CocoonProtocolExample

Here is a sitemap snippet that shows how the cocoon:/ protocol can be used to modularize pipelines and aggregate data from multiple sources. – Bertrand Delacretaz

```
<!-- assuming the required components are correctly configured -->

<!-- for the body use an XMLized HTML file -->
<map:match pattern="body-*.xml">
  <map:generate type="html" src="body/{1}.html"/>
  <map:serialize type="xml"/>
</map:match>

<!-- for the heading use a database query -->
<map:match pattern="head-*.xml">
  <map:generate src="sql-query.xml"/>
  <map:transform type="sql">
    <map:parameter name="use-connection" value="my_pool"/>
  </map:transform>
  <map:transform src="format-query-results.xsl"/>
  <map:serialize type="xml"/>
</map:match>

<!-- combine head and body in XML -->
<map:match pattern="fulldoc-*.xml">
  <map:aggregate element="fulldoc">
    <map:part src="cocoon:/head-{1}.xml"/>
    <map:part src="cocoon:/body-{1}.xml"/>
  </map:aggregate>
  <map:serialize type="xml"/>
</map:match>

<!-- here's the HTML version -->
<map:match pattern="fulldoc-*.html">
  <map:generate src="cocoon:/fulldoc-{1}.xml"/>
  <map:transform src="xml-to-html.xsl"/>
  <map:serialize type="html"/>
</map:match>

<!-- and the RTF version -->
<map:match pattern="fulldoc-*.rtf">
  <map:generate src="cocoon:/fulldoc-{1}.xml"/>
  <map:transform src="xml-to-fo.xsl"/>
  <map:serialize type="fo2rtf"/>
</map:match>
```

Note that the various parts can be tested separately, by requesting them directly like "body-xyz.xml". This is very useful when debugging or testing complex pipelines (although it can also be done with views).

How does this work? – Gabridome

When you ask for fulldoc-someContent.rtf the request will match with the last matcher.

This pipeline (the last) will generate sax events by calling the pipeline fulldoc-someContent.xml with the cocoon protocol.

This internal request will match with the third matcher (fulldoc-*.xml). Inside this pipeline the content of other two pipelines - head-someContent.xml and body-someContent.xml - will be Aggregator. Body-someContent.xml will read body/someContent.html and it will generate and return xhtml. Head-someContent.xml will extract its contents from the connection "mypool" and it will return xml after a transformation.

While the body part depends on the page requested, the head is always extracted with the same query (sql-query.xml).

The fulldoc-*.xml pipeline, after having aggregated the results of the head and of the body pipeline, return the content to the main pipeline from where we started which transform the content to rtf.

The pipeline flow can be represented as follow:

```
fulldoc-*.rtf  --> fulldoc-*.xml --> head-*.xml -->  fulldoc-*.xml --> fulldoc-*.rtf
                                     body-*.xml --^
```

cocoon:/?

If your cocoon structure is divided into sub sitemaps you will be interested in the fact that you can call pipeline in others sitemaps by calling the right path:

- cocoon:/aPipeline ask the results of the pipeline "aPipeline" which resides in the current sitemap
- cocoon://aPipeline ask the results of the pipeline "aPipeline" which resides in the **main** (i.e. the Cocoon top-level) sitemap (or, of course, a subsitemap that it mounts)

See also

- Aggregator