Title: getting a java.lang.ClassNotFoundException when starting a Docker container

Post Body:

I am having a problem starting a .jar file within a Docker Container. The error message I am getting is:

    Exception in thread 'main' java.lang.ClassNotFoundException: ccinfw.MainApplication

I have added information below. Why am I getting this error and how can I fix it?

TIA

Manifest File

```
Manifest-Version: 1.0 Implementation-Title: SpringBootErrorHandlingDemoy Implementation-Version: 0.0.1-SNAPSHOT Archiver-Versi
```

ran jar -xvf BackEndRestFunctionality-0.0.1-SNAPSHOT.jar > expanded. Below is a portion of the output. MainApplication.class does exist in the jar file

```
 [... snip ...]   inflated: BOOT-INF/classes/ccinfw/security/user/ApplicationUser.class  inflated: BOOT-INF/classes/ccinfw/sec
```

I am using the following software

```
jenkins-slave-one:/var/jenkins/workspace/build-cc-restapi-dev/target/docker# java -version openjdk version '1.8.0_131' OpenJDK
```

Maven Dependencies

I am getting the following error message when starting it:

```
Exception in thread 'main' java.lang.ClassNotFoundException: ccinfw.MainApplication at java.net.URLClassLoader.findClass(URLCl
```

the maven command executed within Jenkins

```
$ /opt/maven/bin/mvn -f /var/jenkins/workspace/build-cc-restapi-dev/pom.xml -Pdevelopment clean package docker:build -B
```

portion of pom.xml file responsible for building

```
<profile>     <id>development</id>     <build>         <plugins>             <plugin>                 <artifactId>maven-antrun
```

the Dockerfile being accessed

```
FROM java:8 ADD /BackEndRestFunctionality-0.0.1-SNAPSHOT.jar // ENTRYPOINT ['java', '-jar', '/BackEndRestFunctionality-0.0.1-S
```

Here is a part of the Jenkins log file where the Docker Image file is being built using the generated .jar file - again - the .jar file works fine when ran on its own

```
[INFO] Copying /var/jenkins/workspace/build-cc-restapi-dev/target/BackEndRestFunctionality-0.0.1-SNAPSHOT.jar -> /var/jenkins/
```

Docker Image Layers

```
351.5 MB     RUN set -x && apt-get update && apt-get install -y openjdk-8-jdk='$JAVA_DEBIAN_VERSION' ca-certificates-java='$CA
```

Accepted Answer:

I found out what the problem was - thank goodness for having a version in Gitlab to refer to (as well as a 'previous' version on another machine)

Basically, I used Maven to put the project together. I executed the following command from Jenkins to build the .jar file

```
$ /opt/maven/bin/mvn -f /var/jenkins/workspace/build-cc-restapi-dev/pom.xml -Pdevelopment clean package docker:build -B
```

In the pom.xml file, I did not specify :

```
<artifactId></artifactId>
```

When I went ahead and set it

```
ex: <artifactId>stuff</artifactId>
```

Docker started the jar file correctly and everything worked fine.

Highest Rated Answer:

I found out what the problem was - thank goodness for having a version in Gitlab to refer to (as well as a 'previous' version on another machine)

Basically, I used Maven to put the project together.

To build the project, I set up the command below and ran it under Jenkins:

```
$ /opt/maven/bin/mvn -f /var/jenkins/workspace/build-cc-restapi-dev/pom.xml -Pdevelopment clean package docker:build -B
```

In the pom.xml file, I did not specify :

```
<artifactId></artifactId>
```

When I went ahead and set it

```
ex: <artifactId>stuff</artifactId>
```

Docker started the jar file correctly and everything worked fine.

UPDATE Need to report items more accurately.

```
<artifactId></artifactId>
```

was defined. But

```
<artifactId>stuff</artifactId> => worked <artifactId>ccinfieldworkserver</artifactId> => worked <artifactId>BackEndRestFunctio
```

TIA