XModuleSource

Scratchpad component

Overview

Read and write XML data (DOM and `XMLizable`) from input and output modules. This can replace the `[''''[Read]DOMSessionTransformers` and should also be useful in flowscripts together with processToStream.

Read and writable source that are accessed with uri:s like:

`xmodule:[<input-module>|<output-module>]:<attribute-name>[#xpath]`

For reading the object that is found by applying the XPath (`JXPath`), on the attribute from the input-module, is supposed to be a DOM Document, a DOM Node or an `XMLizable` object, the object is serialized to SAX.

For writing the input is serialized into a DOM Document, this document is put in the attribute from the output-module if the XPath is empty. If the XPath not is empty, an input-module is used to find the attribute and JXPath is applied. If the object that is found is a DOM tree, the input document is imported into it, otherwize it is just assigned into that position.

There is a delete fuction as well, that use `removeAll(xpath)` from `JXPath`.

Flow example

http://marc.theaimsgroup.com/?l=xml-cocoon-dev&m=107279968120084&w=2

Configuration

`RequestAttributeOutputModule` and `SessionAttributeOutputModule` as default prefix all attribute names with `org.apache.cocoon.components.modules.output.OutputModule`.

To make the samples for the xmodule source work this must be reconfigured to using attribute names without prefixes. This is done in the cocoon.xconf by puting an empty key-prefix" element:

`<key-prefix/>`

as child to the configurations of the output-modules "request-attr" and "session-attr".

To make writing with or without XPaths work in a decent way, there is supposed to be both an input and an output module that are configured to have the same name and that gets and sets the same attribute.

Relation to [Read|Write]SessionTransformers

For the `[[Read|Write]DOMSessionTransformers` as well as the `[[Read|Write]DOMTransformers` submitted in http://nagoya.apache.org/bugzilla/show_bug.cgi?id=23921, I would guess that the read functionality can be replaced by using URIs like:

`xmodule:session-attr:field`

together with the `[[C|X]IncludeTransformer` or the `FileGenerator`. The write functionality can probably be replaced by using URIs on the same form together with the `SourceWritingTransformer` or doing the writing by using `processToStream` within flowscripts.

Writing to the xmodule stream is less efficient than using the `WriteDOMSessionTransformer`, as a serialize/reparse step is needed. This would be quite easy to fix, what we need is a convention for what interface a source that one can write SAX to should implement. I think that it would be most convinient to implement the `XMLConsumer` interface. Then the `SourceWritingTransformer` can check if the source implements `XMLConsumer` and in that case redirect the input events to that instead. For use within flowscripts, `processToSAX` can be used.

Future work

A nice enhancement would be to let the `XModuleSource` implement `ContentHandler`, then one would avoid the serialize/parse step in some cases. It could e.g. be used with `processToSAX` in flowscripts.