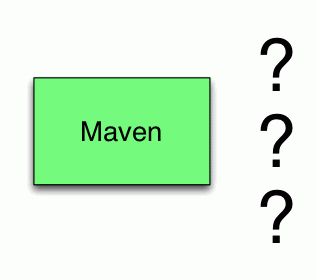
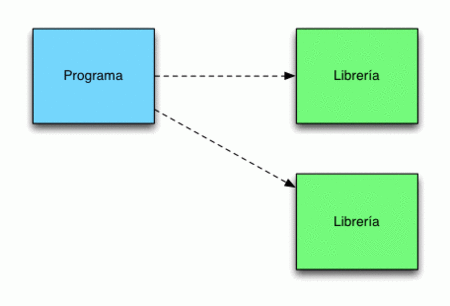
**Maven ???**

En muchas ocasiones me he encontrado con que es difícil hacer llegar a los desarrolladores la necesidad de usar una herramienta **como Maven.** Incluso a veces es difícil de explicar para que sirve exactamente. Vamos a ver una introducción.

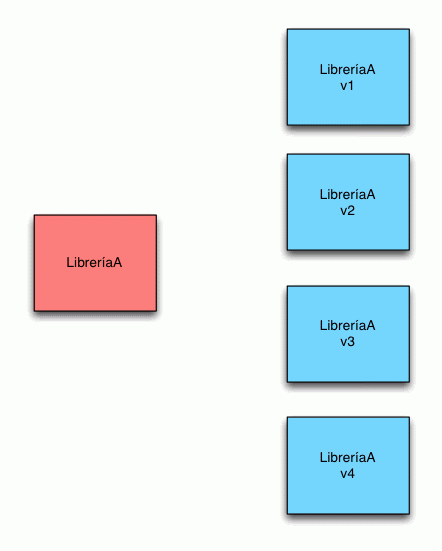


Normalmente cuando nosotros trabajamos con Java/JavaEE el uso de librerías es algo común como en cualquier otro lenguaje de programación.

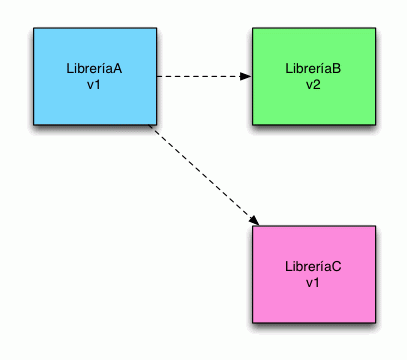


**Librerías y limitaciones**

El concepto de librería **es un concepto que a veces es limitado.** Por ejemplo nosotros podemos querer utilizar la librería A en nuestro proyecto. Sin embargo no nos valdrá con simplemente querer utilizar la librería sino que ademas necesitaremos saber que versión exacta de ella necesitamos.

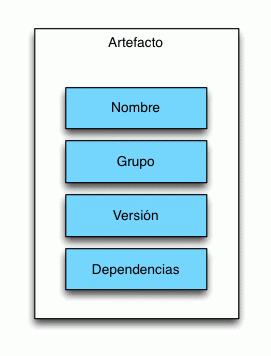


**¿Es esto suficiente?**. Lamentablemente no lo es, una librería puede depender de otras librerías para funcionar de forma correcta. Así pues necesitamos más información para gestionarlo todo de forma correcta.



**Maven y Artefactos**

Maven solventa esta problema a traves **del concepto de Artefacto. Un Artefacto puede verse como una librería con esteroides** (aunque agrupa mas conceptos). Contiene las clases propias de la librería pero ademas **incluye toda la información necesaria para su correcta gestión (grupo, versión, dependencias etc).**



**Artefactos y POM**

Para definir un **Artefacto** necesitamos crear un **fichero POM.xml (Proyect Object Model)** que es el encargado de almacenar toda la información que hemos comentado anteriormente:

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.genbetadev.proyecto1</groupId>

<artifactId>proyecto1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<dependencies>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

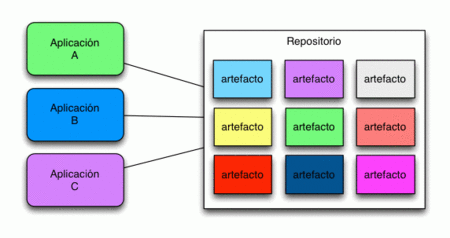
</dependencies>

</project>

La estructura del fichero puede llegar a ser muy compleja y puede llegar a depender de otros POM. En este ejemplo **estamos viendo el fichero más sencillo posible.** En el se define el nombre del Artefacto (artifactID) el tipo de empaquetado (jar) y también las dependencias que tiene (log4j). **De esta manera nuestra librería queda definida de una forma mucho más clara.**

**Maven Repositorio y Artefactos**

Una vez definidos correctamente todos los Artefactos que necesitamos, Maven nos provee de un Repositorio donde alojar, mantener y distribuir estos. Permitiéndonos una gestión correcta de nuestra librerías, proyectos y dependencias.



El uso de Maven es a día de hoy una necesitad en cualquier proyecto Java/Java EE de cierta entidad.

# Maven

|  |  |
| --- | --- |
| **Maven** | |
| [Maven logo.svg](https://commons.wikimedia.org/wiki/File:Maven_logo.svg) | |
| **Desarrollador(es)** | |
| [Apache Software Foundation](https://es.wikipedia.org/wiki/Apache_Software_Foundation) [*http://maven.apache.org*](http://maven.apache.org/) | |
| **Información general** | |
| [**Autor(es)**](https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software) | Jason van Zyl |
| **Lanzamiento inicial** | [30 de marzo](https://es.wikipedia.org/wiki/30_de_marzo) de [2002](https://es.wikipedia.org/wiki/2002) |
| [**Última versión estable**](https://es.wikipedia.org/wiki/Fases_del_desarrollo_de_software) | 3.3.9 [14 de noviembre](https://es.wikipedia.org/wiki/14_de_noviembre) de [2015](https://es.wikipedia.org/wiki/2015) (1 año, 6 meses y 1 día) |
| [**Género**](https://es.wikipedia.org/wiki/Anexo:G%C3%A9neros_de_software) | [Desarrollo Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) |
| [**Programado en**](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) | [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)) |
| [**Plataforma**](https://es.wikipedia.org/wiki/Plataforma_(inform%C3%A1tica)) | [Máquina virtual Java](https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java) |
| [**Licencia**](https://es.wikipedia.org/wiki/Licencia_de_software) | [Licencia Apache 2.0](https://es.wikipedia.org/wiki/Apache_License) |
| [[editar datos en Wikidata](https://www.wikidata.org/wiki/Q139941)] | |

**Maven** es una herramienta de software para la gestión y construcción de proyectos [Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) creada por Jason van Zyl, de Sonatype, en 2002.

Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Este repositorio y su sucesor reorganizado, el repositorio Maven 2, pugnan por ser el mecanismo *de facto* de distribución de aplicaciones en Java, pero su adopción ha sido muy lenta. Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.

Maven está construido usando una arquitectura basada en plugins que permite que utilice cualquier aplicación controlable a través de la entrada estándar. En teoría, esto podría permitir a cualquiera escribir plugins para su interfaz con herramientas como compiladores, herramientas de pruebas unitarias, etcétera, para cualquier otro lenguaje. En realidad, el soporte y uso de lenguajes distintos de Java es mínimo. Actualmente existe un plugin para .Net Framework y es mantenido,[1](https://es.wikipedia.org/wiki/Maven#cite_note-1) y un plugin nativo para C/C++ fue alguna vez mantenido por Maven 1.[2](https://es.wikipedia.org/wiki/Maven#cite_note-2)

## Teoría[[editar](https://es.wikipedia.org/w/index.php?title=Maven&action=edit&section=1" \o "Editar sección: Teoría)]

### Convención sobre configuración[[editar](https://es.wikipedia.org/w/index.php?title=Maven&action=edit&section=2" \o "Editar sección: Convención sobre configuración)]

La filosofía general de Maven es la estandarización de las construcciones generadas por seguir el principio de [Convención sobre Configuración](https://es.wikipedia.org/wiki/Convenci%C3%B3n_sobre_Configuraci%C3%B3n), a fin de utilizar existentes modelos en la producción de software. Esta estrategia necesariamente restringe ampliamente la variabilidad, lo que se refleja en la exhortación de Maven a adherirse a su modelo de proyecto.[3](https://es.wikipedia.org/wiki/Maven#cite_note-3) Mientras que Maven es adecuado para nuevos proyectos, los proyectos complejos ya existentes pueden ser simplemente no adaptables para que utilicen Maven. La falta de restricciones de la convención de capas del proyecto fue hecha de alguna manera más configurable con el lanzamiento de Maven 2.

### Reutilización[[editar](https://es.wikipedia.org/w/index.php?title=Maven&action=edit&section=3" \o "Editar sección: Reutilización)]

Maven está construido alrededor de la idea de reutilización, y más específicamente, a la reutilización de la lógica de construcción. Como los proyectos generalmente se construyen en patrones similares, una elección lógica podría ser reutilizar los procesos de construcción. La principal idea es no reutilizar el código o funcionalidad (como Apache Ant), sino simplemente cambiar la configuración o también código escrito. Esa es la principal diferencia entre Apache Ant y Apache Maven: el primero es una librería de utilidades y funciones buenas y útiles, mientras que la otra es un framework configurable y altamente extensible.[4](https://es.wikipedia.org/wiki/Maven#cite_note-4)

Aunque Maven es configurable, históricamente el proyecto Maven ha enfatizado seriamente que los usuarios deben adherirse a su concepto de un modelo de proyecto estándar tanto como sea posible.

### Ciclo de vida[[editar](https://es.wikipedia.org/w/index.php?title=Maven&action=edit&section=4)]

Las partes del ciclo de vida principal del proyecto Maven son:

1. *compile*: Genera los ficheros .class compilando los fuentes .java
2. *test*: Ejecuta los test automáticos de JUnit existentes, abortando el proceso si alguno de ellos falla.
3. *package*: Genera el fichero .jar con los .class compilados
4. *install*: Copia el fichero .jar a un directorio de nuestro ordenador donde maven deja todos los .jar. De esta forma esos .jar pueden utilizarse en otros proyectos maven en el mismo ordenador.
5. *deploy*: Copia el fichero .jar a un servidor remoto, poniéndolo disponible para cualquier proyecto maven con acceso a ese servidor remoto.

Cuando se ejecuta cualquiera de los comandos maven, por ejemplo, si ejecutamos **mvn install**, maven irá verificando todas las fases del ciclo de vida desde la primera hasta la del comando, ejecutando solo aquellas que no se hayan ejecutado previamente.

También existen algunas metas que están fuera del ciclo de vida que pueden ser llamadas, pero Maven asume que estas metas no son parte del ciclo de vida por defecto (no tienen que ser siempre realizadas). Estas metas son:

1. *clean*: Elimina todos los .class y .jar generados. Después de este comando se puede comenzar un compilado desde cero.
2. *assembly*: Genera un fichero .zip con todo lo necesario para instalar nuestro programa java. Se debe configurar previamente en un fichero xml qué se debe incluir en ese zip.
3. *site*: Genera un sitio web con la información de nuestro proyecto. Dicha información debe escribirse en el fichero pom.xml y ficheros .apt separados.
4. *site-deploy*: Sube el sitio web al servidor que hayamos configurado.
5. etc.

Pero estas metas pueden ser añadidas al ciclo de vida a través del Project Object Model (POM).