Saxon - Bug #5106

Gizmo fails with ConcurrentModificationException using "delete //prefix:local~

2021-09-25 00:22 - Michael Kay

Status:	Resolved	**Start date:**	2021-09-24
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	0%
Category:	Gizmo	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fixed in Maintenance Release:**	
Applies to branch:	10, 11, trunk	**Platforms:**	
Fix Committed on Branch:	10, trunk		

Description

For example

```
java -cp ~/bin/10.5/he/saxon-he-10.5.jar:/Users/mike/bin/10.5/he/jline-2.14.6.jar  net.sf.saxon.Gizmo
Saxon Gizmo 10.5
/>load /Users/mike/GitHub/saxon2020/src/samples/schemas/validation-reports.xsd
/>delete //xs:annotation
Exception in thread "main" java.util.ConcurrentModificationException
 at java.util.ArrayList$Itr.checkForComodification(ArrayList.java:901)
 at java.util.ArrayList$Itr.next(ArrayList.java:851)
 at net.sf.saxon.Gizmo.delete(Gizmo.java:439)
 at net.sf.saxon.Gizmo.executeCommands(Gizmo.java:304)
 at net.sf.saxon.Gizmo.<init>(Gizmo.java:231)
 at net.sf.saxon.Gizmo.main(Gizmo.java:168)
```

History

#1 - 2021-09-25 10:03 - Martin Honnen

https://docs.oracle.com/javase/8/docs/api/java/util/ConcurrentModificationException.html says "If a single thread issues a sequence of method invocations that violates the contract of an object, the object may throw this exception. For example, if a thread modifies a collection directly while it is iterating over the collection with a fail-fast iterator, the iterator will throw this exception.". I think the Iterator Saxon creates breaks the contract to implement and solely use the remove method when deleting items it iterates over.

#2 - 2021-09-25 10:28 - Michael Kay

Yes, something like that must be going on, but I'm having trouble seeing exactly where. We call the delete() method on the Item affected, but that doesn't actually delete the Java object or remove it from the List of items we are iterating over.

I've established that if we change the code to just use the SequenceIterator returned by getSelectedItems(buffer, Token.EOF) directly, then it works, but I feel there must be a reason why the code wasn't written to do just that, and that changing it would break something else. Unfortunately debugging and testing Gizmo isn't a very well-developed art, partly because there have been very few problems with it.

#3 - 2021-09-26 11:24 - Martin Honnen

For delete //child, the GroundedValue all has a property value with an ArrayList of ElementImpl that I think is iterated over with for (Item item : all.asIterable()). It is not changed during the iteration and not shared with other objects.

Meanwhile, for namespace ex http://example.com and delete //ex:child that same ArrayList is shared as the _value property of the elementList property of the DocumentImpl that is the root of all elements and that way in deIndex the element nodes are removed from that ArrayList in line 509 list.remove(node); which then causes the iterator to fail as it doesn't expect any deletion from the ArrayList.

#4 - 2021-09-27 08:41 - Martin Honnen

ListIterator in public GroundedValue materialize() does return SequenceExtent.makeSequenceExtent(list); but the description of SequenceExtent.makeSequenceExtent says "@param input a List containing the items in the sequence. The caller guarantees that this list will not be subsequently modified.". So perhaps ListIterator needs to call makeSequence with a shallow copy return SequenceExtent.makeSequenceExtent(new ArrayList<>(list)); instead?

#5 - 2021-10-19 12:43 - Michael Kay

I have now created a unit testing framework which makes it much easier to write and debug repeatable tests for individual Gizmo commands.

The relevant test (testDelete002) is not failing for me in quite the same way, but it is failing, and the cause is probably the same: after a delete operation, the index of elements by name is not properly updated.

#6 - 2021-10-19 15:31 - Michael Kay

Test rename003 is also failing; the elements are renamed, but the index by element name is not updated, so empty(//X) on the old name returns false. I suspect this affects rename in XQuery Update as well: the method ElementImpl.rename() appears to make no attempt to update the document-level index.

rename() should not only remove the element from the index for the old name, it should also add it to the index for the new name.

In fact, rather than making incremental modifications to the index, it would probably be better to bulk-delete the index information and allow it to be reconstructed when next needed.

#7 - 2021-10-19 15:54 - Michael Kay

In fact XQuery update does a bulk reset of the indexes on all affected documents after applying the pending update list. (DocumentImpl.resetIndexes()). This operation is cheap, so I think Gizmo should do it (on the current document) after any updating operation.

The only question is when. If it's done right at the start, then evaluate the select expression (e.g rename //my:elements as ...) could rebuild the indexes before the renaming.

Related to this, Gizmo (unlike XQuery Update) doesn't attempt to ensure complete evaluation of the select expression before any changes are made. It's possible that renaming one selected node could affect the value of a predicate evaluated within the select expression on a subsequent node. I think it's safest if all updating commands now do the same as we're doing with delete - first build a list of selected nodes, then apply the changes.

We can then invalidate the indexes as soon as this list of nodes has been built.

#8 - 2021-10-19 17:54 - Michael Kay

- Status changed from New to Resolved

- Applies to branch 10, 11, trunk added

- Fix Committed on Branch 10, trunk added

All the new unit tests are now working.