Resources

Resources are reusable pipelines. They are a means to define a pipeline, using any of the normal components and obeying the usual constraints (e.g. generator, transformer, serializer or reader) and then have this reusable processing definition invoked from within other pipelines.

Resource pipelines are declared in the `map:resources` section of the sitemap and do not respond to requests directly. They are invoked by using the `map:call` element, which identifies the particular resource by its name; all resources must have a unique name, specified by its `name` attribute.

Parameters can be passed to the resource pipeline by including one or more `map:parameter` elements as children of the `map:call` element. The parameters can be referenced, by name, within the resource pipeline as normal using the curly-brace syntax for sitemap parameters.

Resources, therefore, are a way to factor out common pipelines from within a complex sitemap, so they are more easily maintained. The ability to parameterise their processing means that they can still be very flexible.

However resources are not like a macro, which might describe a pipeline fragment. The flow of processing does not return from the point at which the resource is called*. Therefore a resource must be a complete pipeline, or must invoke *another resource which is a complete pipeline*. To put this differently, a resource is called and this call must eventually end up at a Serializer or Reader. The call may progress through several other resources which might, for example, contain Selector or Actions that further affect the flow of processing but must end up at a pipeline terminating component.

*[*Note: "Since 2.1 the sitemap interpreter strategy allows for the Resources to be templating **any** composed portion of a pipeline. (Prior to 2.1 Resources were 'ending' pipelines and as such a call of resource was said 'not to return')" - Cocoon 2.1 docs]*

Example

A simple Cocoon application may involve simply taking a static XML document, transforming it with a given stylesheet, and then serializing the results as HTML. This common pipeline structure can be defined as a resource:

```
<map:resources>
  <map:resource name="simple-transform">
     <map:generate src="{src}"/>
     <map:transform src="{stylesheet}"/>
     <map:serialize type="html"/>
  </map:resource>
</map:resources>
```

The "simple-transform" resource can then be invoked as follows:

```
<map:match pattern="foo/a.html">
  <map:call resource="simple-transform">
    <map:parameter name="src" value="data/a.xml"/>
    <map:parameter name="stylesheet" value="xslt/a2html.xsl"/>
  </map:call>
</map:match>

<map:match pattern="bar/baz-rss.html">
  <map:call resource="simple-transform">
    <map:parameter name="src" value="http://www.baz.com/baz.rss"/>
    <map:parameter name="stylesheet" value="xslt/rss2html.xsl"/>
  </map:call>
</map:match>
```

Resources seem like a good way to factor out common Cocoon pipeline patterns into reusable definitions.

See also: CleanerSiteMapsThroughResources