PDF Serializer (2.1 legacy document)

Table of contents

1 Comments.....5

Table of contents

1 PDF Serializer..... 3
2 FOP and Embedding Fonts.....3
2.1 Create the font(s) metric file(s).....3
2.1.1 Example: Create the Arial metric files.....3
2.2 Create a custom configuration file.....4
2.3 Sitemap and fo2pdf serializer..... 4

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. PDF Serializer

The PDF serializer takes XSL FO SAX events as input. By using the FOP project it creates PDF out of the SAX events.

This serializer is optional and requires the FOP package in the lib directory when building Cocoon 2. However, the distribution includes this package already.

- Name : fo2pdf
- Class: org.apache.cocoon.serialization.FOPSerializer
- Cacheable: yes

2. FOP and Embedding Fonts

Dynamically generating a PDF file (with embedded fonts) in Cocoon is basically 8 steps:

1. Create the font(s) metric file(s).
2. Create a custom configuration file for FOP (Cocoons PDF renderer), which tells it what fonts are available and where to find them.
3. Create your xml (left as an exercise for the reader ;)
4. Create your xslt (again, up to you)
5. In the sitemap, tell the fo2pdf serializer where your custom configuration file is located.
6. Add a match for your PDF (I'm sure you can do the rest...).
7. Start Cocoon.
8. Request your PDF.

Easy yeah? OK. Step-by-step...

2.1. Create the font(s) metric file(s).

All java calls have nothing else in the classpath or ext directory. Also, instructions which have wrapped should be entered as one single instruction.

The instruction to generate a font metric file is:

Windows:

```
$ cd %PATH_TO_COCOON%\lib $ java -cp
optional\fop-0.20.4.jar;core\xercesImpl-2.0.0.jar;core\xml-apis.jar \
org.apache.fop.fonts.apps.TTFReader \ %PATH_TO_FONT%
%PATH_TO_METRICS_DIR%\%FONT_NAME%.xml
```

Unix:

```
$ cd $PATH_TO_COCOON/lib $ java -cp
optional/fop-0.20.4.jar;core/xercesImpl-2.0.0.jar;core/xml-apis.jar \
org.apache.fop.fonts.apps.TTFReader \ $PATH_TO_FONT
$PATH_TO_METRICS_DIR/$FONT_NAME.xml
```

2.1.1. Example: Create the Arial metric files.

For the sake of the rest of this tutorial, I'm going to be using Windows NT, converting the Arial family of fonts and storing the metrics files in the location D:\fop-fonts.

My TTF files are located in C:\WINNT\Fonts. If you are running on Linux/Windows 9x/ME/2000/XP please alter as appropriate.

I normally use Cygwin; a Unix shell environment which runs on Windows. If I slip some Unix into here, please excuse me (although I'd welcome the feedback...).

Start a command session (as appropriate to your env), then change to Cocoon libs directory.

```
$ cd %PATH_TO_COCOON%\lib
```

Create the metrics directory (D:\fop-fonts)

```
$ mkdir d:\fop-fonts
```

Create the metrics for arial.ttf, arialb.ttf, arialbi.ttf, ariali.ttf

```
$ java -cp optional\fop-0.20.4.jar;core\xercesImpl-2.0.0.jar;core\xml-apis.jar \
org.apache.fop.fonts.apps.TTFReader \ C:\WINNT\Fonts\arial.ttf D:\fop-fonts\arial.ttf.xml $
java -cp optional\fop-0.20.4.jar;core\xercesImpl-2.0.0.jar;core\xml-apis.jar \
org.apache.fop.fonts.apps.TTFReader \ C:\WINNT\Fonts\arialb.ttf D:\fop-fonts\arialb.ttf.xml $
java -cp optional\fop-0.20.4.jar;core\xercesImpl-2.0.0.jar;core\xml-apis.jar \
org.apache.fop.fonts.apps.TTFReader \ C:\WINNT\Fonts\arialbi.ttf D:\fop-fonts\arialbi.ttf.xml
$ java -cp optional\fop-0.20.4.jar;core\xercesImpl-2.0.0.jar;core\xml-apis.jar \
org.apache.fop.fonts.apps.TTFReader \ C:\WINNT\Fonts\ariali.ttf D:\fop-fonts\ariali.ttf.xml
```

If everything went to plan, you should now have the metrics for the Arial fonts in your fop-fonts directory.

2.2. Create a custom configuration file

I normally store this with the metrics file in the fop-fonts directory (called config.xml (ensure there's not a font called config ;)) although I fully qualify all the filenames just incase I move it ;)

I also find it useful to retain the .ttf as it is also possible to add other types of fonts (if you want to read the FOP docs) and the .ttf tells me where to locate the font.

```
<configuration> <fonts> <font metrics-file="D:/fop-fonts/arial.ttf.xml" kerning="yes"
embed-file="C:/WINNT/Fonts/arial.ttf"> <font-triplet name="Arial" style="normal"
weight="normal"/> <font-triplet name="ArialMT" style="normal" weight="normal"/> </font>
<font metrics-file="D:/fop-fonts/arialb.ttf.xml" kerning="yes"
embed-file="C:/WINNT/Fonts/arialb.ttf"> <font-triplet name="Arial" style="normal"
weight="bold"/> <font-triplet name="ArialMT" style="normal" weight="bold"/> </font> <font
metrics-file="D:/fop-fonts/arialbi.ttf.xml" kerning="yes"
embed-file="C:/WINNT/Fonts/arialbi.ttf"> <font-triplet name="Arial" style="italic"
weight="bold"/> <font-triplet name="ArialMT" style="italic" weight="bold"/> </font> <font
metrics-file="D:/fop-fonts/ariali.ttf.xml" kerning="yes" embed-file="C:/WINNT/Fonts/ariali.ttf">
<font-triplet name="Arial" style="italic" weight="normal"/> <font-triplet name="ArialMT"
style="italic" weight="normal"/> </font> </fonts> </configuration>
```

There are other things you can add to this file, look at the FOP documentation for further information.

If you are wondering why each font has been added twice, it has to do with the font lookup. If the font is specified as 'Arial' and the weight is 'bold' then FOP searches for a matching <font-triplet/>, then uses the parent <font/> tag to get the actual font information. If the font is specified as 'ArialMT' (it's proper name) it will still work. Think of it as an alias capability.

2.3. Sitemap and fo2pdf serializer.

All that remains is to tell the serializer, where your config file is located. Find the line in your sitemap which looks like:

<map:serializer name="fo2pdf" src="org.apache.cocoon.serialization.FOPSerializer" mime-type="application/pdf"/>

and replace it with:

<map:serializer name="fo2pdf" src="org.apache.cocoon.serialization.FOPSerializer" mime-type="application/pdf"> <user-config>D:/fop-fonts/config.xml</user-config> </map:serializer>

You can use absolute paths like above or relative ones. The relative paths will be resolved to the sitemap's directory. Furthermore it's possible to use Cocoon protocols like cocoon:// or context://.

In an older version of Cocoon (2.0.3 and earlier) the config file location was specified by using an attribute 'src' on <user-config/>. If you still have this in your sitemap, it's recommended to change it to the above provided configuration.

And that's it. Oh, one final thing to remember: the cache isn't aware of your config file; **always** delete your cache-dir after modifying your config file.

1. Comments

add your comments