**linux下word模板操作及PDF处理笔记**

最近尝试在linux下处理word文档模板，并转为PDF进行处理，网上搜罗不少资料，记录下来备忘。

**业务需求：**有一批WORD文档模板，通过系统将其中的某些信息动态替换，包括表格动态生成，然后合并成一份文档，添加页眉页脚及水印，页眉和水印为动态内容，页眉需要添加logo图片，完成后转换为PDF。

**方案：**最初想当然的考虑jacob，但考虑到系统需要部署在linux环境中，被否定，最终选择使用itext   
**方案一**，WORD文档可以通过XML格式来进行操作，考虑使用模板freemarker进行WORD文档模板操作，包括内容替换和表格动态生成等。但遇到问题：一是合并文件功能实现未找到合适的实现方式，二是生成的文档格式openoffice无法正常转为PDF。   
附上为解决合并WORD文档问题，网上搜集到的方案,通过分析RTF文件格式后以流的方式进行合并，将WORD转为RTF进行处理，成功实现了WORD文档合并问题，记录下来。

/\*\*

\* word合并

\*

\* @param toFilePath

\* @param sourceFilePath

\*/

public static void mergeWord(String toFilePath, List<String> sourceFilePath,String outFile)

{

if (StringUtil.isEmpty(toFilePath))

{

System.out.println("toFilePath is null");

return;

}

if (null == sourceFilePath || sourceFilePath.isEmpty())

{

System.out.println("sourceFilePath is null");

return;

}

OutputStream out = null;

try

{

out = new FileOutputStream(outFile);

int len = sourceFilePath.size();

int i = 0;

for (String path : sourceFilePath)

{

File f = new File(path);

InputStream in = new FileInputStream(f);

byte[] b = new byte[1024];

int tmp = 0;

int fLen = 0;

String str = "";

// 除了第一个文件，其他都处理掉头部

if (i > 0)

{

// 处理头开始

tmp = in.read(b);

if (tmp == -1)

{

str = new String(b);

}

else

{

str = new String(b, 0, tmp);

}

b = str.replaceFirst("\\{", "").getBytes();

// 处理头结束

out.write(b);

}

while ((tmp = in.read(b)) != -1)

{

fLen += tmp;

if (fLen + 1024 >= f.length())

{

// 最后一批

if (i < len - 1)

{

// 除了最后一个文件，其他文件处理尾

tmp = in.read(b);

if (tmp == -1)

{

str = new String(b);

}

else

{

str = new String(b, 0, tmp);

}

int index = str.lastIndexOf("}");

b = (str.substring(0, index) + "\\page").getBytes();

// 处理尾结束

out.write(b);

break;

}

}

out.write(b);

}

i++;

}

out.flush();

}

catch (Exception e)

{

}

finally

{

if (null != out)

{

try

{

out.close();

}

catch (IOException e)

{

out = null;

}

}

}

}

**方案二，**通过rtftemplate对rtf模板文件进行内容处理，生成rtf文件，将rtf文件转为pdf，将所有pdf合并并添加页眉页脚水印。

1. 通过rtftemplate将rtf模板文件进行内容处理，生成rtf文件

/\*\*

\*因为需要中需要处理多个模板文件，此处使用批量操作

\* 根据模板生成rtf文档

\*

\* @param dataMap

\*/

public static boolean createRTF(Map<String, Object> dataMap,

Map<String, String> rtfTmpList)

{

if (null == rtfTmpList || rtfTmpList.isEmpty())

{

logger.error("rtfTmpList is null");

return false;

}

// rtf生成器

RTFGenerator generator = new RTFGenerator();

generator.setContextMap(dataMap);

Iterator<Entry<String, String>> it = rtfTmpList.entrySet().iterator();

while (it.hasNext())

{

Entry<String, String> entry = it.next();

String src = entry.getKey();

String target = entry.getValue();

File inputFile = new File(src);

if (inputFile.exists())

{

// 找不到源文件, 则返回

File outputFile = new File(target);

if (!outputFile.getParentFile().exists())

{ // 假如目标路径不存在, 则新建该路径

outputFile.getParentFile().mkdirs();

}

try

{

generator.run(src, target);

logger.info("createRTF finish.inputFilePath=" + src

+ ",output=" + target);

}

catch (Exception e)

{

logger.error("createRTF failed.", e);

}

}

else

{

logger.info("createRTF finish.inputFile not exist.src=" + src);

}

}

return true;

}

2.使用openoffice+jodconverter将rtf文件转为pdf格式

/\*\*

\*因为需求中一次需要处理的文件比较多，此处在一次服务启动后批量处理所有文件，再关闭服务

\* String inputFilePath, String outputFilePath

\*

\* @param rtfList

\* @return

\*/

public static boolean rtf2pdf(Map<String, String> rtfList)

{

if (null == rtfList || rtfList.isEmpty())

{

return false;

}

OfficeManager officeManager = null;

try

{

// DefaultOfficeManagerConfiguration config = new

// DefaultOfficeManagerConfiguration();

// //

// String officeHome = getOfficeHome();

// config.setOfficeHome(officeHome);

// //

// officeManager = config.buildOfficeManager();

// officeManager.start();

officeManager = getConn();

if (null == officeManager)

{

logger.error("officeManager 为空");

return false;

}

OfficeDocumentConverter converter = new OfficeDocumentConverter(

officeManager);

Iterator<Entry<String, String>> it = rtfList.entrySet().iterator();

while (it.hasNext())

{

Entry<String, String> entry = it.next();

String src = entry.getKey();

String target = entry.getValue();

File inputFile = new File(src);

if (inputFile.exists())

{

// 找不到源文件, 则返回

File outputFile = new File(target);

if (!outputFile.getParentFile().exists())

{ // 假如目标路径不存在, 则新建该路径

outputFile.getParentFile().mkdirs();

}

converter.convert(inputFile, outputFile);

logger.info("rtf2pdf finish.inputFilePath=" + src

+ ",output=" + target);

}

else

{

logger.info("rtf2pdf finish.inputFile not exist.src=" + src);

}

}

}

finally

{

if (null != officeManager)

{

officeManager.stop();

logger.info("停止office转换服务。");

}

}

return true;

}

/\*\*

\*获取连接的方式，如果有已经启动的服务，直接连接已经启动的服务，如果没有启动的服务则启动服务。

\*/

public static OfficeManager getConn()

{

OfficeManager officeManager = null;

try

{

// 默认本地2002端口，linux启动默认8100

int port = 2002;

try

{

port = StringUtil.toInteger(getOfficePort());

}

catch (Exception e)

{

port = 2002;

}

logger.info("准备启动服务....");

try

{

logger.info("尝试连接已启动的服务...");

ExternalOfficeManagerConfiguration externalProcessOfficeManager = new ExternalOfficeManagerConfiguration();

externalProcessOfficeManager.setConnectOnStart(true);

externalProcessOfficeManager.setPortNumber(port);

officeManager = externalProcessOfficeManager

.buildOfficeManager();

officeManager.start();

logger.info("office转换服务启动成功!");

return officeManager;

}

catch (Exception ex)

{

ex.printStackTrace();

logger.info("没有已启动的服务...");

}

logger.info("创建并连接新服务...");

DefaultOfficeManagerConfiguration configuration = new DefaultOfficeManagerConfiguration();

configuration.setOfficeHome(getOfficeHome());

configuration.setPortNumbers(port);

configuration.setTaskExecutionTimeout(1000 \* 60L);

configuration.setTaskQueueTimeout(1000 \* 60 \* 2L);

officeManager = configuration.buildOfficeManager();

officeManager.start();

logger.info("office转换服务启动成功!");

return officeManager;

}

catch (Exception ce)

{

ce.printStackTrace();

logger.error("office转换服务启动失败!详细信息:" + ce);

return null;

}

}

 3.使用itext合并所有pdf文件，对合并后的文件添加页眉页脚及水印处理

/\*\*

\* 添加footer

\*

\* @param fileName

\* @param savepath

\* @return int -1:failed

\*/

public static int addFooterAndWater(String fileName, String savepath,

String waterMarkName, String pageHeade, String foot)

{

// 文档总页数

int num = 0;

Document document = new Document();

try

{

PdfReader reader = new PdfReader(fileName);

BaseFont base = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H",

BaseFont.EMBEDDED);

num = reader.getNumberOfPages();

PdfCopy copy = new PdfCopy(document, new FileOutputStream(savepath));

document.open();

for (int i = 0; i < num;)

{

PdfImportedPage page = copy.getImportedPage(reader, ++i);

PageStamp stamp = copy.createPageStamp(page);

Font f = new Font(base);

// 添加页脚，左侧文字，右侧页码

ColumnText.showTextAligned(stamp.getUnderContent(),

Element.ALIGN\_RIGHT,

new Phrase(String.format("Page %d of %d", i, num), f),

550f, 28, 0);

ColumnText.showTextAligned(stamp.getUnderContent(),

Element.ALIGN\_LEFT, new Phrase(foot, f), 50f, 28, 0);

// 添加页眉 (文字页眉，居中)

ColumnText.showTextAligned(stamp.getUnderContent(),

Element.ALIGN\_CENTER, new Phrase(pageHeade, f), 150f,

800, 0);

// 页眉添加logo （图片页眉，居右）

Image img = Image.getInstance("template/logo.png");// 选择图片

img.setAlignment(1);

img.scaleAbsolute(436 / 5, 96 / 5);// 控制图片大小

img.setAbsolutePosition(450f, 800);// 控制图片位置

stamp.getUnderContent().addImage(img);

// 添加水印

PdfContentByte under = stamp.getUnderContent();

under.beginText();

under.setColorFill(Color.LIGHT\_GRAY);

// 字符越长，字体越小，设置字体

int fontSize = getFontSize(waterMarkName.length());

under.setFontAndSize(base, fontSize);

// 设置水印文字字体倾斜 开始

float pageWidth = reader.getPageSize(i).getWidth();

float pageHeight = reader.getPageSize(i).getHeight();

under.showTextAligned(Element.ALIGN\_CENTER, waterMarkName,

pageWidth / 2, pageHeight / 2, 60);// 水印文字成60度角倾斜,且页面居中展示

// 字体设置结束

under.endText();

stamp.alterContents();

copy.addPage(page);

}

}

catch (Exception e)

{

logger.error("addFooter failed.msg=" + e.toString());

return -1;

}

finally

{

if (null != document)

{

document.close();

}

}

logger.info("pdf totalpages:" + num);

return num;

}

关于遇到的几个问题及解决：

1、openoffice启动时，默认端口8100,windows本地默认端口为2002

 问题：在启动openoffice服务后，找不到8100的监听端口，或者启动服务失败。

尝试解决：安装图形化界面，在图形化界面下启动openoffice服务看是否可以解决，我们是通过此方式解决。

2.如果使用centos 6，由于centos 6系统自带的没有openoffice，尝试使用liberoffice，代码实现方面和openoffice一样使用，不需要任何修改，修改liberoffice对应的根路径配置即可。

3.将rtf转pdf时，原本的内容有30页，但转pdf后内容变大（因为页码发生变化，要求页码与模板一致），观察发现字体及样式发生变化导致。

解决：因为linux下没有对应的中文字体，需要添加相应的中文字体，记录解决方案；

将本地系统下的字体文件拷贝到linux系统上，本地在c:\windows\fonts,将需要的字体文件拷贝到linux上，在/usr/share/fonts目录下新建一个自定义目录即可，我比较偷懒，直接全部拷贝过去了。   
执行以下命令刷新，重启openoffice服务，问题解决，本地转pdf后是多少页，服务器上一样的效果。   
mkfontscale   
mkfontdir//这两条命令是生成字体的索引信息   
fc-cache //更新字体缓存

4.linux下openoffice默认页眉是奇偶页不一样，如果需要统一页眉，需要通过图形界面在linux上将openoffice的页眉设置为奇偶页一样，可以新建一个文档，在”格式“-"页面“-”页眉“属性中选中"奇偶页相同"，再进行操作的时候就默认所有页面页眉一致了。

5.制作rtf模板的问题

 a、在模板中添加循环时，始终不生效。

原因：如果需要循环时（startloop+循环对象或循环对象的属性+endloop），附件中的dot模板不适合在word2010，2007生成rtf模板，需要在word2003中生成rtf模板，在03下就可以正常的生成循环。   
b、模板中中文乱码问题，

 原因1：生成的模板中，对应的变量使用的字体默认为Times New Roman，替换后的中文也会使用该字体，该字体是不识别中文的，需要将字体修改为宋体后，就可以正常显示中文，删除变量两侧的符号。   
原因2：通过rtf模板生成rtf文件时，需要将中文字体转为rtf的格式，通过将rtf使用文本编辑器打开看，发现中文都是转码过后的，将填充的中文信息转码后就可以正常显示中文。

c、已经生成的rtf模板，有时候需要进行修改，但经常发现，原本好好的rtf模板，比如再添加几个变量后，就发现总显示不对，要么就是无法显示，要么就是会多个<>。   
解决：没有找到具体原因，1、如果已经有正常的变量，直接拷贝过来可以解决.2、可以通过dot新建一个空白的文档，在空白文档中添加变量，显示会多个<>，再拷贝一个正常的变量粘贴到该文档中后，都正常了，此时再按方法1解决。

主要jar包：   
itext-rtf-2.1.7.jar   
jodconverter-core-3.0-4.jar   
iText-2.1.5.jar   
rtftemplate-1.0.1-b13.jar