

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

copyright

Category Column: poi



1 Subscribe 1 article

Subscribe to
our column

Using POI-3.9

```
1 package com.utils;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.FileOutputStream;
6 import java.io.InputStream;
7 import java.math.BigInteger;
8 import java.util.ArrayList;
9 import java.util.List;
10 import java.util.Map;
11 import java.util.Map.Entry;
12 import java.util.Set;
13 import org.apache.commons.lang.StringUtils;
14 import org.apache.poi.util.Units;
15 import org.apache.poi.xwpf.usermodel.XWPFDocument;
16 import org.apache.poi.xwpf.usermodel.XWPFFooter;
17 import org.apache.poi.xwpf.usermodel.XWPFFooter;
18 import org.apache.poi.xwpf.usermodel.XWPFFooter;
19 import org.apache.poi.xwpf.usermodel.XWPFFooter;
20 import org.apache.poi.xwpf.usermodel.XWPFFooter;
21 import org.apache.poi.xwpf.usermodel.XWPFFooter;
22 import org.apache.poi.xwpf.usermodel.XWPFFooter;
23 import org.apache.poi.xwpf.usermodel.XWPFFooter;
24 import org.apache.poi.xwpf.usermodel.XWPFFooter;
25 import org.apache.xmlbeans.XmlToken;
26 import org.openxmlformats.schemas.drawingml.x2006.main.CTNonVisualDrawingProps;
27 import org.openxmlformats.schemas.drawingml.x2006.main.CTPositiveSize2D;
28 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTInline;
29 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTFonts;
30 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTHpsMeasure;
31 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTJc;
32 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTPPr;
33 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTRPr;
34 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTSpace;
35 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTbl;
36 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTblPr;
37 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTblWidth;
38 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTc;
39 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTcPr;
40 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTVerticalJc;
41 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STJc;
42 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STLineSpacingRule;
43 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STMerge;
44 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STTblWidth;
45 import org.slf4j.Logger;
46 import org.slf4j.LoggerFactory;
47
48 public final class XWPFDUtil {
49     /**
50      * 当前打开文档总页数
51      */
52     private int pages = 1;
53
54     /**
55      * 页眉
56      */
57     public static final String DOC_HEADER = "header";
58
59     /**
60      * 页脚
61      */
62     public static final String DOC_FOOTER = "footer";
63     /**
64      * Logger
65      */
66     private static Logger log = LoggerFactory.getLogger(XWPFDUtil.class);
67
68     /**
69      * 打开Word
70      * @param filePath 文件全路径
71      */
72     public static XWPFDocument open(String filePath) throws Exception {
73         InputStream is = null;
74         try {
75             log.info("POI打开Word文件: " + filePath);
76             is = new FileInputStream(filePath);
77         }
```

```

    }
    return new XWPFDocument(is);
} finally {
    if (is != null) {
        is.close();
    }
}
/** 构造函数 */
public XWPFTUtil() {
};

/**
 * 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<XWPFFParagraph> getAllParagraphs(XWPFDocument document) {
    List<XWPFFParagraph> paragraphs = new ArrayList<XWPFFParagraph>();
    // 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<XWPFFParagraph> getTableParagraphs(XWPFDocument document, XWPFTable table) {
    List<XWPFFParagraph> paragraphs = new ArrayList<XWPFFParagraph>();
    // 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<XWPFFParagraph> getTableColParagraphs(XWPFDocument document, XWPFTable table, int col) {
    List<XWPFFParagraph> paragraphs = new ArrayList<XWPFFParagraph>();
    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

```

```

167 */
168 public static List<XWPFParagraph> getAllHeaderParagraphs(XWPFDocument document) {
169     List<XWPFParagraph> paragraphs = new ArrayList<XWPFParagraph>();
170     // 取得页眉
171     List<XWPFFooter> headerList = document.getHeaderList();
172     // 取得页眉列表外段落
173     List<XWPFParagraph> headerParas = headerList.get(0).getParagraphs();
174     paragraphs.addAll(headerParas);
175     // 取得页眉列表内段落
176     List<XWPFTable> tables = headerList.get(0).getTables();
177     for (XWPFTable table : tables) {
178         List<XWPFTableRow> rows = table.getRows();
179         for (XWPFTableRow row : rows) {
180             List<XWPFTableCell> cells = row.getTableCells();
181             for (XWPFTableCell cell : cells) {
182                 paragraphs.addAll(cell.getParagraphs());
183             }
184         }
185     }
186     return paragraphs;
187 }
188 /**
189 * 取得页脚段落 (含列表)
190 * @param document Word文档
191 */
192 public static List<XWPFParagraph> getAllFooterParagraphs(XWPFDocument document) {
193     List<XWPFParagraph> paragraphs = new ArrayList<XWPFParagraph>();
194     // 取得页脚
195     List<XWPFFooter> footerList = document.getFooterList();
196     // 取得页脚列表外段落
197     List<XWPFParagraph> footerParas = footerList.get(0).getParagraphs();
198     paragraphs.addAll(footerParas);
199     // 取得页脚列表内段落
200     List<XWPFTable> tables = footerList.get(0).getTables();
201     for (XWPFTable table : tables) {
202         List<XWPFTableRow> rows = table.getRows();
203         for (XWPFTableRow row : rows) {
204             List<XWPFTableCell> cells = row.getTableCells();
205             for (XWPFTableCell cell : cells) {
206                 paragraphs.addAll(cell.getParagraphs());
207             }
208         }
209     }
210     return paragraphs;
211 }
212 /**
213 * 替换全部文本
214 * @param textMap 替换信息
215 * @param paragraphs 段落
216 */
217 public static void replaceAllTexts(Map<String, String> textMap, List<XWPFParagraph> paragraphs) {
218     Set<Entry<String, String>> textEntrySet = textMap.entrySet();
219     for (Entry<String, String> entry : textEntrySet) {
220         String key = entry.getKey();
221         String value = CommonUtil.converObjToStr(entry.getValue());
222         // 替换全部文本
223         replaceAllTexts(key, value, paragraphs);
224     }
225 }
226 }
227 /**
228 * 替换全部文本
229 * @param key 替换键
230 * @param value 替换值
231 * @param paragraphs 段落
232 */
233 public static void replaceAllTexts(String key, String value, List<XWPFParagraph> paragraphs) {
234     for (XWPFParagraph paragraph : paragraphs) {
235         // 待替换文本
236         String text = paragraph.getText();
237         if (StringUtil.isNotEmpty(text) && text.indexOf(key) != -1) {
238             List<XWPFRun> runs = paragraph.getRuns();
239             // 只保留第一个run
240             for (int i = (runs.size() - 1); i > 0; i--) {
241                 paragraph.removeRun(i);
242             }
243             runs.get(0).setText(text.replace(key, CommonUtil.converObjToStr(value)), 0);
244         }
245     }
246 }
247 /**
248 * 替换全部图片
249 * @param key 替换键
250 * @param picturePath 图片路径
251 * @param pictureType 图片类型
252 * @param width 宽带
253 */
254 
```

```

256 * @param height 高度
257 * @param document Word文档
258 * @param paragraphs 段落
259 * @param type 类型 header(页眉) footer(页脚)
260 */
261 public static void replaceAllPictures(String key, String picturePath, int pictureType, int width, int height, XWPFDocument document,
262 List<XWPPParagraph> paragraphs, String type) throws Exception {
263
264     width = Units.toEMU(width);
265     height = Units.toEMU(height);
266
267     for (XWPPParagraph paragraph : paragraphs) {
268         // 待替换文本
269         String text = paragraph.getText();
270         if (StringUtil.isNotEmpty(text) && text.indexOf(key) != -1) {
271             List<XWPRun> runs = paragraph.getRuns();
272             // 只保留第一个run
273             for (int i = (runs.size() - 1); i > 0; i--) {
274                 paragraph.removeRun(i);
275             }
276             runs.get(0).setText(text.replace(key, ""), 0);
277
278             InputStream is = null;
279             String blipId = null;
280             try {
281                 // 每个图片不可以用同一个流
282                 is = new FileInputStream(picturePath);
283                 if (DOC_HEADER.equals(type)) {
284                     // 取得页眉
285                     List<XWPFFooter> headerList = document.getHeaderList();
286                     blipId = headerList.get(0).addPictureData(is, pictureType);
287                 } else if (DOC_FOOTER.equals(type)) {
288                     // 取得页脚
289                     List<XWPFFooter> footerList = document.getFooterList();
290                     blipId = footerList.get(0).addPictureData(is, pictureType);
291                 } else {
292                     blipId = document.addPictureData(is, pictureType);
293                 }
294             } finally {
295                 if (is != null) {
296                     is.close();
297                 }
298             }
299             int id = document.getNextPicNameNumber(pictureType);
300
301             String picXml = ""
302                 + "<a:graphic xmlns:a=\"http://schemas.openxmlformats.org/drawingml/2006/main\">"
303                 + "    <a:graphicData uri=\"http://schemas.openxmlformats.org/drawingml/2006/picture\">"
304                 + "        <pic:pic xmlns:pic=\"http://schemas.openxmlformats.org/drawingml/2006/picture\">"
305                 + "            <pic:vPicPr>"
306                 + "                <pic:cNvPr id=\"" + id + "\" name=\"Generated\"/>"
307                 + "                <pic:cNvPicPr/>"
308                 + "            </pic:vPicPr>"
309                 + "            <pic:blipFill>"
310                 + "                <a:blip r:embed=\"" + blipId
311                 + "                xmlns:r=\"http://schemas.openxmlformats.org/officeDocument/2006/relationships\"/>"
312                 + "                    <a:stretch>"
313                 + "                        <a:fillRect/>"
314                 + "                            </a:stretch>"
315                 + "                    </pic:blipFill>"
316                 + "                    <pic:spPr>"
317                 + "                        <a:xfrm>"
318                         <a:off x="0" y="0"/>
319                         <a:ext cx=\"" + width + "\" cy=\"" + height + "\"/>
320                     </a:xfrm>
321                     <a:prstGeom prst=\"rect\"/>
322                         <a:avLst/>
323                     </a:prstGeom>
324                 </pic:spPr>
325             </pic:pic>
326         </a:graphicData>
327     </a:graphic>";
328
329     XmlToken xmlToken = XmlToken.Factory.parse(picXml);
330
331     CTInline inline = paragraph.createRun().getCTR().addNewDrawing().addNewInline();
332     inline.set(xmlToken);
333     inline.setDistT(0);
334     inline.setDistB(0);
335     inline.setDistL(0);
336     inline.setDistR(0);
337
338     CTPositiveSize2D extent = inline.addNewExtent();
339     extent.setCx(width);
340     extent.setCy(height);
341
342     CTNonVisualDrawingProps docPr = inline.addNewDocPr();
343     docPr.setId(id);
344     docPr.setName("Picture " + id);

```

```

345         docPr.setDescr("XWPUtil Generated.");
346     }
347 }
348 }
349 /**
350 * 得到Cell的CTTcPr,不存在则新建
351 * @param cell XWPFTableCell
352 * @return
353 */
354 public static CTTcPr getCellCTTcPr(XWPFTableCell cell) {
355     CTTc ctte = cell.getCTTc();
356     CTTcPr tcPr = ctte.isSetTcPr() ? ctte.getTcPr() : ctte.addNewTcPr();
357     return tcPr;
358 }
359 }
360 /**
361 * 合并列
362 * @param table XWPFTable
363 * @param row 要合并的列所在行号
364 * @param fromCell 起始单元格号
365 * @param toCell 结束单元格号
366 */
367 public static void mergeCellsHorizontal(XWPFTable table, int row, int fromCell, int toCell) {
368     for (int cellIndex = fromCell; cellIndex <= toCell; cellIndex++) {
369         XWPFTableCell cell = table.getRow(row).getCell(cellIndex);
370         if (cellIndex == fromCell) {
371             getCellCTTcPr(cell).addNewHMerge().setVal(STMerge.RESTART);
372         } else {
373             getCellCTTcPr(cell).addNewHMerge().setVal(STMerge.CONTINUE);
374         }
375     }
376 }
377 }
378 /**
379 * 合并行
380 * @param table XWPFTable
381 * @param col 要合并的行所在列号
382 * @param fromRow 起始列
383 * @param toRow 结束列
384 */
385 public static void mergeCellsVertically(XWPFTable table, int col, int fromRow, int toRow) {
386     for (int rowIndex = fromRow; rowIndex <= toRow; rowIndex++) {
387         XWPFTableCell cell = table.getRow(rowIndex).getCell(col);
388         if (rowIndex == fromRow) {
389             getCellCTTcPr(cell).addNewVMerge().setVal(STMerge.RESTART);
390         } else {
391             getCellCTTcPr(cell).addNewVMerge().setVal(STMerge.CONTINUE);
392         }
393     }
394 }
395 }
396 /**
397 * 设置垂直对齐方式
398 * @param cell XWPFTableCell
399 * @param vAlign 对齐方式
400 */
401 public void setCellVAlign(XWPFTableCell cell, STVerticalJc.Enum vAlign) {
402     setCellWidthAndVAlign(cell, null, null, vAlign);
403 }
404 }
405 /**
406 * 设置列宽和垂直对齐方式
407 * @param cell XWPFTableCell
408 * @param width 列宽
409 * @param typeEnum 类型
410 * @param vAlign 垂直对齐方式
411 */
412 public void setCellWidthAndVAlign(XWPFTableCell cell, String width, STTblWidth.Enum typeEnum, STVerticalJc.Enum vAlign) {
413     CTTcPr tcPr = getCellCTTcPr(cell);
414     CTtblWidth tcw = tcPr.isSetTcW() ? tcPr.getTcW() : tcPr.addNewTcW();
415     if (width != null) {
416         tcw.setW(new BigInteger(width));
417     }
418     if (typeEnum != null) {
419         tcw.setType(typeEnum);
420     }
421     if (vAlign != null) {
422         CTVerticalJc vJc = tcPr.isSetVAlign() ? tcPr.getVAlign() : tcPr.addNewVAlign();
423         vJc.setVal(vAlign);
424     }
425 }
426 }
427 /**
428 * 设置表格总宽度与水平对齐方式
429 * @param table XWPFTable
430 * @param width 宽度
431 * @param enumValue 水平对齐方式
432 */
433

```

```

434     public void setTableWidthAndHAlign(XWPFTable table, String width, STJc.Enum enumValue) {
435         CTTblPr tblPr = getTableCTTtblPr(table);
436         CTTblWidth tblWidth = tblPr.isSetTblW() ? tblPr.getTblW() : tblPr.addNewTblW();
437         if (enumValue != null) {
438             CTJc ctJc = tblPr.addNewJc();
439             ctJc.setVal(enumValue);
440         }
441         if (width != null) {
442             tblWidth.setW(new BigInteger(width));
443         }
444         tblWidth.setType(STtblWidth.DXA);
445     }
446
447     /**
448      * 得到Table的CTTtblPr,不存在则新建
449      * @param table XWPFTable
450      * @return
451     */
452     public CTTtblPr getTableCTTtblPr(XWPFTable table) {
453         CTTtbl ttbl = table.getCTTtbl();
454         CTTtblPr tblPr = ttbl.getTblPr() == null ? ttbl.addNewTblPr() : ttbl.getTblPr();
455         return tblPr;
456     }
457
458     /**
459      * 获取当前文档总页数
460      */
461     public int getFilePages() {
462         return this.pages;
463     }
464
465     /**
466      * 删除单元格
467      * @param table XWPFTable
468      * @param row 行号
469      * @param col 列号
470     */
471     public static void delCell(XWPFTable table, int row, int col) {
472         XWPFTableCell removed = table.getRows().get(row).getCell(col);
473         removed.getCTTc().newCursor().removeXml();
474         table.getRows().get(row).removeCell(col);
475     }
476
477     /**
478      * 复制表格-复制完需要重新打开word
479      * @param document XWPFDocument
480      * @param temTable XWPFTable
481     */
482     public static void copyTable(XWPFDocument document, XWPFTable temTable) {
483         CTTtbl ctTtbl = CTTtbl.Factory.newInstance(); // 创建表格xml内容
484         ctTtbl.set(temTable.getCTTtbl()); // 设置表格xml内容
485         XWPFTable newTable = new XWPFTable(ctTtbl, document); // 创建新表格
486         document.createTable(); // 创建一个空表格
487         document.setTable(document.getTables().size() - 1, newTable); // 将空表格替换为新表格
488     }
489
490     /**
491      * 插入分页符
492      * @param document XWPFDocument
493     */
494     public static void insertPageBreak(XWPFDocument document) {
495         XWPFFParagraph p = document.createParagraph();
496         p.setPageBreak(true);
497     }
498
499     /**
500      * 插入空行
501      * @param document XWPFDocument
502     */
503     public static void insertBr(XWPFDocument document) {
504         XWPFFParagraph p = document.createParagraph();
505         setParagraphSpacingInfo(p, true, "0", "0", "0", "0", true, "240", STLineSpacingRule.EXACT);
506     }
507
508     /**
509      * 表格添加行
510      * @param table XWPFTable
511      * @param datanum 条数
512     */
513     public static void createRow(XWPFTable table, int datanum) {
514         for (int i = 0; i < datanum; i++) {
515             table.createRow();
516         }
517     }
518     /**
519      * 表格复制行-复制完需要重新打开word
520      * @param table XWPFTable
521      * @param datanum 复制行数
522      * @param rowNum 要复制的行号

```

```

523     */
524     public static void copyRow(XWPFTable table, int datanum, int rowNum) {
525         for (int i = 0; i < datanum; i++) {
526             CTRow ctRow = CTRow.Factory.newInstance();
527             XWPFTableRow row = table.getRow(rowNum);
528             ctRow.set(row.getCTR());
529             XWPFTableRow newRow = new XWPFTableRow(ctRow, table);
530             table.addRow(newRow);
531         }
532     }
533     /**
534      * 依据所传key值和状态设置写入复选框到当前word文件
535      * @param key 所要替换的key值
536      * @param state 标志位 1为替换uncheckbox 0为替换checkbox
537      * @param paragraphs 段落
538      * @throws Exception
539      */
540     public void writeCheckBoxByState(String key, int state, List<XWPFParagraph> paragraphs) throws Exception {
541         if (0 == state) {
542             replaceAllTexts("#"+key+"#", "☒", paragraphs);
543         } else if (1 == state) {
544             replaceAllTexts("#"+key+"#", "☐", paragraphs);
545         }
546     }
547     /**
548      * @param p XWPFParagraph
549      * @param isInsert isInsert
550      * @param is.NewLine isNewLine
551      * @return
552      */
553     public static XWPFRun getOrAddParagraphFirstRun(XWPFParagraph p, boolean isInsert,
554             boolean isNewLine) {
555         XWPFRun pRun = null;
556         if (isInsert) {
557             pRun = p.createRun();
558         } else {
559             if (p.getRuns() != null && p.getRuns().size() > 0) {
560                 pRun = p.getRuns().get(0);
561             } else {
562                 pRun = p.createRun();
563             }
564         }
565         if (is.NewLine) {
566             pRun.addBreak();
567         }
568         return pRun;
569     }
570     /**
571      * 设置字体信息
572      * @param p XWPFParagraph
573      * @param pRun XWPFRun
574      * @param content 内容
575      * @param fontFamily 字体
576      * @param fontSize 大小
577      */
578     public static void setParagraphRunFontInfo(XWPFParagraph p, XWPFRun pRun,
579             String content, String fontFamily, String fontSize) {
580         CTRPr pRPr = getRunCTRPr(p, pRun);
581         if (StringUtils.isNotBlank(content)) {
582             pRun.setText(content);
583         }
584         // 设置字体
585         CTFonts fonts = pRPr.isSetRFonts() ? pRPr.getRFonts() : pRPr
586             .addNewRFonts();
587         fonts.setAscii(fontFamily);
588         fonts.setEastAsia(fontFamily);
589         fonts.setHAnsi(fontFamily);
590
591         // 设置字体大小
592         CTHeightMeasure sz = pRPr.isSetSz() ? pRPr.getSz() : pRPr.addNewSz();
593         sz.setVal(new BigInteger(fontSize));
594
595         CTHpsMeasure szCs = pRPr.isSetSzCs() ? pRPr.getSzCs() : pRPr
596             .addNewSzCs();
597         szCs.setVal(new BigInteger(fontSize));
598     }
599     /**
600      * 得到XWPFRun的CTRPr
601      * @param p XWPFParagraph
602      * @param pRun XWPFRun
603      * @return
604      */
605     public static CTRPr getRunCTRPr(XWPFParagraph p, XWPFRun pRun) {
606         CTRPr pRPr = null;
607         if (pRun.getCTR() != null) {
608             pRPr = pRun.getCTR().getRPr();
609             if (pRPr == null) {
610                 pRPr = pRun.getCTR().addNewRPr();
611             }
612         }

```

```

612     } else {
613         pRPr = p.getCTP().addNewRPr().addNewRPr();
614     }
615     return pRPr;
616 }
617 /**
618 * 设置段落间距信息,一行=100 一磅=20
619 * @param p XWPFParagraph
620 * @param isSpace 空格
621 * @param before 段前磅数
622 * @param after 段后磅数
623 * @param beforeLines 段前行数
624 * @param afterLines 段后行数
625 * @param isLine [间距]
626 * @param line
627 * @param lineValue
628 */
629 public static void setParagraphSpacingInfo(XWPFParagraph p, boolean isSpace,
630     String after, String before, String beforeLines, String afterLines,
631     boolean isLine, String line, STLineSpacingRule.Enum lineValue) {
632     CTPPr pPPr = getParagraphCTPPr(p);
633     CTSpacing pSpacing = pPPr.getSpacing() != null ? pPPr.getSpacing()
634         : pPPr.addNewSpacing();
635     if (isSpace) {
636         // 段前磅数
637         if (before != null) {
638             pSpacing.setBefore(new BigInteger(before));
639         }
640         // 段后磅数
641         if (after != null) {
642             pSpacing.setAfter(new BigInteger(after));
643         }
644         // 段前行数
645         if (beforeLines != null) {
646             pSpacing.setBeforeLines(new BigInteger(beforeLines));
647         }
648         // 段后行数
649         if (afterLines != null) {
650             pSpacing.setAfterLines(new BigInteger(afterLines));
651         }
652     }
653     // 间距
654     if (isLine) {
655         if (line != null) {
656             pSpacing.setLine(new BigInteger(line));
657         }
658         if (lineValue != null) {
659             pSpacing.setLineRule(lineValue);
660         }
661     }
662 }
663 /**
664 * 得到段落CTPPr
665 * @param p XWPFParagraph
666 * @return
667 */
668 public static CTPPr getParagraphCTPPr(XWPFParagraph p) {
669     CTPPr pPPr = null;
670     if (p.getCTP() != null) {
671         if (p.getCTP().getPPr() != null) {
672             pPPr = p.getCTP().getPPr();
673         } else {
674             pPPr = p.getCTP().addNewPPr();
675         }
676     }
677     return pPPr;
678 }
679 }

```

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