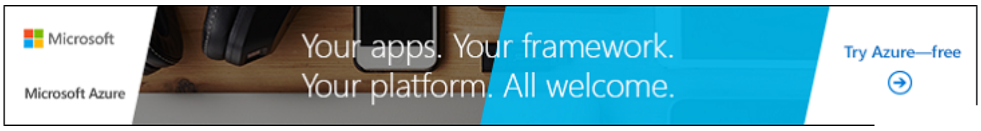
Can I use the path to a Maven dependency as a property?



I have a maven dependency in my pom.xml as such:

```
<dependency>
    <groupId>com.foo</groupId>
    <artifactId>Bar</artifactId>
    <version>1.2.3</version>
</dependency>
```

And I would like to use the system path to the binary as a property (so I can pass it to an external process that is kicked off by maven). I can do this in an awkward way:

```
<properties>
    <my.lib>${settings.localRepository}/com/foo/Bar/1.2.3/Bar.jar</my.lib>
</properties>
```

But I would really like to use a more standard mechanism, such as:

```
<properties>
    <my.lib>${com.foo:Bar:1.2.3}</my.lib>
</properties>
```

I something like that possible?

maven-2

asked Mar 1 '10 at 23:25
codefinger
2,779 4 18 28

I am a bit confused: if you want to refer `Bar.jar` as a system library, you need to specify `<scope>system</scope><systemPath>${my.lib}</systemPath>` but it seems you want to use `${my.lib}` somewhere else. Show the complete example of how you want to use `${my.lib}` ... – dma_k Mar 1 '10 at 23:40

1 @dma_k The OP wants to pass the physical path to a dependency to an external process triggered by maven. – Pascal Thivent Mar 1 '10 at 23:44

5 Answers

Assuming that the `com.foo:Bar:jar:1.2.3` artifact is declared as dependency in your POM, the following property returns the path to the jar in the local repository:

```
${maven.dependency.com.foo.Bar.jar.path}
```

Update: Here is a simple POM demonstrating this:

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.stackoverflow</groupId>
  <artifactId>q2359872</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>q2359872</name>
  <properties>
    <my.lib>${maven.dependency.junit.junit.jar.path}</my.lib>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-antrun-plugin</artifactId>
        <executions>
          <execution>
            <phase>process-resources</phase>
            <configuration>
              <tasks>
                <echo>${my.lib}</echo>
              </tasks>
            </configuration>
            <goals>
              <goal>run</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

Running `mvn process-resources` produces the following output:

```
$ mvn process-resources
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building q2359872
[INFO]    task-segment: [process-resources]
[INFO] ------------------------------------------------------------------------
[INFO] [resources:resources {execution: default-resources}]
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/home/pascal/Projects/stackoverflow/q2359872/src/main/resources
[INFO] [antrun:run {execution: default}]
[INFO] Executing tasks
     [echo] /home/pascal/.m2/repository/junit/junit/3.8.1/junit-3.8.1.jar
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 7 seconds
[INFO] Finished at: Tue Mar 02 14:41:32 CET 2010
[INFO] Final Memory: 7M/68M
[INFO] ------------------------------------------------------------------------
```

edited Mar 2 '10 at 13:42 answered Mar 1 '10 at 23:41

Pascal Thivent
371k 64 768 958

I cannot prove this feature works in Maven. That only works for `maven-antrun-plugin` (see jira.codehaus.org/browse/MANTRUN-110). Please, provide a complete pom example, as I suppose, you refer not `<project><properties>` but some other properties. – dma_k Mar 2 '10 at 13:18

@dma_k The Jira issue you are mentioning doesn't show anything except that there was a bug in the antrun documentation. Now, feel free to test this solution yourself. And BTW, I always test my answers :) – Pascal Thivent Mar 2 '10 at 13:53

@Pascal Thanks for the update! I fully trust you, that it works at your site :) My question was: is it supposed to work in combination with `maven-antrun-plugin` . And you show this in your example, great! And from example I see that this is `maven-antrun-plugin` -specific feature, i.e. if I want to substitute `${my.lib}` variable for resources (without using any additional pugin) - I cannot do it, right? – dma_k Mar 2 '10 at 16:05

@dma_k Not sure it wouldn't work outside antrun (I'm just echoing ${my.lib} after all) but I would have to test resources filtering (sometimes, the way property resolution works is a bit obscure for me). – Pascal Thivent Mar 2 '10 at 16:32

4 @dma_k I did a test and the property is not available during filtering. I don't know if it's the same situation as in stackoverflow.com/questions/2246524/… i.e. if the expression is not available during filtering because filtering and interpolation don't share the same algorithm or if it's an antrun property. – Pascal Thivent Mar 2 '10 at 22:54

Here is a correct implementation which can be used anywhere in a pom

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.stackoverflow</groupId>
    <artifactId>q2359872</artifactId>
    <version>2.0-SNAPSHOT</version>
    <name>q2359872</name>

    <properties>
        <!-- Must be listed in the dependencies section otherwise it will be null. -->
        <my.lib>${org.jmockit:jmockit:jar}</my.lib>
    </properties>
    <dependencies>
        <dependency>
```

```
            <groupId>org.jmockit</groupId>
            <artifactId>jmockit</artifactId>
            <version>1.11</version>
        </dependency>
    </dependencies>
    <build>
        <defaultGoal>generate-sources</defaultGoal>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <version>2.3</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>properties</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <!-- Example usage: -->
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.2</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>exec</goal>
                        </goals>
                        <phase>generate-sources</phase>
                    </execution>
                </executions>
                <configuration>
                    <executable>echo</executable>
                    <arguments>
                        <argument>path to jar=</argument>
                        <argument>${org.jmockit:jmockit:jar}</argument>
                        <argument>my.lib=</argument>
                        <argument>${my.lib}</argument>
                    </arguments>
                </configuration>
            </plugin>
            <!-- end of Example usage -->
        </plugins>
    </build>
</project>
```

And the output is...

```
jpyeron@black /projects/wkspc/tmp/foo
$ /cygdrive/c/programs.x86_64/apache-software-foundation/apache-maven-3.1.1/bin/mvn
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building q2359872 2.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-dependency-plugin:2.3:properties (default) @ q2359872 ---
[INFO]
[INFO] --- exec-maven-plugin:1.2:exec (default) @ q2359872 ---
path to jar= C:\Documents and
Settings\jpyeron\.m2\repository\org\jmockit\jmockit\1.11\jmockit-1.11.jar my.lib=
C:\Documents and Settings\jpyeron\.m2\repository\org\jmockit\jmockit\1.11\jmockit-1.11.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.032s
[INFO] Finished at: Wed Sep 17 12:07:18 EDT 2014
[INFO] Final Memory: 10M/153M
[INFO] ------------------------------------------------------------------------
```

edited Nov 2 '15 at 12:40
DanR
28 6

answered Aug 3 '11 at 23:32
Jason Pyeron
744 8 19

This worked perfectly for me. Thanks for the post! – kurzweil4 Sep 15 '14 at 6:47

It might be less confusing if your example used a different dependency. For those interested, if you needed the jar for JMockit, your property would be like this: <my.lib>${org.jmockit:jmockit:jar}</my.lib> That is to say: ${groupId:artifactId:jar} – kurzweil4 Sep 15 '14 at 6:56

good point, edited. – Jason Pyeron Sep 17 '14 at 16:12

1 This should be the accepted answer. The other one is specific to the ant plugin. – Bogdan Calmac Nov 21 '15 at 3:22

Jason, please mention that the key here is to use the maven-dependency-plugin. Initially I just used the ${org.jmockit:jmockit:jar} syntax and it took a long time to realize what the problem was. – Bogdan Calmac Nov 21 '15 at 3:25

There is a plugin which might be what you are looking for... bitstrings.org (home).

edited Feb 19 at 11:14

answered Nov 21 '11 at 3:30

Sergey Brunov
6,396 1 20 44

gqman69
91 1 1

You need to write a new maven plugin that sets a property value to the fully-resolved pathname of a dependency. The maven-dependency-plugin won't do that for you.

It will *copy* your dependency somewhere and then you can refer to it by that pathname.

answered Mar 1 '10 at 23:53
bmargulies
60.7k 23 109 216

If none of the upper work, you can always use gmaven to agressively dive into `MavenProject` object and get your artifact infos. In my case, I had the following artifact declared in a profile :

```
            <!-- Neo4J connector. This dependency is scoped to be usable by maven-exec-
plugin
                which installs it in Glassfish -->
            <dependency>
                <groupId>com.netoprise</groupId>
                <artifactId>neo4j-connector</artifactId>
                <version>${neo4j.connector.version}</version>
                <type>rar</type>
                <!-- Set in test scope to avoid release issues -->
                <scope>test</scope>
            </dependency>
```

To get its path and put it in a maven property, I wrote the following gmaven script :

```
                <!-- Small script used to build maven property for neo4j-connector path --
>
                <plugin>
                    <groupId>org.codehaus.gmaven</groupId>
                    <artifactId>gmaven-plugin</artifactId>
                    <version>1.3</version>
                    <executions>
                        <execution>
                            <id>get-neo4j-connector-rar-path</id>
                            <phase>validate</phase>
                            <goals>
                                <goal>execute</goal>
                            </goals>
                            <configuration>
                                <source>
                                    <![CDATA[
println "initial value of neo4j.connector.rarPath is
\""+project.properties['neo4j.connector.rarPath']+"\""

// Duplicate model in a Mavenproject, allowing me to get associated artifact
// So sad I can't get the embdder object

// More info here : http://maven.apache.org/ref/3.0.3/maven-
core/apidocs/org/apache/maven/project/MavenProject.html
def mavenProject = new org.apache.maven.project.MavenProject(project)

// More infos on Artifact there : http://maven.apache.org/ref/3.0.3/maven-
artifact/apidocs/org/apache/maven/artifact/Artifact.html
def neo4jConnector = mavenProject.getArtifacts().find { artifact ->
artifact.getArtifactId()=='neo4j-connector' }
// Now resolve dependency to produce an artifact
// notice maven property interpolation doesn't do toString, so we have to do it ourselves
project.properties['neo4j.connector.rarPath'] = neo4jConnector.getFile().getAbsolutePath()

println "usable neoj4Connector can be found at
"+project.properties['neo4j.connector.rarPath']

                                    ]]>
                                </source>
                            </configuration>
                        </execution>
                    </executions>
                </plugin>
```

It's some kind of brute-force method, but it DO work far better than the previous solutions I've seen there.

answered Aug 4 '11 at 12:44
Riduidel
15.3k 4 39 106