Apache POI.Word

an open source library developed and distributed by Apache Software Foundation to design or modify Microsoft Office files
Written in Java

By: Mobina Goodarzi

Table of Contents

Introduction

Apache POI is a popular API that allows programmers to create, modify, and display MS Office files using Java programs. It is an open-source library developed and distributed by the Apache Software Foundation to design or modify Microsoft Office files using Java programs. It contains classes and methods to decode the user input data or a file into MS Office documents.

It is for working with the various file formats based on the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2).

This project focuses on the support of Apache POI for Microsoft Word.

Setting up and using Apache POI library in Java

The only dependency that is required for Apache POI to handle MS Word files is:

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.5</version>
</dependency>
```

I used some other dependencies like:

Apache POI:

Purpose: Apache POI is a Java library for working with various Microsoft document formats, such as Word, Excel, and PowerPoint. It provides classes and methods to create, read, and modify these documents programmatically.

Apache POI - OOXML:

Purpose: This module extends the functionality of Apache POI to support the Office Open XML (OOXML) file formats, which include newer versions of Microsoft Office documents such as .docx, .xlsx, and .pptx.

Apache Log4j API:

Purpose: Apache Log4j API provides a logging framework for Java applications. It allows developers to log messages at various levels of severity and configure logging behavior dynamically.

Apache Log4j Core:

Purpose: Apache Log4j Core is the implementation of the Log4j API. It provides the core functionality for logging, including handling log messages, formatting, and outputting them to various destinations such as files, console, or databases.

Document Creator Class

```
1 usage  Mobina Goodarzi
public class DocumentCreator {
    1 usage  Mobina Goodarzi
    public static void createWordDocument(String documentPath) throws Exception {
        // Create a new document
        XWPFDocument doc = new XWPFDocument();

        XWPFParagraph paragraph = doc.createParagraph();
        XWPFRun run = paragraph.createRun();
        paragraph.setAlignment(ParagraphAlignment.CENTER);
        run.setText("Apache POI ساخته شده توسط");
        run.setFontFamily("IranNastaliq"); // Set font for Persian text
        run.getCTR().getRPr().addNewRtl().setVal(true); // Set RTL for Persian text

        // Add a line break
        paragraph.createRun().addBreak();


        // Save the document in DOCX format
        try (FileOutputStream out = new FileOutputStream(documentPath)) {
            doc.write(out);
            doc.close();
        }
    }
}
```

this code creates a Word document containing the text " Apache POI ساخته شده توسط ", with the Persian text formatted using the "IranNastaliq" font and right-to-left directionality.

- First, an instance of the XWPFDocument class is created, which serves as the main structure for the Word document.
- A new paragraph is created using doc.createParagraph().
- An XWPFRun object is created as the textual part of the paragraph, and the desired text is added to it.
- The alignment of the paragraph text is set to center using paragraph.setAlignment(ParagraphAlignment.CENTER).
- The font family of the Persian text is set to "IranNastaliq" using run.setFontFamily("IranNastaliq").
- Right-to-left directionality is enabled for the Persian text using run.getCTR().getRPr().addNewRtl().setVal(true).
- A new line break is added using paragraph.createRun().addBreak().
- Finally, the Word document is written to the output file using doc.write(out), and the file is closed after completion.

Document Editor Class

```java
2 usages  Mobina Goodarzi
public class DocumentEditor {
    no usages  Mobina Goodarzi
    public static void accessToTableCells(String documentPath) throws Exception {
        XWPFDocument doc = new XWPFDocument(new FileInputStream(documentPath));

        XWPFTable table = doc.getTableArray( pos: 0);

        // Access the first cell in the table
        XWPFTableCell cell = table.getRow( pos: 0).getCell( pos: 0);

        // Remove all content from the cell
        cell.removeParagraph( pos: 0);

        // Add new content to the cell
        XWPFParagraph paragraph = cell.addParagraph();
```

DocumentEditor class provides several methods for editing Word documents like;

1. **accessToTableCells(String documentPath)**: This method allows access to the cells of table in the specified Word document. It retrieves the selected cell of the selected row, and applies the desired changes.
2. **addTableToDocument(String documentPath)**: This method adds a table to the specified Word document.
3. **addImage(String documentPath)**: This method inserts an image into the specified Word document. It inserts the image located at the desired path.

Tables Class

```java
1 usage  Mobina Goodarzi
public class Tables {
    1 usage  Mobina Goodarzi
    public static void formattedTable(XWPFDocument document) throws Exception {
        // Create a table
        XWPFTable table = document.createTable();
        table.setWidth("100%");

        // Create the first row
        XWPFTableRow tableRowOne = table.getRow( pos: 0);
        // Create a separate run for each cell
        XWPFRun runCell1 = tableRowOne.getCell( pos: 0).addParagraph().createRun();
        XWPFRun runCell2 = tableRowOne.addNewTableCell().addParagraph().createRun();
        XWPFRun runCell3 = tableRowOne.addNewTableCell().addParagraph().createRun();

        // Set text and properties for each cell
        runCell1.setText("(Number) شماره");
        runCell1.setFontFamily("IranNastalig");
        runCell1.getCTR().getRPr().addNewRtl().setVal(true);
        runCell1.getParagraph().setAlignment(ParagraphAlignment.CENTER);
        tableRowOne.getCell( pos: 0).setColor("7e8b9e");
```

This **Tables** class provides methods to generate various types of tables in Word documents.

These methods offer flexibility in creating tables with different formats and styles.

Existence Checker Of Document Class

```
2 usages  Mobina Goodarzi
public class ExistenceCheckerOfDocument {
    2 usages  Mobina Goodarzi
    public static boolean isDocumentExists(String documentPath) {
        File file = new File(documentPath);
        return file.exists();
    }
}
```

The **ExistenceCheckerOfDocument** class contains a static method **isDocumentExists(String documentPath)** that checks whether a document file exists at the specified path. It takes a **String** parameter **documentPath**, which represents the path to the document file, and returns a boolean value.

Word view

ApachePOI

Number	FamilyName Name	Age
1	name	--
2	Mobina Goodarzi	23

Adding an image to the document:

In conclusion

Despite its potential drawbacks, Apache POI remains one of the most popular and powerful libraries for working with Office documents in the Java programming language. Its extensive features, along with the wealth of resources and free availability, make it a preferred choice for many developers.