POI operates word2007 through XWPFDocument

原创 A lazy programmer Posted on 2018-04-16 17:54:19 Read 3.4w Collection 60 Likes 8

Category Column: poi

Using POI-3.9

```
package com.utils;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;
import org.apache.commons.lang.StringUtils;
import org.apache.poi.util.Units;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFFooter;
import org.apache.poi.xwpf.usermodel.XWPFHeader;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;
import org.apache.poi.xwpf.usermodel.XWPFRun;
import org.apache.poi.xwpf.usermodel.XWPFTable;
import org.apache.poi.xwpf.usermodel.XWPFTableCell;
import org.apache.poi.xwpf.usermodel.XWPFTableRow;
import org.apache.xmlbeans.XmlToken;
import org.openxmlformats.schemas.drawingml.x2006.main.CTNonVisualDrawingProps;
import org.openxmlformats.schemas.drawingml.x2006.main.CTPositiveSize2D;
import org.openxmlformats.schemas.drawingml.x2006.wordprocessingDrawing.CTInline;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTFonts;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTHpsMeasure;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTJc;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTPPr;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTRPr;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTRow;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTSpacing;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTbl;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTblPr;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTblWidth;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTc;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTcPr;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTVerticalJc;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.STJc;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.STLineSpacingRule;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.STMerge;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.STTblWidth;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.STVerticalJc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public final class XWPFUtil {
    /**
     * 当前打开文档总页数
     */
    private int pages = 1;

    /**
     * 页眉
     */
    public static final String DOC_HEADER = "header";

    /**
     * 页脚
     */
    public static final String DOC_FOOTER = "footer";
    /**
     * Logger
     */
    private static Logger log = LoggerFactory.getLogger(XWPFUtil.class);

    /**
     * 打开Word
     * @param filePath 文件全路径
     */
    public static XWPFDocument open(String filePath) throws Exception {
        InputStream is = null;
        try {
            log.info("POI打开Word文件: " + filePath);
            is = new FileInputStream(filePath);
```

```
            return new XWPFDocument(is);
        } finally {
            if (is != null) {
                is.close();
            }
        }
    }
    /** 构造函数 */
    public XWPFUtil() {
    };
    /**
     * 保存Word
     * @param document Word文档
     * @param filePath 保存路径
     */
    public static void save(XWPFDocument document, String filePath) throws Exception {
        FileOutputStream fos = null;
        try {
            log.info("POI生成Word文件: " + filePath);
            File file = new File(filePath);
            FileUtil.makeDir(file.getParentFile());
            fos = new FileOutputStream(file);
            document.write(fos);
        } finally {
            if (fos != null) {
                fos.close();
            }
        }
    }

    /**
     * 取得全部段落（含列表）
     * @param document Word文档
     */
    public static List<XWPFParagraph> getAllParagraphs(XWPFDocument document) {
        List<XWPFParagraph> paragraphs = new ArrayList<XWPFParagraph>();
        // 列表外段落
        paragraphs.addAll(document.getParagraphs());
        // 列表内段落
        List<XWPFTable> tables = document.getTables();
        for (XWPFTable table : tables) {
            List<XWPFTableRow> rows = table.getRows();
            for (XWPFTableRow row : rows) {
                List<XWPFTableCell> cells = row.getTableCells();
                for (XWPFTableCell cell : cells) {
                    paragraphs.addAll(cell.getParagraphs());
                }
            }
        }
        return paragraphs;
    }

    /**
     * 取得某个表格的段落
     * @param document Word文档
     * @param table XWPFTable
     */
    public static List<XWPFParagraph> getTableParagraphs(XWPFDocument document, XWPFTable table) {
        List<XWPFParagraph> paragraphs = new ArrayList<XWPFParagraph>();
        // 列表内段落
        List<XWPFTableRow> rows = table.getRows();
        for (XWPFTableRow row : rows) {
            List<XWPFTableCell> cells = row.getTableCells();
            for (XWPFTableCell cell : cells) {
                paragraphs.addAll(cell.getParagraphs());
            }
        }
        return paragraphs;
    }

    /**
     * 取得某个表格某一列的段落
     * @param document Word文档
     * @param table XWPFTable
     * @param col 列号
     */
    public static List<XWPFParagraph> getTableColParagraphs(XWPFDocument document, XWPFTable table, int col) {
        List<XWPFParagraph> paragraphs = new ArrayList<XWPFParagraph>();
        List<XWPFTableRow> rows = table.getRows();
        for (XWPFTableRow row : rows) {
            XWPFTableCell cell = row.getTableCells().get(col);
            paragraphs.addAll(cell.getParagraphs());
        }
        return paragraphs;
    }

    /**
     * 取得页眉段落（含列表）
     * @param document Word文档
```

```java
     */
    public static List<XWPFParagraph> getAllHeaderParagraphs(XWPFDocument document) {
        List<XWPFParagraph> paragraphs = new ArrayList<XWPFParagraph>();
        // 取得页眉
        List<XWPFHeader> headerList = document.getHeaderList();
        // 取得页眉列表外段落
        List<XWPFParagraph> headerParas = headerList.get(0).getParagraphs();
        paragraphs.addAll(headerParas);
        // 取得页眉列表内段落
        List<XWPFTable> tables = headerList.get(0).getTables();
        for (XWPFTable table : tables) {
            List<XWPFTableRow> rows = table.getRows();
            for (XWPFTableRow row : rows) {
                List<XWPFTableCell> cells = row.getTableCells();
                for (XWPFTableCell cell : cells) {
                    paragraphs.addAll(cell.getParagraphs());
                }
            }
        }
        return paragraphs;
    }

    /**
     * 取得页脚段落（含列表）
     * @param document Word文档
     */
    public static List<XWPFParagraph> getAllFooterParagraphs(XWPFDocument document) {
        List<XWPFParagraph> paragraphs = new ArrayList<XWPFParagraph>();
        // 取得页脚
        List<XWPFFooter> footerList = document.getFooterList();
        // 取得页脚列表外段落
        List<XWPFParagraph> footerParas = footerList.get(0).getParagraphs();
        paragraphs.addAll(footerParas);
        // 取得页脚列表内段落
        List<XWPFTable> tables = footerList.get(0).getTables();
        for (XWPFTable table : tables) {
            List<XWPFTableRow> rows = table.getRows();
            for (XWPFTableRow row : rows) {
                List<XWPFTableCell> cells = row.getTableCells();
                for (XWPFTableCell cell : cells) {
                    paragraphs.addAll(cell.getParagraphs());
                }
            }
        }
        return paragraphs;
    }

    /**
     * 替换全部文本
     * @param textMap 替换信息
     * @param paragraphs 段落
     */
    public static void replaceAllTexts(Map<String, String> textMap, List<XWPFParagraph> paragraphs) {
        Set<Entry<String, String>> textEntrySet = textMap.entrySet();
        for (Entry<String, String> entry : textEntrySet) {
            String key = entry.getKey();
            String value = CommonUtil.converObjToStr(entry.getValue());
            // 替换全部文本
            replaceAllTexts(key, value, paragraphs);
        }
    }

    /**
     * 替换全部文本
     * @param key 替换键
     * @param value 替换值
     * @param paragraphs 段落
     */
    public static void replaceAllTexts(String key, String value, List<XWPFParagraph> paragraphs) {
        for (XWPFParagraph paragraph : paragraphs) {
            // 待替换文本
            String text = paragraph.getText();
            if (StringUtil.isNotEmpty(text) && text.indexOf(key) != -1) {
                List<XWPFRun> runs = paragraph.getRuns();
                // 只保留第一个Run
                for (int i = (runs.size() - 1); i > 0; i--) {
                    paragraph.removeRun(i);
                }
                runs.get(0).setText(text.replace(key, CommonUtil.converObjToStr(value)), 0);
            }
        }
    }

    /**
     * 替换全部图片
     * @param key 替换键
     * @param picturePath 图片路径
     * @param pictureType 图片类型
     * @param width 宽带
```

```
     * @param height 高度
     * @param document Word文档
     * @param paragraphs 段落
     * @param type 类型 header(页眉) footer(页脚)
     */
    public static void replaceAllPictures(String key, String picturePath, int pictureType, int width, int height, XWPFDocument document,
        List<XWPFParagraph> paragraphs, String type) throws Exception {

        width = Units.toEMU(width);
        height = Units.toEMU(height);

        for (XWPFParagraph paragraph : paragraphs) {
            // 待替换文本
            String text = paragraph.getText();
            if (StringUtil.isNotEmpty(text) && text.indexOf(key) != -1) {
                List<XWPFRun> runs = paragraph.getRuns();
                // 只保留第一个Run
                for (int i = (runs.size() - 1); i > 0; i--) {
                    paragraph.removeRun(i);
                }
                runs.get(0).setText(text.replace(key, ""), 0);

                InputStream is = null;
                String blipId = null;
                try {
                    // 每个图片不可以用同一个流
                    is = new FileInputStream(picturePath);
                    if (DOC_HEADER.equals(type)) {
                        // 取得页眉
                        List<XWPFHeader> headerList = document.getHeaderList();
                        blipId = headerList.get(0).addPictureData(is, pictureType);
                    } else if (DOC_FOOTER.equals(type)) {
                        // 取得页脚
                        List<XWPFFooter> footerList = document.getFooterList();
                        blipId = footerList.get(0).addPictureData(is, pictureType);
                    } else {
                        blipId = document.addPictureData(is, pictureType);
                    }
                } finally {
                    if (is != null) {
                        is.close();
                    }
                }
                int id = document.getNextPicNameNumber(pictureType);

                String picXml = ""
                    + "<a:graphic xmlns:a=\"http://schemas.openxmlformats.org/drawingml/2006/main\">"
                    + "   <a:graphicData uri=\"http://schemas.openxmlformats.org/drawingml/2006/picture\">"
                    + "      <pic:pic xmlns:pic=\"http://schemas.openxmlformats.org/drawingml/2006/picture\">"
                    + "         <pic:nvPicPr>"
                    + "            <pic:cNvPr id=\"" + id + "\" name=\"Generated\"/>"
                    + "            <pic:cNvPicPr/>"
                    + "         </pic:nvPicPr>"
                    + "         <pic:blipFill>"
                    + "            <a:blip r:embed=\"" + blipId
                    + "\" xmlns:r=\"http://schemas.openxmlformats.org/officeDocument/2006/relationships\"/>"
                    + "            <a:stretch>"
                    + "               <a:fillRect/>"
                    + "            </a:stretch>"
                    + "         </pic:blipFill>"
                    + "         <pic:spPr>"
                    + "            <a:xfrm>"
                    + "               <a:off x=\"0\" y=\"0\"/>"
                    + "               <a:ext cx=\"" + width + "\" cy=\"" + height + "\"/>"
                    + "            </a:xfrm>"
                    + "            <a:prstGeom prst=\"rect\">"
                    + "               <a:avLst/>"
                    + "            </a:prstGeom>"
                    + "         </pic:spPr>"
                    + "      </pic:pic>"
                    + "   </a:graphicData>"
                    + "</a:graphic>";

                XmlToken xmlToken = XmlToken.Factory.parse(picXml);

                CTInline inline = paragraph.createRun().getCTR().addNewDrawing().addNewInline();
                inline.set(xmlToken);
                inline.setDistT(0);
                inline.setDistB(0);
                inline.setDistL(0);
                inline.setDistR(0);

                CTPositiveSize2D extent = inline.addNewExtent();
                extent.setCx(width);
                extent.setCy(height);

                CTNonVisualDrawingProps docPr = inline.addNewDocPr();
                docPr.setId(id);
                docPr.setName("Picture " + id);
```

```
                docPr.setDescr("XWPFUtil Generated.");
            }
        }
    }

    /**
     * 得到Cell的CTTcPr,不存在则新建
     * @param cell XWPFTableCell
     * @return
     */
    public static CTTcPr getCellCTTcPr(XWPFTableCell cell) {
        CTTc cttc = cell.getCTTc();
        CTTcPr tcPr = cttc.isSetTcPr() ? cttc.getTcPr() : cttc.addNewTcPr();
        return tcPr;
    }

    /**
     * 合并列
     * @param table XWPFTable
     * @param row 要合并的列所在行号
     * @param fromCell 起始单元格号
     * @param toCell 结束单元格号
     */
    public static void mergeCellsHorizontal(XWPFTable table, int row, int fromCell, int toCell) {
        for (int cellIndex = fromCell; cellIndex <= toCell; cellIndex++) {
            XWPFTableCell cell = table.getRow(row).getCell(cellIndex);
            if (cellIndex == fromCell) {
                getCellCTTcPr(cell).addNewHMerge().setVal(STMerge.RESTART);
            } else {
                getCellCTTcPr(cell).addNewHMerge().setVal(STMerge.CONTINUE);
            }
        }
    }

    /**
     * 合并行
     * @param table XWPFTable
     * @param col 要合并的行所在列号
     * @param fromRow 起始列
     * @param toRow 结束列
     */
    public static void mergeCellsVertically(XWPFTable table, int col, int fromRow, int toRow) {
        for (int rowIndex = fromRow; rowIndex <= toRow; rowIndex++) {
            XWPFTableCell cell = table.getRow(rowIndex).getCell(col);
            if (rowIndex == fromRow) {
                getCellCTTcPr(cell).addNewVMerge().setVal(STMerge.RESTART);
            } else {
                getCellCTTcPr(cell).addNewVMerge().setVal(STMerge.CONTINUE);
            }
        }
    }

    /**
     * 设置垂直对齐方式
     * @param cell XWPFTableCell
     * @param vAlign 对齐方式
     */
    public void setCellVAlign(XWPFTableCell cell, STVerticalJc.Enum vAlign) {
        setCellWidthAndVAlign(cell, null, null, vAlign);
    }

    /**
     * 设置列宽和垂直对齐方式
     * @param cell XWPFTableCell
     * @param width 列宽
     * @param typeEnum 类型
     * @param vAlign 垂直对齐方式
     */
    public void setCellWidthAndVAlign(XWPFTableCell cell, String width, STTblWidth.Enum typeEnum, STVerticalJc.Enum vAlign) {
        CTTcPr tcPr = getCellCTTcPr(cell);
        CTTblWidth tcw = tcPr.isSetTcW() ? tcPr.getTcW() : tcPr.addNewTcW();
        if (width != null) {
            tcw.setW(new BigInteger(width));
        }
        if (typeEnum != null) {
            tcw.setType(typeEnum);
        }
        if (vAlign != null) {
            CTVerticalJc vJc = tcPr.isSetVAlign() ? tcPr.getVAlign() : tcPr.addNewVAlign();
            vJc.setVal(vAlign);
        }
    }

    /**
     * 设置表格总宽度与水平对齐方式
     * @param table XWPFTable
     * @param width 宽度
     * @param enumValue 水平对齐方式
     */
```

```java
    public void setTableWidthAndHAlign(XWPFTable table, String width, STJc.Enum enumValue) {
        CTTblPr tblPr = getTableCTTblPr(table);
        CTTblWidth tblWidth = tblPr.isSetTblW() ? tblPr.getTblW() : tblPr.addNewTblW();
        if (enumValue != null) {
            CTJc cTJc = tblPr.addNewJc();
            cTJc.setVal(enumValue);
        }
        if (width != null) {
            tblWidth.setW(new BigInteger(width));
        }
        tblWidth.setType(STTblWidth.DXA);
    }

    /**
     * 得到Table的CTTblPr,不存在则新建
     * @param table XWPFTable
     * @return
     */
    public CTTblPr getTableCTTblPr(XWPFTable table) {
        CTTbl ttbl = table.getCTTbl();
        CTTblPr tblPr = ttbl.getTblPr() == null ? ttbl.addNewTblPr() : ttbl.getTblPr();
        return tblPr;
    }

    /**
     * 获取当前文档总页数
     */
    public int getFilePages() {
        return this.pages;
    }

    /**
     * 删除单元格
     * @param table XWPFTable
     * @param row 行号
     * @param col 列号
     */
    public static void delCell(XWPFTable table, int row, int col) {
        XWPFTableCell removed = table.getRows().get(row).getCell(col);
        removed.getCTTc().newCursor().removeXml();
        table.getRows().get(row).removeCell(col);
    }

    /**
     * 复制表格-复制完需要重新打开word
     * @param document XWPFDocument
     * @param temTable XWPFTable
     */
    public static void copyTable(XWPFDocument document, XWPFTable temTable) {
        CTTbl ctTbl = CTTbl.Factory.newInstance(); // 创建表格xml内容
        ctTbl.set(temTable.getCTTbl()); // 设置表格xml内容
        XWPFTable newTable = new XWPFTable(ctTbl, document); // 创建新表格
        document.createTable(); // 创建一个空表格
        document.setTable(document.getTables().size() - 1, newTable); // 将空表格替换为新表格
    }

    /**
     * 插入分页符
     * @param document XWPFDocument
     */
    public static void insertPageBreak(XWPFDocument document) {
        XWPFParagraph p = document.createParagraph();
        p.setPageBreak(true);
    }

    /**
     * 插入空行
     * @param document XWPFDocument
     */
    public static void insertBr(XWPFDocument document) {
        XWPFParagraph p = document.createParagraph();
        setParagraphSpacingInfo(p, true, "0", "0", "0", "0", true, "240",STLineSpacingRule.EXACT);
    }

    /**
     * 表格添加行
     * @param table XWPFTable
     * @param datanum 条数
     */
    public static void createRow(XWPFTable table, int datanum) {
        for (int i = 0; i < datanum; i++) {
            table.createRow();
        }
    }
    /**
     * 表格复制行-复制完需要重新打开word
     * @param table XWPFTable
     * @param datanum 复制行数
     * @param rowNum 要复制的行号
```

```
     */
    public static void copyRow(XWPFTable table, int datanum, int rowNum) {
        for (int i = 0; i < datanum; i++) {
            CTRow ctRow = CTRow.Factory.newInstance();
            XWPFTableRow row = table.getRow(rowNum);
            ctRow.set(row.getCtRow());
            XWPFTableRow newRow = new XWPFTableRow(ctRow, table);
            table.addRow(newRow);
        }
    }
    /**
     * 依据所传key值和状态设置写入复选框到当前word文件
     * @param key 所要替换的key值
     * @param state 标志位 1为替换uncheckbox 0为替换checkbox
     * @param paragraphs 段落
     * @throws Exception
     */
    public void writeCheckBoxByState(String key, int state, List<XWPFParagraph> paragraphs) throws Exception {
        if (0 == state) {
            replaceAllTexts("#"+key+"#", "☑", paragraphs);
        } else if (1 == state) {
            replaceAllTexts("#"+key+"#", "☐", paragraphs);
        }
    }
    /**
     * @param p XWPFParagraph
     * @param isInsert isInsert
     * @param isNewLine isNewLine
     * @return
     */
    public static XWPFRun getOrAddParagraphFirstRun(XWPFParagraph p, boolean isInsert,
        boolean isNewLine) {
        XWPFRun pRun = null;
        if (isInsert) {
            pRun = p.createRun();
        } else {
            if (p.getRuns() != null && p.getRuns().size() > 0) {
                pRun = p.getRuns().get(0);
            } else {
                pRun = p.createRun();
            }
        }
        if (isNewLine) {
            pRun.addBreak();
        }
        return pRun;
    }
    /**
     * 设置字体信息
     * @param p XWPFParagraph
     * @param pRun XWPFRun
     * @param content 内容
     * @param fontFamily 字体
     * @param fontSize 大小
     */
    public static void setParagraphRunFontInfo(XWPFParagraph p, XWPFRun pRun,
            String content, String fontFamily, String fontSize) {
        CTRPr pRpr = getRunCTRPr(p, pRun);
        if (StringUtils.isNotBlank(content)) {
            pRun.setText(content);
        }
        // 设置字体
        CTFonts fonts = pRpr.isSetRFonts() ? pRpr.getRFonts() : pRpr
                .addNewRFonts();
        fonts.setAscii(fontFamily);
        fonts.setEastAsia(fontFamily);
        fonts.setHAnsi(fontFamily);

        // 设置字体大小
        CTHpsMeasure sz = pRpr.isSetSz() ? pRpr.getSz() : pRpr.addNewSz();
        sz.setVal(new BigInteger(fontSize));

        CTHpsMeasure szCs = pRpr.isSetSzCs() ? pRpr.getSzCs() : pRpr
                .addNewSzCs();
        szCs.setVal(new BigInteger(fontSize));
    }
    /**
     * 得到XWPFRun的CTRPr
     * @param p XWPFParagraph
     * @param pRun XWPFRun
     * @return
     */
    public static CTRPr getRunCTRPr(XWPFParagraph p, XWPFRun pRun) {
        CTRPr pRpr = null;
        if (pRun.getCTR() != null) {
            pRpr = pRun.getCTR().getRPr();
            if (pRpr == null) {
                pRpr = pRun.getCTR().addNewRPr();
            }
```

```
        } else {
            pRpr = p.getCTP().addNewR().addNewRPr();
        }
        return pRpr;
    }
    /**
     * 设置段落间距信息,一行=100 一磅=20
     * @param p XWPFParagraph
     * @param isSpace 空格
     * @param before 段前磅数
     * @param after 段后磅数
     * @param beforeLines 段前行数
     * @param afterLines 段后行数
     * @param isLine 间距
     * @param line
     * @param lineValue
     */
    public static void setParagraphSpacingInfo(XWPFParagraph p, boolean isSpace,
            String before, String after, String beforeLines, String afterLines,
            boolean isLine, String line, STLineSpacingRule.Enum lineValue) {
        CTPPr pPPr = getParagraphCTPPr(p);
        CTSpacing pSpacing = pPPr.getSpacing() != null ? pPPr.getSpacing()
                : pPPr.addNewSpacing();
        if (isSpace) {
            // 段前磅数
            if (before != null) {
                pSpacing.setBefore(new BigInteger(before));
            }
            // 段后磅数
            if (after != null) {
                pSpacing.setAfter(new BigInteger(after));
            }
            // 段前行数
            if (beforeLines != null) {
                pSpacing.setBeforeLines(new BigInteger(beforeLines));
            }
            // 段后行数
            if (afterLines != null) {
                pSpacing.setAfterLines(new BigInteger(afterLines));
            }
        }
        // 间距
        if (isLine) {
            if (line != null) {
                pSpacing.setLine(new BigInteger(line));
            }
            if (lineValue != null) {
                pSpacing.setLineRule(lineValue);
            }
        }
    }
    /**
     * 得到段落CTPPr
     * @param p XWPFParagraph
     * @return
     */
    public static CTPPr getParagraphCTPPr(XWPFParagraph p) {
        CTPPr pPPr = null;
        if (p.getCTP() != null) {
            if (p.getCTP().getPPr() != null) {
                pPPr = p.getCTP().getPPr();
            } else {
                pPPr = p.getCTP().addNewPPr();
            }
        }
        return pPPr;
    }
}
```

Reference: https://www.tuicool.com/articles/Nr2IV3q