



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 too), so it had limited functionality: write text, paint lines and rectangles... well, it covered pretty well the needs that my weekend project had at least, and since then it also addressed the needs of many other projects out there with simple reporting needs.

Many things have been added to the library, so please check carefully this documentation to learn what you can achieve with this library and what things are up to you (text wrapping for example).

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 (this library proved to be a good base to be extended, due to it's simple class structure), please let me know since it might be helpful for others too.

THANKS FOR SUPPORT, COMMENTS, REPORTS, SUGGESTIONS AND DONATIONS:

- | | | |
|---|---|--|
| • Jon
<i>Gigaram Technologies</i> | • Reza Assareh
<i>Screen Interaction</i> | • Howard Shaw
<i>GBC Systems</i> |
| • Alessandro Valbonesi
<i>Algos Software</i> | • Joseph Fernandez
<i>TEKSystems</i> | • Sarah Palser
<i>University of Cape Town</i> |
| • Mark Heilpern | • Luis Marin | • Willem Vermeer
<i>Peecho Tech</i> |
| • Tomas Verhelst
<i>RootSoft</i> | • Michael Whetmore | • Nuno Ferreira |
| • Sarah West | • Ed Loeffler | • Alexander Naberezhnov |
| • Matt Moskal | • Jagat Mahapatra
<i>Wipro</i> | • Ulrich Mende |
| • Juan Manzanera Gil | • Florian Bühlmann | • Vicente Giner Tendero |
| • David Yost
<i>IGC Software</i> | • Cezmi Topaloğlu | • Shankar Bhumkar |
| • Janusz Wśniewski | • Juan Lopez Tamayo
<i>PSL</i> | • Alex Kutter
<i>Anton Paar</i> |

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** setCurrentPage(**int** pageNumber)
this procedure set the current page to the number specified
- **int** getPageCount()
this function retrieves the page count
- **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

NOTE: There is no mechanism for automatic text wrapping provided, so it is up to you to determine which is the best solution for your own needs.

The Library

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.

- | | | |
|-------------------------------|-------------------------------|-------------------------------------|
| • <code>int A0_WIDTH</code> | • <code>int B0_WIDTH</code> | • <code>int LETTER_WIDTH</code> |
| • <code>int A0_HEIGHT</code> | • <code>int B0_HEIGHT</code> | • <code>int LETTER_HEIGHT</code> |
| • <code>int A1_WIDTH</code> | • <code>int B1_WIDTH</code> | • <code>int TABLOID_WIDTH</code> |
| • <code>int A1_HEIGHT</code> | • <code>int B1_HEIGHT</code> | • <code>int TABLOID_HEIGHT</code> |
| • <code>int A2_WIDTH</code> | • <code>int B2_WIDTH</code> | • <code>int LEDGER_WIDTH</code> |
| • <code>int A2_HEIGHT</code> | • <code>int B2_HEIGHT</code> | • <code>int LEDGER_HEIGHT</code> |
| • <code>int A3_WIDTH</code> | • <code>int B3_WIDTH</code> | • <code>int LEGAL_WIDTH</code> |
| • <code>int A3_HEIGHT</code> | • <code>int B3_HEIGHT</code> | • <code>int LEGAL_HEIGHT</code> |
| • <code>int A4_WIDTH</code> | • <code>int B4_WIDTH</code> | • <code>int STATEMENT_WIDTH</code> |
| • <code>int A4_HEIGHT</code> | • <code>int B4_HEIGHT</code> | • <code>int STATEMENT_HEIGHT</code> |
| • <code>int A5_WIDTH</code> | • <code>int B5_WIDTH</code> | • <code>int EXECUTIVE_WIDTH</code> |
| • <code>int A5_HEIGHT</code> | • <code>int B5_HEIGHT</code> | • <code>int EXECUTIVE_HEIGHT</code> |
| • <code>int A6_WIDTH</code> | • <code>int B6_WIDTH</code> | • <code>int FOLIO_WIDTH</code> |
| • <code>int A6_HEIGHT</code> | • <code>int B6_HEIGHT</code> | • <code>int FOLIO_HEIGHT</code> |
| • <code>int A7_WIDTH</code> | • <code>int B7_WIDTH</code> | • <code>int QUARTO_WIDTH</code> |
| • <code>int A7_HEIGHT</code> | • <code>int B7_HEIGHT</code> | • <code>int QUARTO_HEIGHT</code> |
| • <code>int A8_WIDTH</code> | • <code>int B8_WIDTH</code> | |
| • <code>int A8_HEIGHT</code> | • <code>int B8_HEIGHT</code> | |
| • <code>int A9_WIDTH</code> | • <code>int B9_WIDTH</code> | |
| • <code>int A9_HEIGHT</code> | • <code>int B9_HEIGHT</code> | |
| • <code>int A10_WIDTH</code> | • <code>int B10_WIDTH</code> | |
| • <code>int A10_HEIGHT</code> | • <code>int B10_HEIGHT</code> | |

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** DEGREES_225_ROTATION
- **String** DEGREES_270_ROTATION
- **String** DEGREES_315_ROTATION

License

Android PDF Writer

Copyright (c) 2010-2013, Javier Santo Domingo (j-a-s-d@coderesearchlabs.com).

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- * Neither the name of the Code Research Laboratories nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

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
2013.03.10	added getPageCount() and setCurrentPage()
	adjusted the xref generation to increase the compatibility with certain readers