### PDF和Java技术(PDFBox)

2008-07-17 08:29:44|  分类： [java(B/S)](http://hyw520110.blog.163.com/blog/#m=0&t=1&c=fks_094067083081088071086082074065086080083071085094081)|字号 订阅

原文标题：[Making PDFs Portable: Integrating PDF and Java Technology](http://www.sys-con.com/story/print.cfm?storyid=48543)  
原文日期：2005年3月24日  
原文作者：Ben Litchfield

### 摘要

　　自从Adobe公司1993年第一次发布公共PDF参考以来，支持各种语言和平台的PDF工具和类库就如雨后春笋般涌现。然而，Java应用开发中Adobe技术的支持相对滞后了。  
  
  
　　自从Adobe公司1993年第一次发布公共PDF参考以来，支持各种语言和平台的PDF工具和类库就如雨后春笋般涌现。然而，Java应用开发中Adobe技术的支持相对滞后了。这是个奇怪的现象，因为PDF文档是企业信息系统存储和交换信息的大势所趋，而Java技术特别适合这种应用。然而，Java开发人员似乎直到最近才获得成熟可用的PDF支持。  
  
　　PDFBox（一个BSD许可下的源码开放项目）是一个为开发人员读取和创建PDF文档而准备的纯Java类库。它提供如下特性：

* 提取文本，包括Unicode字符。
* 和Jakarta Lucene等文本搜索引擎的整合过程十分简单。
* 加密/解密PDF文档。
* 从PDF和XFDF格式中导入或导出表单数据。
* 向已有PDF文档中追加内容。
* 将一个PDF文档切分为多个文档。
* 覆盖PDF文档。

### PDFBox API

　　PDFBox设计时采用面向对象的方式来描述PDF文档。PDF文档的数据是一系列基本对象的集合：数组，布尔型，字典，数字，字符串和二进制流。PDFBox在org.pdfbox.cos包（COS模型）中定义这些基本对象类型。你可以使用这些对象与PDF文档进行任何交互，但你应该先对PDF文档内部结构以及高层概念作一些深入的了解。例如，页面和字体都是带有特殊属性的字典对象；PDF参考手册提供这些特殊属性的含义和类型的说明，但这是一个枯燥的文档查阅过程。  
  
　　于是，org.pdfbox.pdfmodel包（PD模型）应运而生，它的基础是COS模型，但提供了以一种熟悉的方式访问PDF文档对象的高层API（如图1）。对底层COS模型进行了封装的PDPage和PDFont等类就在这个包中。  
  
  
  
　　注意，虽然PD模型提供了一些优秀的功能，但它依然是一个开发中的模型。在有些实例中，你可能需要借助于COS模型才能访问PDF的特定功能性。所有的PD模型对象都提供返回相应的COS模型对象的方法。所以，在一般情况下，你都会使用PD模型，但PD模型鞭长莫及时你可以直接操作底层的COS模型。  
  
　　上文对PDFBox作了大体上的介绍，现在是举一些例子的时候了。我们从如何读已存在的PDF文档开始：

1. PDDocument document =
2. PDDocument.load( "./test.pdf" );

　　上面的语句解析指定的PDF文件并在内存中创建其文档对象。考虑到处理大文档时的效率问题，PDFBox只在内存中存储文档结构，图像、内嵌字体和页面内容等对象将被缓存在一个临时文件中。  
  
　　注意：PDDocument对象使用完毕时需要调用其close()方法来释放创建时使用的资源。

### 文本提取和Lucene整合

　　这是一个信息展现时代（an information retrieval age），不管信息存放在哪种媒体中，应用程序都应该支持检索和索引。对信息进行组织和分类从而形成可检索的格式是很关键的。这对于文本文档和HTML文档来说是很简单的，但PDF文档包含大量的结构和元信息，提取文档内容决不是一件简单的事情。PDF语言和Postscript相似，二者中的对象都是作为矢量绘制在页面的某些位置。例如：

1. /Helv 12 Tf
2. 0 13.0847 Td
3. (Hello World) Tj

　　上面的指令将字体设为12号的Helvetica，移到下一行然后打印“Hello World”。这些命令流通常是经过压缩的，文字在屏幕上的显示顺序并不一定是文件中的字符出现顺序。因此，你有时无法直接从原始PDF文档中提取字符串。然而，PDFBox成熟的文本提取算法使得开发人员可以提取文档内容，就像在阅读器中呈现的那样。  
  
　　Lucene是Apache Jakarta项目的子项目，它是一个流行的源代码开放的搜索引擎库。开发人员可以使用Lucene来创建索引，并基于该索引对大量的文本内容进行复杂的检索。Lucene只支持文本内容的检索，所以开发人员需要将其他形式的数据转换为文本形式才能使用Lucene。例如，Microsoft Word和StarOffice文档都必须先转换为文本形式才能添加到Lucene索引中。  
  
　　PDF文件也不例外，但PDFBox提供一个特殊的整合对象，这让在Lucene索引中包含PDF文档变得非常容易。将一个基本PDF文档转换为Lucene文档只需要一条语句：

1. Document doc = LucenePDFDocument.getDocument( file );

　　这条语句解析指定的PDF文档，提取其内容并创建一个Lucene文档对象。然后你就可以将该对象添加到Lucene索引中了。如上文所述，PDF文档中也包含作者信息和关键词等元数据，在索引PDF文档时对这些元数据进行跟踪时很重要的。表1列出了创建Lucene文档时PDFBox将填写（populate）的字段。  
  
  
  
　　这种整合使得开发人员可以轻松地使用Lucene来支持PDF文档的检索和索引。当然，有些应用程序要求更成熟的文本提取方法。此时可以直接使用PDFTextStripper类，或继承该类来满足这种复杂的需求。  
  
　　通过继承PDFTextStripper并覆盖showCharacter()方法，你可以从许多方面对文本提取进行控制。例如，使用x、y位置信息进行限制以提取特定文本块。你可以有效地忽略所有的y坐标大于某个值的文本，这样文档头部内容就会被排除。  
  
　　另一个例子。常常有这种情况：从表单创建了一组PDF文档，但这些原始数据被丢失了。也就是说，这些文档都包含一些你感兴趣的文本，而且这些文本都在相似的位置上，但填充文档的表单数据丢失了。例如，你有一些信封，在相同的位置上都有名字和地址信息。这时，你就可以使用PDFTextStripper的派生类来提取期望的字段，这个类就像一种截取屏幕区域的设备。

### 加密/解密

　　PDF的一个流行特性是允许对文档内容进行加密、对访问进行控制，限制只能阅读未加密文档。PDF文档加密时采用一个主密码和一个可选的用户密码。如果设定了用户密码，那么PDF阅读器（如Acrobat）将在显示文档之前提示输入密码。而主密码则用于授权修改文档内容。  
  
　　PDF规范允许PDF文档的创建者对用户使用Acrobat阅读器查看文档时的某些操作进行限制。这些限制包括：

* 打印
* 修改内容
* 提取内容

　　PDF文档安全的讨论不在本文范畴之内，有兴趣的读者可以参考PDF规范的相关部分。PDF文档的安全模型是可插拔式的（pluggable），你可以在加密文档时使用不同的安全处理器（security handler）。对本文而言，PDFBox支持标准的安全处理器，它是大多数PDF文档所使用的。  
  
　　加密文档时必须先指定一个安全处理器，然后使用一个主密码和用户密码进行加密。在下面的代码中，文档被加密，用户不需要敲入就可以在Acrobat中打开它（没有设置用户密码），但是该文档不可被打印。

1. //load the document
2. PDDocument pdf =
3. PDDocument.load( "test.pdf" );
4. //create the encryption options
5. PDStandardEncryption encryptionOptions =
6. **new** PDStandardEncryption();
7. encryptionOptions.setCanPrint( **false** );
8. pdf.setEncryptionDictionary(
9. encryptionOptions );
10. //encrypt the document
11. pdf.encrypt( "master", **null** );
12. //save the encrypted document
13. //to the file system
14. pdf.save( "test-output.pdf");

　　更详细的示例参见PDFBox发布版中包含的加密工具类源代码：org.pdfbox.Encrypt。  
  
　　许多应用程序可以生成PDF文档，但不支持控制文档的安全选项。这时PDFBox就可以用来在发送给用户之前截获并加密PDF文档。

### 表单整合

　　当应用程序的输出是一系列表单域的值时，提供将表单保存成文件的功能是很必要的。这时PDF技术将是一个很好的选择。开发人员可以手动编写PDF指令来绘制图形、表格和文本。或者将数据存成XML形式并使用XSL-FO模版来创建PDF文档。然而，这些办法都是比较耗时，容易出错，而且灵活性也比较差。对于简单的表单而言，一个更好的办法是创建模版，然后将给定的输入数据填入该模版，从而生成文档。  
  
　　Employment Eligibility Verification是一个大多数人都熟悉的表单，它又叫做“I-9表单”，参见：<http://uscis.gov/graphics/formsfee/forms/files/i-9.pdf>  
  
　　你可以使用PDFBox发布版中的一个示例程序列出表单域名单：

1. java org.pdfbox.examples.fdf.PrintFields i-9.pdf

　　还有一个示例程序用于向指定的域中插入文本形式的数据：

1. java org.pdfbox.examples.fdf.SetField i-9.pdf NAME1 Smith

　　在Acrobat中打开这个PDF文档你就会看到"Last Name"域已被填写了。你也可以使用以下代码来完成相同的操作：

1. PDDocument pdf =
2. PDDocument.load( "i-9.pdf" );
3. PDDocumentCatalog docCatalog =
4. pdf.getDocumentCatalog();
5. PDAcroForm acroForm =
6. docCatalog.getAcroForm();
7. PDField field =
8. acroForm.getField( "NAME1" );
9. field.setValue( "Smith" );
10. pdf.save( "i-9-copy.pdf" );

　　下面的代码可用于提取刚才填写的表单域的值：

1. PDField field =
2. acroForm.getField( "NAME1" );
3. [**System**](http://www.javaresearch.org/source/jdk142/java/lang/System.java.html).out.println(
4. "First Name=" field.getValue() );

　　Acrobat支持将表单数据导入或导出到一个特定的文件格式“表单数据格式”（Forms Data Format）。这种文件有两类：FDF和XFDF。FDF文件存放表单数据的格式与PDF相同，而XFDF则以XML格式存放表单数据。PDFBox在一个类中处理FDF和XFDF：FDFDocument。下面的代码片断演示了如何从上面的I-9表单导出FDF数据：

1. PDDocument pdf =
2. PDDocument.load( "i-9.pdf" );
3. PDDocumentCatalog docCatalog =
4. pdf.getDocumentCatalog();
5. PDAcroForm acroForm =
6. docCatalog.getAcroForm();
7. FDFDocument fdf = acroForm.exportFDF();
8. fdf.save( "exportedData.fdf" );

　　PDFBox表单整合步骤：

1. 使用Acrobat或其他可视化工具创建PDF表单模版
2. 记下每个需要的（desirable）表单域的名称
3. 将模版存放在应用程序可以访问到的地方
4. 当PDF被请求时，使用PDFBox解析PDF模版
5. 填充指定的表单域
6. 将填充结果（PDF）返回给用户

### 工具

　　除了上文介绍的API之外，PDFBox还提供一系列命令行工具。表2列出了这些工具类并作简短介绍。

### 备注

　　PDF规范共有1172页之多，其实现的确是一浩大工程。同样，PDFBox发布版中说它“正在进行中”，新的功能会慢慢地添加上去。它的主要弱点是从零开始创建PDF文档。然而，有一些源码开放的Java项目可用于填补这个缺口。例如，Apache FOP项目支持从特殊的XML文档生成PDF，这个XML文档描述了要生成的PDF文档。此外，iText提供一个高层API用于创建表格和列表。  
  
　　PDFBox的下一个版本将支持新的PDF 1.5 对象流和交叉引用流。然后将提供内嵌字体和图像的支持。在PDFBox的努力下，Java应用程序中的PDF技术有望得到充分的支持。

### 参考资源

　　PDFBox: [www.pdfbox.org](http://www.pdfbox.org/)  
　　Apache FOP: <http://xml.apache.org/fop/>  
　　iText: [www.lowagie.com/iText/](http://www.lowagie.com/iText/)  
　　PDF Reference: <http://partners.adobe.com/asn/tech/pdf/specifications.jsp>  
　　Jakarta Lucene: <http://jakarta.spache.org/lucene/>

### 关于作者

　　Ben Litchfield是一位拥有LPA Systems开发和整合经验的商业系统顾问。在过去的两年中，他领导着PDFBox开发组。Ben拥有Rochester技术研究院的软件工程方面的学士学位。在过去的5年中，他一直在提供企业应用程序的解决方案。（[更多](http://www.sys-con.com/author/?id=5583)）