Saxon - Bug #4382

NullPointerException when running collection() with recurse=yes over a directory tree containing a large number of files

2019-11-14 20:45 - David Cramer

Status:	Closed	**Start date:**	2019-11-14
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	100%
Category:	Internals	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fixed in Maintenance Release:**	9.9.1.7
Applies to branch:	9.9, trunk	**Platforms:**	
Fix Committed on Branch:	9.9, trunk		

Description

See readme in attached zip file for details and a set of files that should reproduce the error as well as details about the environment.

```
dcramer@anatine ~/Desktop/SaxonCollectionOpenFilesBug (master %)
$ java -jar ~/Downloads/SaxonEE9-9-1-5J/saxon9ee.jar -repeat:10 -xsl:copy-images.xsl -o:out.xml do
cument/document.xml
 Error code: saxon:XXXX9999
                              Reason: java.lang.NullPointerException(null)

Error in xsl:copy-of/@select on line 54 column 63 of copy-images.xsl:
  SXRD0004: Unable to write to output destination:
  /Users/dcramer/Desktop/SaxonCollectionOpenFilesBug/zipmanifest.xml (Too many open files)
  In template rule with match="/" on line 50 of copy-images.xsl
Unable to write to output destination
```

History

#1 - 2019-11-21 14:01 - Michael Kay

- Category set to Internals

- Status changed from New to In Progress

Problem reproduced.

The problem still occurs with --allow-multithreading:off but the diagnostics are much more straightforward:

```
Error in xsl:result-document/@href on line 53 column 56 of copy-images.xsl:
  SXRD0004: Unable to write to output destination:
  /Users/mike/bugs/2019/4382-cramer/bug/zipmanifest.xml (Too many open files)
  In template rule with match="/" on line 50 of copy-images.xsl
Unable to write to output destination
```

#2 - 2019-11-21 15:30 - Michael Kay

The reason that too many input files are open is that we get an input stream for each resource early on, in order to "peek" at the content to help determine the content type.

In addition, the policy for closing the input streams seems haphazard at best. For example, in the case of images, when we have decided to allocate a BinaryResource, the BinaryResource initially contains the open InputStream.

It's hard to see what the intent of this particular stylesheet is. The function db:compute-collection() returns a sequence of maps; it looks as if it's designed to be a map from URIs to resources (and if collection() succeeded, then this function would fail because the returned items wouldn't be maps), but it only ever looks at the URIs (which means it could use the much simpler uri-collection() function). But let's assume the real stylesheet is doing something more interesting.

Certainly in this scenario where the resources in the collection are never examined, the input streams that were opened for determining the media type never get closed.

#3 - 2019-11-21 16:09 - Michael Kay

- Assignee set to Michael Kay

- Priority changed from Low to Normal

This one isn't going to be easy, and needs some thought.

(A) We could abandon the early "peeking" at the resource content used to classify the resource, and rely solely on the URI/filename extension; or we could suppress this at user option. We would then need to change the implementation of Resource.getItem() so it fetches the resource using the URI, rather than relying on a connection being open.

(B) We could abandon the early classification of resources by media type: instead returning some kind of UnidentifiedResource which remains unidentified until its content type or content is requested. This would be pretty disruptive, as users can register a ResourceFactory for particular content types, and this relies on identifying the content type before the Resource is created. But we might be able to combine this with (A), so we allocate an UnidentifiedResource only at user option.

(C) We could limit the number of InputStreams that we hold open during processing of a collection to (say) 50, or perhaps to some number linked to the number of threads. When an input stream is opened to peek at the content, it is put in a LRU pool and is closed when the limit is exceeded. If the resource is read and the input stream is no longer open, a new input stream is obtained.

(D) Finally, we need to ensure that all the Resource implementations close the input stream after getItem() is called to materialize the resource.

Doing (C) feels complex, but it also feels like the least disruptive option.

#4 - 2019-12-29 01:07 - Michael Kay

I've been taking another look at this. We clearly need to resolve this, and it needs a design change.

The current design is broken because it tries to get an InputStream eagerly, while creating a Resource, and to consume the InputStream lazily, while reading the Resource. This would only work if InputStreams were cheap resources to keep around, but this bug reminds us that they are not cheap; there are OS limits on how many can be held.

My current inclination is

(1) change the code in AbstractResourceCollection.guessContentType() so that (a) if the content type can be inferred from the name, it doesn't look at the actual content; and (b) if it does have to look at the content, it should immediately close the stream after doing so. But perhaps if it knows the resource is small then it should save the content at this point so it doesn't have to be re-read later.

(2) don't hold an InputStream as a field in InputDetails or in resources such as UnparsedTextResource; instead get a new stream when needed and close it after use.

#5 - 2020-01-02 00:22 - Michael Kay

Now fixed on the 9.9 branch; the fix needs to be applied also to the development branch.

The fix is a significant redesign of the collections framework, and may impact advanced applications that control how collections work (for example by introducing new implementations of collections). For example, the change required changes to the XSLT3 and QT3 test drivers.

The essence of the change is that an "unexpanded" resource in a collection never contains an InputStream or a JAXP Source object, because by holding such an object we have no way of controlling how many open streams exist, or of guaranteeing that they are eventually closed. In a few cases this means that we may need to open two URLConnections to the resource, one to determine its type and the other to expand it.

#6 - 2020-01-02 00:51 - Michael Kay

- Status changed from In Progress to Resolved

- Applies to branch 9.9, trunk added

- Fix Committed on Branch 9.9, trunk added

#7 - 2020-02-26 17:36 - Jorge Williams

Hello,

Has this changed been released? If not, will we be seeing this in a release soon?

Thanks

#8 - 2020-02-26 17:59 - Michael Kay

No, this hasn't yet been released. We're aware that a new 9.9 maintenance release is now becoming due.

#9 - 2020-03-05 11:09 - O'Neil Delpratt

- Status changed from Resolved to Closed

- % Done changed from 0 to 100

- Fixed in Maintenance Release 9.9.1.7 added

Patch applied in the 9.9.1.7 maintenance release.

Files

SaxonCollectionOpenFilesBug.zip	2.3 MB	2019-11-14	David Cramer