Saxon - Bug #4035

Exporting a stylesheet containing node-valued static variables

2018-11-19 18:27 - Michael Kay

Status:	Closed	**Start date:**	2018-11-19
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	100%
Category:	XSLT export	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fix Committed on Branch:**	9.9, trunk
Applies to branch:	9.9, trunk	**Fixed in Maintenance Release:**	9.9.1.5

Description

It is possible to create and export a stylesheet containing a node-valued static variable. For example:

```
<xsl:variable name="e" static="true" select="parse-xml(.....)//x"/>
```

There are restrictions here which are not currently documented. We should review these restrictions and document them.

The restrictions include:

- We can only export document and element nodes (not, for example, text nodes, attributes, and comments)
- The nodes, after re-importing, will be parentless (whereas the originals, as in the above example, might have parents)
- Relationships between nodes are lost: in the above example, ($x[1]/following-sibling::x is $x[2]) might be true in the original stylesheet, false after exporting and importing.

Related issues:

Related to Saxon-JS - Bug #4829: Static variable initialised to node values c...	**New**	**2020-11-18**

History

#1 - 2018-11-19 18:34 - Michael Kay

I am wondering whether it would be simpler and safer to have a more coarse restriction: "A stylesheet containing node-valued static variables cannot be exported".

Or perhaps we should only allow document nodes. Even then, some of the properties are lost, such as base URI, document URI, and "$x is $y" where $x and $y are two exported document nodes.

Another approach would be export not the value of the static variable, but the expression for constructing its value, and to re-evaluate this expression when the SEF file is reloaded. But this doesn't help in the case of node-valued static parameters.

#2 - 2018-11-19 19:03 - Michael Kay

It's not just nodes that cause problems.

We have code for exporting "external Java objects", but make no attempt at corresponding import code. It's basically impossible (except perhaps by Java serialization) so we should just disallow it. I've modified the code so this is an error at export time, rather than causing an error at import time.

For nodes, I feel the solution might be to build up a list of distinct trees that are referenced from the SEF, with identifiers and base URIs / document URIs, and output these all at the end (perhaps in CDATA to avoid namespace pollution); the value itself should then be output in the form <node tree="823" path="(path-to-node)" kind="(node kind)"/>. This would allow retention of relationships within a tree.

#3 - 2018-11-20 12:39 - Michael Kay

- Status changed from New to In Progress

- Applies to branch 9.9 added

- Fix Committed on Branch 9.9 added

I have fixed the side-issue of export of external objects (by disallowing it at export time) -- not a trivial change because export and explain were sharing the same code and we want different behaviour in the two cases. Patch committed to class Literal.

#4 - 2019-01-29 11:28 - Michael Kay

- Has duplicate Bug #4117: java.lang.AssertionError : Target of component reference variable keywords is undefined (caused by inlining function calls across a package boundary) added

#5 - 2019-01-29 11:45 - Michael Kay

This has come up in a user-submitted bug report, bug #4117.

In that case the scenario is fairly simple: a single global variable whose value is a document node. We can handle that case by generating code to reconstruct the document, for example by applying parse-xml() to a serialization.

The complication arises when we have global variable referring to multiple nodes within such a document. Do we attempt to generate one variable containing the document as a whole, and further variables referring to selected nodes within it?

I think an easier approach is probably to export the variables as if they were not static. The static processing has already been done before the export happens; it should be possible to simply re-evaluate the variables at execution time, as if they were not static at all. As already stated, this doesn't work for static parameters. But perhaps we impose a restriction for that case.

#6 - 2019-02-21 11:20 - Michael Kay

I decided to look more closely at how we export stylesheets containing static parameters. For example test static-002 has source

```
<xsl:param name="static-param" static="yes" select="'success'"/>
<xsl:param name="alt-param" static="yes" select="upper-case($static-param)"/>
```

and compiles to:

```
<co id='0' binds=''>
  <globalParam name='Q{}static-param' type='item()*' line='6' module='file:/Users/...../attr/static/static-002
.xsl' visibility='PRIVATE'>
   <str val='success'/>
  </globalParam>
 </co>
 <co id='1' binds=''>
  <globalParam name='Q{}alt-param' type='item()*' line='8' module='file:/Users/...../attr/static/static-002.xs
l' visibility='PRIVATE'>
   <str val='SUCCESS'/>
  </globalParam>
 </co>
```

So (a) there seems to be no indication that the parameter was originally declared static, which means a value can be supplied at run-time, (b) however, the default value for $alt-param will not be affected by any run-time value supplied for $static-param. I think it would make more sense if the compile-time value of a static param were "frozen" to prevent a different value being supplied at run time. This is most easily achieved by exporting it as a variable rather than a param.

(This doesn't solve the original problem, but it removes one of the complications.)

I have made this change (9.9 and trunk) and have also added flags="s" for static variables/parameters, essentially for future use if needed.

#7 - 2019-02-21 13:13 - Michael Kay

- Has duplicate deleted (Bug #4117: java.lang.AssertionError : Target of component reference variable keywords is undefined (caused by inlining function calls across a package boundary))

#8 - 2019-08-15 17:23 - Michael Kay

- Status changed from In Progress to Resolved

- Applies to branch trunk added

- Fix Committed on Branch trunk added

I have resolved this as follows. We can now export all 7 kinds of node to the SEF file, and reconstitute the nodes when the SEF file is loaded. The reconstitution is imperfect, in that inter-node relationships are lost: the reconstituted nodes will all be parentless, as if they were produced using copy-of() on the originals. This restriction is documented, and is likely to be permanent.

#9 - 2019-09-05 16:42 - O'Neil Delpratt

- Status changed from Resolved to Closed

- % Done changed from 0 to 100

- Fixed in Maintenance Release 9.9.1.5 added

Bug fix applied in the Saxon 9.9.1.5 maintenance release.

#10 - 2021-01-14 17:07 - Michael Kay

- Related to Bug #4829: Static variable initialised to node values cause run-time failure added