



### 1.1 ¿Qué es Maven?

Cuando desarrollamos un proyecto sin Maven, nos empiezan a surgir preguntas de como instalo mi proyecto, por qué cada proyecto tiene una estructura distinta, como modifico una versión de una librería, como meto una nueva librería, como lo pruebo, nos van surgiendo dudas haciendo que la creación de un proyecto sea complejo y diferente en cada caso.

La finalidad principal con la que Jason Van Zyl desarrolló Maven para la empresa Sonatype (año 2002), fue para **crear una manera de gestionar y construir proyecto que otorgará un conjunto de convenciones y otorgar simplicidad a la creación y gestión de proyectos.**

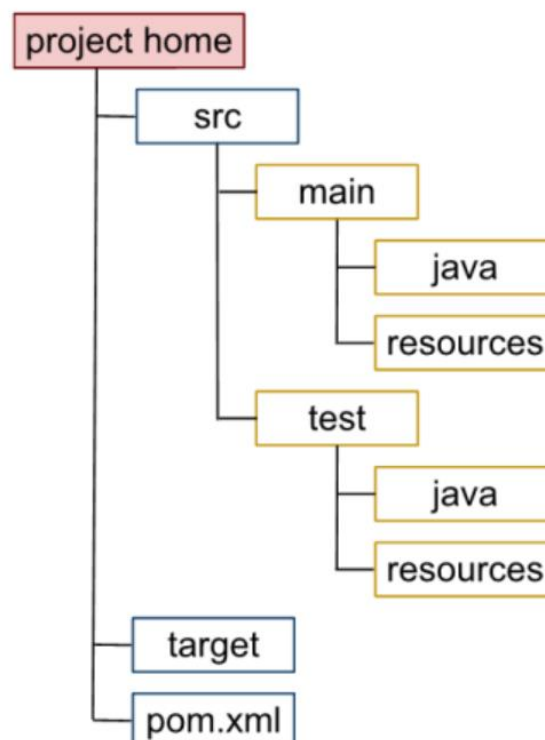
Con Maven se puede:

- **Gestionar las dependencias** del proyecto, para descargar e instalar módulos, paquetes y herramientas que sean necesarios para el mismo.
- **Compilar el código fuente** de la aplicación de manera automática.
- **Empaquetar el código** en archivos .jar o .zip.
- **Instalar los paquetes** en un repositorio (local, remoto o en el central de la empresa)
- **Generar documentación** a partir del código fuente.
- **Gestionar las distintas fases del ciclo de vida** de las *build*: validación, generación de código fuente, procesamiento, generación de recursos, compilación, ejecución de test ...

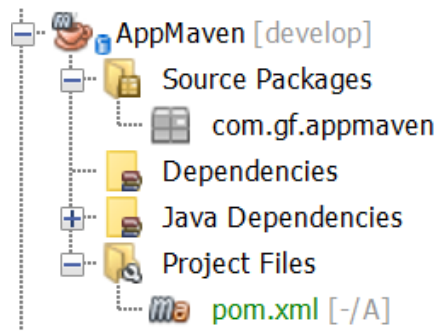
## 1.2 Estructura de un proyecto Maven

Apache Maven es un software que nos permite gestionar proyectos. Al crear un proyecto de Maven, automáticamente se nos generará una estructura de carpetas muy concreta que ya viene predefinida.

Por ejemplo, por defecto, el directorio donde está el código fuente es `src/main/java`, donde se compila el proyecto es `target` y donde ubicamos los test unitarios es en `src/main/test`, etc...



Si creamos un proyecto simple desde NetBeans la estructura que se genera es la siguiente:



### 1.3 Project Object Model (POM)

El corazón del proyecto es el Modelo de Objetos de Proyecto. Se trata de un archivo XML llamado pom.xml que se encuentra por defecto en la raíz de los proyectos y que **contiene toda la información del proyecto**: estructura mínima, configuración, dependencias, etc.

El fichero pom con la estructura más básica posible, contendrá:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/200
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.gf</groupId>
  <artifactId>AppMaven</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>16</maven.compiler.source>
    <maven.compiler.target>16</maven.compiler.target>
  </properties>
</project>
```

En cuanto al ejemplo anterior, su nombre de artefacto completo es "com.gf:AppMaven"

- **Project: root.**
- **ModelVersion:** se debe establecer en 4.0.0
- **GroupId:** el id del grupo del proyecto (comienza por org o com normalmente seguido del nombre de la entidad).
- **ArtifactId:** el id del artefacto (proyecto).
- **Version:** la versión del artefacto en el grupo especificado.
- **Empaquetado:** jar o war
- **Propiedades**

Dentro de un proyecto pueden existir **varios archivos pom.xml** en distintas subcarpetas. Cuando una subcarpetas tiene su propio POM, este **hereda los valores de las carpetas superiores**, sobrescribiéndolos en caso de estar definidos de nuevo dentro de él.

El archivo **pom.xml** que tenemos en la raíz está actuando ya de esta manera: **hereda todos los valores del Súper POM** que son los que tenemos por defecto y, si en nuestro pom.xml propio en el proyecto establecemos otros diferentes, estos prevalecerán sobre los que hay en el Súper POM.

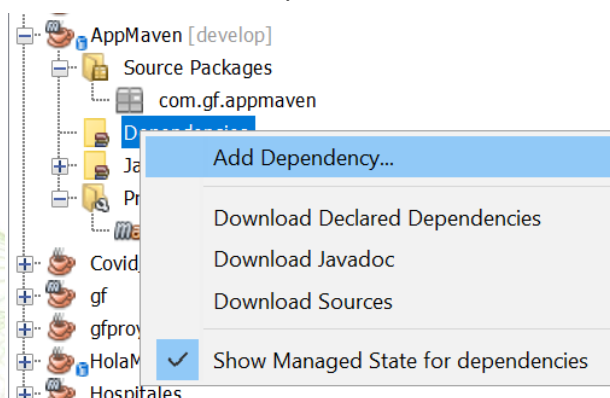
Este tipo de funcionamiento es típico en muchas herramientas de configuración y nos ofrece una manera muy potente de definir valores particulares para casos concretos teniendo una configuración global por defecto.

## Dependencias en Maven

Las **dependencias**, son componentes que nuestro software necesitará para su correcta ejecución durante algún ciclo de vida. Si descargamos un proyecto y lo implementamos en nuestro sistema. Maven, obtendrá las dependencias del proyecto que son necesarias o bien desde el repositorio local, o bien desde un repositorio remoto. Realmente estas dependencias Maven, no son nada más que **archivos con extensión .jar**

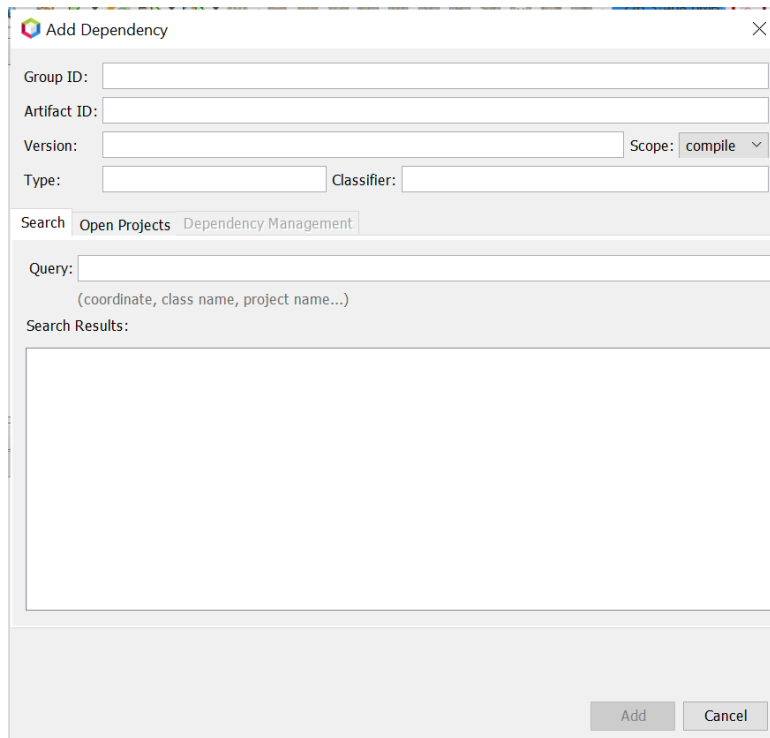
Para declarar dependencias a la hora de crear un proyecto, tenemos dos opciones:

- Declararlas cuando creamos el proyecto, no tendremos que añadirlas manualmente, si no que se añadirán automáticamente.





# CENTRO DE ENSEÑANZA CONCERTADA "Gregorio Fernández"



**Add Dependency**

Group ID:

Artifact ID:

Version:  Scope:

Type:  Classifier:


Search

Query:

(coordinate, class name, project name...)

Search Results:

- Declararlas manualmente visitando el maven central repository ([link a la maven central repository](#)).



**Indexed Artifacts (14.6M)**

**Popular Categories**

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities

**Spring Boot Web Starter**

Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container

**License**

**Tags**

**Used By** 3,715 artifacts

Central (81) Spring Releases (1) Spring Plugins (38) Spring Lib M (1)

Spring Milestones (2) Redhat GA (1) JBoss Public (2) ICM (2) Alfresco (1)

SpringFramework (1) Mulesoft (2)

Version	Repository	Usages	Date
2.1.6.RELEASE	Central	10	Jun, 2019
2.1.5.RELEASE	Central	92	May, 2019
2.1.4.RELEASE	Central	244	Apr, 2019
2.1.x 2.1.3.RELEASE	Central	582	Feb, 2019
2.1.2.RELEASE	Central	481	Jan, 2019
2.1.1.RELEASE	Central	510	Nov, 2018

En el caso que las añadamos de una manera o de otra las dependencias, el resultado final será el mismo, un fichero pom.xml. Que deberá contener un elemento root llamado:

```
<dependencies></dependencies>
```

```
<dependency></dependency>
```

Si por ejemplo quisiéramos añadir una dependencia y ya tenemos el proyecto creado (o las queremos añadir manualmente), vamos al mvnrepository, copiamos la dependencia y la añadimos dentro de nuestro elemento root dependencies.

The screenshot shows the mvnrepository.com search results for 'opencsv'. The search bar at the top contains 'opencsv'. The results are sorted by 'relevance' and show 28 results. The first result is 'OpenCSV' by 'com.opencsv', with 440 usages and a last release on Feb 21, 2022. The second result is 'OpenCSV' by 'net.sf.opencsv', with 655 usages and a last release on Jul 28, 2011. The third result is 'OpenCSV' by 'au.com.bytecode', with 49 usages. The left sidebar shows a list of repositories and groups. The right sidebar shows a list of indexed repositories.

Antes de llegar a poder copiar la dependencia, tenemos que seleccionar la versión de que queremos bajar. En esta página, además aparecen las usages (veces que se ha utilizado dicha versión).

The screenshot shows the mvnrepository.com artifact page for 'com.opencsv:opencsv:5.6'. The page displays the license (Apache 2.0), categories (CSV Libraries), homepage (http://opencsv.sf.net), date (Feb 21, 2022), files (pom (31 KB), jar (230 KB)), repositories (Central), and used by (440 artifacts). The left sidebar shows a list of popular categories. The bottom section shows a code snippet for the dependency in a Maven pom.xml file.

```
https://mvnrepository.com/artifact/com.opencsv/opencsv ->
<dependency>
  <groupId>com.opencsv</groupId>
  <artifactId>opencsv</artifactId>
  <version>5.6</version>
</dependency>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.gf</groupId>
  <artifactId>AppMaven</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>16</maven.compiler.source>
    <maven.compiler.target>16</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>com.opencsv</groupId>
      <artifactId>opencsv</artifactId>
      <version>5.6</version>
    </dependency>
  </dependencies>
</project>
```

### ***Elementos de una dependencia Maven***

Las dependencias en Maven suelen tener:

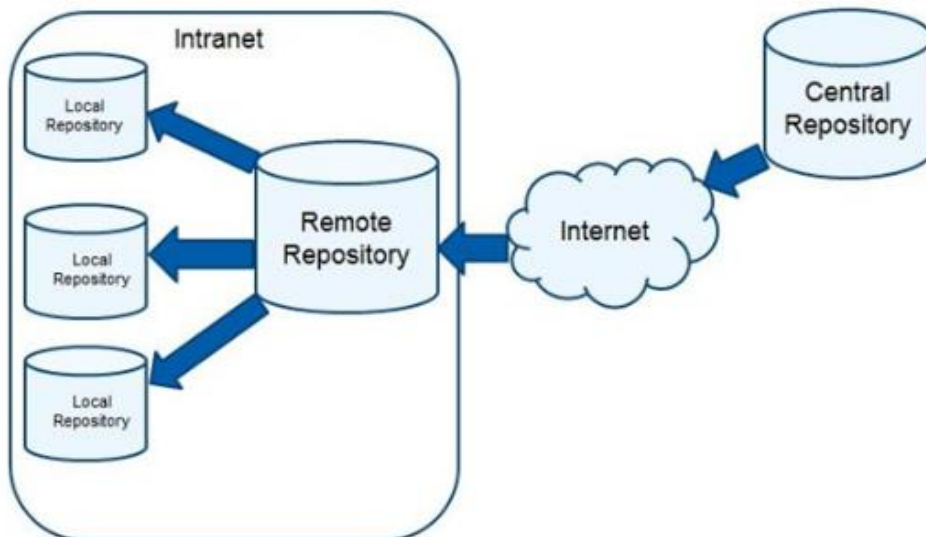
- **dependency:** elemento contenedor padre que especifica que es una dependencia.
- **groupId:** puede no ser obligatorio dependiendo del caso aunque si se pone no pasa nada.
- **artifactId:** Es obligatorio definir el artifactId de una dependencia.
- **version:** Si no lo especificamos Maven cogerá la última versión o la que especifiquemos en el pom padre (superpom.xml). Si lo especificamos no está demás pero si no se especifica no pasa nada, siempre que esta dependencia no entre en conflicto con otras.

### ***Proceso de búsqueda de dependencias de Maven***

A partir de la versión 2 de Maven, las dependencias no van acompañadas del código fuente de nuestro desarrollo sino en una carpeta local llamada **m2**. A la hora de buscar las dependencias de un proyecto, Maven, va al pom y comprueba primeramente si

existen en el repositorio **m2 (repositorio remoto)**. En caso de que allí no existan, irá al **maven central repository (repositorio central)** y las descargará en nuestro repositorio remoto (m2) para que se pueda utilizar en todos los repositorios locales (proyectos).

Para la próxima compilación, ya dispondrá de dicha dependencies dentro del repositorio remoto. Y por tanto, si cargamos un proyecto, este mirará en su repositorio local, no encontrará las dependencies e irá al repositorio remoto donde sí que las encontrará. De modo que, no será necesario descargarla otra vez del Maven Central Repository. Ya que esa dependencia la podemos satisfacer sin necesidad de contarnos a internet.



El repositorio m2 (remote repositorio) por defecto está ubicado por defecto en directorio: `${user.home}/.m2/repository`, aunque dependiendo del sistema operativo la ruta cambia.

- Windows – `C:\Users\{your-username}\.m2\repository`





CENTRO DE ENSEÑANZA CONCERTADA  
"Gregorio Fernández"

repository

Archivo Inicio Compartir Vista

Andar al Acceso rápido Copiar Pegar Cortar Copiar ruta de acceso Pegar acceso directo Mover a Copiar a Eliminar Cambiar nombre Nueva carpeta Nuevo elemento Fácil acceso

Propiedades Historial

Seleccionar todo No seleccionar nada Invertir selección

Este equipo > OS (C:) > Usuarios > rhtuf > .m2 > repository >

Nombre	Fecha de modificación	Tipo	Tamaño
antlr	29/01/2021 21:38	Carpeta de archivos	
aopalliance	29/01/2021 20:44	Carpeta de archivos	
asm	20/04/2021 11:45	Carpeta de archivos	
avalon-framework	15/06/2021 10:28	Carpeta de archivos	
backport-util-concurrent	29/01/2021 20:10	Carpeta de archivos	
br	02/10/2021 12:14	Carpeta de archivos	
ch	29/01/2021 20:44	Carpeta de archivos	
classworlds	29/01/2021 20:10	Carpeta de archivos	
com	16/02/2022 9:32	Carpeta de archivos	
commons-beanutils	21/04/2021 10:08	Carpeta de archivos	
commons-chain	15/06/2021 10:28	Carpeta de archivos	
commons-cli	29/01/2021 20:10	Carpeta de archivos	
commons-codec	20/04/2021 11:45	Carpeta de archivos	
commons-collections	01/02/2021 14:32	Carpeta de archivos	
commons-dbc	01/02/2021 14:32	Carpeta de archivos	
commons-digester	15/06/2021 10:28	Carpeta de archivos	
commons-io	29/01/2021 20:44	Carpeta de archivos	
commons-lang	29/01/2021 20:10	Carpeta de archivos	
commons-logging	01/02/2021 14:32	Carpeta de archivos	
commons-pool	01/02/2021 14:32	Carpeta de archivos	
dom4j	01/02/2021 14:32	Carpeta de archivos	

42 elementos