How to deal with LinkageErrors in Java?

Asked 16 years, 1 month ago Modified 5 years, 1 month ago

Viewed 127k times

66

Developing a heavily XML-based Java-application, I recently encountered an interesting problem on Ubuntu Linux.

My application, using the Java Plugin Framework, appears unable to convert a dom4j-created XML document to Batik's implementation of the SVG specification.

On the console, I learn that an error occurs:

```
Exception in thread "AWT-EventQueue-0"
java.lang.LinkageError: loader constraint
violation in interface itable initialization: when
resolving method
"org.apache.batik.dom.svg.SVGOMDocument.createAttrib
the class loader (instance of
org/java/plugin/standard/StandardPluginClassLoader)
of the current class,
org/apache/batik/dom/svg/SVGOMDocument, and the
class loader (instance of <bootloader>) for
interface org/w3c/dom/Document have different
Class objects for the type org/w3c/dom/Attr used
in the signature
    at
org.apache.batik.dom.svg.SVGDOMImplementation.create
    at
org.dom4j.io.DOMWriter.createDomDocument(DOMWriter.j
    at
org.dom4j.io.DOMWriter.write(DOMWriter.java:138)
```

I figure that the problem is caused by a conflict between the original classloader from the JVM and the classloader deployed by the plugin framework.

To my knowledge, it's not possible to specify a classloader for the framework to use. It might be possible to hack it, but I would prefer a less aggressive approach to solving this problem, since (for whatever reason) it only occurs on Linux systems.

Has one of you encountered such a problem and has any idea how to fix it or at least get to the core of the issue?

java linux classloader linkageerror

Share

Improve this question

Follow

edited Apr 3, 2016 at 3:10

Krease

16.2k ● 8 ● 56 ● 89

asked Oct 28, 2008 at 19:18

Urs Reupke

6,901 ● 3 ● 38 ● 49

7 Answers

Sorted by: Highest score (default)

78

LinkageError is what you'll get in a classic case where you have a class C loaded by more than one classloader and those classes are being used together in the same code (compared, cast, etc). It doesn't matter if it is the same Class name or even if it's loaded from the identical jar - a Class from one classloader is always treated as a different Class if loaded from another classloader.

The message (which has improved a lot over the years) says:

```
Exception in thread "AWT-EventQueue-0" java.lang.Linka
loader constraint violation in interface itable initia
when resolving method
"org.apache.batik.dom.svg.SVGOMDocument.createAttribut
the class loader (instance of
org/java/plugin/standard/StandardPluginClassLoader)
of the current class, org/apache/batik/dom/svg/SVGOMDo
and the class loader (instance of ) for interface org/
have different Class objects for the type org/w3c/dom/
signature
```

So, here the problem is in resolving the SVGOMDocument.createAttribute() method, which uses org.w3c.dom.Attr (part of the standard DOM library). But, the version of Attr loaded with Batik was loaded from a different classloader than the instance of Attr you're passing to the method.

You'll see that Batik's version seems to be loaded from the Java plugin. And yours is being loaded from " ", which is most likely one of the built-in JVM loaders (boot classpath, ESOM, or classpath).

The three prominent classloader models are:

- delegation (the default in the JDK - ask parent, then me)
- post-delegation (common in plugins, servlets, and places where you want isolation - ask me, then parent)
- sibling (common in dependency models like OSGi, Eclipse, etc)

I don't know what delegation strategy the JPF classloader uses, but the key is that you want one version of the dom library to be loaded and everyone to source that class from the same location. That may mean removing it from the classpath and loading as a plugin, or preventing Batik from loading it, or something else.

Share Improve this answer Follow

answered Oct 28, 2008 at 22:00

Alex Miller
70.1k ● 25 ● 124 ● 168

What other classloader models are there? – Pacerier Aug 28, 2014 at 0:09

Sounds like a classloader hierarchy problem. I can't tell what type of environment your application is deployed in, 23

but sometimes this problem can occur in a web environment - where the application server creates a hierarchy of classloaders, resembling something like:

javahome/lib - as root
appserver/lib - as child of root
webapp/WEB-INF/lib - as child of child of root
etc

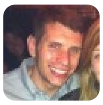
Usually classloaders delegate loading to their parent classloader (this is known as "`parent-first`"), and if that classloader cannot find the class, then the child classloader attempts to. For example, if a class deployed as a JAR in webapp/WEB-INF/lib tries to load a class, first it asks the classloader corresponding to appserver/lib to load the class (which in turn asks the classloader corresponding to javahome/lib to load the class), and if this lookup fails, then WEB-INF/lib is searched for a match to this class.

In a web environment, you can run into problems with this hierarchy. For example, one mistake/problem I've run into before was when a class in WEB-INF/lib depended on a class deployed in appserver/lib, which in turn depended on a class deployed in WEB-INF/lib. This caused failures because while classloaders are able to delegate to the parent classloader, they cannot delegate back down the tree. So, the WEB-INF/lib classloader would ask appserver/lib classloader for a class, appserver/lib classloader would load that class and try to load the dependent class, and fail since it could not find that class in appserver/lib or javahome/lib.

So, while you may not be deploying your app in a web/app server environment, my too-long explanation might apply to you if your environment has a hierarchy of classloaders set up. Does it? Is JPF doing some sort of classloader magic to be able to implement it's plugin features?

Share Improve this answer Follow

answered Oct 28, 2008 at 20:22



matt b
140k ● 66 ● 284 ● 350

Regarding the example of the problem you're describing, is this a realistic scenario? Why would a class in `appserver/lib` depend on anything in `webapp/WEB-INF/lib` ? – Johnny Baloney Nov 28, 2017 at 15:41

@JohnnyBaloney, here's the scenario for example: the class in `appserver/lib` may use implementation of interface which resides in `webapp/WEB-INF/lib` using some dynamic linking, e.g. a routine in `ParentClass` is called from `WebAppClass` , next `ParentClass` uses `SomeServiceImpl` (`WEB-INF/lib`) class as an implementation of `SomeService` (interface from `appserver/lib`). – Bryn Mar 12, 2020 at 10:25

▲

7

May be this will help someone because it works out pretty good for me. The issue can be solve by integrating your own dependencies. Follow this simple steps

First check the error which should be like this :

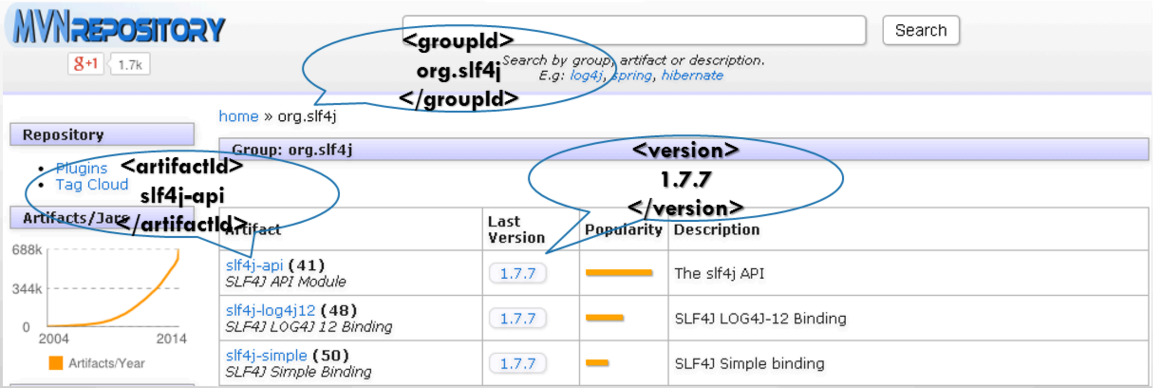
- Method execution failed:
- java.lang.LinkageError: loader constraint violation:
- when resolving method "org.slf4j.impl.**StaticLoggerBinder**.getLoggerFactory()Lorg/slf4j/ILoggerFactory;"
- the class loader (instance of org/openmrs/module/ModuleClassLoader) of the current class, org/slf4j/**LoggerFactory**,
- and the class loader (instance of org/apache/catalina/loader/WebappClassLoader) for resolved class, org/slf4j/impl/**StaticLoggerBinder**,
- have different Class objects for the type taticLoggerBinder.getLoggerFactory()Lorg/slf4j/ILoggerFactory; used in the signature

---

2. See the two highlighted class. Google search for them like "StaticLoggerBinder.class jar download" & "LoggeraFactory.class jar download". This will show you first or in some case second link (Site is http://www.java2s.com) which is one of the jar version you have included in your project. You can smartly identify it yourself, but we are addicted of google ;)
3. After that you will know the jar file name, in my case it is like slf4j-log4j12-1.5.6.jar & slf4j-api-1.5.8

4. Now the latest version of this file is available here http://mvnrepository.com/ (actually all version till date, this is the site from where maven get your dependencies).
5. Now add both file as a dependencies with the latest version (or keep both file version same, either chosen version is old). Following is the dependency you have to include in pom.xml

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.7</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.7</version>
</dependency>
```


MVNREPOSITORY
8+1 1.7k
Search
Search by group, artifact or description.
E.g: log4j, spring, hibernate
<groupId> org.slf4j </groupId>
<artifactId> slf4j-api </artifactId>
<version> 1.7.7 </version>
Repository
Plugins
Tag Cloud
Artifacts/Jars
688k
344k
0
2004
2014
Artifacts/Year

home » org.slf4j

Group: org.slf4j

Artifact	Last Version	Popularity	Description
slf4j-api (41) SLF4J API Module	1.7.7		The slf4j API
slf4j-log4j12 (48) SLF4J LOG4J 12 Binding	1.7.7		SLF4J LOG4J-12 Binding
slf4j-simple (50) SLF4J Simple Binding	1.7.7		SLF4J Simple binding

Share Improve this answer Follow

edited Jan 8, 2016 at 12:18

Buhake Sindi
89.1k ● 30 ● 174 ● 232

answered Jun 24, 2014 at 20:03

vineetv2821993
937 ● 12 ● 25

5

Can you specify a class loader? If not, try specifying the context class loader like so:

```
Thread thread = Thread.currentThread();
ClassLoader contextClassLoader = thread.getContextClas
try {
    thread.setContextClassLoader(yourClassLoader);
    callDom4j();
} finally {
    thread.setContextClassLoader(contextClassLoader);
}
```

I'm not familiar with the Java Plugin Framework, but I write code for Eclipse, and I run into similar issues from time to time. I don't guarantee it'll fix it, but it's probably worth a shot.

Share Improve this answer Follow

answered Oct 28, 2008 at 20:28

Adam Crume
15.8k ● 8 ● 48 ● 51

3

The answers from Alex and Matt are very helpful. I could benefit from their analysis too.

I had the same problem when using the Batik library in a Netbeans RCP framework, the Batik library being included as a "Library Wrapper Module". If some other module makes use of XML apis, and no dependency on Batik is needed and established for that module, the class loader constraint violation problem arises with similar error messages.

In Netbeans, individual modules use dedicated class loaders, and the dependence relationship between modules implies suitable class loader delegation routing.

I could resolve the problem by simply omitting the xml-apis jar file from the Batik library bundle.

Share Improve this answer

Follow

answered Nov 23, 2009 at 16:56

Anselm Schuster

31 ● 1

1

As specified in [this question](), enabling the `-verbose:class` will make the JVM log information about all classes being loaded, which can be incredibly helpful to understand where the classes are coming from in more complex scenarios & applications.

The output you get looks roughly like this (copied from that question):

```
[Opened /usr/java/j2sdk1.4.1/jre/lib/rt.jar]
[Opened /usr/java/j2sdk1.4.1/jre/lib/sunrsasign.jar]
[Opened /usr/java/j2sdk1.4.1/jre/lib/jsse.jar]
[Opened /usr/java/j2sdk1.4.1/jre/lib/jce.jar]
[Opened /usr/java/j2sdk1.4.1/jre/lib/charsets.jar]
[Loaded java.lang.Object from /usr/java/j2sdk1.4.1/jre
[Loaded java.io.Serializable from /usr/java/j2sdk1.4.1
[Loaded java.lang.Comparable from /usr/java/j2sdk1.4.1
```

```
[Loaded java.lang.CharSequence from /usr/java/j2sdk1.4
[Loaded java.lang.String from /usr/java/j2sdk1.4.1/jre
```

Share Improve this answer

Follow

answered Feb 18, 2019 at 9:19

Per Lundberg

4,211 ● 1 ● 38 ● 49

1

I find this class be loaded twice. Find the reason is that parallelWebappClassLoader load class by itself first rather than use it's parent classLoader.

Share Improve this answer

Follow

edited Oct 29, 2019 at 13:20

ChrisF ♦

137k ● 31 ● 262 ● 333

answered Oct 29, 2019 at 12:00

zero zero

11 ● 1

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.