SVG

Embedding SVG in FOP

1 SVG in FOP

1.1 Introduction

FOP uses the SVG library from Batik to handle SVG. This format can be handled as an `fo:instream-foreign-object` or in a separate file referenced with `fo:external-graphic`. Either way the SVG document will be read in and converted into a DOM in Batik. This DOM will then be used by the renderer to create the graphical image.

The AWT and Print renderers simply use batik to draw the SVG into a graphic.

In the case of the PDF renderer there is a PDFGraphics2D class that Batik uses to render the image into. This class converts the drawing instructions into PDF markup which is placed into the current PDF document.

1.2 Converting SVG to a PDF Document

It is possible to convert a standalone SVG document directly into a simple page PDF document. This is possible through the use of Batik's transcoder mechanism.

```
java org.apache.batik.apps.rasterizer.Main -m application/pdf document.svg
```

This will output the svg document as "document.pdf" containing a PDF rendering of the SVG file.

It is also possible to specify the width and/or height of the PDF document on the command line with -w and -h or if you are using the transcoder api you can use the transcoding hints.

Currently the SVG image is drawn at the SVG document size and simply scaled in PDF to the new size. So the result may not be the best possible. For example if you have any images or effects it will draw them at the original resolution of the svg document. When this is viewed in the pdf it will have an incorrect resolution for the size of the pdf.

The size of the pdf file will also remain the same regardless of what size the page is.

For more information see Batik for how transcoders work.

1.3 Important Notes

The svg is inserted into PDF by using PDF commands to draw and fill lines and curves. This means that the graphical objects created with this remain as vector graphics.

There are a number of SVG things that cannot be converted directly into PDF. Parts of the graphic such as effects, patterns and images are inserted into the PDF as a raster graphic. The resolution of this graphic may not be ideal depending on the FOP dpi (72dpi) and the scaling for that graphic. This needs to be improved.

Another important note is that text is converted and drawn as a set of shapes by batik. This means that a typical character will have about 10 curves (each curve consists of at least 20 characters). This can make the pdf files large and when the pdf is viewed the viewer does not normally draw those fine curves very well (turning on Smooth Line Art in the Acrobat preferences will fix this). If the text is inserted into the PDF using the inbuilt text commands for PDF it will use a single character.

It is possible to make sure that all text is drawn into PDF using the PDF text commands by adding the following to the user config:

```
<entry>
  <key>strokeSVGText</key>
  <value>false</value>
</entry>
```

The drawback from this is that all text will be confined to text that is possible for PDF fonts (including embedded fonts) and implemented with this workaround. The fonts available are the standard pdf fonts and any fonts that you have embedded using FOP. The font sizes will be rounded to an integer value. In future this will be improved.

Currently transparency is not supported in PDF so many svg images that contain effects or graphics with transparent areas will not be displayed correctly.

1.4 Classes

These are the relevant classes, found in the package org.apache.fop.svg :

- *PDFGraphics2D*
  used for drawing onto a Graphics2D into an existing pdf document, used internally to draw the svg.
- *PDFDocumentGraphics2D*
  used to create a pdf document and inherits from PDFGraphics2D to do the rest of the drawing. Used by the transcoder to create a standalone pdf document from an svg. Can be

used independantly the same as any Graphics2D.

- *PDFTranscoder*
  used by Batik to transcode an svg document into a standalone pdf, via PDFDocumentGraphics2D.