TEST module

Test document for generating module documentation

Erik Siegel - Xatapult
2019-11-08 11:04:48

0 Table of Contents

0 Preface part

Can we do something with the preface part, like using it for a home page?

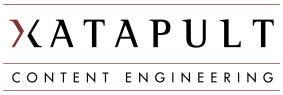

Figure 0-1 - An example of a figure...

1 Introduction

WARNING:
This document is for testing the features of the `xtpxlib-xdoc` component.

The `xtpxlib-xdoc` module uses DocBook 5.1 as its target vocabulary. However, it does not implement the full standard (which is huge!) but only those elements/attributes that were deemed necessary. The `xtpxlib-xdoc` component adds several extensions which are tested by running this document through the conversion pipelines. This means it also serves as partial documentation and example.

2 Stuff 1

Link to another chapter: chapter 3
Link to this chapter: "Parameters" on page 4

2.1 Parameters

Parameter	Value
DATE	2019-11-08
DATETIME	2019-11-08 11:04:48
HREF-PARAMETERS	
HREF-SOURCE	file:///C:/Data/Erik/work/xatapult/xtpxlib-xdoc/test/xdoc-moduledoc-test.xml
TIME	11:04:48

Table 2-1 - Parameters

2.2 Coming from transformations

Comes from an XProc transformation:
Line 1 from test.xpl
Line 2 from test.xpl

Comes from an XSLT transformation:
From the test.xsl

3 Stuff 2

3.1 XML Descriptions

```
<p:declare-step name? = xs:NCName
                type? = xs:QName
                psvi-required? = xs:boolean
                xpath-version? = xs:decimal
                exclude-inline-prefixes? = xs:string
                version? = xs:decimal
                visibility? = "public" | "private" >
  ( <p:import> |
    <p:import-functions> )*
  ( <p:input> |
    <p:output> |
    <p:option> |
    <p:variable> )*
  <p:declare-step>*
  <!-- Step's body… -->
</p:declare-step>
```

Main description of p:declare-step...

The attribute-table-header, yoho!

Attribute	#	Type	Description
`name`	?	`xs:NCName` Bla bla bla additioal type description	The name of this step. You need this name to refer to its ports from within the step itself.
`type`	?	`xs:QName`	The value of the `type` attribute is used to invoke/call this step. A step's type *must* always be in a namespace, so it *must* use a namespace prefix (and of course this prefix must be declared). You cannot use the XProc `p:` namespace.
`psvi-required`	?	`xs:boolean`	Default: `false` Whether or not PSVI ("Post Schema Validation Infoset") support is required. PSVI allows you to carry type information together with your XML document. PSVI information is added when validating documents against a schema. Advanced XSLT and other transformations can use this to make decisions based on the type of something (e.g. elements or attributes). If your XProc transformation requires PSVI support, set this attribute to `true`. This will make sure that non-PSVI-supporting XProc processors will not even dare to execute them.
`xpath-version`	?	`xs:decimal`	Default: `3.1` The XPath version that must be used for evaluating the XPath expressions within the pipeline. The only value XProc 3.0 processors are required to support is `3.1`. It's not allowed to specify a version lower than `3.1` and whether your XProc processor supports higher values is implementation defined. Versions If a new version of XPath arrives (if ever) *and* the latest and greatest release of your XProc processor starts supporting this, there might be a good reason to change it to the new (higher) value. In all other cases I would say: don't bother.
`exclude-inline-prefixes`	?	`xs:string`	Defines what to do with inline namespace definitions in your documents

Attribute	#	Type	Description
`version`	?	`xs:decimal`	This identifies the XProc version for your pipeline. **Important:** This attribute is *mandatory* on `<p:declare-step>` elements that are the root element of a document. It is *optional* on nested `<p:declare-step>` elements (if not set, these inherit the version setting from their parent). If specified, its value *must* be `3.0`. Specifying a different value is an error.
`visibility`	?	`xs:string`	Default: `public` Controls whether this step is visible for the importing pipeline when part of a library (`<p:library>`, see . If this step is not part of a library the attribute is ignored. Value: `public` — The step is visible to the importing pipeline Value: `private` — The step is *in*visible to the importing pipeline and can only be used in the library itself.

The elements in the prolog of a `<p:declare-step>` are about:

- Imports of external XProc pipelines, libraries or XPath function libraries.
- Declaring static variables.
- Declarations of input ports, output ports, options and/or variables.
- Declarations of locally defined steps.

Child element	#	Description
`p:import`	*	Imports a step from an external source. You can also use this to import step *libraries*.
`p:import-functions`	*	Imports functions (not global variables) from another language so these become useable inside the XPath expressions in your pipeline. The most common ones are XSLT and XQuery libraries, but other languages are not ruled out. Whether importing functions is supported is implementation specific.
`p:input`	*	Declares an input port for this step.
`p:output`	*	Declares an output port for this step.
`p:option`	*	Declares an option for this step.
`p:variable`	*	Declares a variable. Variables declared in the prolog must be *static* (they must always carry the `static="true"` attribute).
`p:declare-step`	*	Use nested `<p:declare-step>` elements to declare steps that are local to the encompassing step. The effect is the same as for steps imported by a `<p:import>` element. You may also use this element to declare an *atomic step*.

3.2 Documentation generation

3.2.1 XSLT Module

File: `test.mod.xsl`

XSLT library module with general constants and code.

Prefix	Namespace
`xtlc`	`http://www.xtpxlib.nl/ns/common`

Table 3-4 - Namespaces in test.mod.xsl

Parameter	Type	Required	Default	Description
xyz	xs:string	true		Trala la la
xyz2	xs:string	false	'yeah'	Trala la la more...

Table 3-5 - Parameters in test.mod.xsl

Variable	Description
$xtlc:default-dt-format	Default date/time format string (yyyy-mm-dd ...).
$xtlc:default-dt-format-en	Date/time format string (English: mm-dd-yyyy ...).
$xtlc:default-dt-format-nl	Date/time format string (Dutch: dd-mm-yyyy ...).
$xtlc:internal-error-prompt	Add this in front of any internal error raised.
$xtlc:language-en	Language code for English
$xtlc:language-nl	Language code for Dutch
$xtlc:namespace-xtlc-common	Name of the xtpxlib common namespace.
$xtlc:status-codes	Sequence with all valid status codes.
$xtlc:status-debug	Generic debug status/severity code.
$xtlc:status-error	Generic error status/severity code.
$xtlc:status-info	Generic info (a.k.a. OK) status/severity code.
$xtlc:status-warning	Generic warning status/severity code.

Table 3-6 - Variables in test.mod.xsl

Function	Description
xtlc:att2str	Turns an attribute into a string representation, suitable for display.
xtlc:capitalize	Capitalizes a string (makes the first character uppercase).
xtlc:char-repeat	Returns a string with a single character repeated a given number of times.
xtlc:elm2str	Turns an element into a descriptive string (the element with all the attributes (excluding schema references).
xtlc:item2element	Tries to find the element belonging to a given item:
xtlc:items2str	Creates a string from a sequence of items. Useful for easy creation of messages consisting of multiple parts and pieces.
xtlc:prefix-to-length	Prefixes a string with a given character so it will get at least a given length.
xtlc:q	Returns the input string quoted ("$in")
xtlc:str2bln	Safe conversion of a string into a boolean.
xtlc:str2id	Turns a string into a valid identifier, adding a prefix.
xtlc:str2id	Turns a string into a valid identifier.
xtlc:str2int	Safe conversion of a string into an integer.
xtlc:str2seq	Converts a string with a list of words into a sequence of words.

Table 3-7 - Functions in test.mod.xsl

Named template	Description
xtlc:raise-error	Stops any processing by raising an error.

Table 3-8 - Named templates in test.mod.xsl

Function: xtlc:att2str => xs:string

Turns an attribute into a string representation, suitable for display.

Parameter	Type	Description
att	attribute()?	Attribute to convert.

Function: xtlc:capitalize => xs:string

Capitalizes a string (makes the first character uppercase).

Parameter	Type	Description
in	xs:string	The string to work on.

Function: xtlc:char-repeat => xs:string

Returns a string with a single character repeated a given number of times.

Parameter	Type	Description
char	xs:string	The first character of this string is the character to repeat. If empty, an empty string is returned.
repeat	xs:integer	The number of repeats. If <= 0, an empty string is returned.

Function: xtlc:elm2str => xs:string

Turns an element into a descriptive string (the element with all the attributes (excluding schema references).

Parameter	Type	Description
elm	element()?	Element to convert

Function: xtlc:item2element => element()?

Tries to find the element belonging to a given item:

- When the item is of type xs:string or xs:anyURI, it is assumed to be a document reference. The root element of this is returned.
- When the item is of type document-node(), the root element of this document is returned
- When the item is of type element(), this is returned

You can choose whether to produce an error message or () when the item cannot be resolved.

Parameter	Type	Description
item	item()	The item to work on
error-on-non-resolve	xs:boolean	Whether to generate an error when $item could not be resolved. Otherwise, the function will return ().

Function: xtlc:items2str => xs:string

Creates a string from a sequence of items. Useful for easy creation of messages consisting of multiple parts and pieces.

Parameter	Type	Description
items	item()*	The message parts to combine

Function: xtlc:prefix-to-length => xs:string

Prefixes a string with a given character so it will get at least a given length.

Parameter	Type	Description
in	xs:string	String to prefix
prefix-char	xs:string	String to prefix with. Only first character is used. If empty, a * is used.
length	xs:integer	The length to reach.

Function: xtlc:q => xs:string

Returns the input string quoted ("$in")

Parameter	Type	Description
in	xs:string?	String to convert.

Named template: xtlc:raise-error

Stops any processing by raising an error.

Parameter	Type	Required	Default	Description
msg-parts	item()+	true		Error message to show (in parts, all parts will be concatenated by xtlc:items2str()).
error-name	xs:string	false	$xtlc:status-error	The (optional) name of the error. Must be a NCName.

Function: xtlc:str2bln => xs:boolean

Safe conversion of a string into a boolean. When $in is empty or not convertible into a boolean, $default is returned.

Parameter	Type	Description
in	xs:string?	String to convert.
default	xs:boolean	Default value to return when $in is empty or cannot be converted.

Function: xtlc:str2id => xs:string

Turns a string into a valid identifier, adding a prefix. All characters that are not allowed in an identifier are converted into underscores. When the result does not start with a letter or underscore, the extra prefix 'id-' is added.

Parameter	Type	Description
in	xs:string	String to convert.
prefix	xs:string?	Prefix to apply.

Function: xtlc:str2id => xs:string

Turns a string into a valid identifier. All characters that are not allowed in an identifier are converted into underscores. When the result does not start with a letter or underscore, the extra prefix 'id-' is added.

Parameter	Type	Description
in	xs:string	String to convert.

Function: xtlc:str2int => xs:integer

Safe conversion of a string into an integer. When $in is empty or not convertible into an integer, $default is returned.

Parameter	Type	Description
in	xs:string?	String to convert.
default	xs:integer	Default value to return when $in is empty or cannot be converted.

Function: xtlc:str2seq => xs:string*

Converts a string with a list of words into a sequence of words.

Parameter	Type	Description
in	xs:string?	String to convert.

3.2.2 XPROC Module

File: data/test.mod.xpl

XProc library with generic steps.

Prefix	Namespace
xtlc	http://www.xtpxlib.nl/ns/common

Table 3-23 - Namespaces in test.mod.xpl

Step	Description
xtlc:copy-directory	Copies a full directory structure.
xtlc:copy-file	Copies a file, if necessary from inside a zip file.
xtlc:log	Writes a message to a log file.

Step	Description
xtlc:recursive-directory-list	Returns the contents of a directory, going into sub-directories recursively.
xtlc:remove-dir	Removes a full directory (since the normal processing using pxf:delete does not work properly some older Calabash versions... :-( )
xtlc:tee	Tees the input to a file and passes it unchanged (like the Unix tee command).
xtlc:zip-directory	Zips a directory and its sub-directories into a single zip file.

Table 3-24 - Steps in test.mod.xpl

Step: xtlc:copy-directory

Copies a full directory structure.

Port	Type	Description
source	input, primary	Input, will be passed unchanged.
result	output, primary	The input unchanged.

Option	Required	Description
href-source-dir	true	Reference to the directory to copy from.
href-target-dir	true	Reference to the directory to copy to.

Step: xtlc:copy-file

Copies a file, if necessary from inside a zip file.

IMPORTANT: For older versions of Calabash (before January 2017) there is a huge bug in this step: A file inside a zip file must have a straight filename without any characters that normally would have been escaped (like, the most inportant one, spaces).

Port	Type	Description
source	input, primary	Input, will be passed unchanged.
result	output, primary	The input unchanged.

Option	Required	Default	Description
href-source	true		Reference to the source file to copy.
href-source-zip	false	''	Document reference to a zip file. When filled, $href-source is assumed to be a path inside this zip.
href-target	true		Reference to the target.
enable	false	true()	Whether the copying is done at all.

Step: xtlc:log

Writes a message to a log file.

Port	Type	Description
source	input, primary	Input to the logging, will be passed unchanged to the output
result	output, primary	The input unchanged.

Option	Required	Default	Description
href-log	true		Name of the file to write the logmessages to (must have a leading file:// specifier!)
enable	false	true()	Whether the logging is done at all.
status	false	'ok'	Status of the message. Must be ok, warning, error or debug.
message	true		The actual log message
keep-messages	false	100	The number of messages to keep in the logfile. If le 0, all messages are kept. Set by default to 100 to prevent overflowing files...

Step: xtlc:recursive-directory-list

Returns the contents of a directory, going into sub-directories recursively. When the requested directory does not exist, it returns only a c:directory root element with @error="true".

Adapated from Norman Walsh example code at https://github.com/xquery/xquerydoc/blob/master/deps/xmlcalabash/recursive-directory-list.xpl

Port	Type	Description
`result`	`output, primary`	The resulting directory structure listing in XML format.

Option	Required	Default	Description
`path`	`true`		The path to get the directory listing from.
`include-filter`	`false`		An optional regexp include filter.
`exclude-filter`	`false`		An optional regexp exclude filter.
`depth`	`false`	`-1`	The sub-directory depth to go. When le 0, all sub-directories are processed.
`flatten`	`false`	`false()`	When true, the list will be "flattened": Only c:file children within the root c:directory element. All c:file elements have a @name, @href-abs (absolute filename) and @href-rel (relative filename) attribute.

Step: `xtlc:remove-dir`

Removes a full directory (since the normal processing using pxf:delete does not work properly some older Calabash versions... :-( ) When the directory does not exist everything continues without error.

Port	Type	Description
`source`	`input, primary`	Input, will be passed unchanged.
`result`	`output, primary`	The input unchanged.

Option	Required	Default	Description
`href-dir`	`true`		Reference to the directory to remove.
`enable`	`false`	`true()`	Whether the removal is done at all.

Step: `xtlc:tee`

Tees the input to a file and passes it unchanged (like the Unix tee command).

Port	Type	Description
`source`	`input, primary`	Input to the tee.
`result`	`output, primary`	The input unchanged (unless $root-attribute-href was specified).

Option	Required	Default	Description
`href`	`true`		Name of the file to write to (must have a leading file:// specifier!)
`enable`	`false`	`true()`	Whether to actually do the write. When false nothing happens.
`root-attribute-href`	`false`	`''`	If filled, $href is recorded as an attribute with this name on the root element of the original input. Must be a valid attribute name.

Step: `xtlc:zip-directory`

Zips a directory and its sub-directories into a single zip file.

Port	Type	Description
`result`	`output, primary`	The output of the actual zip step, listing all the files that went in

Option	Required	Default	Description
`href-target-zip`	`true`		Document reference for the zip file to produce (must have a leading file:// specifier!)
`include-base`	`false`	`true()`	When true, the last part of $base-path (e.g. a/b/c ==> c) is used as the root directory in the zip file.
`base-path`	`true`		Directory which contents will be stored in the zip (must have a leading file:// specifier!)

3.2.3 XPROC Step

File: `C:/Data/Erik/work/xatapult/xtpxlib-xdoc/test/data/test.xpl`
TBD

Port	Type	Description
`source`	`input, primary`	The document to generate documentation for, wrapped in an xdoc:transform element.
`result`	`output, primary`	The resulting docbook 5 output

Option	Required	Description
`blabla`	`true`	More bla

3.2.4 XSD

File: `C:/Data/Erik/work/xatapult/xtpxlib-xdoc/test/data/test.xsd`
This schema describes the intermediate format used for docgen

Global element	Description
`document`	The description of a (code) document accoriding to the docgen intermediate format

3.2.5 XML

Some utter nonsense