Specifying Different XSLT Processor Options (2.1 legacy document)

Table of contents

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

Table of contents

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Overview

This Snippet shows how to specify different options of the same XSLT processor when processing different pipeline requests. The following example is based on the need to turn incremental processing both off and on. This is useful in situations when processing requests for both PDF and HTML output. For example, you may want to turn incremental processing "off" for PDF requests but leave it "on" for HTML requests. In addition, you can follow the **same** approach in order to utilize two **different** XSLT processors when processing requests.

2. Version

At the time of this writing, this Snippet was tested against the Cocoon version 2.0.3. Please note that this approach does not work with Cocoon 2.1. Stay tuned for an upcoming Snippet which illustrates a different approach for version 2.1.

3. cocoon.xconf

Here is a snippet from cocoon.xconf which declares a default XSLT processor. It assumes the use of Xalan. You can see that the default XSLT processor is configured to allow the incremental processing of SAX events. Check your cocoon.xconf file, as the value for incremental-processing may be different from what is shown here.

<xslt-processor class="org.apache.cocoon.components.xslt.XSLTProcessorImpl" logger="core.xslt-processor"> <parameter name="use-store" value="true"/> <parameter name="incremental-processing" value="true"/> </xslt-processor>

As discussed above, we need the ability to turn incremental processing of our XSLT processor both on and off. Even though we are using the same XSLT processor implementation, we need to declare an additional component in our cocoon.xconf file in order to use the XSLT processor differently. So, in the next snippet, we add a new component (as a child element of the root cocoon element) to cocoon.xconf. Don't be concerned if you don't see any other component declarations like this in the particular cocoon.xconf file you happen to be using.

<component role="org.apache.cocoon.components.xslt.XSLTProcessor/NotIncremental" class="org.apache.cocoon.components.xslt.XSLTProcessorImpl"> <parameter name="use-store" value="true"/> <parameter name="incremental-processing" value="false"/> </component>

In the snippet above, the component is identified by the role it plays. Note the value for the role attribute: "org.apache.cocoon.components.xslt.XSLTProcessor/NotIncremental". We will use this value in additional snippets below. Note also that the incremental-processing parameter and its value of "false" is also specified in this particular snippet.

4. sitemap.xmap

Next, we need to add one more XSLT transformer component to the components declaration section of our sitemap.xmap file.

<map:components> <!-- other map:components here --> <map:transformers> <!-- other map:transformers here --> <map:transformer name="xslt-notinc"

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

src="org.apache.cocoon.transformation.TraxTransformer" logger="sitemap.transformer.xslt" pool-max="32"> <use-request-parameters>false</use-request-parameters> <use-browser-capabilities-db>false</use-browser-capabilities-db> <use-deli>false</use-deli> <xslt-processor-role> org.apache.cocoon.components.xslt.XSLTProcessor/NotIncremental </xslt-processor-role> </map:transformer> <!-- other map:transformers here --> </map:transformers> <!-- other map:components here --> </map:components>

In the snippet above, we are using the value for the role attribute, specified earlier in the cocoon.xconf snippet, for the value of xslt-processor-role element. Note also that we are giving this transformer the name "xslt-notinc". Please note that the text node for xslt-processor-role was manually broken across several lines for page viewing purposes only. Do not do this in your sitemap.xmap file.

Finally, in the relevant map:match elements in our sitemap.xmap, we can specify exactly which xslt transformer component to use by supplying its name as map:transformer's type attribute. Remember that the default xslt transformer, as declared in cocoon.xconf, has incremental processing turned on. Because it's the *default* transformer, you don't need to specify its name in map:transform's type attribute. In fact, you don't have to supply a type attribute at all when you use default components.

<map:pipeline> <!-- incremental processing = false --> <map:match pattern="*.pdf"> <map:generate src="docs/{1}.xml"/> <map:transform type="xslt-notinc" src="stylesheets/page2fo.xsl"/> <map:serialize type="fo2pdf"/> </map:match> <!-- incremental processing = true --> <map:match pattern="*.html"> <map:generate src="docs/{1}.xml"/> <map:transform src="stylesheets/page2html.xsl"/> <map:serialize type="html"/> </map:match> </map:pipeline>

5. Comments

Care to comment on this Snippet? Got another tip? Help keep this Snippet relevant by passing along any useful feedback to the cocoon users list.

1. Comments

add your comments

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.