Saxon - Bug #4212

ClassNotFoundException: com.saxonica.ee.bytecode.GeneratedCode

2019-05-06 12:59 - Pranay Deshpande

Status:	Rejected	**Start date:**	2019-05-06
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	0%
Category:	Build and release	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fix Committed on Branch:**	
Applies to branch:		**Fixed in Maintenance Release:**	

Description

Dear Colleagues,

This is Pranay from SAP Labs. We are using SaxonEE library for XSLT processing in our product. Where we are getting ClassNotFoundException for ‘com.saxonica.ee.bytecode.GeneratedCode’

org.apache.camel.CamelExecutionException: Exception occurred during execution on the exchange: Exchange[ID-vsa6585629-1556884301644-4-1], cause: java.lang.ClassNotFoundException: com.saxonica.ee.bytecode.GeneratedCode

We started getting this issue in SaxonEE-9.8 version once we started using EnterpriseTransformerFactory, but as recommended by Michael in below thread <https://saxonica.plan.io/issues/3814 > this issue is fixed in 9.9.1 version. So we also tried executing the same transformation using SaxonEE-9.9.1, but still facing the same issue. I’m currently trying to put the workaround in place i.e. configuration.getDynamicLoader().setClassLoader(Configuration.class.getClassLoader()); - as recommended in the same thread 3814

PS: We are getting this issue only if the XML payload is More than 4000 lines approx. For smaller payload, the transformation is working fine.

Meanwhile could you please provide us some insight into this problem. Is it a known issue, can we expect a fix for this in the upcoming releases, or is there any other way to avoid this.

In our stack, we have Apache Camel in our runtime, SaxonEE as XSLT processor with OSGI (Karaf and Equinox containers)

Thanks in advance Best Regards, Pranay Deshpande SAP Labs

History

#1 - 2019-05-06 21:06 - Michael Kay

This is essentially a duplicate of bug #3814; comment #2 describes a workaround, and at the moment that's the best we can do.

The reason the failure occurs above a certain document size is that we only generate bytecode for code that has been executed a given number of times. Another workaround is to switch bytecode generation off completely (-opt:-c on the command line, or equivalents in the XsltCompiler class in s9api). Bytecode generation can be important for the performance of some workloads, but for others it makes very little difference.

#2 - 2019-05-07 12:45 - Pranay Deshpande

Hi Michael,

Thanks for providing the quick info on this. Could you please confirm what's the exact value of this "document size" above which, the bytecode is generated (I believe thats for optimizing the processing of XSLT) Based on the size, we could be able to provide some workarounds for our customers.

Meanwhile, I've tried this: enterpriseConfiguration.setConfigurationProperty(Feature.GENERATE_BYTE_CODE,false); I hope this should disable the byte code genaration.

Correct me if I'm wrong here.

Regards Pranay

#3 - 2019-05-07 13:35 - Michael Kay

The trigger is only indirectly related to document size - it's based on the number of times particular expressions are executed, which will tend to increase as the document size increases, but isn't a straightforward relationship.

Switching off the bytecode generation seems a more promising approach; but there will probably be other cases where Saxon is doing dynamic loading too (as evidenced by your other bug report) so you should probably try and fix it by setting the ClassLoader for Saxon to use.

#4 - 2019-05-09 08:27 - Pranay Deshpande

Hi Michael,

Switching off the bytecode generation seems to solve the current problem. But in general, do you have any timeline to fix the class loader problems with OSGI environment?

You can reduce down the priority of this ticket though as the immediate blocker is resolved.

-Pranay

#5 - 2019-05-10 11:18 - Michael Kay

I would imagine that using the Configuration.class.getClassLoader() workaround is going to work for Saxon's dynamic loading of its own classes, but it may not work for loading classes in a different OSGi bundle from Saxon (such as reflexive extension functions), unless the OSGi bundle that contains Saxon registers a dependency on the packages containing those extension functions.

I should stress that we have done no work on OSGi and it is technically unsupported. But I would like to help you, if only because it will help us learn about what we need to do to make Saxon work in an OSGi environment.

The first thing I need to understand is how Saxon has been bundled in this environment: what does the bundle descriptor for the bundle containing Saxon look like, and are there other things in the same bundle? What does the bundle descriptor for the calling application look like, and which Saxon packages does it declare dependencies on?

I suspect the answer to these questions is that this has all been done by Apache Camel, rather than yourselves. That means we need to start understanding what Apache Camel has done to OSGi-enable Saxon. This is a slippery slope from a support point of view: we normally make a point of not getting involved in the detail of how third parties have integrated Saxon into their environments; support is their problem not ours. At the same time, understanding what they have done may help us to understand what is possible, what its limitations are, and what changes we need to make to make it work better.

Reading around, dynamic class loading under OSGi looks like a real challenge, and there seems to be a lot of advice to avoid it wherever possible. This suggests that if and when we do support OSGi, we will have to move away from the highly-dynamic loading of reflexive extension functions (where we deduce the Java class name from the namespace URI of the function) to a model where extension function implementations are explicitly registered. This suggests that we might well have to move to something closer to "integrated extension functions". Looking more closely at your bug #4213 it seems you are already doing that, so perhaps I have misinterpreted the cause and need to investigate that separately.

Getting byte code generation to work under OSGi might be simply a case of doing further tweaking of the class loader that we use for this purpose. Other cases where we do dynamic loading might require redesign of APIs so the required classes are supplied to us as Class objects rather than as class names to be loaded.

#6 - 2019-05-10 12:50 - Juan Lopez

- File AMEX.xml added

- File XMLtoCSV.xslt added

- File XMLtoCSV_CONF.xslt added

- File xslt error.bmp added

Hi

Currently I am working for a customer with SAP cloud platform intregation tool.

XSLT processor has next properties:

<?xml version="1.0" encoding="UTF-8"?> 3.0 Saxonica http://www.saxonica.com/ SAXON EE 9.8.0.12 no yes yes

Attachment example is failing in Saxonica XSLT processor but not with other tools like Altova Xmlspy 2011 in which was developed. Example is returning same error commented by SAP labs.

Doing some experiments XSLT failed with release 1.2 (SCPI) but not with release 1.0.

I can imagine that error will be solved in next months.I would like to modify my current transformation to provide same features and use it. Which points i should change for avoiding current issue?

Kr

#7 - 2019-05-10 13:27 - Michael Kay

Juan, yes, you seem to be hitting the same problem with generated byte code under Apache Camel, and the same circumventions should work: either suppress bytecode generation, or change the ClassLoader registered with the configuration so that it works under OSGi.

#8 - 2019-09-05 12:13 - Michael Kay

- Category set to Build and release

- Status changed from New to Rejected

- Assignee set to Michael Kay

- Priority changed from High to Normal

I am closing this for the time being without a resolution: OSGi is currently not supported by Saxon, therefore this is a feature request, not a bug (and it appears on our list of desirable new features for future releases).

Files

saxonlogs.txt	3.75 KB	2019-05-06	Pranay Deshpande
AMEX.xml	644 Bytes	2019-05-10	Juan Lopez
XMLtoCSV.xslt	11.1 KB	2019-05-10	Juan Lopez
XMLtoCSV_CONF.xslt	3.69 KB	2019-05-10	Juan Lopez
xslt error.bmp	522 KB	2019-05-10	Juan Lopez