Hey, what’s up?

This is Palle Cogburn. In this video I will discuss class loading in Java, and in particular the problem that gave me the idea to do this video in the first place.

# The problem

The problem is that you have some Java based application that has been developed a long time ago. The original developers are long gone. It works in production, you have the source code in a source control system, but no one can remember how to setup a development environment for this application anymore.

Now of course there are some problems reported by customers, and you would like to fix those problems and release a new version of the product. Maybe you want to replace the product all together, or fold the functionality into another product. But in order to do that safely, you need to be confident that you know what the existing application did, and maybe at least re-use some of the old code.

The source code is not directly sufficient to determine the dependencies of the application. It only contains import statements that contain the names or packages of the dependent classes. It is missing the version information, and the location of the dependencies.

[Show examples of import statements here both for classes, and packages]

I want to find out where the application finds these dependencies. To do this, we will create a custom class loader.

A desktop Java application can be started either with the classpath specified on the command-line, in the CLASSPATH environment variable, or a combination like this:

[Show command-line where CLASSPATH is specified as a combination of a specific path, and the environment variable]

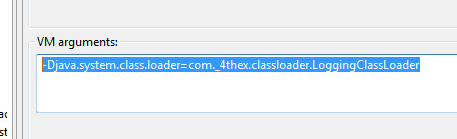
java -classpath %CLASSPATH%;bin com.\_4thex.Main

If the production machine runs multiple applications, and they are all started without a specific classpath, it is impossible to know where the dependencies are loaded from, without a custom classloader.

# How to specify the custom class loader

If your application is a desktop application, you can specify a custom class loader with the system property ‘java.system.class.loader’.

[Show example of specifying the property on the command-line when starting the application]



You also need to add the location of your custom classloader to the classpath.

If your application is a Web Application that runs in Tomcat 7 for example, you need to specify the classloader in the file

/META-INF/context.xml

Here is an example context.xml file:

[Show example context.xml file that shows the Loader element]



The classloader implementation must extend ‘org.apache.catalina.loader.WebappClassLoader’.

We will override the ‘loadClass’ method.