JarProtocolExample

Hi tried this a few months ago and found that the "jar:/" protocol don't check if the source have changed. I can't say if it's problem of cocoon cache or "jar:/" specification, but what I know is I can't use it to display i, real time an oo doc that people may edit. Some more ideas here [XfolioOpenOfficeGeneration]

Here is a sitemap snippet that shows how the jar:/ protocol can be used to read data from jar files or zip files. – Con

```
<map:pipelines>
        <!-- read content from out of a zip file -->
        <map:pipeline>
                <map:match pattern="*/**.xml">
                        <map:generate src="jar:http://localhost/jar-test/{1}!/{2}.xml"/>
                        <map:serialize type="xml"/>
                </map:match>
        </map:pipeline>
        <map:pipeline>
                <map:match pattern="*/**.jpg">
                        <map:read src="jar:http://localhost/jar-test/{1}!/{2}.jpg"/>
                </map:match>
        </map:pipeline>

<!--
if you don't want to "hard-code" localhost,
don't forget the {servletPath} sitemap variable

<map:read src="jar:http://{servletPath}/{1}.sxw!/meta.xml"/>

-->

        <!-- read the archive file -->
        <map:pipeline>
                <map:match pattern="*.zip">
                        <map:read src="{1}.zip"/>
                </map:match>
                <map:match pattern="*.jar">
                        <map:read src="{1}.jar"/>
                </map:match>
        </map:pipeline>
</map:pipelines>
```

Note:

This sitemap is intended to be mounted as /jar-test - it uses http to access the zip file. This is because the jar: protocol is not aware of the cocoon: protocol. An alternative would be to access the jar file with the file: protocol, which would involve specifying a complete file path name. You could do this using an input module to specify the file name.

To test it, access it as:
`http://localhost/jar-test/test.zip/test.jpg`

Using the context protocol

Here is an example using a file relative to the context. In this case we get a file named `content.xml` inside an `open-office zip` file with a `.sxw` extension. The `realpath` module allows us to convert relative path to absolute ones.

```
<map:match src='**.sxw'>
        <map:generate src="jar:file:///{realpath:docu}/{0}!/content.xml"/>
        <map:transform src='open-office-to-docbook.xml'/>
        <map:serialize type='xml'/>
</map:match>
```