# POM.xml

El archivo pom.xml (Project Object Model) es el corazón de cualquier proyecto Maven. Es un archivo XML que contiene toda la información necesaria para construir un proyecto, incluyendo detalles de configuración, dependencias, plugins, y metadatos del proyecto. Maven utiliza este archivo para determinar cómo construir y empaquetar el proyecto.

**Componentes Principales del pom.xml**

<project>: Es el elemento raíz del archivo pom.xml.

<modelVersion>: Especifica la versión del modelo POM. Debe ser 4.0.0 para proyectos modernos.

<groupId>: Identifica de manera única al grupo de proyectos al que pertenece el proyecto. Usualmente sigue el formato inverso del dominio de la organización.

<artifactId>: Identifica de manera única al proyecto dentro de su grupo. Se utiliza para nombrar el archivo JAR o WAR que se genera.

<version>: Especifica la versión del proyecto.

<dependencies>: Lista todas las dependencias externas que necesita el proyecto.

<plugins>: Define los plugins de Maven que se utilizarán durante la fase de construcción.

<build>: Contiene configuraciones específicas de la fase de construcción, como directorios de salida y configuraciones de plugins.

**Estructura Básica de un pom.xml**

***Aquí tienes un ejemplo básico de cómo se ve un archivo pom.xml:***

<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.ejemplo</groupId>

<artifactId>mi-proyecto</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Ejemplo de dependencia -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Ejemplo de plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

Este archivo define un proyecto Maven con un groupId com.ejemplo, un artifactId mi-proyecto, y una versión 1.0-SNAPSHOT. También incluye una dependencia a JUnit para pruebas y configura el plugin maven-compiler-plugin para compilar el código.

# Agregar dependencias en Apache Maven

La gestión de **dependencias de un proyecto** es una característica central de Maven que se utiliza para definir bibliotecas de las que depende un proyecto. Las dependencias se pueden especificar en el archivo pom.xml y Maven las descargará automáticamente a su **repositorio Maven** local.

Apache Maven es una herramienta de gestión de proyectos que utiliza el **Project Object Model** (POM) para gestionar la creación, los reportes y la documentación del proyecto.

Maven también tiene una función de ámbito de dependencia que permite definir el ámbito o alcance de las dependencias y reducir la cantidad de dependencias transitivas. Además, Maven permite definir dependencias externas, usar dependencias específicas del perfil y administrar dependencias en un proyecto multi módulo. En este artículo veremos cómo agregar dependencias en Apache Maven en unos pocos pasos.

Configurar dependencias en Apache Maven

El archivo pom.xml es un documento que contiene información sobre el proyecto y su configuración. Por lo general, se encuentra en la raíz de un proyecto de Maven y contiene el nombre del proyecto, la versión, las dependencias Maven, la configuración de compilación y las variables de entorno. De su estructura podemos destacar las siguientes secciones:

* La sección **<build>** se usa para definir la configuración de compilación del proyecto, como la versión del compilador, el directorio de origen, el directorio de salida y los plugins que se usarán durante el proceso de compilación. Puede contener una sección <resources> para administrar recursos como imágenes y archivos de texto, y una sección <profiles> para adaptar el proceso de construcción a diferentes entornos.
* La sección **<dependencies>** se usa para definir las dependencias de su proyecto. Estas dependencias pueden ser bibliotecas u otros proyectos de los que depende su proyecto. Cada dependencia se especifica con un elemento <dependency> que incluye groupId, artifactId y la versión de la dependencia. Esta sección también le permite especificar el scope o alcance que determina cuándo se debe incluir la dependencia en el proyecto. Los scopes disponibles incluyen:
  + **compile**: este es el ámbito predeterminado, que se utiliza si no se especifica ninguno. Las dependencias de compilación están disponibles en todos los classpaths.
  + **provided**: esto es muy parecido a compilar, pero indica que espera que el JDK o un contenedor lo proporcionen. Solo está disponible en el classpath de compilación, no es transitivo ni está empaquetado.
  + **runtime**: este scope indica que la dependencia no es necesaria para la compilación, pero sí para la ejecución. Estará disponible en tiempo de ejecución y en test, pero no estará disponible para compilación.
  + **test**: este scope indica que la dependencia no es necesaria para un uso normal de la aplicación y solo está disponible para las fases de compilación y ejecución de tests.
  + **system**: este scope es similar a provided, excepto que se debe proporcionar el JAR que lo contiene explícitamente. El artefacto siempre está disponible y no se busca en un repositorio.
  + **import**

1. Buscar la librería en el Repositorio Central

La mejor manera de buscar en el **Repositorio Central de Maven** es utilizar el sitio web del repositorio central de Maven en <https://search.maven.org/>. Este sitio web proporciona una interfaz de búsqueda fácil de usar, que le permite encontrar rápidamente los artefactos que está buscando. Además, el sitio web ofrece opciones de búsqueda avanzada, como buscar por nombre de clase, buscar por coordenadas, etc.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

2. Seleccionar la versión

En el listado resultante seleccionamos la versión que se ajuste a nuestras necesidades.

Tabla

Descripción generada automáticamente

3. Copiar la dependencia

La siguiente pantalla muestra los detalles de la dependencia seleccionada, en donde encontraremos opciones para descargar el jar y de copiar los datos de la dependencia en formato XML para agregar al POM de nuestro proyecto.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

4. Agregar la dependencia al POM

Una vez que tenemos el XML con las coordenadas de la dependencia la podemos agregar al archivo POM de nuestro proyecto.

La dependencia se debe agregar dentro del elemento **<dependencies>**.

Texto

Descripción generada automáticamente