Saxon - Bug #4683

chained descendant axes gathering extra nodes

2020-08-13 19:30 - Erica Coan

Status:	Closed	**Start date:**	2020-08-13
Priority:	Normal	**Due date:**	
Assignee:	O'Neil Delpratt	**% Done:**	100%
Category:	DOM Interface	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fix Committed on Branch:**	10, 9.9
Applies to branch:	10, 9.9	**Fixed in Maintenance Release:**	10.2, 9.9.1.8

Description

I am getting unexpected results from the attached(greatly simplified) transform when run via Saxon HE 9.9.1.5 for .NET. I would expect the 3 messages to have values of 0, 1, and 0. Instead, I get 3, 3, and 0. I've played around with different versions of the select, and it seems to me that the descendant axis is picking up ALL descendants, not just the descendants of the set of p nodes gathered up to that point.

select = "count(ancestor::p/preceding-sibling::p/descendant::sectPr)"

Goal: Find preceding p siblings of ancestor p nodes. Then gather sectPr descandants of those p nodes.

History

#1 - 2020-08-14 14:31 - Martin Honnen

Are you sure the two files demonstrate the issue?

I wrote

```
            string xmlUri = "https://saxonica.plan.io/attachments/download/49089/input.xml";
            string xsltUri = "https://saxonica.plan.io/attachments/download/49088/transform.xsl";

            Processor processor = new Processor();

            Console.WriteLine("{0}: {1}", processor.ProductTitle, processor.ProductVersion);

            XsltCompiler compiler = processor.NewXsltCompiler();

            Xslt30Transformer xslt30Transformer = compiler.Compile(new Uri(xsltUri)).Load30();

            DocumentBuilder docBuilder = processor.NewDocumentBuilder();

            var input = docBuilder.Build(new Uri(xmlUri));

            xslt30Transformer.GlobalContextItem = input;

            xslt30Transformer.ApplyTemplates(input, processor.NewSerializer(Console.Out));
```

with .NET and using the version of Saxon you use and get

```
SAXON-HE 9.9.1.5N from Saxonica: 9.9.1.5
0
1
0
```

#2 - 2020-08-14 18:32 - Erica Coan

I confirmed that the files do have the issue for me. However, by you including your code, I was able to see where the difference is between your results and mine. I ran your code as-is, and also got 0 1 0 for results. Then I adjusted it to look more like my code, until the results switched to 3 3 0. The key difference turned out to be the way that I'm supplying the input:

Dim myInputDoc As New Xml.XmlDocument Dim myFileStream As FileStream = New FileStream(xmlUri,FileMode.Open,FileAccess.Read, FileShare.ReadWrite) Dim mByte() As Byte ReDim mByte(CInt(myFileStream.Length) - 1) myFileStream.Read(mByte, 0, CInt(myFileStream.Length)) myFileStream.Close() Dim myMemStream As MemoryStream = New MemoryStream(mByte) myInputDoc.Load(myMemStream)

xslt30Transformer.ApplyTemplates(docBuilder.Wrap(myInputDoc), processor.NewSerializer(Console.Out))

When I load the input this way, I get 3 3 0 for results.

#3 - 2020-08-14 18:45 - Martin Honnen

That might be a bug then in https://dev.saxonica.com/repos/archive/opensource/latest9.9/hej/net/sf/saxon/dotnet/DotNetNodeWrapper.java or at least in the code behind the Wrap method.

#4 - 2020-08-15 00:56 - Martin Honnen

It seems that, with a wrapped XmlDocument, for the empty p elements, the XPath descendant::sectPr finds all descendant sectPr elements of the following sibling p elements instead of (simply?) returning an empty sequence.

#5 - 2020-08-15 01:29 - Martin Honnen

A more simple test case is

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <item/>
  <item>
    <foo id="foo1"/>
  </item>
  <item></item>
  <item>
    <foo id="foo2"/>
  </item>
</root>
```

running

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:transform version="3.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="#all">

  <xsl:param name="DEBUG" static="yes" as="xs:boolean" select="true()"/>

  <xsl:output method="text" item-separator="&#10;"/>

  <xsl:template match="/">
    <xsl:for-each select="//item">
      <xsl:assert test="not(*) eq not(descendant::foo)" use-when="$DEBUG">No child elements but descendants
</xsl:assert>
      <xsl:value-of select="position(), not(*), descendant::foo/@id"/>
    </xsl:for-each>
  </xsl:template>

</xsl:transform>
```

gives

```
1 true
2 false foo1
3 true
4 false foo2
```

for Saxon's Xdm but

```
1 true foo1 foo2
2 false foo1
3 true foo2
4 false foo2
```

for a wrapped .NET XmlDocument.

#6 - 2020-08-17 00:55 - Michael Kay

- Category set to DOM Interface

- Status changed from New to In Progress

- Assignee set to O'Neil Delpratt

Thanks Martin for your help in diagnosing this. It does indeed look like a problem with the DOM wrapping code on .NET.

Erica, we will of course investigate and fix this, but you should be aware that running Saxon on a DOM tree is much slower (sometimes 5 - 10 times slower) than letting Saxon build the tree itself using its native TinyTree format.

I suspect that we aren't running a sufficient set of tests using the DOM interface on .NET. On the Java side running the full QT3 test suite against third party tree models is part of our standard prerelease checklist, but I'm not sure this is the case on .NET.

#7 - 2020-08-17 01:22 - Michael Kay

I wrote a unit test for the example in comment #5 which shows that we are getting it right with the Java DOM, which uses almost identical code.

#8 - 2020-08-17 09:31 - Michael Kay

For the Java DOM, the method DOMNodeWrapper.getSuccessorNode() has the following test:

```
        if (anchor != null && start.isSameNode(anchor)) {
            return null;
        }
```

which is missing from the corresponding method for the .NET DOM, DotNetNodeWrapper.getSuccessorNode(). I suspect these lines need to be added - but we need to test this on .NET. The test ensures that if we are positioned at the "anchor" node (the one whose descendants we are seeking), we return "end of sequence" rather than moving on to the next following sibling,

#9 - 2020-08-17 09:46 - Michael Kay

These lines were added to DOMNodeWrapper on 2013-04-07 (revision 2272): no bug number was recorded. It appears this revision fixed bugs in all the third-party tree models using the SteppingNode framework (which was introduced in 2012-09) with the exception of the .NET DOM code. So it looks like the bug has gone undetected for quite a while.

#10 - 2020-08-17 13:21 - O'Neil Delpratt

- Applies to branch 10 added

Hi,

I have taken the repo from comment #5 and reproduced the bug issue on C# when using the XmlDocument (C# DOM node).

Another difference I have noticed between DOM on Java and C# is that the Java DOM Node supports DOM level 3, whereas C# DOM only support is DOM level 2. The method isSameNode is available since DOM level 3. This is unfortunate because on C# this method is not available on the class XmlNode.

Currently looking at workarounds.

#11 - 2020-08-17 13:45 - Michael Kay

It's probably safe with the C# DOM to compare DOM nodes for identity using the "==" operator, that is to assume that node identity implies object identity. This isn't a safe assumption with the Java DOM, since nodes can be instantiated lazily.

#12 - 2020-08-17 13:51 - Martin Honnen

Or use https://docs.microsoft.com/en-us/dotnet/api/system.object.referenceequals?view=netframework-4.6 as that always compares object identity while == might have been overridden (although it doesn't seem to be the case in the XmlNode/XmlDocument classes).

#13 - 2020-08-17 14:23 - O'Neil Delpratt

Yes I had thought I could use == operator. The DOM spec description is clear on what to expect from the isSameNode method.

Also looking at the Xerces implementation on Java it see to have the same implementation:

```
    public boolean isSameNode(Node other) {
        // we do not use any wrapper so the answer is obvious
        return this == other;
    }
```

#14 - 2020-08-17 14:47 - O'Neil Delpratt

- Status changed from In Progress to Resolved

- Fix Committed on Branch 10, 9.9 added

The bug fix identity using the "==" operator applied in the getSuccessorNode method of DotNetNodeWrapper class. Bug fix available in the next release of Saxon on .NET. This will be released on Saxon 10 before it is released on the 9.9 branch.

#15 - 2020-08-17 14:50 - O'Neil Delpratt

Added a nunit test case for testing of this bug for regression purposes. Looking at the W3C test suites for better testing when have a C# DOM node

as the tree model within Saxon.

#16 - 2020-08-17 15:10 - Michael Kay

Added QT3 test case prod-AxisStep/K2-Axes-106

#17 - 2020-08-26 10:23 - O'Neil Delpratt

- % Done changed from 0 to 100

- Fixed in Maintenance Release 10.2 added

Bug fix applied in the Saxon 10.2 maintenance release.

#18 - 2020-10-22 18:17 - O'Neil Delpratt

- Status changed from Resolved to Closed

- Fixed in Maintenance Release 9.9.1.8 added

Bug fix applied on the Saxon 9.9.1.8 maintenance release.

Files

transform.xsl	843 Bytes	2020-08-13	Erica Coan
input.xml	2.06 KB	2020-08-13	Erica Coan