ResourceNaming

Cocoon allows you to assemble portions of reusable pipelines into Resources. In their turn those start acting as if they are Generators, Transformers, Serializers or even full blown Pipelines by themselves.

(pretty much like 'grouping' in a drawing application: multiple drawing-objects suddenly become one entity you can modify, bring to front, copy all over the place)

The Resources concept in fact allows you to somewhat define your own more specialized Sitemap-components without the need to start coding Java. However: your results will be limited to smart combinations of existing components.

You should mainly use them to achieve CleanerSiteMapsThroughResources.

The biggest drawback on the usage of Resources in your pipelines, is that you tend to loose vision of what role these Pipeline-parts are taking up. The fact is that you can easily end up trying to e.g. put two Generators into one pipeline 'cause one of them could be hidden inside a Resource. Leading you straight to a lamentable run-time exception.

As a mild way to prevent such horrors to come upon you, you can make sure that the name you give to your Resources help you see those differences.

Resource Role	Suggested prefix in the name attribute	Meaning
Generator	generate-	First in the pipe: Expects subsequent components after this resource
Tranformer	tranform-	Middle of the pipe: Expects components in front and after this resource
Serializer	serialize-	End of the Pipe: Expects components before this resource
Pipeline	pipe-	Does not stand any other components before or after

Example

A resource behaving as a Generator

```
    <map:resource name="generate-data-xml" >
        <map:generate type="weatherdata" src="{input-src}" />
    </map:resource>
```

A resource behaving as a Transformer

```
    <map:resource name="transform-data2svg" >
        <map:transform src="xsl/datafilter.xsl" />
        <map:transform src="xsl/data2svg.xsl" />
    </map:resource>
```

Three resources behaving as a full Pipelines

```
    <map:resource name="pipe-data-txt">
        <map:read mime-type="text/plain" src="{input-src}" />
    </map:resource>

    <map:resource name="pipe-data-xml">
        <map:call resource="generate-data-xml" >
                <map:parameter name="input-src" value="{input-src}" />
        </map:call>
        <map:serialize type="xml" />
    </map:resource>

    <map:resource name="pipe-data-svg">
            <map:call resource="generate-data-xml" >
                    <map:parameter name="input-src" value="{input-src}" />
            </map:call>
            <map:call resource="transform-data2svg" />
        <map:serialize type="svgxml"/>
    </map:resource>
  </map:resources>
```

Updates

- 2003_05_28 (MarcPortier): See here for a relevant remark on the cocoon-dev list on this topic *(on the side of the real subject of the thread, near the end of the message)*