` в качестве наименования листа табличного документа. Данное значение должно быть уникальным, т.к. может использоваться для ссылки на таблицу, например, из формул.

```
void setName(string newName);
```

4.28.2.2 Размеры таблицы

Методы `getRowsCount` и `getColumnsCount` позволяют получить количество строк и столбцов на листе табличного документа или для отдельной таблицы в составе текстового документа.

```
uint getRowsCount();
uint getColumnsCount();
```

4.28.2.3 Управление столбцами и строками

Методы `insertColumnBefore` и `insertColumnAfter` позволяют добавить в таблицу новые столбцы в количестве `columnsCount`, до или после столбца с индексом `columnIndex`. Индексация столбцов начинается с нуля.

Параметр `copyColumnStyle` указывает, следует ли использовать для новых столбцов элементы оформления столбца `columnIndex`.

```
void insertColumnAfter(uint columnIndex, bool copyColumnStyle, uint
columnsCount);
void insertColumnAfter(uint columnIndex, bool copyColumnStyle);
void insertColumnAfter(uint columnIndex);

void insertColumnBefore(uint columnIndex, bool copyColumnStyle, uint
columnsCount);
void insertColumnBefore(uint columnIndex, bool copyColumnStyle);
void insertColumnBefore(uint columnIndex);
```

Методы `insertRowBefore` и `insertRowAfter` позволяют добавить в таблицу новые строки в количестве `rowsCount` до или после строки с индексом `rowIndex`. Индексация строк начинается с нуля.

Параметр `copyRowStyle` указывает, следует ли использовать для новых строк элементы оформления строки `rowIndex`.

```
void insertRowAfter(uint rowIndex, bool copyRowStyle, uint rowsCount);
void insertRowAfter(uint rowIndex, bool copyRowStyle);
void insertRowAfter(uint rowIndex);

void insertRowBefore(uint rowIndex, bool copyRowStyle, uint rowsCount);
void insertRowBefore(uint rowIndex, bool copyRowStyle);
void insertRowBefore(uint rowIndex);
```

Метод `removeColumn` позволяет удалить столбцы в количестве `columnsCount` начиная с индекса `columnIndex`. Индексация столбцов начинается с нуля.

```
void removeColumn(uint columnIndex, uint columnsCount);
void removeColumn(uint columnIndex);
```

Метод `removeRow` позволяет удалить строки в количестве `rowsCount` начиная с индекса `rowIndex`. Индексация строк начинается с нуля.

```
void removeRow(uint rowIndex, uint rowsCount);
void removeRow(uint rowIndex);
```

Метод `setColumnsVisible` позволяет задавать видимость столбцов в количестве `columnsCount`, начиная с индекса `first`. Индексация столбцов начинается с нуля.

```
void setColumnsVisible(uint first, uint columnsCount, bool visible);
```

Метод `setRowsVisible` позволяет задавать видимость строк в количестве `rowsCount`, начиная с индекса `first`. Индексация строк начинается с нуля.

```
void setRowsVisible(uint first, uint rowsCount, bool visible);
```

Следующий набор методов позволяет группировать строки и колонки таблицы. Редактор дает возможность отображать группы в виде иерархии. Совместно с данными методами можно использовать методы `setColumnsVisible` и `setRowsVisible` чтобы раскрывать и закрывать фрагменты иерархии групп.

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

```
void groupRows(uint first, uint rowsCount);
void ungroupRows(uint first, uint rowsCount);
```

```
void clearRowGroups(uint first, uint rowsCount);
void groupColumns(uint first, uint columnsCount);
void ungroupColumns(uint first, uint columnsCount);
void clearColumnnGroups(uint first, uint columnsCount);
```

Метод `setColumnWidth` позволяет установить значение `width` в качестве ширины столбца с индексом `columnIndex`. Индексация столбцов начинается с нуля. Метод вызывает исключение `OutOfRangeError`, если параметр `columnIndex` выходит за границы таблицы.

```
void setColumnWidth(uint columnIndex, float width);
```

Метод `setRowHeight` позволяет установить значение `height` в качестве высоты строки с индексом `rowIndex`. Индексация строк начинается с нуля. Метод вызывает исключение `OutOfRangeError`, если параметр `rowIndex` выходит за границы таблицы.

Класс `RowHeightRule` задает точность значения, где `Exact` – точно, а `AtLeast` – не меньше.

```
void setRowHeight(uint columnIndex, float width, RowHeightRule heightRule);
```

4.28.2.4 Управление ячейками и диапазонами ячеек

Метод `getCell` позволяет получить доступ к ячейкам таблицы по адресу `cellRef` или положению, указанному в параметре `position`.

Параметр `cellRef` задает адрес ячейки с использованием наиболее распространенной системы адресации ячеек, при которой столбцы задаются буквами, а строки – числами, например, «A1».

Параметр `position` использует для адресации ячейки объект класса `CellPosition`, который позволяет указать строки и столбцы по индексу. Например, положение ячейки «A1» задается как `CellPosition(0,0)`.

```
Cell getCell(string cellRef);
Cell getCell(CellPosition position);
```

Метод `getCellRange` позволяет получить доступ к диапазону ячеек таблицы по адресу в параметре `range` или положению, указанному в параметре `CellRangePosition range`.

Параметр `range` задает адрес диапазона ячеек с использованием наиболее распространенной системы адресации ячеек, при которой столбцы задаются буквами, а строки – числами, например, «A1:C4».

Параметр `CellRangePosition range` использует для адресации диапазона ячеек объект класса `CellRangePosition`, который позволяет указать строки и столбцы по индексу. Например, диапазон «A1:C4» задается как `CellRangePosition(0,0,2,3)`.

```
CellRange getCellRange(string range);
CellRange getCellRange(CellRangePosition range);
```

4.28.2.5 Список диаграмм

Для получения списка диаграмм таблицы используется метод `getCharts`.

4.28.2.6 Создание копии листа в табличном документе

Для создания копии листа в табличном документе используется метод `duplicate`. Созданная копия листа размещается после копируемого листа. Метод может быть использован только в табличном документе.

```
void duplicate();
```

4.28.2.7 Перемещение листа в табличном документе

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.

```
void moveTo(uint index);
```

4.28.2.8 Управление видимостью листа в табличном документе

Для управления видимостью листа таблицы в табличном документе используется метод `setVisible`. Если значение параметра `visible` равно `true`, то лист таблицы отображается в редакторе таблиц.

```
void setVisible(bool visible);
```

Для проверки видимости листа таблицы в табличном документе используется метод `isVisible`.

```
bool isVisible();
```

4.28.2.9 Список именованных выражений

Для получения списка именованных выражений `NamedExpressions` используется метод `getNamedExpressions`.

```
NamedExpressions getNamedExpressions();
```

4.28.2.10 Получение области печати в табличном документе

Метод `getPrintAreas` возвращает коллекцию текущих областей печати `CellRangePosition`.

```
CellRangePosition getPrintAreas(CellRangePosition range);
```

4.28.2.11 Задание области печати в табличном документе

Для задания области печати `CellRangePosition` в табличном документе для экспорта документа, печати и т. д. используется метод `setPrintArea`. Для задания нескольких областей печати используется метод `setPrintAreas`. Если значение параметра равно `null`, то область печати считается неопределенной.

```
void setPrintArea(CellRangePosition range);
void setPrintAreas(CellRangePositions range);
```

4.28.3 Именованные выражения

Именованное выражение – это выражение (являющееся описанием диапазона или формулой), которому присвоено имя. Преимуществом именованного выражения является его информативность. Именованные выражения упрощают работу с ячейками, также их удобно использовать при работе с формулами. На данный момент доступна возможность работы с именованными выражениями, представляющими собой ссылки на диапазоны ячеек. Доступ к именованным выражениям осуществляется посредством методов Document::getNamedExpressions() и Table::getNamedExpressions().

4.28.3.1 Класс NamedExpression

Класс описывает структуру именованного выражения.

Пример:

```
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpressionsEnumerator enumerator = namedExpressions.getEnumerator();
while (enumerator.MoveNext()) {
    NamedExpression namedExpression = enumerator.Current;
    Console.WriteLine(namedExpression.getName());
    Console.WriteLine(namedExpression.getExpression());
    CellRange cellRange = namedExpression.getCellRange();
    Console.WriteLine(cellRange.getBeginColumn());
    Console.WriteLine(cellRange.getLastColumn());
}
```

4.28.3.1.1 Метод getName

Возвращает имя именованного выражения.

4.28.3.1.2 Метод getExpression

Возвращает текст выражения (формулы).

4.28.3.1.3 Метод getCellRange

Возвращает диапазон ячеек `CellRange`, если выражение является ссылкой на диапазон.

4.28.3.2 Класс NamedExpressions

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

4.28.3.2.1 Метод get

Возвращает именованное выражение `NamedExpression` по имени, 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());
}
```

4.28.3.2.2 Метод getEnumerator

Позволяет получить доступ ко всему списку именованных выражений.

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpressionsEnumerator enumerator = namedExpressions.getEnumerator();
while (enumerator.MoveNext()) {
    NamedExpression namedExpression = enumerator.Current;
    Console.WriteLine(namedExpression.getName());
}
```

4.28.3.2.3 Метод addExpression

Добавляет новое выражение в список именованных выражений, возвращает результат операции NamedExpressionsValidationResult.

Пример:

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

4.28.3.2.4 Метод removeExpression

Удаляет выражение по заданному имени, возвращает результат операции NamedExpressionsValidationResult.

Пример:

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

4.28.3.3 Тип NamedExpressionsValidationResult

Класс `NamedExpressionsValidationResult` описывает результат операций NamedExpressions::addExpression(), NamedExpressions::removeExpression().

Тип описывает следующие варианты результатов операций:

- `Success` – операция выполнена успешно;
- `WrongName` – неправильный формат имени;
- `IsUsedInFormula` – имя уже используется в формуле.

4.28.4 Сводные таблицы

Сводная таблица - инструмент обработки данных, служащий для их обобщения и удобства обработки.

4.28.4.1 Класс PivotTable

Класс для представления сводной таблицы. Может быть получен из ячейки

Cell::getPivotTable(), либо при создании новой сводной таблицы PivotTablesManager::create().

4.28.4.1.1 Метод 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();
}
```

4.28.4.1.2 Метод 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) {
    Console.WriteLine(pivotTable.getSourceRangeAddress());
}
```

4.28.4.1.3 Метод 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());
}
```

4.28.4.1.4 Метод getPivotRange

Метод возвращает диапазон ячеек, в котором размещена сводная таблица.

Пример:

```
PivotTable pivotTable = pivotTablesManager.create(cellRange);
CellRange pivotRange = pivotTable.getPivotRange();
Console.WriteLine(pivotRange.getBeginColumn() + ", " +
pivotRange.getLastColumn());
```

4.28.4.1.5 Метод changeSourceRange

Метод позволяет задать новый диапазон исходных данных сводной таблицы без обновления самой таблицы.

Пример:

```
pivotTable.changeSourceRange("I3:K5");
CellRange sourceRange = pivotTable.getSourceRange();
Console.WriteLine(sourceRange.getBeginColumn() + ", " +
sourceRange.getLastColumn());
```

4.28.4.1.6 Метод isRowGrandTotalEnabled

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

Пример:

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

4.28.4.1.7 Метод isColumnGrandTotalEnabled

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

Пример:

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

4.28.4.1.8 Метод getPivotTableCaptions

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

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);
```

4.28.4.1.9 Метод 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);
```

4.28.4.1.10 Метод getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства PivotTableUnsupportedFeatures сводной таблицы.

Пример:

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

4.28.4.1.11 Метод getFieldsList

Метод возвращает список PivotTableField 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);
}
```

4.28.4.1.12 Метод getRowFields

Метод возвращает список полей 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);
}
```

4.28.4.1.13 Метод getColumnFields

Метод возвращает список полей 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);
}
```

4.28.4.1.14 Метод getValueFields

Метод возвращает список полей 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);
}
```

4.28.4.1.15 Метод getPageFields

Метод возвращает список полей из области фильтров.

Пример:

```
PivotTableCategoryFields pageFields = pivotTable.getPageFields();
PivotTableCategoryFields.PivotTableCategoryFieldsEnumerator enumerator =
pageFields.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableCategoryField pivotTableCategory = enumerator.Current;
    Console.WriteLine(pivotTableCategory.fieldProperties.fieldAlias);
    Console.WriteLine(pivotTableCategory.fieldProperties.subtotalAlias);
    Console.WriteLine(pivotTableCategory.fieldProperties.fieldName);
}
```

4.28.4.1.16 Метод getFieldCategories

Метод возвращает список категорий PivotTableFieldCategories, содержащих заданное поле.

Пример:

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

4.28.4.1.17 Метод getFieldItems

Метод возвращает все элементы сводной таблицы по заданному имени поля.

Пример:

```
PivotTableItems pivotTableItems = pivotTable.getFieldItems("Age");
PivotTableItemsEnumerator enumerator = pivotTableItems.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableItem pivotTableItem = enumerator.Current;
    Console.WriteLine(pivotTableItem.getAlias());
}
```

4.28.4.1.18 Метод getFieldItemsByName

Метод возвращает все элементы из заданного поля по имени.

Пример:

```
PivotTableItems itemsByName = pivotTable.getFieldItemsByName("Ultimate Question
of Life", "42");
PivotTableItemsEnumerator enumerator = itemsByName.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableItem pivotTableItems = enumerator.Current;
    Console.WriteLine(pivotTableItems.getName());
}
```

4.28.4.1.19 Метод getFilter

Метод возвращает фильтр PivotTableFilter по заданному имени поля.

Пример:

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

4.28.4.1.20 Метод getFilters

Метод возвращает список фильтров PivotTableFilters сводной таблицы.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter pivotTableFilter = enumerator.Current;
    Console.WriteLine(pivotTableFilter.setHidden());
}
```

4.28.4.1.21 Метод update

Метод обновляет и полностью пересчитывает сводную таблицу. Возвращает PivotTableUpdateResult.

Пример:

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

4.28.4.1.22 Метод 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();
```

4.28.4.2 Класс PivotTableFields

Класс `PivotTableFields` предназначен для хранения коллекции объектов типа `PivotTableField`.

Поля класса:

- `Count` – текущий размер списка полей;
- `Capacity` – максимально доступный размер списка полей.

4.28.4.2.1 Метод для получения элемента списка по индексу

```
PivotTableField this[int index];
```

4.28.4.3 Класс PivotTableCaptions

Класс `PivotTableCaptions` хранит все пользовательские заголовки сводной таблицы.

Класс содержит следующие поля:

- `errorCaption` – алиас (псевдоним) для значений, которые возвращают ошибку;
- `emptyCaption` – алиас для значений, которые возвращают пустое значение;
- `grandTotalCaption` - алиас общих итогов;
- `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 имеет тип ‘outline’ или ‘tabular’;
- `rowHeaderCaption` – алиас заголовка строк (виден только при включенном компактном макете, это алиас по умолчанию);
- `columnHeaderCaption` – алиас заголовка колонок (виден только при включенном компактном макете, это алиас по умолчанию).

4.28.4.4 Класс PivotTableLayoutSettings

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

Класс содержит следующие поля:

- `reportLayout` – настройка вида макета сводной таблицы `PivotTableReportLayout` (компактный, табличный, структурный);

- `valueFieldsOrientation` – `ValueFieldsOrientation`, позволяет расположить положение значений в случае, если в сводной таблице более двух полей значений;
- `pageFieldOrder` – настройка порядка полей фильтров `PageFieldOrder` (вниз, затем поперек или сначала поперек, потом вниз);
- `indentForCompactLayout` – размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей);
- `pageFieldWrapCount` – настройка связана с полем `pageFieldOrder`, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д);
- `isMergeAndCenterLabelsEnabled` – настройка позволяет объединить ячейки заголовков;
- `useGridDropZones` – использовать классический вид (как в Excel 2003). Влияет только на расположение полей в отчете;
- `displayFieldCaptions` – отображать заголовки полей.

4.28.4.5 Тип PivotTableUnsupportedFeature

Класс `PivotTableUnsupportedFeature` описывает неподдерживаемую функциональность сводных таблиц. Получение неподдерживаемой функциональности сводных таблиц описано в `PivotTable::getUnsupportedFeatures()`.

Варианты неподдерживаемых свойств:

- `CalculatedField` – вычисляемые поля;
- `CalculatedItem` – вычисляемые элементы;
- `CollapsedValues` – свернутые поля;
- `ShowDataAs` – дополнительные вычисления (Show data как в MS Excel).

4.28.4.6 Тип PivotTableReportLayout

Таблица `PivotTableReportLayout` описывает внешний вид отчетов сводной таблицы. Является полем класса `PivotTableLayoutSettings`.

Варианты внешнего вида отчетов:

- `Compact` – компактный;
- `Tabular` – табличный;
- `Outline` – структурный.

4.28.4.7 Тип ValueFieldsOrientation

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

Варианты ориентации:

- `ByRows` – по строкам;
- `ByColumns` – по столбцам.

4.28.4.8 Тип PageFieldOrder

Класс `PageFieldOrder` описывает вид отображения полей из области фильтров. Является полем класса `PivotTableLayoutSettings`.

Варианты ориентации:

- `DownThenOver` – вниз, затем поперек;
- `OverThenDown` – поперек, затем вниз.

4.28.4.9 Тип PivotTableFieldCategory

Класс `PivotTableFieldCategory` описывает флаги, которые задают категорию области полей.

Доступны следующие варианты категорий:

- `Pages` – область фильтров;
- `Rows` – область строк;
- `Columns` – область колонок;
- `Values` – область значений.

4.28.4.10 Класс PivotTableFieldCategories

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

4.28.4.10.1 Метод getEnumerator

Возвращает объект `PivotTableFieldCategorysEnumerator` для доступа к коллекции категорий.

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();
    while (enumerator.MoveNext()) {
        PivotTableFieldCategory category = (PivotTableFieldCategory)
enumerator.Current;
        if (category != null) {
            Console.WriteLine(category);
        }
    }
}
```

4.28.4.11 Тип PivotTableFunction

Тип `PivotTableFunction` описывает функции, которые могут быть использованы в сводных таблицах. Используется в качестве поля `subtotalFunctions` класса `PivotTableCategoryField`.

Доступны следующие типы функций сводных таблиц:

- `Auto` – автозаполнение;
- `Sum` – суммирует все числовые данные;
- `Count` – количество всех ячеек;
- `CountNums` – количество числовых ячеек;
- `Average` – среднее значение;
- `Max` – наибольшее значение;
- `Min` – наименьшее значение;
- `Product` – произведение всех ячеек;
- `StdDeviation` – стандартное смещенное отклонение;
- `StdDeviationPopulation` – стандартное несмещенное отклонение;
- `Variance` – смещенная дисперсия;
- `VariancePopulation` – несмещенная дисперсия.

4.28.4.12 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();
    while (enumerator.MoveNext()) {
        PivotTableFilter pivotTableFilter = enumerator.Current;
        for (uint i = 0; i < pivotTableFilter.getCount(); i++) {
            pivotTableFilter.setHidden(i, false);
        }
    }
    PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
    pivotTableEditor.setFilters(pivotTableFilters);
    pivotTableEditor.apply();
}
```

4.28.4.12.1 Метод getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter pivotTableFilter = enumerator.Current;
    Console.WriteLine(pivotTableFilter.getFieldName());
}
```

4.28.4.12.2 Метод getCount

Возвращает количество фильтруемых полей.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter pivotTableFilter = enumerator.Current;
    Console.WriteLine(pivotTableFilter.getCount());
}
```

4.28.4.12.3 Метод getName

Возвращает имя поля для заданного индекса.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter pivotTableFilter = enumerator.Current;
    for (uint i = 0; i < pivotTableFilter.getCount(); i++) {
        Console.WriteLine(pivotTableFilter.getName());
    }
}
```

4.28.4.12.4 Метод isHidden

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

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter pivotTableFilter = enumerator.Current;
    for (uint i = 0; i < pivotTableFilter.getCount(); i++) {
        Console.WriteLine(pivotTableFilter.isHidden(i));
    }
}
```

4.28.4.12.5 Метод setHidden

Устанавливает видимость поля для заданного индекса. Параметры: `itemName` – индекс поля, `hidden` – видимость (`true` – поле скрыто).

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter pivotTableFilter = enumerator.Current;
    for (uint i = 0; i < pivotTableFilter.getCount(); i++) {
        Console.WriteLine(pivotTableFilter.setHidden(i, false));
    }
}
```

4.28.4.13 Класс PivotTableFilters

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

4.28.4.13.1 Метод 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();
    while (enumerator.MoveNext()) {
        PivotTableFilter filter = (PivotTableFilter)enumerator.Current;
        if (filter != null) {
            Console.WriteLine(filter.getFieldName());
        }
    }
}
```

4.28.4.14 Класс PivotTableFieldProperties

Класс содержит свойства поля `PivotTableField` сводной таблицы

Поля класса:

- `fieldName` – имя поля;
- `fieldAlias` – псевдоним поля (пользовательское имя поля);
- `subtotalAlias` – псевдоним подытогов конкретного поля.

4.28.4.15 Класс PivotTableField

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

Поля класса:

- `fieldProperties` – свойства полей сводной таблицы;
- `fieldCategories` – категории полей сводной таблицы;
- `customFormula` – вычисляемая формула.

4.28.4.16 Класс PivotTableCategoryField

`PivotTableCategoryField` содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. Таблицу 1). Объект может быть получен посредством вызовов `PivotTable::getRowFields()`, `PivotTable::getColumnFields()`.

Поля класса:

- `fieldProperties` – свойства поля;
- `subtotalFunctions` – список функций для вычисления подытога.

4.28.4.17 Класс PivotTableValueField

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

Поля класса:

- `baseFieldName` – оригинальное поле на основе которого было создано данное поле;
- `valueFieldName` – автоматический уникальный псевдоним такой как “Sum of %имя поля%“;

- `cellNumberFormat` – числовой формат для конкретного поля значений;
- `totalFunction` – агрегирующая функция поля значений (SUM, COUNT, MAX и т.д.);
- `customFormula` – вычисляемая формула для поля значений.

4.28.4.18 Класс PivotTablePageField

Содержит свойства поля из области фильтров (см. Таблицу 1). Объект может быть получен посредством вызова `PivotTable::getPageFields()`.

Поле класса:

- `fieldProperties` – свойства поля.

4.28.4.19 Класс PivotTableItem

Класс описывает элемент сводной таблицы.

```
class PivotTableItem
```

4.28.4.19.1 Метод getName

Метод возвращает имя элемента сводной таблицы.

```
string getName();
```

4.28.4.19.2 Метод getAlias

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем).

```
string getAlias();
```

4.28.4.19.3 Метод getItemType

Метод возвращает тип элемента сводной таблицы.

```
PivotTableItemType getItemType();
```

4.28.4.19.4 Метод isCollapsed

Метод возавращает `true`, если элемент сводной таблицы свернут.

```
bool isCollapsed();
```

4.28.4.20 Тип PivotTableItemType

Описывает возможные варианты элементов сводной таблицы.

Доступные типы полей:

- `Number` – числовой;
- `String` – строковый;

- `Boolean` – логический;
- `DateTime` – дата / время;
- `Empty` – пустой тип;
- `Error` – ошибка;
- `NumberGroup` – интервальная группировка;
- `DateIntervalGroup` – интервальная группировка по датам;
- `DateTimeGroup` – группировка по дате / времени;
- `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 enumerator = pivotTableItems.GetEnumerator();
    while (enumerator.MoveNext()) {
        PivotTableItem pivotTableItem = enumerator.Current;
        if (pivotTableItem != null) {
            Console.WriteLine(pivotTableItem.getItemType());
        }
    }
}
```

4.28.4.21 Класс PivotTableEditor

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

4.28.4.21.1 Метод addField

Метод добавляет новое поле в сводную таблицу, используя параметры: `fieldName` - имя поля, `toCategory` - категория поля, `index` - позиция в категории.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.addField("CC",
```

```
PivotTableFieldCategory.Values);
pivotTableEditor.apply();
```

4.28.4.21.2 Метод moveField

Метод перемещает поле между категориями. Параметры: `fieldName` - имя поля, `toCategory` - область, в которую перемещается поле, `index` - позиция в новой категориии.

Пример:

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

4.28.4.21.3 Метод removeField

Метод удаляет поле из категории. Параметры: `fieldName` - имя поля, `fromCategory` - область, из которой удаляется поле.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.removeField("BB",
PivotTableFieldCategory.Values);
pivotTableEditor.apply();
```

4.28.4.21.4 Метод reorderField

Метод изменяет позицию поля в пределах категории. Параметры: `fieldName` - имя поля, `category` - область, `toIndex` - новая позиция поля.

Пример:

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

4.28.4.21.5 Метод 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();
```

4.28.4.21.6 Метод 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();
```

4.28.4.21.7 Метод setSummmarizeFunction

Метод задает суммирующую функцию для поля из области значений. Параметр `valueFieldName` - имя поля, `summarizeFunction` – суммирующая функция `PivotTableFunction`.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFunction summarizeFunction = PivotTableFunction.Average;
pivotTableEditor = pivotTableEditor.setSummarizeFunction("CC",
summarizeFunction);
```

4.28.4.21.8 Метод setFilter

Метод задает фильтр `PivotTableFilter` сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter filter = enumerator.Current;
    uint filterSize = filter.getCount();
    for (uint i = 0; i < filterSize; i++) {
        filter.setHidden(i, true);
    }
```

```
    pivotTableEditor.setFilter(filter);
}
PivotTableUpdateResult pivotTableUpdateResult = pivotTableEditor.apply();
```

4.28.4.21.9 Метод setFilters

Метод задает фильтры сводной таблицы. Если какой-то из фильтров не может быть применен, он пропускается.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableFilter filter = enumerator.Current;
    uint filterSize = filter.getCount();
    for (uint i = 0; i < filterSize; i++) {
        filter.setHidden(i, true);
    }
    pivotTableEditor.setFilter(filter);
}
pivotTableEditor.setFilters(pivotTableFilters);
PivotTableUpdateResult pivotTableUpdateResult = pivotTableEditor.apply();
```

4.28.4.21.10 Метод setCaptions

Метод задает заголовки сводной таблицы.

Пример:

```
PivotTableCaptions captions = pivotTable.getPivotTableCaptions();
captions.grandTotalCaption = "Общий итог за год";

PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor.setCaptions(captions);
pivotTableEditor.apply();
```

4.28.4.21.11 Метод setLayoutSettings

Метод устанавливает настройки отображения PivotTableLayoutSettings сводной таблицы.

Пример:

```
PivotTableLayoutSettings pivotTableLayoutSettings =
pivotTable.getPivotTableLayoutSettings();
pivotTableLayoutSettings.reportLayout = PivotTableReportLayout.Tabular;

PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.setLayoutSettings(pivotTableLayoutSettings);
pivotTableEditor.apply();
```

4.28.4.21.12 Метод 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();
```

4.28.4.21.13 Метод apply

Метод обновляет сводную таблицу с заданными свойствами и возвращает `PivotTableUpdateResult`.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
if (PivotTableUpdateResult.Success == pivotTableEditor.apply()) {
    Console.WriteLine("Successfully applied");
}
```

4.28.4.22 Тип PivotTableUpdateResult

Тип описывает возможные результаты обновления сводной таблицы (см. методы `PivotTable::update()`, `PivotTableEditor:apply()`).

Значения полей:

- `Success` – успешное обновление таблицы;
- `NoPivotTable` – сводная таблица не найдена;
- `NoSuchFieldInCategory` – не найдено поле в категории;
- `NoSuchFieldInPivotTable` – не найдено поле в сводной таблице;

- `InvalidIndex` – ошибка в индексе;
- `FieldAlreadyEnabled` – поле уже разрешено;
- `MovingFieldToTheSameCategoryForbidden` – попытка перемещения поля в рамках текущей категории;
- `InvalidFunction` – неправильная функция;
- `InvalidDataSourceRange` – ошибка диапазона исходных данных;
- `NoDataRowsInDataSource` – в исходных данных нет строк с данными;
- `EmptyDataSourceHeaders` – пустые заголовки исходных данных;
- `NoReferenceUnderDefine` – попытка обновить/создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу;
- `AnotherPivotInsideDataSource` – найдена другая сводная таблица в этом же диапазоне.

4.28.4.23 Класс PivotTablesManager

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

Пример:

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

4.28.4.23.1 Метод create

Метод создает сводную таблицу на основе диапазона исходных данных. Если местоположение (`destination`) не задано, создается новый лист (таблица), и сводная таблица будет расположена по умолчанию.

```
PivotTable create(CellRange cellRange, Cell destination);
```

Метод создает сводную таблицу на основе строки формулы. Если местоположение (`destination`) не задано, создается новая таблица и сводная таблица будет расположена по умолчанию.

```
PivotTable create(string formula, Cell destination);
```

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
CellRange cellRange = sheet.getCellRange("B3:C4");
```

```
PivotTablesManager pivotTablesManager = document.getPivotTablesManager();
PivotTable pivotTable = pivotTablesManager.create(cellRange);
```

4.28.5 Диаграммы

Работа с диаграммами реализована только в табличных документах.

Доступ к списку диаграмм производится через класс `Table`, соответствующий листу табличного документа.

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Console.WriteLine(charts.getChartsCount());
}
```

4.28.5.1 Класс Chart

Класс `Chart` представляет диаграмму в документе со всеми элементами (заголовок, легенда, тип, данные, диапазон и т.д.).

4.28.5.1.1 Метод 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());
    }
}
```

4.28.5.1.2 Метод setType

Метод устанавливает тип диаграммы `ChartType`.

Пример:

```
Charts charts = sheetDocumentPage.getCharts();
Chart chart = charts.getChart(0);
if (chart != null) {
```

```
    chart.setType(ChartType.ColumnStacked);
    Console.WriteLine(chart.getType());
}
```

4.28.5.1.3 Метод getRangesCount

Метод возвращает количество серий диаграммы.

Пример:

```
Charts charts = sheetDocumentPage.getCharts();
Chart chart = charts.getChart(0);
if (chart != null) {
    Console.WriteLine(chart.getRangesCount());
}
```

4.28.5.1.4 Метод getRange

Метод возвращает диапазон ячеек ChartRangeInfo с исходными данными диаграммы.

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);
    }
}
```

4.28.5.1.5 Метод getTitle

Метод возвращает заголовок диаграммы ChartType.

Пример:

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

4.28.5.1.6 Метод setRange

Метод задает диапазон ячеек `CellRangePosition` 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());
```

4.28.5.1.7 Метод 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));
```

4.28.5.1.8 Метод isEmpty

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

Пример:

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

4.28.5.1.9 Метод 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());
```

4.28.5.1.10 Метод is3D

Метод возвращает `true`, если диаграмма трехмерная.

Пример:

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

4.28.5.1.11 Метод getDirectionType

Метод возвращает направление серий диаграммы `ChartSeriesDirectionType`.

Пример:

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

4.28.5.1.12 Метод getChartLabels

Метод возвращает коллекцию меток диаграммы типа `ChartLabelsInfo`.

Пример:

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

4.28.5.1.13 Метод getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

Пример:

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

4.28.5.1.14 Метод 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);
```

4.28.5.2 Класс Charts

Класс `Charts` обеспечивает доступ к списку диаграмм табличного документа. Доступ к списку диаграмм осуществляется с помощью метода `Table::getCharts()`.

4.28.5.2.1 Метод getChartsCount

Метод возвращает общее количество диаграмм.

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Console.WriteLine(charts.getChartsCount());
}
```

4.28.5.2.2 Метод getChart

Метод возвращает диаграмму `Chart` по индексу 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());
    }
}
```

4.28.5.2.3 Метод getChartIndexByDrawingIndex

Метод возвращает диаграмму `Chart` по индексу отрисовки.

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Console.WriteLine(charts.getChartIndexByDrawingIndex(0));
}
```

4.28.5.3 Тип ChartLabelsDetectionMode

Тип `ChartLabelsDetectionMode` описывает режимы автоматического определения меток диаграмм.

Доступны следующие типы автоматического определения меток диаграмм:

- `Unknown` – неопределенный тип;
- `FirstRow` – метка на первой строке;
- `FirstColumn` – метка на первой колонке;
- `NoLabels` – не отрисовывать метки.

Пример:

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

4.28.5.4 Класс ChartLabelsInfo

Класс `ChartLabelsInfo` описывает настройки автоматического определения меток диаграммы. Инициализируется конструктором, в который передаются параметры: `categoriesMode` - режим автоматического определения меток для категорий, `seriesNameMode` - режим автоматического определения меток для серий, `oneColumnRow` - передается `true`, если диапазон диаграммы содержит только одну строку или одну колонку.

Поля класса:

- `categoriesMode` – режим автоматического определения меток для категорий;
- `seriesNameMode` – режим автоматического определения меток для серий;

- `isOneColumnRowChart` – поле содержит `true`, если диапазон диаграммы содержит только одну строку или одну колонку.

Примеры:

```
ChartLabelsInfo chartInfo = new
ChartLabelsInfo(ChartLabelsDetectionMode.FirstRow,
ChartLabelsDetectionMode.NoLabels, false);
```

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

4.28.5.5 Класс ChartRangeInfo

Класс `ChartRangeInfo` описывает серию диаграммы. Инициализируется конструктором, в который передаются следующие параметры: `tableRangeInfo` - диапазон ячеек, `color` – цвет серии диаграммы, `hidden` – видимость серии, `rangeType` – тип диапазона исходных данных диаграммы.

Поля класса:

- `tableRangeInfo` – исходный диапазон ячеек для серии;
- `rangeColor` – цвет для отрисовки серии;
- `isHidden` – задает видимость серии диаграммы;
- `rangeType` – тип диапазона диаграммы.

Пример:

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

4.28.5.6 Тип ChartRangeType

Тип `ChartRangeType` описывает варианты диапазона исходных данных диаграммы.

Возможные значения:

- `Series` – серии;
- `SeriesName` – имена серий;
- `Categories` – категории;

- `DataPoint` – разметка данных.

Пример:

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

4.28.5.7 Тип ChartSeriesDirectionType

Тип `ChartSeriesDirectionType` описывает направление серий диаграмм.

Возможные значения:

- `Unknown` – неопределенный тип;
- `ByRow` – серии направлены по строкам;
- `ByColumn` – серии направлены по колонкам.

Пример:

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

4.28.5.8 Тип ChartType

Тип `ChartType` описывает все поддерживаемые типы диаграмм.

Варианты типов диаграмм:

- `Unknown` – неопределенный тип;
- `Bar` – линейчатая диаграмма с группировкой;
- `BarStacked` – линейчатая диаграмма с накоплением;
- `BarPercentStacked` – линейчатая нормированная диаграмма с накоплением;
- `Column` – гистограмма с группировкой;
- `ColumnStacked` – гистограмма с накоплением;
- `ColumnPercentStacked` – нормированная гистограмма с накоплением;
- `Line` – стандартный график;
- `LineStacked` – график с накоплением;
- `LinePercentStacked` – нормированный график с накоплением;
- `LineWithMarker` – стандартный график с маркерами;
- `LineWithMarkerStacked` – график с накоплением и маркерами;

- `LineWithMarkerPercentStacked` – нормированный график с накоплением и маркерами;
- `Area` – стандартная диаграмма с областями;
- `AreaStacked` – диаграмма с областями с накоплением;
- `AreaPercentStacked` – нормированная диаграмма с областями с накоплением;
- `Pie` – круговая диаграмма;
- `PieExploded` – круговая диаграмма с отделенными секторами;
- `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);
}
```

4.28.6 Класс TableRangeInfo

Предоставляет положение диапазона ячеек в таблице.

Свойство класса:

- `tableRange` – диапазон, содержащий ячейки, тип `CellRangePosition`.

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);
```

4.28.6.1 Метод TableRangeInfo

Конструктор по умолчанию.

```
TableRangeInfo(CellRangePosition range);
```

4.28.7 Класс Paragraphs

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

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();
```

4.28.7.1 Метод setListSchema

Метод `setListSchema` позволяет установить тип маркированного или нумерованного списка (см. описание класса `ListSchema`).

Пример:

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.setListSchema(NCT.MyOfficeSDK.ListSchema.BulletCheckmark);
```

4.28.7.2 Методы setListLevel, increaseListLevel, decreaseListLevel

Методы позволяют управлять глубиной вложенности элемента списка. Значение может быть равным `null`, если схема нумерации не установлена для абзаца. В этом случае будет установлено минимальное значение.

Примеры:

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.setListLevel(1);
```

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.increaseListLevel();
```

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
paragraphs.decreaseListLevel();
```

4.28.7.3 Метод GetEnumerator

Возвращают объект типа `ParagraphsEnumerator` для доступа к коллекции абзацев.

Пример:

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
while (paragraphs.GetEnumerator().MoveNext()) {
    Paragraph paragraph = paragraphsEnumerator.Current;
    Console.WriteLine(paragraph.getRange().extractText());
};
```

4.28.7.4 Класс Paragraph

Класс `Paragraph` в объектной модели документа представляет отдельный абзац текста и является наследником класса `Block`.

4.28.7.4.1 Метод 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();
```

```
while (paragraphsEnumerator.MoveNext()) {
    Paragraph paragraph = paragraphsEnumerator.Current;
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
};
```

4.28.7.4.2 Методы 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();
while (paragraphsEnumerator.MoveNext()) {
    Paragraph paragraph = paragraphsEnumerator.Current;
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    paragraphProperties.alignment = Alignment.Left;
    paragraph.setParagraphProperties(paragraphProperties);
};
```

4.28.7.4.3 Метод 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());
}
```

4.28.7.4.4 Метод 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());
}
```

4.28.7.4.5 Метод 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());
}
```

4.28.7.4.6 Метод 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());
}
```

4.28.7.4.7 Метод 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());
}
```

4.28.7.4.8 Метод 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());
}
```

4.28.8 Класс ParagraphProperties

Класс `ParagraphProperties` представляет свойства абзаца.

Список свойств:

- `beforeSpacing` – интервал перед абзацем;

- `afterSpacing` – интервал после абзаца;
- `lineSpacing` – межстрочный интервал;
- `alignment` – выравнивание абзаца;
- `firstLineIndent` – отступ первой строки;
- `leftIndent` – отступ слева;
- `rightIndent` – отступ справа.

Пример для текстового документа:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.setListSchema(ListSchema.BulletCheckmark);
    ParagraphProperties paraProps = paragraph.getParagraphProperties();
    paraProps.afterSpacing = 28.3f;    // соответствует 1 см
    paraProps.beforeSpacing = 28.3f;   // соответствует 1 см
    paraProps.firstLineIndent = 28.3f; // соответствует 1 см
    paraProps.leftIndent = 28.3f;      // соответствует 1см
    paraProps.rightIndent = 28.3f;     // соответствует 1см
    paraProps.alignment = NCT.MyOfficeSDK.Alignment.Center;
    paraProps.lineSpacing = new LineSpacing(5.0f, LineSpacingRule.Multiple);
}
```

Пример для табличного документа:

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
NCT.MyOfficeSDK.Range range = cell.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
while (paragraphsEnumerator.MoveNext()) {
    Paragraph paragraph = paragraphsEnumerator.Current;
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
};
```

4.28.9 Класс Shape

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

4.28.9.1 Метод getShapeProperties

Метод возвращает свойства фигуры `ShapeProperties`.

Пример:

```
Shape shape = document.getBlocks().getShape(0);
if (shape != null) {
    ShapeProperties shapeProperties = shape.getShapeProperties();
    Console.WriteLine(shapeProperties.verticalAlignment);
}
```

4.28.9.2 Метод 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);
}
```

4.28.10 Класс ShapeProperties

Класс описывает свойства фигуры и содержит следующие поля: `verticalAlignment` – вертикальное выравнивание, `borderProperties` – свойства границ фигуры, `fill` – свойства заполнения фигуры, `shapeTextLayout` – свойства текста внутри фигуры.

Поля класса:

- `verticalAlignment` – вертикальное выравнивание, тип `VerticalAlignment`;
- `borderProperties` – свойства границ фигуры, тип `LineProperties`;
- `fill` – свойства заполнения фигуры, тип `Fill`;
- `shapeTextLayout` – свойства текста внутри фигуры, тип `ShapeTextLayout`.

4.28.11 Тип ShapeTextLayout

Тип описывает свойства текста, находящегося в фигуре.

Представлены следующие значения:

- `DoNotAutoFit` – размещение текста внутри фигуры по умолчанию;

- `FitShapeExtensionToText` – расширение фигуры под размер текста;
- `FitTextToShape` – заполнение фигуры текстом.

4.28.12 Класс Fill

Класс описывает свойства заполнения фигуры: цвет заполнения, путь к изображению фона.

4.28.12.1 Метод getColor

Метод возвращает цвет заполнения `Color`.

4.28.12.2 Метод getUrl

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

4.28.12.3 Метод isNoFill

Метод возвращает `true`, если заполнения нет.

4.28.13 Класс Field

Класс `Field` наследуется от класса `Block` и предназначен для реализации некоторых полей, например, таких как содержание.

4.28.14 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());
    };
};
```

4.28.14.1 Метод 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());
}
```

4.28.14.2 Метод 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);
}
```

4.28.14.3 Метод 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");
    }
}
```

4.28.14.4 Метод 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());
    };
};
```

4.28.15 Класс Script

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

4.28.15.1 Метод getName

Метод возвращает имя макрокоманды.

Пример:

```
while (enumerator.MoveNext()) {
    Script script = enumerator.Current;
    Console.WriteLine(script.getName());
};
```

4.28.15.2 Метод setName

Метод устанавливает имя для макрокоманды.

Пример:

```
while (enumerator.MoveNext()) {
    Script script = enumerator.Current;
    script.setName("Script name")
    Console.WriteLine(script.getName());
};
```

4.28.15.3 Метод getBody

Метод возвращает текст макрокоманды в виде строки.

Пример:

```
while (enumerator.MoveNext()) {
    Script script = enumerator.Current;
    Console.WriteLine(script.getBody());
};();
```

4.28.15.4 Метод setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

Пример:

```
while (enumerator.MoveNext()) {
    Script script = enumerator.Current;
    script.setName("local scripts = document:getScripts()\nfor script in scripts
: enumerate() do\nprint(script:getName())\nend")
    Console.WriteLine(script.getName());
};
```

4.28.16 Класс Blocks

Класс `Blocks` обеспечивает доступ к блокам `Block` документа или диапазона документа. Объект класса `Blocks` может быть получен вызовом метода `Document::getBlocks` или `HeaderFooter::getBlocks`.

4.28.16.1 Метод 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");
}
```

4.28.16.2 Метод 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");
}
```

4.28.16.3 Метод getTable

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

Пример:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Console.WriteLine(table.getRange().extractText());
} else {
    Console.WriteLine("No tables found");
}
```

В качестве параметра метода также можно указать имя таблицы.

Пример:

```
Table table = document.getBlocks().getTable("Sheet1");
```

4.28.16.4 Метод 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");
}
```

4.28.16.5 Метод 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");
}
```

4.28.16.6 Метод GetEnumerator

Позволяет реализовать перечисление объектов `Block`.

Пример:

```
BlocksEnumerator blocksEnumerator = document.getBlocks().getBlocksEnumerator();
while (blocksEnumerator.MoveNext()) {
    Block block = blocksEnumerator.Current;
    if (block != null) {
        Console.WriteLine(block.getRange().extractText());
    }
}
```

4.28.16.7 Метод getParagraphsEnumerator

Позволяет реализовать перечисление абзацев `Paragraph`.

Пример:

```
ParagraphsEnumerator paragraphsEnumerator =
document.getBlocks().getParagraphsEnumerator();
while (paragraphsEnumerator.MoveNext()) {
    Paragraph paragraph = paragraphsEnumerator.Current;
    if (paragraph != null) {
        Console.WriteLine(paragraph.getRange().extractText());
    }
}
```

4.28.16.8 Метод getTablesEnumerator

Позволяет перечислить объекты типа `Table`.

Пример:

```
TablesEnumerator tablesEnumerator = document.getBlocks().getTablesEnumerator();
while (tablesEnumerator.MoveNext()) {
    Table table = tablesEnumerator.Current;
    if (table != null) {
        Console.WriteLine(table.getRange().extractText());
    }
}
```

4.28.16.9 Метод getShapesEnumerator

Позволяет перечислить объекты типа `Shape`.

Пример:

```
ShapesEnumerator shapesEnumerator = document.getBlocks().getShapesEnumerator();
while (shapesEnumerator.MoveNext()) {
    Shape shape = shapesEnumerator.Current;
    if (shape != null) {
        Console.WriteLine(shape.getRange().extractText());
    }
}
```

4.28.16.10 Метод getFieldsEnumerator

Позволяет перечислить объекты типа `Field`.

Пример:

```
FieldsEnumerator fieldsEnumerator = document.getBlocks().getFieldsEnumerator();
while (fieldsEnumerator.MoveNext()) {
    Field field = fieldsEnumerator.Current;
    if (field != null) {
        Console.WriteLine(field.getRange().extractText());
    }
}
```

4.28.17 Класс Block

Класс `Block` является базовой для всех блоков документа. От нее наследуются таблицы `Paragraph`, `Table`, `Shape`, `Field`.

4.28.17.1 Методы toParagraph, toTable, toShape, toField

Преобразует объект `Block` в блок соответствующего типа. Если это невозможно, возвращает `null`.

```
Paragraph toParagraph();
Table toTable();
Shape toShape();
Field toField();
```

4.28.17.2 Метод 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());
}
```

4.28.17.3 Метод 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();
}
```

4.28.17.4 Метод getSection

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());
}
```

4.28.18 Класс Borders

Класс `Borders` предоставляет методы для управления границами диапазона ячеек. Он позволяет установить внутренние границы другого стиля, отличающегося от внешних границ (слева, справа, сверху и снизу).

4.28.18.1 Методы для считывания свойств

Методы позволяют получить настройки свойств границы для отдельной ячейки. Если граница не установлена, то поля структуры `LineProperties` имеют значение `null`.

```
LineProperties getLeft();
LineProperties getRight();
LineProperties getTop();
LineProperties getBottom();
LineProperties getDiagonalDown();
LineProperties getDiagonalUp();
LineProperties getInnerHorizontal();
LineProperties getInnerVertical();
```

4.28.18.2 Методы для установки свойств

Методы позволяют устанавливать настройки свойств `LineProperties` границы для отдельной ячейки. Свойства границ, у которых свойства имеют значение `null`, не будут изменены.

```
Borders setLeft(LineProperties lineProperties);
Borders setRight(LineProperties lineProperties);
Borders setTop(LineProperties lineProperties);
Borders setBottom(LineProperties lineProperties);
```

```
Borders setDiagonalDown(LineProperties lineProperties);
Borders setDiagonalUp(LineProperties lineProperties);
Borders setOuter(LineProperties lineProperties);
Borders setDiagonals(LineProperties lineProperties);
Borders setInnerHorizontal(LineProperties lineProperties);
Borders setInnerVertical(LineProperties lineProperties);
Borders setInner(LineProperties lineProperties);
Borders setAll(LineProperties lineProperties);
```

4.28.19 Класс RangeBorders

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

4.28.19.1 Методы для считывания свойств

Позволяет получить свойства `LineProperties` для внутренних и внешних (но не диагональных) границ. Свойства границ, имеющие значение `null`, являются неопределенными.

```
LineProperties getAll();
LineProperties getLeft();
LineProperties getRight();
LineProperties getTop();
LineProperties getBottom();
LineProperties getDiagonalDown();
LineProperties getDiagonalUp();
```

Метод `getInner` позволяет получить свойства внутренних границ. Свойства границ, имеющие значение `null`, являются неопределенными.

```
LineProperties getInner();
```

Следующие методы позволяют получить внутренние свойства границы:

```
LineProperties getInnerHorizontal();
LineProperties getInnerVertical();
```

4.28.20 Класс CellPosition

Класс `CellPosition` представляет положение ячейки в таблице. Значение 0, 0 соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.

Свойства клааса:

- `row` – номер строки ячейки, нумерация начинается с нуля;
- `column` – номер столбца ячейки, нумерация начинается с нуля.

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);
```

4.28.20.1 Метод 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());
```

4.28.21 Класс CellRangePosition

Класс `CellRangePosition` представляет положение диапазона ячеек в таблице.

Список свойств:

- `topLeft` – 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` – 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);
```

4.28.21.1 Метод 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)]
```

4.28.22 Класс CellRange

Класс `CellRange` представляет диапазон (коллекцию) ячеек.

Пример:

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

4.28.22.1 Метод getTable

Возвращает таблицу `Table`, содержащую текущий диапазон.

4.28.22.2 Метод getBeginRow

Возвращает индекс начальной строки диапазона.

Пример:

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

4.28.22.3 Метод getBeginColumn

Возвращает индекс начального столбца диапазона.

Пример:

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

4.28.22.4 Метод getLastRow

Возвращает индекс последней строки диапазона.

Пример:

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

4.28.22.5 Метод getLastColumn

Возвращает индекс последнего столбца диапазона.

Пример:

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

4.28.22.6 Метод GetEnumerator

Возвращает объект типа `CellEnumerator` для доступа к коллекции ячеек.

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());
}
```

4.28.22.7 Метод setBorders

Устанавливает границы ячеек в диапазоне.

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);
```

4.28.22.8 Метод 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.backgroundColor = new ColorRGBA(55, 146, 179, 200);
cellRange.setCellProperties(cellProperties);
```

4.28.22.9 Метод 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.backgroundColor.r);
```

4.28.22.10 Метод 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));
```

4.28.22.11 Метод merge

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

Пример:

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

4.28.22.12 Метод unmerge

Метод разъединяет ранее объединенные ячейки.

Пример:

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

4.28.23 Класс Cell

Класс `Cell` представляет отдельную ячейку на листе электронной таблицы, либо ячейку таблицы в составе текстового документа.

4.28.23.1 Метод getRange

Возвращает диапазон, позволяющий работать с содержимым ячейки как с абзацем (`Paragraph`) текста.

4.28.23.2 Метод getFormat

Возвращает тип данных ячейки (`CellFormat`: Общий, Числовой, Денежный, Текстовый и т.д.).

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

// Формат : Экспоненциальный
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
```

4.28.23.4 Метод getCustomFormat

Возвращает строку формата ячейки.

Пример:

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

4.28.23.5 Метод setCustomFormat

Устанавливает формат ячейки. В случае невозможности установки формата ячейки возникает исключение ParseError.

Пример:

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

4.28.23.6 Методы setBool, setNumber, setText

Устанавливают для ячейки значение соответствующего типа.

```
void setBool(bool value);
void setNumber(double value);
void setText(string value);
```

4.28.23.7 Метод setFormula

Метод используется для ввода формулы в ячейку.

Формула – это любое выражение в ячейке, которое начинается со знака равенства (=). Формулы могут содержать функции, значения, адреса ячеек, имена, операторы действий и др.

Функция – это предустановленная формула приложения МойОфис Таблица, для вычисления которой необходимо использовать аргументы. Полный список функций приведен в Приложении 1 «Перечень функций и их описание» в документе «МойОфис Таблица. Руководство пользователя».

Основные принципы ввода формул и функций:

- формула всегда начинается со знака равенства (=);
- после знака равенства могут следовать функции, константы, адреса ячеек, операторы действий и другие элементы;
- все открывающие и закрывающие скобки должны быть согласованы;
- обязательные аргументы используемых функций должны быть указаны;
- константы не должны содержать символ «$».

В случае невозможности ввода формулы возникает исключение ParseError.

Пример:

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

4.28.23.8 Метод 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());
```

4.28.23.9 Метод 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());
```

4.28.23.10 Метод 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.

4.28.23.11 Метод getRawValue

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

Пример:

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

4.28.23.12 Метод 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");
```

4.28.23.13 Методы getCellProperties, setCellProperties

Методы для управления оформлением ячейки (`CellProperties`: фон, вертикальное выравнивание и т. д.). Неустановленные свойства, которые имеют значение `null` не будут изменены.

Примеры:

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

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

4.28.23.14 Метод getBorders

Возвращает границы отдельной ячейки `Borders`

Пример:

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

4.28.23.15 Метод isPivotTableRoot

Возвращает `true`, если ячейка является корнем сводной таблицы.

4.28.23.16 Метод 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());
}
```

4.28.23.17 Метод setBorders

Позволяет получить границы ячейки.

Пример:

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

4.28.23.18 Методы getParagraphProperties, setParagraphProperties

Методы `getParagraphProperties` и `setParagraphProperties` предназначены для управления свойствами `ParagraphProperties` абзаца текста отдельной ячейки, такими как горизонтальное выравнивание текста, межстрочный интервал, межсимвольный интервал. В сочетании с методами `getCellProperties` и `setCellProperties` достигается возможность управления всеми настройками ячейки и ее содержимого.

Примеры:

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

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

4.28.23.19 Метод unmerge

Разъединяет несколько ячеек, которые были объединены ранее.

Допустимо разъединение только тех ячеек, которые были объединены ранее. После завершения операции данные, содержавшиеся в объединенной ячейке, будут помещены в верхнюю левую ячейку диапазона.

Пример:

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

4.28.24 Класс CellProperties

Содержит свойства оформления ячейки.

Список свойств:

- `verticalAlignment` – вертикальное выравнивание ячейки `VerticalAlignment`;
- `backgroundColor` – цвет фона ячейки `ColorRGBA`;
- `textLayout` – свойства форматирования текста в ячейке `TextLayout`;
- `textOrientation` – ориентация текста в ячейке `TextOrientation`.

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.backgroundColor = new ColorRGBA(255, 255, 0, 255);
cellProps.textOrientation = new TextOrientation(45);

cell.setCellProperties(cellProps);
```

4.28.25 Класс LineProperties

Класс `LineProperties` содержит свойства оформления линий.

Список свойств:

- `style` – стиль линии `LineStyle`;

- `width` – ширина линии;
- `color` – цвет линии `Color`;
- `headLineEndingProperties` – оформление начала линии `LineEndingProperties`;
- `tailLineEndingProperties` – оформление конца линии `LineEndingProperties`.

Пример:

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

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

Borders borders = Borders();
borders.setTop(lineProperties);
cell.setBorders(borders);
```

4.28.26 Класс LineEndingProperties

Класс `LineEndingProperties` содержит свойства оформления окончаний линий.

Список свойств:

- `style` – стиль окончания линии `LineStyle`;
- `relativeExtent` – размер окончания линии относительно ее ширины.

Пример:

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

LineProperties lineProperties = LineProperties();
lineProperties.headLineEndingProperties = LineEndingProperties();
lineProperties.headLineEndingProperties.get().style = LineEndingStyle::Arrow;
lineProperties.headLineEndingProperties.get().relativeExtent = Size<float>();
lineProperties.headLineEndingProperties.get().relativeExtent.get().width = 2;
lineProperties.headLineEndingProperties.get().relativeExtent.get().height = 2;
```

```
lineProperties.tailLineEndingProperties = LineEndingProperties();
lineProperties.tailLineEndingProperties.get().style = LineEndingStyle::Arrow;
lineProperties.tailLineEndingProperties.get().relativeExtent = Size<float>();
lineProperties.tailLineEndingProperties.get().relativeExtent.get().width = 2;
lineProperties.tailLineEndingProperties.get().relativeExtent.get().height = 2;

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

Borders borders = Borders();
borders.setTop(lineProperties);
cell.setBorders(borders);
```

4.28.27 Класс LineSpacing

Позволяет управлять межстрочным интервалом.

Список свойств:

- `value` – значение межстрочного интервала;
- `rule` – тип 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;
```

4.28.28 Класс Position

Класс `Position` обозначает местоположение в документе при вставке нового объекта.

4.28.28.1 Метод insertText

Вставляет текст в указанную позицию.

Пример:

```
Range range = document.getRange();
Position startPosition = range.getBegin();
startPosition.insertText("Текст в начале строки");
```

4.28.28.2 Метод insertTable

Метод предназначен для вставки таблицы с заданным числом строк и столбцов в заданное местоположение в документе. Возвращает объект таблицы.

Следует учитывать, что при вставке таблицы к ее имени автоматически добавляется порядковый номер, начинающийся с единицы. Таким образом, вызов

```
Table table = position.insertTable(3, 3, "Table");
```

приведет к созданию в текстовом документе таблицы с именем «Table1».

Пример вставки таблицы в текстовый документ:

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

В табличном документе данный метод используется для вставки нового рабочего листа.

Пример вставки нового листа в табличный документ:

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

При невозможности вставки таблицы вызывает исключение DocumentModificationError.

4.28.28.3 Метод insertPageBreak

Вставляет разделитель страниц в указанную позицию.

Пример:

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

4.28.28.4 Метод insertLineBreak

Вставляет переход на следующую строку, не разрывающий абзац, в указанную позицию.

Пример:

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

4.28.28.5 Метод insertBookmark

Вставляет закладку с наименованием `name` в текущую позицию.

Пример:

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

4.28.28.6 Метод insertImage

Вставляет рисунок, размещенный в файле, в текущую позицию. С помощью параметра `url` задается полный путь к файлу. Параметр `size` задает геометрические размеры изображения для вставки.

Пример:

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

4.28.28.7 Метод insertSectionBreak

Вставляет в позицию разрыв раздела.

Пример:

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

4.28.28.8 Метод removeBackward

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

Пример:

```
Range range = document.getRange();
Position beginPosition = range.getBegin();
beginPosition.removeBackward(3);
```

4.28.28.9 Метод removeForward

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

Пример:

```
Range range = document.getRange();
Position beginPosition = range.getBegin();
beginPosition.removeForward(3);
```

4.28.29 Класс Range

Класс `Range` управляет информацией о диапазоне позиций в документе и предоставляет начальную и конечную позицию.

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();
while (sectionsEnumerator.MoveNext()) {
    Section section = sectionsEnumerator.Current;
    Console.WriteLine(section.getRange().extractText());
};

-- диапазон комментариев
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
while (commentsEnumerator.MoveNext()) {
    Comment comment = commentsEnumerator.Current;
    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 enumerator = headers.GetEnumerator();
    while (enumerator.MoveNext()) {
        HeaderFooter headerFooter = enumerator.Current;
        Console.WriteLine(headerFooter.getRange().extractText());
    };
}

-- диапазон отслеживаемых изменений
TrackedChangesEnumerator enumerator =
document.getRange().getTrackedChangesEnumerator();
while (enumerator.MoveNext()) {
    TrackedChange trackedChange = enumerator.Current;
    Console.WriteLine(trackedChange.getRange().extractText());
};
```

4.28.29.1 Метод getBegin

Возвращает начальную позицию диапазона Position, значение только для чтения.

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");
}
```

4.28.29.2 Метод getEnd

Возвращает конечную позицию диапазона Position, значение только для чтения.

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");
}
```

4.28.29.3 Метод extractText

Возвращает текстовое представление содержимого диапазона. При этом часть содержимого (например, фотографии) будет пропущена, часть (например, таблицы) будет преобразована.

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());
}
```

4.28.29.4 Метод removeContent

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());
}
```

4.28.29.5 Метод lockContent

Метод запрещает изменения содержимого диапазона. В случае попытки изменения вызывается исключение DocumentModificationError.

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());
}
```

4.28.29.6 Метод 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());
}
```

4.28.29.7 Метод 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());
}
```

4.28.29.8 Метод replaceText

Заменяет содержимое диапазона 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");
}
```

4.28.29.9 Метод 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);
}
```

4.28.29.10 Метод setTextProperties

Настройка оформления текста. Неустановленные свойства (которые имеют значение `null`) не будут изменены.

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);
}
```

4.28.29.11 Метод getBlocksEnumerator

Возвращает объект типа `BlocksEnumerator` для доступа к коллекции блоков в диапазоне.

Пример для текстового документа:

```
Range range = document.getRange();
BlocksEnumerator blocksEnumerator = range.getBlocksEnumerator();
while (blocksEnumerator.MoveNext()) {
    Block block = blocksEnumerator.Current;
    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();
```

```
    while (blocksEnumerator.MoveNext()) {
        Block block = blocksEnumerator.Current;
        Console.WriteLine(block.getRange().extractText());
    };
}
```

4.28.29.12 Метод getTrackedChangesEnumerator

Возвращает объект типа `TrackedChangesEnumerator` для доступа к коллекции отслеживаемых изменений.

Пример:

```
TrackedChangesEnumerator enumerator =
document.getRange().getTrackedChangesEnumerator();
while (enumerator.MoveNext()) {
    TrackedChange trackedChange = enumerator.Current;
    Console.WriteLine(trackedChange.getRange().extractText());
};
```

4.28.29.13 Метод getComments

Обеспечивает доступ к комментариям в диапазоне. Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам, если таковые имеются. Если дат нет, то порядок комментариев не определен.

Пример:

```
var commentsEnumerator = document.getComments().GetEnumerator();
while (commentsEnumerator.MoveNext())
{
    var comment = commentsEnumerator.Current;
    Console.WriteLine(comment.getText());
    Console.WriteLine(comment.getInfo().author.name);
    Console.WriteLine(comment.getRange().extractText());
}
```

4.28.29.14 Метод getParagraphs

Обеспечивает доступ к абзацам в диапазоне.

Пример для текстового документа:

```
Paragraphs paragraphs = document.getRange().getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
```

```
while (paragraphsEnumerator.MoveNext()) {
    Paragraph paragraph = paragraphsEnumerator.Current;
    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();
    while (paragraphsEnumerator.MoveNext()) {
        Paragraph paragraph = paragraphsEnumerator.Current;
        Console.WriteLine(paragraph.getRange().extractText());
    };
}
```

4.28.29.15 Метод getImages

Обеспечивает доступ к изображениям в диапазоне.

Пример:

```
Images images = document.getRange().getImages();
ImagesEnumerator imagesEnumerator = images.GetEnumerator();
while (imagesEnumerator.MoveNext()) {
    Image image = imagesEnumerator.Current;
    Console.WriteLine(image.getFrame().getWrapType());
};
```

4.28.30 Класс Search

Класс `Search` предоставляет интерфейс для выполнения поиска в документе.

```
class Search
```

4.28.30.1 Метод findText

Обеспечивает поиск текста в документе, без учета регистра.

```
virtual RangesEnumerator findText(string text);
```

Обеспечивает поиск текста в указанном диапазоне, без учета регистра.

```
virtual RangesEnumerator findText(string text, Range range);
```

Примеры:

```
-- Поиск по всему документу
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult  = search.findText("API");
while (searchResult.MoveNext()) {
    var range = searchResult.Current;
    Console.WriteLine(range.extractText());
}
```

```
-- Поиск только в диапазоне первого блока
Block firstBlock = document.getBlocks().getBlock(0);
Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult  = search.findText("API", firstBlock);
while (searchResult.MoveNext()) {
    var range = searchResult.Current;
    Console.WriteLine(range.extractText());
}
```

4.28.30.2 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);
search.findText("API");
```

4.28.31 Класс TextProperties

Класс `TextProperties` содержит свойства оформления фрагмента текста (диапазона). Это может быть либо весь абзац, либо его часть, либо даже текст в нескольких ячейках в таблице. Свойства оформления текста могут быть применены к любому объекту `Range`.

```
class TextProperties
```

Список свойств:

- `fontName` – наименование шрифта;
- `fontSize` – размер шрифта;
- `bold` – принимает значение `true`, если шрифт текста «полужирный»;
- `italic` – принимает значение `true`, если шрифт текста «курсив»;

- `underline` – принимает значение `true`, если шрифт текста «подчеркнутый»;
- `strikethtough` – принимает значение `true`, если шрифт текста «зачеркнутый»;
- `allCapitals` – принимает значение `true`, если в тексте все символы прописные;
- `scriptPosition` – тип надстрочного/подстрочного форматирования;
- `textColor` – цветовая модель оформления текста;
- `backgroundColor` – цветовая модель фона текста;
- `characterSpacing` – межзнаковый интервал.

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);
}
```

4.28.32 Класс Bookmarks

Предоставляет доступ к операциям с закладками в документе.

4.28.32.1 Метод getBookmarkRange

Возвращает экземпляр объекта Range для дальнейшей работы с содержимым закладки.

Пример:

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

4.28.32.2 Метод removeBookmark

Удаляет закладку по ее названию.

Пример:

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

4.28.33 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`.

4.28.33.1 Метод getRange

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

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator enumerator = comments.GetEnumerator();
if (enumerator != null) {
    while (enumerator.MoveNext()) {
        Comment comment = enumerator.Current;
        Range commentRange = comment.getRange();
        Console.WriteLine(commentRange.extractText());
    };
}
```

4.28.33.2 Метод getText

Метод возвращает текст комментария.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator enumerator = comments.GetEnumerator();
if (enumerator != null) {
    while (enumerator.MoveNext()) {
        Comment comment = enumerator.Current;
        Console.WriteLine(comment.getText());
```

```
    };
}
```

4.28.33.3 Метод getAudioUrl

Метод возвращает путь к файлу аудиокомментария.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator enumerator = comments.GetEnumerator();
if (enumerator != null) {
    while (enumerator.MoveNext()) {
        Comment comment = enumerator.Current;
        Console.WriteLine(comment.getAudioUrl());
    };
}
```

4.28.33.4 Метод getInfo

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

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator enumerator = comments.GetEnumerator();
if (enumerator != null) {
    while (enumerator.MoveNext()) {
        Comment comment = enumerator.Current;
        TrackedChangeInfo info = comment.getInfo();
        if (info != null) {
            Console.WriteLine(info.author);
        }
    };
}
```

4.28.33.5 Метод isResolved

Метод возвращает значение `true`, если комментарий принят.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator enumerator = comments.GetEnumerator();
```

```
if (enumerator != null) {
    while (enumerator.MoveNext()) {
        Comment comment = enumerator.Current;
        Console.WriteLine(comment.isResolved());
    };
}
```

4.28.33.6 Метод getReplies

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

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
if (commentsEnumerator != null) {
    while (commentsEnumerator.MoveNext()) {
        Comment comment = commentsEnumerator.Current;
        Comments replies = comment.getReplies();
        if (replies != null) {
            CommentsEnumerator repliesEnumerator = replies.GetEnumerator();
            if (repliesEnumerator != null) {
                while (repliesEnumerator.MoveNext()) {
                    Comment reply = repliesEnumerator.Current;
                    Console.WriteLine(reply.isResolved());
                }
            }
        }
    }
}
```

4.28.34 Класс Comments

Класс `Comments` содержит коллекцию комментариев диапазона (см. Рисунок 1).

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

Пример:

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

4.28.34.1 Метод GetEnumerator

Метод возвращает коллекцию комментариев всего документа.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator enumerator = comments.GetEnumerator();
while (enumerator.MoveNext()) {
    Comment comment = enumerator.Current;
    Console.WriteLine(comment.getInfo().author);
};
```

4.28.35 Класс TrackedChange

Класс `TrackedChange` предоставляет интерфейс для отслеживания изменений в документе.

Пример:

```
TrackedChangesEnumerator trackedChangesEnumerator =
document.getRange().getTrackedChangesEnumerator();
if (trackedChangesEnumerator != null) {
    while (trackedChangesEnumerator.MoveNext()) {
        TrackedChange trackedChange = trackedChangesEnumerator.Current;
        Console.WriteLine(trackedChange.getRange().extractText());
    }
}
```

4.28.35.1 Метод getRange

Возвращает объект `Range`, который соответствует измененному диапазону внутри абзаца.

Пример:

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

4.28.35.2 Методы getType

Позволяет получить тип отслеживаемого изменения TrackedChangeType.

Пример:

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

4.28.35.3 Методы getInfo

Позволяет получить информацию об отслеживаемых изменениях TrackedChangeInfo.

Пример:

```
var trackedChangesEnumerator =
document.getRange().getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator)
{
    Console.WriteLine(trackedChange.getInfo().author.name);
}
```

4.28.36 Класс TrackedChangeInfo

Класс TrackedChangeInfo содержит информацию об авторе и дате отслеживаемого изменения.

Список свойств:

- author – автор изменений;
- timeStamp – дата и время изменений DateTime.

Пример:

```
TrackedChangesEnumerator enumerator =
document.getRange().getTrackedChangesEnumerator();
while (enumerator.MoveNext()) {
    TrackedChange trackedChange = enumerator.Current;
    Console.WriteLine(trackedChange.getInfo().author);
```

```
    Console.WriteLine(trackedChange.getInfo().timeStamp);
};
```

4.28.37 Класс Section

Класс `Section` представляет собой раздел в документе.

4.28.37.1 Метод setPageProperties

Устанавливает высоту и ширину страниц раздела `PageProperties`.

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);
}
```

4.28.37.2 Метод getPageProperties

Возвращает размеры высоты и ширины `PageProperties` для страниц раздела.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator enumerator = sections.GetEnumerator();
while (enumerator.MoveNext()) {
    Section section = enumerator.Current;
    PageProperties pageProperties = section.getPageProperties();
    Console.WriteLine(pageProperties.height);
};
```

4.28.37.3 Метод setPageOrientation

Задает ориентацию `PageOrientation` страниц раздела.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator enumerator = sections.GetEnumerator();
while (enumerator.MoveNext()) {
    Section section = enumerator.Current;
```

```
    PageProperties pageProperties = section.getPageProperties();
    Console.WriteLine(pageProperties.height);
};
```

4.28.37.4 Метод getPageOrientation

Возвращает ориентацию страниц раздела `PageOrientation` или неопределенное значение, если ориентация не установлена.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator enumerator = sections.GetEnumerator();
while (enumerator.MoveNext()) {
    Section section = enumerator.Current;
    Console.WriteLine(section.getPageOrientation());
};
```

4.28.37.5 Метод getRange

Возвращает диапазон `Range` в документе, соответствующий данному разделу.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator enumerator = sections.GetEnumerator();
while (enumerator.MoveNext()) {
    Section section = enumerator.Current;
    Range range = section.getRange();
    Console.WriteLine(range.extractText());
};
```

4.28.37.6 Метод getHeaders

Предоставляет доступ к коллекции верхних колонтитулов `HeadersFooters`, содержащихся в разделе.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator enumerator = sections.GetEnumerator();
while (enumerator.MoveNext()) {
    Section section = enumerator.Current;
    HeadersFooters headers = section.getHeaders();
```

```
    Console.WriteLine(headers.getType());
};
```

4.28.37.7 Метод getFooters

Предоставляет доступ к коллекции нижних колонтитулов `HeadersFooters`, содержащихся в разделе.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator enumerator = sections.GetEnumerator();
while (enumerator.MoveNext()) {
    Section section = enumerator.Current;
    HeadersFooters footers = section.getFooters();
    Console.WriteLine(footers.getType());
};
```

4.28.38 Класс Sections

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

4.28.38.1 Метод getEnumerator

Метод позволяет перечислить коллекцию секций документа.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
while (sectionsEnumerator.MoveNext()) {
    Section section = sectionsEnumerator.Current;
    Console.WriteLine(section.getRange().extractText());
};
```

4.28.39 Класс PageProperties

Класс `PageProperties` содержит размеры страницы.

Список свойств:

- `height` – высота страницы;
- `width` – ширина страницы;

- `margins` – поля страницы.

Примеры:

```
PageProperties pageProperties = section.getPageProperties();
pageProperties.height = 100;
pageProperties.width = 200;
section.setPageProperties(pageProperties);
```

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

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

4.28.40 Класс Insets

Класс `Insets` содержит поля страницы (`left`, `right`, `top`, `bottom`).

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);
```

Список свойств:

- `left` – поле слева;
- `top` – поле сверху;
- `right` – поле справа;
- `bottom` – поле снизу.

4.28.41 Класс HeaderFooter

Класс `HeaderFooter` определяет отдельный колонтитул документа.

4.28.41.1 Метод getType

Предоставляет информацию о типе колонтитула HeaderFooterType.

Пример:

```
HeaderFooter headerFooter = headersEnumerator.Current;
HeaderFooterType headerFootertype = headerFooter.getType();
Console.WriteLine(headerFootertype);
```

4.28.41.2 Метод getBlocks

Предоставляет доступ к блокам, которые содержатся в колонтитуле.

Пример:

```
HeaderFooter headerFooter = headersEnumerator.Current;
Blocks blocks = headerFooter.getBlocks();
BlocksEnumerator blocksEnumerator = blocks.GetEnumerator();
while (blocksEnumerator.MoveNext()) {
    Block block = blocksEnumerator.Current;
    Console.WriteLine(block.getRange().extractText());
};
```

4.28.41.3 Метод getRange

Предоставляет диапазон с содержанием верхнего или нижнего колонтитулов.

Пример:

```
HeaderFooter headerFooter = headersEnumerator.Current;
Range range = headerFooter.getRange();
Console.WriteLine(range.extractText());
```

4.28.42 Класс HeadersFooters

Класс HeadersFooters представляет коллекцию верхних и нижних колонтитулов в разделе (Section) документа.

4.28.42.1 Метод GetEnumerator

Возвращает объект HeaderFootersEnumerator для доступа к коллекции колонтитулов.

Пример:

```
HeadersFooters headers = section.getHeaders();
HeaderFootersEnumerator headersEnumerator = headers.GetEnumerator();
```

```
while (headersEnumerator.MoveNext()) {
    HeaderFooter headerFooter = headersEnumerator.Current;
    Console.WriteLine(headerFooter.getType());
};
```

4.28.43 Класс TextOrientation

Класс `TextOrientation` предоставляет доступ к свойствам ориентации текста в ячейке, фигуре и т. д (см. `CellProperties`).

Пример:

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

CellProperties cellProps = cell.getCellProperties();
CellProps.textOrientation = new TextOrientation(45);
cell.setCellProperties(cellProps);
```

4.28.43.1 Метод getAngle

Возвращает угол направления текста в ячейке. Значение угла указывается в градусах.

Пример:

```
CellProperties cellProps = cell.getCellProperties();
CellProps.textOrientation = new TextOrientation(45);
if (cellProps.textOrientation.has_value()) {
    Console.WriteLine(cellProps.textOrientation.getAngle());
}
```

4.28.44 Класс TextExportSettings

Класс `TextExportSettings` содержит настройки для экспорта текстовых документов.

Свойство:

- `pageNumbers` – коллекция страниц для экспорта (нечетные, четные, список и т.д.). Тип: `PageNumbers`.

Пример:

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

4.28.45 Класс WorkbookExportSettings

Класс `WorkbookExportSettings` содержит настройки для экспорта табличных документов.

Список свойств:

- `sheetNames` – представляет коллекцию имен листов для экспорта. Если коллекция пуста, экспортируются все листы. Тип: `VectorString`;
- `printingScope` – представляет область печати (весь документ, область печати, пользовательский диапазон и т. д.). Тип: `PrintingScope`;
- `pageProperties` – представляет свойства страницы для выходного документа (высота/ширина страницы и т. д.). Тип: `PageProperties`;
- `scale` – представляет масштаб экспорта выходного документа в процентах (например, 50,0%, 150,63%, 400,0% и т. д.). Тип: `float`.

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);
```

4.28.46 Класс PrintingScope

Класс `PrintingScope` содержит настройки для экспорта табличных документов.

Возможные значения:

- `PrintArea` – представляет выбранную область печати;
- `WholeSheet` – представляет область печати – весь документ.

4.28.46.1 Метод PrintingScope

Конструктор по умолчанию:

```
PrintingScope();
```

Конструктор для создания области печати типа `PrintArea` или `WholeSheet`:

```
PrintingScope(PrintingScope.Type type);
```

Конструктор для создания области печати типа `CellRangePosition`:

```
PrintingScope(CellRangePosition range);
```

4.28.46.2 Метод usePrintArea

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

4.28.46.3 Метод getCellRange

Метод возвращает диапазон `CellRangePosition` ячеек таблицы.

4.28.47 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);
```

4.28.47.1 Метод 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));
```

4.28.47.2 Метод getLast

Метод `PageNumbers::getLast` возвращает последний номер страницы.

Пример:

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

4.28.48 Класс Color

Класс `Color` представляют собой цветовой объект RGBA, либо заданные цвета идентификатора темы.

Пример:

```
Color color = new Color();
Color rgbaColor = new Color(new ColorRGBA(255, 0, 0, 255));
Color themeColor = new Color(ThemeColorID.Text1);
```

4.28.48.1 Метод setTransforms

Метод устанавливает правило трансформации цвета, реализованное классом `ColorTransforms`.

Пример:

```
Color color = new Color(new ColorRGBA(255, 0, 0, 255));
ColorTransforms colorTransforms = new ColorTransforms();
colorTransforms.apply(new ColorRGBA(0xF0, 0xF0, 0xF0, 0xF0));
color.setTransforms(colorTransforms);
```

4.28.48.2 Метод getTransforms

Метод возвращает объект `ColorTransforms`, реализующий трансформацию цвета.

Пример:

```
Color color = new Color(new ColorRGBA(255, 0, 0, 255));
ColorTransforms colorTransforms = color.getTransforms();
```

4.28.48.3 Метод 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);
}
```

4.28.48.4 Метод getThemeColorID

Метод возвращает цвет идентификатора темы ThemeColorID или null.

Пример:

```
Color color = new Color(new ColorRGBA(255, 0, 0, 255));
ThemeColorID? themeColorId = color.getThemeColorID();
if (themeColorId == null && themeColorId.HasValue) {
    Console.WriteLine(themeColorId.Value);
}
```

4.28.49 Класс ColorTransforms

Класс ColorTransforms предназначен для конвертации цвета.

4.28.49.1 Метод apply

Метод задает правило конвертирования цвета. Принимает входящий параметр ColorRGBA color и на его основе возвращает цвет с применением конвертации.

Пример:

```
ColorTransforms colorTransforms = new ColorTransforms();
colorTransforms.apply(new ColorRGBA(0xF0, 0xF0, 0xF0, 0xF0));
```

4.28.50 Класс Frame

Класс Frame представляет рамку встроенного объекта. Класс предназначен для изменения положения и геометрии встроенного объекта, управления переносом текста вокруг объекта.

4.28.50.1 Метод setPosition

Метод задает положение встроенного объекта. Для текстовых документов позиция может быть установлена только для встроенных объектов, тип переноса текста которых не является `TextWrapType.Inline`.

4.28.50.2 Метод getPosition

Метод возвращает позицию `AnchoredPosition` встроенного объекта на странице, если позиция не определена, то возвращает `null`.

4.28.50.3 Метод setDimensions

Метод задает размеры (изменяет размер) встроенного объекта. Размеры считаются неопределенными, если их значение равно `null`.

4.28.50.4 Метод getDimensions

Возвращает размеры встроенного объекта или `null`, если размеры не определены.

4.28.50.5 Метод setWrapType

Устанавливает вариант обтекания `TextWrapType` текстом встроенного объекта.

4.28.50.6 Метод getWrapType

Возвращает вариант обтекания `TextWrapType` текстом встроенного объекта.

4.28.51 Класс Image

Класс `Image` представляет собой изображение, как встроенный объект.

4.28.51.1 Метод getFrame

Метод возвращает область встроенного объекта `Frame`.

4.28.52 Класс Images

Класс `Images` представляет собой интерфейс для доступа к коллекции изображений.

4.28.52.1 Метод GetEnumerator

Возвращают объект типа `ImagesEnumerator` для доступа к коллекции изображений.

4.28.53 Класс TextAnchoredPosition

Класс `TextAnchoredPosition` представляют позицию объекта на странице

текстового документа.

4.28.53.1 Метод TextAnchoredPosition

Конструктор по умолчанию создает объект с координатами (0, 0) по горизонтали относительно столбца и по вертикали относительно верхней границы страницы.

```
TextAnchoredPosition();
```

Конструктор создает объект с позицией с заданными смещениями `horizontalOffset` по горизонтали относительно столбца и `verticalOffset` по вертикали относительно верхней границы страницы.

```
TextAnchoredPosition(float horizontalOffset, float verticalOffset);
```

Конструктор создает объект с заданными координатами `horizontal` по горизонтали и `vertical` по вертикали.

```
TextAnchoredPosition(HorizontalRelativeAnchoredPosition horizontal,
                     VerticalRelativeAnchoredPosition vertical);
```

Метод использует следующие параметры:

- `horizontal` – позиция по горизонтали `HorizontalRelativeAnchoredPosition`;
- `vertical` – позиция по вертикали `VerticalRelativeAnchoredPosition`.

4.28.54 Класс HorizontalRelativeAnchoredPosition

Класс `HorizontalRelativeAnchoredPosition` используется для управления расположением объекта относительно закрепленной позиции по горизонтали.

4.28.54.1 Метод HorizontalRelativeAnchoredPosition

Конструктор создает объект с заданным типом относительного позиционирования по горизонтали `HorizontalRelativeTo` и значением смещения `offset`.

```
HorizontalRelativeAnchoredPosition(HorizontalRelativeTo relativeTo,
                                   float offset);
```

Конструктор создает объект с заданным типом относительного позиционирования по горизонтали `HorizontalRelativeTo` и значением выравнивания `HorizontalAnchorAlignment`.

```
HorizontalRelativeAnchoredPosition(HorizontalRelativeTo relativeTo,
                                   HorizontalAnchorAlignment alignment);
```

Метод использует следующие параметры:

- `relativeTo` – это перечисление типа `HorizontalRelativeTo`;
- `offset` – значение смещения;
- `alignment` – это перечисление типа `HorizontalAnchorAlignment`.

4.28.55 Класс VerticalRelativeAnchoredPosition

Класс `VerticalRelativeAnchoredPosition` используется для управления расположением объекта относительно закрепленной позиции по вертикали.

4.28.55.1 Метод VerticalRelativeAnchoredPosition

Конструктор создает объект с заданным типом относительного позиционирования по вертикали `VerticalRelativeTo` и значением смещения `offset`.

```
VerticalRelativeAnchoredPosition(VerticalRelativeTo relativeTo, float offset);
```

Конструктор создает объект с заданным типом относительного позиционирования по вертикали `VerticalRelativeTo` и значением выравнивания `VerticalAnchorAlignment`.

```
VerticalRelativeAnchoredPosition(VerticalRelativeTo relativeTo,
                                 VerticalAnchorAlignment alignment);
```

Метод использует следующие параметры:

- `relativeTo` – перечисление типа `VerticalRelativeTo`;
- `offset` – значение смещения;
- `alignment` – это перечисление типа `VerticalAnchorAlignment`.

4.28.56 Класс AnchoredPosition

Класс `AnchoredPosition` представляет позицию с относительными смещениями на странице текстового документа.

```
class AnchoredPosition
```

4.28.56.1 Метод AnchoredPosition

Конструктор по умолчанию создает объект с закрепленной позицией в текстовом документе.

```
AnchoredPosition(TextAnchoredPosition textPosition);
```

Метод использует следующий параметр:

- `textPosition` – позиция на странице текстового документа `TextAnchoredPosition`.

4.28.57 Класс InlineObject

Класс `InlineObject` представляет собой встроенный объект, который позиционируется как символ в строке текста.

4.28.57.1 Метод toImage

Метод возвращает изображение `Image`, связанное со встроенным объектом. В случае ошибки возвращает `null`.

4.28.57.2 Метод getFrame

Метод возвращает область встроенного объекта `Frame`.

4.28.58 Класс InlineObjects

Класс `InlineObjects` представляет собой интерфейс для доступа к коллекции встроенных объектов.

4.28.58.1 Метод GetEnumerator

Возвращает объект типа `InlineObjectsEnumerator` для доступа к коллекции абзацев.

4.29 Классы и методы для работы с макрокомандами (Scripting)

4.29.1 Класс Scripting

Класс `Scripting` управляет виртуальной машиной Lua. Он предоставляет интерфейс для выполнения макрокоманд Lua, которые хранятся в электронном документе.

Пример:

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

4.29.1.1 Метод runScript

Запускает макрокоманду с указанным именем. В случае невозможности запуска макрокоманды возникает исключение `ScriptExecutionError`.

Пример:

```
Scripting scripting = DocumentAPI.createScripting(document);
scripting.runScript("ScriptName");
```

4.30 Классы для работы с коллекциями элементов

Классы предоставляют возможность последовательного доступа к элементам коллекций объектов соответствующего типа:

```
class BlocksEnumerator
class CellsEnumerator
class CommentsEnumerator
class HeaderFootersEnumerator
class ParagraphsEnumerator
class RangesEnumerator
class ScriptsEnumerator
class SectionsEnumerator
class TablesEnumerator
class ImagesEnumerator
class InlineObjectsEnumerator
class TrackedChangesEnumerator
class ShapesEnumerator
class FieldsEnumerator
class PivotTableFieldCategorysEnumerator
class PivotTableFiltersEnumerator
class PivotTableItemsEnumerator
class NamedExpressionsEnumerator
```

Классы для работы с коллекциями элементов включают свойство `Current`, которое содержит ссылку на текущий объект:

```
Block Current
Cell Current
Comment Current
HeaderFooter Current
Paragraph Current
Range Current
Script Current
Section Current
Table Current
Image Current
```

```
InlineObject Current
TrackedChange Current
Shape Current
Field Current
PivotTableFieldCategory Current
PivotTableFilter Current
PivotTableItem Current
NamedExpression Current
```

4.30.1 Метод MoveNext

Метод `MoveNext` делает текущей ссылку на следующий элемент.

4.30.2 Метод Reset

Метод `Reset` сбрасывает текущую ссылку на первый элемент.

4.31 Исключения

4.31.1 Класс ApplicationCreateError

Исключение `ApplicationCreateError` возникает в случае, когда объект `Application` не может быть создан.

```
public class DocumentAPI {
    public class ApplicationCreateError : System.ApplicationException
}
```

4.31.2 Класс IncorrectArgumentError

Исключение `IncorrectArgumentError` возникает в случае, когда один из аргументов метода или функции имеет недействительное значение.

```
public class DocumentAPI {
     public class IncorrectArgumentError : System.ApplicationException
}
```

4.31.3 Класс InvalidObjectError

Исключение `InvalidObjectError` возникает в случае, когда объект больше не может быть использован.

```
public class DocumentAPI {
    public class InvalidObjectError : System.ApplicationException
}
```

4.31.4 Класс DocumentCreateError

Исключение `DocumentCreateError` возникает в случае, когда документ не может быть создан.

```
public class DocumentAPI {
    public class DocumentCreateError : System.ApplicationException
}
```

4.31.5 Класс DocumentLoadError

Исключение `DocumentLoadError` возникает в случае, когда документ не может быть загружен.

```
public class DocumentAPI {
    public class DocumentLoadError : System.ApplicationException
}
```

4.31.6 Класс DocumentSaveError

Исключение `DocumentSaveError` возникает в случае, когда документ не может быть сохранен.

```
public class DocumentAPI {
    public class DocumentSaveError : System.ApplicationException
}
```

4.31.7 Класс DocumentExportError

Исключение `DocumentExportError` возникает в случае, когда документ не может быть экспортирован.

```
public class DocumentAPI {
    public class DocumentExportError : System.ApplicationException
}
```

4.31.8 Класс NoSuchElementError

Исключение `NoSuchElementError` возникает в случае, когда элемент не существует.

```
public class DocumentAPI {
    public class NoSuchElementError : System.ApplicationException
}
```

4.31.9 Класс NotImplementedError

Исключение `NotImplementedError` возникает в случае, если обнаружена нереализованная функциональность.

```
public class DocumentAPI {
    public class NotImplementedError : System.ApplicationException
}
```

4.31.10 Класс OutOfRangeError

Исключение `OutOfRangeError` возникает в случае обнаружения выхода значения за пределы диапазона.

```
public class DocumentAPI {
    public class OutOfRangeError : System.ApplicationException
}
```

4.31.11 Класс ParseError

Исключение `ParseError` возникает в случае, когда параметр текста не прошел синтаксический анализ.

```
public class DocumentAPI {
    public class ParseError : System.ApplicationException
}
```

4.31.12 Класс UnknownError

Исключение `UnknownError` возникает в случае, когда критическое исключение возникло по неизвестной причине. Приложение должно быть завершено, поскольку возникло неопределенное состояние ядра Document API.

```
public class DocumentAPI {
    public class UnknownError : System.ApplicationException
}
```

4.31.13 Класс ForbiddenActionError

Исключение `ForbiddenActionError` возникает в случае выполнения запрещенной операции.

```
public class DocumentAPI {
    public class ForbiddenActionError : System.ApplicationException
}
```

4.31.14 Класс DocumentModificationError

Исключение `DocumentModificationError` возникает в случае, когда невозможно выполнить операцию по изменению документа.

```
public class DocumentAPI {
    public class DocumentModificationError : System.ApplicationException
}
```

4.31.15 Класс PivotTableError

Исключение `PivotTableError` возникает в случае ошибки при работе со сводными таблицами. Например, применение фильтра, который не может быть применен к сводной таблице.

```
public class DocumentAPI {
    public class PivotTableError : System.ApplicationException
}
```

4.31.16 Класс PositionDocumentsMismatchError

Исключение `PositionDocumentsMismatchError` 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.

```
public class DocumentAPI {
    public class PositionDocumentsMismatchError : System.ApplicationException
}
```

4.31.17 Класс ScriptExecutionError

Исключение `ScriptExecutionError` возникает в случае, когда сценарий не удается выполнить.

```
public class DocumentAPI {
    public class ScriptExecutionError : System.ApplicationException
}
```

5 ВЕРСИИ DOCUMENT API

5.1 Механизм контроля версий

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.

В разделе Изменения указывается какие несовместимые изменения были внесены в Document API.

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 серьезной ошибке 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);
}
```

5.2 Изменения

Обратно несовместимые изменения в Document API отсутствуют.