题目： pdf文档转png图片的那些事

1. 项目需要将移动端客户上传的pdf转成png图片，一页一个文件，提供给微信小程序用于预览。
2. 这里怎么上传，怎么预览图片都不是“那些事”，那些事是java开源库在工作时的故事。

故事1：

项目开始时使用的是Aspose公司（这是澳大利亚的一家公司）的com.aspose.pdf这个开源库。 总的说，这个库的质量是蛮高的，但是效率比较低（鱼和熊掌不可兼容嘛）。线上超过10页的pdf往往就要超过10秒的响应，这个多数时候都是无法忍受的。在没有改变业务调用流程的情况下，就有了故事2。

故事2：

效率低就不用aspose的库， 因为之前也没有经验，在评测了icepdf，pdfbox等的效率之后，就直接上了icepdf库（以为比较下来效率较高，是aspose的两倍）。本地（windows）、测试环境都上了，没啥问题；结果一到线上，一部分中文乱码，图章（项目的重点）中的中文或消失或整章不识别。

排查一番，原来是linux服务器上没有中文字库，而icepdf是加载系统字块，aspose是动态加载包括用户路径、自定义路径和系统库的可能路径（是每次转换都加载一次）。这里也就理解了为什么aspose的效率低了。

于是乎，整了一番系统字库的配置后，终于中文不再乱码了。

你以为就这样了吗？故事3怎么来的呢？

故事3：

过来一个年，又赶上疫情（2020），终于到岗的开始项目的继续工作。呵呵，甲方的一个测试pdf文档，转换出来了“汉字重叠”。而且还只有一行文字出现重叠。查同一个文档的同字体的文字没有问题。

经多方排查，icepdf的字体引擎的问题，换了一个较旧的版本，重叠的情况解决了。

呵呵，到此为止了吗？没有，接着一个神奇的对勾符号“☑”变成了“88”，还这是要“拜拜”。

换成pdfbox库解决，然后呢？！心里面还是不踏实啊。

以下是依赖库名和实现代码。

aspose:

|  |
| --- |
| 库： aspose.pdf.Document  mvn 地址  <!-- https://mvnrepository.com/artifact/com.aspose.pdf/aspose-pdf-jdk16 -->  <dependency>  <groupId>com.aspose.pdf</groupId>  <artifactId>aspose-pdf-jdk16</artifactId>  <version>9.3.1.0</version>  </dependency>  调用示例：  去版权水印（pdf）  String license = "<License>\n" +  " <Data>\n" +  " <Products>\n" +  " <Product>Aspose.Total for Java</Product>\n" +  " <Product>Aspose.Pdf for Java</Product>\n" +  " </Products>\n" +  " <EditionType>Enterprise</EditionType>\n" +  " <SubscriptionExpiry>20991231</SubscriptionExpiry>\n" +  " <LicenseExpiry>20991231</LicenseExpiry>\n" +  " <SerialNumber>8bfe198c-7f0c-4ef8-8ff0-acc3237bf0d7</SerialNumber>\n" +  " </Data>\n" +  " <Signature>sNLLKGMUdF0r8O1kKilWAGdgfs2BvJb/2Xp8p5iuDVfZXmhppo+d0Ran1P9TKdjV4ABwAgKXxJ3jcQTqE/2IRfqwnPf8itN8aFZlV3TJPYeD3yWE7IT55Gz6EijUpC7aKeoohTb4w2fpox58wWoF3SNp6sK6jDfiAUGEHYJ9pjU=</Signature>\n" +  "</License>";  try (InputStream is =new ByteArrayInputStream(license.getBytes("UTF-8"));){  com.aspose.pdf.License asposeLic = new com.aspose.pdf.License();  asposeLic.setLicense(is); } catch (Exception e) {  throw new RuntimeException("获取权限失败"); }  FileInputStream fio = null;  try{  fio = new FileInputStream(new File(pdfFilePath));  Document pdfDocument = new Document(fio);  Resolution resolution = new Resolution(130);  JpegDevice jpegDevice = new JpegDevice(resolution);   List<byte[]> imgList = new ArrayList<>();  for (int index = 1; index <= pdfDocument.getPages().size(); index++) {  try (  OutputStream out = new ByteArrayOutputStream();  ) {  jpegDevice.process(pdfDocument.getPages().get\_Item(index), out);  byte[] ecByts = ((ByteArrayOutputStream) out).toByteArray();  imgList.add(ecByts);  } catch (Exception e) {  throw new RuntimeException(e);  }  }  return imgList;  }catch (Exception e){  e.printStackTrace();  }  return null;  返回的是每一页是图片数据的数组。 |

icepdf:

|  |
| --- |
| mvn ： 这个是旧版本（故事2）；6.2.2需要剔除media库  <dependency>  <groupId>org.icepdf</groupId>  <artifactId>icepdf-core</artifactId>  <version>4.1.1</version>  <!--<exclusions>  <exclusion>  <groupId>javax.media</groupId>  <artifactId>jai\_core</artifactId>  </exclusion>  </exclusions>--> </dependency> <dependency>  <groupId>org.icepdf.os</groupId>  <artifactId>icepdf-viewer</artifactId>  <version>6.2.2</version> </dependency>  示例代码：  Document document = new Document(); long st = System.*currentTimeMillis*(); try {  document.setFile(pdfPath);  float scale = 1.0f; //缩放比例  float rotation = 0f; //旋转角度   int page = document.getNumberOfPages();  logger.info("待转换文件共{}页", page);  List<byte[]> imgList = new ArrayList<>();   for (int i = 0; i < page; i++) {  BufferedImage image = (BufferedImage) document.getPageImage(i,  GraphicsRenderingHints.*SCREEN*, Page.*BOUNDARY\_CROPBOX*,  rotation, scale);  RenderedImage rendImage = image;  OutputStream out = null;  try {  out = new ByteArrayOutputStream();  ImageIO.*write*(rendImage, "png", out);  byte[] ecByts = ((ByteArrayOutputStream) out).toByteArray();  imgList.add(ecByts);  out.close();  } catch (Exception e) {  throw new RuntimeException(e);  } finally {  try {  if (out != null) {  out.close();  }  } catch (Exception e) {  }  }  image.flush();  }  document.dispose();  return imgList; } catch (Exception e) {  logger.error("转换出错", e); } return null; |

最后使用pdfbox：

|  |
| --- |
| pdfbox库和itext库：  <!-- https://mvnrepository.com/artifact/com.itextpdf/itextpdf --> <dependency>  <groupId>com.itextpdf</groupId>  <artifactId>itextpdf</artifactId>  <version>5.5.3</version> </dependency>  <!-- https://mvnrepository.com/artifact/com.itextpdf/itext-asian --> <dependency>  <groupId>com.itextpdf</groupId>  <artifactId>itext-asian</artifactId>  <version>5.2.0</version> </dependency>  <!-- pdfbox --> <dependency>  <groupId>org.apache.pdfbox</groupId>  <artifactId>fontbox</artifactId>  <version>2.0.18</version> </dependency> <dependency>  <groupId>org.apache.pdfbox</groupId>  <artifactId>pdfbox</artifactId>  <version>2.0.18</version> </dependency>  调用示例：  File file = new File(PdfFilePath);  PDDocument pdDocument;  try {  pdDocument = PDDocument.*load*(file);  PDFRenderer renderer = new PDFRenderer(pdDocument);  /\* dpi越大转换后越清晰，相对转换速度越慢 \*/  PdfReader reader = new PdfReader(PdfFilePath);  int pages = reader.getNumberOfPages();  List<byte[]> imgList = new ArrayList<>();   if(pages >10){  pages = 2;// 测试使用  }   for (int i = 0; i < pages; i++) {   try (  OutputStream out = new ByteArrayOutputStream();  ) { // BufferedImage image = renderer.renderImageWithDPI(i, dpi);  BufferedImage image = renderer.renderImage(i);  ImageIO.*write*(image, "png", out);  byte[] ecByts = ((ByteArrayOutputStream) out).toByteArray();  imgList.add(ecByts);  } catch (Exception e) {  throw new RuntimeException(e);  }  }  return imgList;  } catch (IOException e) {  e.printStackTrace();  }  return null; |