XfolioOpenOfficeGeneration

WHAT

http://cvs.berlios.de/cgi-bin/viewcvs.cgi/xfolio/webapp/WEB-INF/classes/org/apache/cocoon/generation/SXWGenerator.java

Here is a regular Cocoon Generator for SXW OpenOffice Writer documents. OO becomes more and more a real world production text tool. The original format is quite xml, except it is zipped, and could embed other resources (images and other media). This generator provides text informations, metadatas, and enough styles to resolve some tagging from XSL.

STATUS

Tested on thousands of OO docs in production context, proposed as a Cocoon commit if someone wants it. All licence mentions could be let to Apache, except author name and company.

WHO

[FG] FredericGlorieux http://www.ajlsm.com

CHANGES

- 2004-10-17:[FG] Implemented as generator

- 2004-06-30:[FG] Creation as a accheable XSP

HOW

Verify your XML catalog for resolution of entities. Default Cocoon Catalog here
`src\webapp\WEB-INF\entities\catalog`
have the declaration
`PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN" "open-office/dummy.dtd"`
where dummy.dtd is an empty file. This is enough to parse XML, but the file is not validate.

```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
  <map:components>
    <map:generators>
      <map:generator name="sxw" label="content" logger="sitemap.generator.sxw" src="org.apache.cocoon.
generation.SXWGenerator"/>
    </map:generators>
  </map:components>
  <map:pipelines>
    <map:pipeline>
      <map:match pattern="**.html">
        <map:select type="exists">
          <!-- OpenOfficeWriter -->
          <map:when test="{context-attr:xfolio.efolder}{1}.sxw">
            <map:generate type="sxw" src="{context-attr:xfolio.efolder}{1}.sxw"/>
            <map:transform src="transform/oo/oo2html.xsl"/>
            <map:serialize/>
          </map:when>
        </map:select>
      </map:match>
    </map:pipeline>
  </map:pipelines>
</map:sitemap>
```

WHY

OpenOffice Writer documents could be used like other XML documents as sources. To provide an HTML view or some other transformation, but also to extract metadata on vast amount of documents. So, performances is an issue, a generator should be cacheable.

This was first implemented as an XSP, easy for testing, and for people to integrate in their cocoon apps, it's now a Cocoon generator, ready to be commited in cocoon.

Fast performances test (thanks to profiler)

- direct pipe cost ~10 ms
- xinclude cost ~160ms
- cinclude ~320ms
- SXWGenerator cost ~200ms on first call, but is *cached* (~40 ms on second call)

Forget

- default src="jar:myzip!myentry.xml" seems to load the file entry in memory, but never remember the changes (isnt'it the goal to load a class from a jar one type ?)

Forrest

For now, the best solution is in Forrest, using the new zip protocol from cocoon, able to resolve things like src="zip://content.xml@{folder}/test.sxw". This is a part of the trick. The problem of their solution is to use a cinclude transformer, which is not cacheable. You need to regenerate your aggregation each time it is requested. This is not a big problem in Forrest context (essentially generate a site), but on a real life server...

Their solution looks like that

```
<map:generate src="transform/cocoon/dummy.xml"/>
<map:transform src="transform/cocoon/sxw2oo.xsl">
  <map:parameter name="src" value="{context-attr:xfolio.efolder}{1}.sxw"/>
</map:transform>
<map:transform type="cinclude"/>
<map:serialize type="xml"/>
```

The dummy.xml is only there because you need something to begin with pure cocoon. The job is done by the XSL, to write something like that.

```
<office:document xmlns:**>
  <c:include select="/*/*">
    <xsl:attribute name="src">zip://meta.xml@<xsl:value-of select="$src"/></xsl:attribute>
  </c:include>
  <c:include select="/*/*">
    <xsl:attribute name="src">zip://content.xml@<xsl:value-of select="$src"/></xsl:attribute>
  </c:include>
</office:document>
```

Problems are

- an Xpath include is expensive (all document need to be loaded as DOM)
- Cinclude (like Xinclude) haven't seem to be cacheable for me

You may say, try a `<map:agreggate/>`... I tried, and produce so much problems of validation that I stopped.

REFERENCES

- Forrest Open Office
- XSPCaching
- cacheable.xsp
- XIncludeTransformer.java
- ZipSourceFactory.java
- ZipSourceFactory.java
- JarProtocolExample
- OpenOfficeGeneration

SEE ALSO

cited by	about xfolio	about Open Office
<<FullSearch>>	<<PageList(xfolio)>>	<<PageList(office)>>