SitemapPatterns

While working with Cocoon, I have come upon some useful patterns for use in the sitemap. Feel free to change/tweak these as needed. – TonyCollen

Match/Generate/Transform/Serialize

- Most basic usage of sitemap components
- Usually generating from static XML files
- Good candidate for pre-generation using CLI

```
<map:match pattern="*.html">
    <map:generate src="documents/{1}.xml"/>
    <map:transform src="stylesheets/docbook2html.xsl"/>
    <map:serialize type="xhtml"/>
</map:match>
```

Recursive Match/Generate/Transform/Serialize

- Useful for automated M/G/T/S
- Allows people to create subdirectories and Cocoon will automatically serve them
- Will match foo/blah.html, foo/bar/blah.html, foo/bar/baz/blah.html, etc.
- Centralized XSLT for unified document structure

```
<map:match pattern="**/*.html">
    <map:generate src="documents/{1}/{2}.xml"/>
    <map:transform src="stylesheets/docbook2site.xsl"/>
    <map:serialize type="html"/>
</map:match>
```

Redirect "empty" requests to a known URI

- Similar to `DirectoryIndex` directive in httpd.conf

```
<map:match pattern="">
    <map:redirect-to uri="welcome"/>
</map:match>
```

Sub-Sitemap Automounting

- Used in default `[Sitemap]`
- Minimizes maintenance of parent sitemaps
- Extremely useful

```
<map:match pattern="*/**">
    <map:mount check-reload="yes" src="{1}/" uri-prefix="{1}"/>
</map:match>
```

Dynamic Mounting – AndreasHartmann

- dynamic selection of the sitemap to process a URL
- allows specific handling of a certain URL by subsites or site sections

```
<map:match pattern="*/**-info.html">
  <map:act type="resource-exists" src="{1}/info.xmap">
    <map:mount src="{1}/info.xmap" uri-prefix="{1}"/>
  </map:act>
</map:match>

[...] (further processing of this URL)
```

Fallback Transformation – AndreasHartmann

- typical examply of resource-exists selector usage
- allows dynamic selection of a general or specific stylesheet
- easily reusable when implemented as a resource
- allows "inheritance" of stylesheet sets - typically the specific stylesheet imports the general one
- also useful for XML source documents

```
<map:match pattern="**/*.html"
  <map:generate src="content/{1}/{2}.xml"/>
  <map:select type="resource-exists">
    <map:when test="xslt/{1}/document2page.xsl">
      <map:transform src="xslt/{1}/document2page.xsl"/>
    </map:when>
    <map:otherwise>
      <map:transform src="xslt/document2page.xml"/>
    </map:otherwise>
  </map:select>
  <map:serialize/>
</map:match>
```

DirectoryGenerator and CIncludeTransformer

- aggregate a set of files from a certain directory
- see ContentAggregationExample for a more detailed example

```
<!-- overview page -->
<!-- pattern="{area}/{doctype}/{document-id}.html" -->
<map:match pattern="*/overview.html">
  <map:generate type="directory" src="content/{1}">
    <map:parameter name="sort" value="date"/>
  </map:generate>
  <map:transform src="xslt/directory2include.xsl"/>
  <map:transform type="cinclude"/>
  <map:transform src="xslt/overview/include2html.xsl"/>
  <map:serialize/>
</map:match>
```

Other Pattern Pages:

- ContentAggregationExample - generating an index page with detailed metadata.
- MetaStylesheets - applying dynamically generated stylesheets

---

Readers' comments

Wonderful. Thank you. I needed this! — Gabridome

Thanks a lot, though I'm missing pattern for usefull errorhandling! — Gernot

- Really simple, when processing something you can most often catch unknown errors with a generic exceptionhandler, those errors can at least give different messages based on a match in the errorhandler.

```
<map:selector name="exception" src="org.apache.cocoon.selection.XPathExceptionSelector">
       <exception class="java.lang.Throwable" unroll="true"/>
<map:selector>
```

- When your content is generated by xsl, the dummy xml file can be used for error messages.

```
<map:match pattern="**.xml">
        <map:generate src="common/error.xml"/>
        <map:select type="resource-exists">
                <map:when test="{1}.jpg">
                <map:transform src="style/picture.xsl">
                        <map:parameter name="name" value="{1}"/>
                </map:transform>
                </map:when>
        </map:select>
        <map:serialize/>
</map:match>
```

Errorhandling documentation --- *Patrik C*