FOP Design: Embedding FOP in Other Applications

$Revision: 911792 $

by Keiron Liddle

Table of contents

1. Introduction

This is the design for the external interface when FOP is to be embedded inside another java application.

Common places where FOP is embedded is in a report production application of a server side application such as Cocoon.

2. Settings

2.1. User Agent

Possible meanings for a user agent:

- something that makes decisions where the specifiction indicates that the user agent should decide
- FOP as the user agent, represented by a class that handles various setup and decision values
- an class that handles context for a particular FOP conversion that can be configured/overridden when embedding

The user agent is responsible for supplying user or context specific information. The list of user agent values can be found on the User Agent page.

2.2. Logging

- logging level
- logging messages of various levels
- error handling
- Logging setup (LogKit, Log4J, JDK14Logging)

2.3. XML input

- various ways to supply FOP with the xsl:fo file, fo, xml+xsl
- sax handler

2.4. general options

- base url
- uri resolvers
- which implementation of a particular LayoutManager to use

2.5. Rendering Options

- embedding fonts
- compression in pdf
- image embedding

for the PS renderer (eventually):

- PostScript Level
- PPD to use
- binary/ascii switch

2.6. Render Results

Generate Output statistics from FOP:

- Number of pages total
- Number of pages of each page-sequence
- page-master used for each page (could be used to control the paper bin to get paper from, important for me in conjunction with PS Renderer)
- recoverable errors such as overflow

2.7. Setting Up

The Driver handles the XML input. The user agent information is through the FOUserAgent. Handle logging through the user agent. Options could also be handled through the user agent, using mime type selection for renderer options.

2.8. Others

Render to more than one renderer at once (maybe not from the command line). For example you could generate a PDF for the archive and the PS for the printer in one run. It would probably be faster than converting the PDF to PostScript afterwards. Make the fo tree reuseable. If the fonts are the same then use the same area tree to output to different renderers.

Several code pieces for resolving URLs and/or file locations are scattered all over FOP and Batik. These should be replaced with an URIResolver invocation to unify behaviour and remove redundancies.