Saxon - Bug #4567

Failure trying to return a JDOM2 document from a reflexive extension function

2020-06-02 14:03 - Michael Kay

Status:	Closed	**Start date:**	2020-06-02
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	100%
Category:	Saxon extensions	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fixed in Maintenance Release:**	10.3
Applies to branch:	10, trunk	**Platforms:**	
Fix Committed on Branch:	10, trunk		

Description

The following test case:

```
public static JDOM2DocumentWrapper parseToJdom(XPathContext context, String content) throws Except
ion {

        SAXBuilder builder = new SAXBuilder();
        Document document = builder.build(new StringReader(content));
        return new JDOM2DocumentWrapper(document, context.getConfiguration());
    }

    public void testJDOM2extension() {
        try {
            Processor proc = new Processor(true);
            XQueryEvaluator eval = proc.newXQueryCompiler().compile(
                    "Q{java:jaxptest.ExtensionTest}parseToJdom('<magic/>')//*/name()").load();
            XdmValue result = eval.evaluate();
            assertEquals("magic", ((XdmAtomicValue) result).getStringValue());
        } catch (SaxonApiException e) {
            e.printStackTrace();
            fail();
        }
    }
```

fails with the exception:

```
Error on line 1 column 43
   Unknown source class net.sf.saxon.option.jdom2.JDOM2DocumentWrapper
net.sf.saxon.s9api.SaxonApiException: Unknown source class net.sf.saxon.option.jdom2.JDOM2Document
Wrapper
 at net.sf.saxon.s9api.XQueryEvaluator.evaluate(XQueryEvaluator.java:434)
 at jaxptest.ExtensionTest.testJDOM2extension(ExtensionTest.java:392)
...
Caused by: net.sf.saxon.trans.XPathException: Unknown source class net.sf.saxon.option.jdom2.JDOM2
DocumentWrapper
 at net.sf.saxon.Configuration.buildDocumentTree(Configuration.java:4171)
 at net.sf.saxon.expr.JPConverter$FromSource.convert(JPConverter.java:674)
 at net.sf.saxon.expr.JPConverter$FromSource.convert(JPConverter.java:664)
 at com.saxonica.expr.JavaExtensionFunctionCall.asSequence(JavaExtensionFunctionCall.java:665)
 at com.saxonica.expr.JavaExtensionFunctionCall.call(JavaExtensionFunctionCall.java:576)
 at com.saxonica.expr.JavaExtensionFunctionCall.iterate(JavaExtensionFunctionCall.java:445)
```

History

#1 - 2020-06-02 14:05 - Michael Kay

I also tried returning the JDOM2 Document object directly, hoping the implicit J-to-X conversion would handle it:

```
public static Document parseToJdom(XPathContext context, String content) throws Exception {
        SAXBuilder builder = new SAXBuilder();
        return builder.build(new StringReader(content));
}
```

This fails at compile time saying

```
Type error on line 1 column 1
  XPTY0020  Axis step descendant-or-self::node() cannot be used here: the context item is not a node
```

#2 - 2020-06-02 14:57 - Michael Kay

At one time JDOM2DocumentWrapper implemented NodeInfo and would therefore be handled directly as a node, without conversion. With the split of NodeInfo/TreeInfo this is no longer the case. The logic for allocating a converter is treating it as a generic Source, so the converter JPConverter.FromSource is allocated, but this does not recognise this class. Adding this

```
             if (object instanceof TreeInfo) {
                return ((TreeInfo)object).getRootNode();
            }
```

to the logic of FromSource fixes the problem.

#3 - 2020-06-02 15:00 - Michael Kay

As for returning a JDOM2 Document from the extension function directly, the problem seems to be that the run-time code is capable of doing the conversion (by wrapping the JDOM2 Document in a suitable wrapper), but the compile-time type-checking is not aware of this possibility. (Note that AxisExpression has it own custom type-checking code, it doesn't use the generic static type checker, which could possibly be smarter).

#4 - 2020-06-02 20:21 - Michael Kay

I tried another variation: changing the XPath expression to

```
base-uri(Q{java:jaxptest.ExtensionTest}parseToJdom('<magic/>'))
```

This one fails in a different way: a run-time error

```
java.lang.ClassCastException: net.sf.saxon.value.ObjectValue cannot be cast to net.sf.saxon.om.NodeInfo
 at net.sf.saxon.functions.BaseUri_1.call(BaseUri_1.java:25)
 at net.sf.saxon.functions.BaseUri_1.call(BaseUri_1.java:22)
```

The static type checking here has established that the required type (node()) subsumes the supplied type Q{http://saxon.sf.net/java-type}org.jdom2.Document and has therefore not inserted any converter into the expression tree.

I've tracked this down to the fact that the JDOM2 object model isn't registered with the Configuration. When I add the line to the test case

```
proc.getUnderlyingConfiguration().registerExternalObjectModel(new JDOM2ObjectModel());
```

it fails again, but this time legitimately, because the test is asking for the base URI of a node that doesn't have one.

Looking at the ClassCastException, the static typechecking is letting this through because of a bug in TypeHierarchy.computeRelationship(). At line 449 the code

```
if (t2.isPlainType() || t2 instanceof FunctionItemType) ...
```

isn't allowing for the possibility that t2 is an external object type. If I fix this, we now get a type error instead of the ClassCastException. This raises the question of whether we could do better: could the "function conversion rules" invoke the wrapping of the JDOM2 Document node as a Saxon NodeInfo? The documentation at http://www.saxonica.com/documentation/index.html#!extensibility/functions/function-result doesn't say that this should work.

#5 - 2020-06-02 20:25 - Michael Kay

A documentation point: we say at http://www.saxonica.com/documentation/index.html#!sourcedocs/thirdparty

**The support code for Axiom, DOM4J, JDOM2, and XOM is integrated into the main JAR files for Saxon-PE and Saxon-EE, but (unlike the case of DOM) it is not activated unless the object model is registered with the Configuration. This is done automatically if the relevant classes are found on the classpath. If object models other than these are to be supported, the implementation must either be included it in the relevant section of the configuration file, or it must be nominated to the configuration using the method registerExternalObjectModel().* *

I don't think the sentence "This is done automatically if the relevant classes are found on the classpath. " is true any more (it probably was once) -- except that it seems to happen in the JAXP XPathFactory.

#6 - 2020-09-14 13:09 - Michael Kay

Discussed at team meeting. We decided it's probably correct to require external object models to be explicitly registered with the Configuration, and the documentation should be changed to say so.

#7 - 2020-09-15 13:47 - Michael Kay

- Category set to Saxon extensions

- Status changed from New to Resolved

- Priority changed from Low to Normal

- Applies to branch 10, trunk added

- Fix Committed on Branch 10, trunk added

The code changes had already been made. I have now added JUnit tests and made documentation changes.

#8 - 2020-10-28 18:57 - O'Neil Delpratt

Bug fix applied in the Saxon 10.3 maintenance release

#9 - 2020-10-28 19:13 - O'Neil Delpratt

- Status changed from Resolved to Closed

- % Done changed from 0 to 100

- Fixed in Maintenance Release 10.3 added