Cocoon sitemap explained

This document is intended to be a concise explanation of the Apache Cocoon Sitemap and its use in Apache Forrest. This is a worked example showing the automatically generated Table of Contents. Please follow the various sitemaps as we explain.

```
cd $FORREST_HOME/site-author
forrest run
```

In a separate browser window, open `localhost:8888/linkmap.html` to see the generated Table of Contents. This has been transformed from the site.xml navigation configuration to show the layout of the whole site as a ToC.

Cocoon consults the sitemaps to find out how to process the `linkmap.html` request.

The main sitemap is `$FORREST_HOME/main/webapp/sitemap.xmap` and if the match is not found there then other sitemaps are consulted. The first match wins. Various sitemaps are responsible for different types of processing and there are also sitemaps in the many plugins.

So let us see how `linkmap.html` is handled.

Open `$FORREST_HOME/main/webapp/sitemap.xmap` in another window. Search for "linkmap" to find the following snippet:

```
<map:match pattern="linkmap.*">
  <map:mount uri-prefix="" src="linkmap.xmap" check-reload="yes" />
</map:match>
```

Cocoon has passed through the other potential matches earlier in the sitemap and now does further handling via the `linkmap.xmap` sitemap.

Before going any further, it is necessary to understand the "**" and "*" pattern matching and replacements. See the email thread: "Re: explain sitemap matches and pass parameters to transformers" FOR-874.

Okay we will skip some explanation of processing. At this stage we are only concerned with generating the internal xml. Later steps of processing will transform that into the final html output and adorn it with navigation menus and headers, etc. This is your main aim for most of your sitemap work for input formats: handle the incoming requests, and transform into the standard internal xml format. Then Forrest automatically does the rest.

In another browser window, open `localhost:8888/linkmap.xml` to see the internal xml format.

Open `$FORREST_HOME/main/webapp/linkmap.xmap` sitemap. Move to the "map:pipeline" section.

A digression: The first match is not triggered because our request is for `linkmap.xml` and this match handles `linkmap.source.xml` to essentially re-direct it to `linkmap.xml` instead. That is what the cocoon:// means: generate it via a different request within this sitemap. Try `localhost:8888/linkmap.source.xml` to see the exact same internal xml format.

The second match exactly meets our pattern `linkmap.xml`

```
<map:match pattern="linkmap.xml">
  <map:generate src="cocoon://abs-linkmap" />
  <map:transform src="{lm:transform.linkmap.document}"/>
  <map:serialize type="xml" />
</map:match>
```

As with all pipelines, it starts with a generator to commence the xml stream, then transforms it with a single transformer (there could be multiple sequential transformers) and finally the serializer component. Here it is:

The generator is not simply reading an xml file. It produces the xml via a different part of this sitemap. Let us explain that later and assume for now that it produces the xml from your site.xml file.

Move on to the transformer. It transforms the xml obtained from the site.xml into the internal document xml format using an XSLT stylesheet. The locationmap reference defines the source for that stylesheet: "lm:transform.linkmap.document" is evaluated by the Locationmap to be the `main/webapp/resources/stylesheets/linkmap-to-document.xsl` stylesheet. See the Locationmap documentation for explanation.

Now let us get back to that new request for "abs-linkmap". This is used a number of times within this sitemap, hence it is its own pipeline. As usual it starts with a generator, then a transformer, then a serializer.

Again the generator is sent to some other part of the sitemap hierarchy, because this request is needed by many other parts of the system beyond just this linkmap handling. You see that it is not matched within this linkmap.xmap sitemap. Go to the main sitemap.xmap and search for "site.navigation.links.xml" where you find the match that handles this by looking for various Locationmap definitions to find and transform the site.xml file.

Don't get lost, come back to the linkmap.xmap sitemap.

Following this generator, the transformer turns the links into absolute references. This is then serialized as xml to finish this "abs-linkmap" match which is the end of the generator in our main match.

A developer's trick will help to understand what is happening. Edit the linkmap.xmap to comment-out the transformer ...

```
      <map:match pattern="linkmap.xml">
        <map:generate src="cocoon://abs-linkmap" />
<!--
        <map:transform src="{lm:transform.linkmap.document}"/>
-->
        <map:serialize type="xml" />
      </map:match>
```

Browser `localhost:8888/linkmap.xml` to see the result of the "abs-linkmap" generation before it is transformed into the internal document xml.

So now you understand some of the power of sitemaps.

A basic understanding of Cocoon's pipelines and their components will help you to realise the true power. You should know about matchers, generators, transformers and serializers and have a rough

idea how they work together in a pipeline. A good place to start learning about Cocoon is Understanding Apache Cocoon. The Forrest Sitemap Reference will also be helpful.