DebuggingXMLandXSLTransformation

The problem

Though very important, SAX events have some drawbacks. One of the drawbacks is that by using event-based transformers (TransformationHandler ContentHandler), it is difficult to not get lost in hundreds lines of the dumped method stack.

The solution

A debug mode might be a solution, which may consume some time - that would not be much of a problem, since errors like mal-formed style sheets or invalid xml files are things, which clearly should not occur in a production environments.

I have to look at the SAXTransformer components, and how introducing a kind of Context for better error handling should work.

Other tips

Please fill free to add your ideas here:

- The build targets "validate-config" (which also validates the xsl stylesheets) and "validate-xdocs" will help to eliminate some errors.
- HTML-Kit is great app for debugging XML – you can validate the XML or check the well-formedness of your files. Best of all – it's free. MichaelTiffany
- Try the transformations offline: Shorten your pipeline as much as possible, so that you can still reproduce the error, but you already exclude possible sources of errors. If a transformation fails, save the XML before the transformation to disk (e.g. by adding a <map:serialize type="xml"/> in front of the transformer) and test it with your stylesheet.

Another Proposal

In this message, Bart Guijt suggested the following approach:

"I am using Cocoon for almost a year now and if one thing bugs me, it is this: if something fails to operate successfully, the Java stacktraces fail most of the time to correctly show how the error occurred.

In Cocoon, we use sitemaps, XSL transformers, (additional) Java components and URI resolving to get the job done. If something fails, we get a (possibly very deep!) stacktrace from the SAX pipeline, a compiled XSLT stylesheet, the appserver and the treeprocessor in no particular order, with sometimes very misleading information.

What about a stacktrace like this:

```
Error <foo bar> occurred at:
Java: org.apache.cocoon.components.SomeComponent.Configure:237:4
    Sitemap: /test/sitemap.xmap:25 (map:generate src="..."
type="SomeComponent")
    Sitemap: /test/sitemap.xmap:24 (map:match pattern="someresource")
    Sitemap: /sitemap.xmap:20 (map:mount check-reload="yes" src="{1}/"
uri-prefix="{1}" - {1}="test")
    Sitemap: /sitemap.xmap:19 (map:match pattern="*/**")
    URI: cocoon://test/someresource
    XSLT: /xsl/foo-bar.xslt:454:53 (document())
    XSLT: /xsl/foo-bar.xslt:25:10 (template: /)
    Sitemap: /sitemap.xmap:30 (map:transform src="xsl/foo-bar.xslt")
    Sitemap: /sitemap.xmap:28 (map:match pattern="page.html")
    URI: page.html
```

The information provided is probably mentioned somewhere in the Cocoon logs, but is scattered all over the place. The logs are a collection of events in no particular order (multiple users) and are not traceable to a single session. Even if you could trace these events leading to failure, it is a PITA to figure out what the execution stacktrace actually was!

I don't know much about any Cocoon treeprocessor internals (Sylvain?), but is such a thing possible? It would probably be a little difficult with the XSLT part, but even then: Xalan fires trace events which are perfectly suitable for this job.

So... how about it?

For the Cocoon 2.2 version of course!!"