/

Java EE Support Patterns

Cloud, Java, Middleware, APM & Tools tutorials

java.lang.NoClassDefFoundError: Parent first Classloader

8/23/2012 PIERRE-HUGUES CHARBONNEAU 8 COMMENTS

This is part 4 of our "Exception in thread main java.lang.NoClassDefFoundError" troubleshooting series. As you saw from my previous articles, missing Jar files and failure of static initializers are the most common sources of this problem. However, sometimes the root cause is due to a more complex Java class loader problem which is quite common when developing Java EE applications involving multiple parent and child class loaders. The following article will describe one common problem pattern when using the default class loader delegation model.

A simple Java program will again be provided in order to help you understand this problem pattern.

Default JVM Classloader delegation model

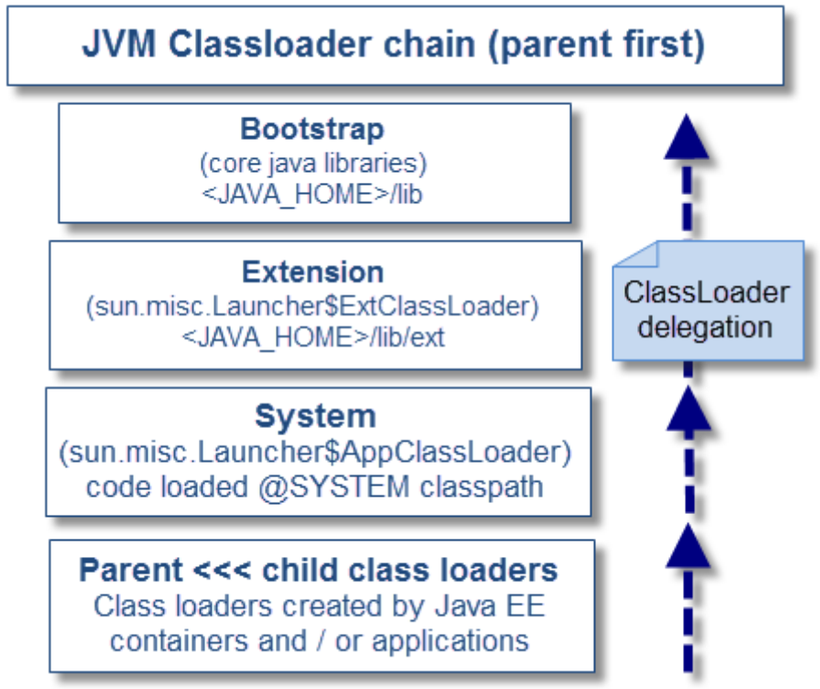
As we saw from the first article, the default class loader delegation model is from bottom-up e.g. parent first. This means that the JVM is going up to the class loader chain in order to find and load each of your application Java classes. If the class is not found from the parent class loaders, the JVM will then attempt to load it from the current Thread context class loader; typically a child class loader.

RECENT ARTICLES

Oracle WebLogic Native IO & Java Muxers
This article will provide the complete root cause analysis details...

Oracle Open World and Java One 2016 summary later this week
This post is to inform you that I will publish...

Java 8 Performance Optimization - DZone Refcard Update



NoClassDefFoundError problems can occur, for example, when you wrongly package your application (or third part API's) between the parent and child class loaders. Another example is code / JAR files injection by the container itself or third party API's deployed at a higher level in the class loader chain.

In the above scenarios:

- The JVM loads one part of the affected code to a **parent** class loader (SYSTEM or parent class loaders)
- The JVM loads the other parts of the affected code to a **child** class loader (Java EE container or application defined class loader)

Now what happens when Java classes loaded from the parent attempt to load reference classes deployed only to the child classloader? NoClassDefFoundError!

I am happy to inform you that I published recently...

DZone's Guide to Building and Deploying Applications on the Cloud
This post is to inform you that DZone has just...

Java 8 - CPU Flame Graph
Brendan Gregg and Martin Spier from Netflix recently shared a...

HEADLINES BY FEEDBURNER

ABOUT THE AUTHOR

PIERRE-HUGUES CHARBONNEAU

P-H is an IT Architect currently working for CGI Inc. in Canada. He has 15 years+ of experience developing and troubleshooting Java & Web enterprise systems. Email: phcharbonneau@hotmail.com.

VIEW MY COMPLETE PROFILE

BLOG ARCHIVE

Please remember that a parent class loader has no visibility or access to child class loaders. This means that any referencing code must be found either from the parent class loader chain (bottom-up) or at the current Thread context class loader level; otherwise java.lang.NoClassDefFoundError is thrown by the JVM.

This is exactly what the following Java program will demonstrate.

Sample Java program

The following simple Java program is split as per below:

- The main Java program `NoClassDefFoundErrorSimulator` is packaged in **MainProgram.jar**
- A logging utility class `JavaEETrainingUtil` is packaged in **MainProgram.jar**
- The Java class caller `CallerClassA` is packaged in **caller.jar**
- The referencing Java class `ReferencingClassA` is packaged in **referencer.jar**

These following tasks are performed:

- Create a child class loader (java.net.URLClassLoader)
- Assign the caller and referencing Java class jar files to the child class loader
- Change the current Thread context ClassLoader to the child ClassLoader
- Attempt to load and create a new instance of `CallerClassA` from the current Thread context class loader e.g. child
- Proper logging was added in order to help you understand the class loader tree and Thread context class loader state

It will demonstrate that a wrong packaging of the application code is leading to a NoClassDefFoundError as per the default class loader delegation model.

** *`JavaEETrainingUtil` source code can be found from the article part #2*

NoClassDefFoundErrorSimulator.java

```
package org.ph.javaee.training3;
```

```
import java.net.URL;
import java.net.URLClassLoader;

import org.ph.javaee.training.util.JavaEETrainingUtil;

/**
 * NoClassDefFoundErrorSimulator
 * @author Pierre-Hugues Charbonneau
 *
 */
public class NoClassDefFoundErrorSimulator {

    /**
     * @param args
     */
    public static void main(String[] args) {

        System.out.println("java.lang.NoClassDefFoundError Simulator - Training 3");
        System.out.println("Author: Pierre-Hugues Charbonneau");
        System.out.println("http://javaeesupportpatterns.blogspot.com");

        // Local variables
        String currentThreadName = Thread.currentThread().getName();
        String callerFullClassName = "org.ph.javaee.training3.CallerClassA";

        // Print current ClassLoader context & Thread
        System.out.println("\nCurrent Thread name: '"+currentThreadName+"'");
        System.out.println("Initial ClassLoader chain: "+JavaEETrainingUtil.getCurrentClassloaderDetail());

        try {
            // Location of the application code for our child ClassLoader
            URL[] webAppLibURL = new URL[] {new URL("file:caller.jar"),new URL("file:referencer.jar")};
```

```
                // Child ClassLoader instance creation
                URLClassLoader childClassLoader = new URLClassLoader(webAppLibURL);

                /*** Application code execution... ***/

                // 1. Change the current Thread ClassLoader to the child ClassLoader
                Thread.currentThread().setContextClassLoader(childClassLoader);
                System.out.println(">> Thread '"+currentThreadName+"' Context ClassLoader now changed to
'"+childClassLoader+"'");
                System.out.println("\nNew ClassLoader chain: "+JavaEETrainingUtil.getCurrentClassloaderDetail());

                // 2. Load the caller Class within the child ClassLoader...
                System.out.println(">> Loading '"+callerFullClassName+"' to child ClassLoader '"+childClassLoader+"'...");
                Class<?> callerClass = childClassLoader.loadClass(callerFullClassName);

                // 3. Create a new instance of CallerClassA
                Object callerClassInstance = callerClass.newInstance();

            } catch (Throwable any) {
                System.out.println("Throwable: "+any);
                any.printStackTrace();
            }

            System.out.println("\nSimulator completed!");
        }
}
```

CallerClassA.java

```
package org.ph.javaee.training3;

import org.ph.javaee.training3.ReferencingClassA;
```

```
/**
 * CallerClassA
 * @author Pierre-Hugues Charbonneau
 *
 */
public class CallerClassA {

	private final static Class<CallerClassA> CLAZZ = CallerClassA.class;

	static {
		System.out.println("Class loading of "+CLAZZ+" from ClassLoader '"+CLAZZ.getClassLoader()+"' in progress...");
	}

	public CallerClassA() {
		System.out.println("Creating a new instance of "+CallerClassA.class.getName()+"...");

		doSomething();
	}

	private void doSomething() {

		// Create a new instance of ReferencingClassA
		ReferencingClassA referencingClass = new ReferencingClassA();
	}
}
```

ReferencingClassA.java

```
package org.ph.javaee.training3;

/**
```

```
 * ReferencingClassA
 * @author Pierre-Hugues Charbonneau
 *
 */
public class ReferencingClassA {

	private final static Class<ReferencingClassA> CLAZZ = ReferencingClassA.class;

	static {
		System.out.println("Class loading of "+CLAZZ+" from ClassLoader '"+CLAZZ.getClassLoader()+"' in progress...");
	}

	public ReferencingClassA() {
		System.out.println("Creating a new instance of "+ReferencingClassA.class.getName()+"...");
	}

	public void doSomething() {
		//nothing to do...
	}
}
```

Problem reproduction

In order to replicate the problem, we will simply “voluntary” split the packaging of the application code (caller & referencing class) between the parent and child class loader.

For now, let’s run the program with the right JAR files deployment and class loader chain:

- The main program and utility class are deployed at the **parent** class loader (SYSTEM classpath)

- `CallerClassA` and `ReferencingClassA` and both deployed at the **child** class loader level

Baseline (normal execution)

<JDK_HOME>\bin>java -classpath MainProgram.jar org.ph.javaee.training3.NoClassDefFoundErrorSimulator

java.lang.NoClassDefFoundError Simulator - Training 3

Author: Pierre-Hugues Charbonneau

http://javaeesupportpatterns.blogspot.com

Current Thread name: 'main'

Initial ClassLoader chain:

---

sun.misc.Launcher$ExtClassLoader@17c1e333

--- delegation ---

sun.misc.Launcher$AppClassLoader@214c4ac9 **Current Thread 'main' Context ClassLoader**

>> Thread 'main' Context ClassLoader now changed to 'java.net.URLClassLoader@6a4d37e5'

New ClassLoader chain:

---

sun.misc.Launcher$ExtClassLoader@17c1e333

--- delegation ---

sun.misc.Launcher$AppClassLoader@214c4ac9

--- delegation ---

java.net.URLClassLoader@6a4d37e5 **Current Thread 'main' Context ClassLoader**

>> Loading 'org.ph.javaee.training3.CallerClassA' to child ClassLoader 'java.net.URLClassLoader@6a4d37e5'...

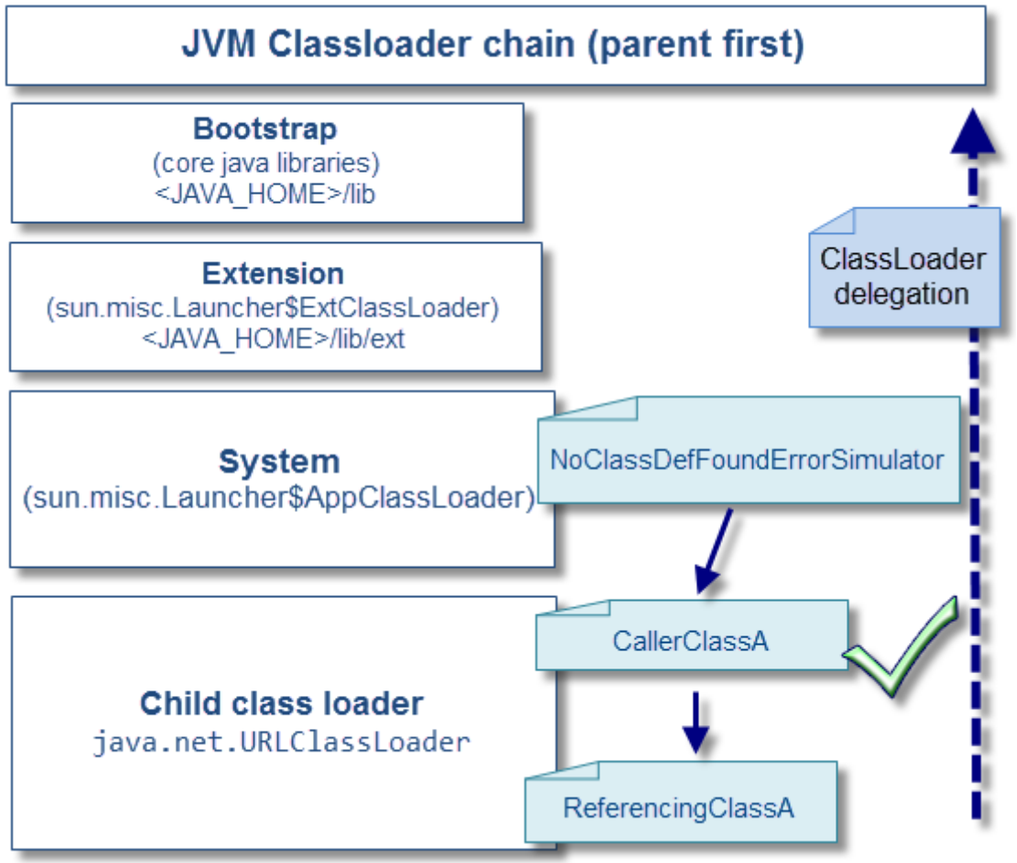
Class loading of class org.ph.javaee.training3.CallerClassA from ClassLoader 'java.net.URLClassLoader@6a4d37e5' in progress...

Creating a new instance of org.ph.javaee.training3.CallerClassA...

Class loading of class org.ph.javaee.training3.ReferencingClassA from ClassLoader 'java.net.URLClassLoader@6a4d37e5' in progress...

Creating a new instance of org.ph.javaee.training3.ReferencingClassA...

Simulator completed!



For the initial run (baseline), the main program was able to create successfully a new instance of `CallerClassA` from the child class loader (java.net.URLClassLoader) along with its referencing class with no problem.

Now let's run the program with the **wrong** application packaging and class loader chain:

- The main program and utility class are deployed at the **parent** class loader (SYSTEM classpath)
- `CallerClassA` and `ReferencingClassA` and both deployed at the **child** class loader level
- `CallerClassA` (caller.jar) is also deployed at the parent class loader level

Problem reproduction run (static variable initializer failure)

```
<JDK_HOME>\bin>java -classpath MainProgram.jar;caller.jar org.ph.javaee.training3.NoClassDefFoundErrorSimulator

java.lang.NoClassDefFoundError Simulator - Training 3
Author: Pierre-Hugues Charbonneau
http://javaeesupportpatterns.blogspot.com

Current Thread name: 'main'
Initial ClassLoader chain:
-----------------------------------------------------------------
sun.misc.Launcher$ExtClassLoader@17c1e333
--- delegation ---
sun.misc.Launcher$AppClassLoader@214c4ac9 **Current Thread 'main' Context ClassLoader**
-----------------------------------------------------------------

>> Thread 'main' Context ClassLoader now changed to 'java.net.URLClassLoader@6a4d37e5'

New ClassLoader chain:
-----------------------------------------------------------------
sun.misc.Launcher$ExtClassLoader@17c1e333
--- delegation ---
sun.misc.Launcher$AppClassLoader@214c4ac9
--- delegation ---
java.net.URLClassLoader@6a4d37e5 **Current Thread 'main' Context ClassLoader**
```

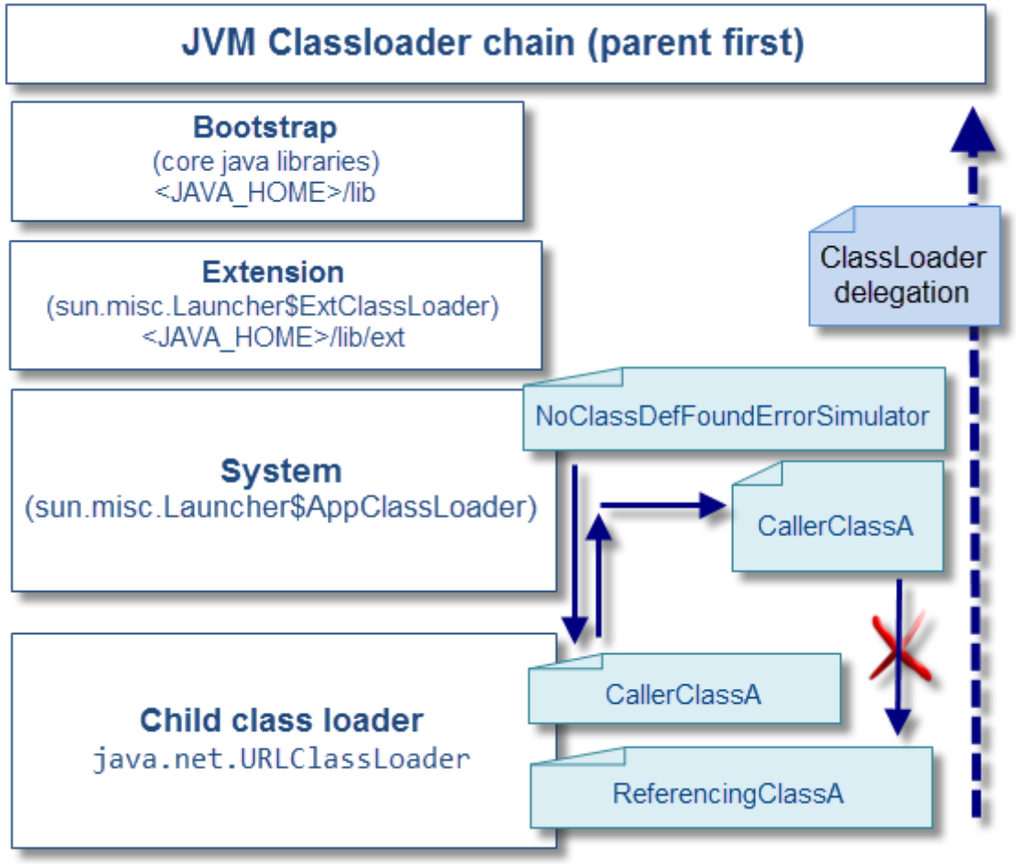
```
------------------------------------------------------------------

>> Loading 'org.ph.javaee.training3.CallerClassA' to child ClassLoader 'java.net.URLClassLoader@6a4d37e5'...
Class loading of class org.ph.javaee.training3.CallerClassA from ClassLoader
'sun.misc.Launcher$AppClassLoader@214c4ac9' in progress...// Caller is loaded from the parent class loader, why???
Creating a new instance of org.ph.javaee.training3.CallerClassA...
Throwable: java.lang.NoClassDefFoundError: org/ph/javaee/training3/ReferencingClassA
java.lang.NoClassDefFoundError: org/ph/javaee/training3/ReferencingClassA
        at org.ph.javaee.training3.CallerClassA.doSomething(CallerClassA.java:27)
        at org.ph.javaee.training3.CallerClassA.<init>(CallerClassA.java:21)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
        at java.lang.reflect.Constructor.newInstance(Unknown Source)
        at java.lang.Class.newInstance0(Unknown Source)
        at java.lang.Class.newInstance(Unknown Source)
        at org.ph.javaee.training3.NoClassDefFoundErrorSimulator.main(NoClassDefFoundErrorSimulator.java:51)
Caused by: java.lang.ClassNotFoundException: org.ph.javaee.training3.ReferencingClassA
        at java.net.URLClassLoader$1.run(Unknown Source)
        at java.net.URLClassLoader$1.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        ... 9 more

Simulator completed!
```

What happened?

- The main program and utility classes were loaded as expected from the parent class loader (sun.misc.Launcher$AppClassLoader)
- The Thread context class loader was changed to child class loader as expected which includes both caller and referencing jar files
- However, we can see that `CallerClassA` was actually loaded by the **parent** class loader (sun.misc.Launcher$AppClassLoader) instead of the child class loader
- Since `ReferencingClassA` was not deployed to the parent class loader, the class cannot be found from the current class loader chain since the parent class loader has no visibility on the child class loader, NoClassDefFoundError is thrown



The key point to understand at this point is why `CallerClassA` was loaded by the parent class loader. The answer is with the default class loader delegation model. Both child and parent class loaders contain the caller JAR files. However, the default delegation model is always parent first which is why it was loaded at that level. The problem is that the caller contains a class reference to ReferencingClassA which is only deployed to the child class loader; java.lang.NoClassDefFoundError condition is met.

As you can see, a packaging problem of your code or third part API can easily lead to this problem due to the default class loader delegation behaviour. It is very important that you review your class loader chain and determine if you are at risk of duplicate code or libraries across your parent and child class loaders.

Recommendations and resolution strategies

Now find below my recommendations and resolution strategies for this problem pattern:

- Review the java.lang.NoClassDefFoundError error and identify the Java class that the JVM is complaining about
- Review the packaging of the affected application(s), including your Java EE container and third part API's used. The goal is to identify duplicate or wrong deployments of the affected Java class at runtime (SYSTEM class path, EAR file, Java EE container itself etc.).
- Once identified, you will need to remove and / or move the affected library/libraries from the affected class loader (complexity of resolution will depend of the root cause).
- Enable JVM class verbose e.g. –verbose:class. This JVM debug flag is very useful to monitor the loading of the Java classes and libraries from the Java EE container your applications. It can really help you pinpoint duplicate Java class loading across various applications and class loaders at runtime

Please feel free to post any question or comment. The next and last article of this series will focus on NoClassDefFoundError when using the "child first" delegation model.

Posted in: ClassLoader,Java

8 comments:

Willard9876 says:
December 27, 2012 at 9:59 AM

[part1]
Hi P-H, can you recreate and solve this error please?
It's enough important.

I use eclipse.
There is this software, rssowl 2.1.x (http://wiki.rssowl.org/index.php/Build_RSSOwl_2.0 https://rssowl.svn.sourceforge.net/svnroot/rssowl/branches/2.1.x/) a news-aggregator.
I am trying to add the possibility do download news (web pages) in mht format.
There is another software HTML2MHTCompiler (http://www.uplook.cn/blog/6/62956/) that download web pages and save them in mht.
I am trying to integrate HTML2MHTCompiler in rssowl.

I have tested HTML2MHTCompiler apart (not in rssowl) and works fine.
I have problems when call HTML2MHTCompiler from rssowl.

To integrate HTML2MHTCompiler in rssowl 2.1.x:
1)I create a package org.rssowl.core.downloader in org.rssowl.core/src, and put Html2MHTCompiler(.java) into it.
2)I replaced the original file RSSInterpreter at org.rssowl.core.internal.interpreter with this (http://justpaste.it/1ogc).

There is a little difference between this two files.
In mine I added a block of code (named Download block) which calls HTML2MHTCompiler.

//Download block

```
{

//Html2MHTCompiler
String strUrl = "http://www.mtime.com/my/tropicofcancer/blog/843555/"; //$NON-NLS-1$
String strEncoding = "utf-8"; //$NON-NLS-1$



System.out.println("rssi_mht1"); //$NON-NLS-1$
Html2MHTCompiler h2t = new Html2MHTCompiler(strUrl, strEncoding,
"/home/bender/my_software/test/test.mht"); //$NON-NLS-1$
h2t.compile();
System.out.println("rssi_mht2"); //$NON-NLS-1$
//Html2MHTCompiler



} // End Download Block
```

When in eclipse I run rssowl, it runs.
[see pic1: http://imageshack.us/a/img94/9794/pic1hf.png] RSSInterpreter is called when I click update (a feed).

RSSInterpreter (should) call Html2MHTCompiler.
I have a NoClassDefFoundError at line of code: Html2MHTCompiler h2t = new Html2MHTCompiler(strUrl, strEncoding, "/home/bender/my_software/test/test.mht"); //$NON-NLS-1$
(Note I use an ablsolute path, "/home/bender/my_software/test/test.mht", if you try to recreate the error you should change it)

Reply

Willard9876 says:

December 27, 2012 at 10:00 AM

[part2]
!ENTRY org.rssowl.core 4 2 2012-12-27 14:32:03.250
!MESSAGE Problems occurred when invoking code from plug-in: "org.rssowl.core".
!STACK 0
java.lang.NoClassDefFoundError: javax/mail/Address
at org.rssowl.core.internal.interpreter.RSSInterpreter.processItems(RSSInterpreter.java:484)
at org.rssowl.core.internal.interpreter.RSSInterpreter.processChannel(RSSInterpreter.java:163)
at org.rssowl.core.internal.interpreter.RSSInterpreter.processFeed(RSSInterpreter.java:136)
at org.rssowl.core.internal.interpreter.RSSInterpreter.interpret(RSSInterpreter.java:104)
at
org.rssowl.core.internal.interpreter.InterpreterServiceImpl.interpretJDomDocument(Interpreter
ServiceImpl.java:159)
at
org.rssowl.core.internal.interpreter.InterpreterServiceImpl.interpret(InterpreterServiceImpl.java:
124)
at
org.rssowl.core.internal.connection.DefaultProtocolHandler.reload(DefaultProtocolHandler.java:
181)
at
org.rssowl.core.internal.connection.ConnectionServiceImpl.reload(ConnectionServiceImpl.java:1
96)
at org.rssowl.ui.internal.Controller.reload(Controller.java:820)
at org.rssowl.ui.internal.Controller.access$0(Controller.java:746)
at org.rssowl.ui.internal.Controller$ReloadTask.run(Controller.java:335)
at org.rssowl.core.util.JobQueue$2$1.run(JobQueue.java:329)
at org.eclipse.core.runtime.SafeRunner.run(SafeRunner.java:42)
at org.rssowl.core.util.JobQueue$2.run(JobQueue.java:326)
at org.eclipse.core.internal.jobs.Worker.run(Worker.java:54)
Caused by: java.lang.ClassNotFoundException: javax.mail.Address
at org.eclipse.osgi.internal.loader.BundleLoader.findClassInternal(BundleLoader.java:513)
at org.eclipse.osgi.internal.loader.BundleLoader.findClass(BundleLoader.java:429)
at org.eclipse.osgi.internal.loader.BundleLoader.findClass(BundleLoader.java:417)
at
org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader.loadClass(DefaultClassLoader.java:107
)
at java.lang.ClassLoader.loadClass(ClassLoader.java:264)
at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:332)
... 15 more
##

In figure2 (http://imageshack.us/f/89/pic2fjvclo.png/) there is my Java Build Path for the project org.rssowl.core

This are the JARs:
jaf-1.1.1: http://www.oracle.com/technetwork/java/javase/downloads/index-135046.html
HTMLParser-2.0-SNAPSHOT-bin.zip: http://sourceforge.net/projects/htmlparser/files/
javamail 1.4.5: http://www.oracle.com/technetwork/java/index-138643.html

In figure3 and 4 there is an overview of the project:
http://imageshack.us/a/img255/8326/pic3y.png
http://imageshack.us/a/img543/5379/pic4wc.png

I spent last 3-4 day on internet to try to solve this problem without success.
Thanks

Reply

Willard9876 says:
December 27, 2012 at 8:48 PM

[part3]
I have tried to change software: I replace HTML2MHTCompiler with htmlunit (http://sourceforge.net/projects/htmlunit/files/htmlunit/2.11/)
With htmlunit you can download complete page. At this link there is an example (http://stackoverflow.com/questions/6467170/fetching-the-entire-web-page-from-a-specific-url-using-java)
I have a similar error.

I tested htmlunit apart (not in rssowl) and works fine.
When I call htmlunit from rssowl I have a NoClassDefFoundError.

In RSSInterpreter I replace the previous "Download block" with this:

```
//Download block
{

//start htmlunit block
WebClient webClient = new WebClient();
//end htmlunit block
} // End Download Block
```

In this new block I only create a new instance of "WebClient".
I do this only for test.
I have this error

```
############################
!ENTRY org.rssowl.core 4 2 2012-12-28 02:39:41.069
!MESSAGE Problems occurred when invoking code from plug-in: "org.rssowl.core".
!STACK 0
java.lang.NoClassDefFoundError: com/gargoylesoftware/htmlunit/WebClient
at org.rssowl.core.internal.interpreter.RSSInterpreter.processItems(RSSInterpreter.java:539)
at org.rssowl.core.internal.interpreter.RSSInterpreter.processChannel(RSSInterpreter.java:164)
at org.rssowl.core.internal.interpreter.RSSInterpreter.processFeed(RSSInterpreter.java:137)
at org.rssowl.core.internal.interpreter.RSSInterpreter.interpret(RSSInterpreter.java:105)
at
org.rssowl.core.internal.interpreter.InterpreterServiceImpl.interpretJDomDocument(Interpreter
ServiceImpl.java:159)
at
org.rssowl.core.internal.interpreter.InterpreterServiceImpl.interpret(InterpreterServiceImpl.java:
124)
at
org.rssowl.core.internal.connection.DefaultProtocolHandler.reload(DefaultProtocolHandler.java:
181)
at
org.rssowl.core.internal.connection.ConnectionServiceImpl.reload(ConnectionServiceImpl.java:1
96)
at org.rssowl.ui.internal.Controller.reload(Controller.java:820)
at org.rssowl.ui.internal.Controller.access$0(Controller.java:746)
```

at org.rssowl.ui.internal.Controller$ReloadTask.run(Controller.java:335)
at org.rssowl.core.util.JobQueue$2$1.run(JobQueue.java:329)
at org.eclipse.core.runtime.SafeRunner.run(SafeRunner.java:42)
at org.rssowl.core.util.JobQueue$2.run(JobQueue.java:326)
at org.eclipse.core.internal.jobs.Worker.run(Worker.java:54)

##############################
Obviously I add all the libraries of htmlunit in the "Java Build Path"

At this point I think the problem is eclipse or rssowl

Reply

christi parks says:
January 28, 2013 at 4:01 AM

Hello, sir i would like to ask that what is the scope of java training, what all topics should be covered and it is kinda bothering me ... and has anyone studies from this course http://www.wiziq.com/course/1779-core-and-advance-java-concepts of core and advance java online ?? or tell me any other guidance...
would really appreciate help... and Also i would like to thank for all the information you are providing on java concepts.

Reply

RobManc says:
May 12, 2013 at 9:24 AM

Hi Pierre Thanks for you articles they are informative. I wonder if I could ask you to help me. I am getting the following error when running an applet in my browser that accesses a WCF web service.
java.lang.NoClassDefFoundError: javax/xml/rpc/ServiceException

I have checked and added the jaxrpc.jar file to my build path but the problem still continues.

The WCF service seems to be the issue as the stack trace complains at the line of code that i first try to instantiate a proxy service. The applet works fine in eclipse but when I export the project to a jar file and try to run in a html5 page it throws this error. The stack trace is:

java.lang.RuntimeException: java.lang.NoClassDefFoundError: javax/xml/rpc/ServiceException
at com.sun.deploy.uitoolkit.impl.awt.AWTAppletAdapter.instantiateApplet(Unknown Source)
at sun.plugin2.applet.Plugin2Manager.initAppletAdapter(Unknown Source)
at sun.plugin2.applet.Plugin2Manager$AppletExecutionRunnable.run(Unknown Source)
at java.lang.Thread.run(Unknown Source)
Caused by: java.lang.NoClassDefFoundError: javax/xml/rpc/ServiceException
at ListProducts.ListProducts.(ListProducts.java:25)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
at java.lang.reflect.Constructor.newInstance(Unknown Source)
at java.lang.Class.newInstance0(Unknown Source)
at java.lang.Class.newInstance(Unknown Source)
at com.sun.deploy.uitoolkit.impl.awt.AWTAppletAdapter$1.run(Unknown Source)
at java.awt.event.InvocationEvent.dispatch(Unknown Source)
at java.awt.EventQueue.dispatchEventImpl(Unknown Source)
at java.awt.EventQueue.access$200(Unknown Source)
at java.awt.EventQueue$3.run(Unknown Source)
at java.awt.EventQueue$3.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.ProtectionDomain$1.doIntersectionPrivilege(Unknown Source)
at java.security.ProtectionDomain$1.doIntersectionPrivilege(Unknown Source)
at java.awt.EventQueue$4.run(Unknown Source)
at java.awt.EventQueue$4.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.ProtectionDomain$1.doIntersectionPrivilege(Unknown Source)
at java.awt.EventQueue.dispatchEvent(Unknown Source)
at java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown Source)
at java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
at java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown Source)
at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.awt.EventDispatchThread.run(Unknown Source)
Caused by: java.lang.ClassNotFoundException: javax.xml.rpc.ServiceException
at sun.plugin2.applet.Applet2ClassLoader.findClass(Unknown Source)

at sun.plugin2.applet.Plugin2ClassLoader.loadClass0(Unknown Source)
at sun.plugin2.applet.Plugin2ClassLoader.loadClass(Unknown Source)
at sun.plugin2.applet.Plugin2ClassLoader.loadClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
... 27 more

Hope you can give me a hint
Regards
Rob

Reply

Andy says:
May 31, 2013 at 5:43 AM

Hi!! Can you help me?? This is my error code:
javax.servlet.ServletException: java.lang.NoClassDefFoundError: Could not initialize class com.octo.captcha.image.gimpy.GimpyFactory
at
org.apache.jasper.runtime.PageContextImpl.doHandlePageException(PageContextImpl.java:862)
at org.apache.jasper.runtime.PageContextImpl.handlePageException(PageContextImpl.java:791)
at
org.apache.jsp.WEB_002dINF.jsp.offline.system.modules.com_saga_opencms_templatesaga_formaccesible.pages.captcha_jsp._jspService(captcha_jsp.java:71)
at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:70)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
at org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:377)
at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:313)
at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:260)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:290)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at org.apache.catalina.core.ApplicationDispatcher.invoke(ApplicationDispatcher.java:646)
at org.apache.catalina.core.ApplicationDispatcher.doInclude(ApplicationDispatcher.java:551)
at org.apache.catalina.core.ApplicationDispatcher.include(ApplicationDispatcher.java:488)
at

```
org.opencms.flex.CmsFlexRequestDispatcher.includeExternal(CmsFlexRequestDispatcher.java:1
94)
at org.opencms.flex.CmsFlexRequestDispatcher.include(CmsFlexRequestDispatcher.java:169)
at org.opencms.loader.CmsJspLoader.service(CmsJspLoader.java:555)
at
org.opencms.flex.CmsFlexRequestDispatcher.includeInternalWithCache(CmsFlexRequestDispatc
her.java:423)
at org.opencms.flex.CmsFlexRequestDispatcher.include(CmsFlexRequestDispatcher.java:173)
at org.opencms.loader.CmsJspLoader.dispatchJsp(CmsJspLoader.java:829)
at org.opencms.loader.CmsJspLoader.load(CmsJspLoader.java:512)
at org.opencms.loader.CmsResourceManager.loadResource(CmsResourceManager.java:1052)
at org.opencms.main.OpenCmsCore.showResource(OpenCmsCore.java:1492)
at org.opencms.main.OpenCmsServlet.doGet(OpenCmsServlet.java:153)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
[....]
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:191)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:127)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:109)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:298)
at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:857)
at
org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Protocol.ja
va:588)
at org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:489)
at java.lang.Thread.run(Thread.java:722)
Caused by: java.lang.NoClassDefFoundError: Could not initialize class
com.octo.captcha.image.gimpy.GimpyFactory
at
com.saga.opencms.formgenerator.CmsCaptchaEngine.initGimpyFactory(CmsCaptchaEngine.jav
a:174)
at com.saga.opencms.formgenerator.CmsCaptchaEngine.(CmsCaptchaEngine.java:96)
at com.saga.opencms.formgenerator.CmsCaptchaService.(CmsCaptchaService.java:67)
at
com.saga.opencms.formgenerator.CmsCaptchaServiceCache.getCaptchaService(CmsCaptchaSer
viceCache.java:155)
at
com.saga.opencms.formgenerator.CmsCaptchaField.writeCaptchaImage(CmsCaptchaField.java:
```

265)
at
org.apache.jsp.WEB_002dINF.jsp.offline.system.modules.com_saga_opencms_templatesaga_formaccesible.pages.captcha_jsp._jspService(captcha_jsp.java:63)
... 35 more

That library (com.octo.captcha.image.gimpy.GimpyFactory) is in jcaptcha-all-1.0-RC6.jar, and it's referenced in my project. I can´t understand that error. Oh my god....For three days with the same error and I can´t solve it :'(

Bye and thanks from Spain!! ;)

Reply

Pierre-Hugues Charbonneau says:
June 11, 2013 at 1:10 PM

Hi Andy,

Did you look for errors prior to NoClassDefFoundError: com.octo.captcha.image.gimpy.GimpyFactory?

Sometimes static initializer code can fail, preventing class loading and leading to this error. Check your logs and check if you see any error thrown from com.octo.captcha.image.gimpy.GimpyFactory prior to the FIRST occurrence of NoClassDefFoundError.

Regards,
P-H

Reply

Sam says:
June 5, 2015 at 2:01 PM

Nice articles indeed. An issue with this kind of duplicate loading (parent + child) is that in the case above it is easy to find that. But many times frameworks may load jars specifically which

makes it impossible to spot in code. In such cases, when code may not be available, a good idea would be to just remove offending jar from classpath and see if it still goes through or not. Would you agree? Or am I missing something?

Reply

Post a Comment