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://xml.apache.org/fop*

B) Downloading FOP

The latest release version is *FOP 0.14* () .

NOTE: you do not have to unjar or unzip 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 seperately 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://xml.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://xml.apache.org/xalan/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.x\jre\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@xml.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)

F) Limitations

Although FOP implements the above listed fo objects and properties, sometimes it does so only in a limited way.

list-block

The fo working draft allows describes two ways to markup lists.The list-block must have as children either: 1) pairs of fo:list-item-label and fo:list-item-body formatting objects, or 2) fo:list-item formatting objects.

At the moment FOP only implements the second way. Therefore a list has a basic structure like this:

```
<fo:list-block>
<fo:list-item>
<fo:list-item-label><fo:block></fo:block></fo:list-item-label>
<fo:list-item-body><fo:block></fo:block></fo:list-item-body>
</fo:list-item>
</fo:list-block>
```

Padding

Padding works in conjunction with indents and spaces. It is only implemented for blocks. At

the moment padding can't be used to make extra space (indents+spaces must be used), but only to control how much the background-color extends beyond the content rectangle.

Tables

There two limitations for tables: 1) FOP needs you to explicitly specify column widths 2) Cells have to contain block-level FOs. They can't contain straight character data.

A working basic example of a table looks like this:

```
<fo:table>
<fo:table-column column-width="150pt"/>
<fo:table-column column-width="150pt"/>
<fo:table-body font-size="10pt" font-family="sans-serif">
<fo:table-row>
<fo:table-cell>
<fo:block>text</fo:block>
</fo:table-cell>
<fo:table-cell>
<fo:block>text</fo:block>
</fo:table-cell>
</fo:table-row>
<fo:table-row>
<fo:table-cell>
<fo:block>text</fo:block>
</fo:table-cell>
<fo:table-cell>
<fo:block>text</fo:block>
</fo:table-cell>
</fo:table-row>
<fo:table-row>
<fo:table-cell>
<fo:block>text</fo:block>
</fo:table-cell>
<fo:table-cell>
<fo:block>text</fo:block>
</fo:table-cell>
</fo:table-row>
</fo:table-body>
</fo:table>
```

G) Bugs

see STATUS file

H) Compiling FOP

1. Prerequisites

a) Java 1.1.x or later

If you use Java 1.1.x you must also seperately include the swing classes, which can be found at the *Sun website* (http://java.sun.com/products/jfc/#download-swing) . From Java 1.2 on (aka Java 2) they are part of the standard distribution.

b) An XML parser

An XML parser which supports DOM like *Xerces-J* (http://xml.apache.org/xerces-j/index.html) .

c) XT from James Clark

Some of the Java source code in FOP is generated from XML using XSLT. XT must be used to generate this code.

XT is an XSL stylesheet processor written in java. At the moment you can't use any other processor, because the make file makes use of some proprietary features of Clark's xt which allow to write output in more then one document. You can find XT at *James Clark's website* (http://www.jclark.com/xml/xt.html) . You have to use XT version 19991105 or later.

(Under windows you shouldn't use the prepackaged xt.exe but also the generic jar file, otherwise make won't work)

XT relies on an sax parser like XP (also J. Clark), which can be downloaded at *James Clark's Website* (http://www.jclark.com/xml/xp/index.html)

d) make

Under windows it has been reported that the use of the cygnus solutions port of the GNU utilities works. You can find it at *Cygnus Solutions* (http://sourceware.cygnus.com/cygwin/)

Compiling FOP on MacOS

We strongly recommend the use of Codewarrior Java. This Readme will contain a link to more information in the near future.

I) Getting involved

1. Subscribe to fop-dev@xml.apache.org by sending an email to fop-dev-subscribe@xml.apache.org

2. Read the archives to fop-dev to get an idea of the issues being discussed.

3. Subscribe to fop-cvs@xml.apache.org by sending an email to fop-cvs-subscribe@xml.apache.org (it is important that you follow changes being made).

4. Try :-) to wrap your head around the XSL working draft.

5. Get CVS working on your system.

6. Ask, on fop-dev, any questions you have at all about the code, design, etc.

7. When you feel comfortable modifying the code, send diffs to fop-dev with your contributions.

8. Have fun!

J) FOP Relevant Specifications

- *XML Recommendation* (http://www.w3.org/TR/REC-xml)

- *XSL-FO Working Draft* (http://www.w3.org/TR/WD-xsl/)
- *XSLT Recommendation* (http://www.w3.org/TR/xslt)
- *PDF Documentation* (http://partners.adobe.com/asn/developer/acrosdk/DOCS/pdfspec.pdf)
- *Simple API for XML (SAX)* (http://www.megginson.com/SAX/)
- *Document Object Model (DOM)* (http://www.w3.org/TR/REC-DOM-Level-1)
- *Namespaces in XML Recommendation* (http://www.w3.org/TR/REC-xml-names/)
- *Java JDK 1.1 Documentation* (http://java.sun.com/products/jdk/1.1/docs/index.html)

K) Licence

Apache Software Foundation and was originally created by James Tauber <jtauber@jtauber.com>. For more information on the Apache Software Foundation, please see *http://www.apache.org/* (http://www.apache.org/) .

Content