DocbookTransformation

Using Docbook and the stylesheet by Norman Walsh in Cocoon

Overview

If you want to use Docbook as the system to write important structured documents and you want to transform them to PDF or to HTML with the stylesheets already prepared and maintained by Norman Walsh on the official docbook repository , you could meet some problem. Here is what I did to solve them.

Xalan (the default xslt processor of Cocoon) has some limitation in using Norman Walsh's stylesheets inside the pipelines. Norman Walsh recently has written :

..The following engines are not currently recommended for use with the DocBook XSL stylesheets: ..Xalan..

On the other hand, using Saxon, I personally have obtained incomplete transformations (I lost all the sec1/title part in the result pdf document!). To solve this problem you have to choices:

- Use saxon 6.5.2 (the version suggested by the site) and hack the jar to make it use xerces as parser
- Use saxon 7.2 and following wich use the jaxp parser (they are not considerated the official version).

Version

At the time of this writing, this Snippet was tested against: Cocoon version 2.0.3, Xalan 4.0.5, Saxon 6.5.2, Tomcat 4.0.4.

I have also included the differences you have to apply if you want to follow the way of using Saxon 7.3.1 which in my environment works. It will remain valid as long as Xalan has problems transforming docbook files with Norman Walsh's stylesheets

Obtaining the drivers

Download the xslt processor from Saxon site .

Open the zip file with an utiliy and estract the saxon.jar file into $COCOON/WEB-INF/lib directory (or an other directory in which cocoon can read the classes).

cocoon.xconf

To make Saxon available in your sitemap you have to make pretty much the same steps explained in the XSLT snippet in which these arguments are treated deeper. Here you have the xslt-processor element declaring the default xslt processor behaviour.

```
<xslt-processor class="org.apache.cocoon.components.xslt.XSLTProcessorImpl"
logger="core.xslt-processor">
<parameter name="use-store" value="true"/>
<parameter name="incremental-processing" value="true"/>
</xslt-processor>
```

Saxon 6.5.2

If you want to add the possibility to use an other xslt processor (Saxon 6.5.2 in this example) we have to add an additional component (Add it immediately under the xslt-processor element so you remember why it's there). If you use this version you must hack the processor to make it use the xerces parser (explained after).

```
<component
role="org.apache.cocoon.components.xslt.XSLTProcessor/Saxon"
class="org.apache.cocoon.components.xslt.XSLTProcessorImpl"
logger="core.xslt-processor">
<parameter name="use-store" value="true"/>
<parameter name="incremental-processing" value="false"/>
<parameter name="transformer-factory" value="com.icl.saxon.TransformerFactoryImpl"/>
</component>
```

Hacking Saxon (only with 6.5.2)

FredericGlorieux It seems to me that the xml parser problem is encountered in resolution of entities for the localisation messages {your xsl-docbook distrib} /common/l10n.xml, include by {your xsl-docbook distrib}/common/l10n.xsl# <xsl:param name="l10n.xml" select="document('../common/l10n.xml')"/>. With the last saxon6, in core.log jetty says "java.net.MalformedURLException: no protocol: en.xml" (and same for all localisations). Fast workaround (and not so bad practice in fact, because of the size of the file), edit your own l10n.xml with the localisations you need, instead of the entities.

I have received a comment which says that normally Saxon use Xerces. I really can't tell anything about it but what I'm sure of is that on my system the only way to make the thing work is to "patch" Saxon.jar to make it use Xerces this way:
Change the content of META-INF/services/javax.xml.parsers.SAXParserFactory from com.icl.saxon.aelfred.SAXParserFactoryImpl to org.apache.xerces.jaxp.SAXParserFactoryImpl. This tells Saxon to use xerces instead of aelfred as parser.#1

Don't be worried it is not so complicated: you have to extract the file (javax.xml.parsers.SAXParserFactory) from saxon.jar edit it (is a text file) and to put it again in the same place in saxon.jar

Saxon 7.5.2

FredericGlorieux Try before an offline transformation with you prefered saxon7.jar. You will have lots of warns because of bugs in you docbook-xsl distrib, essentially, confusion between strings and numbers in comparisons, especially in the generated titlepage.templates.xsl.

If you want to use instead Saxon 7.5.2 the only thing to change is the parameter "transformer-factory" and remember that in this case you don't have to change the parser used by Saxon.

```
<component
role="org.apache.cocoon.components.xslt.XSLTProcessor/Saxon"
class="org.apache.cocoon.components.xslt.XSLTProcessorImpl"
logger="core.xslt-processor">
<parameter name="use-store" value="true"/>
<parameter name="incremental-processing" value="false"/>
<!-- the path to the transformer-factory has changed ... -->
<parameter name="transformer-factory" value="net.sf.saxon.TransformerFactoryImpl"/>
</component>
```

sitemap.xmap

components

In the sitemap.xmap we have to add the new transformer which is identified by the *xsl-processor-role*

```
<map:components>

<!-- other map:components here -->

<map:transformers>

<!-- other map:transformers here -->
<!-- Xalan transformer (no xslt-processor-role) -->
<map:transformer name="xslt-xalan" pool-grow="2" pool-max="32" pool-min="8"
src="org.apache.cocoon.transformation.TraxTransformer">
<use-request-parameters>false</use-request-parameters>
<use-browser-capabilities-db>false</use-browser-capabilities-db>
</map:transformer>

<!-- Saxon transformer look at the new element: xslt-processor-role -->
<map:transformer name="xslt-saxon" pool-grow="2" pool-max="32" pool-min="8"
src="org.apache.cocoon.transformation.TraxTransformer">
<use-request-parameters>false</use-request-parameters>
<use-browser-capabilities-db>false</use-browser-capabilities-db>
<xslt-processor-role>Saxon</xslt-processor-role>
</map:transformer>

<!-- other map:transformers here -->

</map:transformers>

<!-- other map:components here -->

</map:components>
```

pipelines

Now in the pipeline of your sitemap.xmap you can decide whether to use Saxon or Xalan for xslt processing

```
<map:pipeline>

<!-- using saxon -->
<map:match pattern="*.pdf">
<map:generate src="docs/{1}.xml"/>
<map:transform type="xslt-saxon" src="stylesheets/page2fo.xsl"/>
<!-- If you want to use xalan you have to set type="xslt-xalan" -->
<map:serialize type="fo2pdf"/>
</map:match>

</map:pipeline>
```

Comments

Care to comment on this Snippet? Got another tip? Help keep this Snippet relevant by passing along any useful feedback to the cocoon-users@xml.apache.org or directly down here.

Rate the article

Interesting?

x: no
xx: mmh.. something
xxx: yes
xxxx: What I was waiting for

rate	judge	
xxxx	Gabridome 😉	

Readers comments

Since version 7.2 of Saxon you no longer need to "hack" the JAR since it uses the JAXP API by default. – AnonymousDonor

Might be worthwhile to have links to a concoon.xconf and sitemap.xmap file which implement the dual xlan/saxon usage described above. I followed the instructions and Cocoon no longer works, it can't find the components xslt, xslt-xalan or xslt-saxon.

Also, in the cocoon.xconf section above there is mention of an "XSLT snippit" it would be worth while to include a link to it. – PaulGitings

Example(s) of the serving DocBook Content on Cocoon:
1.http://www.xml-dev.com:8080/cocoon/mount/docbook/Apache-WebDAV-LDAP-HOWTO.html
1.http://www.xml-dev.com:8080/cocoon/mount/docbook/openjade.html

More hints:

- It is necessary to enable FOP extensions to render Docbook to PDF, use a custom stylesheet
- PNG format is not supported out of the box. Download and install JAI for PNG support
- references to image files must be absolute (see Bug 15316)

The following snippets might be helpful:
custom stylesheet (custom.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
                exclude-result-prefixes="#default">
  <!-- link to docbook.xsl-->
  <xsl:import href="docbook/fo/docbook.xsl"/>
  <!--~~~~~~~~~~~~~~~-->
  <!-- enable fop -->
  <!--~~~~~~~~~~~~~~~-->
  <xsl:param name="use.extensions">1</xsl:param>
  <xsl:param name="fop.extensions">1</xsl:param>
  <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
  <!-- enable admonition graphics -->
  <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
  <xsl:param name="admon.graphics">1</xsl:param>
  <xsl:param name="admon.graphics.extension" select="'.png'"/>
  <!-- CAUTION: this path must be absolute -->
  <!-- SEE: http://nagoya.apache.org/bugzilla/show_bug.cgi?id=15316 -->
  <!-- and don't forget the trailing slash -->
  <xsl:param name="admon.graphics.path" select="'/jetty/webapps/cocoon/docbook/images/'"/>
</xsl:stylesheet>
```

a very simple docbook article (example.xml)

```
<article>
        <title>Render Docbook with Cocoon</title>
        <para>
                <caution>
                        <para>Use an absolute path for admonition graphics in your custom.xsl
                        and don&amp;#39;t forget to install JAI to render PNG images.</para>
                        <figure>
                                <title>image example</title>
                                <!-- the fileref must be an absolute path -->
                                <graphic fileref="/home/rockhead/images/example.gif"/>
                        </figure>
                </caution>
        </para>
</article>
```

a sitemap snippet

```
<map:match pattern="example.pdf">
                <map:generate src="example.xml"/>
                <map:transform src="custom.xsl" type="xalan"/>
                <map:serialize type="fo2pdf"/>
</map:match>
```

Have a look at the generated PDF file

Attachment: article.001.xml

Attachment: example.pdf