МойОфис

Комплект Средств Разработки (SDK)

Руководство программиста

MYOFFICE DOCUMENT API (C#)

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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 236 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.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

СОДЕРЖАНИЕ

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

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

Таблица 1 - Сокращения и расшифровки

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение библиотеки

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

1.2 Библиотека MyOffice Document API для языка программирования C#

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

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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 {
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
                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).

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

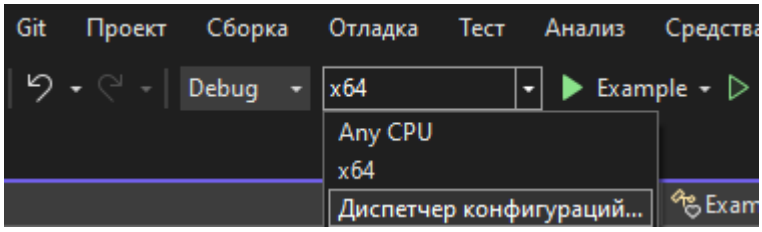


Рисунок 1 – Выбор диспетчера конфигурация в Microsoft Visual Studio

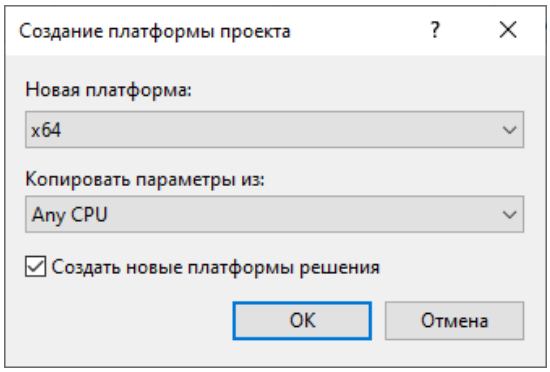


Рисунок 2 – – Окно создания платформы проекта

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**.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

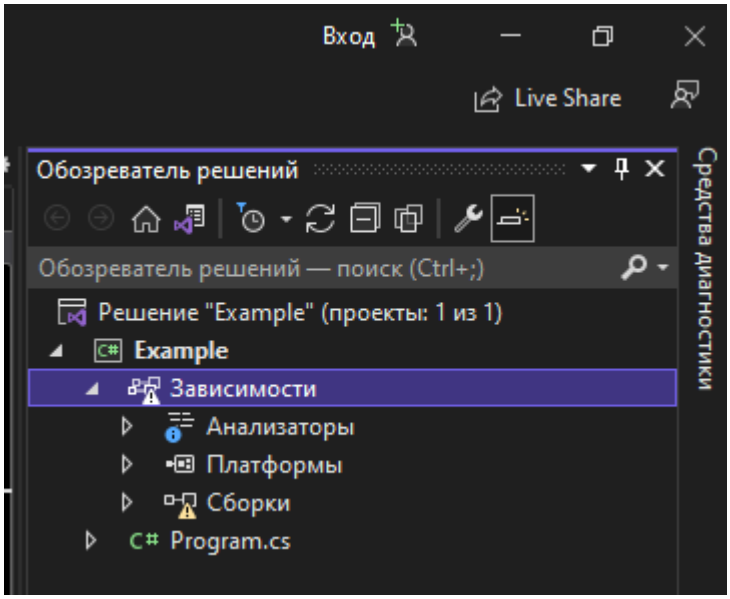


Рисунок 3 – Окно обозревателя решений Microsoft Visual Studio

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).

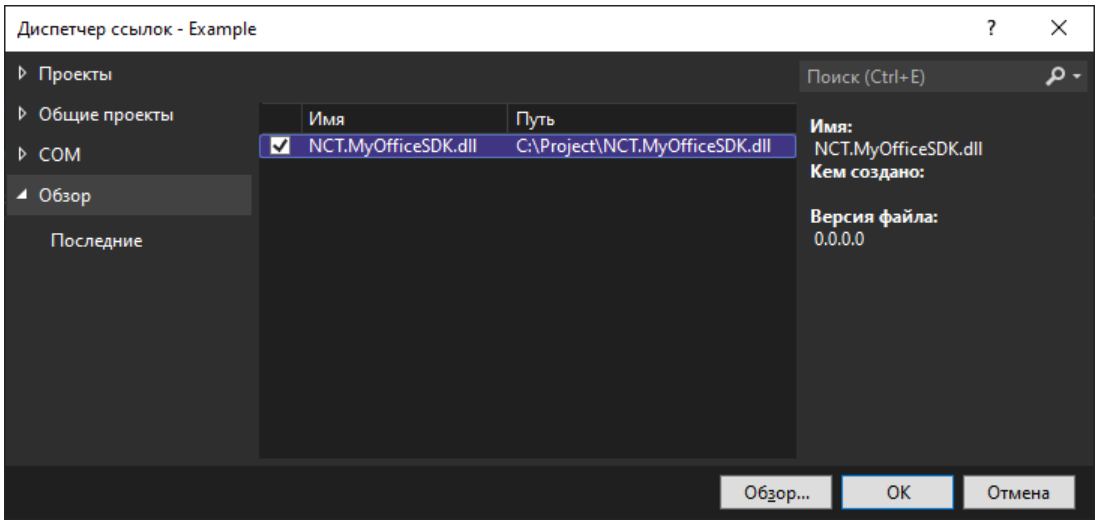


Рисунок 4 – Окно выбора файла библиотеки

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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).

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

3 ОБЪЕКТНАЯ МОДЕЛЬ МОЙОФИС C# SDK

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

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

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

Для этого используются классы, расположенные в пространстве имен (namespace) `NCT.MyOfficeSDK` (см. Рисунок 5). `NCT.MyOfficeSDK` содержит основной класс `NCT.MyOfficeSDK.Application`, который служит для создания и открытия документа, а также классы и функции для представления документа и всех его составляющих, которые поддерживает МойОфис: абзацы, таблицы, ячейки, рисунки, колонтитулы и т.д.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

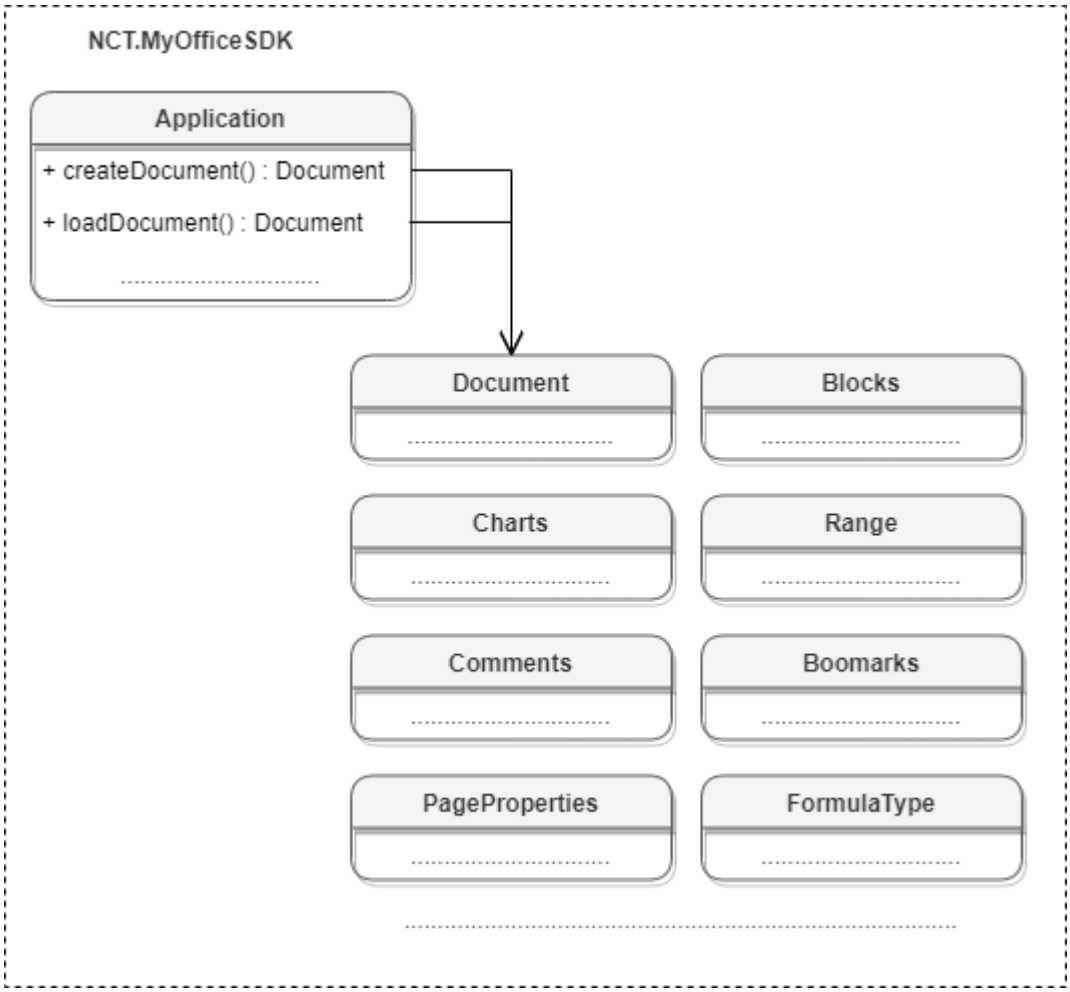


Рисунок 5 – Объектная модель МойОфис C# SDK.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

4 РАБОТА С ДОКУМЕНТАМИ

4.1 Работа с текстовым документом

4.1.1 Создание и открытие текстового документа

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

Примеры создания текстового документа:

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

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

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

Примеры загрузки текстового документа:

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

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

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

4.1.2 Сохранение и экспорт текстового документа

Метод Document::saveAs сохраняет документ по указанному пути.

Примеры сохранения текстового документа:

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

```
SaveDocumentSettings saveDocumentSettings = new SaveDocumentSettings();

saveDocumentSettings.documentFormat = DocumentFormat.OXML;
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
saveDocumentSettings.documentType = DocumentType.Text;
saveDocumentSettings.documentPassword = "password";
saveDocumentSettings.isTemplate = false;

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

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

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

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

Примеры экспорта текстового документа:

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

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

4.1.3 Разделы (секции) документа

На рисунке 6 изображена объектная модель классов, относящихся к работе с секциями текстового документа.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

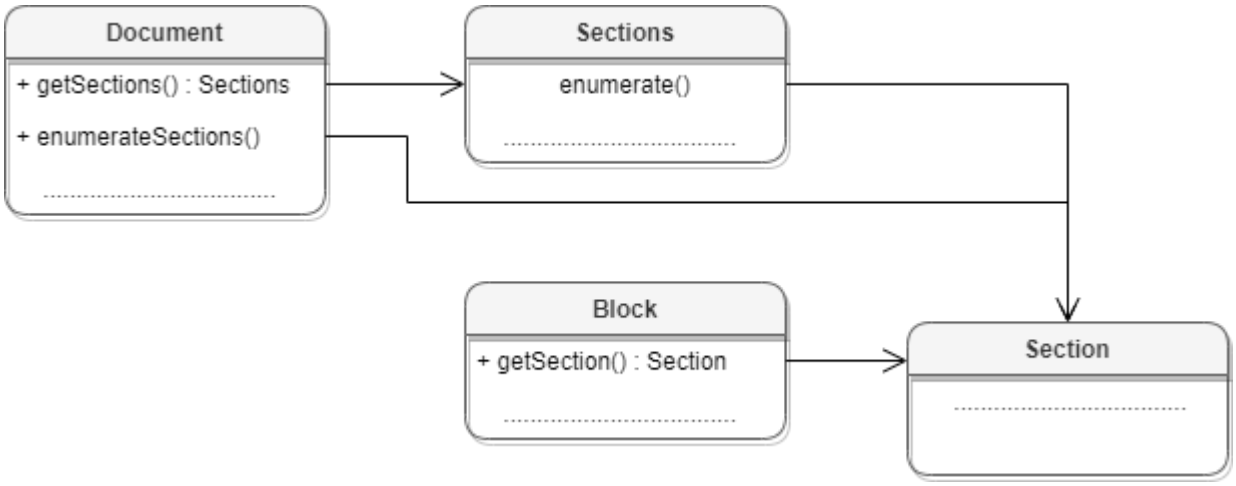


Рисунок 6 – Объектная модель классов для работы с секциями

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

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

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

Примеры:

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

4.1.3.1 Работа с колонтитулами раздела

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

Пример:

```
Application app = new Application();

// Загрузка текстового документа
// Документ sections.docx содержит несколько
// разделов (sections). На каждой странице присутствует
// верхний (header) и нижний (footer) колонтитул.
var doc = app.loadDocument("sections.docx");

var section = doc.getSectionsEnumerator().Current;

var header = section.getHeaders().GetEnumerator().Current;

var footer = section.getFooters().GetEnumerator().Current;

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

4.1.3.2 Управление ориентацией и свойствами страниц раздела

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

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

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

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

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

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

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

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

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

4.1.4 Работа со встроенными объектами

Объектная модель текстового документа МойОфис поддерживает такие графические объекты, как изображения (Image) и фигуры (Shape).

Возможности Document API в части управления изображениями развиваются и дополняются. В настоящее время доступны следующие операции:

– вставка изображений в текстовый документ;
– перечисление графических объектов, находящихся в текстовом документе, определение их типа и геометрических размеров;
– перемещение графических объектов в текстовом документе, изменение варианта обтекания текстом, позиции, размера.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

Пример:

```
var inlineObjects = document.getRange().getInlineObjects();
```

4.1.4.1 Вставка изображения

Для вставки изображения используется метод `Position::insertImage()`.

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

4.1.4.2 Перечисление встроенных объектов

```
var inlineObjects = document.getRange().getInlineObjects();
var enumerator = inlineObjects.GetEnumerator();
// Перебор коллекции встроенных объектов
foreach (var inlineObject in enumerator)
{
    Console.WriteLine(inlineObject.getFrame().getWrapType());
    Console.WriteLine(inlineObject.getFrame().getDimensions().height);
    var image = inlineObject.toImage();
    if (image != null) {
        Console.WriteLine("Image was found");
    }
}
```

4.1.4.3 Определение типа встроенных объектов

Для определения типа графического объекта (`Image`/`Shape`) может быть использован метод `InlineObject::toImage()`. В случае, если объект является изображением, метод вернет ненулевой объект.

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

4.1.4.4 Изменение параметров встроенного объекта

Размеры графического объекта могут быть получены из объекта Frame, который может быть получен посредством использованием метода InlineObject::getFrame().

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
InlineObjectsEnumerator enumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in enumerator)
{
    var frame = inlineObject.getFrame();
    Console.WriteLine(frame.getDimensions().width);
}
```

Помимо этого, 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();
if (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 TextAnchoredPosition position = new TextAnchoredPosition();

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
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 TextAnchoredPosition 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 TextAnchoredPosition position = new TextAnchoredPosition();

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
frame.setPosition(position);
```

С помощью метода Frame::setDimensions() можно изменить размеры встроенных объектов

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

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

Вариант обтекания текстом графического объекта TextWrapType может быть задан посредством использованием метода Frame::setWrapType().

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

4.1.5 Работа с таблицами текстового документа

В текстовом документе таблицы могут быть расположены являются листы документа.

Доступ к объектам Table осуществляется из Blocks (см. Рисунок 7).

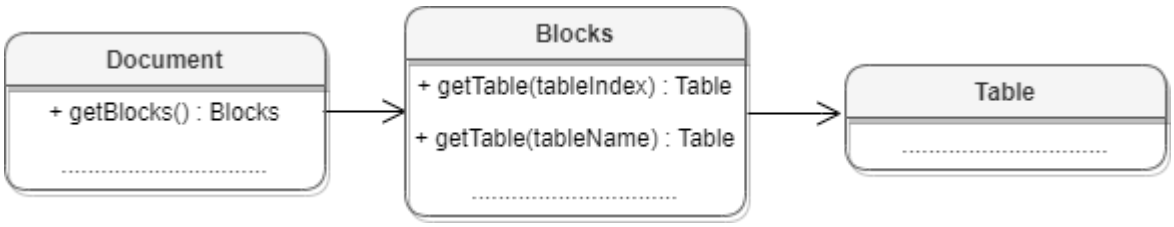


Рисунок 7 – Объектная модель для работы с таблицами

Для работы с таблицами доступны следующие операции:

– перечисление таблиц документа;
– получение таблицы документа;
– вставка таблицы в позицию документа;
– переименование таблицы;
– удаление таблицы.

Ниже приведены примеры работы с таблицами в текстовых документах:

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Перечисление таблиц документа:

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

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

Получение таблицы текстового документа:

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

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

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

Вставка таблицы в текстовый документ:

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

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

Переименование таблицы:

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

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

Удаление таблицы:

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
{
    table.remove();
}
```

4.1.6 Работа с закладками

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

Доступны следующие операции с закладками:

Вставка закладки в указанное местоположение

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

Удаление закладки с заданным именем

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

Поиск закладки по имени

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

Замена текстового содержимого закладки

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

Вставка текста в закладку

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Удаление содержимого закладки

```
Bookmarks bookmarks = document.getBookmarks();
Range bookmarkRange = bookmarks.getBookmarkRange("Boomark");
if (bookmarkRange != null) {
    bookmarkRange.getBegin().removeBackward();
}
```

Получение текстового содержимого закладки

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

Вставка таблицы в закладку

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

4.1.7 Рецензирование документов

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

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

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

Инструменты рецензирования применяются к диапазону документа, по этой причине методы доступа к ним находятся в классе `Range` (см. Рисунок 8).

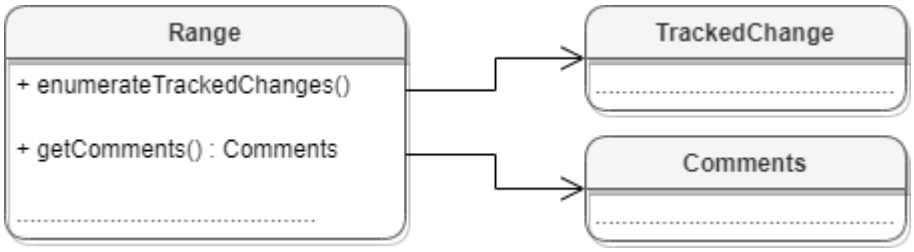


Рисунок 8 – Инструменты рецензирования документа

4.1.8 Поиск в текстовом документе

Для поиска в текстовом документе необходимо создать экземпляр класса `Search` посредством вызова `DocumentAPI.createSearch(document)`, затем использовать метод `Search::findText` (см. Рисунок 9).

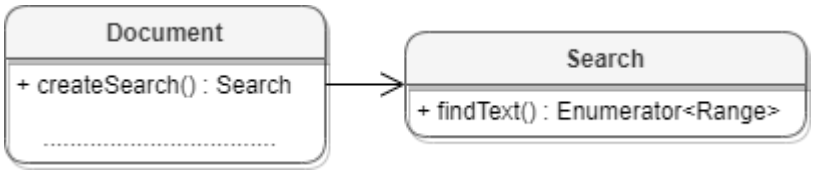


Рисунок 9 – Объектная модель для поиска в документе

Примеры для текстового документа:

```
// 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);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
RangesEnumerator searchResult = search.findText("API", firstBlock.getRange());
while (searchResult.MoveNext())
{
   var range = searchResult.Current;
   Console.WriteLine(range.extractText());
}
```

4.2 Работа с табличным документом

4.2.1 Создание и открытие табличного документа

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

Пример создания табличного документа:

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

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

Примеры загрузки табличного документа:

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

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

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

4.2.2 Сохранение и экспорт табличного документа

Метод Document::saveAs сохраняет документ по указанному пути.

Примеры сохранения табличного документа:

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

```
SaveDocumentSettings saveDocumentSettings = new SaveDocumentSettings();

saveDocumentSettings.documentFormat = DocumentFormat.OXML;
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
saveDocumentSettings.documentType = DocumentType.Workbook;
saveDocumentSettings.documentPassword = "password";
saveDocumentSettings.isTemplate = false;

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

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

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

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

Примеры экспорта табличного документа:

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

4.2.3 Диаграммы

Работа с диаграммами доступна только в табличных документах (см. Рисунок 10).

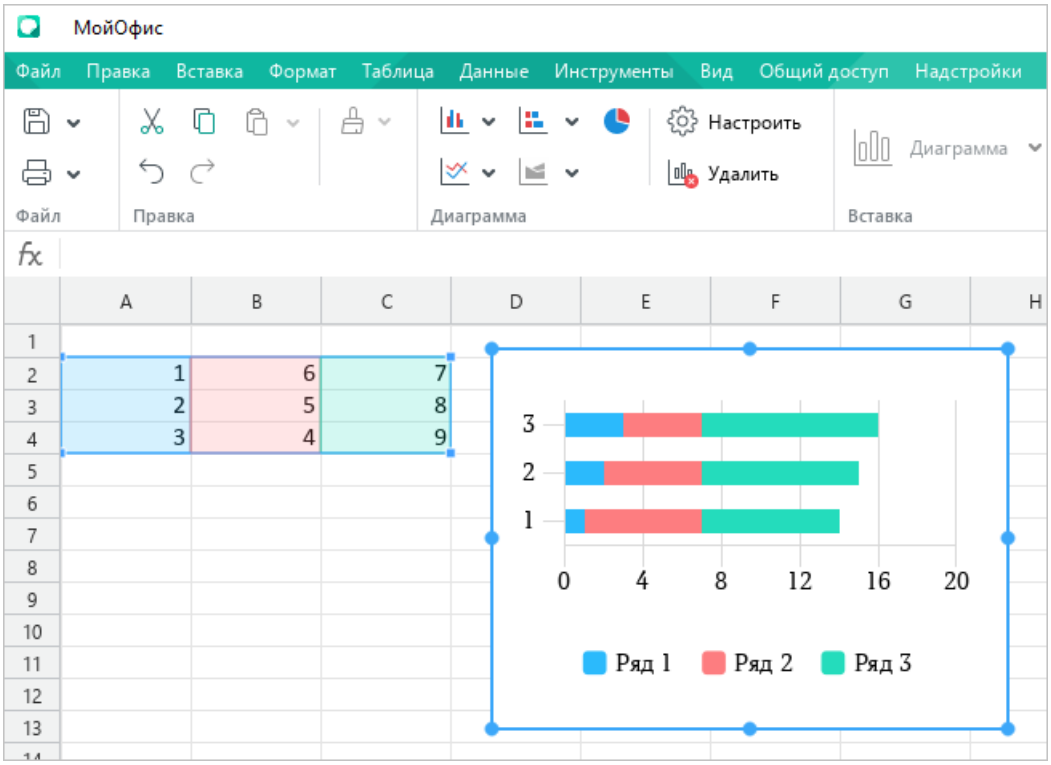


Рисунок 10 – Пример отображения диаграммы в «МойОфис Таблица»

На рисунке 11 изображена объектная модель классов, относящихся к работе с диаграммами.

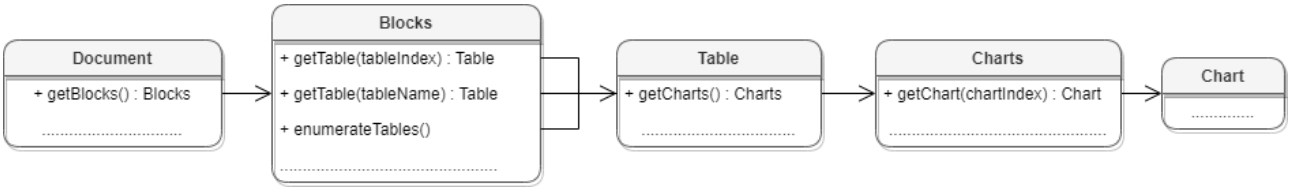


Рисунок 11 – Классы для работы с диаграммами

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

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

Создание и удаление диаграмм в текущей версии не поддерживаются.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

4.2.4 Поиск в табличном документе

Для поиска в табличном документе необходимо создать экземпляр класса `Search` посредством вызова `DocumentAPI.createSearch(document)`, затем использовать метод `Search::findText` (см. Рисунок 12).

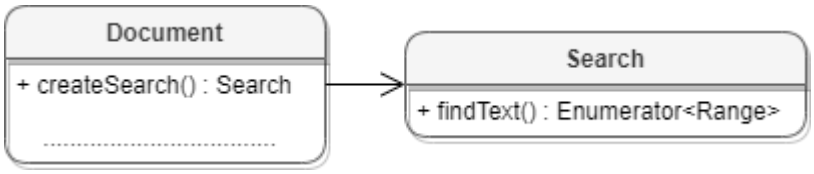


Рисунок 12 – Объектная модель для поиска в документе

Пример для табличного документа:

```
// Поиск в ячейке A1 страницы L1
Table firstSheet = document.getBlocks().getTable("L1");
Cell cell = firstSheet.getCell("A1");

Search search = DocumentAPI.createSearch(document);
RangesEnumerator searchResult = search.findText("Test", cell.getRange());
while (searchResult.MoveNext())
{
    var range = searchResult.Current;
    Console.WriteLine(range.extractText());
}
```

4.2.5 Работа с графическими объектами

На данный момент работа с изображениями в табличном документе не поддерживается.

4.2.6 Работа с листами табличного документа

В табличном документе таблицами являются страницы документа.

Доступ к объектам `Table` осуществляется из `Blocks` (см. Рисунок 13).

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

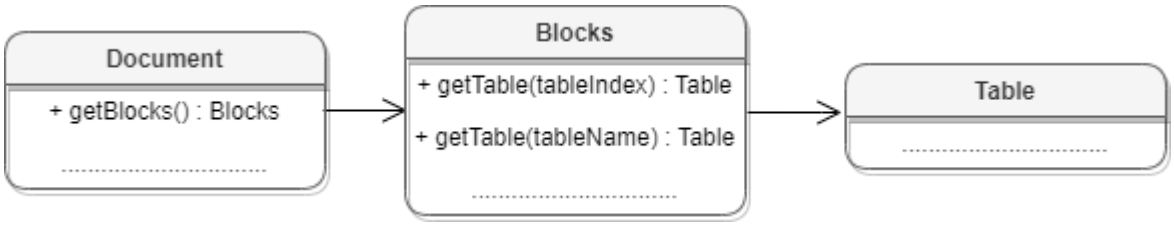


Рисунок 13 – Объектная модель для работы с таблицами

Получение листа табличного документа:

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

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

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

Перечисление страниц табличного документа:

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

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

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Вставка страницы в табличный документ:

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

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

Переименование страницы:

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

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

Скрытие и отображение страниц табличного документа:

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

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

Копирование страницы:

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

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

Удаление страницы:

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

4.2.7 Работа со сводными таблицами

Сводная таблица - инструмент обработки данных, служащий для их обобщения и удобства обработки. Схема взаимодействия объектов, связанных со сводными таблицами, приведена на рисунке 14.

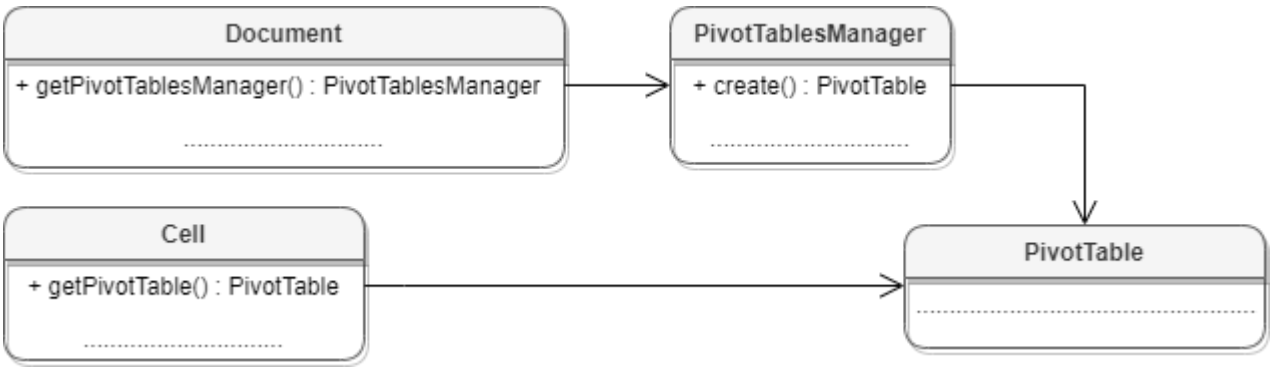


Рисунок 14 – Сводные таблицы

4.2.7.1 Получение диапазона исходных данных сводной таблицы

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

Пример:

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

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

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

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

4.2.7.2 Получение диапазона размещения сводной таблицы

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

4.2.7.3 Получение неподдерживаемых свойств сводной таблицы

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

Пример:

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

4.2.7.4 Получение флагов отображения общих итогов для строк и колонок

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

Пример:

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

4.2.7.5 Получение заголовков сводной таблицы

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

Пример:

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


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

4.2.7.6 Получение и применение фильтра для сводной таблицы

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

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

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

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

4.2.7.7 Получение полей из области фильтров

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

Пример:

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

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

4.2.7.8 Получение полей из области значений

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

4.2.7.9 Получение полей из области строк

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

Пример:

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

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

4.2.7.10 Получение полей из области колонок

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

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

4.2.7.11 Получение настроек отображения сводной таблицы

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

Пример:

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

4.2.7.12 Обновление сводной таблицы

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

4.3 Работа с макрокомандами

Класс `Scripts` предоставляет доступ к списку макрокоманд документа. На рисунке 15 изображена объектная модель классов, относящихся к работе с макрокомандами.

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

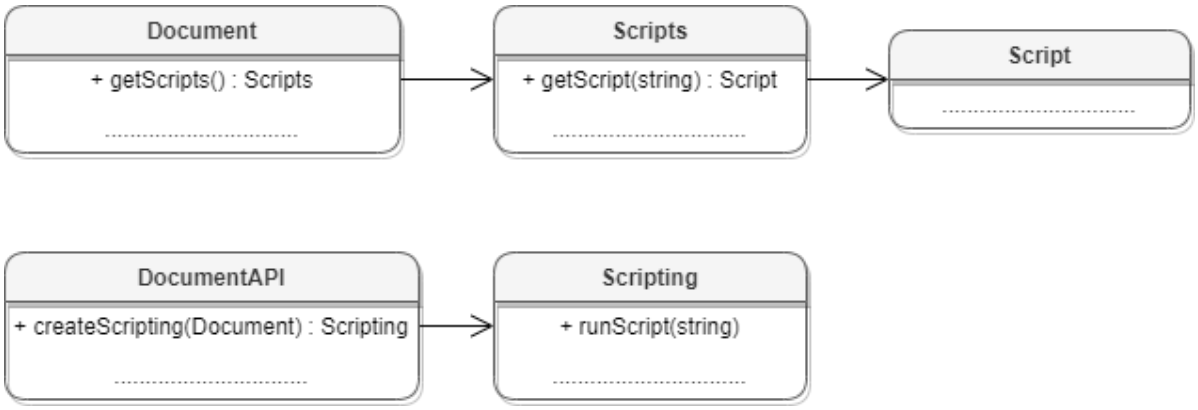


Рисунок 15 – Объектная модель таблиц для работы с макрокомандами

Доступны следующие операции:

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

4.4 Работа с именованными диапазонами

Именованный диапазон – это диапазон ячеек таблицы, которому присвоено имя. Преимуществом именованного диапазона является его информативность: использование имени в текстовом формате выглядит более удобным, чем адреса ячеек. Именованные диапазоны упрощают работу с ячейками, также их удобно использовать при работе с формулами. На данный момент доступна возможность работы с именованными диапазонами, представляющими собой ссылки на диапазоны ячеек. Взаимодействие объектов, связанных с именованными диапазонами, описано на диаграмме (см. Рисунок 16).

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

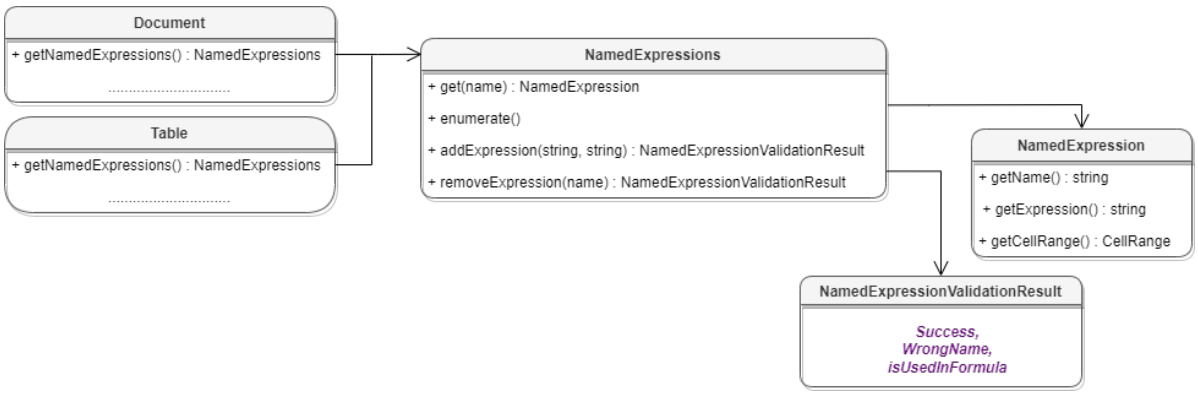


Рисунок 16 – Таблицы для работы с именованными диапазонами

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

4.4.1 Доступ к именованным диапазонам

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

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


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

4.4.2 Получение свойств именованного диапазона

Пример:

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

4.4.3 Получение коллекции именованных диапазонов

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Примеры:

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

// Перебор коллекции именованных выражений
while (enumerator.isValid()) {
    var current = enumerator.Current;
    // Использование полей структуры NamedExpression
    Console.WriteLine(current.getName());
    Console.WriteLine(current.getExpression());
    Console.WriteLine(current.getCellRange());

    enumerator. MoveNext();
}
```

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

4.4.4 Добавление именованного диапазона

Для добавления именованного диапазона используется метод NamedExpressions::addExpression(). В качестве результата операции метод возвращает значение типа NamedExpressionsValidationResult.

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

4.4.5 Удаление именованного диапазона

Для удаления именованного диапазона используется метод NamedExpressions::removeExpression(). В качестве результата операции метод возвращает значение типа NamedExpressionsValidationResult.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

4.4.6 Получение параметров именованного диапазона

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

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

4.5 Работа со строками и столбцами таблиц

4.5.1 Группировка строк и колонок таблицы

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

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

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

4.5.2 Управление видимостью строк / колонок

Метод Table::setColumnsVisible позволяет задавать видимость столбцов, начиная с заданного индекса.

Метод Table:setRowsVisible позволяет задавать видимость строк, начиная с заданного индекса.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

4.6 Работа с ячейками таблиц

4.6.1 Доступ к ячейкам

Доступ к ячейкам таблицы возможен двумя способами (см. Рисунок 17):

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

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

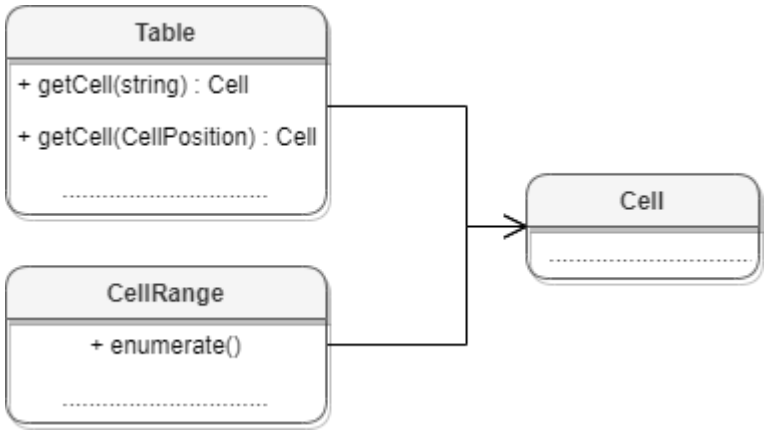


Рисунок 17 – Объектная модель для работы с ячейками таблиц

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

Пример:

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

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

Пример:

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Примеры:

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

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

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

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

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

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

Пример:

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

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

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
cell.setNumber(12);
Console.WriteLine(cell.getFormattedValue());
```

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

Пример:

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

4.6.2 Форматирование ячеек

При работе с ячейками таблиц можно использовать следующие варианты форматирования:

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

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

Пример установки и получения свойств параграфа ячейки:

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

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

позволяет получить экземпляр класса `TextProperties`, представляющий свойства текста. После изменения значения свойств их необходимо применить к тексту ячейки с помощью метода `Range::setTextProperties()`.

Пример настроек текста ячейки:

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

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

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

4.6.3 Форматирование границ ячеек

Для оформления границ ячеек используется класс `Borders` (см. Рисунок 18). Он описывает свойства полей, соответствующих границам и диагоналям ячейки: `Left`, `Right`, `Top`, `Bottom`, `DiagonalDown`, `DiagonalUp`, `InnerHorizontal`, `InnerVertical`. Каждая граница ячейки описывается классом `LineProperties`, который, в свою очередь, обладает свойствами `LineStyle`, `LineEndingProperties`, `Color`, `LineWidth`.

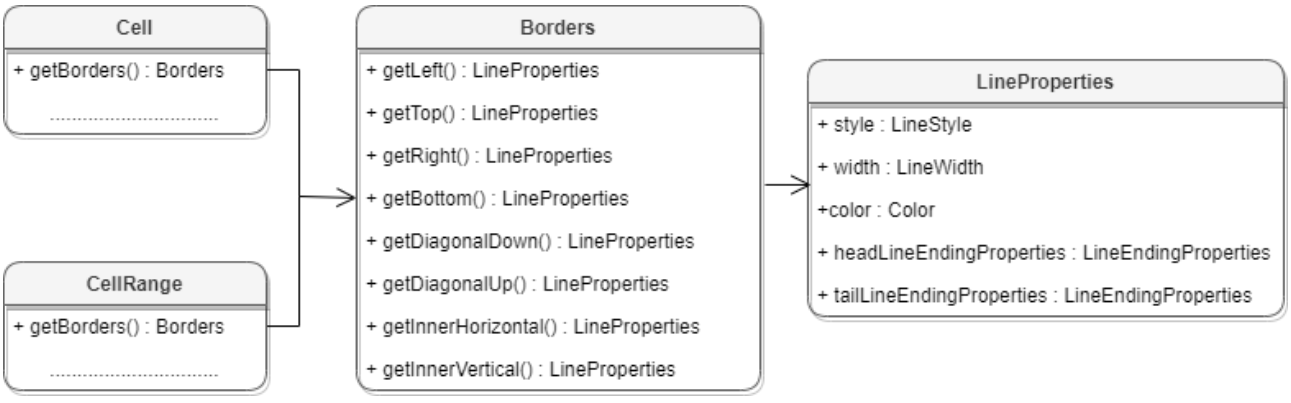


Рисунок 18 – Классы для работы с границами ячеек

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

– получить ячейку `Cell` или область ячеек `CellRange`;

– настроить параметры для рисования линии границы с помощью экземпляра класса `LineProperties`;

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

– настроить свойства линии: левой границы, верхней границы и т.д. с помощью экземпляра класса Borders;
– установить границы ячеек с помощью Cell::setBorders() или CellRange::setBorders().

Пример настройки границ ячеек:

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

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

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

cellRange.setBorders(borders);
```

4.6.4 Объединение и разделение ячеек таблицы

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

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

Пример:

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

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

Для разъединения ячеек используйте метод CellRange::unmerge().

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5 СПРАВОЧНИК КЛАССОВ

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

5.1 Класс Alignment

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

Варианты горизонтального выравнивания текста:

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

Пример:

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

5.2 Класс AnchoredPosition

Класс `AnchoredPosition` представляет позицию объекта на странице текстового документа. Используется в методах `Frame::getPosition`, `Frame::setPosition`. Описание полей представлено в таблице 2.

Таблица 2 – Описание полей класса `AnchoredPosition`

Поле	Описание
`AnchoredPosition.textPosition`	Позиция в текстовом документе `TexAnchoredPosition`

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
InlineObjectsEnumerator enumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in enumerator)
{
    var frame = inlineObject.getFrame();
    AnchoredPosition position = frame.getPosition();
    Console.WriteLine(position.textPosition.horizontal.alignment);
}
```

5.3 Класс Application

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

Пример:

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

5.3.1 Метод Application::createDocument

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

Существуют следующие варианты реализации метода:

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

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

5.3.2 Метод Application::loadDocument

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

Существуют следующие варианты реализации метода:

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

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

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

5.3.3 Метод Application::getMessenger

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

Пример:

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

5.4 Класс Block

Класс `Block` является базовой для всех блоков документа. От нее наследуются классы `Paragraph`, `Table`, `Shape`, `Field` (см. Рисунок 19).

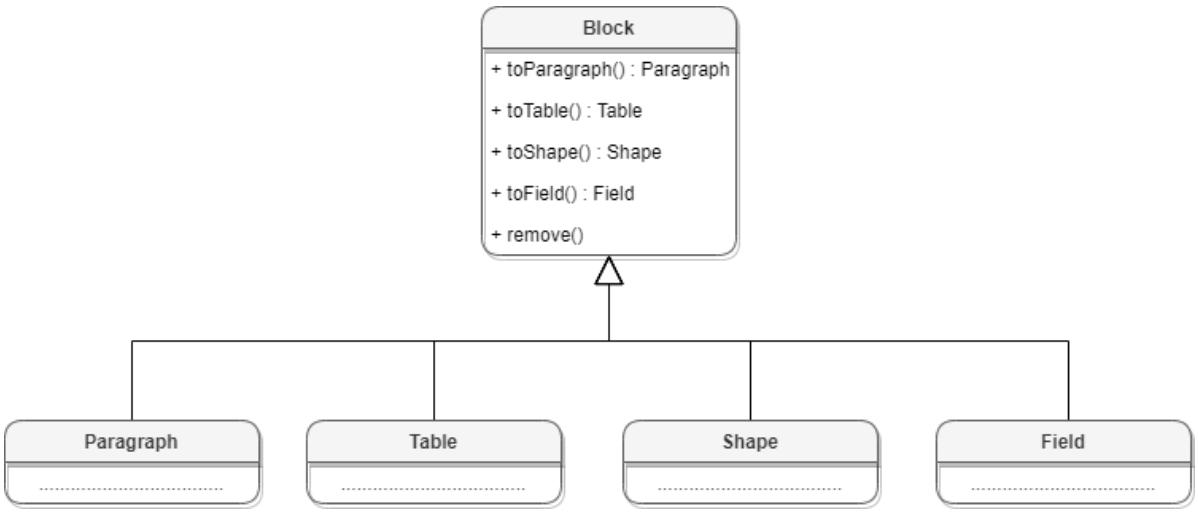


Рисунок 19 – Объектная модель класса `Block`

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.4.1 Методы toParagraph, toTable, toShape, toField

Преобразует объект `Block` в объект соответствующего типа.

Пример:

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

5.4.2 Метод Block::getRange

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

Пример:

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

5.4.3 Метод Block::remove

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

Пример:

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

5.4.4 Метод Block::getSection

Метод возвращает раздел `Section`, содержащий блок.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Section section = block.getSection();
    Console.WriteLine(section.getRange().extractText());
}
```

5.5 Класс Blocks

Класс `Blocks` обеспечивает доступ к блокам `Block` документа или диапазона документа (см. Рисунок 20). Объект класса `Blocks` может быть получен вызовом метода `Document::getBlocks` или `HeaderFooter::getBlocks`.

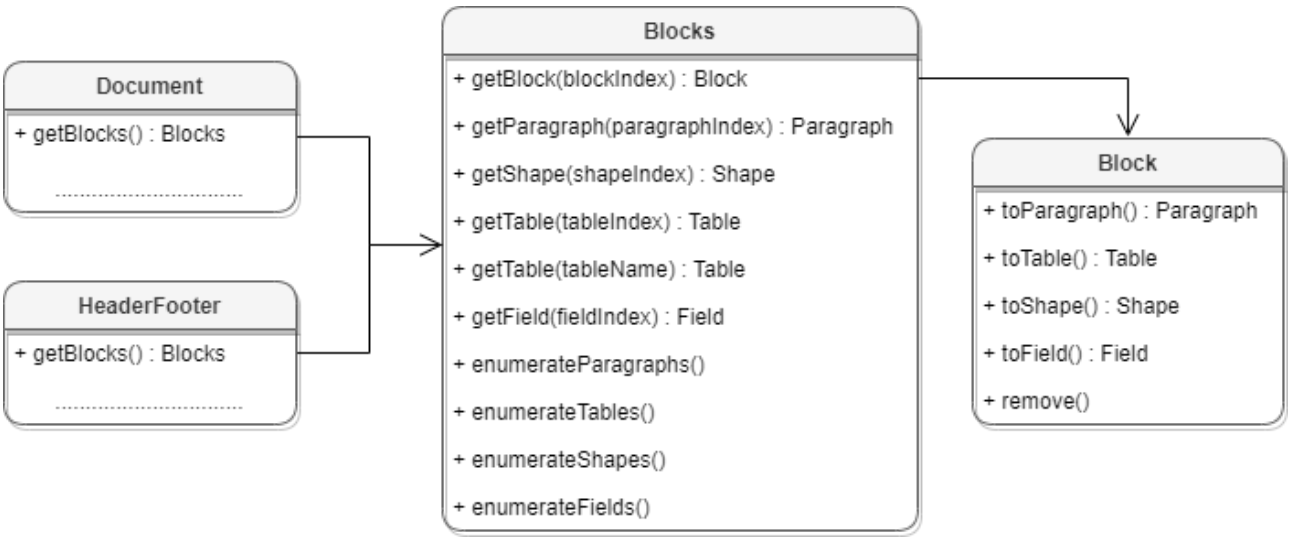


Рисунок 20 – Объектная модель класса `Blocks`

5.5.1 Метод Blocks::getBlock

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.5.2 Метод Blocks::getParagraph

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

Пример:

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

5.5.3 Метод Blocks::getTable

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

Варианты реализации метода:

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

Примеры:

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

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

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

5.5.4 Метод Blocks::getShape

Возвращает фигуру `Shape` по заданному индексу.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.5.5 Метод Blocks::getField

Возвращает объект типа `Field` по заданному индексу.

Пример:

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

5.5.6 Метод Blocks::GetEnumerator

Позволяет реализовать перечисление объектов `Block`.

Пример:

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

5.5.7 Метод Blocks::getParagraphsEnumerator

Позволяет реализовать перечисление абзацев `Paragraph`.

Пример:

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

5.5.8 Метод Blocks::getTablesEnumerator

Позволяет перечислить объекты типа `Table`.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

5.5.9 Метод Blocks::getShapesEnumerator

Позволяет перечислить объекты типа Shape.

Пример:

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

5.5.10 Метод Blocks::getFieldsEnumerator

Позволяет перечислить объекты типа Field.

Пример:

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

5.6 Класс Bookmarks

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

5.6.1 Метод Bookmarks::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());
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.6.2 Метод Bookmarks::removeBookmark

Удаляет закладку по ее названию.

Пример:

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

5.7 Класс Borders

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

Таблица 3 – Описание методов класса `Borders`

Метод	Описание
`Borders setLeft(LineProperties lp)`	Установка левой границы ячейки
`Borders setRight(LineProperties lp)`	Установка правой границы ячейки
`Borders setTop(LineProperties lp)`	Установка верхней границы ячейки
`Borders setBottom(LineProperties lp)`	Установка нижней границы ячейки
`Borders setDiagonalDown(LineProperties lp)`	Установка диагональной линии 
`Borders setDiagonalUp(LineProperties lp)`	Установка диагональной линии 
`Borders setOuter(LineProperties lp)`	Установка внешних границ ячейки
`Borders setDiagonals(LineProperties lp)`	Установка обеих типов диагональных линий одновременно
`Borders setInnerHorizontal(LineProperties lp)`	Установка внутренних горизонтальных границ ячейки
`Borders setInnerVertical(LineProperties lp)`	Установка внутренних вертикальных границ ячейки
`Borders setInner(LineProperties lp)`	Установка внутренних границ ячейки
`Borders setAll(LineProperties lp)`	Установка всех границ ячейки
`Borders getLeft(LineProperties lp)`	Получение левой границы ячейки
`Borders getRight(LineProperties lp)`	Получение правой границы ячейки
`Borders getTop(LineProperties lp)`	Получение верхней границы ячейки
`Borders getBottom(LineProperties lp)`	Получение нижней границы ячейки
`Borders getDiagonalDown(LineProperties lp)`	Получение диагональной линии
`Borders getDiagonalUp(LineProperties lp)`	Получение диагональной линии

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Метод	Описание
`Borders getInnerHorizontal(LineProperties lp)`	Получение внутренних горизонтальных границ ячейки
`Borders getInnerVertical(LineProperties lp)`	Получение внутренних вертикальных границ ячейки

Пример:

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

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

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

5.8 Класс Cell

Класс `Cell` предоставляет доступ к ячейке в таблице текстового документа или на листе табличного документа (см. Рисунок 21).

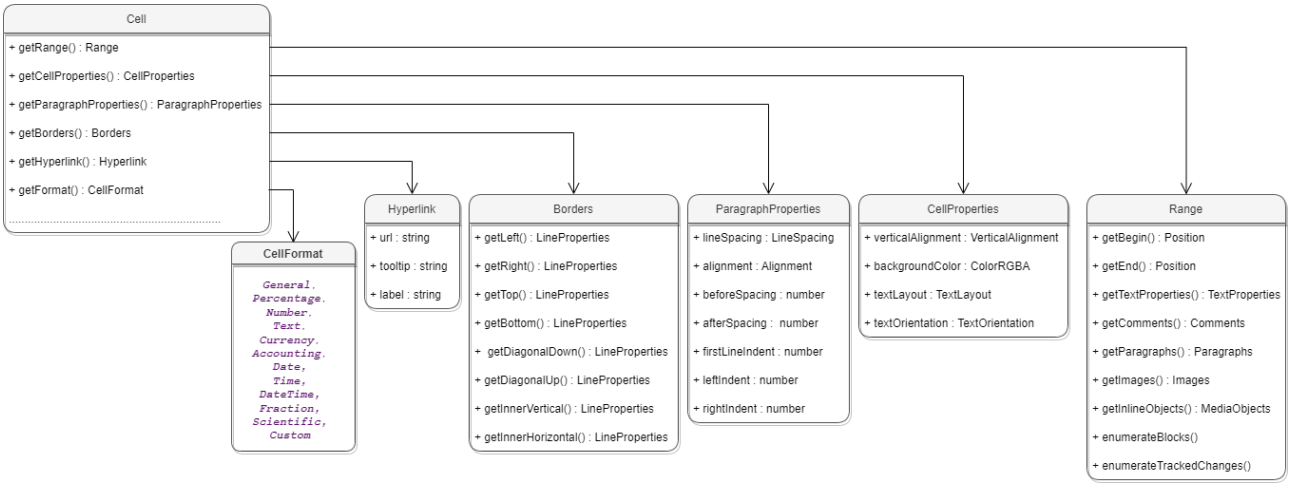


Рисунок 21 – Объектная модель ячейки таблиц

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.8.1 Метод Cell::getRange

Метод возвращает объект Range для управления содержимым ячейки.

5.8.2 Метод Cell::setBorders

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

5.8.3 Метод Cell::setFormula

Метод позволяет вставить формулу в ячейку табличного документа.

Пример:

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

5.8.4 Метод Cell::setFormat

Метод устанавливает формат ячейки. Существуют несколько вариантов использования метода.

Варианты вызова метода:

```
setFormat(cellFormat)
```

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

```
setFormat(numberCellFormatting)
```

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

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

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

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
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
// 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 : Дата / Время
DateTimeCellFormatting dateTimeCellFormatting = new DateTimeCellFormatting();
dateTimeCellFormatting.dateListID = DatePatterns.FullDate;
dateTimeCellFormatting.timeListID = TimePatterns.ShortTime;
cell.setFormat(dateTimeCellFormatting, CellFormat.DateTime);
Console.WriteLine(cell.getFormat()); // 8
Console.WriteLine(cell.getRange().extractText()); // понедельник, 1 января 1900
г. 7 : 12
```

5.8.5 Метод Cell::getFormat

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

Пример:

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

5.8.6 Метод Cell::getFormattedValue

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.8.7 Метод Cell::setFormattedValue

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

Список поддерживаемых форматов см. в разделе `CellFormat`.

Пример:

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

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

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

5.8.8 Метод Cell::unmerge

Разъединяет несколько ячеек, которые были объединены ранее.

Пример:

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

5.8.9 Метод Cell::setContent

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.8.10 Метод Cell::getBorders

Позволяет получить границы ячейки.

Пример:

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

5.8.11 Метод Cell::getRawValue

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

Пример:

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

5.8.12 Метод Cell::getCustomFormat

Возвращает строку формата ячейки.

Пример:

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

5.8.13 Метод Cell::setCustomFormat

Устанавливает формат ячейки.

Пример:

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

5.8.14 Метод Cell::setBool

Устанавливает для ячейки значение логического типа.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.8.15 Метод Cell::setNumber

Устанавливает для ячейки значение числового типа.

Пример:

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

5.8.16 Метод Cell::setText

Устанавливает для ячейки значение строкового типа.

Пример:

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

5.8.17 Метод Cell::getFormulaAsString

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

Пример:

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

5.8.18 Метод Cell::getCellProperties

Позволяет получить свойства `CellProperties` ячейки.

Пример:

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

5.8.19 Метод Cell::setCellProperties

Позволяет установить свойства ячейки `CellProperties`.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
cellProperties.verticalAlignment = VerticalAlignment.Center;
cell.setCellProperties(cellProperties);
```

5.8.20 Метод Cell::getParagraphProperties

Возвращает свойства абзаца ParagraphProperties, находящегося в ячейке.

Пример:

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

5.8.21 Метод Cell::setParagraphProperties

Устанавливает свойства абзаца ParagraphProperties, находящегося в ячейке.

Пример:

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

5.8.22 Метод Cell::getPivotTable

Возвращает сводную таблицу PivotTable, относящуюся к ячейке.

Пример:

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

5.9 Класс CellFormat

По умолчанию при создании документа всем ячейкам присваивается формат «Общий». Полный список форматов представлен в таблице 4.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 4 – Поддерживаемые форматы ячеек таблицы

Наименование константы	Описание
`CellFormat.General`	Формат ячейки «Общий». В этом формате в ячейке отображаются первые 9 символов числа, остальные доступны для просмотра в строке формул. Для дробных чисел в формате «Общий» незначащие нули в дробной части не отображаются. Числа, состоящие более чем из 12 символов, переводятся в экспоненциальную форму после завершения ввода в ячейку.
`CellFormat.Percentage`	Формат ячейки «Процентный». Этот формат используется для представления чисел как процентов. При применении формата «Процентный» введенное число умножается на 100 и обозначается знаком «%».
`CellFormat.Number`	Формат ячейки «Числовой». Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя.
`CellFormat.Text`	Формат ячейки «Текстовый».
`CellFormat.Currency`	Формат ячейки «Денежный». Этот формат используется для представления чисел со знаком или кодом валюты.
`CellFormat.Accounting`	Формат ячейки «Финансовый». Этот формат применяется для чисел, используемых в бухгалтерских документах. В формате «Финансовый» введенное число автоматически дополняется названием валюты, которая соответствует настройкам системы компьютера. Отрицательные числа в формате «Финансовый» заключаются в круглые скобки в ячейке, а в строке формул остаются в том виде, в котором они были введены.
`CellFormat.Date`	Формат ячейки «Дата». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ.
`CellFormat.Time`	Формат ячейки «Время». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

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

Примеры использования:

```
Table firstSheet = document.getBlocks().getTable("Лист1");
Cell cell = firstSheet.getCell("B1");
cell.setFormat(CellFormat.General);
cell = firstSheet.getCell("B2");
cell.setFormat(CellFormat.Percentage);
cell = firstSheet.getCell("B3");
cell.setFormat(CellFormat.Number);
```

Результат:

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.10 Класс CellPosition

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

Примеры:

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

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

5.10.1 Поле CellPosition::column

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

Примеры:

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

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

5.10.2 Поле CellPosition::row

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

Примеры:

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

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

5.10.3 Метод CellPosition::toString

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

5.11 Класс CellProperties

Класс `CellProperties` предназначен для форматирования содержимого в ячейках таблицы. Описание полей представлено в таблице 5.

Для задания свойств ячейки используется метод `Cell::setCellProperties()`. Для получения свойств ячейки используется метод `Cell::getCellProperties()`. Иерархия классов и полей для работы с `CellProperties` отображена на рисунке 22.

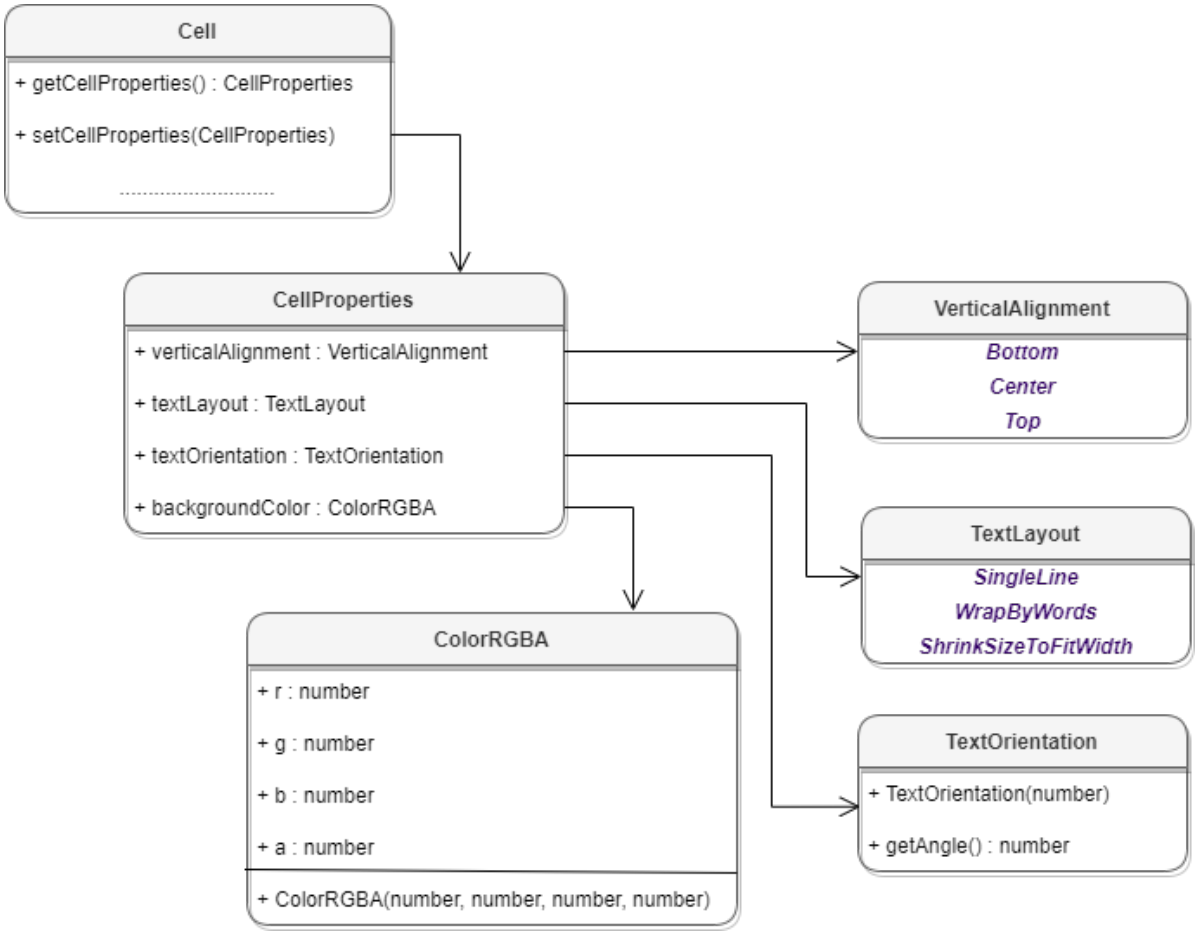


Рисунок 22 – Объектная модель для работы со свойствами ячеек таблицы

Таблица 5 – Описание полей класса `CellProperties`

Поле	Тип	Значение
`CellProperties.verticalAlignment`	`VerticalAlignment`	Вертикальное выравнивание в ячейке

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Поле	Тип	Значение
CellProperties.textLayout	TextLayout	Способ отображения значения ячейки
CellProperties.backgroundColor	ColorRGBA	Цвет фона ячейки
CellProperties.textOrientation	TextOrientation	Ориентация текста в ячейке (угол поворота)

Пример:

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

CellProperties cellProps = cell.getCellProperties();
cellProps.verticalAlignment = VerticalAlignment.Center;
cellProps.textLayout = TextLayout.ShrinkSizeToFitWidth;
cellProps.backgroundColor = new ColorRGBA(255, 255, 0, 255);
cellProps.textOrientation = new TextOrientation(45);

cell.setCellProperties(cellProps);
```

5.12 Класс CellRange

Класс CellRange описывает диапазон ячеек таблицы.

Пример:

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

5.12.1 Метод CellRange::getTable

Возвращает таблицу Table, содержащую текущий диапазон.

5.12.2 Метод CellRange::getEnumerator

Метод возвращает коллекцию ячеек в диапазоне.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(cell.getFormattedValue());
}
```

5.12.3 Метод CellRange:getBeginRow

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

Пример:

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

5.12.4 Метод CellRange:getBeginColumn

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

Пример:

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

5.12.5 Метод CellRange:getLastRow

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

Пример:

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

5.12.6 Метод CellRange:getLastColumn

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.12.7 Метод CellRange:setBorders

Метод предназначен для установки границ диапазона ячеек. Отдельные границы устанавливаются с помощью методов класса Borders.

Пример:

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

LineProperties lineProperties = new LineProperties();
lineProperties.style = LineStyle.Dash;
lineProperties.width = 1.5f;
lineProperties.color = new Color(new ColorRGBA(255, 0, 0, 255));

Borders newBorders = new Borders();
newBorders.setLeft(lineProperties);
newBorders.setRight(lineProperties);
newBorders.setTop(lineProperties);
newBorders.setBottom(lineProperties);

cell.setBorders(newBorders);
```

5.12.8 Метод CellRange:insertCurrentDateTime

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

Пример:

```
Table sheet = document.getBlocks().getTable(0);
CellRange cellRange = sheet.getCellRange("A1");
cellRange.insertCurrentDateTime(DateTimeFormat.DateTime);
```

5.12.9 Метод CellRange:getCellProperties

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
CellProperties cellProperties = cellRange.getCellProperties();
Console.WriteLine(cellProperties.backgroundColor.r);
```

5.12.10 Метод CellRange:setCellProperties

Метод предназначен для установки свойств `CellProperties` ячеек диапазона.

Пример:

```
Table firstSheet = document.getBlocks().getTable(0);
CellRange cellRange = firstSheet.getCellRange("B3:C4");
CellProperties cellProperties = new CellProperties();
cellProperties.backgroundColor = new ColorRGBA(55, 146, 179, 200);
cellRange.setCellProperties(cellProperties);
```

5.12.11 Метод CellRange:merge

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

Пример:

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

5.12.12 Метод CellRange:unmerge

Метод разъединяет ранее объединенные ячейки.

Пример:

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

5.12.13 Метод CellRange::autoFill

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

Метод `autoFill` автоматически интерполирует исходные точки и находит алгоритм аппроксимации, который используется для экстраполяции значений в диапазоне ячеек

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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));
```

5.13 Класс CellRangePosition

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

Описание полей класса `CellRangePosition` представлено в таблице 6.

Таблица 6 – Поля класса `CellRangePosition`

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

Примеры:

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

```
Table table = document.getBlocks().getTable(0);
Charts charts = table.getCharts();
ChartRangeInfo rangeInfo = charts.getChart(0).getRange(0);
TableRangeInfo cellRangePosition = rangeInfo.tableRangeInfo;
CellRangePosition tableRange = cellRangePosition.tableRange;
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

5.13.1 Метод CellRangePosition:toString

Возвращает информацию о диапазоне ячеек в виде строкового значения формата `(topLeft: <value>, bottomRight: <value>)`.

Пример:

```
Table table = document.getBlocks().getTable(0);
Charts charts = table.getCharts();
ChartRangeInfo rangeInfo = charts.getChart(0).getRange(0);
TableRangeInfo cellRangePosition = rangeInfo.tableRangeInfo;
CellRangePosition tableRange = cellRangePosition.tableRange;
Console.WriteLine(tableRange.toString()); // [topLeft: (row: 0, column: 0),
bottomRight: (row: 5, column: 5)];
```

5.14 Класс Charts

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

Пример получения списка диаграмм:

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

5.14.1 Метод Charts::getChartsCount

Метод возвращает общее количество диаграмм в табличном документе.

Пример:

```
Charts charts = sheetDocumentPage.getCharts();
Console.WriteLine(charts.getChartsCount());
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.14.2 Метод Charts::getChart

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

Пример:

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

5.14.3 Метод Charts::getChartIndexByDrawingIndex

Метод возвращает индекс диаграммы по индексу отрисовки `drawingIndex`.

Пример:

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

5.15 Класс Chart

Класс `Chart` представляет диаграмму в табличном документе и описывает все ее элементы (заголовок, легенда, тип, данные, диапазон и т.д). Объектная модель класса `Chart` приведена на Рис. 23.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

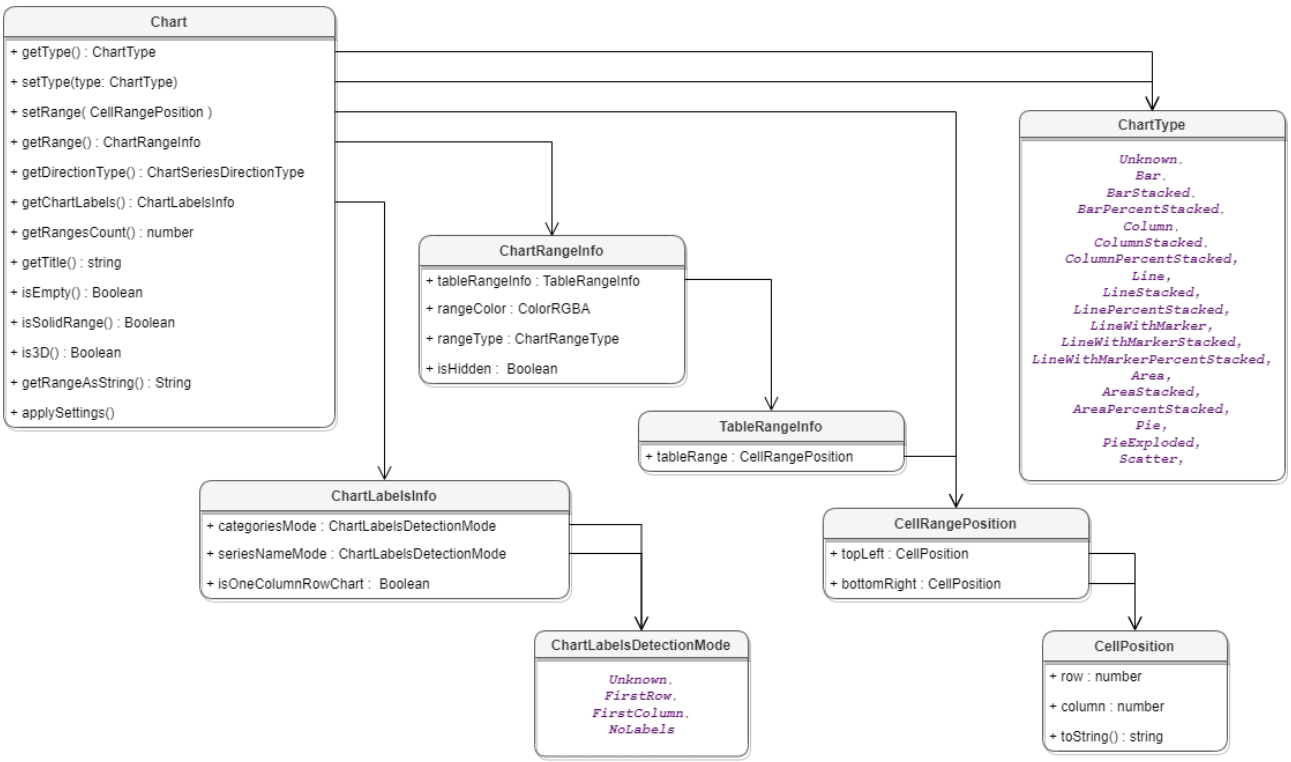


Рисунок 23 – Объектная модель класса `Chart`

5.15.1 Метод Chart::getType

Метод возвращает тип диаграммы `ChartType`.

Пример:

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

5.15.2 Метод Chart::setType

Метод устанавливает тип диаграммы `ChartType`. В качестве параметра передается новый тип диаграммы.

Пример:

```
Charts charts = sheetDocumentPage.getCharts();
Chart chart = charts.getChart(0);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
if (chart != null) {
    chart.setType(ChartType.ColumnStacked);
    Console.WriteLine(chart.getType());
}
```

5.15.3 Метод Chart::getRangesCount

Метод возвращает количество серий диаграммы.

Пример:

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

5.15.4 Метод Chart::getRange

Метод возвращает диапазон ячеек ChartRangeInfo, содержащий исходные данные диаграммы. Параметр метода – индекс диапазона.

Пример:

```
Charts charts = sheetDocumentPage.getCharts();
Chart chart = charts.getChart(0);
if (chart != null) {
    NCT.MyOfficeSDK.ChartRangeInfo chartRangeInfo = chart.getRange(0);
    if (chartRangeInfo != null) {
        Console.WriteLine(chartRangeInfo.rangeType);
    }
}
```

5.15.5 Метод Chart::getTitle

Метод возвращает заголовок диаграммы.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.15.6 Метод Chart::setRange

Метод задает диапазон CellRangePosition ячеек с исходными данными для диаграммы.

Пример:

```
Chart chart = charts.getChart(0);
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
chart.setRange(cellRangePosition);
Console.WriteLine(chart.getRangeAsString());
```

5.15.7 Метод Chart::setRect

Метод задает область расположения диаграммы.

Внимание ! Метод устаревший (deprecated), оставлен для обратной совместимости и не рекомендован к использованию.

Пример:

```
Chart chart = charts.getChart(0);
chart.setRect(new RectU(0, 0, 20, 20));
```

5.15.8 Метод Chart::isEmpty

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

Пример:

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

5.15.9 Метод Chart::isSolidRange

Метод возвращает `true`, если диапазон исходных данных диаграммы может быть выделен одним прямоугольником и не имеет промежутков.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.15.10 Метод Chart::is3D

Метод возвращает `true`, если диаграмма трехмерная.

Пример:

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

5.15.11 Метод Chart::getDirectionType

Метод возвращает направление `ChartSeriesDirectionType` серий диаграммы.

Пример:

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

5.15.12 Метод Chart::getChartLabels

Метод возвращает коллекцию меток диаграммы типа `ChartLabelsInfo`.

Пример:

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

5.15.13 Метод Chart::getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

Пример:

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

5.15.14 Метод Chart::applySettings

Метод позволяет обновить параметры текущей выбранной диаграммы.

Вызов:

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

Параметры:

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

– `directionType` – направление серий `ChartSeriesDirectionType`;

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
CellRange cellRange = sheetDocumentPage.getCellRange("B3:C4");
ChartLabelsInfo chartLabelsInfo = new
ChartLabelsInfo(ChartLabelsDetectionMode.FirstColumn,
ChartLabelsDetectionMode.FirstRow, false);
chart.applySettings(cellRange, null, "Title", chartLabelsInfo);
```

5.16 Класс ChartLabelsDetectionMode

Класс описывает режимы автоматического определения меток диаграмм.

Поля класса соответвуют следующим режимам автоматического определения меток диаграмм:

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

Пример:

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

5.17 Класс ChartLabelsInfo

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

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

Параметры конструктора:

- `categoriesMode` - режим автоматического определения меток для категорий, тип `ChartLabelsDetectionMode`;
- `seriesNameMode` - режим автоматического определения меток для серий, тип `ChartLabelsDetectionMode`;
- `oneColumnRow` - передается `true`, если диапазон диаграммы содержит только одну строку или одну колонку.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Описание полей класса ChartLabelsInfo представлено в таблице 7.

Таблица 7 – Описание полей класса ChartLabelsInfo

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

Примеры:

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

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

5.18 Класс ChartRangeInfo

Класс ChartRangeInfo описывает серию диаграммы. Инициализируется конструктором:

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

Параметры конструктора:

– tableRangeInfo - диапазон ячеек, тип TableRangeInfo;

– color – цвет серии диаграммы, тип ColorRGBA;

– hidden – видимость серии, тип Boolean;

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

Описание полей класса представлено в таблице 8.

Таблица 8 – Описание полей класса ChartRangeInfo

Поле	Описание	Тип
tableRangeInfo	Исходный диапазон ячеек для серии	TableRangeInfo

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Поле	Описание	Тип
rangeColor	Цвет для отрисовки серии	ColorRGBA
isHidden	Задает видимость серии диаграммы	Boolean
rangeType	Тип диапазона диаграммы	ChartRangeType

Пример:

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

5.19 Класс ChartRangeType

Класс описывает тип диапазона исходных данных диаграммы.

Возможные значения:

- ChartRangeType.Series – серии;
- ChartRangeType.SeriesName – имена серий;
- ChartRangeType.Categories – области;
- ChartRangeType.DataPoint – разметка данных.

Пример:

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

5.20 Класс ChartSeriesDirectionType

Класс описывает направление серий диаграмм.

Возможные значения:

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.21 Класс ChartType

Перечисление `ChartType` описывает все поддерживаемые типы диаграмм.

Поля класса соответвуют следующим типам диаграмм:

- `ChartType.Unknown` – неопределенный тип;
- `ChartType.Bar` – линейчатая диаграмма с группировкой;
- `ChartType.BarStacked` – линейчатая диаграмма с накоплением;
- `ChartType.BarPercentStacked` – линейчатая нормированная диаграмма с накоплением;
- `ChartType.Column` – гистограмма с группировкой;
- `ChartType.ColumnStacked` – гистограмма с накоплением;
- `ChartType.ColumnPercentStacked` – нормированная гистограмма с накоплением;
- `ChartType.Line` – стандартный график;
- `ChartType.LineStacked` – график с накоплением;
- `ChartType.LinePercentStacked` – нормированный график с накоплением;
- `ChartType.LineWithMarker` – стандартный график с маркерами;
- `ChartType.LineWithMarkerStacked` – график с накоплением и маркерами;
- `ChartType.LineWithMarkerPercentStacked` – нормированный график с накоплением и маркерами;
- `ChartType.Area` – стандартная диаграмма с областями;
- `ChartType.AreaStacked` – диаграмма с областями с накоплением;
- `ChartType.AreaPercentStacked` – нормированная диаграмма с областями с накоплением;
- `ChartType.Pie` – круговая диаграмма;
- `ChartType.PieExploded` – круговая диаграмма с отделенными секторами;
- `ChartType.Scatter` – диаграмма рассеяния.

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
if (sheetDocumentPage != null) {
    Charts charts = sheetDocumentPage.getCharts();
    Chart chart = charts.getChart(0);
    ChartType chartType = chart.getType();
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(chartType);
}
```

5.22 Класс Color

Класс `Color` представляет либо цветовой объект RGBA, либо заданные цвета идентификатора темы. В качестве параметров конструктора используются объекты `ColorRGBA`, `ThemeColorID`.

Пример:

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

5.22.1 Метод Color::getRGBAColor

Метод возвращает цвет `ColorRGBA`.

Пример:

```
Color color = new Color(new ColorRGBA(255, 0, 0, 255));
ColorRGBA rgbaColor = color.getRGBAColor();
if (rgbaColor != null) {
    Console.WriteLine(rgbaColor.r);
}
```

5.22.2 Метод Color::getThemeColorID

Метод возвращает цвет идентификатора темы `ThemeColorID`.

Пример:

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

5.23 Класс ColorRGBA

Класс `ColorRGBA` предназначен для задания цвета текста, линии, фона и т.д. Используется четырехканальный формат, содержащий данные для красного (`r`), зеленого (`g`), голубого (`b`) цветов и альфа-канала (`a`).

Для создания нового объекта используется один из конструкторов:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Описание полей класса `ColorRGBA` представлено в таблице 9.

Таблица 9 – Описание полей класса `ColorRGBA`

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

Примеры использования:

```
ColorRGBA rgba = new ColorRGBA();
rgba.r = 0;
rgba.g = 0;
rgba.b = 255;
rgba.a = 200;
// r: 0, g: 0, b: 255, a: 200
Console.WriteLine("r:" + rgba.r + ", g:" + rgba.g + ", b:" + rgba.b + ", a:" +
rgba.a);
```

```
rgba = new ColorRGBA(55, 146, 179, 200);
// r: 55, g: 146, b: 179, a: 200
Console.WriteLine("r:" + rgba.r + ", g:" + rgba.g + ", b:" + rgba.b + ", a:" +
rgba.a);
```

```
LineProperties lineProps = new LineProperties();
lineProps.color = new Color(rgba);
```

5.24 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`.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.24.1 Метод Comment::getRange

Метод возвращает диапазон документа Range, которому соотвествует комментарий.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Range commentRange = comment.getRange();
        Console.WriteLine(commentRange.extractText());
    }
}
Comments comments = document.getRange().getComments();
```

5.24.2 Метод Comment::getText

Метод возвращает текст комментария.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Console.WriteLine(comment.getText());
    }
}
```

5.24.3 Метод Comment::getInfo

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

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Console.WriteLine(comment.getInfo().author);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    }
}
```

5.24.4 Метод Comment::isResolved

Метод возвращает значение `true`, если комментарий принят.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Console.WriteLine(comment.isResolved());
    }
}
```

5.24.5 Метод Comment::getReplies

Метод предоставляет доступ к ответам на комментарии. Ответы представляются классом `Comments` так же, как и сами комментарии документа.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
{
    foreach (var comment in commentsEnumerator)
    {
        Comments replies = comment.getReplies();
        CommentsEnumerator repliesEnumerator = replies.GetEnumerator();
        foreach (var reply in repliesEnumerator)
        {
            Console.WriteLine(reply.isResolved());
        }
    }
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.25 Класс Comments

Класс `Comments` содержит коллекцию комментариев диапазона (см. Рисунок 24).

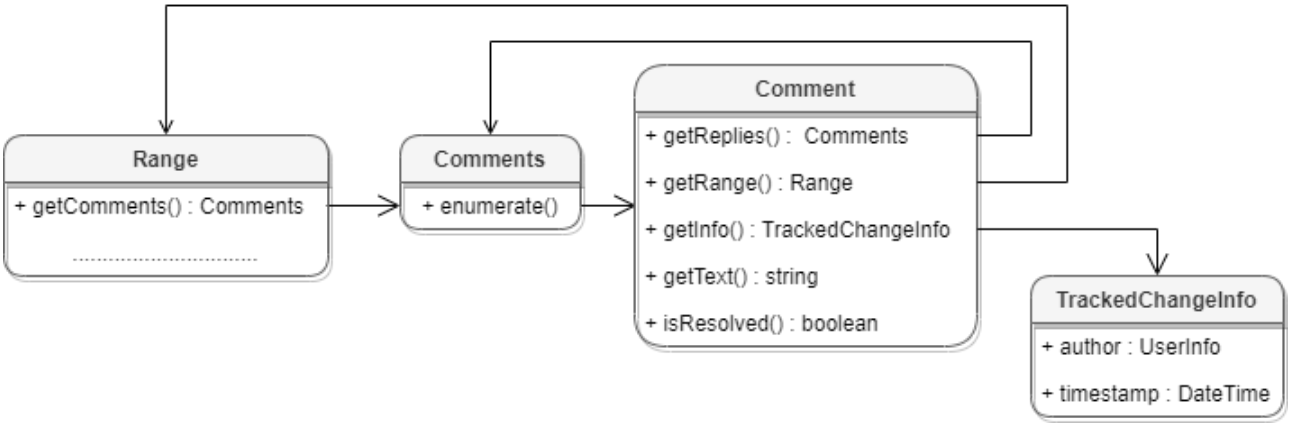


Рисунок 24 – Объектная модель классов для работы с комментариями

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

Пример:

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

5.25.1 Метод Comments::GetEnumerator

Метод используется для перечисления комментариев диапазона.

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
foreach (var comment in commentsEnumerator)
{
    Console.WriteLine(comment.getInfo().author);
}
```

5.26 Класс Connection

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
Connection connection = messenger.subscribe(messageHandler);
.....................................................
connection.unsubscribe();
```

5.27 Класс CurrencySignPlacement

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

Описание полей таблицы `CurrencySignPlacement` представлено в таблице 10.

Таблица 10 – Описание вариантов расположения знаков валюты

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

5.28 Класс DateTime

Класс `DateTime` предоставляет дату и время с точностью до секунды. Используется для поля `TrackedChangeInfo.timeStamp`. Описание полей класса `DateTime` представлено в таблице 11.

Таблица 11 – Описание полей класса `DateTime`

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

5.29 Класс DateTimeCellFormatting

Класс содержит параметры для формата ячеек таблицы типа Дата и Время, используется в качестве аргумента метода `Cell::setFormat()`. Описание полей класса `DateTimeCellFormatting` представлено в таблице 12.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 12 – Описание полей класса DateTimeCellFormatting

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

Пример:

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

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

cell.setFormat(cellFormat, CellFormat.DateTime);
Console.WriteLine(cell.getFormattedValue());
```

5.30 Класс DatePatterns

Форматы даты представлены в таблице 13. Пример использования см. в главе DateTimeCellFormatting.

Таблица 13 – Форматы даты

Наименование константы	Описание
DatePatterns.DayMonthTextLongYearLong	'mmmm dd, yyyy' для языка en_US
DatePatterns.FullDate	'день недели, mmmm dd, yyyy' для языка en_US
DatePatterns.DayMonthNumberLongYearLong	'mm/dd/yyyy' для языка en_US
DatePatterns.DayMonthNumberLongYearShort	'm/dd/yy' для языка en_US
DatePatterns.DayMonthNumberShortYearShort	'dd-mmm' для языка en_US
DatePatterns.DayMonthTextShort	'mmm-yy' для языка en_US
DatePatterns.DayMonthTextShortYearShort	'mmm dd, yy' для языка en_US
DatePatterns.DayMonthYearLongNonLocalizableHyphen	Нелокализуемый шаблон 'dd-mm-yyyy'
DatePatterns.DayMonthYearLongNonLocalizableSlash	Нелокализуемый шаблон 'dd/mm/yyyy'

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.31 Класс Document

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

Пример:

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

5.31.1 Метод Document::saveAs

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

При необходимости, в качестве второго аргумента можно использовать объект класса `SaveDocumentSettings`, которая содержит формат документа `DocumentFormat`, тип документа `DocumentType` и пароль для защиты документа от несанкционированного доступа.

Существуют следующие варианты реализации метода:

```
Document saveAs(String filePath);
Document saveAs(String filePath, SaveDocumentSettings saveDocumentSettings);
```

Примеры использования метода `saveAs` приведены в разделах Сохранение и экспорт текстового документа и Сохранение и экспорт табличного документа.

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

5.31.2 Метод Document::exportAs

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

Расширенные версии метода позволяют указать дополнительные настройки экспорта документа:

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

– для текстовых документов – класс `TextExportSettings`;
– для табличных документов – класс `WorkbookExportSettings`.

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

Существуют следующие варианты реализации метода:

```
exportAs(String filePath, ExportFormat exportFormat);
exportAs(String filePath, ExportFormat exportFormat, TextExportSettings
textExportSettings);
exportAs(String filePath, ExportFormat exportFormat, WorkbookExportSettings
workbookExportSettings);
```

Примеры использования метода `exportAs` приведены в разделах Сохранение и экспорт текстового документа и Сохранение и экспорт табличного документа.

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

5.31.3 Метод Document::merge

Метод `Document::merge` сравнивает текущий документ с другим документом, который передается в параметре типа `Document`.

Метод возвращает объект `Document`, содержащий результат сравнения в виде отслеживаемых изменений.

Пример:

```
var firstDoc = application.loadDocument("C:/Tmp/Sample1.docx", loadSettings);
var secondDoc = application.loadDocument("C:/Tmp/Sample2.docx", loadSettings);

var mergedDoc = firstDoc.merge(secondDoc);
var outputFilePath = "C:/Tmp/Sample3.docx";

mergedDoc.saveAs(outputFilePath);
```

Результат выполнения данного примера (сравнение двух документов, содержащих "1111" и "2222") приведен на рисунке 25.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

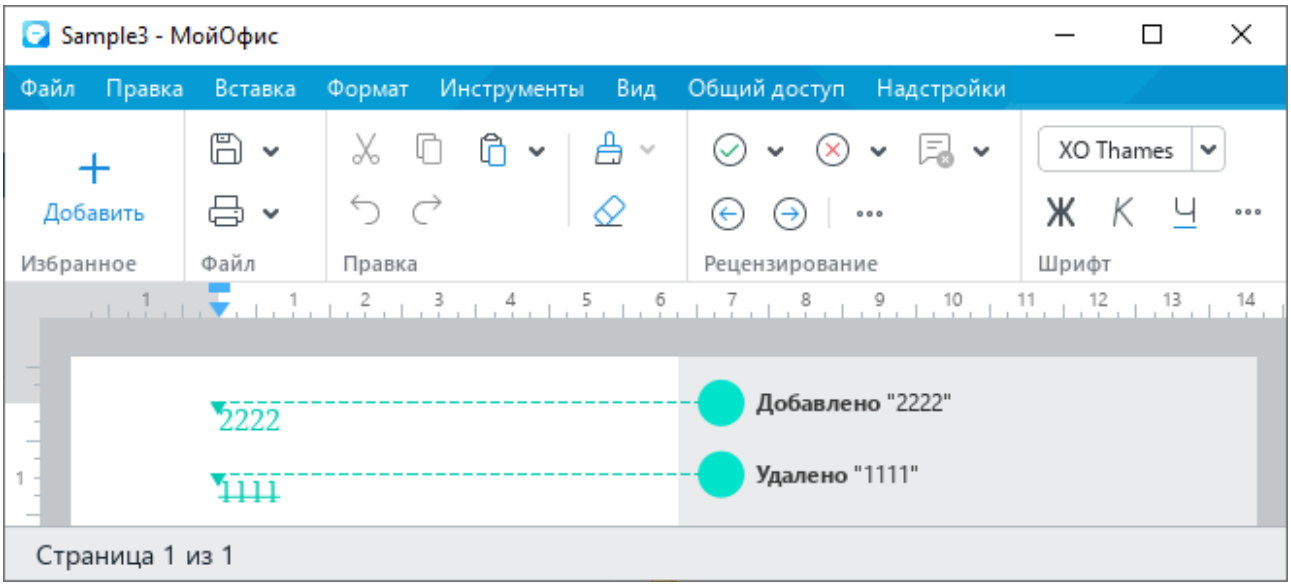


Рисунок 25 – Результат выполнения метода `merge`

5.31.4 Метод Document::getBlocks

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

Пример:

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

5.31.5 Метод Document::getBookmarks

Метод предоставляет доступ к списку закладок `Bookmarks`.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.31.6 Метод Document::getScripts

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

Пример:

```
Scripts scripts = document.getScripts();
ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
foreach (var script in scriptsEnumerator)
{
    Console.WriteLine(script.getName());
}
```

5.31.7 Метод Document::getRange

Метод предоставляет доступ ко всему диапазону Range документа.

Пример:

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

5.31.8 Метод Document::isChangesTrackingEnabled

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

Пример:

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

5.31.9 Метод Document::setChangesTrackingEnabled

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

Пример:

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

5.31.10 Метод Document::getComments

Метод обеспечивает доступ к комментариям документа, возвращает объект Comments.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
foreach (var comment in commentsEnumerator)
{
    Console.WriteLine(comment.getRange().getComment());
}
```

5.31.11 Метод Document::setPageProperties

Метод устанавливает свойство PageProperties в документе.

Пример:

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

5.31.12 Метод Document::setFormulaType

Метод устанавливает поддерживаемую адресацию ячеек FormulaType документа.

Пример:

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

5.31.13 Метод Document::getFormulaType

Метод возвращает поддерживаемую адресацию ячеек FormulaType документа.

Пример:

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

5.31.14 Метод Document::setPageOrientation

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.31.15 Метод Document::getSectionsEnumerator

Позволяет перечислить секции документа, тип Section.

Пример:

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

5.31.16 Метод Document::getSections

Возвращает объект типа Sections.

Пример:

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

5.31.17 Метод Document::setMirroredMarginsEnabled

Метод позволяет включать и выключать зеркальные поля в документе.

Пример:

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

5.31.18 Метод Document::areMirroredMarginsEnabled

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

Пример:

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

5.31.19 Метод Document::getPivotTablesManager

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
PivotTablesManager pivotTablesManager = document.getPivotTablesManager();
if (pivotTablesManager != null) {
    ..........................
}
```

5.31.20 Метод Document::getNamedExpressions

Используется для получения списка именованных диапазонов `NamedExpressions`.

Пример:

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

5.32 Метод DocumentAPI.createSearch

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

Пример:

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

5.33 Метод DocumentAPI.createScripting

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

Пример:

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

5.34 Класс DateTimeFormat

В таблице 14 представлены варианты масштабирования при печати табличных документов. Используется в качестве параметра метода `CellRange::insertCurrentDateTime()`.

Таблица 14 – Варианты масштабирования при печати табличных документов

Наименование константы	Описание
`DateTimeFormat.DateTime`	Дата/время
`DateTimeFormat.Date`	Дата
`DateTimeFormat.Time`	Время

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.35 Класс DocumentFormat

В таблице 15 приведены поддерживаемые форматы документов, используются в поле `documentType` класса `DocumentSettings`.

Таблица 15 – Форматы документов

Константа	Описание
`PlainText`	Используется для работы с файлами TXT.
`DSV`	Используется для работы с табличными данными в текстовой форме (CSV, DSV). Строка текста содержит одно или несколько полей данных, разделенных запятыми или иным разделителем.
`OXML`	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML.
`ODF`	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010).
`HTML`	Используется для работы с веб-документами в формате HTML. Работа с веб-документами в формате HTML средствами Document API в настоящий момент не поддерживается.
`PDF`	Используется для работы с документами в формате Portable Document Format (PDF) версии 1.4.
`PDFA`	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

5.36 Класс DocumentSettings

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

Описание полей класса `DocumentSettings` представлено в таблице 16.

Таблица 16 – Описание полей класса `DocumentSettings`

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

5.37 Класс DocumentType

В таблице 17 приведены поддерживаемые типы документов, используются при

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

создании документа Application::createDocument, DocumentSettings.

Таблица 17 - Типы документов

Наименование константы	Описание
DocumentType.Text	Используется для работы с текстовыми документами в форматах DOCX, ODT, XODT, TXT
DocumentType.Workbook	Используется для работы с табличными документами в форматах XLSX, ODS, XODS
DocumentType.Presentation	Используется для работы с презентациями в форматах PPTX, ODP. Работа с презентационными документами средствами Document API в настоящий момент не поддерживается

5.38 Класс DSVSettings

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

Описание полей класса DSVSettings представлено в таблице 18.

Таблица 18 – Описание полей класса DSVSettings

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

Пример:

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

5.39 Класс Encoding

В таблице 19 приведены поддерживаемые кодировки документов. Используется в LoadDocumentSettings.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 19 - Кодировки документов

Наименование константы	Кодировка
`Encoding.Unknown`	Невозможно определить кодировку
`Encoding.UTF8`	UTF8
`Encoding.UTF16BE`	UTF16BE
`Encoding.UTF16LE`	UTF16LE
`Encoding.UTF32BE`	UTF32BE
`Encoding.UTF32LE`	UTF32LE
`Encoding.Windows1250`	Windows1250
`Encoding.Windows1251`	Windows1251
`Encoding.Windows1252`	Windows1252
`Encoding.ISO8859Part5`	ISO8859Part5
`Encoding.KOI8R`	KOI8R
`Encoding.KOI8U`	KOI8U
`Encoding.CP866`	CP866

5.40 Класс ExportFormat

В таблице 20 приведены поддерживаемые форматы экспорта документов (см. `Document::exportAs`).

Таблица 20 - Форматы экспорта документов

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

5.41 Класс Field

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

5.42 Класс Fill

Класс определяет заполнение фигуры, используется в качестве поля `fill` класса `ShapeProperties`.

Варианты заполнения:

- без заполнения;
- заполнение цветом;

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

– фон задается путем к изображению фона.

Примеры:

```
// Без заполнения
shapeProperties.fill = new Fill();

// Заполнение цветом
shapeProperties.fill = new Fill(new Color(new ColorRGBA(55, 146, 179, 200)));

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

5.42.1 Метод Fill:getColor

Метод возвращает цвет заполнения `Color`.

5.42.2 Метод Fill:getUrl

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

5.42.3 Метод Fill:isNoFill

Метод возвращает `true`, если заполнения нет.

5.43 Класс FormulaType

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в таблице 21. Используется в `Document::getFormulaType()`, `Document::setFormulaType()`, `DocumentSettings`.

Таблица 21 – Системы адресации ячеек в табличном документе

Константа	Система адресации ячеек	Описание
`A1`	A1	Вариант A1 соответствует наиболее распространенной системе адресации ячеек, при которой столбцы задаются буквами, а строки – числами.
`R1C1`	R1C1	Вариант R1C1 соответствует альтернативной системе адресации ячеек, при которой столбцы и строки задаются числами.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.44 Класс Frame

Класс `Frame` описывает прямоугольную область графического объекта, находящегося в текстовой позиции документа (см. Рисунок 26). Предназначен для получения и изменения свойств позиции графических объектов. Используется в текстовом документе.

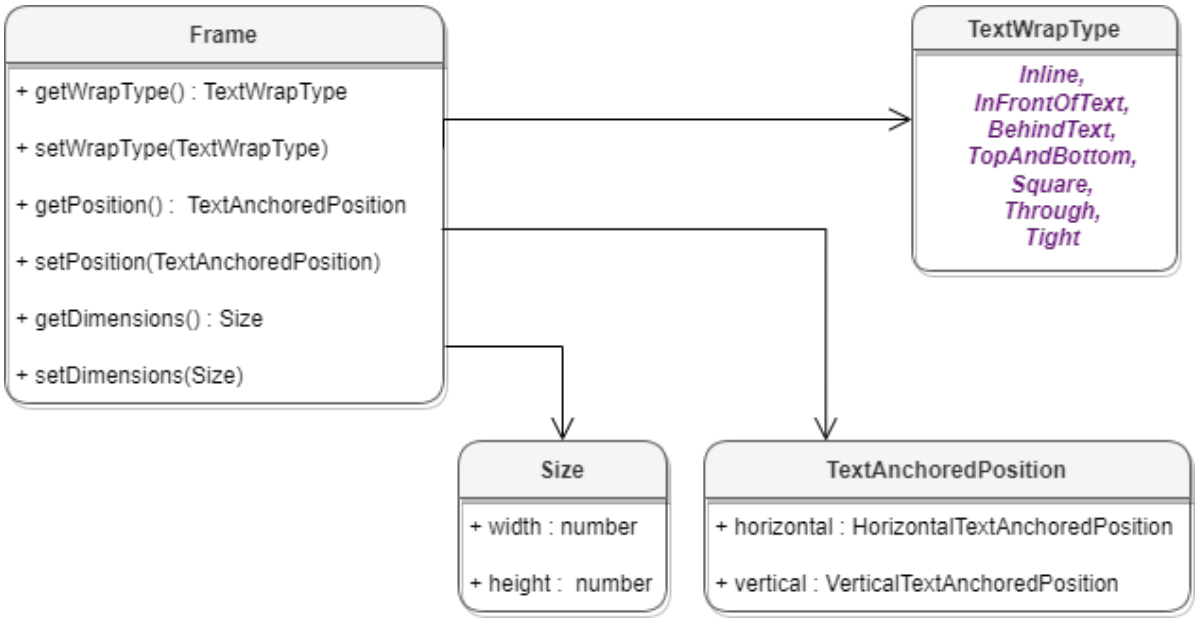


Рисунок 26 – Объектная модель класса `Frame`

Пример для текстового документа:

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
InlineObjectsEnumerator enumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in enumerator)
{
    var frame = inlineObject.getFrame();
    Console.WriteLine(frame.getDimensions().width);
}
```

5.44.1 Метод Frame:setPosition

Метод задает положение встроенного объекта, тип аргумента `TextAnchoredPosition`. Новая позиция может быть установлена только для встроенных объектов, тип переноса текста которых не является типом `TextWrapType.Inline`.

Примеры:

```
// Позиция встроенного объекта не может быть задана,
// если стиль переноса текста - inline.
// Сначала его следует изменить на тип, отличный от inline.
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
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();

position.horizontal = new
HorizontalTextAnchoredPosition(HorizontalRelativeTo.Page, 12.f);
position.vertical = new
VerticalTextAnchoredPosition(VerticalRelativeTo.PageTopMargin, 122.f);

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();

position.horizontal = new
HorizontalTextAnchoredPosition(HorizontalRelativeTo.Page,
HorizontalAnchorAlignment.Center);
position.vertical = new
VerticalTextAnchoredPosition(VerticalRelativeTo.PageTopMargin,
VerticalAnchorAlignment.Top);

frame.setPosition(position);
```

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
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);
```

5.44.2 Метод Frame:getPosition

Метод возвращает позицию встроенного объекта, тип AnchoredPosition.

Пример:

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
InlineObjectsEnumerator enumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in enumerator)
{
    var frame = inlineObject.getFrame();
    AnchoredPosition position = frame.getPosition();
    Console.WriteLine(position.textPosition.horizontal.alignment);
}
```

5.44.3 Метод Frame:setDimensions

Метод позволяет задать размер встроенного объекта (тип SizeU).

Пример:

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
InlineObjectsEnumerator enumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in enumerator)
{
    var frame = inlineObject.getFrame();
    SizeU frameDimensions = new SizeU(50, 50);
    frame.setDimensions(frameDimensions);
    Console.WriteLine(frame.getDimensions().width);
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.44.4 Метод Frame:getDimensions

Метод возвращает задает размеры встроенного объекта, тип - SizeU.

Пример:

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
InlineObjectsEnumerator enumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in enumerator)
{
    var frame = inlineObject.getFrame();
    Console.WriteLine(frame.getDimensions().width);
}
```

5.44.5 Метод Frame:setWrapType

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

Пример:

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

5.44.6 Метод Frame:getWrapType

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

Пример:

```
    var frame = inlineObject.getFrame();
    Console.WriteLine(frame.getWrapType());
```

5.45 Класс HeadersFooters

Класс HeadersFooters представляет коллекцию верхних и нижних колонтитулов раздела (см. Рисунок 27). Доступ к колонтитулам осуществляется посредством методов Section::getHeaders(), Section::getFooters().

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

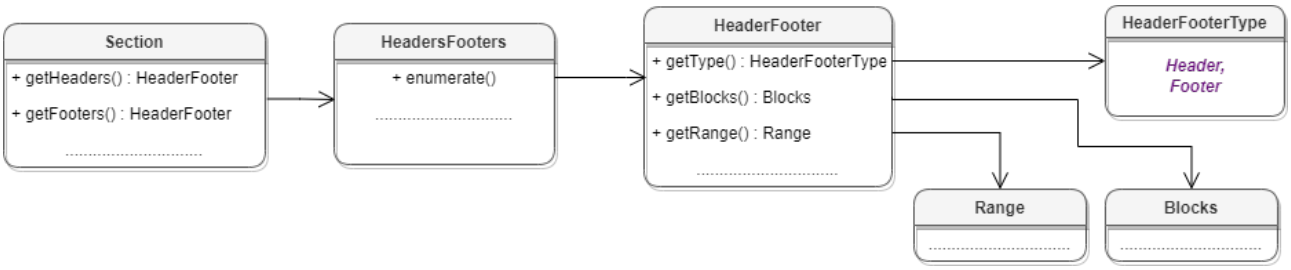


Рисунок 27 – Классы для работы с колонтитулами

5.45.1 Метод HeadersFooters::GetEnumerator

Метод возвращает коллекцию колонтитулов.

Пример:

```
Block block = document.getBlocks().getBlock(0);
if (block != null)
{
    Section section = block.getSection();
    HeadersFooters headersFooters = section.getHeaders();
    HeaderFootersEnumerator headersEnumerator = headersFooters.GetEnumerator();
    foreach (var header in headersEnumerator)
    {
        Console.WriteLine(header.getType());
    }
}
```

5.46 Класс HeaderFooter

Класс `HeaderFooter` определяет колонтитул текстового документа.

5.46.1 Метод HeaderFooter::getType

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.46.2 Метод HeaderFooter::getBlocks

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

Пример:

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

5.46.3 Метод HeaderFooter::getRange

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

Пример:

```
foreach (var headerFooter in headersEnumerator)
{
    Range range = headerFooter.getRange();
    Console.WriteLine(range.extractText());
}
```

5.47 Класс HeaderFooterType

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

5.48 Класс HorizontalTextAnchoredPosition

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

Описание полей класса `HorizontalTextAnchoredPosition` представлено в таблице 22.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 22 – Описание полей класса HorizontalTextAnchoredPosition

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

5.49 Класс HorizontalRelativeTo

В таблице 23 представлены типы размещения объекта относительно закрепленной позиции по горизонтали (см. описание класса HorizontalTextAnchoredPosition).

Таблица 23 – Типы размещения объекта относительно закрепленной позиции по горизонтали

Наименование константы	Описание
HorizontalRelativeTo.Character	Символ
HorizontalRelativeTo.Column	Столбец
HorizontalRelativeTo.ColumnLeftMargin	Левое поле столбца
HorizontalRelativeTo.ColumnRightMargin	Правое поле столбца
HorizontalRelativeTo.ColumnInsideMargin	Внутреннее поле столбца
HorizontalRelativeTo.ColumnOutsideMargin	Внешнее поле столбца
HorizontalRelativeTo.Page	Страница
HorizontalRelativeTo.PageContent	Содержимое страницы
HorizontalRelativeTo.PageLeftMargin	Левое поле страницы
HorizontalRelativeTo.PageRightMargin	Правое поле страницы
HorizontalRelativeTo.PageInsideMargin	Внутреннее поле страницы
HorizontalRelativeTo.PageOutsideMargin	Внешнее поле страницы

5.50 Класс HorizontalAnchorAlignment

В таблице 24 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали (см. описание класса HorizontalTextAnchoredPosition).

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 24 – Типы выравнивания объекта относительно закрепленной позиции по горизонтали

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

5.51 Класс Image

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

5.51.1 Метод Image:getFrame

Метод аналогичен методу `InlineObject::getFrame()`, он возвращает свойства позиции изображения. В зависимости от текущего редактора метод возвращает разные типы рамок. Графические объекты текстового редактора для описания местоположения и размеров используют тип `Frame`.

Пример для текстового документа:

```
var images = document.getRange().getImages();
var imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    Console.WriteLine(image.getFrame()); // NCT.MyOfficeSDK.Frame
}
```

5.52 Класс Images

Класс `Images` используется для доступа к коллекции изображений. Объект может быть получен в текстовом документе посредством вызова метода `Range::getImages()`.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(image); // NCT.MyOfficeSDK.Image
}
```

5.52.1 Метод Images:GetEnumerator

Метод позволяет перечислить коллекцию изображений.

Пример для текстового документа:

```
var images = document.getRange().getImages();
var imagesEnumerator = images.GetEnumerator();
foreach (var image in imagesEnumerator)
{
    Console.WriteLine(image); // NCT.MyOfficeSDK.Image
}
```

5.53 Класс InlineObject

Класс `InlineObject` представляет собой встроенный объект документа. Может быть получен посредством использования метода `InlineObjects::getEnumerator()`.

5.53.1 Метод InlineObject:toImage

Метод возвращает изображение `Image`, связанное со встроенным объектом текстового документа. Если объект не является изображением, метод возвращает `null`.

Пример:

```
var inlineObjectsEnumerator =
document.getRange().getInlineObjects().GetEnumerator();
foreach (var inlineObject in inlineObjectsEnumerator)
{
    var image = inlineObject.toImage();
    if (image != null)
    {
        Console.WriteLine("Текущий объект является изображением");
    }
    else
    {
        Console.WriteLine("Текущий объект является фигурой");
    }
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.53.2 Метод InlineObject:getFrame

Метод возвращает свойства позиции встроенного объекта. Графические объекты текстового редактора привязаны к позиции в документе, поэтому для описания местоположения и размеров используют класс `Frame`.

Пример для текстового документа:

```
var inlineObjectsEnumerator =
document.getRange().getInlineObjects().GetEnumerator();
foreach (var inlineObject in inlineObjectsEnumerator)
{
    var frame = inlineObject.getFrame();
    Console.WriteLine(frame.getDimensions().width);
}
```

5.54 Класс InlineObjects

Класс `InlineObjects` предназначен для доступа к коллекции графических объектов в текстовом документе. Объект может быть получен вызовом метода `Range::getInlineObjects()` (см. Рисунок 28).

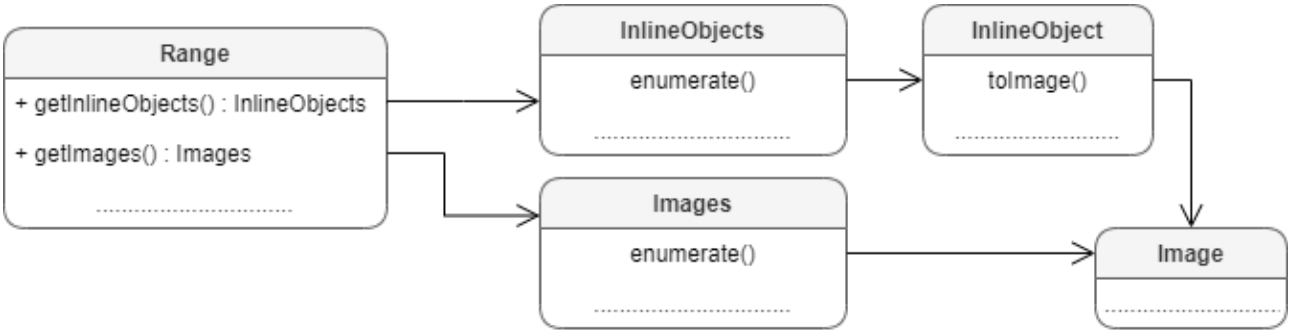


Рисунок 28 – Встроенные объекты

5.54.1 Метод InlineObjects::getEnumerator

Метод позволяет перечислить коллекцию встроенных объектов в текстовом документе.

Пример для текстового документа:

```
InlineObjects inlineObjects = document.getRange().getInlineObjects();
InlineObjectsEnumerator iinlineObjectsEnumerator =
inlineObjects.GetEnumerator();
foreach (var inlineObject in iinlineObjectsEnumerator)
{
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(inlineObject.getFrame().getWrapType());
}
```

5.55 Класс Insets

Класс `Insets` предназначен для задания полей, например, страницы. Поля класса `Insets` представлены в таблице 25. Используется в поле `margins` класса `PageProperties`.

Таблица 25 – Описание полей класса `Insets`

Поле	Тип	Описание
`left`	`boost::optional<Unit>`	Левая граница поля
`top`	`boost::optional<Unit>`	Верхняя граница поля
`right`	`boost::optional<Unit>`	Правая граница поля
`bottom`	`boost::optional<Unit>`	Нижняя граница поля

Пример:

```
PageProperties pageProperties = new PageProperties();
Insets insets = new Insets();
insets.left = 0;
insets.top = 0;
insets.right = 100;
insets.bottom = 100;
pageProperties.margins = insets;
document.setPageProperties(pageProperties);
```

5.56 Класс LineEndingProperties

Класс `LineEndingProperties` содержит варианты оформления окончаний линий. Описание полей класса `LineEndingProperties` представлено в таблице 26. Используется в полях `headLineEndingProperties` и `tailLineEndingProperties` класса `LineProperties`.

Таблица 26 – Описание полей класса `LineEndingProperties`

Поле	Тип	Описание
`LineEndingProperties.style`	`LineEndingStyle`	Стиль окончания линии
`LineEndingProperties.relativeExtent`	`SizeU`	Размер окончания линии относительно ее ширины

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

LineProperties lineProperties = new LineProperties();
lineProperties.headLineEndingProperties = new LineEndingProperties();
lineProperties.headLineEndingProperties.style = LineEndingStyle.Arrow;
lineProperties.headLineEndingProperties.relativeExtent = new SizeU(2, 2);

lineProperties.tailLineEndingProperties = new LineEndingProperties();
lineProperties.tailLineEndingProperties.style = LineEndingStyle.Arrow;
lineProperties.tailLineEndingProperties.relativeExtent = new SizeU(2, 2);

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

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

5.57 Класс LineEndingStyle

В таблице 27 приведены типы окончания линии. Используется в поле `style` класса `LineEndingProperties`.

Таблица 27 – Типы окончания линии

Наименование константы	Описание
`LineEndingStyle.Arrow`	
`LineEndingStyle.Diamond`	
`LineEndingStyle.Oval`	
`LineEndingStyle.Stealth`	
`LineEndingStyle.Triangle`	
`LineEndingStyle.None`	

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

LineProperties lineProperties = new LineProperties();
lineProperties.headLineEndingProperties = new LineEndingProperties();
lineProperties.headLineEndingProperties.style = LineEndingStyle.Arrow;

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

5.58 Класс LineProperties

Класс `LineProperties` предназначен для установки таких параметров линии, как тип, ширина, цвет (см. Рисунок 29).

Поля класса:

– `style` - тип линии, допустимые значения представлены в разделе `LineStyle`;

– `width` - толщина линии;

– `color` - цвет линии, тип - `Color`;

– `headLineEndingProperties` - тип начала линии, тип - `LineEndingProperties`;

– `tailLineEndingProperties` - тип окончания линии, тип - `LineEndingProperties`.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

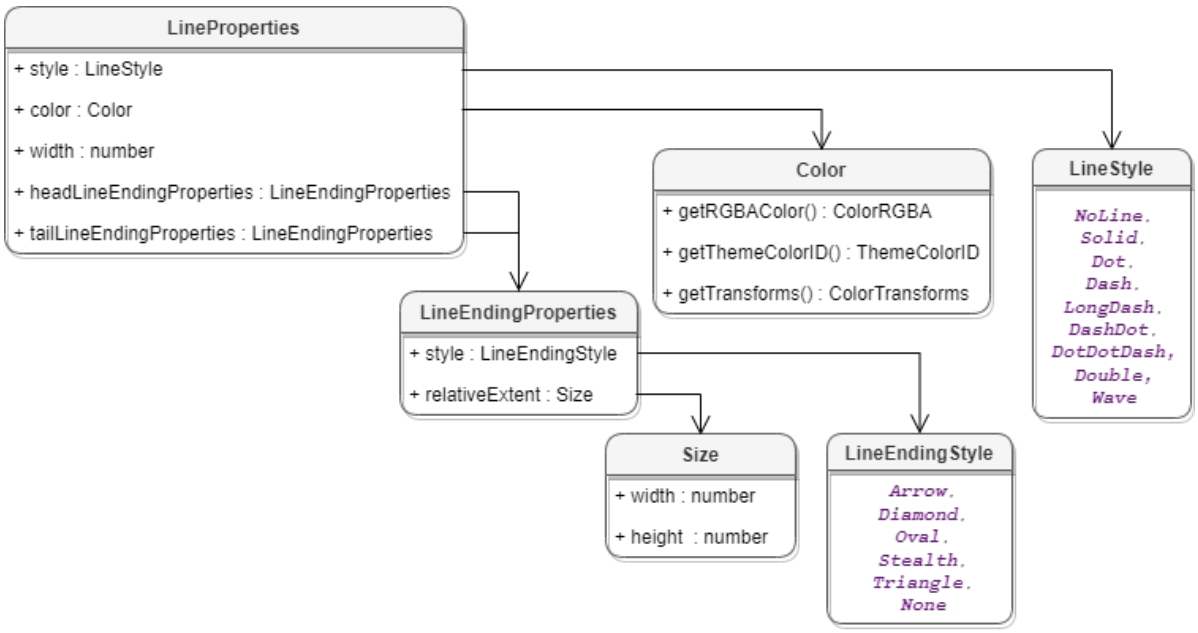


Рисунок 29 – Свойства границ ячеек

Пример:

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

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

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.59 Класс LineSpacing

Класс LineSpacing задает межстрочный интервал абзаца. Поля класса приведены в таблице 28. Для управления значением межстрочного интервала используются значения, представленные в разделе `LineSpacingRule`.

Таблица 28 – Параметры межстрочного интервала

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

Пример:

```
Paragraph paragraph = paragraphsEnumerator.Current;
ParagraphProperties paragraphProperties = paragraph.getParagraphProperties();
// Конструктор
paragraphProperties.lineSpacing = new LineSpacing(5.0f,
LineSpacingRule.Multiple);
// Обращение к полям
paragraphProperties.lineSpacing.value = 1;
paragraphProperties.lineSpacing.rule = LineSpacingRule.Exact;
```

5.60 Класс LineSpacingRule

Класс `LineSpacingRule` содержит типы межстрочного интервалов.

Типы межстрочных интервалов:

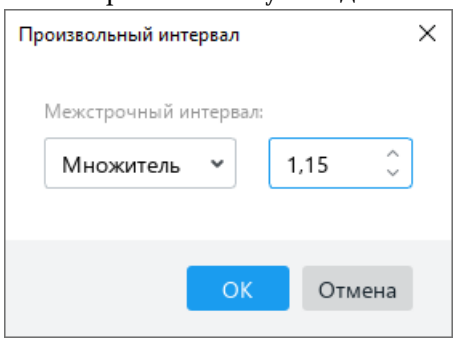
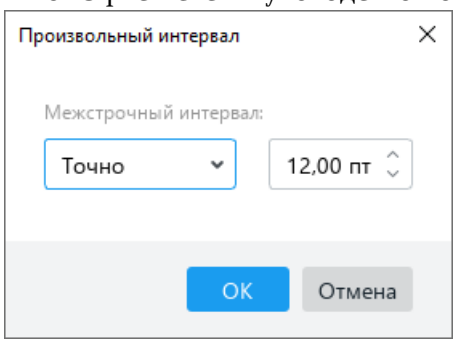
- `Multiple` – межстрочный интервал с использованием множителя;
- `Exact` – межстрочный интервал с использованием точного значения;
- `AtLeast` – межстрочный интервал с использованием минимального значения.

В таблице 29 представлены описания правил формирования межстрочного интервала текстового абзаца.

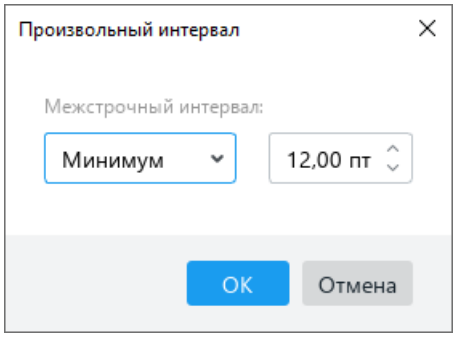
Таблица 29 – Виды межстрочного интервала

Наименование константы	Описание
`LineSpacingRule.Multiple`	Установка произвольного межстрочного интервала с использованием множителя. При вызове необходимо указать значение множителя, например: `pPr.lineSpacing= new LineSpacing(1.15f, LineSpacingRule.Multiple)`

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Наименование константы	Описание
	

Пример:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    ParagraphProperties paraProps = paragraph.getParagraphProperties();
    paraProps.lineSpacing = new LineSpacing(5.0f, LineSpacingRule.Multiple);
    paragraph.setParagraphProperties(paraProps);
}
```

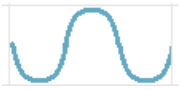
5.61 Класс LineStyle

В таблице 30 приведены типы линий. Используется в поле `style` класса `LineProperties`.

Таблица 30 – Типы линий

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Наименование константы	Описание
LineStyle.Double	
LineStyle.Wave	

Пример:

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

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

5.62 Класс ListSchema

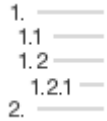
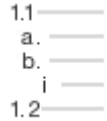
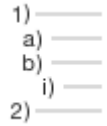
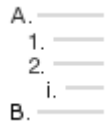
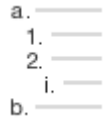
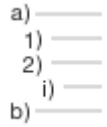
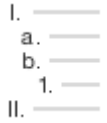
Класс `ListSchema` содержит типы схем форматирования списков, которые могут быть применены к абзацам текста. Данные константы используются в методах `Paragraph::getListSchema()`, `Paragraph::setListSchema()`.

Описания схем форматирования списков представлены в таблице 31.

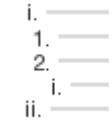
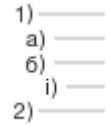
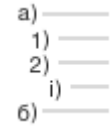
Таблица 31 – Описания схем списков

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Тип схемы списка	Описание типа схемы списка	Изображение
`ListSchema.EnumeratorRomanLowercaseDot`	Нумерация римскими строчными цифрами с точкой	
`ListSchema.EnumeratorDecimalRussianBracket`	Десятичная нумерация через запятую со скобкой	
`ListSchema.EnumeratorRussianLowercaseBracket`	Нумерация с русскими строчными буквами со скобкой	

Пример:

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

5.63 Класс LoadDocumentSettings

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

Описание полей класса `LoadDocumentSettings` представлено в таблице 32.

Таблица 32 – Описание полей класса `LoadDocumentSettings`

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.64 Класс LocaleInfo

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

Описание полей `LocaleInfo` представлено в таблице 33.

Таблица 33 – Описание полей класса `LocaleInfo`

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

5.65 Класс Message

Класс `Message` предназначен для формирования событий лога.

5.65.1 Класс Message::Severity

Класс `Message.Severity` (Таблица 34) описывает уровни сообщений лога (информация, предупреждение, ошибка).

Таблица 34 – Описание уровней лога `Message.Severity`

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Поле	Описание
`Message.Severity.Error`	Ошибка

5.65.2 Метод Message::getSeverity

Метод возвращает уровень лога `Message.Severity`.

5.65.3 Метод Message::getText

Метод возвращает текст сообщения.

5.65.4 Метод Message::makeInfo

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

5.65.5 Метод Message::makeWarning

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

5.65.6 Метод Message::makeError

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

5.66 Класс Messenger

Служит для организации логов приложений.

5.66.1 Метод Messenger:subscribe

Метод служит для подписки на события лога.

Пример:

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

5.66.2 Метод Messenger:notify

Метод используется для создания события лога

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.67 Класс NamedExpressions

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

5.67.1 Метод NamedExpressions::get

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

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpression namedExpression = namedExpressions.get("Продажи");
if (namedExpression != null) {
    Console.WriteLine(namedExpression.getName());
}
```

5.67.2 Метод NamedExpressions::getEnumerator

Позволяет получить доступ ко всему списку именованных диапазонов.

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpressionsEnumerator enumerator = namedExpressions.getEnumerator();
foreach (var namedExpression in enumerator)
{
    Console.WriteLine(namedExpression.getName());
}
```

5.67.3 Метод NamedExpression::addExpression

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

Пример:

```
String expressionName = "Покупки";
String expressionValue = "=Формула покупки!$E$6:$E$14";
NamedExpressionsValidationResult validationResult =
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
namedExpressions.addExpression(expressionName, expressionValue);
Console.WriteLine(validationResult);
```

5.67.4 Метод NamedExpressions::removeExpression

Удаляет диапазон 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);
```

5.68 Класс NamedExpression

Класс описывает структуру именованного диапазона.

Пример:

```
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpressionsEnumerator enumerator = namedExpressions.getEnumerator();
foreach (var namedExpression in enumerator)
{
    Console.WriteLine(namedExpression.getName());
    Console.WriteLine(namedExpression.getExpression());
    CellRange cellRange = namedExpression.getCellRange();
    Console.WriteLine(cellRange.getBeginColumn());
    Console.WriteLine(cellRange.getLastColumn());
}
```

5.68.1 Метод NamedExpression::getName

Возвращает имя именованного диапазона. Пример см. в NamedExpression.

5.68.2 Метод NamedExpression::getExpression

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

5.68.3 Метод NamedExpression::getCellRange

Возвращает диапазон ячеек CellRange. Пример см. в NamedExpression.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.69 Класс NamedExpressionsValidationResult

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

Класс содержит следующие поля:

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

5.70 Класс NumberCellFormatting

Класс содержит параметры для числового формата ячеек таблицы, используется в качестве аргумента метода `Cell::setFormat()`. Описание полей класса `NumberCellFormatting` представлено в таблице 35.

Таблица 35 – Описание полей класса `NumberCellFormatting`

Поле	Описание
`NumberCellFormatting.decimalPlaces`	Количество десятичных позиций
`NumberCellFormatting.useThousandsSeparator`	Использовать разделитель для тысячных
`NumberCellFormatting.useRedForNegative`	Использовать красный цвет для отрицательных значений
`NumberCellFormatting.useBracketsForNagative`	Использовать скобки для отрицательных значений
`NumberCellFormatting.hideSign`	Скрывать знак «минус» для отрицательных значений

Пример:

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

NumberCellFormatting cellFormat = new NumberCellFormatting();
cellFormat.decimalPlaces = 2;
cellFormat.useThousandsSeparator = true;
cellFormat.useRedForNegative = true;
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
cellFormat.useBracketsForNegative = true;
cellFormat.hideSign = false;

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

5.71 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);
```

5.71.1 Метод PageNumbers::contains

Метод служит для проверки вхождения заданного номера страницы в коллекцию номеров страниц `PageNumbers`.

Пример:

```
PageNumbers pageNumbers = new PageNumbers(1, 20);
Console.WriteLine(pageNumbers.contains(2));
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.71.2 Метод PageNumbers::getLast

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

Пример:

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

5.72 Класс PageOrientation

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

Примеры:

```
Block block = document.getBlocks().getBlock(0);
if (block != null) {
    Section section = block.getSection();
    section.setPageOrientation(PageOrientation.Portrait);
    Console.WriteLine(section.getPageOrientation());
}
```

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    section.setPageOrientation(PageOrientation.Portrait);
    Console.WriteLine(section.getPageOrientation());
}
```

5.73 Класс PageParity

Варианты выбора страниц для экспорта и печати представлены в таблице 36. Используется в `PageNumbers`.

Таблица 36 – Варианты выбора страниц для экспорта и печати

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.74 Класс PageProperties

Класс `PageProperties` предоставляет такие свойства страницы как высота, ширина, размеры полей. Описание полей приведено в таблице 37. Используется в `Document::setPageProperties()`, `Section::getPageProperties()`, `Section::setPageProperties()`.

Таблица 37 – Описание полей класса `PageProperties`

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

Примеры:

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

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

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

5.75 Класс Paragraphs

Класс `Paragraphs` предоставляет доступ к коллекции абзацев типа `Paragraph` (см. Рисунок 30). Коллекция абзацев может быть получена из объекта `Range` посредством использования метода `Range::getParagraphs()`.

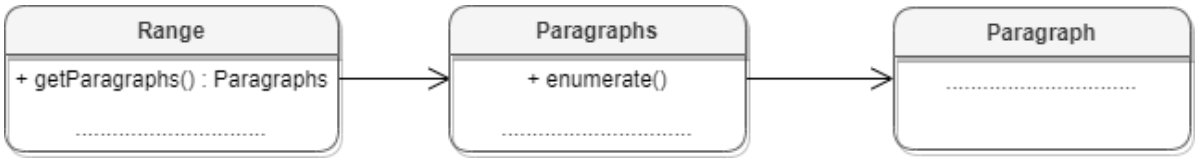


Рисунок 30 – Объектная модель для работы со списком абзацев

Пример для текстового документа:

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
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();
```

5.75.1 Метод Paragraphs::setListSchema

Метод устанавливает тип маркированного или нумерованного списка `ListSchema`. Данный метод используется только в текстовом документе.

Пример:

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

5.75.2 Метод Paragraphs::setListLevel

Метод устанавливает глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример:

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

5.75.3 Метод Paragraphs::increaseListLevel

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.75.4 Метод Paragraphs::decreaseListLevel

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

Пример:

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

5.75.5 Метод Paragraphs::GetEnumerator

Метод позволяет перечислить коллекцию абзацев.

Пример:

```
NCT.MyOfficeSDK.Range range = document.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    Console.WriteLine(paragraph.getRange().extractText());
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.76 Класс Paragraph

Класс `Paragraph` предоставляет доступ к свойствам абзаца (см. Рисунок 31).

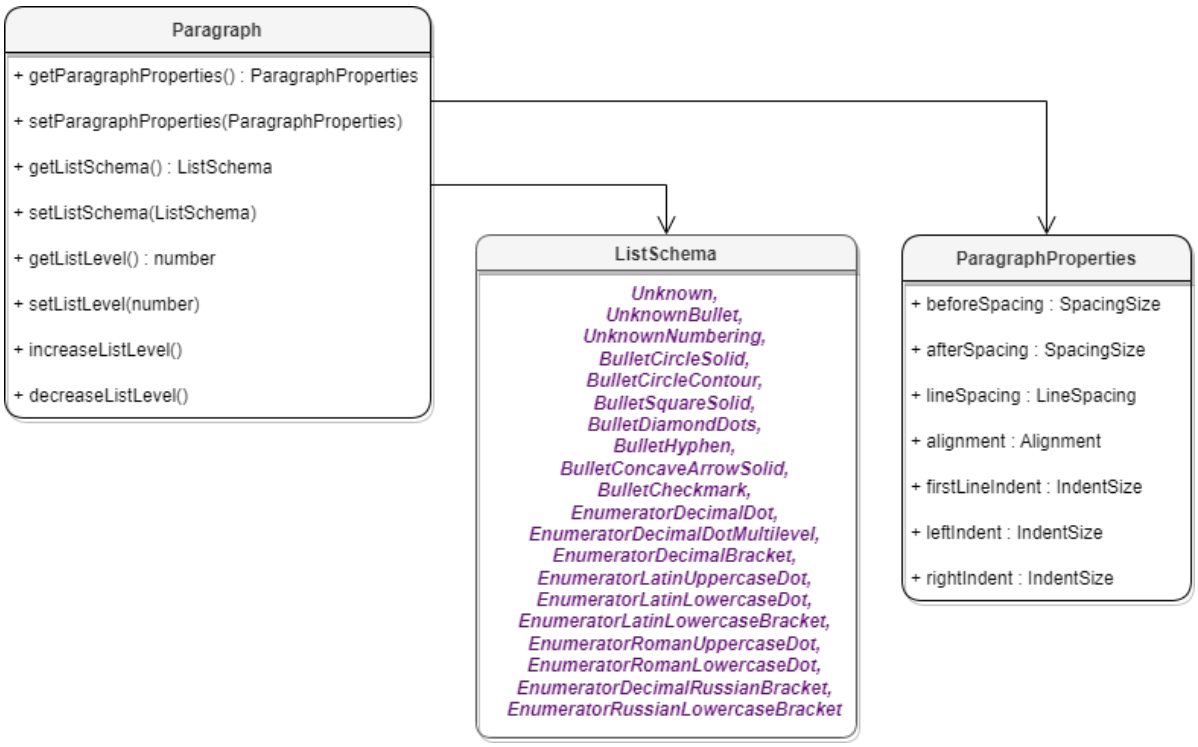


Рисунок 31 – Объектная модель классов для работы со свойствами параграфа

5.76.1 Метод Paragraph::getParagraphProperties

Метод предоставляет доступ к классу, определяющему такие свойства абзаца `ParagraphProperties`, как выравнивание текста, межстрочные интервалы, отступы и т. д.

Пример для текстового документа:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
}
```

Пример для табличного документа:

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
NCT.MyOfficeSDK.Range range = cell.getRange();
Paragraphs paragraphs = range.getParagraphs();
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var Paragraphs in paragraphsEnumerator)
{
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    Console.WriteLine(paragraphProperties.alignment);
}
```

5.76.2 Метод Paragraph::setParagraphProperties

Метод предназначен для обновления свойств форматирования абзаца ParagraphProperties.

Пример для текстового документа:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    paragraphProperties.alignment = Alignment.Left;
    paragraph.setParagraphProperties(paragraphProperties);
}
```

Пример для табличного документа:

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("B2");
NCT.MyOfficeSDK.Range range = cell.getRange();
Paragraphs paragraphs = range.getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    ParagraphProperties paragraphProperties =
paragraph.getParagraphProperties();
    paragraphProperties.alignment = Alignment.Left;
    paragraph.setParagraphProperties(paragraphProperties);
}
```

5.76.3 Метод Paragraph::getListSchema

Метод возвращает схему форматирования абзаца ListSchema, если схема нумерации

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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());
}
```

5.76.4 Метод Paragraph::setListSchema

Метод позволяет установить тип маркированного или нумерованного списка `ListSchema`. Данный метод используется только в текстовом документе.

Пример:

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

5.76.5 Метод Paragraph::getListLevel

Метод позволяет получить глубину вложенности элемента списка. Данный метод используется только в текстовом документе.

Пример:

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

5.76.6 Метод Paragraph::setListLevel

Метод позволяет установить глубину вложенности элемента списка.

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
Blocks blocks = document.getBlocks();
Paragraph paragraph = blocks.getParagraph(0);
if (paragraph != null) {
    paragraph.setListLevel(1);
    Console.WriteLine(paragraph.getListLevel());
}
```

5.76.7 Метод Paragraph::increaseListLevel

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

Пример:

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

5.76.8 Метод Paragraph::decreaseListLevel

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

Пример:

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

5.77 Класс ParagraphProperties

Класс `ParagraphProperties` предназначен для управления свойствами форматирования (см. Рисунок 32). Класс `ParagraphProperties` используется в методах `Paragraph::getParagraphProperties()` и `Paragraph::setParagraphProperties()`.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

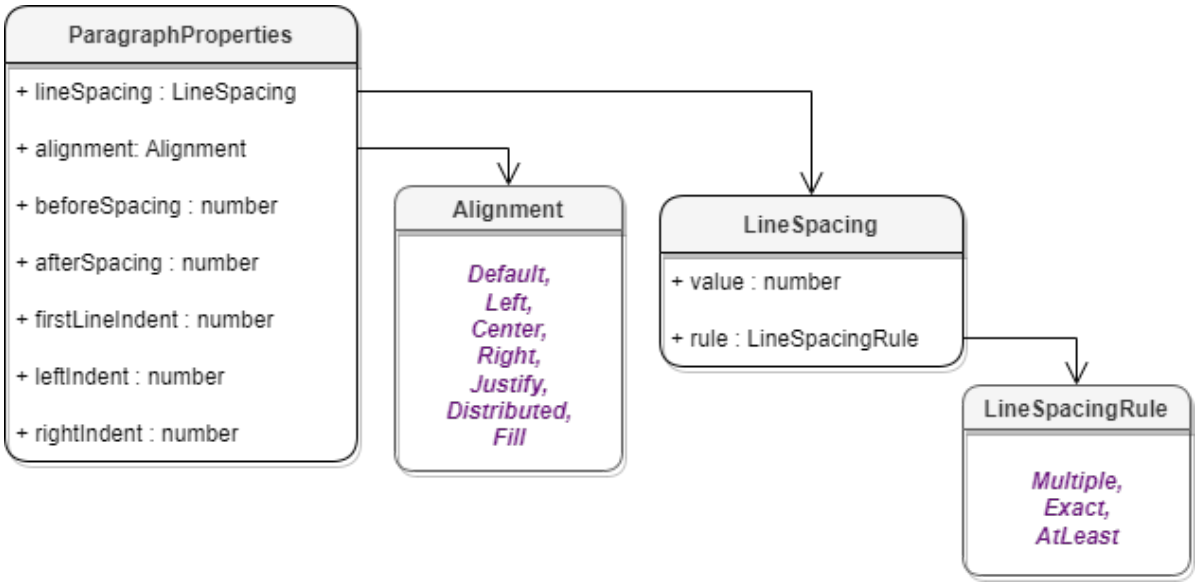
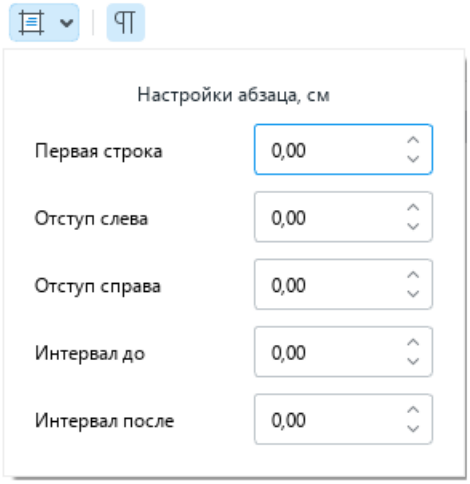


Рисунок 32 – Объектная модель классов для работы со свойствами параграфа

Описание полей класса `ParagraphProperties` представлено в таблице 38.

Таблица 38 – Описание полей класса `ParagraphProperties`

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

Пример для текстового документа:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    paraProps.lineSpacing = new LineSpacing(5.0f, LineSpacingRule.Multiple);
}
```

Пример для табличного документа:

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

5.78 Класс PivotTablesManager

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

Пример:

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

5.78.1 Метод PivotTablesManager:create

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

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.79 Класс PivotTable

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

5.79.1 Метод PivotTable::remove

Метод удаляет сводную таблицу.

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null) {
    pivotTable.remove();
}
```

5.79.2 Метод PivotTable::getSourceRangeAddress

Метод возвращает текстовое представление диапазона исходных данных сводной таблицы.

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null) {
    pivotTable.remove();
}
```

5.79.3 Метод PivotTable::getSourceRange

Метод возвращает диапазон `CellRange` исходных данных сводной таблицы.

Пример:

```
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null) {
    CellRange sourceRange = pivotTable.getSourceRange();
    Console.WriteLine(sourceRange.getBeginColumn());
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.79.4 Метод PivotTable::getPivotRange

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

Пример:

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

5.79.5 Метод PivotTable::changeSourceRange

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

Пример:

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

5.79.6 Метод PivotTable::isRowGrandTotalEnabled

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

Пример:

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

5.79.7 Метод PivotTable::isColumnGrandTotalEnabled

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

Пример:

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

5.79.8 Метод PivotTable::getPivotTableCaptions

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

Пример:

```
PivotTableCaptions pivotTableCaptions = pivotTable.getPivotTableCaptions();
Console.WriteLine(pivotTableCaptions.errorCaption);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
Console.WriteLine(pivotTableCaptions.emptyCaption);
Console.WriteLine(pivotTableCaptions.grandTotalCaption);
Console.WriteLine(pivotTableCaptions.valuesHeaderCaption);
Console.WriteLine(pivotTableCaptions.columnHeaderCaption);
Console.WriteLine(pivotTableCaptions.rowHeaderCaption);
```

5.79.9 Метод PivotTable::getPivotTableLayoutSettings

Метод возвращает настройки отображения PivotTableLayoutSettings сводной таблицы.

Пример:

```
PivotTableLayoutSettings layoutSettings =
pivotTable.getPivotTableLayoutSettings();
Console.WriteLine(layoutSettings.displayFieldCaptions);
Console.WriteLine(layoutSettings.indentForCompactLayout);
Console.WriteLine(layoutSettings.isMergeAndCenterLabelsEnabled);
Console.WriteLine(layoutSettings.pageFieldOrder);
Console.WriteLine(layoutSettings.pageFieldWrapCount);
Console.WriteLine(layoutSettings.reportLayout);
Console.WriteLine(layoutSettings.useGridDropZones);
Console.WriteLine(layoutSettings.valueFieldsOrientation);
```

5.79.10 Метод PivotTable::getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства PivotTableUnsupportedFeature сводной таблицы.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.79.11 Метод PivotTable::getFieldsList

Метод возвращает список PivotTableFields всех полей сводной таблицы.

Пример:

```
PivotTableFields pivotTableFields = pivotTable.getFieldsList();
PivotTableFields.PivotTableFieldsEnumerator enumerator =
pivotTableFields.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableField pivotTableField = enumerator.Current;
    Console.WriteLine(pivotTableField.customFormula);
}
```

5.79.12 Метод PivotTable::getRowFields

Метод возвращает список полей PivotTableCategoryField из области строк.

Пример:

```
PivotTableCategoryFields rowFields = pivotTable.getRowFields();
PivotTableCategoryFields.PivotTableCategoryFieldsEnumerator enumerator =
rowFields.GetEnumerator();
while (enumerator.MoveNext())
{
    PivotTableCategoryField pivotTableCategoryField = enumerator.Current;
    Console.WriteLine(pivotTableCategoryField.fieldProperties.fieldAlias);
    Console.WriteLine(pivotTableCategoryField.fieldProperties.subtotalAlias);
    Console.WriteLine(pivotTableCategoryField.fieldProperties.fieldName);
    PivotTableFunctions subtotalFunctions =
pivotTableCategoryField.subtotalFunctions;
    Console.WriteLine(subtotalFunctions.Count);
}
```

5.79.13 Метод PivotTable::getColumnFields

Метод возвращает список полей PivotTableCategoryField из области колонок.

Пример:

```
PivotTableCategoryFields columnFields = pivotTable.getColumnFields();
PivotTableCategoryFields.PivotTableCategoryFieldsEnumerator enumerator =
columnFields.GetEnumerator();
while (enumerator.MoveNext())
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
{
    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);
}
```

5.79.14 Метод PivotTable::getValueFields

Метод возвращает список полей PivotTableValueField из области значений.

Пример:

```
PivotTableValueFields valueFields = pivotTable.getValueFields();
PivotTableValueFields.PivotTableValueFieldsEnumerator enumerator =
valueFields.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableValueField pivotTableValueField = enumerator.Current;
    Console.WriteLine(pivotTableValueField.baseFieldName);
    Console.WriteLine(pivotTableValueField.cellNumberFormat);
    Console.WriteLine(pivotTableValueField.customFormula);
    Console.WriteLine(pivotTableValueField.totalFunction);
    Console.WriteLine(pivotTableValueField.valueFieldName);
}
```

5.79.15 Метод PivotTable::getPageFields

Метод возвращает список полей PivotTablePageField из области фильтров.

Пример:

```
PivotTablePageFields pageFields = pivotTable.getPageFields();
PivotTablePageFields.PivotTablePageFieldsEnumerator enumerator =
pageFields.GetEnumerator();
while (enumerator.MoveNext())
{
    PivotTablePageField pivotTableCategory = enumerator.Current;
    Console.WriteLine(pivotTableCategory.fieldProperties.fieldAlias);
    Console.WriteLine(pivotTableCategory.fieldProperties.subtotalAlias);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(pivotTableCategory.fieldProperties.fieldName);
}
```

5.79.16 Метод PivotTable::getFieldCategories

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

Пример:

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

5.79.17 Метод PivotTable::getFieldItems

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

Пример:

```
PivotTableItems pivotTableItems = pivotTable.getFieldItems("Age");
PivotTableItemsEnumerator enumerator = pivotTableItems.GetEnumerator();
foreach (var pivotTableItem in enumerator)
{
    Console.WriteLine(pivotTableItem.getAlias());
}
```

5.79.18 Метод PivotTable::getFieldItemsByName

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

Пример:

```
PivotTableItems itemsByName = pivotTable.getFieldItemsByName("Ultimate Question
of Life", "42");
PivotTableItemsEnumerator enumerator = itemsByName.GetEnumerator();
foreach (var PivotTableItem in enumerator)
{
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(PivotTableItem.getName());
}
```

5.79.19 Метод PivotTable::getFilter

Метод возвращает фильтр PivotTableFilter по заданному имени поля fieldName.

Пример:

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

5.79.20 Метод PivotTable::getFilters

Метод возвращает список фильтров PivotTableFilter сводной таблицы.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    Console.WriteLine(pivotTableFilter.isHidden(0));
}
```

5.79.21 Метод PivotTable::update

Метод обновляет и полностью пересчитывает сводную таблицу, возвращает PivotTableUpdateResult.

Пример:

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

5.79.22 Метод PivotTable::createPivotTableEditor

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor.addField("Age", PivotTableFieldCategory.Rows);
pivotTableEditor.apply();
```

5.80 Класс PivotTableCaptions

Класс `PivotTableCaptions` хранит все пользовательские заголовки сводной таблицы (см. `PivotTable::getPivotTableCaptions()`). Описание полей таблицы представлено в таблице 39.

Таблица 39 – Описание полей класса `PivotTableCaptions`

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

5.81 Класс PivotTableLayoutSettings

Класс `PivotTableLayoutSettings` содержит настройки отображения сводной таблицы. Данный объект может быть получен в результате вызова `PivotTable::getPivotTableLayoutSettings()` и установлен методом `PivotTableEditor::setLayoutSettings()`. Описание полей таблицы представлено в таблице 40.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 40 – Описание полей класса PivotTableLayoutSettings

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

5.82 Класс PivotTableReportLayout

Класс PivotTableReportLayout описывает внешний вид отчетов сводной таблицы. Является полем класса PivotTableLayoutSettings. Описание полей представлено в таблице 41.

Таблица 41 – Описание полей класса PivotTableReportLayout

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.83 Класс PageFieldOrder

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

Таблица 42 – Описание полей класса `PageFieldOrder`

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

5.84 Класс PivotTableUnsupportedFeature

Класс `PivotTableUnsupportedFeature` описывает неподдерживаемую функциональность сводных таблиц. Получение неподдерживаемой функциональности сводных таблиц описано в `PivotTable::getUnsupportedFeatures()`. Описание полей класса представлено в таблице 43.

Таблица 43 – Описание полей класса `PivotTableUnsupportedFeature`

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

5.85 Класс PivotTableFieldCategories

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

5.85.1 Метод PivotTableFieldCategories::getEnumerator

Метод для перечисления категорий поля `PivotTableFieldCategory`.

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
    PivotTableFieldCategories categories = pivotTable.getFieldCategories("Age");
    PivotTableFieldCategorysEnumerator enumerator = categories.GetEnumerator();
    foreach (var pivotTableFieldCategory in enumerator)
    {
        Console.WriteLine(pivotTableFieldCategory);
    }
}
```

5.86 Класс PivotTableFunction

Класс `PivotTableFunction` описывает функции, которые могут быть использованы в сводных таблицах. Описание полей представлено в таблице 44. Используется в качестве поля `subtotalFunctions` класса `PivotTableCategoryField`.

Таблица 44 – Описание полей класса `PivotTableFunction`

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

5.87 Класс PivotTableFilters

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

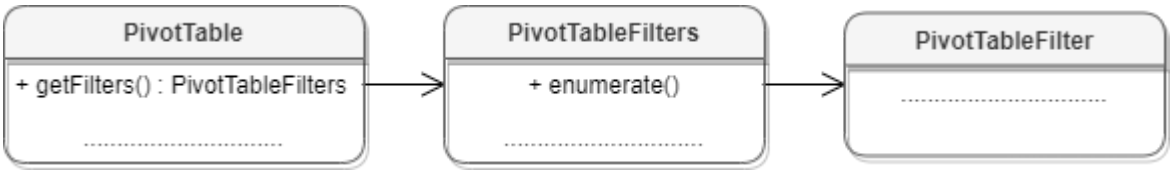


Рисунок 33 – Объектная модель классов для работы с фильтрами

5.87.1 Метод PivotTableFilters::getEnumerator

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

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
    PivotTableFilters pivotTableFilters = pivotTable.getFilters();
    PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
    foreach (var pivotTableFilter in enumerator)
    {
        Console.WriteLine(pivotTableFilter.getFieldName());
    }
}
```

5.88 Класс PivotTableField

Класс `PivotTableField` содержит свойства полей сводной таблицы (см. Таблицу 45). Объект может быть получен посредством вызова `PivotTable::getFieldsList()`.

Таблица 45 – Описание полей класса `PivotTableField`

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

5.89 Класс PivotTableFilter

Позволяет осуществить доступ к списку фильтров таблицы, каждый из которых обладает свойством видимости. При любом изменении фильтров они должны быть

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

применены к сводной таблице посредством использования методов PivotTableEditor::setFilter(), PivotTableEditor::setFilters().

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
    PivotTableFilters pivotTableFilters = pivotTable.getFilters();
    PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
    foreach (var pivotTableFilter in enumerator)
    {
        for (uint i = 0; i < pivotTableFilter.getCount(); i++)
        {
            pivotTableFilter.setHidden(i, false);
        }
    }
    PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
    pivotTableEditor.setFilters(pivotTableFilters);
    pivotTableEditor.apply();
}
```

5.89.1 Метод PivotTableFilter::getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    Console.WriteLine(pivotTableFilter.getFieldName());
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.89.2 Метод PivotTableFilter::getCount

Возвращает количество фильтруемых полей.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    Console.WriteLine(pivotTableFilter.getCount());
}
```

5.89.3 Метод PivotTableFilter::getName

Возвращает имя поля для заданного индекса фильтра.

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    for (uint i = 0; i < pivotTableFilter.getCount(); i++)
    {
        Console.WriteLine(pivotTableFilter.getName(i));
    }
}
```

5.89.4 Метод PivotTableFilter::isHidden

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

Пример:

```
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator enumerator = pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in enumerator)
{
    for (uint i = 0; i < pivotTableFilter.getCount(); i++)
    {
        Console.WriteLine(pivotTableFilter.isHidden(i));
    }
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.89.5 Метод PivotTableFilter::setHidden

Устанавливает видимость поля для заданного индекса, используя параметры:

– `itemName` – индекс поля;

– `hidden` – видимость (`true` – поле скрыто).

Пример:

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

5.90 Класс PivotTableFields

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

5.90.1 Метод PivotTableFields::getEnumerator

Используется для перечисления полей сводной таблицы.

Пример:

```
PivotTableFields pivotTableFields = pivotTable.getFieldsList();
PivotTableFields.PivotTableFieldsEnumerator enumerator =
pivotTableFields.GetEnumerator();
while (enumerator.MoveNext()) {
    PivotTableField pivotTableField = enumerator.Current;
    Console.WriteLine(pivotTableField.customFormula);
}
```

5.91 Класс PivotTableFieldProperties

`PivotTableFieldProperties` содержит свойства поля `PivotTableField` сводной таблицы (см. Таблицу 46).

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 46 – Описание полей класса PivotTableFieldProperties

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

5.92 Класс PivotTableCategoryField

PivotTableCategoryField содержит свойства поля сводной таблицы, использующегося как строка / столбец (см. Таблицу 47). Объект может быть получен посредством вызовов PivotTable::getRowFields(), PivotTable::getColumnFields().

Таблица 47 – Описание полей PivotTableCategoryField

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

5.93 Класс PivotTableValueField

PivotTableValueField содержит свойства поля сводной таблицы, использующегося как значение столбец (см. Таблицу 48). Таблица может быть получена посредством вызова PivotTable::getValueFields().

Таблица 48 – Описание полей класса PivotTableValueField

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Поле	Описание
PivotTableValueField.customFormula	Вычисляемая формула для поля значений, тип - строка.

5.94 Класс PivotTablePageField

Содержит свойства поля из области фильтров (см. Таблицу 49). Объект может быть получен посредством вызова PivotTable::getPageFields().

Таблица 49 – Описание полей класса PivotTablePageField

Поле	Описание
PivotTablePageField.fieldProperties	Свойства поля PivotTableFieldProperties

5.95 Класс PivotTableItems

Класс обеспечивает доступ к списку элементов сводной таблицы PivotTable (см. Рисунок 34).

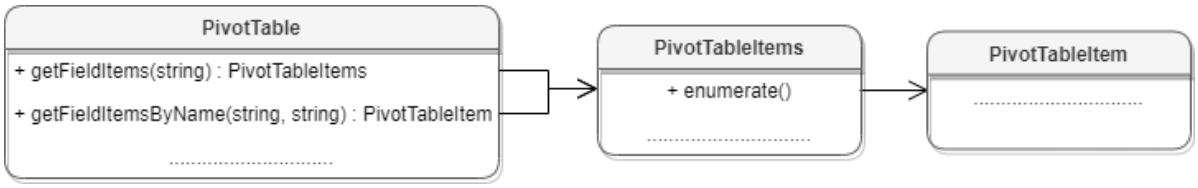


Рисунок 34 – Объектная модель классов для работы с элементами сводных таблиц

5.95.1 Метод PivotTableItems::GetEnumerator

Используется для перечисления элементов сводной таблицы.

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
    PivotTableItems pivotTableItems = pivotTable.getFieldItems("Age");
    PivotTableItemsEnumerator pivotTableItemsEnumerator =
pivotTableItems.GetEnumerator();
    foreach (var pivotTableItem in pivotTableItemsEnumerator)
    {
        Console.WriteLine(pivotTableItem.getAlias());
        Console.WriteLine(pivotTableItem.getName());
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
        Console.WriteLine(pivotTableItem.getItemType());
        Console.WriteLine(pivotTableItem.isCollapsed());
    }
}
```

5.96 Класс PivotTableItem

`PivotTableItem` описывает элемент сводной таблицы (см. Рисунок 35). См. пример в главе `PivotTableItems::GetEnumerator()`.

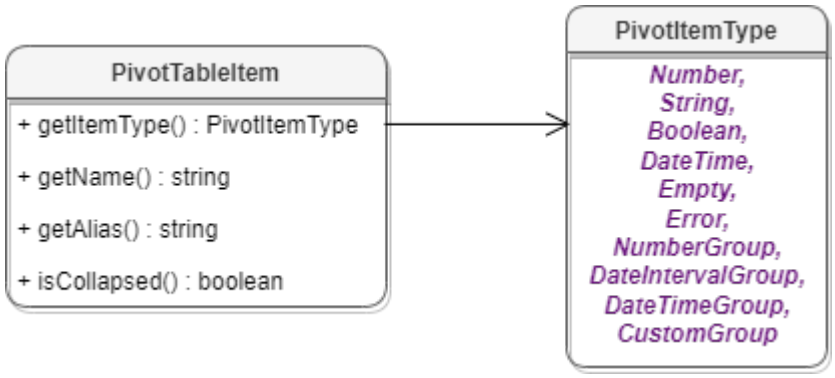


Рисунок 35 – Класс `PivotTableItem`

5.96.1 Метод PivotTableItem::getName

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

5.96.2 Метод PivotTableItem::getAlias

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

5.96.3 Метод PivotTableItem::getItemType

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

5.96.4 Метод PivotTableItem::isCollapsed

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.97 Класс PivotTableItemType

Класс `PivotTableItemType` содержит возможные типы элементов сводной таблицы. Описание полей представлено в таблице 50.

Таблица 50 – Описание полей класса `PivotTableItemType`

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

Пример:

```
Table sheet = document.getBlocks().getTable("Лист1");
Cell cell = sheet.getCell("A1");
PivotTable pivotTable = cell.getPivotTable();
if (pivotTable != null)
{
    PivotTableItems pivotTableItems = pivotTable.getFieldItems("Age");
    PivotTableItemsEnumerator pivotTableItemsEnumerator =
pivotTableItems.GetEnumerator();
    foreach (var pivotTableItem in pivotTableItemsEnumerator)
    {
        Console.WriteLine(pivotTableItem.getItemType());
    }
}
```

5.98 Класс PivotTableEditor

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.98.1 Метод PivotTableEditor::addField

Метод добавляет новое поле в сводную таблицу, используя параметры:

– `fieldName` - имя поля;

– `toCategory` - категория поля (тип - `PivotTableFieldCategory`);

– `index` - позиция в категории. Метод возвращает объект `PivotTableEditor`.

Пример:

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

5.98.2 Метод PivotTableEditor::moveField

Метод перемещает поле между категориями, используя параметры:

– `fieldName` - имя поля;

– `toCategory` - область, в которую перемещается поле (тип - `PivotTableFieldCategory`);

– `index` - позиция в новой категориии. Метод возвращает объект `PivotTableEditor`.

Пример:

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

5.98.3 Метод PivotTableEditor::removeField

Метод удаляет поле из категории, используя параметры:

– `fieldName` - имя поля;

– `fromCategory` - область, из которой удаляется поле (тип - `PivotTableFieldCategory`).

Метод возвращает объект `PivotTableEditor`.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

5.98.4 Метод PivotTableEditor::reorderField

Метод изменяет позицию поля в пределах категории, используя параметры:

– `fieldName` - имя поля;

– `category` - область (тип - `PivotTableFieldCategory`);

– `toIndex` - новая позиция поля.

Метод возвращает объект `PivotTableEditor`.

Пример:

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

5.98.5 Метод PivotTableEditor::enableField

Метод добавляет поле в область, зависящую от типа поля. Параметр `fieldName` - имя поля. Метод возвращает объект `PivotTableEditor`.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.enableField("Age");
pivotTableEditor.apply();
```

5.98.6 Метод PivotTableEditor::disableField

Метод удаляет поле из всех областей. Параметр `fieldName` - имя поля (тип - строка). Метод возвращает объект `PivotTableEditor`.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.disableField("Age");
pivotTableEditor.apply();
```

5.98.7 Метод PivotTableEditor::setSummarizeFunction

Метод задает суммирующую функцию для поля из области значений, используя параметры:

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

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

Метод возвращает объект `PivotTableEditor`.

Пример:

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

5.98.8 Метод PivotTableEditor::setFilter

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

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator pivotTableItemsEnumerator =
pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in pivotTableItemsEnumerator)
{
    uint filterSize = pivotTableFilter.getCount();
    for (uint i = 0; i < filterSize; i++)
    {
        pivotTableFilter.setHidden(i, true);
    }
    pivotTableEditor.setFilter(pivotTableFilter);
}
PivotTableUpdateResult pivotTableUpdateResult = pivotTableEditor.apply();
```

5.98.9 Метод PivotTableEditor::setFilters

Метод задает фильтры `PivotTableFilters` сводной таблицы. Если какой-то из фильтров не может быть применен, он пропускается. Метод возвращает объект `PivotTableEditor`.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
PivotTableFilters pivotTableFilters = pivotTable.getFilters();
PivotTableFiltersEnumerator pivotTableItemsEnumerator =
pivotTableFilters.GetEnumerator();
foreach (var pivotTableFilter in pivotTableItemsEnumerator)
{
    uint filterSize = pivotTableFilter.getCount();
    for (uint i = 0; i < filterSize; i++)
    {
        pivotTableFilter.setHidden(i, true);
    }
    pivotTableEditor.setFilter(pivotTableFilter);
}
pivotTableEditor.setFilters(pivotTableFilters);
PivotTableUpdateResult pivotTableUpdateResult = pivotTableEditor.apply();
```

5.98.10 Метод PivotTableEditor::setCaptions

Метод задает заголовки сводной таблицы PivotTableCaptions, возвращает объект PivotTableEditor.

Пример:

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

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

5.98.11 Метод PivotTableEditor::setLayoutSettings

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

5.98.12 Метод PivotTableEditor::setGrandTotalSettings

Метод задает настройки отображения общего итога. Параметры: `isRowGrandTotalEnabled` – показывать общие итоги для строк, `isColGrandTotalEnabled` – показывать общие итоги для столбцов.

Пример:

```
PivotTableEditor pivotTableEditor = pivotTable.createPivotTableEditor();
pivotTableEditor = pivotTableEditor.setGrandTotalSettings(true, true);
pivotTableEditor.apply();
```

5.98.13 Метод PivotTableEditor::apply

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

Пример:

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

5.99 Класс PivotTableUpdateResult

В таблице 51 приведены константы, которые соответствуют возможным результатам обновления сводной таблицы (см. методы `PivotTable::update()`, `PivotTableEditor:apply()`).

Таблица 51 – Результаты обновления сводной таблицы

Наименование константы	Описание
`PivotTableUpdateResult.Success`	Успешное обновление таблицы
`PivotTableUpdateResult.NoPivotTable`	Сводная таблица не найдена
`PivotTableUpdateResult.NoSuchFieldInCategory`	Не найдено поле в категории
`PivotTableUpdateResult.NoSuchFieldInPivotTable`	Не найдено поле в сводной таблице

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Наименование константы	Описание
PivotTableUpdateResult.InvalidIndex	Ошибка в индексе
PivotTableUpdateResult.FieldAlreadyEnabled	Поле уже существует
PivotTableUpdateResult.MovingFieldToTheSameCategoryForbidden	Попытка перемещения поля в рамках текущей категории
PivotTableUpdateResult.InvalidFunction	Неправильная функция
PivotTableUpdateResult.InvalidCategory	Неправильная область
PivotTableUpdateResult.InvalidDataSourceRange	Ошибка диапазона исходных данных
PivotTableUpdateResult.NoDataRowsInDataSource	В исходных данных нет строк с данными
PivotTableUpdateResult.EmptyDataSourceHeaders	Пустые заголовки исходных данных
PivotTableUpdateResult.NoReferenceUnderDefine	Попытка обновить или создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу
PivotTableUpdateResult.NoSuchItem	Элемент не найден
PivotTableUpdateResult.CannotExpandCollapseLeafItem	Не удается раскрыть свернутый элемент
PivotTableUpdateResult.AnotherPivotInsideDataSource	Найдена другая сводная таблица в этом же диапазоне
PivotTableUpdateResult.Canceled	Обновление сводной таблицы отменено

5.100 Класс PivotTableFieldCategory

Класс `PivotTableFieldCategory` описывает флаги, которые задают категорию области полей. Описание полей представлено в таблице 52. Пример использования см. в `PivotTable::getFieldCategories()`.

Таблица 52 – Описание полей класса `PivotTableFieldCategory`

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.101 Класс PointU

Класс `PointU` представляет собой точку в двухмерном пространстве.

Список свойств:

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

Примеры:

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

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

5.102 Класс Position

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

5.102.1 Метод Position:insertText

Метод предназначен для вставки текстовой строки в заданное местоположение в документе.

Пример:

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

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

5.102.2 Метод Position:insertTable

Метод предназначен для вставки таблицы в текстовый документ или страницы в табличный документ. В качестве параметров передаются число строк, столбцов, а также имя таблицы. Метод возвращает объект таблицы.

```
Table insertTable(int rows, int cols, String tableName);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

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

приведет к созданию таблицы с именем «Table1».

Примеры добавления таблицы в текстовый документ приведены в разделе Работа с таблицами текстового документа.

Примеры добавления страницы в текстовый документ приведены в разделе Работа с листами табличного документа.

5.102.3 Метод Position:insertPageBreak

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

Пример:

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

5.102.4 Метод Position:insertLineBreak

Метод предназначен для вставки перевода строки.

Пример:

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

5.102.5 Метод Position:insertBookmark

Вставляет закладку с наименованием в текущую позицию.

Пример:

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

5.102.6 Метод Position:insertSectionBreak

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Примеры:

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

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

5.102.7 Метод Position:insertImage

Вставляет изображение из файла в текущую позицию.

Параметры:

– `url` – полный путь к файлу, строка;

– `size` – геометрические размеры изображения для вставки, тип `SizeU`.

Пример:

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

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

5.102.8 Метод Position:removeBackward

Метод удаляет заданное количество объектов (символов, картинок и т.д.) до текущей позиции.

Пример:

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

5.102.9 Метод Position:removeForward

Метод удаляет заданное количество объектов (символов, картинок и т.д.) после текущей позиции.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

5.103 Класс PrintingScope

Класс `PrintingScope` содержит настройки для экспорта табличных документов. Используется в поле `printingScope` класса `WorkbookExportSettings`.

Позволяет создать области печати следующих типов:

- выбранная область печати либо весь документ (см. `PrintingScope::Type`);
- указанный диапазон ячеек (см. `CellRangePosition`).

Примеры:

```
-- по умолчанию - PrintingScope.Type.PrintArea
PrintingScope printingScope = new PrintingScope();
```

```
-- область печати
PrintingScope printingScope = new PrintingScope(PrintingScope.Type.PrintArea);
```

```
-- диапазон ячеек
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
PrintingScope printingScope = new PrintingScope(cellRangePosition);
```

5.103.1 Метод PrintingScope::getCellRange

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

5.103.2 Метод PrintingScope::usePrintArea

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

5.103.3 Тип PrintingScope.Type

Варианты выбора диапазона страниц для экспорта и печати представлены в таблице 53. Используется в `PrintingScope`

Таблица 53 – Диапазон страниц для экспорта и печати

Наименование константы	Описание
`PrintingScope.Type.PrintArea`	Выбранная область печати (по умолчанию)

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Наименование константы	Описание
`PrintingScope.Type.WholeSheet`	Печать всего документа

5.104 Класс Range

Класс `Range` предоставляет доступ к диапазону документа. На рисунке 36 изображена объектная модель классов, относящихся к работе с диапазонами.

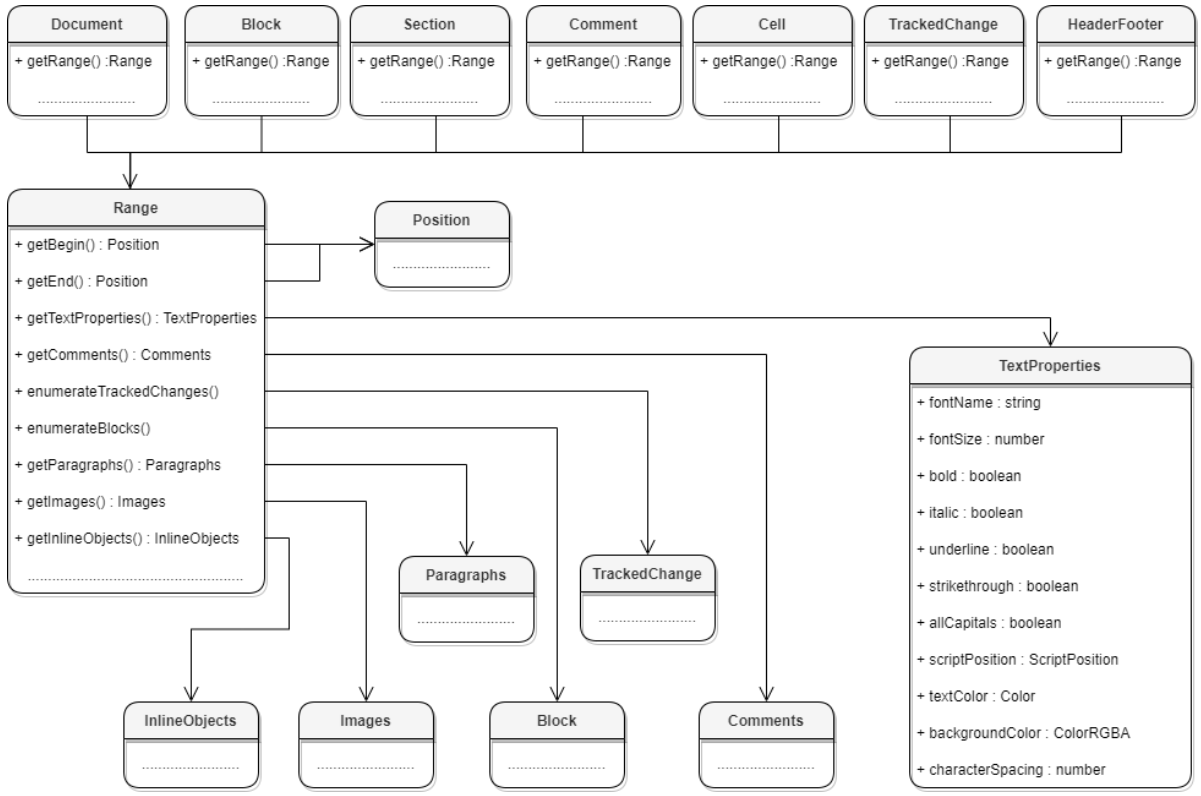


Рисунок 36 – Объектная модель для работы с классом `Range`

Варианты получения диапазона для текстового документа:

```
-- диапазон всего документа
Range range = document.getRange();

-- диапазон блока
Block block = document.getBlocks().getBlock(0);
if (block != null) {
    Range blockRange = block.getRange();
}

-- диапазон секций
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
foreach (var section in sectionsEnumerator)
{
    Console.WriteLine(section.getRange().extractText());
}

-- диапазон комментариев
Comments comments = document.getRange().getComments();
CommentsEnumerator commentsEnumerator = comments.GetEnumerator();
foreach (var comment in commentsEnumerator)
{
    Console.WriteLine(comment.getRange().extractText());
}

-- диапазон ячейки
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
}

-- диапазон верхних колонтитулов
Block block = document.getBlocks().getBlock(0);
if (block != null)
{
    Section section = block.getSection();
    HeadersFooters headers = section.getHeaders();
    HeaderFootersEnumerator headerFootersEnumerator = headers.GetEnumerator();
    foreach (var headerFooter in headerFootersEnumerator)
    {
        Console.WriteLine(headerFooter.getRange().extractText());
    }
}

-- диапазон отслеживаемых изменений
var trackedChangesEnumerator =
document.getRange().getTrackedChangesEnumerator();
foreach (var trackedChange in trackedChangesEnumerator)
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
{
    Console.WriteLine(trackedChange.getType().ToString());
    Console.WriteLine(trackedChange.getInfo().author.name);
    Console.WriteLine(trackedChange.getRange().extractText());
}
```

5.104.1 Метод Range::getBegin

Метод возвращает позицию в начале диапазона.

Пример для текстового документа:

```
Position beginDocPosition = document.getRange().getBegin();
beginDocPosition.insertText("API");
```

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    Position beginDocPos = cellRange.getBegin();
    beginDocPos.insertText("API");
}
```

5.104.2 Метод Range::getEnd

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

Пример для текстового документа:

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

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    Position endDocPos = cellRange.getEnd();
    endDocPos.insertText("API");
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.104.3 Метод Range::extractText

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

Пример для текстового документа:

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

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    Console.WriteLine(cellRange.ExtractText());
}
```

5.104.4 Метод Range::removeContent

Метод полностью удаляет содержимое диапазона.

Пример для текстового документа:

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

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    cellRange.removeContent();
    Console.WriteLine(cellRange.ExtractText());
}
```

5.104.5 Метод Range::lockContent

Метод запрещает изменения содержимого диапазона.

Внимание ! Метод может быть использован только в текстовых документах.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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());
}
```

5.104.6 Метод Range::unlockContent

Метод разрешает изменения содержимого диапазона.

Внимание ! Метод может быть использован только в текстовых документах.

Пример для текстового документа:

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

Пример для таблицы внутри текстового документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    cellRange.unlockContent();
    Console.WriteLine(cellRange.isContentLocked());
}
```

5.104.7 Метод Range::isContentLocked

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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());
}
```

5.104.8 Метод Range::replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример для текстового документа:

```
Range range = document.getRange();
range.replaceText("Range text");
```

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    cellRange.replaceText("New text");
}
```

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

5.104.9 Метод Range::getTextProperties

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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);
}
```

5.104.10 Метод Range::setTextProperties

Метод применяет настройки форматирования TextProperties для диапазона.

Пример для текстового документа:

```
Range range = document.getRange();
TextProperties textProperties = range.getTextProperties();
textProperties.fontName = "Arial";
range.setTextProperties(textProperties);
```

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null) {
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    TextProperties textProperties = cellRange.getTextProperties();
    textProperties.fontName = "Arial";
    cellRange.setTextProperties(textProperties);
}
```

5.104.11 Метод Range::getBlocksEnumerator

Предоставляет возможность итерации по блокам.

Пример для текстового документа:

```
Range range = document.getRange();
BlocksEnumerator blocksEnumerator = range.getBlocksEnumerator();
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
foreach (var block in blocksEnumerator)
{
    Console.WriteLine(block.getRange().extractText());
}
```

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    Cell cell = table.getCell("B2");
    Range cellRange = cell.getRange();
    BlocksEnumerator blocksEnumerator = cellRange.getBlocksEnumerator();
    foreach (var block in blocksEnumerator)
    {
        Console.WriteLine(block.getRange().extractText());
    }
}
```

5.104.12 Метод Range::getTrackedChangesEnumerator

Предоставляет возможность итерации по отслеживаемым изменениям TrackedChange. Метод может быть использован только в текстовых документах.

Пример:

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

5.104.13 Метод Range::getComments

Обеспечивает доступ к комментариям в диапазоне.

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

Пример:

```
var commentsEnumerator = document.getRange().getComments().GetEnumerator();
foreach (var comment in commentsEnumerator)
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
{
    Console.WriteLine(comment.getText());
    Console.WriteLine(comment.getInfo().author.name);
    Console.WriteLine(comment.getRange().extractText());
}
```

5.104.14 Метод Range::getParagraphs

Обеспечивает доступ к абзацам Paragraphs в диапазоне.

Пример для текстового документа:

```
Paragraphs paragraphs = document.getRange().getParagraphs();
ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
foreach (var paragraph in paragraphsEnumerator)
{
    Console.WriteLine(paragraph.getRange().extractText());
}
```

Пример для табличного документа:

```
Table table = document.getBlocks().getTable(0);
if (table != null)
{
    Cell cell = table.getCell("B2");
    Range range = cell.getRange();
    Paragraphs paragraphs = range.getParagraphs();
    ParagraphsEnumerator paragraphsEnumerator = paragraphs.GetEnumerator();
    foreach (var paragraph in paragraphsEnumerator)
    {
        Console.WriteLine(paragraph.getRange().extractText());
    }
}
```

5.104.15 Метод Range::getImages

Обеспечивает доступ к изображениям (Image) в диапазоне.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(image.getFrame().getWrapType());
}
```

5.104.16 Метод Range::getInlineObjects

Обеспечивает доступ к перечислению `InlineObjects` графических объектов диапазона.

Пример:

```
Range range = document.getRange();
InlineObjects inlineObjects = range.getInlineObjects();
InlineObjectsEnumerator inlineObjectsEnumerator = inlineObjects.GetEnumerator();
foreach (var inlineObject in inlineObjectsEnumerator)
{
    Console.WriteLine(inlineObject.getFrame().getWrapType());
}
```

5.105 Класс RangeBorders

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

5.106 Класс RectU

Класс `RectU` описывает прямоугольник в двухмерном пространстве.

Список свойств:

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

Примеры:

```
RectU rect = new RectU(new PointU(50, 50), new PointU(50, 50));
```

```
RectU rect = new RectU();
rect.topLeft = new PointU(50, 50);
rect.bottomRight = new PointU(50, 50);
```

5.107 Класс SaveDocumentSettings

Класс `SaveDocumentSettings` предоставляет настройки, используемые для сохранения документа в файл (см. `Document::saveAs()`). Описание полей класса `SaveDocumentSettings` представлено в таблице 54.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 54 – Описание полей класса `SaveDocumentSettings`

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

5.108 Класс Script

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

5.108.1 Метод Script::getName

Метод возвращает имя макрокоманды.

Пример:

```
ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
foreach (var script in scriptsEnumerator)
{
    Console.WriteLine(script.getName());
}
```

5.108.2 Метод Script::setName

Метод устанавливает имя для макрокоманды.

Пример:

```
ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
foreach (var script in scriptsEnumerator)
{
    script.setName("Script name")
    Console.WriteLine(script.getName());
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.108.3 Метод Script::getBody

Метод возвращает текст макрокоманды в виде строки.

Пример:

```
ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
foreach (var script in scriptsEnumerator)
{
    Console.WriteLine(script.getBody());
}
```

5.108.4 Метод Script::setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

Пример:

```
foreach (var script in scriptsEnumerator)
{
    script.setName("local scripts = document:getScripts()\nfor script in scripts
: enumerate() do\nprint(script:getName())\nend")
    Console.WriteLine(script.getName())
}
```

5.109 Класс ScriptPosition

Варианты представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлены в таблице 55. Используется в качестве поля `scriptPosition` класса `TextProperties`.

Таблица 55 – Типы надстрочного и подстрочного форматирования

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.110 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());
    }
}
```

5.110.1 Метод Scripts::getScript

Метод возвращает объект класса `Script`, описывающий макрокоманду. В качестве аргумента используется имя макрокоманды.

Пример:

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    Script script = scripts.getScript("ScriptName");
    Console.WriteLine(script.getName());
}
```

5.110.2 Метод Scripts::setScript

Метод добавляет макрокоманду в текущий документ. Если макрокоманда с таким именем уже существует, будет обновлено ее содержимое.

Пример:

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    String scriptName = "Enumerate scripts for document";
    String scriptCode = "local scripts = document:getScripts()\nfor script in
scripts:enumerate() do\nprint(script:getName())\nend";
    scripts.setScript(scriptName, scriptCode);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Script script = scripts.getScript(scriptName);
}
```

5.110.3 Метод Scripts::removeScript

Метод удаляет макрокоманду из текущего документа. В качестве аргумента используется имя макрокоманды.

Пример:

```
Scripts scripts = document.getScripts();
if (scripts != null) {
    String scriptName = "Enumerate scripts for document";
    scripts.removeScript(scriptName);
    Script script = scripts.getScript(scriptName);
    if (script == null) {
        Console.WriteLine("Script was removed");
    }
}
```

5.110.4 Метод Scripts::GetEnumerator

Метод возвращает коллекцию макрокоманд для их дальнейшего перечисления.

Пример:

```
Scripts scripts = document.getScripts();
if (scripts != null)
{
    ScriptsEnumerator scriptsEnumerator = scripts.GetEnumerator();
    foreach (var script in scriptsEnumerator)
    {
        Console.WriteLine(script.getName());
    }
}
```

5.111 Класс Search

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.111.1 Метод Search::findText

Метод выполняет поиск строки без учета регистра во всем документе или выбранном диапазоне документа. Результат возвращается в виде диапазона `Range`, содержащего искомый фрагмент.

Существует два варианта реализации метода.

`Range findText(string searchText)`: используется для поиска в текстовом документе.

`Range findText(string searchText, Range range)`: используется для поиска в текстовом и табличном документе.

Параметры:

- `searchText` – строка для поиска;
- `range` – диапазон поиска, тип `Range`.

Если строка не обнаружена, возвращается пустой диапазон.

Примеры использования метода `findText` приведены в разделах Поиск в текстовом документе и Поиск в табличном документе.

5.112 Класс Section

Класс `Section` представляет собой раздел в документе.

5.112.1 Метод Section::setPageProperties

Метод устанавливает параметры `PageProperties` страниц, находящихся в разделе.

Пример:

```
Block block = document.getBlocks().getBlock(0);
if (block != null)
{
    Section section = block.getSection();
    PageProperties pageProperties = section.getPageProperties();
    pageProperties.height = 100;
    pageProperties.width = 200;
    section.setPageProperties(pageProperties);
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.112.2 Метод Section::getPageProperties

Метод возвращает параметры страниц раздела PageProperties.

Пример:

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

5.112.3 Метод Section::setPageOrientation

Метод задает ориентацию страниц раздела типа PageOrientation.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    section.setPageOrientation(PageOrientation.Portrait);
    Console.WriteLine(section.getPageOrientation());
}
```

5.112.4 Метод Section::getPageOrientation

Метод возвращает ориентацию страниц раздела типа PageOrientation.

Пример:

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

5.112.5 Метод Section::getRange

Метод возвращает диапазон Range в документе, соответствующий данному разделу.

Пример:

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    Range range = section.GetRange();
    Console.WriteLine(range.extractText());
}
```

5.112.6 Метод Section::getHeaders

Метод возвращает коллекцию HeadersFooters верхних колонтитулов данного раздела.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    HeadersFooters headers = section.getHeaders();
    Console.WriteLine(headers.getType());
}
```

5.112.7 Метод Section::getFooters

Метод возвращает коллекцию HeadersFooters нижних колонтитулов данного раздела.

Пример:

```
Sections sections = document.getSections();
SectionsEnumerator sectionsEnumerator = sections.GetEnumerator();
foreach (var section in sectionsEnumerator)
{
    HeadersFooters footers = section.getFooters();
    Console.WriteLine(footers.getType());
}
```

5.113 Класс SectionBreakType

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

Описание полей класса SectionBreakType представлено в таблице 56.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 56 – Описание полей класса `SectionBreakType`

Поле	Описание
`SectionBreakType.NextPage`	Следующий раздел начинается с новой страницы
`SectionBreakType.Continious`	Следущий раздел продолжается на текущей странице без вставки разрыва страницы
`SectionBreakType.EvenPage`	Следующий раздел начинается на ближайшей четной странице
`SectionBreakType.OddPage`	Следующий раздел начинается на ближайшей нечетной странице

5.114 Класс Sections

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

5.114.1 Метод Sections::getEnumerator

Метод позволяет перечислить коллекцию секций документа.

Пример:

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

5.115 Класс Shape

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

5.115.1 Метод Shape::getShapeProperties

Метод возвращает свойства фигуры `ShapeProperties`.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    Console.WriteLine(shapeProperties.verticalAlignment);
}
```

5.115.2 Метод Shape::setShapeProperties

Метод устанавливает свойства фигуры ShapeProperties.

Пример:

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

5.116 Класс ShapeProperties

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

Содержит следующие поля:

– verticalAlignment - вертикальное выравнивание, тип VerticalAlignment;

– borderProperties - свойства границ фигуры, тип LineProperties;

– fill - свойства заполнения фигуры, тип Fill;

– shapeTextLayout - свойства текста внутри фигуры, тип ShapeTextLayout.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.116.1 Поле ShapeProperties::borderProperties

Поле предназначено для установки свойств границ фигуры LineProperties.

Пример:

```
ShapeProperties shapeProperties = shape.getShapeProperties();
LineProperties borderProperties = new LineProperties();
borderProperties.style = LineStyle.Solid;
borderProperties.width = 1.5f;
borderProperties.color = new Color(new ColorRGBA(55, 146, 179, 200));
shapeProperties.borderProperties = borderProperties;
shape.setShapeProperties(shapeProperties);
```

5.116.2 Поле ShapeProperties::verticalAlignment

Поле предназначено для установки типа вертикального выравнивания VerticalAlignment.

Пример:

```
ShapeProperties shapeProperties = shape.getShapeProperties();
shapeProperties.verticalAlignment = VerticalAlignment.Center;
shape.setShapeProperties(shapeProperties);
```

5.116.3 Поле ShapeProperties::fill

Поле предназначено для установки свойств заполнения фигуры Fill.

Пример:

```
ShapeProperties shapeProperties = shape.getShapeProperties();

// Без заполнения
shapeProperties.fill = new Fill();
// Заполнение цветом
shapeProperties.fill = new Fill(new Color(new ColorRGBA(55, 146, 179, 200)));
// Заполнение шаблоном из url
shapeProperties.fill = new Fill("https://fillpattern.url");

shape.setShapeProperties(shapeProperties);
```

5.116.4 Поле ShapeProperties::shapeTextLayout

Поле предназначено для установки свойств текста внутри фигуры, тип -

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

ShapeTextLayout.

Пример:

```
ShapeProperties shapeProperties = shape.getShapeProperties();
shapeProperties.shapeTextLayout = ShapeTextLayout.FitTextToShape;
shape.setShapeProperties(shapeProperties);
```

5.117 Класс ShapeTextLayout

Класс ShapeTextLayout описывает свойства текста, находящегося внутри фигуры. Описание полей представлено в таблице 57. Используется в качестве поля shapeTextLayout класса ShapeProperties.

Таблица 57 – Описание полей класса ShapeTextLayout

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

5.118 Класс SizeU

Класс SizeU описывает размер объекта в двухмерном пространстве.

Список свойств:

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

Примеры:

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

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

5.119 Класс Scripting

Объект класса Scripting может быть получен путем вызова DocumentAPI.createScripting(document), и содержит метод runScript, который используется для запуска макрокоманды.

Пример:

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
Scripting scripting = DocumentAPI.createScripting(document);
scripting.runScript("ScriptName");
```

5.119.1 Метод Scripting::runScript

Запускает макрокоманду с указанным именем. В случае невозможности запуска макрокоманды вызывает исключение `ScriptExecutionError`.

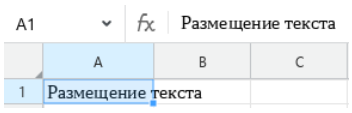
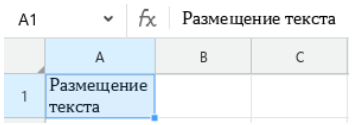
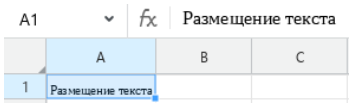
Пример:

```
Scripting scripting = DocumentAPI.createScripting(document);
scripting.runScript("ScriptName");
```

5.120 Класс TextLayout

В таблице 58 приведены варианты размещения текста в ячейках таблицы. Данное значение используется в поле `textLayout` класса `CellProperties`.

Таблица 58 – Варианты размещения текста в ячейках таблицы

Наименование константы	Описание	Отображение
`TextLayout.SingleLine`	Текст располагается в одну строку с наложением на соседние ячейки.	
`TextLayout.WrapByWords`	Текст внутри ячейки переносится по словам. Высота ряда увеличивается чтобы разместить текст полностью.	
`TextLayout.ShrinkSizeToFitWidth`	Текст располагается в одну линию, отображение масштабируется таким образом, чтобы полностью разместиться в ячейке без изменения ее размера. Размер шрифта не изменяется, данная настройка влияет только на отображение содержимого ячейки таблицы.	

Пример:

```
Table firstSheet = document.getBlocks().getTable(0);
Cell cell = firstSheet.getCell("A3");
CellProperties cellProps = cell.getCellProperties();
cellProps.textLayout = TextLayout.ShrinkSizeToFitWidth;
cell.setCellProperties(cellProps);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.121 Класс TextOrientation

Класс `TextOrientation` содержит угол поворота текста в ячейке, фигуре и т. д (см. `CellProperties`).

Пример:

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

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

5.121.1 Метод TextOrientation:getAngle

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

Пример:

```
CellProperties cellProps = cell.getCellProperties();
cellProps.textOrientation = new TextOrientation(45);
Console.WriteLine(cellProps.textOrientation.getAngle());
```

5.122 Класс TableRangeInfo

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

Описание полей класса `TableRangeInfo` представлено в таблице 59.

Таблица 59 – Поля класса `TableRangeInfo`

Поле	Тип	Описание
`tableRange`	`CellRangePosition`	Диапазон ячеек

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.123 Класс Table

Класс `Table` предоставляет доступ к листу в табличном документе или таблице в составе текстового документа (см. Рисунок 37).

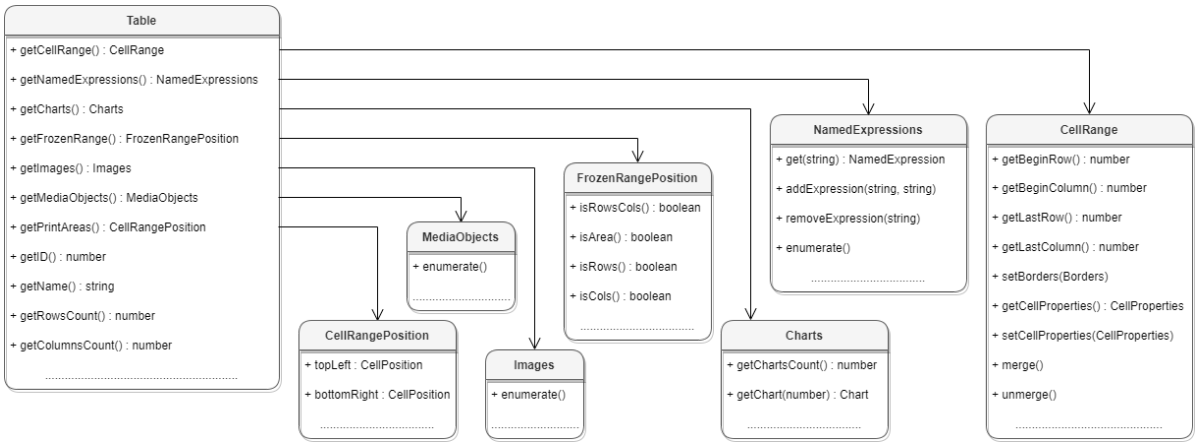


Рисунок 37 – Структура полей класса `Table`

5.123.1 Метод Table::setName

Метод задает имя таблицы. В случае с табличным документом это текстовое значение будет являться именем страницы. В текстовых документах имя таблицы нигде не отображается, но в дальнейшем его можно использовать для доступа к таблице по имени.

Устанавливаемое значение должно быть уникальным.

Пример использования:

```
Table table = document.getBlocks().getTable("Лист1");
if (table != null)
{
    String newTableName = "Лист2";
    table.setName(newTableName);
    table = document.getBlocks().getTable(newTableName);
    if (table != null)
    {
        // Table was renamed
    }
}
```

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.123.2 Метод Table::getName

Метод позволяет получить имя листа табличного документа.

Пример:

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

5.123.3 Метод Table::getRowsCount

Метод позволяет получить количество строк таблицы.

Пример:

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

5.123.4 Метод Table::getColumnsCount

Метод позволяет получить количество столбцов таблицы.

Пример:

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

5.123.5 Метод Table::getCell

Метод позволяет получить доступ к отдельной ячейке таблицы. В качестве аргумента может выступать текстовое представление адреса ячейки, либо экземпляр класса CellPosition.

Примеры:

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

```
CellPosition cellPosition = new CellPosition(2, 1);
Cell cell = table.getCell(cellPosition);
Console.WriteLine(cell.getFormattedValue());
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.123.6 Метод Table::getCellRange

Метод позволяет получить доступ к диапазону ячеек класса `CellRange`. В качестве аргумента может использоваться строка, описывающая диапазон (`"B3:C4"`), либо объект типа `CellRangePosition`.

Примеры:

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

```
Table firstSheet = document.getBlocks().getTable("Лист1");
CellRangePosition cellRangePosition = new CellRangePosition(0, 0, 5, 5);
CellRange cellRange = firstSheet.getCellRange(cellRangePosition);
```

5.123.7 Метод Table::insertColumnAfter

Метод предназначен для вставки нового столбца после указанной позиции в таблице.

Вызов:

```
insertColumnAfter(columnIndex, copyColumnStyle, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца в таблице, после которого производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример:

```
// Создать в документе новую таблицу 2x2
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");
// Добавление двух столбцов в середину таблицы, без наследования настроек
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
форматирования
table.insertColumnAfter(0, false, 2);
```

5.123.8 Метод Table::insertColumnBefore

Метод предназначен для вставки нового столбца до указанной позиции в таблице.

Вызов:

```
insertColumnBefore(columnIndex, copyColumnStyle, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца в таблице, перед которым производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию 1.

Пример:

```
// Создать в документе новую таблицу 2x2
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");
// Добавление двух столбцов в середину таблицы, без наследования настроек
форматирования
table.insertColumnBefore(1, false, 2);
```

5.123.9 Метод Table::insertRowAfter

Метод предназначен для вставки новой строки после указанной позиции в таблице.

Вызов:

```
insertRowAfter(rowIndex, copyRowStyle, rowsCount)
```

Параметры:

- `rowIndex` – индекс строки в таблице, после которой производится вставка. Индексация строк начинается с нуля.
- `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

– `rowsCount` – количество вставляемых строк. Значение по умолчанию 1.

Пример:

```
// Создать в документе новую таблицу 2x2
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");

// Добавление двух строк в середину таблицы, без наследования настроек форматирования
table.insertRowAfter(0, false, 2);
```

5.123.10 Метод Table::insertRowBefore

Метод предназначен для вставки новой строки до указанной позиции в таблице.

Вызов:

```
insertRowBefore(rowIndex, copyRowStyle, rowsCount)
```

Параметры:

– `rowIndex` – индекс строки в таблице, перед которой производится вставка. Индексация строк начинается с нуля.

– `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение `true`, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение `false`, то настройки форматирования не копируются. Значение по умолчанию `true`.

– `rowsCount` – количество вставляемых строк. Значение по умолчанию 1.

Пример:

```
// Создать в документе новую таблицу 2x2
Table table = document.getRange().getBegin().insertTable(2, 2, "NewTable");

// Добавление двух строк в середину таблицы, без наследования настроек форматирования
table.insertRowBefore(1, false, 2);
```

5.123.11 Метод Table::removeColumn

Метод предназначен для удаления столбца таблицы, начиная с заданного индекса.

Вызов:

```
removeColumn(columnIndex, columnsCount)
```

Параметры:

– `columnIndex` – индекс столбца, начиная с которого будет удалено заданное количество столбцов. Индексация столбцов начинается с нуля.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

– `columnsCount` – количество столбцов для удаления, необязательный параметр. Значение по умолчанию 1.

5.123.12 Метод Table::removeRow

Метод предназначен для удаления строки таблицы, начиная с заданного индекса.

Вызов:

```
removeRow(rowIndex, rowsCount)
```

Параметры:

– `rowIndex` – индекс строки, начиная с которого будет удалено `rowsCount` строк. Индексация строк начинается с нуля.

– `rowsCount` – количество строк для удаления, необязательный параметр. Значение по умолчанию 1.

5.123.13 Метод Table:groupRows

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

Вызов:

```
groupRows(rowIndex, rowsCount)
```

Параметры:

– `rowIndex` – индекс строки, начиная с которого будет начата группировка строк;

– `rowsCount` – количество строк для группировки.

5.123.14 Метод Table:ungroupRows

Метод предназначен для разгруппировки строк таблицы, начиная с заданного индекса. Индексация строк начинается с нуля.

Вызов:

```
ungroupRows(rowIndex, rowsCount)
```

Параметры:

– `rowIndex` – индекс строки, начиная с которого будет начата разгруппировка строк;

– `rowsCount` – количество строк для разгруппировки.

5.123.15 Метод Table:clearRowGroups

Метод предназначен для очистки группированных строк таблицы, начиная с заданного индекса. Индексация строк начинается с нуля.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Вызов:

```
clearRowGroups(rowIndex, rowCount)
```

Параметры:

- `rowIndex` – индекс строки, начиная с которой будет начата очистка групп;
- `rowCount` – количество строк для очистки групп.

5.123.16 Метод Table:groupColumns

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

Вызов:

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

Параметры:

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

5.123.17 Метод Table:ungroupColumns

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

Вызов:

```
ungroupColumns(columnIndex, columnsCount)
```

Параметры:

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

5.123.18 Метод Table:clearColumnGroups

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

Вызов:

```
clearColumnGroups(columnIndex, columnsCount)
```

Параметры:

- `columnIndex` – индекс столбца, начиная с которого будет начата очистка групп;

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

– `columnsCount` – количество столбцов для очистки групп.

5.123.19 Метод Table:setColumnsVisible

Метод `Table::setColumnsVisible` позволяет задавать видимость столбцов, начиная с заданного индекса. Индексация столбцов начинается с нуля.

Вызов:

```
setColumnsVisible(first, columnsCount, visible)
```

Параметрs:

`first` – начальный индекс;

`columnsCount` – количество столбцов;

`visible` – видимость.

5.123.20 Метод Table:setRowsVisible

Метод `Table::setRowsVisible` позволяет задавать видимость строк, начиная с заданного индекса. Индексация строк начинается с нуля.

Вызов:

```
setRowsVisible(first, rowsCount, visible)
```

Параметрs:

`first` – начальный индекс;

`columnsCount` – количество строк;

`visible` – видимость.

5.123.21 Метод Table::setColumnWidth

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

Вызов:

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

Параметры:

– `columnIndex` – индекс столбца в таблице, для которого устанавливается значение ширины. Индексация столбцов начинается с нуля.

– `width` – ширина столбца в пунктах (1/72 дюйма).

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

5.123.22 Метод Table::setRowHeight

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

Вызов:

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

Параметры:

- `rowIndex` – индекс строки в таблице, для которой устанавливается значение высоты. Индексация строк начинается с нуля.
- `height` – высота строки в пунктах (1/72 дюйма).
- `rowHeightRule` – точность значения (`RowHeightRule.Exact` – точно, `RowHeightRule.AtLeast` – не меньше).

Пример:

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

// Установить высоту строки в 100 pt
table.setRowHeight(1, 100, RowHeightRule.Exact);
```

5.123.23 Метод Table::duplicate

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

Пример:

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

5.123.24 Метод Table::remove

Для удаления таблицы в текстовом документе или листа в табличном документе используется метод `remove()`.

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
    table.remove();
}
```

5.123.25 Метод Table::moveTo

Для перемещения листа таблицы по указанному индексу в табличном документе используется метод `moveTo`. Указанный индекс должен быть меньше или равен количеству листов в документе. Индексация листов начинается с нуля. Метод может быть использован только в табличном документе.

Пример:

```
Table table = document.getBlocks().getTable(0);
table.moveTo(1);
```

5.123.26 Метод Table::setShowZeroValue

Для упрощения чтения таблицы нулевые значения ячеек могут быть скрыты. Для управления скрытием/показом ячеек используется метод `setShowZeroValue`.

Пример:

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

5.123.27 Метод Table::getShowZeroValue

Для проверки режима отображения нулевых значений ячеек используется метод `getShowZeroValue`.

Пример:

```
Table table = document.getBlocks().getTable(0);
table.setShowZeroValue(false);
Console.WriteLine(table.getShowZeroValue());
```

5.123.28 Метод Table::setVisible

Метод управляет видимостью листа таблицы. Используется только в табличном документе.

Вызов:

```
    setVisible(bool visible)
```

Параметр:

`visible` – параметр, задающий видимость листа. Если значение параметра `visible` равно `true`, то лист таблицы отображается в редакторе таблиц.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

5.123.29 Метод Table::isVisible

Метод возвращает значение true, если лист таблицы в табличном документе отображается в редакторе таблиц.

Пример:

```
Table table = document.getBlocks().getTable(0);
if (!table.isVisible()) {
    table.setVisible(true);
}
```

5.123.30 Метод Table::setPrintArea

Метод служит для установки и сброса области печати, тип аргумента CellRangePosition.

Пример:

```
Table firstSheet = document.getBlocks().getTable(0);
firstSheet.setPrintArea(CellRangePosition(0, 0, 5, 5));
```

5.123.31 Метод Table::getCharts

Для получения списка диаграмм (Charts) таблицы используется метод Table:getCharts.

Пример:

```
TablesEnumerator tablesEnumerator = document.getBlocks().getTablesEnumerator();
foreach (var table in tablesEnumerator)
{
    Charts charts = table.getCharts();
    Console.WriteLine(charts.getChartsCount());
}
```

5.123.32 Метод Table::getNamedExpressions

Для получения списка именованных диапазонов NamedExpressions используется метод Table:getNamedExpressions().

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

```
Table sheetDocumentPage = document.getBlocks().getTable(0);
NamedExpressions namedExpressions = sheetDocumentPage.getNamedExpressions();
NamedExpression namedExpression = namedExpressions.get("Продажи");
if (namedExpression != null) {
    Console.WriteLine(namedExpression.getName());
}
```

5.124 Класс TextAnchoredPosition

Класс `TextAnchoredPosition` представляет позицию объекта на странице текстового документа (см. `Frame::setPosition()`). Описание полей представлено в таблице 60.

Таблица 60 – Описание полей класса `TextAnchoredPosition`

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

Пример:

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

5.125 Класс TextExportSettings

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

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.126 Класс TextProperties

Класс `DocumentAPI.TextProperties` содержит поля, задающие параметры текста. На рисунке 38 изображена объектная модель класса `DocumentAPI.TextProperties`.

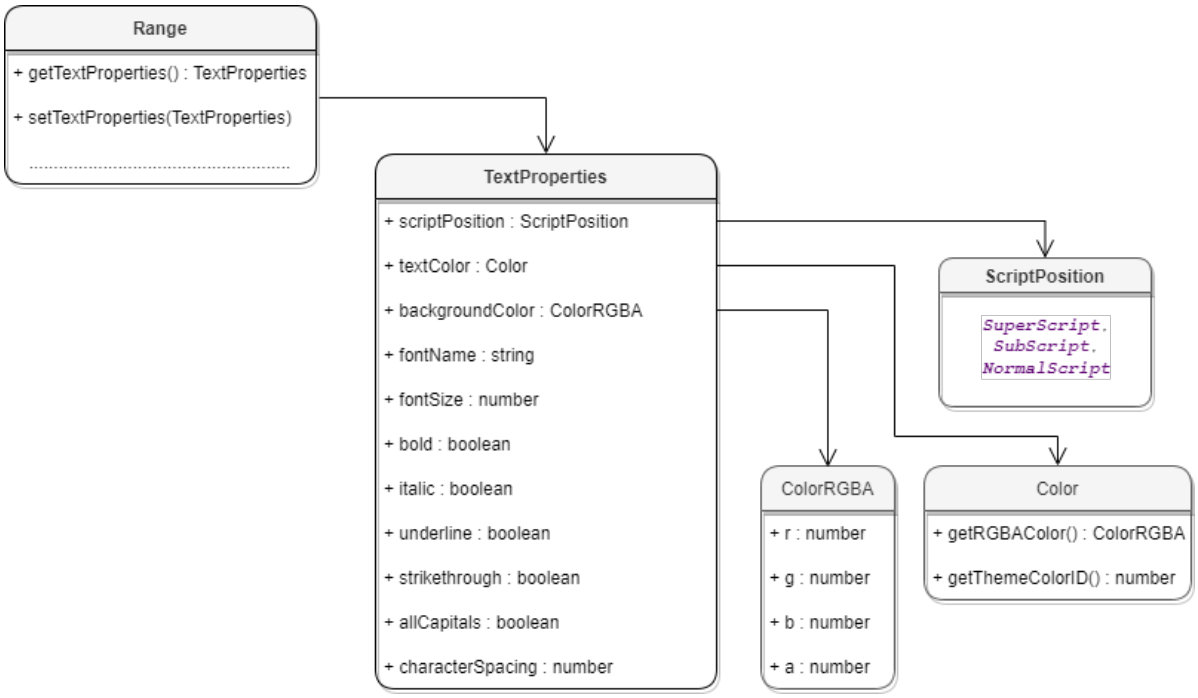


Рисунок 38 – Объектная модель для работы с классом `DocumentAPI.TextProperties`

Описание полей класса `TextProperties` представлено в таблице 61. Свойства `TextProperties` применяются к диапазону текста `Range` (методы `Range::getTextProperties()`, `Range::setTextProperties()`).

Таблица 61 – Описание полей класса `TextProperties`

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

Пример:

```
TextProperties textProperties = new TextProperties();
textProperties.fontName = "XO Oriel";
textProperties.fontSize = 20;
// доступ к тексту третьего абзаца
Paragraph paragraph = document.getBlocks().getParagraph(2);
if (paragraph != null) {
    Range range = paragraph.getRange();
    // установить свойства фрагмента текста
    range.setTextProperties(textProperties);
}
```

5.127 Класс TextWrapType

В таблице 62 представлены варианты обтекания текстом встроенного объекта. Используется в Frame::setWrapType(), Frame::getWrapType().

Таблица 62 – Варианты обтекания текстом встроенного объекта

Наименование константы	Описание
TextWrapType.Inline	Встроенный объект располагается в тексте

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Наименование константы	Описание
TextWrapType.InFrontOfText	Встроенный объект располагается перед текстом
TextWrapType.BehindText	Встроенный объект располагается за текстом
TextWrapType.TopAndBottom	Текст располагается сверху и снизу от встроенного объекта
TextWrapType.Square	Текст располагается вокруг прямоугольной рамки встроенного объекта
TextWrapType.Through	Текст обтекает встроенный объект по сторонам и внутри

Пример:

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

5.128 Класс TimePatterns

Форматы времени представлены в таблице 63. Пример использования см. в разделе DateTimeCellFormatting.

Таблица 63 – Форматы времени

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

5.129 Класс TimeZone

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

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

5.130 Класс TrackedChange

Класс TrackedChange представляет отслеживаемое изменение в диапазоне документа (см. Рисунок 39).

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

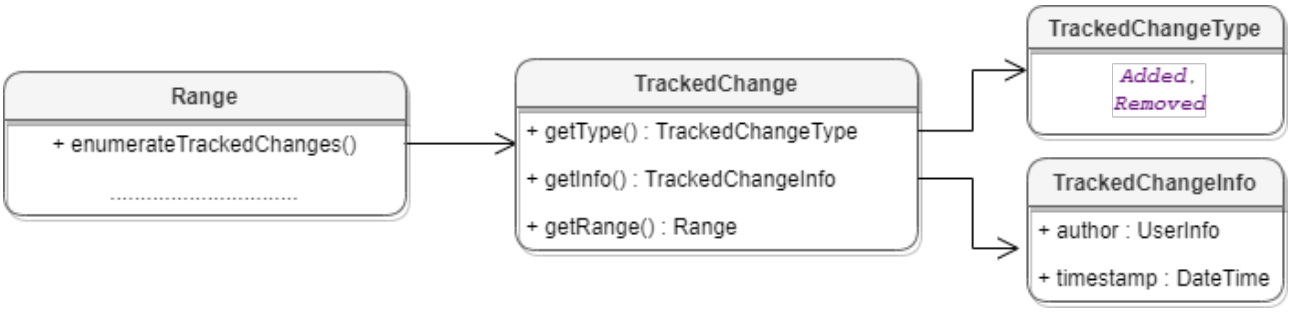


Рисунок 39 – Объектная модель классов для работы с отслеживаемыми изменениями

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

Пример:

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

5.130.1 Метод TrackedChange::getRange

Метод возвращает объект Range, который соответствует измененному диапазону внутри абзаца.

Пример:

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

5.130.2 Метод TrackedChange::getType

Метод позволяет получить информацию о типе отслеживаемого изменения TrackedChangeType.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример:

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

5.130.3 Метод TrackedChange::getInfo

Метод позволяет получить информацию об отслеживаемых изменениях TrackedChangeInfo.

Пример:

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

5.131 Класс TrackedChangeInfo

Класс TrackedChangeInfo содержит информацию об отслеживаемых изменениях. Используется в TrackedChange::getInfo, Comment::getInfo. Описание полей представлено в таблице 64.

Таблица 64 – Описание полей класса TrackedChangeInfo

Поле	Тип	Описание
TrackedChangeInfo.author	UserInfo	Автор изменений
TrackedChangeInfo.timeStamp	DateTime	Дата и время изменений

Пример:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

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

5.132 Класс TrackedChangeType

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

Типы отслеживаемых изменений:

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

Пример:

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

5.133 Класс ThemeColorID

В таблице 65 представлены типы идентификаторов цветов тем. Используется в `Color`.

Таблица 65 – Типы идентификаторов цветов тем

Наименование константы	Описание
`ThemeColorID.Background1`	Фон1
`ThemeColorID.Text1`	Текст1
`ThemeColorID.Background2`	Фон2
`ThemeColorID.Text2`	Текст2
`ThemeColorID.Dark1`	Темная1
`ThemeColorID.Dark2`	Темная2

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Наименование константы	Описание
ThemeColorID.Light1	Светлая1
ThemeColorID.Light2	Светлая2
ThemeColorID.Accent1	Акцент1
ThemeColorID.Accent2	Акцент2
ThemeColorID.Accent3	Акцент3
ThemeColorID.Accent4	Акцент4
ThemeColorID.Accent5	Акцент5
ThemeColorID.Accent6	Акцент6
ThemeColorID.Hyperlink	Гиперссылка
ThemeColorID.FollowedHyperlink	Следующая гиперссылка

5.134 Класс UserInfo

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

Описание полей класса `UserInfo` представлено в таблице 66.

Таблица 66 – Описание полей класса `UserInfo`

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

5.135 Класс ValueFieldsOrientation

Класс `ValueFueldsOrientation` описывает варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений. Является полем класса `PivotTableLayoutSettings`. Описание полей представлено в таблице 67.

Таблица 67 – Описание полей `ValueFueldsOrientation`

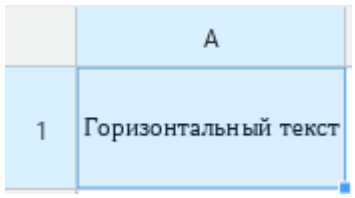
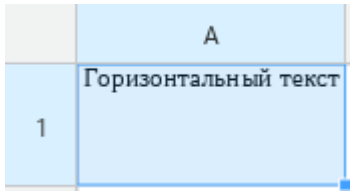
Поле	Описание
ValueFueldsOrientation.ByRows	По строкам
ValueFueldsOrientation.ByColumns	По столбцам

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.136 Класс VerticalAlignment

В таблице 68 представлены константы,описывающие варианты выравнивания текста по вертикали. Используется в CellProperties, ShapeProperties.

Таблица 68 – Виды выравнивания текста по вертикали

Наименование константы	Представление в интерфейсе	
VerticalAlignment.Bottom		
VerticalAlignment.Center		
VerticalAlignment.Top		

Пример:

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

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

cell.setCellProperties(cellProps);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.137 Класс VerticalAnchorAlignment

В таблице 69 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали (см. описание класса VerticalTextAnchoredPosition).

Таблица 69 – Типы выравнивания объекта относительно закрепленной позиции по вертикали

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

5.138 Класс VerticalTextAnchoredPosition

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

Описание полей класса VerticalTextAnchoredPosition представлено в таблице 70.

Таблица 70 – Описание полей класса VerticalTextAnchoredPosition

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

5.139 Класс VerticalRelativeTo

В таблице 71 представлены типы размещения объекта относительно закрепленной позиции по вертикали (см. описание класса VerticalTextAnchoredPosition).

Таблица 71 – Типы размещения объекта относительно закрепленной позиции по вертикали

Наименование константы	Описание
VerticalRelativeTo.Character	Символ

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Наименование константы	Описание
VerticalRelativeTo.BaseLine	Базовая линия
VerticalRelativeTo.Paragraph	Абзац
VerticalRelativeTo.Page	Страница
VerticalRelativeTo.PageContent	Содержимое страницы
VerticalRelativeTo.PageTopMargin	Верхнее поле страницы
VerticalRelativeTo.PageBottomMargin	Нижнее поле страницы
VerticalRelativeTo.PageInsideMargin	Внутреннее поле страницы
VerticalRelativeTo.PageOutsideMargin	Внешнее поле страницы

5.140 Класс VectorString

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

Примеры использования:

```
// Содание вектора
var vector = new VectorString();
// Добавление новых элементов
vector.Add("text1");
vector.Add("text2");
vector.Add("text3");
// Добавление другого вектора
vector.AddRange(vector);
// Определение размера
vector.Capacity = 3;
Console.WriteLine(vector.Capacity);
// Очистка содержимого
vector.Clear();
// Проверка наличия элемента
Console.WriteLine(vector.Contains("text"));
// Сравнение векторов
Console.WriteLine(vector.Equals(vector));
// Определение размера
Console.WriteLine(vector.Count());
// Перечисление элементов
var vectorStringEnumerator = vector.GetEnumerator();
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
while (vectorStringEnumerator.MoveNext())
{
    String stringValue = vectorStringEnumerator.Current;
    Console.WriteLine(stringValue);
}
// Получение подмножества элементов
VectorString range = vector.GetRange(0, 2);
// Получение индекса элемента
Console.WriteLine(vector.IndexOf("text2"));
// Вставка элемента по индексу
vector.Insert(0, "text4");
// Вставка вектора по индексу
vector.InsertRange(0, vector);
// Индекс последнего вхождения элемента
Console.WriteLine(vector.LastIndexOf("text2"));
// Удаление по содержимому элемента
vector.Remove("text2");
// Удаление по индексу
vector.RemoveAt(0);
// Удаление диапазона
vector.RemoveRange(0, 2);
// Преобразование в массив
string[] array = vector.ToArray();
```

5.141 Класс VectorUInt

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

Примеры использования:

```
// Содание вектора
VectorUInt vector = new VectorUInt(3);
// Добавление новых элементов
vector.Add(1);
vector.Add(2);
vector.Add(3);
// Добавление другого вектора
vector.AddRange(vector);
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

```
// Определение размера
vector.Capacity = 3;
Console.WriteLine(vector.Capacity);
// Очистка содержимого
vector.Clear();
// Проверка наличия элемента
Console.WriteLine(vector.Contains(3));
// Сравнение векторов
Console.WriteLine(vector.Equals(vector));
// Определение размера
Console.WriteLine(vector.Count());
// Перечисление элементов
var vectorUIntEnumerator = vector.GetEnumerator();
while (vectorUIntEnumerator.MoveNext())
{
    uint uintValue = vectorUIntEnumerator.Current;
    Console.WriteLine(uintValue);
}
// Получение подмножества элементов
VectorString range = vector.GetRange(0, 2);
// Вставка элемента по индексу
vector.Insert(0, 2);
// Вставка вектора по индексу
vector.InsertRange(0, vector);
// Удаление по индексу
vector.RemoveAt(0);
// Удаление диапазона
vector.RemoveRange(0, 2);
// Преобразование в массив
uint[] array = vector.ToArray();
```

5.142 Класс WorkbookExportSettings

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

Описание полей класса `WorkbookExportSettings` представлено в таблице 72.

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Таблица 72 – Описание полей класса `WorkbookExportSettings`

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

Пример:

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

5.143 Исключения

5.143.1 Класс ApplicationCreateError

Исключение `ApplicationCreateError` наследуется от `System.ApplicationException` и возникает в случае, когда объект `Application` не может быть создан.

Пример:

```
throw new DocumentAPI.ApplicationCreateError("Can not create application");
```

5.143.2 Класс IncorrectArgumentError

Исключение `IncorrectArgumentError` наследуется от `System.ApplicationException` и возникает в случае, когда один из аргументов метода или функции имеет недействительное значение.

Пример:

```
throw new DocumentAPI.IncorrectArgumentError("Invalid arguments");
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.143.3 Класс InvalidObjectError

Исключение `InvalidObjectError` наследуется от `System.ApplicationException` и возникает в случае, когда объект больше не может быть использован.

Пример:

```
throw new DocumentAPI.InvalidObjectError("Can not use this object");
```

5.143.4 Класс DocumentCreateError

Исключение `DocumentCreateError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть создан.

Пример:

```
throw new DocumentAPI.DocumentCreateError("Can not create document");
```

5.143.5 Класс DocumentLoadError

Исключение `DocumentLoadError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть загружен.

Пример:

```
throw new DocumentAPI.DocumentLoadError("Can not load document");
```

5.143.6 Класс DocumentSaveError

Исключение `DocumentSaveError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть сохранен.

Пример:

```
throw new DocumentAPI.DocumentSaveError("Can not save document");
```

5.143.7 Класс DocumentExportError

Исключение `DocumentExportError` наследуется от `System.ApplicationException` и возникает в случае, когда документ не может быть экспортирован.

Пример:

```
throw new DocumentAPI.DocumentExportError("Can not export document");
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.143.8 Класс NoSuchElementError

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

Пример:

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

5.143.9 Класс NotImplementedError

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

Пример:

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

5.143.10 Класс OutOfRangeError

Исключение `OutOfRangeError` наследуется от `System.ApplicationException` и возникает в случае обнаружения выхода значения за пределы диапазона.

Пример:

```
throw new DocumentAPI.OutOfRangeError("Index out of range");
```

5.143.11 Класс ParseError

Исключение `ParseError` наследуется от `System.ApplicationException` и возникает в случае, когда параметр текста не прошел синтаксический анализ.

Пример:

```
throw new DocumentAPI.ParseError("Parse error");
```

5.143.12 Класс UnknownError

Исключение `UnknownError` наследуется от `System.ApplicationException` и возникает в случае, когда критическое исключение возникло по неизвестной причине. Приложение должно быть завершено, поскольку возникло неопределенное состояние ядра Document API.

Пример:

```
throw new DocumentAPI.UnknownError("Unknown error");
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.143.13 Класс ForbiddenActionError

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

Пример:

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

5.143.14 Класс DocumentModificationError

Исключение `DocumentModificationError` наследуется от `System.ApplicationException` и возникает в случае, когда невозможно выполнить операцию по изменению документа.

Пример:

```
throw new DocumentAPI.DocumentModificationError("Can not modify document");
```

5.143.15 Класс PivotTableError

Исключение `PivotTableError` наследуется от `System.ApplicationException` и возникает в случае ошибки при работе со сводными таблицами. Например, применение фильтра, который не может быть применен к сводной таблице.

Пример:

```
throw new DocumentAPI.PivotTableError("Pivot table error");
```

5.143.16 Класс PositionDocumentsMismatchError

Исключение `PositionDocumentsMismatchError` наследуется от `System.ApplicationException` и возникает в случае, когда несколько позиций относятся к различным документам и не могут быть использованы в одной операции. Данное исключение возникает при попытке пользователя создать диапазон (`Range`), включающий позиции (`Position`), принадлежащие нескольким различным документам, и выполнить операцию для такого диапазона.

Пример:

```
throw new DocumentAPI.PositionDocumentsMismatchError("Position mismatch
error");
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

5.143.17 Класс ScriptExecutionError

Исключение `ScriptExecutionError` наследуется от `System.ApplicationException` и возникает в случае, когда сценарий не удается выполнить.

Пример:

```
throw new DocumentAPI.ScriptExecutionError("Can not execute scenario");
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

6 ИСПОЛЬЗОВАНИЕ КОДИРОВОК

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

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

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

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

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

Пример загрузки документа:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

Пример сохранения документа:

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

Пример экспорта документа:

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

Пример заполнения ячейки таблицы:

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

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

Пример вставки текста в документ:

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

Пример замены текста в диапазоне:

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

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023

7 КОНТРОЛЬ ВЕРСИЙ DOCUMENT API

Константы версии Document API `Major` и `Minor` позволяют проверить совместимость предыдущей и текущей версии Document API.

Если была изменена константа `Major` версии Document API, то в Document API произошли обратно несовместимые изменения, и программный код должен быть пересмотрен и обновлен. Обратно несовместимыми изменениями считаются: переименование, удаление или несовместимое изменение подписи существующих классов или методов, а также добавление новых методов, типов и полей класса.

Если была изменена константа `Minor` версии Document API, то в Document API произошли только обратно совместимые изменения, и нет необходимости менять программный код, чтобы он работал с более новой версией Document API. Но гарантируется совместимость только на уровне исходного кода C#, поэтому необходимо перекомпилировать программный код приложения с более новой версией Document API.

Рекомендуется проверить версию Document API до инициализации, как указано ниже:

```
uint ExpectedMajorAPIVersion = 1;
uint ExpectedMinorAPIVersion = 0;
if (!DocumentAPI.isAPIVersionCompatible(ExpectedMajorAPIVersion,
ExpectedMinorAPIVersion)) {
    // Вывод сообщения о несовместимости версии библиотеки Document API и выход
из программы
}
```

Пример проверки совместимости указанной версии Document API с текущей:

```
public class DocumentAPI {
  public static bool isAPIVersionCompatible(uint major, uint minor);
}
```

© ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ», 2013–2023