# Maven

Maven es una herramienta para proyectos Java que facilita la administración de dependencias promoviendo el principio de integración continua sobre la automatización de la construcción y empaquetado de aplicaciones.

## Instalar Maven

Windows: seguir tutorial: https://www.mkyong.com/maven/how-to-install-maven-in-windows/
Linux: apt install maven

## Verificar instalacion:

Ejecutar en línea de comandos: mvn -v

Debe mostrar algo como:

```
MacBook-Pro-de-Lucas:~ lucasmufato$ mvn -v
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-24T15:41:47-03:00)
Maven home: /usr/local/Cellar/maven/3.6.0/libexec
Java version: 1.8.0_201, vendor: Oracle Corporation, runtime: /Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/jre
Default locale: es_AR, platform encoding: UTF-8
OS name: "mac os x", version: "10.14.3", arch: "x86_64", family: "mac"
```

## Crear proyecto:

1. Abrir una terminal en la carpeta y ejecutar:
   *Reemplazar lucasmufato por su nombre y apellido

```
mvn -B archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DgroupId=com.lucasmufato.prueba \
  -DartifactId=my-prueba
```

   *Si es la primer vez que se ejecuta va a descargar varios componentes.

2. Si se ejecuta correctamente por la línea de comandos mostrará algo como:

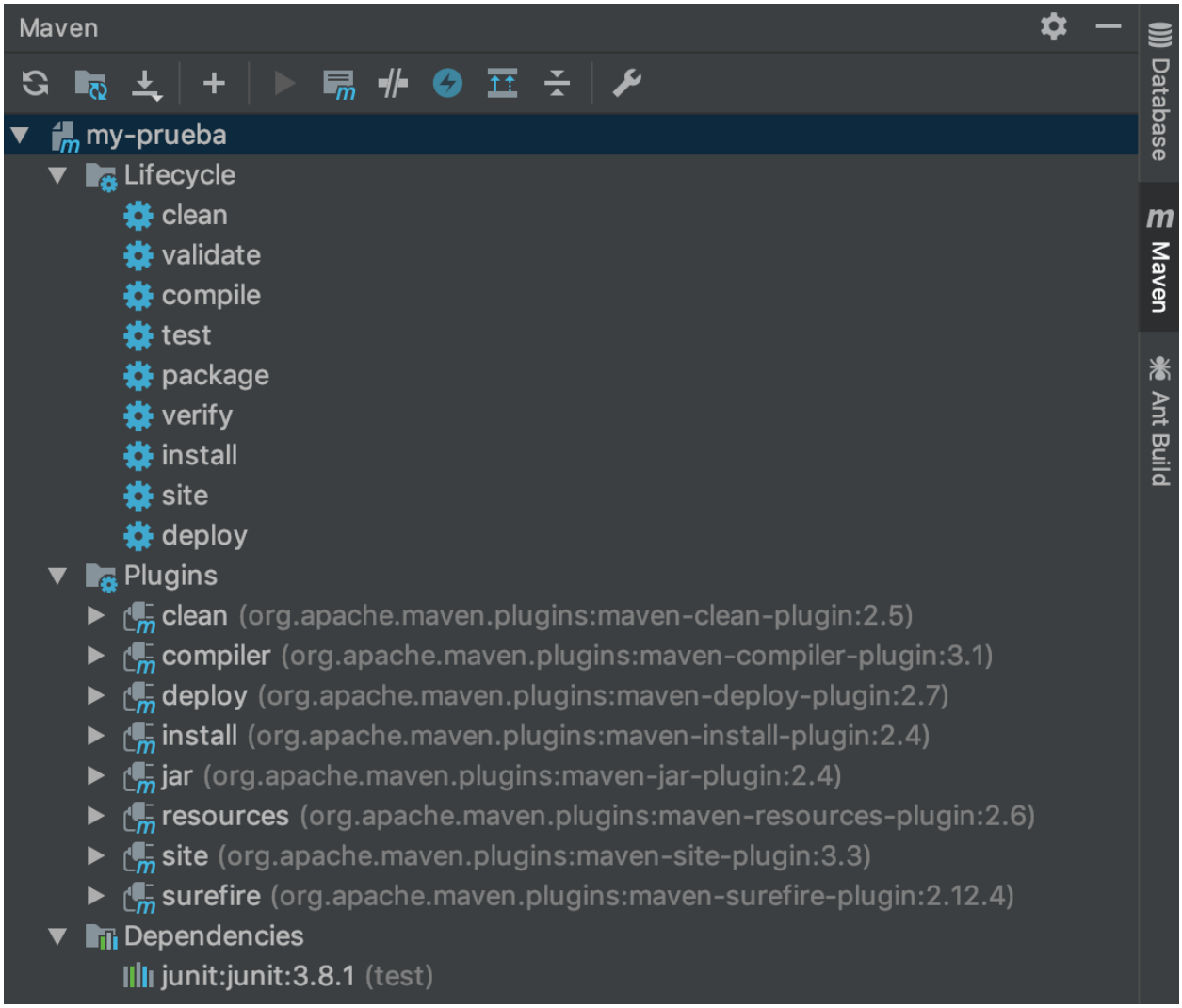
```
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: /Users/lucasmufato/Movies
[INFO] Parameter: package, Value: com.lucasmufato.prueba
[INFO] Parameter: groupId, Value: com.lucasmufato.prueba
[INFO] Parameter: artifactId, Value: my-prueba
[INFO] Parameter: packageName, Value: com.lucasmufato.prueba
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: /Users/lucasmufato/Movies/my-prueba
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  22.228 s
[INFO] Finished at: 2019-03-25T21:15:58-03:00
[INFO] ------------------------------------------------------------------------
```

3. Maven habrá creado una carpeta con el nombre igual al parametro *artefactId*. Dentro del mismo existirá un archivo “pom.xml” y una carpeta “src”. Abrir el archivo “pom.xml” el mismo debería estar inicializado con la siguiente configuración:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.lucasmufato.prueba</groupId>
  <artifactId>my-prueba</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-prueba</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

4. Dentro de la carpeta “src” se encontrará todo el código java. El mismo se encuentra divido en 2 carpetas principales. “Main” donde estarán todas las clases propias de la aplicación y “test” donde se encontrarán las clases necesarias para el testing.

5. Al abrir el proyecto desde el IDE este debe reconocer el archivo pom.xml y darse cuenta que es un proyecto maven. En cuyo caso descargar automáticamente los paquetes y mostrará información (dependiendo el IDE). Ejemplo en IntelliJ:

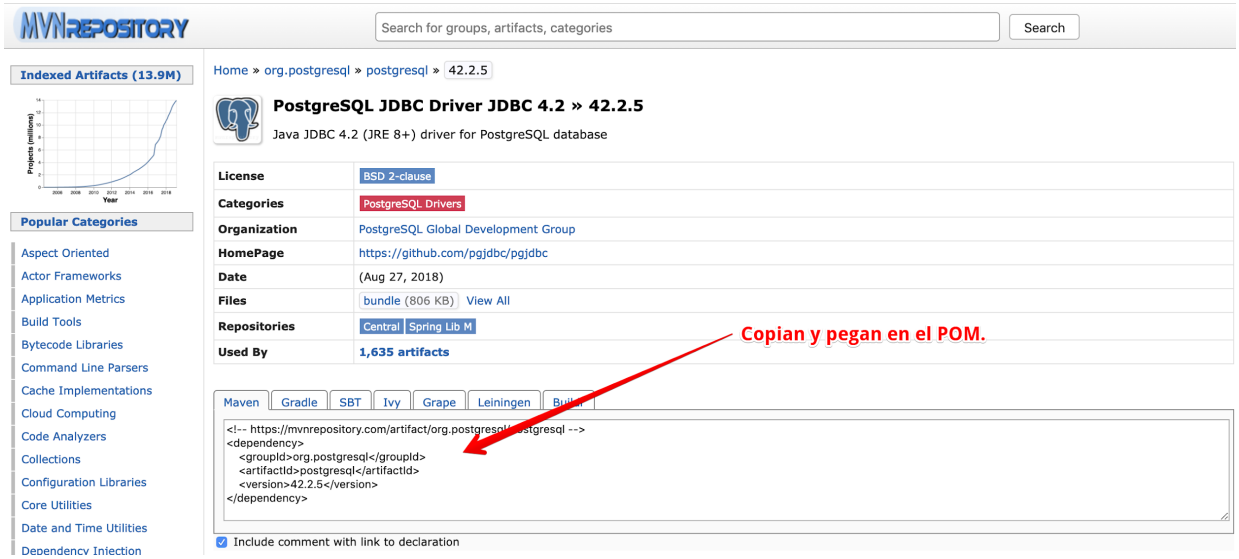



6. Ejecutar las clases App.java. Debe mostrar un “hola mundo”
7. Ejecutar en la línea de comandos: mvn test.
   Este comando ejecutara todos los test dentro de ‘test’( que loco! ). Debe informar que un único test se ejecutó correctamente. Ejemplo:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.lucasmufato.prueba.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.260 s
[INFO] Finished at: 2019-03-25T22:31:55-03:00
[INFO] ------------------------------------------------------------------------
```

Felicidades! Tenes maven instalado y listo para usar!

En la materia se utilizará para facilitar la gestión de las librerías y corrección de TPs mediante test automáticos!
Las librerias se pueden buscar en https://search.maven.org/ o https://mvnrepository.com
Una vez encontrada la que se quiera utilizar la agregan al POM dentro de la etiqueta <dependencies>. EJ:



Una vez agregada la librería al POM, el IDE la descargara automaticamente o sino tambien se puede ejecutar el comando de maven: ***mvn dependency:resolve***

Las librerías a utilizar en la materia son:
*No se van a utilizar todas juntas. Cada trabajo práctico o punto indicara que utilizar.

```
<!--
https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-ap
i -->
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>5.4.1</version>
      <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql
-->
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.2.5</version>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.eclipse.persistence/org.eclips
e.persistence.jpa -->
    <dependency>
      <groupId>org.eclipse.persistence</groupId>
      <artifactId>org.eclipse.persistence.jpa</artifactId>
      <version>2.7.4</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <version>1.4.199</version>
      <scope>test</scope>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
```

```
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.4.2.Final</version>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.firebirdsql.jdbc/jaybird-jdk18
-->
    <dependency>
      <groupId>org.firebirdsql.jdbc</groupId>
      <artifactId>jaybird-jdk18</artifactId>
      <version>3.0.5</version>
      <scope>test</scope>
    </dependency>
```

## Guia de Aprendizaje/Lectura.

1. Profundizar en la definición de qué es maven y que rol cumple.
2. Que son los Maven Repositories?
3. Donde guarda Maven las librerías descargadas?
4. Que es el LifeCycle?
5. Defina cada Phase del default LifeCycle en un renglon.
6. Que hace el LifeCycle *clean*?
7. Que es el POM?
8. Que es un Multi-module proyect?
9. Que es un snapshoot?
10. Que es el scope de una dependencia?
11. Como se integran plugins? Como se pueden utilizar?
12. Que son los Archetypes?
13. Que alternativas hay a maven?(para java)
14. Que usan otro lenguajes con el mismo objetivo?(PHP, Python, Javascript-Node)

Referencias:


- http://maven.apache.org/
- https://mvnrepository.com/
- https://search.maven.org/
- https://www.oracle.com/technetwork/es/articles/java/java-con-maven-2516405-esa.html
- Bibliografia:
    - 2015 - Maven Essentials
    - 2014 - Mastering Apache Maven 3
    - 2011 - Apache Maven 3 Cookbook