



## 3 maneras de agregar un tarro local al proyecto de Maven

MAVEN 28 DE JUNIO DE 2016 11 COMENTARIOS



### Tabla de contenidos [ [ocultar](#) ]

- 1 Introducción
- 2 1- Instale manualmente el JAR en su repositorio local de Maven
- 3 2- Agregando directamente la dependencia como alcance del sistema.
- 4 3- Creando un repositorio local diferente de Maven
- 5 4- Usando el gestor de repositorios Nexus
- 6 Referencias

## Introducción

Es posible que deba agregar un JAR personalizado como una dependencia a su proyecto de Maven. Este tutorial muestra 3 formas de hacerlo:

1. Instale manualmente el JAR en su repositorio local de Maven
2. Añadiendo la dependencia como ámbito del sistema.
3. Creando un repositorio local diferente de Maven
4. Usando un gestor de repositorios Nexus

### PUBLICACIONES RECIENTES ABOUT MAVEN



#### MAVEN

Filtra y renombra recursos con Maven  
17 DIC, 2016



#### MAVEN

3 maneras de agregar un tarro local al proyecto de Maven  
28 DE JUNIO DE 2016



#### MAVEN

3 formas de resolver el error de Maven: No se proporciona ningún compilador en este entorno.  
7 JUN, 2016



#### MAVEN

3 formas de resolver la compilación de errores fatales: lanzamiento de destino no válido en la compilación de Maven  
2 DE JUNIO DE 2016



#### MAVEN

3 maneras de resolver java.lang.OutOfMemoryError: PermGen espacio en Maven build  
30 DE MAYO DE 2016

I am one with the Elastic Stack

[Learn More](#)

[elastic.co/elk-stack](http://elastic.co/elk-stack)



# 1- Instale manualmente el JAR en su repositorio local de Maven

La primera solución es agregar manualmente el JAR en su repositorio local de Maven utilizando el objetivo de **instalación de Maven : archivo de instalación** . El uso del plugin es muy simple como a continuación:

```
1 | mvn install:install-file -Dfile=<path-to-file>
```

Tenga en cuenta que no especificamos *groupId* , *artifactId* , *versión* y *empaquetado* del JAR para instalar. De hecho, desde la versión 2.5 de **Maven-install-plugin** , esta información se puede tomar de un **pomFile** opcionalmente especificado

Esta información también se puede dar en la línea de comando:

```
1 | mvn install:install-file -Dfile=<path-to-file> -DgroupId=<gr
```

Dónde:

- **<path-to-file>**: ruta al JAR para instalar
- **<group-id>**: ID de grupo del JAR para instalar
- **<artifact-id>**: Id. de artefacto del JAR para instalar
- **<version>**: Versión del JAR

Por ejemplo:

```
1 | mvn install:install-file -Dfile=C:\dev\app.jar -DgroupId=com
```

Luego puede agregar la dependencia a su proyecto de Maven agregando esas líneas a su archivo pom.xml:

```
1 | <dependency>
2 |     <groupId>com.roufid.tutorials</groupId>
3 |     <artifactId>example-app</artifactId>
4 |     <version>1.0</version>
5 | </dependency>
```

Esta solución puede ser muy costosa. Por qué ? Debe tener en cuenta que el día que cambie su repositorio local de Maven deberá reinstalar el JAR. O nuevamente, si hay muchas personas trabajando en el proyecto, cada una debe instalar el JAR en su repositorio local. La portabilidad del proyecto debe ser tenida en cuenta.

Otra solución es utilizar el **complemento de instalación de maven** en su **pom.xml**, que instalará el tarro durante la fase de **"inicialización" de Maven** . Para hacer esto, debe especificar la ubicación del jar que desea instalar. La mejor manera es colocar el JAR en una carpeta creada en la raíz del proyecto (*en el mismo directorio que el archivo pom.xml*)

Consideremos que el jar está ubicado en **<PROJECT\_ROOT\_FOLDER> /lib/app.jar**. Debajo de la configuración de maven-install-plugin:

```

1  <plugin>
2    <groupId>org.apache.maven.plugins</groupId>
3    <artifactId>maven-install-plugin</artifactId>
4    <version>2.5</version>
5    <executions>
6      <execution>
7        <phase>initialize</phase>
8        <goals>
9          <goal>install-file</goal>
10       </goals>
11       <configuration>
12         <groupId>com.roufid.tutorials</groupId>
13         <artifactId>example-app</artifactId>
14         <version>1.0</version>
15         <packaging>jar</packaging>
16         <file>${basedir}/lib/app.jar</file>
17       </configuration>
18     </execution>
19   </executions>
20 </plugin>

```

“ **`${basedir}`**

Es posible que encuentre un error al agregar las líneas anteriores, agregue el siguiente complemento a su proyecto para permitir el mapeo del ciclo de vida:

```

1  <pluginManagement>
2    <plugins>
3      <!--This plugin's configuration is used to store Ec
4        It has no influence on the Maven build itself.
5      <plugin>
6        <groupId>org.eclipse.m2e</groupId>
7        <artifactId>lifecycle-mapping</artifactId>
8        <version>1.0.0</version>
9        <configuration>
10         <lifecycleMappingMetadata>
11           <pluginExecutions>
12             <pluginExecution>
13               <pluginExecutionFilter>
14                 <groupId>org.codehaus.mojo<
15                 <artifactId>aspectj-maven-p
16                 <versionRange>[1.0,)</versi
17               <goals>
18                 <goal>test-compile</goa
19                 <goal>compile</goal>
20               </goals>
21             </pluginExecutionFilter>
22             <action>
23               <execute />
24             </action>
25           </pluginExecution>

```

```

26         <pluginExecution>
27             <pluginExecutionFilter>
28                 <groupId>
29                     org.apache.maven.plugin
30                 </groupId>
31                 <artifactId>
32                     maven-install-plugin
33                 </artifactId>
34                 <versionRange>
35                     [2.5,)
36                 </versionRange>
37                 <goals>
38                     <goal>install-file</goal>
39                 </goals>
40             </pluginExecutionFilter>
41             <action>
42                 <execute>
43                     <runOnIncremental>>false
44                 </execute>
45             </action>
46         </pluginExecution>
47     </pluginExecutions>
48 </lifecycleMappingMetadata>
49 </configuration>
50 </plugin>
51 </plugins>
52 </pluginManagement>

```

**I am one with the Elastic Stack**

[Learn More](#)

[elastic.co/elk-stack](https://elastic.co/elk-stack)



## 2- Agregando directamente la dependencia como alcance del sistema.

Otra solución - solución sucia - es agregar la dependencia como alcance del **sistema** y referirse a ella por su ruta completa. Tenga en cuenta que el JAR se encuentra en **<PROJECT\_ROOT\_FOLDER> / lib**. Luego agregue la dependencia en su archivo pom.xml de la siguiente manera:

```

1 <dependency>
2     <groupId>com.roufid.tutorials</groupId>
3     <artifactId>example-app</artifactId>
4     <version>1.0</version>
5     <scope>system</scope>
6     <systemPath>${basedir}/lib/yourJar.jar</systemPath>
7 </dependency>

```



**`${basedir}`** representa el directorio que contiene pom.xml.

### 3- Creando un repositorio local diferente de Maven

La tercera solución es bastante similar a la primera, la diferencia radica en el hecho de que los JAR se instalarán en un repositorio local diferente de Maven.

Consideremos que el nuevo repositorio local de Maven se llama **"maven-repository"** y se encuentra en **`${basedir}`** (el directorio que contiene pom.xml). Lo primero que debe hacer es **implementar** los JAR locales en el nuevo repositorio local de Maven como se muestra a continuación:

```
1 | mvn deploy:deploy-file -Dfile=<path-to-file> -DgroupId=<group
```

Normalmente, el **despliegue de Maven : `deploy-file`** instala el artefacto en un repositorio remoto, pero en nuestro caso el repositorio se encuentra en la máquina local.

Después de instalar los archivos JAR, debe agregar el repositorio a su archivo pom.xml:

```
1 | <repositories>
2 |   <repository>
3 |     <id>maven-repository</id>
4 |     <url>file:///${project.basedir}/maven-repository</url>
5 |   </repository>
6 | </repositories>
```

Luego puedes agregar la dependencia en tu pom.xml

```
1 | <dependency>
2 |   <groupId>com.roufid.tutorials</groupId>
3 |   <artifactId>example-app</artifactId>
4 |   <version>1.0</version>
5 | </dependency>
```



### 4- Usando el gestor de repositorios Nexus

La mejor solución es usar un administrador de repositorio Nexus que contendrá todos sus archivos JAR y usted lo usará como repositorio para descargar la dependencia.

**Este libro del sitio oficial de Nexus** le mostrará cómo instalar y usar el administrador de repositorio Nexus.

## Refrences

- **Maven: instalando JARs de terceros**
- **Ciclo de vida de Maven**
- **Despliegue de Maven: objetivo del archivo de despliegue**
- **Instalación de Maven: objetivo del archivo de instalación**
- **Nexo**



Difundir la palabra:



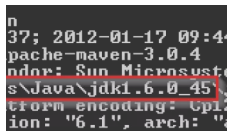
### Relacionado



Ningún atributo  
manifiesto principal, en  
jarra

21 de marzo de 2016

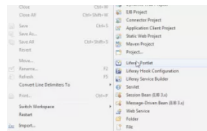
En "java"



3 formas de resolver la  
compilación de errores  
fatales: lanzamiento de  
destino no válido en la  
compilación de Maven

2 de junio de 2016

En "Maven"



Crear y desplegar JSF  
Liferay Portlet

23 de febrero de 2016

En "Liferay"

Etiquetas: Maven



**Radouane ROUFID**

Ingeniero informatico

**Este Sencillo Truco "Derrite" La Grasa  
Abdominal Durante La Noche (Pruébala Ya)**

Entra Para Saber Más

[Learn More](#)

Sponsored by Phendora

[Report ad](#)

11 Comments - roufid.com

Javier Martín Alen

**Important Update**

When you log in with Disqus, we process personal data to facilitate your authentication and posting of comments. We also store the comments you post and those comments are immediately viewable and searchable by anyone around the world.

- ☐ I agree to Disqus' Terms of Service
- ☐ I agree to Disqus' processing of email and IP address, and the use of cookies, to facilitate my authentication and posting of comments, explained further in the Privacy Policy
- ☐ I agree to additional processing of my information, including first and third party cookies, for personalized content and advertising as outlined in our Data Sharing Policy

Proceed



You have a missing package.

^ | v • Reply • Share ›



**Alin** → Radouane Roufid • a month ago

But that I've run

```
"mvn deploy:deploy-file -Dfile=./libs/my-jar.jar -
DgroupId=com.something -DartifactId=my-artifact -Dversion=1
-Dpackaging=jar -Durl=file:./libs/ -DrepositoryId=maven-
repository -DupdateReleaseInfo=true",
```

then I add the repository in my pom.xml

```
<repositories>
<repository>
<id>maven-repository</id>
<url>file:///${project.basedir}/libs</url>
</repository>
</repositories>
```

and then add the dependency

```
<dependency>
```

[see more](#)

^ | v • Reply • Share ›



**Radouane Roufid** Mod → Alin • a month ago

I think you have a compiling problem.

Your jar is referencing a package that does not exists =>  
**User.java**: [3,40] package com.roufid.tutorials does not exist

See the third line of your class : **User.java**

^ | v • Reply • Share ›



**Alin** → Radouane Roufid • a month ago



At this line I'm trying to import my jar

^ | v • Reply • Share ›



**Radouane Roufid** Mod → Alin • a month ago

It seems that you don't have the same problem. What's your error now ?


^ | v • Reply • Share ›





**Alin** → Radouane Roufid • a month ago


Error: (3, 20) java: cannot find symbol  
symbol: class something

location: package com  
^ | v • Reply • Share ›

 **Radouane Roufid** Mod ➔ Alin • a month ago  
Your jar does not contain the Class something. Verify that the class is within the jar. Verfiy the log when making the jar.  
^ | v • Reply • Share ›

 **Alin** ➔ Radouane Roufid • a month ago  
This was the problem. Thanks  
^ | v • Reply • Share ›

 **Vishal Zanzrukia** • 3 months ago  
How can we install multiple jars using first option?  
^ | v • Reply • Share ›

 **Lewis Florez R** • 6 months ago  
second option is not working for me, any advice?

**Este Sencillo Truco "Derrite" La Grasa Abdominal Durante La Noche (Pruébala Ya)**

Entra Para Saber Más

[Learn More](#)

Sponsored by **Phendora**

[Report ad](#)

QUE CALOR ?



**PRIMAVERA**  
Opcional Spring @value  
15 DE MAYO DE 2017

**PRIMAVERA**  
Instala Spring IDE en Eclipse  
4 DE ENERO DE 2017

**SPRING / SPRING MVC**  
Defining a proxy in Spring RestTemplate  
28 DEC, 2016

RECENT COMMENTS

- Radouane Roufid on Java LDAP SSL authentication
- KarthiK E on Java LDAP SSL authentication
- Radouane Roufid on 3 ways to solve the Maven error : No compiler is provided in this environment.
- Radouane Roufid on 3 ways to solve the Maven error : No compiler is provided in this environment.

TAGS

Ajax Apache-Commons-IO  
BufferedReader BufferedWriter Database DB2  
DerbyDB Eclipse faces-config.xml file  
Files hibernate IPC Java JavaScript  
JPA JQuery JSF JVM Liferay liferay-  
display.xml liferay-hook.xml liferay-plugin-  
package.properties liferay-portlet.xml Liferay  
IDE Liferay m2e Liferay plugin M2Eclipse  
MANIFEST.MF Maven MySQL Oracle  
portal-ext.properties Portlet  
portlet.xml PostgreSQL PrintWriter  
Propiedades Público Parámetro de  
procesamiento Escáner spring spring-  
boot spring-data-jpa Tomcat web.xml



💬 Misha on 3 ways to solve the Maven  
error : No compiler is provided in this  
environment.

---

- 
- Casa
  - Liferay
  - Java
  - Primavera
  - Hibernar
  - Maven
  - Contacto
- 



Roufid © 2018. Todos los derechos reservados.