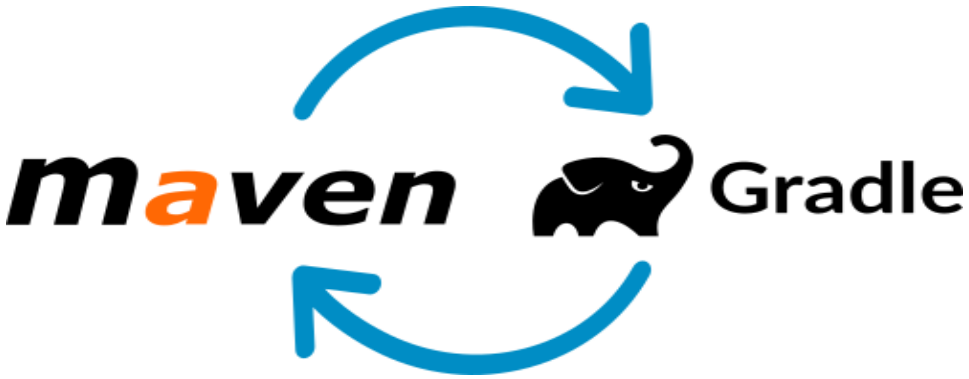


Maven es una herramienta de gestión de proyectos de desarrollo utilizada principalmente en Java utilizando conceptos provenientes de Apache Ant.
Gradle es una herramienta de automatización de compilación del código abierto, que obtuvo una rápida popularidad ya que fué diseñada fundamentalmente para construir multiproyectos, utilizando conceptos provenientes de Maven.

Ant, el cual era muy flexible y con el que lo podías customizar todo. El problema es que era muy verboso y no integraba bien las dependencias. De ahí nació Maven, el cual, seguía el patrón xml aunque menos verboso. Este, integra la gestión de dependencias y distribución de proyectos. Lo malo, es que no es nada flexible.
De ahí nace Gradle, que combina la flexibilidad de Ant, con las convenciones de Maven.

Fué desarrollado por Apache, y su objetivo principal es simplificar el proceso de construcción de proyectos y la gestión de dependencias en el desarrollo de software. Solo funciona para JAVA.
Maven promueve la idea de convención sobre configuración, lo que facilita la comprensión del proyecto.
La segunda ventaja de Maven es la gestión automática de dependencias, no es necesario descargar de uno en uno cada archivo JAR manualmente que cada proyecto necesita.

Explicación de la estructura de directorios de un proyecto Maven (src, target, pom.xml).
SRC -> Contenido para código fuente y recursos
TARGET -> Donde se almacenan los resultados de la construcción.

Maven™
Gradle

POM.XML -> La configuración del proyecto. (Project Object Model)

Ventajas de Maven:
Gestión de dependencias automatizada.
Facilita la comprensión y colaboración en proyectos.
Gestión de dependencias: Maven resuelve automáticamente las dependencias del proyecto. Utiliza un repositorio central donde se almacenan las dependencias de los proyectos. Esto facilita la gestión de dependencias y garantiza la consistencia en los proyectos.
Maven tiene una estructura de proyecto predefinida y poco modificable. En la mayoría de proyectos, esto no suele ser un problema, pero si necesitas más flexibilidad para tu estructura, es mejor Gradle.

Convención sobre configuración:
Utiliza una estructura de proyecto predefinida. Lo que significa que los proyectos Maven tienen una organización uniforme.

Ejemplo: Código en src/main/java y recursos en src/main/resources.

Ciclo de vida del proyecto: Maven define fases para construir, probar, empaquetar y distribuir proyectos.
Sus fases son, clean, validate, compile, test, package, verify, install y deploy.

Clean:Borra todos los .class y .jar generados.
Compile: Compila el código fuente del proyecto.
Test: Ejecuta las pruebas unitarias.
Package: Empaqueta el proyecto en un archivo JAR o WAR.
Install: LLeva el jar a nuestro repositorio local de jars. Queda "visible" para otros proyectos maven en nuestro ordenador.
Deploy: Lleva el jar a nuestro servidor de jars. Queda "visible" para otros proyectos maven en otros ordenadores. Este comando necesita que a maven se le haya indicado dónde está dicho servidor de jars.

(JAR (Java ARchive))
Es un formato de archivo que se utiliza para distribuir aplicaciones Java, bibliotecas y componentes.
Contienen archivos comprimidos, incluyendo archivos de clase, archivos de metadatos y recursos como imágenes y archivos de configuración.

Su contenido puede tener código fuente compilado en forma de archivos de clase, archivos de recursos como imágenes y archivos de configuración) y archivos de metadatos como el **MANIFEST.MF**.(archivo específico contenido en un archivo Jar. Se usa para definir datos relativos a la extensión y al paquete).
Utilizan formato ZIP para reducir el tamaño del archivo y permitir una distribución más eficiente. Pueden ser ejecutados directamente por la máquina virtual de Java, (Se logra especificando la clase principal del archivo en MANIFEST.MF.
Lo que permite que la aplicación se inicie simplemente ejecutando el archivo JAR.

Maven™
Gradle

Los archivos **JAR** son ampliamente utilizados para distribuir bibliotecas y componentes reutilizables en el ecosistema Java. Las dependencias de un proyecto Java se especifican en el POM.XML de Maven o el build.gradle de Gradle obviamente. Maven y Gradle se encargan de descargar automáticamente los JAR necesarios para repositorios remotos.
Se suele utilizar para empaquetar y distribuir aplicaciones JAVA completas. Lo que facilita la distribución y ejecución de la aplicación en diferentes plataformas sin preocuparse por las dependencias del sistema operativo.

Es multi-lenguaje. JAVA,C + +,Python y más.
Tiene una sintaxis más legible y expresiva, lo que permite una configuración más flexible y concisa en comparación con Maven.
La pregunta más común al iniciar un proyecto con Gradle, Groovy o Kotlin?

Groovy era el lenguaje de configuración predeterminado en Gradle y ha sido utilizado durante mucho tiempo.
La sintaxis de Groovy puede ser más familiar para los que tienen conocimientos previos en Java. Tiene un amplio soporte y una gran cantidad de recursos y bibliotecas disponibles.
.
Por otro lado, Kotlin es un lenguaje más moderno con funciones de extensión y sintaxis más concisa. Se está volviendo más popular de Groovy. Tiene una sintaxis más limpia y menos verbosa en comparación con Groovy, lo que
puede hacer que los scripts del build.gradle sean más legibles y fáciles de mantener.
En sí no hay muchas más diferencias, dependerá de las preferencias del equipo y las características del proyecto.

Ventajas de Gradle:
Flexibilidad: Utiliza un DSL (Lenguaje Específico de Dominio) basado en Groovy para definir tareas de construcción.
Esto hace que Gradle sea más flexible y personalizable que Maven.
Rendimiento: Utiliza un modelo de construcción incremental para acelerar la construcción de proyectos.
Soporte multi-proyecto: Facilita la construcción de proyectos grandes y complejos.

Maven™
Gradle

Comandos más utilizados:
gradle build: Construye el proyecto.
gradle clean: Limpia el directorio de construcción.
gradle tasks: Muestra una lista de tareas disponibles en el proyecto.
gradle dependencies: Muestra las dependencias del proyecto.
Se pueden encadenar tantas tareas como quieras.
-> gradle clean compileJava

Diferencias entre Maven y Gradle:

En **Maven**, siempre tienes que escribir una serie de etiquetas
para realizar ciertas tareas,con **Gradle** permite lo mismo, con menos código.

Gradle.

```
apply plugin: 'java'

description = "Gradle Java Project"

repositories{
    mavenCentral()
    mavenLocal()
}

dependencies{
    compile "junit:junit:4.7"
}

task helloWorld () {
    description = "Simplemente hola"

    println "Hello Gradle"
}

```

Maven

```
<project xmlns="http://maven.apache.org/POM/
  xsi:schemaLocation="http://maven.apache.or
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.madridgug.maven</groupId>
  <artifactId>maven-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>GradleVsMaven</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.7</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>

```

Estructura de configuración: Maven utiliza XML **(pom.xml),** mientras que Gradle utiliza **Groovy** o **Kotlin.**

Flexibilidad: Gradle ofrece más flexibilidad y capacidad de personalización que Maven.
Rendimiento: Gradle tiende a ser más rápido en la construcción de proyectos debido a su modelo incremental.

Maven
Gradle

Curva de aprendizaje: Maven es más fácil de empezar debido a su enfoque basado en convenciones, mientras que Gradle puede requerir una curva de aprendizaje más pronunciada debido a su flexibilidad.

Realizar **scripting** en **Maven** puede ser un infierno, además de que la mayoría de las veces acabas embebiendo código en el propio .xml
(**Código ejecutable dentro del pom.xml)**
Requiere la configuración de múltiples plugins con una sintaxis específica.

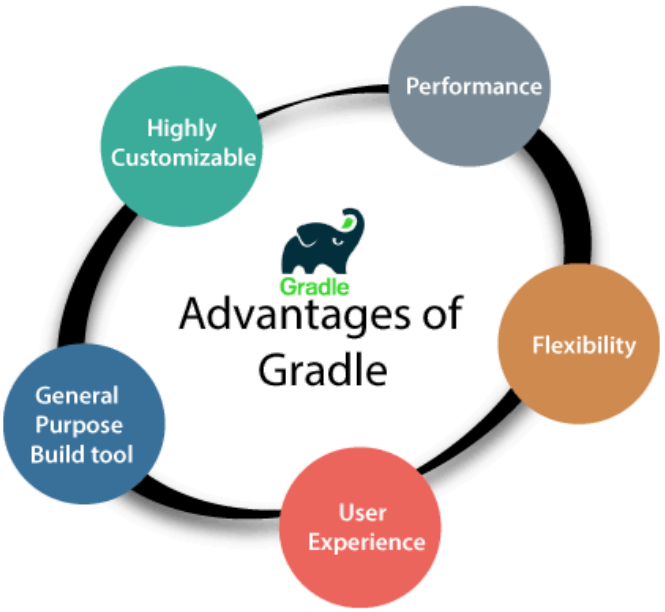
En cambio en **Gradle**, el propio script es código con lo que puedes importar plugins o clases de utilidad que ya tuvieras para realizar las tareas y además categorizarlas.

Actualmente se le da más prioridad a Gradle sobre Maven, debido a que sus archivos de compilación son más limpios y concisos. Es mucho más fácil encontrar dependencias y tener una idea de todo el proyecto.
Aparte se ejecuta en muchos más lenguajes y para proyectos multilenguaje es necesario.

La clave está en la flexibilidad, si el proyecto se puede gestionar con metas estándar o con poca personalización entonces Maven es suficiente, por otro lado si el proyecto requiere diferentes comportamientos en función de múltiples variables en tiempo de construcción quizá Gradle responda más eficazmente a estos requerimientos.

Mejoras en Gradle en próximas versiones.

- Gradle daemon, que mejora el rendimiento.
- Información sobre tareas
- Consola interactiva y autocompletado.



Maven
Gradle