SitemapVariableSubstitution

Within the Sitemap there are several mechanisms to substitute variables by values. You define variables by inclosing them into curly brackets.

The most easiest example is a simple Matcher

```
...
<map:match pattern="images/*.gif">
   <map:read src="resources/images/{1}.gif" mime-type="images/gif"/>
</map:match>
...
```

Where the **1** is replaced by what is matched for the first wildcard "*" and so on.

Now variables are bound their level. So if you insert a new level e.g. by calling an action or a matcher. You have to refer the previous level to get the same value from the Matcher. This can be done by going backwards just like in a directory by adding a "../" prefix per level.

```
...
<map:match pattern="images/*.gif">
   <map:act type="resource-exists">
     <map:parameter name="url" value="resources/images/{1}.gif">
     <!-- new level -->
     <map:read src="resources/images/{../1}.gif" mime-type="images/gif"/>
   </map:act>
</map:match>
...
```

If your pipelines are very deeply nested (or you just don't want to adjust the variables when you e.g. insert an action that surrounds everything) you could either directly refer to the first/root level by putting a leading slash in front of your variable

```
...
<map:match pattern="images/*.gif">
   <map:act type="resource-exists">
     <map:parameter name="url" value="resources/images/{1}.gif">
     <!-- new level -->
     <map:read src="resources/images/{/1}.gif" mime-type="images/gif"/>
   </map:act>
</map:match>
...
```

or you use the new anchor syntax. With sitemap variable anchors you name a specific level and refer to them directly by the specified name.

```
...
<map:match pattern="images/*.gif" name="root">
   <map:act type="resource-exists">
     <map:parameter name="url" value="resources/images/{1}.gif">
     <!-- new level -->
     <map:read src="resources/images/{#root:1}.gif" mime-type="images/gif"/>
   </map:act>
</map:match>
...
```

see also InputModules

Escaping

If you want to include a brace in the value, escape it with a backslash.

Example:

```
<map:transform src="...">
  <map:param name="myvar" value="\{it includes braces\}"/>
</map:transform>
```

At least in Cocoon 2.1.2, the backslashes prevent interpretation as a variable reference, but they aren't removed.