FOP Design: Renderers

$Revision: 426576 $

by Keiron Liddle

Table of contents

1 Introduction.......2

2 Design Issues.......2

2.1 Renderers are Responsible.......2

2.2 Send Output to a Stream.......2

3 Fonts.......2

4 Render Context.......2

5 XML Handling.......3

6 Extensions.......3

7 Renderer Implementations.......3

8 Adding a Renderer.......5

9 Multiple Renderers.......5

10 Status.......6

10.1 To Do.......6

10.2 Work In Progress.......6

10.3 Completed.......6

1. Introduction

A renderer is primarily designed to convert a given area tree into the output document format. It should be able to produce pages and fill the pages with the text and graphical content. Usually the output is sent to an output stream.

Some output formats may support extra information that is not available from the area tree or depends on the destination of the document.

Each renderer is given an area tree to render to its output format. The area tree is simply a representation of the pages and the placement of text and graphical objects on those pages.

The renderer will be given each page as it is ready and an output stream to write the data out. All pages are supplied in the order they appear in the document. In order to save memory it is possble to render the pages out of order. Any page that is not ready to be rendered is setup by the renderer first so that it can reserve a space or reference for when the page is ready to be rendered.The renderer is responsible for managing the output format and associated data and flow.

2. Design Issues

2.1. Renderers are Responsible

Each renderer is totally responsible for its output format.

2.2. Send Output to a Stream

3. Fonts

Because font metrics (and therefore layout) are obtained in two different ways depending on the renderer, the renderer actually sets up the fonts being used. The font metrics are used during the layout process to determine the size of characters.

4. Render Context

The render context is used by handlers. It contains information about the current state of the renderer, such as the page, the position, and any other miscellanous objects that are required to draw into the page.

5. XML Handling

A document may contain information in the form of XML for an image or instream foreign object. This XML is handled through the user agent. A standard extension for PDF is the SVG handler.

If there is XML in the SVG namespace it is given to the handler which renders the SVG into the pdf document at the given location. This separation means that other XML handlers can easily be added.

6. Extensions

Document level extensions are handled with an extension handler. This handles the information from the AreaTree and adds renders it to the document. An example is the pdf bookmarks. This information first needs to have all references resolved. Then the extension handler is ready to put the information into the pdf document.

7. Renderer Implementations

Name	Type	Font Source	Font Embedding?	Out of Order Rendering?	Notes
PDF	Paginated	FOP	Yes	Yes	Uses the PDFDocument classes to create a PDF document. Most of the work is to insert text or create lines. SVG is handled by the XML handler that uses the PDFGraphics2D and batik to draw the svg

					into the pdf page.
PostScript	Paginated	FOP	Not implemented	?	Similar to PDF.
PCL	Paginated	FOP	?	?	Similar to PDF.
SVG	Paginated	?	?	?	Creates a single svg document that contains all the pages rendered with page sequences horizontally and pages vertically. Adds links between the pages so that it can be viewed by clicking on the page to go to the next page.
TXT	Paginated	N/A	N/A	No	Outputs to a text document.
AWT	Paginated	AWT	N/A	?	This draws the pages into an AWT graphic.
XML	Paginated	FOP	No	No	Creates an XML file that represents the AreaTree.
Print	Paginated	AWT	?	No	Prints the document using the java printing facitlities. The AWT rendering is used to draw

					the pages onto the printjob.
RTF	Structural	N/A	N/A	No	Structural format uses a different rendering mechanism.
MIF	Structural	N/A	N/A	No	Structural format uses a different rendering mechanism.

8. Adding a Renderer

You can add other renderers by implementing the Renderer interface. However, the AbstractRenderer does most of what is needed, including iterating through the tree parts, so it is probably better to extend this. This means that you only need to implement the basic functionality such as text, images, and lines. AbstractRenderer's methods can easily be overridden to handle things in a different way or do some extra processing.

The relevent AreaTree structures that will need to be rendered are:

- Page
- Viewport
- Region
- Span
- Block
- Line
- Inline

A renderer implementation does the following:

- render each individual page
- clip and align child areas to a viewport
- handle all types of inline area, text, image etc.
- draw various lines and rectangles

9. Multiple Renderers

The layout of the document depends mainly on the font being used. If two renderers have the same font metrics then it is possible to use the same Area Tree to render both. This can be handled by the AreaTree Handler.

10. Status

10.1. To Do

10.2. Work In Progress

10.3. Completed

- new renderer model
- new interface for structured documents, rtf and mif
- added handlers for xml in renderer