**¿Qué es Maven y para qué se utiliza?**



**Introducción**

En el año 2019 **NO** concibo ningún proyecto sin un **gestor de dependencias**, da igual la tecnología que uses. Para Java tenemos Maven, para android Gradle, para js NPM, para php Composer, etc.

Un gestor de dependencias te da la facilidad de poder descargar paquetes a tus proyectos y el solo va controlando las versiones de los mismos. Ya se queda muy atrás eso de descargar un zip e introducir mil ficheros dentro de nuestros proyectos.

Otras de las ventajas es el trabajo en el equipo, todos trabajan con la misma versión de las dependencias que pueda tener nuestros proyectos. Nos olvidamos de que cada miembro del equipo trabaje con versiones distintas de un mismo paquete.

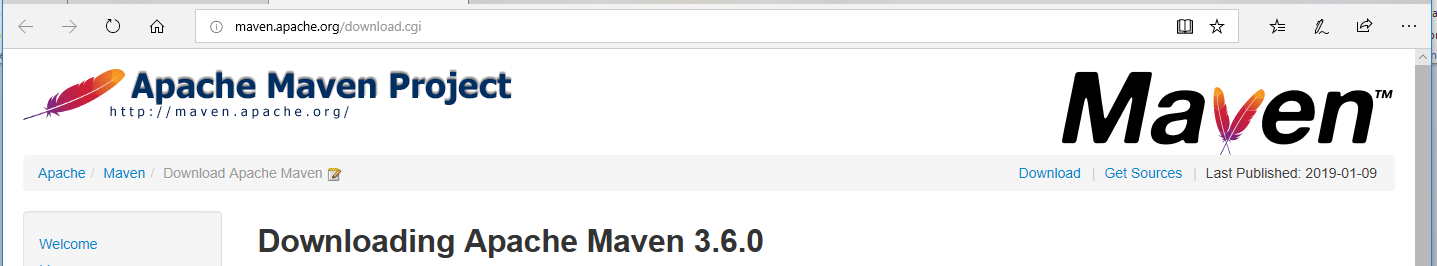
**Que es Maven**

Una de las cosas que más me gusta de Java es la gran cantidad de código que hay escrito allá afuera. Son cientos o quizás miles de librerías que podemos utilizar para múltiples propósitos. Programar no significa inventar la rueda sino aprovechar los recursos que ya existen de la mejor forma para lograr los objetivos propuestos.

Una de las herramientas más útiles a la hora de utilizar librerías de terceros es Maven. **Maven** se utiliza en la gestión y construcción de software. Posee la capacidad de realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Es decir, hace posible la creación de software con dependencias incluidas dentro de la estructura del JAR. Es necesario definir todas las dependencias del proyecto (librerías externas utilizadas) en un fichero propio de todo proyecto Maven, el POM (Project Object Model). Este es un archivo en formato XML que contiene todo lo necesario para que a la hora de generar el fichero ejecutable de nuestra aplicación este contenga todo lo que necesita para su ejecución en su interior.

Sin embargo, la característica más importante de Maven es su capacidad de trabajar en red. Cuando definimos las dependencias de Maven, este sistema se encargará de ubicar las librerías que deseamos utilizar en Maven Central, el cual es un repositorio que contiene cientos de librerías constantemente actualizadas por sus creadores. Maven permite incluso buscar versiones más recientes o más antiguas de un código dado y agregarlas a nuestro proyecto. Todo se hará de forma automática sin que el usuario tenga que hacer nada más que definir las dependencias.

Maven puede ser descargado e instalado desde su [*página* ***oficial***](http://maven.apache.org/download.cgi). Sin embargo el uso de consola para algunos usuarios puede ser un poco tedioso.



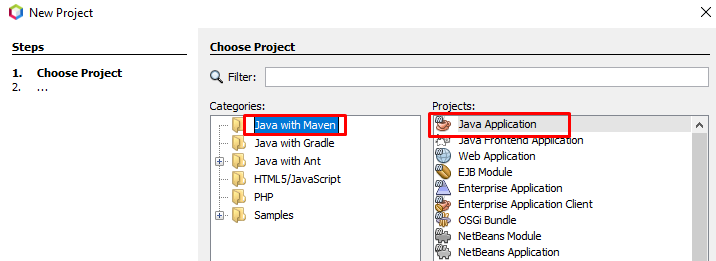
***Netbeans*** posee integración con Maven, es decir, podemos crear proyectos que utilicen Maven aprovechando la interface gráfica de Netbeans. Veamos un ejemplo de cómo crear proyectos Maven, agregar dependencias y descargar las librerías necesarias desde el repositorio remoto.

Un par de cosas que conviene conocer antes de empezar:

* **artefacto**. Es un proyecto que lo gestiona **Maven**y que incluye un fichero llamado **pom.xml**.
* **POM**. Son las siglas de **Project Object Model**. Es un fichero **XML**que contiene la **configuración del artefacto**. Más adelante trabajaremos con él.
* **groupId**. El identificador único para crear nuestro artefacto.

**Creación de un proyecto Maven en Netbeans**

Lo primero que haremos será ir a Netbeans y crear un nuevo proyecto. Debemos tener cuidado de escoger Java with Maven / Java Application a la hora de escoger el tipo de proyecto.



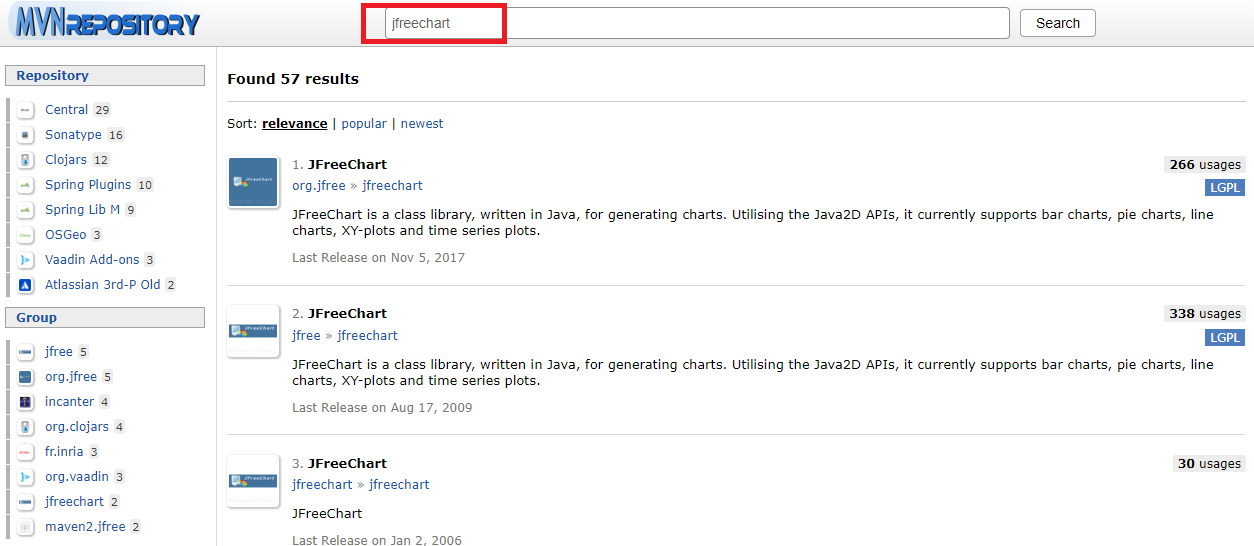
Llamaré mi proyecto **MavenTest**.

En este proyecto voy a agregar las dependencias de JFreeChart. Podemos buscar los repositorios que necesitamos en la siguiente dirección:

[**http://mvnrepository.com/**](http://mvnrepository.com/)

En el caso de JFreeChart podemos encontrar las dependencias en la siguiente dirección:

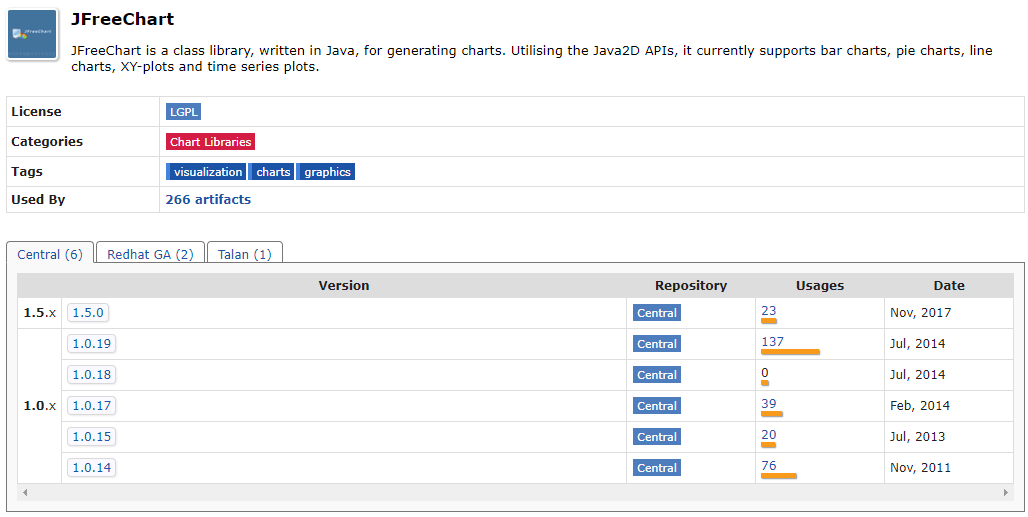
[**http://mvnrepository.com/artifact/org.jfree/jfreechart**](http://mvnrepository.com/artifact/org.jfree/jfreechart)



Yo escogeré la versión 1.0.19 para este ejemplo. Para saber la dependencia lo haremos de la siguiente forma:

Al pulsar sobre el primer elemento que nos sale al buscar jfreechart

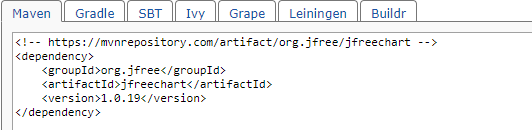
Nos aparecerá todas las versiones de las que disponemos



En este caso vamos a utilizar la versión 1.0.19



Al pulsar sobre la versión, nos aparecerá el siguiente código en XML, que son las dependencias que necesita Maven para utilizar esa librería.



En nuestro caso:

<dependency>

<groupId>org.jfree</groupId>

<artifactId>jfreechart</artifactId>

<version>1.0.19</version>

</dependency>

**Dependencias**

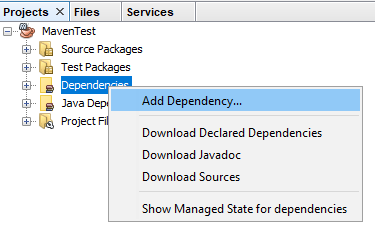
Cuando trabajamos con una aplicación Standar de Java SE, descargamos las librerías (\*.jar) que necesitamos y las colocamos en la carpeta «*Bibliotecas*«, en Maven lo que hacemos es declarar esas librerías en el archivo **POM***( Project Object Model)*

Para utilizar estas dependencias, existen **dos formas de agregar** esto a nuestro proyecto, ya sea a través de la interfaz gráfica de usuario o colocando esta dependencia directamente en el archivo **pom.xml**

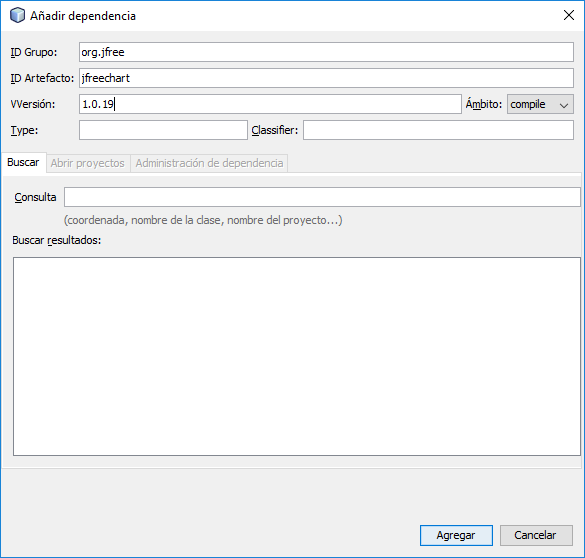
**1ª Forma**

Lo haremos a través de la interfaz gráfica de usuario. El árbol de nuestro proyecto es **ligeramente diferente** al de una aplicación común de Java.

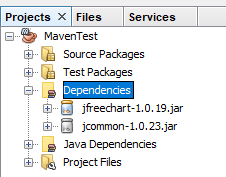
Vemos que en el caso del proyecto Maven tenemos las carpetas Java Dependencies, Dependencies y Project Files. Vamos a darle clic derecho a la carpeta Dependencies y escogeremos la opción Add Dependency.



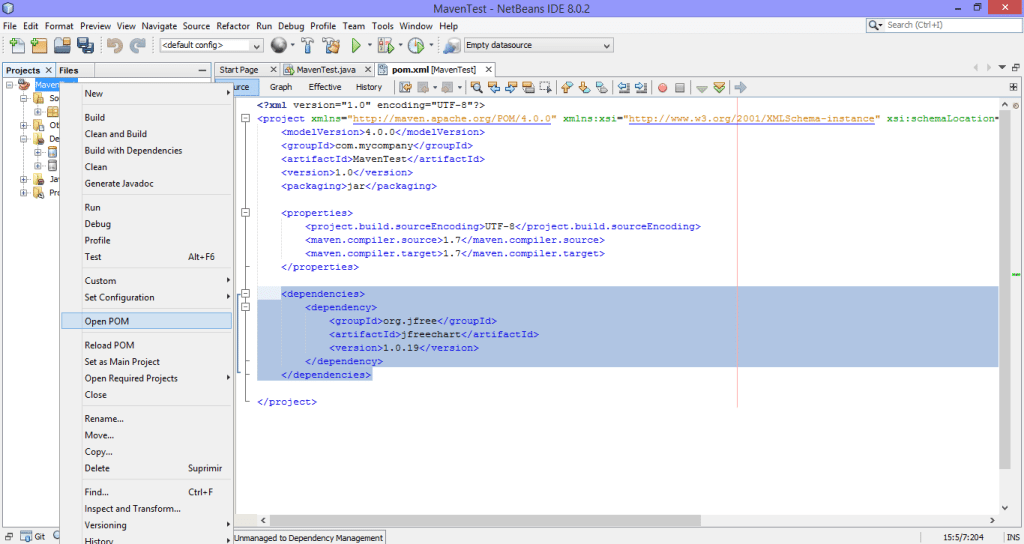
 Nos aparece una ventana donde colocaremos la información de JFreeCharts.



Al hacer clic en “Add” se agregará a nuestro proyecto todo lo que necesitamos para utilizar JFreeCharts. Al final del proceso de descarga tendremos nuestro proyecto con sus respectivas dependencias ya instaladas y listas para utilizar.



Si verificamos el archivo pom.xml (clic derecho sobre el proyecto, Open POM) veremos lo siguiente:



**2ª Forma**

De forma manual basta con copiar el código de dependencias en este archivo y listo. Existen cientos de librerías en los repositorios de Maven. Es posible también utilizar GitHub para hostear repositorios y llamarlos desde Maven. Esta herramienta es sumamente útil y nos facilitará muchísimo la creación de proyectos en Java.

De forma paralela una **copia del repositorio de Maven** comienza a descargarse en tu equipo, al menos la primera vez.

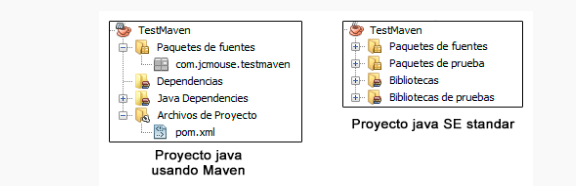
Ya está listo tu proyecto para que puedas agregar más dependencias.

Si es la primera vez que usas Maven, la compilación y ejecución del proyecto se tardara unos minutos (deprendiendo de tu conexión a internet) esto porque Maven debe descargar algunas otras librerías que necesita para funcionar, además de que está creando un repositorio local en tu computadora

http://www.decodigo.com/m/2018/09/java_crear_proyecto_maven_netbeans_05.jpg

**Ejecutables**

La estructura de una aplicación en Netbeans con Maven es **diferente** a la de una aplicación standar java se, y esto es igual con sus carpetas en el disco.



**Usando la dependencia**

La dependencia se usa igual que cualquier otra librería, lo único que cambio en maven, es la forma de hacer referencia a dicha librería.

En la carpeta de fuentes, crea una clase **Ejemplo.java**

También debemos también indicar al proyecto que esta clase, es la clase principal.

A continuación agregamos el siguiente código:

public static void main(String[] args) {

        // TODO code application logic here

        //System.out.println("hola mundo");

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        dataset.setValue(6, "2017", "Jorge");

        dataset.setValue(7, "2018", "Jorge");

        dataset.setValue(8, "2017", "Cristina");

        dataset.setValue(5, "2018", "Cristina");

        dataset.setValue(12, "2017", "Pedro");

        dataset.setValue(9, "2018", "Pedro");

        JFreeChart chart = ChartFactory.createBarChart(

                "Comparación Ventas 2017-2018",

                "Vendedores",

                "Ventas",

                dataset,

                PlotOrientation.HORIZONTAL,

                true,

                false,

                false

        );

        //Mostramos la grafica en pantalla

        ChartFrame frame = new ChartFrame("Ejemplo Grafica de Barras", chart);

        frame.pack();

        frame.setVisible(true);

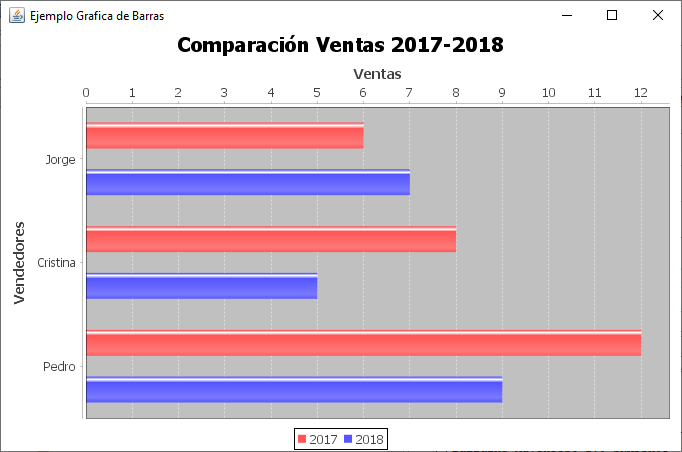
    }

**Ejecución**

Al igual que un proyecto standar java se, **ejecutamos** con clic derecho sobre el proyecto → run

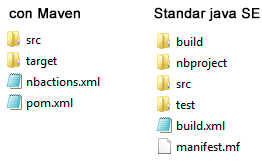
Importante, si es la primera vez que usas *Maven*, la compilación y ejecución del proyecto se tardara unos minutos *(deprendiendo de tu conexión a internet)* esto porque *Maven* debe descargar algunas otras librerías que necesita para funcionar, además de que está creando un repositorio local en tu computadora. De eso hablaremos en otra ocasión.

El resultado en consola de este proyecto es el siguiente:



**Ejecutables**

Como mencionamos en un principio la estructura de una aplicación en Netbeans con Maven es diferente a la de una aplicación standar java se, y esto es igual con sus carpetas en el disco.



En la carpeta SRC se guardan las fuentes del proyecto, mientras en la carpeta TARGET, se encuentran otros archivos referentes a *Maven*, la compilación y los ejecutables *JAR*.

Si hacemos Clic derecho sobre el *proyecto* -> ***Build*** y abrimos la carpeta *target*, veremos que se genero un archivo jar ejecutable “*TestMaven-1.0.1.jar*“, pero este jar no contiene el archivo *manifest* donde se indica la clase principal *Main* y las dependencias de este, si tratamos de ejecutarlo tendremos problemas.

A continuación **dos formas de generar ejecutables** con proyectos con Maven.

**Ejecutable JAR con dependencias**

Abre el archivo **pom.xml** y antes de la etiqueta **</project>** agrega el siguiente código:

<build>

<plugins>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.0.0</version>

<configuration>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<archive>

<manifest>

<mainClass>com.mycompany.maventest.Ejemplo</mainClass>

</manifest>

</archive>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

</plugin>

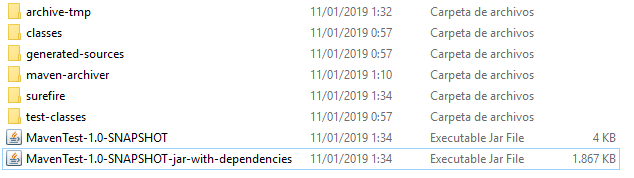
</plugins>

</build>

Donde en **<mainClass>** indicamos los paquetes y clase *Main*.

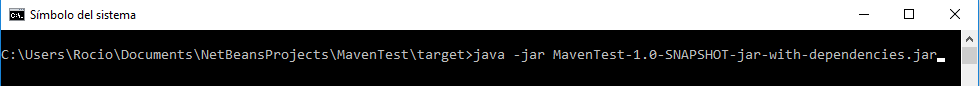
Clic derecho sobre el proyecto -> **Build** para generar el \*.jar

En este caso veremos cómo nuestro jar ha crecido, esto porque además del código de nuestra aplicación (4KB), adjunto las clases de la librería JFreeChart.

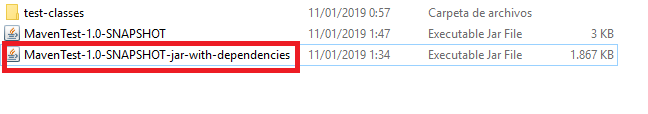


Y para terminar, lo ejecutamos desde consola:

**java -jar MavenTest-1.0-SNAPSHOT-jar-with-dependencies.jar**

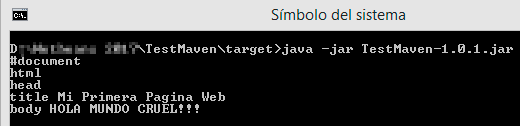


O de una forma más simple haciendo **doble click** sobre el archivo



Generar solo Jar Ejecutable

Nuestra aplicación no cuenta con interfaz por lo que deberemos ejecutarlo desde consola, es decir:



Importante en este caso la librería debería estar en el PATH o en la misma carpeta que nuestro archivo *TestMaven-1.0.1.jar*, caso contrario nos dará error *(java.lang.NoClassDefFoundError: org/………)*.

Por tanto esta forma, **es la peor de las 2** por lo cual he decidido no explicar cómo sería la estructura del archivo pom.xml.

**Conclusión**

Como puedes ver, no es tan complicado crear proyectos **Java**con **Maven**, desde luego existen muchas opciones de configuración para el archivo **pom.xml** que no hemos mencionado.  Sin embargo, la característica principal que te permite descargar dependencias (librerías) y agregarlas a tus proyectos fácilmente es la que usarás con mayor frecuencia.  Al menos ya podrás crear proyectos con este formato y gradualmente irás aprendiendo más opciones para construir aplicaciones y desplegarlas de la manera que tú desees.

Práctica Obligatoria: Maven

**Jsoup** es una libreria Java que proporciona operaciones para trabajar con HTML.

## Utilizando la librería [JSoup Java HTML Parser](https://mvnrepository.com/artifact/org.jsoup/jsoup) y alguna relacionada con JSON , CSV

Crea un proyecto muy sencillo utilizando **Maven** para HACER **WEB SCRAPING** BÁSICO CON JAVA. guarda los datos en UN ARCHIVO **.JSON o csv** PARA POSTERIORMENTE VISUALIZARLOS POR PANTALLA (pOR EJEMPLO EN UN JTABLE).

## ¿Qué es el Web Scraping?

**Web scraping** es la técnica que se utiliza para extraer información de páginas web de forma automática. Consiste en leer el código de una página para obtener datos en bruto y transformarlos en datos estructurados que se pueden guardar en bases de datos u hojas de cálculo para analizar y extraer aquello que nos interesa.  
Esta técnica se puede utilizar para recolectar datos de forma masiva con lo que podríamos decir que es parte de lo que conocemos como **Big Data**.