# How do I put all required JAR files in a library folder inside the final JAR file with Maven?





91

32

I am using Maven in my standalone application, and I want to package all the dependencies in my JAR file inside a library folder, as mentioned in one of the answers here:

*How can I create an executable JAR with dependencies using Maven?*

I want my final JAR file to have a library folder that contains the dependencies as JAR files, not like what the `maven-shade-plugin` that puts the dependencies in the form of folders like the Maven hierarchy in the .m2 folder.

Well, actually the current configuration does what I want, but I am having a problem with loading the JAR files when running the application. I can't load the classes.

Here's my configuration:

```
<plugins>

    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-dependency-plugin</artifactId>
        <executions>
            <execution>
                <id>copy-dependencies</id>
                <phase>prepare-package</phase>
                <goals>
                    <goal>copy-dependencies</goal>
                </goals>
                <configuration>

<outputDirectory>${project.build.directory}/classes/lib</outputDirectory>
                    <overWriteReleases>false</overWriteReleases>
                    <overWriteSnapshots>false</overWriteSnapshots>
                    <overWriteIfNewer>true</overWriteIfNewer>
                </configuration>
            </execution>
        </executions>
    </plugin>

    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <configuration>
            <archive>
                <manifest>
                    <addClasspath>true</addClasspath>
                    <classpathPrefix>lib/</classpathPrefix>
                    <mainClass>com.myapp.MainClass</mainClass>
                </manifest>
            </archive>
        </configuration>
    </plugin>

    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
            <configuration>
                <source>1.6</source>
                <target>1.6</target>
            </configuration>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <executions>
                <execution>
                    <id>install</id>
                    <phase>install</phase>
                    <goals>
                        <goal>sources</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-resources-plugin</artifactId>
            <version>2.5</version>
            <configuration>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>

    </plugins>
```

The project runs fine from Eclipse, and the JAR files are put in the library folder inside my final JAR file as I want, but when running the final JAR file from the target folder I always get `ClassNotFoundException`:

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/springframework/context/ApplicationContext
Caused by: java.lang.ClassNotFoundException:
```

```
org.springframework.context.ApplicationContext
        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)
Could not find the main class: com.myapp.MainClass. Program will exit.
```

How can I fix this exception?

java maven jar build-process classloader

edited May 23 '17 at 12:02

Community ♦

1 1

asked Aug 1 '12 at 11:52

Mahmoud Saleh

16k 104 291 453

1 which command do you use to run the jar? probably you may prefer maven exec plugin? – Andrey Borisov Aug 1 '12 at 12:05

Is the exception message out of date compared with the POM file? It seems the main class `com.myapp.MainClass` is being searched for, not `com.tastycafe.MainClass`. – Duncan Jones Aug 1 '12 at 12:06

@Duncan Jones, copy paste problem, i edited the question – Mahmoud Saleh Aug 1 '12 at 12:28

2 Note that if you want jars inside the jar, then the standard classloaders in Java cannot

understand them. – Thorbjørn Ravn Andersen May 8 '15 at 14:17

How to make it place the maven dependencies in a lib folder but outside the JAR? – ed22 Jul 2 '18 at 13:32

## 8 Answers

70

The following is my solution. Test it if it works for you:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-dependency-plugin</artifac
    <executions>
        <execution>
            <id>copy-dependencies</id>
            <phase>prepare-package</phase>
            <goals>
                <goal>copy-dependencies</goal>
            </goals>
            <configuration>

<outputDirectory>${project.build.directory}/clas
                <overWriteReleases>false</overWr
                <overWriteSnapshots>false</overW
                <overWriteIfNewer>true</overWrit
            </configuration>
        </execution>
    </executions>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <configuration>
```

```
            <archive>
                <manifest>
                    <addClasspath>true</addClasspath
                    <!-- <classpathPrefix>lib</class
                    <!-- <mainClass>test.org.Cliente
                </manifest>
                <manifestEntries>
                    <Class-Path>lib/</Class-Path>
                </manifestEntries>
            </archive>
        </configuration>
    </plugin>
```

The first plugin puts all dependencies in the target/classes/lib folder, and the second one includes the library folder in the final JAR file, and configures the `Manifest.mf` file.

But then you will need to add custom classloading code to load the JAR files.

Or, to avoid custom classloading, you can use "${project.build.directory}/lib, but in this case, you don't have dependencies inside the final JAR file, which defeats the purpose.

It's been two years since the question was asked. The problem of nested JAR files persists nevertheless. I hope it helps somebody.

edited Mar 14 '17 at 21:49

Peter Mortensen

**13.9k** 19 87 114

answered Aug 4 '14 at 10:35

Dalai Llama

**1,583** 1 17 26

1 Usage: mvn install, cd target, java -jar MyJarFile-1.0.jar – djb Apr 9 '15 at 22:52

Cool, this also works for cloudfoundry – Keymon Sep 8 '15 at 16:54

what do I miss? This creates a manifest classpath entry containing "lib/" and all single jar from the lib folder. Is this intended? Why? – gapvision Feb 25 '16 at 17:50

1 It worked after I changed "${project.build.directory}/classes/lib" to ${project.build.directory}/lib – Raju Sep 30 '16 at 13:22

1 Unfortunatelly includes dependencies declared as provided. – Philippe Gioseffi Dec 23 '16 at 18:36



Updated:

26

```
<build>
  <plugins>
    <plugin>
    <artifactId>maven-dependency-plugin</artifac
    <executions>
      <execution>
        <phase>install</phase>
        <goals>
```

```
            <goal>copy-dependencies</goal>
          </goals>
          <configuration>
             <outputDirectory>${project.build.di
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

edited Mar 15 '16 at 12:35

dan

**348** 2 11

answered Aug 1 '12 at 13:02

Ahmet Karakaya

**6,220** 14 56 110

1 that's not what i want, please read the question carefully – Mahmoud Saleh Aug 1 '12 at 13:08

2 this puts the lib folder outside the jar (which is not what i want). is that better than putting the lib inside the jar ? and how to deliver the application to the client in this case ? – Mahmoud Saleh Aug 1 '12 at 13:30

it is more clear for me now. let me make a google search. But I would like to know why you want to copy all dependency jar file in specified folder inside the executable jar file. If all dependency jar files are inside the jar file, why do you need to locate them in a lib folder? – Ahmet Karakaya Aug 1 '12 at 13:35

The simplest and the most efficient way is to use an uber plugin like this:

23

```
<plugin>
  <groupId>org.apache.maven.plugins</g
  <artifactId>maven-shade-plugin</arti
  <executions>
      <execution>
          <phase>package</phase>
          <goals>
              <goal>shade</goal>
          </goals>
      </execution>
  </executions>
  <configuration>
      <finalName>uber-${artifactId}-${
  </configuration>
</plugin>
```

You will have de-normalized all in one JAR file.

edited Mar 14 '17 at 21:46

Peter Mortensen
**13.9k** 19 87 114

answered Aug 1 '12 at 11:57

Andrey Borisov
**2,699** 12 17

use it in addition to my current config ? and what this plugin does exactly ?
– Mahmoud Saleh Aug 1 '12 at 11:59

14 i don't want my jar file to look like this, i want to have all the dependencies in a lib folder inside the jar file as netbean does.
– Mahmoud Saleh Aug 1 '12 at 12:02

12

The [executable packer maven plugin] can be used for exactly that purpose: creating standalone java applications containing all dependencies as JAR files in a specific folder.

Just add the following to your `pom.xml` inside the `<build><plugins>` section (be sure to replace the value of `mainClass` accordingly):

```
<plugin>
    <groupId>de.ntcomputer</groupId>
    <artifactId>executable-packer-maven-plugin</artifactId>
    <version>1.0.1</version>
    <configuration>
        <mainClass>com.example.MyMainClass</mainClass>
    </configuration>
    <executions>
        <execution>
            <goals>
                <goal>pack-executable-jar</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

The built JAR file is located at `target/<YourProjectAndVersion>-pkg.jar` after you run `mvn package`. All of its compile-time and runtime dependencies will be included in the `lib/` folder inside the JAR file.

Disclaimer: I am the author of the plugin.

edited Jun 4 '17 at 11:11

**1,122** 8 20

Would be good to change the name of destination jar – Ilya K. Jun 22 '17 at 2:42

I tried it and I can say it's working... As you said, this plugin creates a jar file, with its libraries (jars) into a lib directory in the jar... The configuration is really easy.. **I think it's what the author of this question is expecting** ... I will give you my vote up... The only drawback I have found on it (that´s the why I couldn´t use it), there is a delay when I execute my jar and the application is shown (about 20 secs) probably because of the process to register the libs in the custom classloader... But it´s a great approach and an excellent plugin... – Adam M. Gamboa G. Aug 6 '17 at 2:21

6

following this link:

How To: Eclipse Maven install build jar with dependencies

i found out that this is not workable solution because the class loader doesn't load jars from within jars, so i think that i will unpack the dependencies inside the jar.

edited May 28 '17 at 12:34 Community ♦


**1** 1


**16k** 104 291 453

3

Here´s how I do it:

```
<plugin>
        <artifactId>maven-assembly-plugin</a
        <version>2.2</version>
        <configuration>
            <appendAssemblyId>false</appendA
            <descriptorRefs>
                <descriptorRef>jar-with-depe
            </descriptorRefs>
            <archive>
                <manifest>
                    <mainClass>com.project.M
                </manifest>
            </archive>
        </configuration>
    </plugin>
```

And then I just run:

```
mvn assembly:assembly
```

answered Aug 1 '12 at 12:36


Eduardo Andrade
**791** 5 10

1 that's not what i want, please read the question carefully. – Mahmoud Saleh Aug 1 '12 at 12:56

1 Note that you should always do a compile before hand because `assembly` will just put whatever is in "target/classes" in the JAR. This will ensure that the JAR includes any changes you recently made to the source code. So, you should do something like: `mvn clean compile assembly:assembly` . – naXa Sep 17 '14 at 10:30

2

I found this answer to the question:

http://padcom13.blogspot.co.uk/2011/10/creating-standalone-applications-with.html

Not only do you get the dependent lib files in a lib folder, you also get a bin director with both a unix and a dos executable.

The executable ultimately calls java with a -cp argument that lists all of your dependent libs too.

The whole lot sits in an appasembly folder inside the target folder. Epic.

========== Yes I know this is an old thread, but it's still coming high on search results so I thought it might help someone like me.

answered Apr 21 '15 at 16:10

user4815755

31 1

1

This is clearly a classpath problem. Take into consideration that the classpath must change a bit when you run your program outside the IDE. This is because the IDE loads the other JARs relative to the root folder of your project, while in the case of the final JAR this is usually not true.

What I like to do in these situations is build the JAR manually. It takes me at most 5 minutes and it always solves the problem. I do not suggest you do this. Find a way to use Maven, that's its purpose.

answered Aug 1 '12 at 11:56

Radu Murzea
**8,248** 9 36 64

what do you mean by build jar manually ?
– Mahmoud Saleh Aug 1 '12 at 11:58

@SoboLAN Building the JAR manually is not a solution here. The intention is to use Maven, which is the exact opposite of "manual"!
– Duncan Jones Aug 1 '12 at 11:59

@DuncanJones You are totally right. I do suggest he uses Maven to do it. However, I have no experience with it and didn't know exactly what solution to recommend. I edited my answer to reflect this. – Radu Murzea Aug 1 '12 at 12:01

**protected** by Community ♦ Dec 4 '15 at 3:51

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?