

FOP: An Open-Source XSL Formatter and Renderer

A) What is FOP?

FOP is the world's first print formatter driven by XSL formatting objects. It is a Java 1.1 application that reads a formatting object tree and then turns it into a PDF document. The formatting object tree, can be in the form of an XML document (output by an XSLT engine like XT or Xalan) or can be passed in memory as a DOM Document or (in the case of XT) SAX events.

FOP is part of Apache's XML project. The homepage of FOP is <http://xmlgraphics.apache.org/fop> <http://xmlgraphics.apache.org/fop>

A1) MiniScribus XSL-Fo editor try its visit page <http://code.google.com/p/fop-miniscribus/>

B) Downloading FOP

The latest release version is *FOP0.14* ().

NOTE: you do not have to unzip or unjar this jar file.

Documentation can be downloaded here as *HMTL file* () or as *PDF file* () .

To run FOP from the command line, see Running FOP. If you are interested in embedding FOP in a Java application of your own, see Embedding FOP.

You can also download the *source code v. 0.14* () as jar file

C) Running FOP

1) Prerequisites

Following software must be installed:

a) Java 1.1.x or later

For the fo-file viewer mode of FOP (see below) you must have the swing classes installed. From Java 1.2 on (aka Java 2) they are part of the standard java distribution. If you use Java 1.1.x you must separately include the swing classes, which can be found at the *Sun website* (<http://java.sun.com/products/jfc/#download-swing>) .

b) An XML parser which supports SAX and DOM like *Xerces-J* (<http://xerces.apache.org/xerces-j/index.html>) .

c) If you have to produce the flow objects files, which are the input for FOP, you need a transformation utility to create this files from your xml files. Normally this is an XSLT stylesheet processor like *XT* (<http://www.jclark.com/xml/xt.html>) or *XALAN* (<http://xalan.apache.org/index.html>) .

2) Starting FOP as an standalone application

There are three ways to run FOP from the command line.

a) Batch processing formatting objects (fo) files:

```
java org.apache.fop.apps.CommandLine fo-file pdf-file
```

b) Batch processing xml files (includes production of the fo-files):

```
java org.apache.fop.apps.CommandLine xml-file xsl-file pdf-file
```

c) Previewing the fo-file:

```
java org.apache.fop.apps.AWTCommandLine fo-file
```

Each method uses next to the fop classes other packages. The following describes each method in detail.

a) Method One

One is to first use an XSLT engine to produce the formatting object tree as an XML document and then running the class `org.apache.fop.apps.CommandLine` with the formatting object file name and PDF filename as arguments. You will need to include FOP and your XML Parser in your classpath and so you might invoke

```
java -cp fop_x_xx_x.jar;xerces.jar  
org.apache.fop.apps.CommandLine fo-file pdf-file
```

If your SAX Parser is other than Xerces, you will need to set the property `org.xml.sax.parser` to the SAX Parser class to use. The following example shows the command line, if you use XP, the XML parser from James Clark:

```
java -Dorg.xml.sax.parser=com.jclark.xml.sax.Driver
-cp fop_x_xx_x.jar;sax.jar;xt.jar;xp.jar;xerces.jar
org.apache.fop.apps.AWTCommandLine formatting-tree-file pdf-file
(You have to include xerces.jar or another xml parser which supports DOM in your classpath.)
```

b) Method Two

Rather than performing transformation with an XSLT before invoking FOP, it is possible, if you use XT as your XSLT engine, to just call FOP and have it call XT for you. To do this, run the class `org.apache.fop.apps.CommandLine` with the source XML file name, XSL file name and PDF file name as arguments. You will need to include FOP, SAX, your SAX Parser and XT in your classpath and so you might invoke

```
java -Dorg.xml.sax.parser=com.jclark.xml.sax.Driver
-cp fop_x_xx_x.jar;xt.jar;xerces.jar
org.apache.fop.apps.CommandLine xml-file xsl-file pdf-file
```

Again, if your SAX Parser is other than Xerces, you will need to set the property `org.xml.sax.parser` to the SAX Parser class to use.

c) Method Three

If you already produced the FO file, you can preview the results of your transformation without using any pdf viewer by invoking FOP with the viewer application. You will need to include FOP and your XML Parser in your classpath

```
java -cp fop_x_xx_x.jar;xerces.jar
org.apache.fop.apps.AWTCommandLine fo-file
```

The viewer uses the swing classes.

Note: If you are using java 2 or later (i.e. jdk 1.2. or later) you can put all needed jar files into the subdirectory `jdk1.2.xjre\lib\ext` (windows example). Then FOP can be started without classpath:

```
java org.apache.fop.apps.CommandLine fo-file pdf-file
```

3) Running FOP on MacOS

Ensure that you have a recent MRJ, and that you have downloaded and unpacked the XP and SAX distributions. The `xp.jar` and `sax.jar` files work as is on MacOS.

Drag the FOP jarfile onto the JBindery icon. When the first dialog appears, type "org.apache.fop.apps.CommandLine" in the "Class name" field. Using UNIX syntax, type the names of the input formatting-object file and the output PDF in the "Optional parameters" field.

Click on the Classpath icon. To add the `xp.jar` and `sax.jar` files, click the "Add .zip file" button, navigate to the file in question, and click Open.

Once both are added (the FOP jarfile will already be in the list), click Run. A "stdout" window will appear and display FOP runtime messages.

4) Problems

If you have problems running FOP, please have a look at the [FOP FAQ](#) ([faq.html](#)). If you don't find a solution there, you can ask for help on the list fop-dev@xmlgraphics.apache.org. Maybe it's bug and maybe somebody is already working on it.

D) Embedding FOP

Instantiate `org.apache.fop.apps.Driver`. Once this class is instantiated, methods are called to set the `Renderer` to use, the (possibly multiple) `ElementMapping(s)` to use and the `PrintWriter` to use to output the results of the rendering (where applicable). In the case of the `Renderer` and `ElementMapping(s)`, the `Driver` may be supplied either with the object itself, or the name of the class, in which case `Driver` will instantiate the class itself. The advantage of the latter is it enables runtime determination of `Renderer` and `ElementMapping(s)`.

Once the `Driver` is set up, the `buildFOTree` method is called. Depending on whether DOM or SAX is being used, the invocation of the method is either `buildFOTree(Document)` or `buildFOTree(Parser, InputSource)` respectively.

A third possibility may be used to build the FO Tree, namely calling `getDocumentHandler()` and firing the SAX events yourself.

Once the FO Tree is built, the `format()` and `render()` methods may be called in that order.

Here is an example use of `Driver` from `CommandLine.java`:

```
Driver driver = new Driver();
driver.setRenderer("org.apache.fop.render.pdf.PDFRenderer", version);
driver.addElementMapping("org.apache.fop.fo.StandardElementMapping");
```

```
driver.addElementMapping("org.apache.fop.svg.SVGElementMapping");
driver.setWriter(new PrintWriter(new FileWriter(args[1])));
driver.buildFOTree(parser, fileInputSource(args[0]));
driver.format();
driver.render();
```

E) What's Implemented?

Also see STATUS for what is being worked on.

1) Formatting Objects

- root
- layout-master-set
- simple-page-master
- region-body
- region-before
- region-after
- page-sequence
- sequence-specification
- sequence-specifier-single
- sequence-specifier-repeating
- sequence-specifier-alternating
- flow
- static-content
- block
- list-block
- list-item
- list-item-label
- list-item-body
- page-number
- display-sequence
- inline
- display-rule
- display-graphic
- table (minimal support)
- table-column (minimal support)
- table-body (minimal support)
- table-row (minimal support)
- table-cell (minimal support)

2) Properties

- end-indent
- page-master-name
- page-master-first
- page-master-repeating
- page-master-odd
- page-master-even
- margin-top (only on pages and regions)
- margin-bottom (only on pages and regions)
- margin-left (only on pages and regions)
- margin-right (only on pages and regions)
- extent
- page-width
- page-height
- flow-name
- font-family
- font-style
- font-weight
- font-size
- line-height
- text-align
- text-align-last
- space-before.optimum
- space-after.optimum
- start-indent
- end-indent
- provisional-distance-between-starts
- provisional-label-separation
- rule-thickness
- color
- wrap-option
- white-space-treatment
- break-before
- break-after
- text-indent
- href
- column-width
- background-color
- padding-top (only in conjunction with background color)
- padding-left (only in conjunction with background color)

- padding-bottom (only in conjunction with background color)
- padding-right (only in conjunction with background color)