Saxon - Bug #4397

Obscure issue when using "instance of" inside function on results of array:filter

2019-11-27 18:03 - Priscilla Walmsley

Status:	Closed	**Start date:**	2019-11-27
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fix Committed on Branch:**	9.9, trunk
Applies to branch:	9.8, 9.9, trunk	**Fixed in Maintenance Release:**	9.9.1.6

Description

The attached XSLT results run in oXygen 21 results in the following error:

Engine name: Saxon-PE 9.8.0.12 Severity: error Description: Internal error evaluating template rule at line 8 in module file:/C:/aaprojects/datypic/functx/functx/saxonissues.xsl

It seems like a very obscure issue because other calls to array:filter are fine, and "instance of" on the results works fine if it is not inside a function. Something to do with function conversion rules? But why this array and not others?

(This was originally part of a larger XSLT that tests my XQuery book examples in the context of FunctX, but I have distilled it down.) I can't test this on 9.9 because it requires PE and I don't have a license. Thanks!

History

#1 - 2019-11-28 10:55 - Michael Kay

No failure using either Saxon-EE 9.8.0.15 or Saxon-EE 9.9.1.5 (Or to be more accurate, using my current development versions, which may include some bug fixes not in the released versions).

Confirmed: no failure using the issued Saxon-EE 9.8.0.15 or 9.9.1.5.

However, the problem does occur using the issued Saxon-PE 9.8.0.15 or 9.9.1.5.

So this is a rare case of something that fails under Saxon-PE but works under EE. That probably means I will have to create (or reconstitute) a development/debugging environment specific to PE.

#2 - 2019-11-28 11:02 - Michael Kay

- Status changed from New to In Progress

Note that the simpler but equivalent expression using

```
starts-with(?, 'a')
```

in place of starts-with#2(?, 'a') seems to work just fine.

I wonder if we should rewrite a dynamic call on a statically-known function with a static call on the same function? (Or perhaps, that's what we're attempting to do when it fails....)

#3 - 2019-11-28 11:22 - Michael Kay

Looking at it by code-reading, without the debugger, we can see that it's failing to bind a call to a user-defined function because there's no current component. In fact, all the signs are that it's failing to resolve the call on local:is-array(). It's not easy to see how that should be affected by the dynamic/partially-applied call on fn:starts-with(). Perhaps a binding slot number is being reused or incorrectly allocated, which would cause completely unpredictable failures.

#4 - 2019-11-28 12:17 - Michael Kay

I can trigger the failure in my normal development environment by running with -opt:0, which suppresses optimizations.

As suspected, the call on local:is-array() is failing because there is no current component. (It doesn't fail with optimization enabled because the function call gets inlined; in fact, the function and the "instance of" expression are never evaluated because it's possible to infer by static type analysis alone that the result must be true).

It seems that the call on starts-with#2 (via various wrapping layers of CurriedFunction and FunctionCoercer instances) is causing the current component in the context to be wrongly set to null, which then causes the completely straightforward call on local:is-array() to fail. The conditions for setting the current component to null should be the same as the conditions that cause a new context to be allocated, but with this particular set of circumstances, we are trying to avoid creating a new context, but are then corrupting the old one.

The dynamic function starts-with#2 is represented as an instance of UserFunctionReference.BoundUserFunction; the comments on this class (and its name) wrongly imply that it is used only for references to user-defined functions, and not for references to system-defined functions. This appears to be the source of the confusion. For a call to a system-defined function there is no need to create a new context; but in that case the current component should not be reset.

#5 - 2019-11-28 12:29 - Michael Kay

- Status changed from In Progress to Resolved

- Applies to branch 9.9, trunk added

- Fix Committed on Branch 9.9, trunk added

I think the simplest fix is to change UserFunctionReference.BoundUserFunction.call() so that it doesn't set the current component in the new context if it is null. It's a bit difficult to convince myself that that changes covers all cases, but it certainly covers this one.

#6 - 2019-12-06 14:48 - O'Neil Delpratt

- Status changed from Resolved to Closed

- Fixed in Maintenance Release 9.9.1.6 added

Patch committed to the Saxon 9.9.1.6 maintenance release.

Files

saxonissues.xsl	724 Bytes	2019-11-27	Priscilla Walmsley
dummyin.xml	66 Bytes	2019-11-27	Priscilla Walmsley