**Doc，docx 格式转html 和 pdf**

有时我们有这样的需求，某一份Word文档需要在浏览器以html形式预览或用户希望以pdf格式导出word文档，这时候我们就需要将word文档转化成相应的格式。转化起来其实还是有点麻烦的，word 转pdf 是不能直接转的，要先将word转成html,然后再将html转化成pdf,由于poi提供的word转html方式转出来的html不很标准，存在标签没有闭合的情况，所以需要使用Jsoup将poi生成的html修改正成标准格式的html后再转pdf,下面是相关依赖和代码：

下面的Maven 依赖必不可少:

<**dependency**>  
 <**groupId**>org.apache.poi</**groupId**>  
 <**artifactId**>poi</**artifactId**>  
 <**version**>3.17</**version**>  
</**dependency**>  
 <**dependency**>  
 <**groupId**>org.apache.poi</**groupId**>  
 <**artifactId**>poi-ooxml</**artifactId**>  
 <**version**>3.17</**version**>  
</**dependency**>  
 <**dependency**>  
 <**groupId**>org.apache.poi</**groupId**>  
 <**artifactId**>poi-ooxml-schemas</**artifactId**>  
 <**version**>3.17</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.apache.poi</**groupId**>  
 <**artifactId**>ooxml-schemas</**artifactId**>  
 <**version**>1.3</**version**>  
</**dependency**>  
*<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-scratchpad -->*<**dependency**>  
 <**groupId**>org.apache.poi</**groupId**>  
 <**artifactId**>poi-scratchpad</**artifactId**>  
 <**version**>3.17</**version**>  
</**dependency**>  
  
  
 <**dependency**>  
 <**groupId**>org.apache.poi</**groupId**>  
 <**artifactId**>ooxml-schemas</**artifactId**>  
 <**version**>1.3</**version**>  
</**dependency**>  
  
  
  
  
 <**dependency**>  
 <**groupId**>net.sf.jtidy</**groupId**>  
 <**artifactId**>jtidy</**artifactId**>  
 <**version**>r938</**version**>  
</**dependency**>  
  
  
<**dependency**>  
 <**groupId**>com.itextpdf</**groupId**>  
 <**artifactId**>itextpdf</**artifactId**>  
 <**version**>5.5.12</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>com.itextpdf</**groupId**>  
 <**artifactId**>itext-asian</**artifactId**>  
 <**version**>5.2.0</**version**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>log4j</**groupId**>  
 <**artifactId**>log4j</**artifactId**>  
 <**version**>1.2.16</**version**>  
</**dependency**>  
  
  
  
  
  
<**dependency**>  
 <**groupId**>org.xhtmlrenderer</**groupId**>  
 <**artifactId**>core-renderer</**artifactId**>  
 <**version**>R8</**version**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>com.itextpdf.tool</**groupId**>  
 <**artifactId**>xmlworker</**artifactId**>  
 <**version**>5.5.6</**version**>  
</**dependency**>  
  
  
<**dependency**>  
 <**groupId**>org.apache.xmlbeans</**groupId**>  
 <**artifactId**>xmlbeans</**artifactId**>  
 <**version**>2.6.0</**version**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>fr.opensagres.xdocreport</**groupId**>  
 <**artifactId**>org.apache.poi.xwpf.converter.core-gae</**artifactId**>  
 <**version**>1.0.2</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>fr.opensagres.xdocreport</**groupId**>  
 <**artifactId**>org.apache.poi.xwpf.converter.xhtml-gae</**artifactId**>  
 <**version**>1.0.2</**version**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>org.jsoup</**groupId**>  
 <**artifactId**>jsoup</**artifactId**>  
 <**version**>1.8.2</**version**>  
</**dependency**>

**import** com.lowagie.text.pdf.BaseFont;  
**import** org.apache.poi.hwpf.HWPFDocument;  
**import** org.apache.poi.hwpf.converter.PicturesManager;  
**import** org.apache.poi.hwpf.converter.WordToHtmlConverter;  
**import** org.apache.poi.hwpf.usermodel.PictureType;  
**import** org.apache.poi.xwpf.converter.core.BasicURIResolver;  
**import** org.apache.poi.xwpf.converter.core.FileImageExtractor;  
**import** org.apache.poi.xwpf.converter.xhtml.XHTMLConverter;  
**import** org.apache.poi.xwpf.converter.xhtml.XHTMLOptions;  
**import** org.apache.poi.xwpf.usermodel.XWPFDocument;  
**import** org.jsoup.Jsoup;  
**import** org.w3c.dom.Document;  
**import** org.w3c.tidy.Tidy;  
**import** org.xhtmlrenderer.pdf.ITextFontResolver;  
**import** org.xhtmlrenderer.pdf.ITextRenderer;  
  
**import** javax.xml.parsers.DocumentBuilderFactory;  
**import** javax.xml.transform.OutputKeys;  
**import** javax.xml.transform.Transformer;  
**import** javax.xml.transform.TransformerFactory;  
**import** javax.xml.transform.dom.DOMSource;  
**import** javax.xml.transform.stream.StreamResult;  
**import** java.io.\*;  
**import** java.nio.file.Files;  
**import** java.nio.file.Path;  
**import** java.nio.file.Paths;  
  
**public class** FileConverter {  
 */\*  
 \* word文件转成html文件  
 \* sourceFilePath:源word文件路径  
 \* targetFilePosition:转化后生成的html文件路径  
 \*/* **public void** wordToHtml(String sourceFilePath, String targetFilePosition) **throws** Exception {  
 **if** (**".docx"**.equals(sourceFilePath.substring(sourceFilePath.lastIndexOf(**"."**, sourceFilePath.length())))) {  
 docxToHtml(sourceFilePath, targetFilePosition);  
 } **else if** (**".doc"**.equals(sourceFilePath.substring(sourceFilePath.lastIndexOf(**"."**, sourceFilePath.length())))) {  
 docToHtml(sourceFilePath, targetFilePosition);  
 } **else** {  
 **throw new** RuntimeException(**"文件格式不正确"**);  
 }  
  
 }  
  
 */\*  
 \* doc转换为html  
 \* sourceFilePath:源word文件路径  
 \* targetFilePosition:生成的html文件路径  
 \*/* **private void** docToHtml(String sourceFilePath, String targetFilePosition) **throws** Exception {  
 **final** Path imagePath = Paths.*get*(targetFilePosition).getParent().resolve(**"image"**);  
 **if** (!Files.*exists*(imagePath)){  
 Files.*createDirectories*(imagePath);  
 }  
 HWPFDocument wordDocument = **new** HWPFDocument(**new** FileInputStream(sourceFilePath));  
  
  
 Document document = DocumentBuilderFactory.*newInstance*().newDocumentBuilder().newDocument();  
 WordToHtmlConverter wordToHtmlConverter = **new** WordToHtmlConverter(document);  
  
 *//提取word文档中的图片， 保存图片，并返回图片的相对路径* wordToHtmlConverter.setPicturesManager(**new** PicturesManager() {  
 @Override  
 **public** String savePicture(**byte**[] content, PictureType pictureType, String name, **float** width, **float** height) {  
 **try** (FileOutputStream out = **new** FileOutputStream(imagePath.resolve(name).toString())) {  
 out.write(content);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return "../tmp/image/"** + name;  
 }  
 });  
 wordToHtmlConverter.processDocument(wordDocument);  
 Document htmlDocument = wordToHtmlConverter.getDocument();  
 DOMSource domSource = **new** DOMSource(htmlDocument);  
 StreamResult streamResult = **new** StreamResult(**new** File(targetFilePosition));  
 TransformerFactory tf = TransformerFactory.*newInstance*();  
 Transformer serializer = tf.newTransformer();  
 serializer.setOutputProperty(OutputKeys.***ENCODING***, **"UTF-8"**);  
 serializer.setOutputProperty(OutputKeys.***INDENT***, **"yes"**);  
 serializer.setOutputProperty(OutputKeys.***METHOD***, **"html"**);  
 serializer.transform(domSource, streamResult);  
 }  
  
  
  
  
  
  
 */\*  
 \* docx转换为html  
 \* sourceFilePath:源word文件路径  
 \* targetFileName:生成的html文件路径  
 \*/* **private void** docxToHtml(String sourceFilePath, String targetFileName) **throws** Exception {  
 String imagePathStr = Paths.*get*(targetFileName).getParent().resolve(**"../tmp/image/word/media"**).toString();  
 System.***out***.println(imagePathStr);  
 OutputStreamWriter outputStreamWriter = **null**;  
 **try** {  
 XWPFDocument document = **new** XWPFDocument(**new** FileInputStream(sourceFilePath));  
 XHTMLOptions options = XHTMLOptions.*create*();  
 *// 提取文档中的图片，存放图片的文件夹* options.setExtractor(**new** FileImageExtractor(**new** File(imagePathStr)));  
 options.setIgnoreStylesIfUnused(**false**);  
 options.setFragment(**true**);  
 *// html中图片的路径* options.URIResolver(**new** BasicURIResolver(**"../tmp/image/word/media"**));  
 outputStreamWriter = **new** OutputStreamWriter(**new** FileOutputStream(targetFileName), **"UTF-8"**);  
 XHTMLConverter xhtmlConverter = (XHTMLConverter) XHTMLConverter.*getInstance*();  
 xhtmlConverter.convert(document, outputStreamWriter, options);  
 } **finally** {  
 **if** (outputStreamWriter != **null**) {  
 outputStreamWriter.close();  
 }  
 }  
 }  
  
  
 */\*  
 \* txt文档转html  
 filePath:txt原文件路径  
 htmlPosition:转化后生成的html路径  
  
 \*/* **public void** txtToHtml(String filePath, String htmlPosition) {  
 **try** {  
 String encoding = **"GBK"**;  
 File file = **new** File(filePath);  
 **if** (file.isFile() && file.exists()) { *// 判断文件是 否存在* InputStreamReader read = **new** InputStreamReader(**new** FileInputStream(file), encoding);  
 *// 考虑到编码格式 z* BufferedReader bufferedReader = **new** BufferedReader(read);  
 *// 写文件* FileOutputStream fos = **new** FileOutputStream(**new** File(htmlPosition));  
 OutputStreamWriter osw = **new** OutputStreamWriter(fos, **"UTF-8"**);  
 BufferedWriter bw = **new** BufferedWriter(osw);  
 String lineTxt = **null**;  
 **while** ((lineTxt = bufferedReader.readLine()) != **null**) {  
 bw.write(lineTxt + **"</br>"**);  
 }  
 bw.close();  
 osw.close();  
 fos.close();  
 read.close();  
 } **else** {  
 System.***out***.println(**"找不到指定的文件"**);  
 }  
 } **catch** (Exception e) {  
 System.***out***.println(**"读取文件内容出错"**);  
 e.printStackTrace();  
 }  
 }  
  
  
 */\*  
 移动图片到指定路径  
 sourceFilePath:原始路径  
 targetFilePosition:移动后存放的路径  
 \*/* **public void** changeImageUrl(String sourceFilePath,String targetFilePosition) **throws** IOException {  
 FileInputStream fis = **new** FileInputStream(sourceFilePath);  
 BufferedInputStream bufis = **new** BufferedInputStream(fis);  
  
 FileOutputStream fos = **new** FileOutputStream(targetFilePosition);  
 BufferedOutputStream bufos = **new** BufferedOutputStream(fos);  
 **int** len = 0;  
 **while** ((len = bufis.read()) != -1) {  
 bufos.write(len);  
 }  
 bufis.close();  
 bufos.close();  
 }  
  
  
  
 */\*  
 \* html文件解析成xhtml，变成标准的html文件  
 \* f\_in:源html文件路径  
 \* outfile: 输出后xhtml的文件路径  
 \*/* **private boolean** parseToXhtml(String f\_in, String outfile) {  
 **boolean** bo = **false**;  
 ByteArrayOutputStream tidyOutStream = **null**; *// 输出流* FileInputStream fis = **null**;  
 ByteArrayOutputStream bos = **null**;  
 ByteArrayInputStream stream = **null**;  
 DataOutputStream to = **null**;  
 **try** {  
 *// Reader reader;* fis = **new** FileInputStream(f\_in);  
 bos = **new** ByteArrayOutputStream();  
 **int** ch;  
 **while** ((ch = fis.read()) != -1) {  
 bos.write(ch);  
 }  
 **byte**[] bs = bos.toByteArray();  
 bos.close();  
 String hope\_gb2312 = **new** String(bs, **"utf-8"**);*// 注意，默认是GB2312，所以这里先转化成GB2312然后再转化成其他的。* **byte**[] hope\_b = hope\_gb2312.getBytes();  
 String basil = **new** String(hope\_b, **"utf-8"**);*// 将GB2312转化成 UTF-8* stream = **new** ByteArrayInputStream(basil.getBytes());  
 tidyOutStream = **new** ByteArrayOutputStream();  
 Tidy tidy = **new** Tidy();  
 tidy.setInputEncoding(**"utf-8"**);  
 tidy.setQuiet(**true**);  
 tidy.setOutputEncoding(**"UTF-8"**);  
 tidy.setShowWarnings(**true**); *// 不显示警告信息* tidy.setIndentContent(**true**);*//* tidy.setSmartIndent(**true**);  
 tidy.setIndentAttributes(**false**);  
 tidy.setWraplen(1024); *// 多长换行  
 // 输出为xhtml* tidy.setXHTML(**true**);  
 tidy.setErrout(**new** PrintWriter(System.***out***));  
 tidy.parse(stream, tidyOutStream);  
 to = **new** DataOutputStream(**new** FileOutputStream(outfile));*// 将生成的xhtml写入* tidyOutStream.writeTo(to);  
 bo = **true**;  
 } **catch** (Exception ex) {  
 System.***out***.println(ex.toString());  
 ex.printStackTrace();  
 **return** bo;  
 } **finally** {  
 **try** {  
 **if** (to != **null**) {  
 to.close();  
 }  
 **if** (stream != **null**) {  
 stream.close();  
 }  
 **if** (fis != **null**) {  
 fis.close();  
 }  
 **if** (bos != **null**) {  
 bos.close();  
 }  
 **if** (tidyOutStream != **null**) {  
 tidyOutStream.close();  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 System.*gc*();  
 }  
 **return** bo;  
 }  
  
 */\*  
 \* xhtml文件转pdf文件  
 \* inputFile:xhtml源文件路径  
 \* outputFile:输出的pdf文件路径  
 \* imagePath:图片的存放路径 例如(file:/D:/test)  
 \*/* **private boolean** convertHtmlToPdf(String inputFile, String outputFile) **throws** Exception {  
 OutputStream os = **new** FileOutputStream(outputFile);  
 ITextRenderer renderer = **new** ITextRenderer();  
 String url = **new** File(inputFile).toURI().toURL().toString();  
 renderer.setDocument(url);  
 *// 解决中文支持问题* ITextFontResolver fontResolver = renderer.getFontResolver();  
 fontResolver.addFont(**"C:/Windows/Fonts/simsun.ttc"**, BaseFont.***IDENTITY\_H***, BaseFont.***NOT\_EMBEDDED***);  
 *// 解决图片的相对路径问题  
// renderer.getSharedContext().setBaseURL("imagePath");* renderer.layout();  
 renderer.createPDF(os);  
 os.flush();  
 os.close();  
 **return true**;  
 }  
  
 */\*  
 \* xhtml转成标准html文件  
 \* targetHtml:要处理的html文件路径  
 \* 这里一定要将不规范的html转化成规范的HTML  
 \* \*/* **private static void** standardHTML(String targetHtml) **throws** IOException {  
 File f = **new** File(targetHtml);  
 org.jsoup.nodes.Document doc = Jsoup.*parse*(f, **"UTF-8"**);  
 doc.select(**"html"**).attr(**"style"**, **"font-family: SimSun;"**);  
 doc.select(**"meta"**).removeAttr(**"name"**);  
 doc.select(**"meta"**).attr(**"content"**, **"text/html; charset=UTF-8"**);  
 doc.select(**"meta"**).attr(**"http-equiv"**, **"Content-Type"**);  
 doc.select(**"meta"**).html(**"&nbsp"**);  
 doc.select(**"img"**).html(**"&nbsp"**);  
 doc.select(**"style"**).attr(**"mce\_bogus"**, **"1"**);  
 doc.select(**"body"**).attr(**"font-family"**, **"simsun"**);  
 doc.select(**"br"**).html(**"&nbsp"**);  
 doc.select(**"html"**).before(**"<?xml version='1.0' encoding='UTF-8'>"**);  
 doc.select(**"p"**).attr(**"style"**, **"font-family: SimSun;"**);  
 */\*  
 \* Jsoup只是解析，不能保存修改，所以要在这里保存修改。  
 \*/* FileOutputStream fos = **new** FileOutputStream(f, **false**);  
 OutputStreamWriter osw = **new** OutputStreamWriter(fos, **"UTF-8"**);  
 osw.write(doc.html());  
 System.***out***.println(doc.html());  
 osw.close();  
 }  
  
  
 **public static void** main(String[] args) **throws** Exception {  
 FileConverter fileConverter=**new** FileConverter();

//docToHtml方法生成的html格式不标准，会缺少闭合标签  
 fileConverter.docToHtml(**"f:/tmp/efg.doc"**, **"f:/tmp/efg.html"**);  
 fileConverter.parseToXhtml(**"f:/tmp/efg.html"**,**"f:/tmp/efg.html"**);

//转化成标准标签，将缺省的闭合标签补上  
 *standardHTML*(**"f:/tmp/efg.html"**);  
  
 fileConverter.convertHtmlToPdf(**"f:/tmp/efg.html"**,**"f:/tmp/efg.pdf"**);  
  
 fileConverter.docxToHtml(**"f:/tmp/abc.docx"**, **"f:/tmp/abc.html"**);  
 fileConverter.parseToXhtml(**"f:/tmp/abc.html"**,**"f:/tmp/abc.html"**);  
 *standardHTML*(**"f:/tmp/abc.html"**);  
 fileConverter.convertHtmlToPdf(**"f:/tmp/abc.html"**,**"f:/tmp/abc.pdf"**);  
 fileConverter.txtToHtml(**"f:/abc.txt"**,**"f:/abc.html"**);;  
 fileConverter.parseToXhtml(**"f:/abc.html"**,**"f:/abc2.html"**);  
 *standardHTML*(**"f:/abc2.html"**);  
 fileConverter.convertHtmlToPdf(**"f:/abc2.html"**,**"f:/abc2.pdf"**);  
 }  
}