MigrateToCocoon2

Purpose of this page

The purpose of this document is to assist current users of Cocoon 1.x migrate their XML documents to Cocoon 2. The irony of the situation is that the fixes and refactorings in Cocoon 2 that make up for the shortcomings of Cocoon 1 also act as a barrier (albeit a very easily overcome one) to migrating to Cocoon 2.

A little background

The most glaring deficiency in Cocoon 1 is an incomplete separation of concerns. Whereas Cocoon 2's sitemap controls the processing flow from beginning to end, Cocoon 1 requires developers to embed processing instructions directly in the source XML document. *Id est*, the view of the content is dependent upon information embedded in the content itself. An example of this dependency is shown below in **Sample 1**.

Sample 1: Processing instructions in an XML document

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="http://www.somesite.com/xsl/xml_to_html.xsl" type="text/xsl"?>
```

As you can see in the above sample, there are two things that violate separation of concern. First is the `<?cocoon-process type="xslt"?>` instruction, because the document specifies how the document's content should be processed, through `xslt`. The second violation is `<?xml-stylesheet href="http://www.somesite.com/xsl/xml_to_html.xsl" type="text/xsl"?>`, as the document specifies which stylesheet will accomplish the transformation. Cocoon 2 is a clear improvement, as neither of these elements is required in the document in order for the document to be processed properly.

How do you remove the unnecessary processing instructions? If you really "get" what XML is about, you know what's about to come next: use XSLT to transform the old Cocoon 1 XML documents into shiny new XML documents that Cocoon 2 can use. The following code samples show you how.

Some samples to show you how

Sample 2: *old-doc.xml*

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="http://www.somesite.com/xsl/xml_to_html.xsl" type="text/xsl"?>
<doc-rt>
<doc-sec-one>
<abbr-element-1><abbr-elem-2>Some text inside this tag</abbr-elem-2></abbr-element-1>
<abbr-elem-2>Next piece of content</abbr-elem-2>
</doc-sec-one>
<doc-section-2>Something goes in here...
</doc-section-2>
</doc-rt>
```

As you can see, **sample 2** has some problems: the processing instructions that we want to remove, the pattern of the element names is not consistent, and the structure could use some clarifying. The XSL stylesheet in **sample 3** *below* shows you how.

Sample 3: *old-to-new.xsl*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<new-document-root>
<xsl:apply-templates/>
</new-document-root>
</xsl:template>

<xsl:template match="doc-sec-one">
<section-one>
<xsl:apply-templates/>
</section-one>
</xsl:template>

<xsl:template match="abbr-element-1">
<xsl:apply-templates/>
</xsl:template>

<xsl:template match="abbr-elem-2">
<new-element-one>
<xsl:apply-templates/>
</new-element-one>
</xsl:template>

<xsl:template match="doc-section-2">
<section-two>
<xsl:apply-templates/>
</section-two>
</xsl:template>

</xsl:stylesheet>
```

Now comes the answer to the question "But how do I actually DO this?" Xalan-Java. If you don't know if Xalan-Java is on your classpath, here's how to find out. Open a command shell (Windows users: *Start* -> *Run* -> Type in *cmd.exe*). From the command prompt, enter `java org.apache.xalan.xslt.Process`. If you get an error or exception, Xalan is not on your classpath. Xalan is a core component of Cocoon 2. If you've already downloaded Cocoon 2, the necessary jars:

- `xalan-2*.jar`
- `xml-apis.jar`
- `xerces*.jar`

(where the asterisk is a wildcard) are already on your system, they are in the `cocoon-2.*.*/lib/core` directory. Just add them to your classpath. If you haven't downloaded Cocoon 2 yet, as an alternative, you can download Xalan-J from the http://xml.apache.org website.

Once you have Xalan and its necessary jars in your classpath, try copying and pasting the code for **samples 2** and **3**. Save them as `old-doc.xml` and `old-to-new.xsl`, respectively. From the command prompt, enter `java org.apache.xalan.xslt.Process -IN old-doc.xml -XSL old-to-new.xsl -OUT new-doc.xml`. The result should be the same as in **sample 4**.

Sample 4: *new-doc.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<new-document-root>
<section-one>
<new-element-one>Some text inside this tag</new-element-one>
<new-element-one>Next piece of content</new-element-one>
</section-one>
<section-two>Something goes in here...
</section-two>
</new-document-root>
```

If the output you got matches what is in **sample 4**, congratulations! It's now a comparitively simple matter to write a script or batch file that automatically processes your files through Xalan. Compared to editing every document by hand, that is, which is time consuming, tedious, and prone to error. More information about using the Xalan command line can be found at http://xml.apache.org/xalan-j/commandline.html.

Now the real work begins

Take the time to examine a your XML. If, like me, your documents were generated periodically in batches, take a random sample form each 'generation' of documents and examine the markup in them. Of course, core files like indexes and tables of contents or other high-dependance pages should be evaluated individually. I found a lot of room for improvement in my application of XML, and so, I suspect, will you. If you have made it thus far, gentle reader, here is a little reward for your perseverance: If you're in a hurry, just write a stylesheet that copies all the nodes below the root to your new XML document.

A Good Freeware Program To Help You Out

Alright! I built a GUI to handle the processing discussed on this page. No tedious batch files! Simple XSLT GUI I hope that you find this useful. Please feel free to email me with your comments.

Remember: *It is far better to make your technology work for you, than for the opposite to be true.*

Reader Comments

About your XSP pages:

- You need to remove the response redirection from the xsps.
- Also with cocoon1 it was possible to write an xsp:logic block and then a xsp:attribute block. This is not anymore possible with cocoon2 xsp. You will not get error messages but your attributes will not be present in the output.
- If you are using helper classes and methods that throw exceptions, you will perhaps need to catch them in your xsps.