Apache Poi-библиотека Java с открытым исходным кодом для форматов файлов Microsoft Office

августа 1, 2023 · 4 мин · Kashif Iqbal

Apache Poi (Плохая реализация запутывания)-это популярная библиотека Java с открытым исходным кодом, разработанную Foundation Apache Software Foundation. POI означает «плохая реализация запутывания» с юмором, ссылаясь на проприетарные бинарные файлы Microsoft. Основная цель Apache Poi состоит в том, чтобы предоставить разработчикам Java набор API, которые позволяют им читать, писать и манипулировать различными форматами файлов Microsoft Office, таких как электронные таблицы Excel (.xls и .xlsx), слово документы (.doc и .docx) и презентации PowerPoint (.ppt и .pptx).

Краткая история Apache Poi {.wp-block heading}

В начале 2000 -х годов, когда разработчикам Java разработчики Java работали с файлами Microsoft Office, не вдаваясь в основные данные форматов файлов, Apache Foundation начал работать над реверсием форматов файлов Microsoft. Это привело к выводам, что форматы были плохо запутаны и были обработаны. Вот почему имя POI, т. Е. **Плохая реализация запутывания** . За прошедшие годы библиотека подверглась значительной разработке, добавив поддержку новых функций и форматов файлов, повышая производительность и повышение удобства использования.

поддерживаемые форматы файлов {.wp-block heading}

Apache Poi поддерживает работу с форматами файлов Microsoft Excel, Microsoft Word и Microsoft PowerPoint.

Microsoft Excel {.wp-block heading}

Apache Poi имеет следующие API для работы с таблицами Microsoft Excel. ** hssf:- ***Оправочный формат электронной таблицы*-поддерживает работу с 97-2003 и перед электронной таблицей Excel xls Формат файла ** xssf:*xml Format* - поддерживает Office Office Открыть формат файла XML из электронной таблицы Excel xlsx Форматы файлов

Microsoft word {.wp-block heading}

Apache Poi имеет следующие API для работы с документами Microsoft Word. ** hwpf:- ***исправление текстового процессора Format*-Читать и написать Microsoft Word 97-

2003 DOC Формат файла ** xwpf:*xml Word Format*** - аналогичная функция, установленная на HWPF, но для Office Open XML DOCX Формат файла

Microsoft PowerPoint {.wp-block heading}

Apache Poi имеет следующие API для работы с презентациями Microsoft PowerPoint. ** HSLF:-***Оправочный макет слайда Формат*****-реализация Java для Microsoft PowerPoint 97-2003 PPT File Format ** xslf:***xml Slide Mayout Format*** - реализация Java для Office Открыть файлы XML Microsoft PowerPoint, т. Е. Формат файла PPTX

microsoft outlook {.wp-block heading}

** hsmf:**_ Образец реализация глупой почты Format_ -Java для Microsoft Outlook формат файла MSG

Microsoft Publisher {.wp-block heading}

** hpbf:**_ -исправление издателя Format_ - реализация Java для Microsoft Publisher Формат файла Pub

Microsoft Visio

** hdgf:**_ -образная диаграмма Формат_ - реализация Java для Microsoft Visio VSD - формат10

Установите Apache Poi для java

На момент написания этой статьи последний стабильный выпуск Apache Poi - 5.2.3, который можно загрузить с сайта Apache Poi, Github и Maven. Мы посмотрим, как вы можете установить API из Maven, а также загрузить его с сайта Apache Poi для использования в вашем проекте Java.

Как установить Apache Poi из Maven?

Apache опубликовал артефакты Apache Poi Maven для автоматической установки в проектах Maven с использованием файлов pom.xml. Мы можем установить зависимость в проекте Maven, чтобы он автоматически приносит файлы JAR, используемые для запуска приложения. Ниже приведены шаги, чтобы включить зависимость в Pom.xml вашего проекта Maven. ** Шаг 1:**Откройте свой проект Maven в своей Java IDE. Вы можете использовать идею Netbeans, Eclipse или Intellij в соответствии с вашим собственным выбором. ** Шаг 2:**Добавьте следующую зависимость в файл POM.

```
<dependency>

    <groupId>org.apache.poi</groupId>

    <artifactId>poi</artifactId>
```

```
    <version>3.9</version>
</dependency>
```

** Шаг 3:**Добавьте зависимость компонента POI для Office Office Open Open File File Formats следующим образом.

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.3</version>
</dependency>
```

** Шаг 4:**Добавьте зависимость Commons-IO следующим образом.

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.7</version>
</dependency>
```

** Шаг 5:**Добавьте зависимость log4j следующим образом.

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.20.0</version>
</dependency>
```

На этом этапе ваш проект принесет зависимости, упомянутые выше в файле pom.xml и включать соответствующие файлы JAR в ваш проект для работы с форматами файлов Microsoft Office.

Установить Apache Poi из github {.wp-block heading}

Apache Poi предоставил экземпляр зеркала на GitHub для доступа и загрузки исходного кода. Вы можете получить доступ к ним из репозитория Apache Poi github.

apache poi скачать {.wp-block heading}

Вы также можете установить Apache Poi, загрузив последнюю версию Apache Poi с официальной загрузочной страницы Apache. После загрузки повреждены содержимое пакета в папку и включите файлы JAR в ваш проект, чтобы начать с API APACHE POI.

apache poi resources {.wp-block heading}

- Apache Poi
- Обзор компонентов Apache POI
- Apache Poi скачать
- Apache Poi - Javadocs
- Apache Poi Maven

Что будет дальше

В наших будущих статьях мы будем писать статьи с примерами:

- **Использование Apache POI для работы с файлами электронных таблиц MS Excel** * Создать рабочую книгу Excel с помощью Apache Poi для Java API * Читать файлы Excel в Java с Apache Poi * Добавить изображение в ячейку Excel с Apache Poi * Работа с формулами Excel с Apache Poi * Вставьте заголовок и нижний колонтитул в Excel с Apache Poi
  - Использование Apache POI для работы с файлами MS Word
  - Использование Apache POI для работы с файлами презентаций MS PowerPoint
  - Java API для доступа к форматам файлов PowerPoint Так что следите за этим.

- Presentation
- Spreadsheet
- Word Processing

- Apache POI
- Apache POI Maven

« ПРЕДЫДУЩАЯ
Создать рабочую книгу Excel в Java, используя Apache PoiСЛЕДУЮЩАЯ »
Как выровнять текст в документах Word программно

Генерация файлов Word в Apache POI

8 мин

22K

Java*Apache*

Для языка Java (как, впрочем, и для любого другого языка программирования) всё еще не придумали более простого и действенного способа генерации документов docx, чем библиотека Apache POI. В конце нулевых появился сей высокоуровнеый API, позволящий говорить с формируемым документом не на языке разметки

XML, а с помощью удобных полей и выводов.

Судя по моим Google-запросам на протяжении более чем года сообщество пользователей этой библиотеки продержалось года этак до 2012, в то время как новые версии библиотеки всё еще появляются на главной странице проекта. Не на все вопросы, касающиеся формирования самого примитивного документа, есть ответы в документации или stackoverflow, не говоря уже о текстах на русском языке. Постараемся компенсировать этот недостаток данных для тех, кому это может понадобиться.

Основные классы API

XWPFDocument — целостное представление Word документа. В нём не только содержится xml-код, интерпретируемый редакторами (Word, LibreOffice), но также содержатся и методы для определения метаданных отображения — набора стилей, сносок и т.п. В этой статье поговорим о первом, так как работа с метаданными не так явно задокументирована, к тому же многие редакторы успешно справляются с отображением документа и без подсказок.

Итак, предположим, у вас на руках есть (ненужный) файл docx. Преобразуем его в файл zip (осторожно, обратное преобразование путем переименования zip -> docx может сделать файл недоступным для вашего редактора(!)), в получившемся архиве откроем папку word, а в ней — файл document.xml. Перед нами xml-представление word-файла, которое также можно было бы получить через Apache POI, с меньшими трудностями.

```
File file = new File("C:/username/document.docx");
FileInputStream fis = new FileInputStream(file.getAbsolutePath());
XWPFDocument document = new XWPFDocument(fis); // Вот и объект описанного нами класса
String documentLine = document.getDocument().toString();
```

Для того, чтобы поближе познакомиться с содержимым документа, придется вооружиться еще двумя классами API: XWPFParagraph и XWPFTable.

XWPFParagraph — как следует из названия, представляет собой параграф документа. Расположен он может быть как внутри XWPFDocument,

```
document.getParagraphs();
XWPFParagraph lastParagraph = document.createParagraph();
```

так и внутри таблицы (если точнее — внутри ячейки таблицы, вложенной в ряд таблицы, вложенного непосредственно в таблицу).

```
document.createTable().createRow().createCell().addParagraph();
```

Параграф предоставляет изрядный набор информации для вёрстки и размещения текста. Официальная документация на этот счёт достаточно красноречива: отступы слева и справа, сверху и снизу, в том числе и между строками, добавление гиперссылок и границ для параграфа.

XWPFTable — класс, олицетворяющий таблицу. Также как и в XWPFParagraph, XWPFTable можно добавлять к самому документу и к ячейке таблицы (создавая, тем самым, таблицу внутри таблицы). Семантика в таком случае чуточку усложняется.

```
XWPFTable table = document.createTable(); //Здесь всё просто, создаем таблицу в документе и работаем с ней.
XWPFCell cell = table.createRow().createCell();//Добавим к таблице ряд, к ряду - ячейку, и используем её.
XWPFTable innerTable = new XWPFTable(cell.getCTTc().addNewTbl(), cell, 2, 2); //Воспользуемся конструктором для добавления таблицы - возьмем cell и её внутренние свойства, а так же зададим число рядов и колонок вложенной таблицы
cell.insertTable(cell.getTables().size(), innerTable);
```

XWPFRun — набор данных о выводе текста внутри параграфа. Находится может только внутри параграфа, создается через вызов метода параграфа-родителя:

```
paragraph.createRun();
```

Из нескольких «ранов», как я предпочитаю их называть, и состоит целый параграф текста в Word. Каждый «ран» имеет свою настройку шрифта, его цвета и размера, а также стилизации. Через добавление различных «ранов», подчиняющихся разметке параграфа, можно выводить тексты с совершенно разной стилизацией.

Как становится видно из обзора классов, перенос, скажем, css-стиля в

документ будет связан с дополнительной сложностью: часть свойств необходимо будет применить к параграфу docx, часть — к объекту класса XWPFRun.

Итак, библиотека легла в External Libraries/jar лежит под рукой, пора творить.

Создадим документ, добавим таблицу 2x2 и параграф.

```
XWPFDocument document = new XWPFDocument();
XWPFTable table = document.createTable(2, 2);
XWPFParagraph paragraph = document.createParagraph();
fillTable(table);
fillParagraph(paragraph);
```

Заполним параграф, добавив ран для вывода текста. После перевода строки стилизация параграфа будет потеряна, и в Word новый параграф будет выведен без красной строки.

```
void fillParagraph(XWPFParagraph paragraph) {
  paragraph.setIndent(20);
  XWPFRun run = paragraph.createRun();
  run.setFontSize(12);
  run.setFontFamily("Times New Roman");
  run.setText("My text");
  run.addBreak();
  run.setText("New line");
}
```

Теперь займёмся заполнением таблицы. Мы можем обращаться не только к уже созданным элементам, но и вызвать у сформированной таблицы метод для добавления рядов или колонок.

```
void fillTable(XWPFTable table) {
XWPFRow firstRow = table.getRows().get(0);
XWPFRow secondRow = table.getRows().get(1);
XWPFRow thirdRow = table.createRow();
fillRow(firstRow);
}
```

Опускаемся глубже, на уровень ряда таблицы. Именно в таком порядке предстаёт разбор таблицы в Apache POI — сначала ряды, потом клетки. Напрямую из таблицы можно получить лишь количество колонок в

таблице:

```
table.getColBandSize();
```

Итак, ряд.

```
void fillRow(XWPFRow row) {
   List<XWPFTableCell> cellsList = row.getCells();
   cellsList.forEach(cell -> fillParagraph(cell.createParagraph()));
}
```

Оказавшись в ячейке двигаться глубже уже некуда, поэтому можно снова вызвать наш дуболомный метод по заполнению параграфа, предварительно создав его в таблице.

Итак, можно легко уловить суть структуры документа в Word: вкладывай одно в другое и предоставляй доступ (в том числе и к созданию новых экземпляров). К сожалению, далеко не всегда есть возможность получить последний элемент во вложенной коллекции. Чаще всего приходится пользоваться такими вот ухищрениями:

```
XWPFRun lastRunOfParagraph = paragraph.getRuns(paragraph.getRuns().size() - 1);
```

Хорошо, с содержимым таблицы разобрались. Что если нам нужно явно уточнить ширину таблицы, а не оставлять её для волной интерпретации редактора?

Для некоторых на первый взгляд числовых значений, например, ширины таблицы, в Apache POI существуют целые классы.

```
CTTblWidth widthRepr = table.getCTTbl().getTblPr().addNewTblW();
widthRepr.setType(STTblWidth.DXA);
widthRepr.setW(BigInteger.valueOf(4000));
```

С помощью типа укажем, какая именно ширина нам нужна: auto, pct или dxa. В первом случае таблицы займёт всю предоставленную ей ширину, во втором — процент от всей ширины, указанный позже методом setW. В нашем же случае вмешиватеся специальная единица измерения — dxa, равная 1/20 точки.

Классы, подобные CTTblWidth, используются повсеместно: для определения ширины страницы (PgSize), ширины ячейки и др.

Единцы измерения в Apache POI

В хорошем документе всё выверенно и расчерчено идеально, вплоть до самого пикселя. Возможно, в теории можно сделать всё средствами Apache POI и без углубления в тему единиц измерения, но лучше уделить им внимание сразу, чтобы избежать недопониманий в духе «почему это схлопнулось» и «когда переместил картинку в word на один сантиметр».

О поддержке сантиметров и остальной метрической системы тут остается только мечтать. Это резонно (каждый шрифт уникален, у каждого редактора своя специфика), но дико неудобно. Придется прибегнуть ко множеству конвертаций, если вы хотите задавать отступы (ведь именно в сантиметрах мы привыкли видеть их в word) в сантиметрах. Итак, указав тип измерения dxa для некоторой ширины, как описно в параграфе выше, мы получаем в распоряжение некоторое точное значение, но абсолютно не представляем как им воспользоваться. Для перевода в сантиметры на stackoverflow есть формула. Для всего остального существует класс Units. В нем определены как методы для перевода единиц измерения, так и сами соотношения между значениями.

Запись готового документа

Для записи в конечный файл есть удобный метод XWPFDocument — write. На вход принимается поток, в который пойдёт запись.

```
document.write(new FileOutputStream(new File("/path/to/file.docx")));
```

Если готовый документ нужно куда-то передать можно подать в качестве аргумента не File-, а ByteArrayOutputStream.

Информация об элементе отображения в формате xml

Имея документ, отображающийся корректно в определенном редакторе,

полезно было бы узнать как именно представлен необходимый параграф или другой элемент. Для этого определенны специальные методы, возвращающие объекты классов пакета org.openxmlformats.schemas.wordprocessingml.x2006.main. Из названия (wordprocessingml) видно, что данный набор классов используется только для работы с документами word. Например, для xlsx документов есть пакет spreadsheetml, некоторые классы которого очень и очень похожи на классы wordprocessingml, поэтому конвертация между форматами достаточно затруднена.

```
paragraph.getCTP();
table.getCTTbl();
```

Так, пустой параграф будет иметь скромное представление

```
<xml-fragment/>
```

Пустая таблица покажет больше интересного.

```
<xml-fragment
xmlns:main="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <main:tblPr>
    <main:tblW main:w="0" main:type="auto"/>
    <main:tblBorders>
      <main:top main:val="single"/>
      <main:left main:val="single"/>
      <main:bottom main:val="single"/>
      <main:right main:val="single"/>
      <main:insideH main:val="single"/>
      <main:insideV main:val="single"/>
    </main:tblBorders>
  </main:tblPr>
  <main:tr>
    <main:tc>
      <main:p/>
    </main:tc>
    <main:tc>
      <main:p/>
    </main:tc>
  </main:tr>
  <main:tr>
    <main:tc>
      <main:p/>
    </main:tc>
    <main:tc>
      <main:p/>
    </main:tc>
  </main:tr>
```

```
  <main:tr>
    <main:tc>
      <main:p/>
    </main:tc>
    <main:tc>
      <main:p/>
    </main:tc>
  </main:tr>
</xml-fragment>
```

Что здесь интересного? Свойства tblPr — всевозможные свойства таблицы. Внутри уже описанная ширина таблицы (установлена 0, но свойство «auto» все равно выведет таблицу в приемлимой, автоматической ширине). Также tblBorders — набор информации о границах таблицы. Далее идёт явно выраженное представление внутренностей таблицы. tr — ряд таблицы, внутри вложенны tc. Внутри tc оказался бы набор вложенный параграфов, если бы мы добавили хотя бы один.
Попробуем пополнить параграф информацией и посмотреть что из этого получится.

```
XWPFParagraph xwpfParagraph = document.getParagraphs().get(0);
        xwpfParagraph.setFirstLineIndent(10);
        XWPFRun run = xwpfParagraph.createRun();
        run.setFontFamily("Times New Roman");
        run.setText("New text");
```

Получаем:

```
<xml-fragment
xmlns:main="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <main:pPr>
    <main:ind main:firstLine="10"/>
  </main:pPr>
  <main:r>
    <main:rPr>
      <main:rFonts main:ascii="Times New Roman" main:hAnsi="Times New Roman"
main:cs="Times New Roman" main:eastAsia="Times New Roman"/>
    </main:rPr>
    <main:t>New text</main:t>
  </main:r>
</xml-fragment>
```

Здесь ситуация ровно такая же: объект с мета-информацией (в него добавлена информация об отступе красной строки, который мы вложили в коде), а так же само содержимое: там размещается список «ранов». В первый и единственный мы добавили текст и информацию о шрифте. Эта информация также разделилась внутри «рана» — информация о

шрифте попала в rPr, сам текст — в элемент t.

Вместо вывода

Apache POI предоставляет удобный, и, что не менее важно, бесплатный API для работы с документами. В нем непросто добиться единого отображения во всех редакторах (Office Online и LibreOffice обязательно будут выглядеть иначе), есть множество неудобств с единицами измерения, а так же непонятно где и какие свойства в элементах должны находиться. Тем не менее, работа с этими свойствами подчинена логике, а возможность подглядеть в xml не нарушая эту логику делает разработку гораздо более удобной.

Теги:

- java
- apache poi
- word
- docx

Хабы:

- Java
- Apache