year 1
2010
project

Android PDF Writer

Description

Android PDF Writer (APW) is a simple Java library to generate simple PDF documents in Google's Android devices released under the BSD license.

The project page is: http://coderesearchlabs.com/androidpdfwriter

To learn more about how to use it, check the Usage section of this documentation.

There is a twin project for RIM's BlackBerry devices: the BlackBerry_PDF_Writer_(BPW) with the same features than this one.

NOTE:
I wrote this library because I couldn't find a library to write PDFs in Android and I needed one to complete a weekend project.
So, the initial APW library was coded in a weekend (May 14-16, 2010) following the PDF 1.4 Reference (I think my implementation is compatible with lower PDF versions also), so expect limited functionality: writes text, paint lines and rectangles... well, it covers pretty well the needs that my weekend project had at least, and may be it also address the needs of many other projects out there with simple reporting needs.

UPDATES:
Six months after that (Nov 15-16, 2010) I'm added two things: first an overload to setFont() with an encoding parameter due to a user requirement for writing special characters in text. Check the PDF 1.4 Reference APPENDIX D, to be sure which encoding fits better for your needs, since as you can see there are symbols that have no octal value in all of the encodings. And second, an overload to addText() with a text transformation parameter.

By April 2011, I ported it to BlackBerry. By June 2011, added multiple pages support. Implementing this, the methods setPageHeight() and setPageWidth() were removed. Instead, use the overloaded constructor of PDFWriter that accepts those parameters. By July 2011, added multiple fonts in a page and also some standard paper size constants. By August 2011, increased compatibility with some readers. By February 2012, XObject images are added (special thanks to Nuno Ferreira).

Finally, there are so many things that can be added (filters, linearization, author information, etc) and enhanced (it does not handle key/values for instance, etc etc etc), so if you add something useful (I think this library is a good base to be extended, check the class structure), please let me know since it might be helpful for others too.

THANKS FOR SUPPORT, COMMENTS, REPORTS AND SUGGESTIONS:

- Jon
  Gigaram Technologies
- Alessandro Valbonesi
  Algos Software
- Mark Heilpern
- Tomas Verhelst
  RootSoft
- Reza Assareh
  Screen Interaction
- Joseph Fernandez
  TEKSystems
- Luis Marin
- Michael Whetmore
- Howard Shaw
  GBC Systems
- Sarah Palser
  University of Cape Town
- Willem Vermeer
  Peecho Tech
- Nuno Ferreira

Enjoy,
Javier Santo Domingo
j-a-s-d@coderesearchlabs.com

Usage

APW is very simple to use, it offers an very straightfoward class to give you full control of the simple PDF document you are creating.

Look at this example:

```
private String generateHelloWorldPDF()
{
    PDFWriter mPDFWriter = new PDFWriter(PaperSize.FOLIO_WIDTH, PaperSize.
    FOLIO_HEIGHT);

    // note that to make this images snippet work
    // you have to uncompress the assets.zip file
    // included into your project assets folder
    AssetManager mngr = getAssets();
    try {
        Bitmap xoiPNG = BitmapFactory.decodeStream(mngr.open("CRL-borders.png"));
        Bitmap xoiJPG = BitmapFactory.decodeStream(mngr.open("CRL-star.jpg"));
        Bitmap xoiBMP1 = BitmapFactory.decodeStream(mngr.open("CRL-1bit.bmp"));
        Bitmap xoiBMP8 = BitmapFactory.decodeStream(mngr.open("CRL-8bits.bmp"));
        Bitmap xoiBMP24 = BitmapFactory.decodeStream(mngr.open("CRL-24bits.bmp"));
        mPDFWriter.addImage(400, 600, xoiPNG, Transformation.
    DEGREES_315_ROTATION);
        mPDFWriter.addImage(300, 500, xoiJPG);
        mPDFWriter.addImage(200, 400, 135, 75, xoiBMP24);
        mPDFWriter.addImage(150, 300, 130, 70, xoiBMP8);
        mPDFWriter.addImageKeepRatio(100, 200, 50, 25, xoiBMP8);
        mPDFWriter.addImageKeepRatio(50, 100, 30, 25, xoiBMP1, Transformation.
    DEGREES_270_ROTATION);
        mPDFWriter.addImageKeepRatio(25, 50, 30, 25, xoiBMP1);
    } catch (IOException e) {
        e.printStackTrace();
    }

    mPDFWriter.setFont(StandardFonts.SUBTYPE, StandardFonts.TIMES_ROMAN);
    mPDFWriter.addRawContent("1 0 0 rg\n");
    mPDFWriter.addText(70, 50, 12, "hello world");
    mPDFWriter.setFont(StandardFonts.SUBTYPE, StandardFonts.COURIER,
    StandardFonts.WIN_ANSI_ENCODING);
    mPDFWriter.addRawContent("0 0 0 rg\n");
    mPDFWriter.addText(30, 90, 10, "© CRL", Transformation.DEGREES_270_ROTATION);
    mPDFWriter.newPage();
    mPDFWriter.addRawContent("[] 0 d\n");
    mPDFWriter.addRawContent("1 w\n");
    mPDFWriter.addRawContent("0 0 1 RG\n");
    mPDFWriter.addRawContent("0 1 0 rg\n");
    mPDFWriter.addRectangle(40, 50, 280, 50);
    mPDFWriter.addText(85, 75, 18, "Code Research Laboratories");
    mPDFWriter.newPage();
    mPDFWriter.setFont(StandardFonts.SUBTYPE, StandardFonts.COURIER_BOLD);
    mPDFWriter.addText(150, 150, 14, "http://coderesearchlabs.com");
    mPDFWriter.addLine(150, 140, 270, 140);
    String s = mPDFWriter.asString();
    return s;
}
```

As you can see, it is strictly a writer where you have to write pages and content sequentially.

To learn more about the library, check the The Library section of this documentation.

The Library

The main class of this library is **PDFWriter**, which presents the following public members:

DOCUMENT, PAGES AND RAW CONTENT RELATED

- PDFWriter()
  this constructor assumes the default sizes of a A4 paper size (see The Paper Sizes for more information)

- PDFWriter(**int** pageWidth, **int** pageHeight)
  this constructor sizes the PDF pages as you specify (see The Paper Sizes for more information)

- **String** asString()
  this function retrieves the generated PDF as a String

- **void** newPage()
  this procedure creates a new page in the PDF

- **void** addRawContent(**String** rawContent)
  this procedure lets you write raw content to the PDF

FONTS RELATED

- **void** setFont(**String** subType, **String** baseFont)
  this procedure sets the current font in the PDF
  see The Standard Fonts for more information

- **void** setFont(**String** subType, **String** baseFont, **String** encoding)
  this procedure sets the current font in the PDF with encoding
  see The Standard Fonts for more information

TEXT RELATED

- **void** addText(**int** leftPosition, **int** topPositionFromBottom, **int** fontSize, **String** text)
  this procedure adds text to the PDF

- **void** addText(**int** leftPosition, **int** topPositionFromBottom, **int** fontSize, **String** text, **String** transformation)
  this procedure adds text to the PDF with transformation
  see The Transformation Constants for more information

SHAPES RELATED

- **void** addLine(**int** fromLeft, **int** fromBottom, **int** toLeft, **int** toBottom)
  this procedure adds a line to the PDF

- **void** addRectangle(**int** fromLeft, **int** fromBottom, **int** toLeft, **int** toBottom)
  this procedure adds a rectangle to the PDF

IMAGES RELATED

- **void** addImage(**int** fromLeft, **int** fromBottom, **Bitmap** bitmap)
  this procedure adds an **Bitmap** *to the PDF*

- **void** addImage(**int** fromLeft, **int** fromBottom, **Bitmap** bitmap, **String** transformation)
  this procedure adds an **Bitmap** *to the PDF with transformation*
  see The Transformation Constants for more information

- **void** addImage(**int** fromLeft, **int** fromBottom, int toLeft, **int** toBottom, **Bitmap** bitmap)
  this procedure adds an **Bitmap** *to the PDF*

- **void** addImage(**int** fromLeft, **int** fromBottom, int toLeft, **int** toBottom, **Bitmap** bitmap, **String** transformation)
  this procedure adds an **Bitmap** *to the PDF with transformation*
  see The Transformation Constants for more information

- **void** addImageKeepRatio(**int** fromLeft, **int** fromBottom, **int** width, **int** height, **Bitmap** bitmap)
  this procedure adds an **Bitmap** *to the PDF with the convenient proportions*

- **void** addImageKeepRatio(**int** fromLeft, **int** fromBottom, **int** width, **int** height, **Bitmap** bitmap, **String** transformation)
  this procedure adds an **Bitmap** *to the PDF with the convenient proportions and with transformation*
  see The Transformation Constants for more information

NOTE: Image handling is smart enough to add just once the same bitmap binary data to the generated PDF. So you can use the same image in multiple places in your document without extra size penalty.

The Paper Sizes

All the following default paper sizes are included in the **PaperSize** class.

- **int A0**_WIDTH
- **int A0**_HEIGHT

- **int A1**_WIDTH
- **int A1**_HEIGHT

- **int A2**_WIDTH
- **int A2**_HEIGHT

- **int A3**_WIDTH
- **int A3**_HEIGHT

- **int A4**_WIDTH
- **int A4**_HEIGHT

- **int A5**_WIDTH
- **int A5**_HEIGHT

- **int A6**_WIDTH
- **int A6**_HEIGHT

- **int A7**_WIDTH
- **int A7**_HEIGHT

- **int A8**_WIDTH
- **int A8**_HEIGHT

- **int A9**_WIDTH
- **int A9**_HEIGHT

- **int A10**_WIDTH
- **int A10**_HEIGHT

- **int B0**_WIDTH
- **int B0**_HEIGHT

- **int B1**_WIDTH
- **int B1**_HEIGHT

- **int B2**_WIDTH
- **int B2**_HEIGHT

- **int B3**_WIDTH
- **int B3**_HEIGHT

- **int B4**_WIDTH
- **int B4**_HEIGHT

- **int B5**_WIDTH
- **int B5**_HEIGHT

- **int B6**_WIDTH
- **int B6**_HEIGHT

- **int B7**_WIDTH
- **int B7**_HEIGHT

- **int B8**_WIDTH
- **int B8**_HEIGHT

- **int B9**_WIDTH
- **int B9**_HEIGHT

- **int B10**_WIDTH
- **int B10**_HEIGHT

- **int LETTER**_WIDTH
- **int LETTER**_HEIGHT

- **int TABLOID**_WIDTH
- **int TABLOID**_HEIGHT

- **int LEDGER**_WIDTH
- **int LEDGER**_HEIGHT

- **int LEGAL**_WIDTH
- **int LEGAL**_HEIGHT

- **int STATEMENT**_WIDTH
- **int STATEMENT**_HEIGHT

- **int EXECUTIVE**_WIDTH
- **int EXECUTIVE**_HEIGHT

- **int FOLIO**_WIDTH
- **int FOLIO**_HEIGHT

- **int QUARTO**_WIDTH
- **int QUARTO**_HEIGHT

These constants are created following ISO 216 (and other popular paper sizes) with a portrait orientation by default and converting sizes to pixels for a common 72 dpi resolution.

The Standard Fonts

All the font and text options are included in the **StandardFonts** class, presenting the following variety of public members.

Subtype constants:

- **String** SUBTYPE

Font constants:

- **String** TIMES_ROMAN
- **String** TIMES_BOLD
- **String** TIMES_ITALIC
- **String** TIMES_BOLDITALIC
- **String** HELVETICA
- **String** HELVETICA_BOLD
- **String** HELVETICA_OBLIQUE
- **String** HELVETICA_BOLDOBLIQUE
- **String** COURIER
- **String** COURIER_BOLD
- **String** COURIER_OBLIQUE
- **String** COURIER_BOLDOBLIQUE
- **String** SYMBOL
- **String** ZAPDINGBATS

Encoding constants:

- **String** MAC_ROMAN_ENCODING
- **String** WIN_ANSI_ENCODING

The Transformation Constants

All the predefined transformations are included in the **Transformation** class, presenting the following variety of public members.

Rotation constants:

- **String** DEGREES_0_ROTATION
- **String** DEGREES_45_ROTATION
- **String** DEGREES_90_ROTATION
- **String** DEGREES_135_ROTATION
- **String** DEGREES_180_ROTATION
- **String** DEGRESS_225_ROTATION
- **String** DEGREES_270_ROTATION
- **String** DEGREES_315_ROTATION

License

Tools

In the development of **Android PDF Writer** the following tools were specifically involved:

for source code editing
Eclipse Foundation's Eclipse
http://www.eclipse.org

for source code compiling
Google's Android SDK
http://developer.android.com/sdk

When working at **Code Research Laboratories** the following tools are used:

in the test management field
Gurock Software's TestRail
http://www.gurock.com/testrail/

in the version control field
VisualSVN Ltd's VisualSVN
http://www.visualsvn.com/

in the similarity analysing field
RedHill Consulting's Simian
http://www.redhillconsulting.com.au/products/simian/

in the application lifecycle management field
Inedo's BuildMaster
http://www.inedo.com/buildmaster/

in the documentation field
EC Software's Help & Manual
http://www.ec-software.com/products_hm_overview.html

in the Java source code edition field
JetBrains's IntelliJ IDEA
http://www.jetbrains.com/idea/

in the bug tracking field
JetBrains's YouTrack
http://www.jetbrains.com/youtrack/

History

DATE	DESCRIPTION
2010.05.14	started coding
2010.05.15	improvements
2010.05.16	more improvements
2010.05.19	initial release
2010.11.15	added setPageFont() overload with encoding parameter
2010.11.16	added addText() overload with text transformation
2011.04.20	ported to BlackBerry (creating the BlackBerry PDF Writer twin project)
2011.04.25	added addRectangle()
2011.05.19	added Documentation
2011.06.15	added multiple pages support
2011.07.01	added paper sizes constants added multiple fonts in a page
2011.08.01	adjusted the objects ordering to increase the compatibility with certain readers
2012.02.23	added XObjectsImage support added several fixes to improve more the compatibility with certain readers