

POI operates word2007 through XWPFDocument

原创 A lazy programmer 1 Posted on 2018-04-16 17:54:19 0 Read 3.4w 0 Collection 60 0 Likes 8

Category Column: poi

copyright



poi This column includes this content

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.XWPFHeader;
18 import org.apache.poi.xwpf.usermodel.XWPFParagraph;
19 import org.apache.poi.xwpf.usermodel.XWPFRun;
20 import org.apache.poi.xwpf.usermodel.XWPFTable;
21 import org.apache.poi.xwpf.usermodel.XWPFTableCell;
22 import org.apache.poi.xwpf.usermodel.XWPFTableRow;
23 import org.apache.xmlbeans.XmlToken;
24 import org.openxmlformats.schemas.drawingml.x2006.main.CTNonVisualDrawingProps;
25 import org.openxmlformats.schemas.drawingml.x2006.main.CTPositiveSize2D;
26 import org.openxmlformats.schemas.drawingml.x2006.wordprocessingDrawing.CTInline;
27 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTFonts;
28 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTHpsMeasure;
29 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTJc;
30 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTPPr;
31 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTRPr;
32 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTRow;
33 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTSpacing;
34 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTbl;
35 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTblPr;
36 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTblWidth;
37 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTC;
38 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTTCPr;
39 import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTVerticalJc;
40 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STJc;
41 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STLineSpacingRule;
42 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STMerge;
43 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STTblWidth;
44 import org.openxmlformats.schemas.wordprocessingml.x2006.main.STVerticalJc;
45 import org.slf4j.Logger;
46 import org.slf4j.LoggerFactory;
47
48 public final class XWPFUtil {
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(XWPFUtil.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 XWPUtil() {
};

/**
 * 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<XWPFPParagraph> getAllParagraphs(XWPFDocument document) {
    List<XWPFPParagraph> paragraphs = new ArrayList<XWPFPParagraph>();
    // 列表外段落
    paragraphs.addAll(document.getParagraphs());
    // 列表内段落
    List<XWPFTTable> tables = document.getTables();
    for (XWPFTTable table : tables) {
        List<XWPFTTableRow> rows = table.getRows();
        for (XWPFTTableRow row : rows) {
            List<XWPFTTableCell> cells = row.getTableCells();
            for (XWPFTTableCell cell : cells) {
                paragraphs.addAll(cell.getParagraphs());
            }
        }
    }
    return paragraphs;
}

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

```

```

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<XWPFParagraph> paragraphs, String type) throws Exception {
263
264     width = Units.toEMU(width);
265     height = Units.toEMU(height);
266
267     for (XWPFParagraph paragraph : paragraphs) {
268         // 待替换文本
269         String text = paragraph.getText();
270         if (StringUtil.isEmpty(text) && text.indexOf(key) != -1) {
271             List<XWPFRun> 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<XWPFHeader> 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:nvPicPr>"
306                 + "                <pic:cNvPr id=\"" + id + "\" name=\"Generated\"/>"
307                 + "                <pic:cNvPicPr/>"
308                 + "            </pic:nvPicPr>"
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("XWPFUtil Generated.");
346     }
347 }
348 }
349
350 /**
351  * 得到Cell的CTTcPr,不存在则新建
352  * @param cell XWPFTableCell
353  * @return
354  */
355 public static CTTcPr getCellCTTcPr(XWPFTableCell cell) {
356     CTTc cttc = cell.getCTTc();
357     CTTcPr tcPr = cttc.isSetTcPr() ? cttc.getTcPr() : cttc.addNewTcPr();
358     return tcPr;
359 }
360
361 /**
362  * 合并列
363  * @param table XWPFTable
364  * @param row 要合并的列所在行号
365  * @param fromCell 起始单元格号
366  * @param toCell 结束单元格号
367  */
368 public static void mergeCellsHorizontal(XWPFTable table, int row, int fromCell, int toCell) {
369     for (int cellIndex = fromCell; cellIndex <= toCell; cellIndex++) {
370         XWPFTableCell cell = table.getRow(row).getCell(cellIndex);
371         if (cellIndex == fromCell) {
372             getCellCTTcPr(cell).addNewHMerge().setVal(STMerge.RESTART);
373         } else {
374             getCellCTTcPr(cell).addNewHMerge().setVal(STMerge.CONTINUE);
375         }
376     }
377 }
378
379 /**
380  * 合并行
381  * @param table XWPFTable
382  * @param col 要合并的行所在列号
383  * @param fromRow 起始列
384  * @param toRow 结束列
385  */
386 public static void mergeCellsVertically(XWPFTable table, int col, int fromRow, int toRow) {
387     for (int rowIndex = fromRow; rowIndex <= toRow; rowIndex++) {
388         XWPFTableCell cell = table.getRow(rowIndex).getCell(col);
389         if (rowIndex == fromRow) {
390             getCellCTTcPr(cell).addNewVMerge().setVal(STMerge.RESTART);
391         } else {
392             getCellCTTcPr(cell).addNewVMerge().setVal(STMerge.CONTINUE);
393         }
394     }
395 }
396
397 /**
398  * 设置垂直对齐方式
399  * @param cell XWPFTableCell
400  * @param vAlign 对齐方式
401  */
402 public void setCellVAlign(XWPFTableCell cell, STVerticalJc.Enum vAlign) {
403     setCellWidthAndVAlign(cell, null, null, vAlign);
404 }
405
406 /**
407  * 设置列宽和垂直对齐方式
408  * @param cell XWPFTableCell
409  * @param width 列宽
410  * @param typeEnum 类型
411  * @param vAlign 垂直对齐方式
412  */
413 public void setCellWidthAndVAlign(XWPFTableCell cell, String width, STTblWidth.Enum typeEnum, STVerticalJc.Enum vAlign) {
414     CTTcPr tcPr = getCellCTTcPr(cell);
415     CTTblWidth tcw = tcPr.isSetTcW() ? tcPr.getTcW() : tcPr.addNewTcW();
416     if (width != null) {
417         tcw.setW(new BigInteger(width));
418     }
419     if (typeEnum != null) {
420         tcw.setType(typeEnum);
421     }
422     if (vAlign != null) {
423         CTVerticalJc vJc = tcPr.isSetVAlign() ? tcPr.getVAlign() : tcPr.addNewVAlign();
424         vJc.setVal(vAlign);
425     }
426 }
427
428 /**
429  * 设置表格总宽度与水平对齐方式
430  * @param table XWPFTable
431  * @param width 宽度
432  * @param enumValue 水平对齐方式
433  */

```

```

434 public void setTableWidthAndHAlign(XWPFTable table, String width, STJc.Enum enumValue) {
435     CTTblPr tblPr = getTableCTTblPr(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CTTblPr,不存在则新建
449  * @param table XWPFTable
450  * @return
451  */
452 public CTTblPr getTableCTTblPr(XWPFTable table) {
453     CTTbl ttbl = table.getCTTbl();
454     CTTblPr 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     CTTbl ctTbl = CTTbl.Factory.newInstance(); // 创建表格xml内容
484     ctTbl.set(temTable.getCTTbl()); // 设置表格xml内容
485     XWPFTable newTable = new XWPFTable(ctTbl, 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     XWPFPParagraph 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     XWPFPParagraph p = document.createParagraph();
505     setParagraphSpacingInfo(p, true, "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 /**
520  * 表格复制行-复制完需要重新打开word
521  * @param table XWPFTable
522  * @param datanum 复制行数
523  * @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.getCTRow());
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unchecked 0为替换checkbox
537   * @param paragraphs 段落
538   * @throws Exception
539   */
540  public void writeCheckBoxByState(String key, int state, List<XWPFPParagraph> 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 XWPFPParagraph
549   * @param isInsert isInsert
550   * @param isNewLine isNewLine
551   * @return
552   */
553  public static XWPFRun getOrAddParagraphFirstRun(XWPFPParagraph 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 (isNewLine) {
566          pRun.addBreak();
567      }
568      return pRun;
569  }
570  /**
571   * 设置字体信息
572   * @param p XWPFPParagraph
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(XWPFPParagraph 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      CTHpsMeasure 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 XWPFPParagraph
602   * @param pRun XWPFRun
603   * @return
604   */
605  public static CTRPr getRunCTRPr(XWPFPParagraph 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      }

```

```

614     } else {
615         pRPr = p.getCTP().addNewR().addNewRPr();
616     }
617     return pRPr;
618 }
619 /**
620  * 设置段落间距信息,一行=100 一磅=20
621  * @param p XWPFParagraph
622  * @param isSpace 空格
623  * @param before 段前磅数
624  * @param after 段后磅数
625  * @param beforeLines 段前行数
626  * @param afterLines 段后行数
627  * @param isLine 间距
628  * @param line
629  * @param lineValue
630 */
631 public static void setParagraphSpacingInfo(XWPFParagraph p, boolean isSpace,
632     String before, String after, String beforeLines, String afterLines,
633     boolean isLine, String line, STLineSpacingRule.Enum lineValue) {
634     CTPPr pPPr = getParagraphCTPPr(p);
635     CTSpacing pSpacing = pPPr.getSpacing() != null ? pPPr.getSpacing()
636         : pPPr.addNewSpacing();
637     if (isSpace) {
638         // 段前磅数
639         if (before != null) {
640             pSpacing.setBefore(new BigInteger(before));
641         }
642         // 段后磅数
643         if (after != null) {
644             pSpacing.setAfter(new BigInteger(after));
645         }
646         // 段前行数
647         if (beforeLines != null) {
648             pSpacing.setBeforeLines(new BigInteger(beforeLines));
649         }
650         // 段后行数
651         if (afterLines != null) {
652             pSpacing.setAfterLines(new BigInteger(afterLines));
653         }
654     }
655     // 间距
656     if (isLine) {
657         if (line != null) {
658             pSpacing.setLine(new BigInteger(line));
659         }
660         if (lineValue != null) {
661             pSpacing.setLineRule(lineValue);
662         }
663     }
664 }
665 /**
666  * 得到段落CTPPr
667  * @param p XWPFParagraph
668  * @return
669 */
670 public static CTPPr getParagraphCTPPr(XWPFParagraph p) {
671     CTPPr pPPr = null;
672     if (p.getCTP() != null) {
673         if (p.getCTP().getPPr() != null) {
674             pPPr = p.getCTP().getPPr();
675         } else {
676             pPPr = p.getCTP().addNewPPr();
677         }
678     }
679     return pPPr;
680 }

```

Reference: <https://www.tuicool.com/articles/Nr2lV3q>



A lazy programmer

focus on