SimpleTransformations

How To Do Simple XSLT Transforms With Cocoon

Arguably the simplest Cocoon application is one that simple serves files that are dynamically generated using XSLT.

This HOW-TO assumes that

- you know how to use XSLT already, and
- you already have sample XML data to transform

The only additional information you need is how to configure the Sitemap to do a transformation, this is described in the following sections.

The XSLT Transformer

Cocoon provides an XSLT transformer component which is already configured in the example Sitemap. If you're starting from a MinimalSitemapConfiguration then here's the declaration that you need:

```
<map:transformers default="xslt">
   <map:transformer name="xslt" src="org.apache.cocoon.transformation.TraxTransformer"/>
</map:transformers>
```

Note: if you're using a sub-sitemap, then you don't need to configure the transformer component again if it's already defined in the root or parent sitemap. For more information see (MinimalSitemapConfiguration and UnderstandingCocoonMounts)

The Basic Pipeline

Here's a basic pipeline that responds to requests for `http://your.server.com/cocoon/my.html`
by parsing a file called `my.xml`, transforms it with a stylesheet `my.xsl`, and serializes the results as HTML:

```
<map:pipeline>
  <map:match pattern="my.html">
    <map:generate src="my.xml"/>
    <map:transform src="stylesheets/my.xsl"/>
    <map:serialize/>
  </map:match>
</map:pipeline>
```

Things to note here:

- The Matcher is responsible for making this pipeline respond to a request for the above URL.
- The Generator is responsible for parsing the document
- The Transformer invokes the XSLT transformation, using the provided stylesheet
- The Serializer is responsible for returning the results back to the user in the HTTP response. Here the pipeline is using the default HTML serializer.

If you want to serialize the results as XML, then use the XML Serializer:

```
<map:serialize type="xml"/>
```

If you want to serialize the results as XHTML (the XHTML Doctype will be automatically added), then use the XHTML Serializer:

```
<map:serialize type="xhtml"/>
```

Passing Parameters

It's possible to have Cocoon pass parameters to the XSLT processor that can be accessed from your stylesheet.

For this to work correctly, you must have first declared the parameter in the XSLT stylesheet using the `xsl:param` element.

Fixed Parameters

The `<map:parameter/>` element can be used to pass a fixed value to a stylesheet, as follows:

```
<map:transform src="stylesheets/my.xsl">
  <map:parameter name="your-parameter-name" value="your-parameter-value"/>
</map:transform>
```

Sitemap Parameters

You can also use the `<map:parameter/>` element to pass dynamic Sitemap parameters to the XSLT processor using the curly brace syntax.

```
<map:transform src="stylesheets/my.xsl">
  <map:parameter name="your-parameter-name" value="{1}"/>
</map:transform>
```

These parameters are set by Matchers and/or Actions.

Request Parameters

You can pass request parameters to the XSL processor, as follows:

```
<map:transform src="stylesheets/my.xsl">
  <map:parameter name="use-request-parameters" value="true"/>
</map:transform>
```

When declaring the Components (see **XSLT Transformer**, above) you can tell it to automatically use request parameters for *all* transformations like this:

```
<map:transformer name="xslt" src="org.apache.cocoon.transformation.TraxTransformer">
   <use-request-parameters>true</use-request-parameters>
</map:transformer>
```

Note that passing request parameters in this way can affect how Cocoon Caching the results of your transformation.

Further Reading

- Read about Matchers for how to use wildcards in your pipeline, allowing more flexibility in the URLs you match on.
- Read about other Serializers if you want to produce other kinds of output
- Read about Resources, which can be used to reuse common Pipeline configurations.
- Read about how to add ErrorHandling to your Pipeline
- Read about how to ServingStaticFiles. Useful if you're creating HTML pages that reference images, etc.